Robotic Manipulation for Parts Transfer and Orienting:

Mechanics, Planning, and Shape Uncertainty

Srinivas Akella

December 1996

CMU-RI-TR-96-38

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This research was funded in part by the Robotics Institute and by NSF under grants IRI-9114208, IRI-9213993, and IRI-9318496 (with ARPA). The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Robotics Institute, NSF, ARPA, or the U. S. Government.

Abstract

Robots can modify their environment by manipulating objects. To fully exploit this ability, it is important to determine the manipulation capabilities of a given robot. Such characterization in terms of the physics and geometry of the task has important implications for manufacturing applications, where simpler hardware leads to cheaper and more reliable systems. This thesis develops techniques for robots to *transfer* parts from a known position and orientation to a goal position and orientation, and to *orient* parts by bringing them from an unknown initial orientation to a goal orientation. This *parts feeding* process is an important aspect of flexible assembly. Designing automatic planners that capture the task mechanics and geometry leads to flexible parts transfer and orienting systems. The implemented parts feeding systems use simple effectors that allow manipulation of a broad class of parts, and simple sensors that are robust and inexpensive.

The main research issues are to identify a set of actions for the robot that is *complete* for the task and to develop *automatic planners* that share this completeness property. That is, the actions should enable the robot to successfully execute the task, and the planners should automatically generate such sequences of actions.

To illustrate this approach, the thesis describes a set of parts transfer and orienting tasks, their mechanics, and planning techniques to solve them. The first example is a parts transfer system that automatically identifies a sensorless sequence of pushes for a robot to move any polygonal part to any goal position and orientation in the plane. The second system demonstrates that a one-joint robot can transfer any polygon to a specified goal position and orientation by pushing it on a conveyor. We present automatic planners that use mathematical programming formulations for these tasks. The thesis then describes a one-joint robot system to perform sensorless orienting of parts. The last system, also for parts orienting, demonstrates the speedup resulting from using inexpensive photosensors in combination with actions. The sensors provide partial information on a part's orientation by measuring its width; the actions rotate the part to orientations the sensors can identify. This system can orient multiple part shapes with a single plan. Further, the thesis analyzes the effects of shape uncertainty arising from manufacturing tolerances on parts orienting and identifies conditions under which we can orient parts with shape uncertainty. Planners for these systems have been implemented and experimentally demonstrated on industrial robots.

Acknowledgments

I count myself fortunate to have studied and worked in the Robotics Institute at Carnegie Mellon University. Graduating and leaving CMU helps me appreciate even more the people I have worked with and environment I have been in. I'd like to thank my advisor, Matt Mason, for giving me the freedom, encouragement, and opportunity to develop as a researcher. His originality, enthusiasm for research, and sense of fun have made working with him a unique and rewarding experience. I have benefited greatly from interactions with my thesis committee members. Mike Erdmann provided code, equipment, financial support, math expertise, and more. Reid Simmons' suggestions and probing questions made me see things I would have missed otherwise. Ken Goldberg provided me with research leads and advice, and was always enthusiastic about my work.

I have learned much and have had fun working with members of the Manipulation Lab. I benefited greatly from the example set by Randy Brost, Alan Christiansen, Ken Goldberg, and Yu Wang. Randy, Alan, and Ken continue to be a source of advice, encouragement, and insights. Tammy Abell, Wes Huang, Yan-bin Jia, Kevin Lynch, Mark Moll, Garth Zeglin, and Nina Zumel provided me with constructive suggestions over the years. Garth and Yan-bin also provided useful comments on the thesis document. Kevin Lynch's Lisp interface to Idraw made drawing figures a lot easier. Many thanks to Costa Nikou and Evan Shechter for implementations of the 1JOC and sensorless 1JOC on the Adept robot, and for being fun to work with.

For advice and helpful discussions at various points during graduate school, I thank Marshall Bern, Ben Brown, Alan Guisewite, Martial Hebert, Ralph Hollis, Katsushi Ikeuchi, Eric Krotkov, N.R. Natraj, Mel Siegel, and Tony Stentz.

I have been fortunate to have had great officemates who were also good friends. Fred Solomon was a source of engineering expertise, and in his Prophet of Doom incarnation, much humor. Dafna Talmor was the mathematics and algorithms whiz, and provided helpful comments on my thesis. Scott Crowder and Marco Zagha were great officemates and sources of advice. Doug Beeferman and Neil Heffernan, the new kids on the block, were the source of much fun and inspired me to graduate.

My friends have been a source of fun, affection, and food. Satish, Lokesh and Bindiya, Nagesh and Ramadevi, Guru and Hema, Toto, Vivek, Gary, Manoj, JC, Sanjiv and Sonia, Jawa, Madhavi, Thierry, and the Mealplan crowd kept me sane. The Robograds, and in particular, fellow Viking Death Rats introduced me to American sports and made grad school a lot more fun.

The CMU Robotics Institute and School of Computer Science environment is a great place to work. I thank Raj Reddy and Takeo Kanade for making this possible. Jean Harpley was a cheerful source of help and found solutions when everyone else was stumped. Marce Zaragoza made all the academic hurdles easier to jump over.

I thank my mother and father, and my sister and brothers and their families for their love and support, and for checking up on me regularly. They were always there for me, and understood the Ph.D. process far better than I could have asked for. Thank you.

Contents

1 Introduction **1**
1.1 Thesis Outline . . . 3
1.2 Related work . . . 5
1.2.1 Uncertainty, Mechanics, and Manipulation Planning . . . 5
1.2.2 Pushing . . . 8
1.2.3 Nonprehensile Manipulation . . . 10
1.2.4 Parallel-jaw grasping . . . 11
1.2.5 Parts Feeding . . . 12
1.2.6 Minimalism . . . 13
1.2.7 Sensor-based manipulation . . . 13
1.2.8 Probabilistic manipulation operations . . . 14
1.2.9 Motion planning with uncertainty . . . 14
1.2.10 Pose determination and object localization . . . 15
1.2.11 Artificial Intelligence Planning . . . 16
1.2.12 Shape Uncertainty and Tolerancing . . . 16
1.2.13 Completeness . . . 17
1.3 Thesis Contributions . . . 18
1.4 Summary . . . 18

2 Posing Parts in the Plane by Pushing **19**
2.1 The Planar Pose Problem . . . 21
2.1.1 Outline of the Approach . . . 21
2.2 The Mechanics of Pushing . . . 23
2.2.1 Types of Pushes . . . 23
2.2.2 Linear Normal Pushes . . . 23
2.2.3 Part Classification . . . 24
2.2.4 Stable Edges and the Angle-Eating Heuristic . . . 25
2.3 Determining the Pushes . . . 26
2.3.1 The Pose Vector Equation . . . 29
2.3.2 A Linear Programming Solution . . . 30
2.3.3 Existence of Solutions . . . 31
2.4 The Pose Planner . . . 32
2.4.1 Planner Completeness . . . 33
2.4.2 Planner Complexity . . . 34

2.5 Minimizing Plan Execution Time 36
2.6 Implementation and Experiments 36
2.7 Conclusion 38

3 Parts Transfer on a Conveyor with a One Joint Robot 39
3.0.1 Terminology 41
3.1 A Progression of Models 42
3.2 The Feeding Property 44
3.2.1 The 1JOC Model 44
3.2.2 1JOC Primitives 45
3.2.3 The Feeding Property 50
3.3 A Feeding Planner 50
3.3.1 A Simple Example 51
3.3.2 Nonlinear Programming Approach 52
3.3.3 Generating Feeding Plans 54
3.4 Implementation 55
3.5 Sensitivity to errors in geometry 55
3.6 Variations on a Theme 56

4 Sensorless Parts Orienting with a One-Joint Manipulator 57
4.1 Sensorless 1JOC Model 57
4.1.1 Primitives 58
4.2 Planning 63
4.2.1 Predicting Part Rotation 63
4.2.2 Plan Existence and Generation 65
4.3 Implementation 67
4.4 Conclusion 68

5 Parts Orienting with Partial Sensor Information 69
5.1 Assumptions 70
5.2 Action Model 70
5.2.1 Mechanics 71
5.2.2 Radius Function, Push Function, and Push-diameter Function 72
5.2.3 The Radius Function and Part Shape 74
5.2.4 Resting Ranges 79
5.2.5 Finding Representative Actions 80
5.2.6 Graph Representation of the Orienting Problem 82
5.3 Sensor Model and State Distinguishability 83
5.4 Sensorless Orienting 85
5.4.1 Search-based Planning 85
5.4.2 Backchaining Algorithm 86
5.5 Sensor-based Orienting 87
5.5.1 AND/OR Search 90
5.5.2 Plan Length 91
5.5.3 Completeness 95

5.5.4 Greedy Algorithm . . . 95
5.5.5 Bottom-to-top Algorithm . . . 97
5.6 Complexity of Orienting Problem . . . 98
5.7 Orienting Multiple Part Shapes . . . 99
5.7.1 Properties of Push and Push-diameter Functions . . . 99
5.7.2 Orientability and Recognizability of Multiple Parts . . . 102
5.7.3 Sensor-based Orienting and Recognizing of Multiple Parts . . . 106
5.7.4 Sensorless Orienting of Multiple Parts . . . 106
5.7.5 Implementation . . . 107
5.8 Conclusion . . . 108

6 Parts Orienting with Shape Uncertainty 111
6.1 Shape Uncertainty Model . . . 112
6.1.1 Selecting a Shape Uncertainty Model . . . 113
6.2 Effects of Shape Uncertainty . . . 114
6.2.1 Nondeterminism . . . 115
6.3 Stability Regions . . . 116
6.3.1 Stability Region Computation . . . 117
6.4 Generating Actions . . . 119
6.4.1 Deterministic Action Ranges . . . 121
6.4.2 Nondeterministic Action Ranges . . . 124
6.4.3 Action Ranges for a Set of States . . . 125
6.5 Sensor-based Orienting with Shape Uncertainty . . . 125
6.6 Is the Planner Complete? . . . 126
6.7 Sensorless Orienting with Shape Uncertainty . . . 129
6.8 Implementation . . . 131
6.9 Multiple Part Shapes with Shape Uncertainty . . . 133
6.10 Qualitative Shape Changes . . . 133
6.10.1 Identifying the Multiple Qualitative Shapes . . . 135
6.10.2 Generating Plans for Multiple Qualitative Shapes . . . 135
6.11 Future Work and Open Issues . . . 137
6.12 Conclusion . . . 138

7 Conclusion 139
7.1 Thesis Contributions . . . 140
7.2 Future Work . . . 141

A The Peshkin Distance 143

B LED Sensor to Measure Part Diameter 145

C Estimating the Center of Mass Locus 147

Chapter 1

Introduction

Robots can modify their environment by manipulating objects. To fully exploit this ability, it is important to determine the manipulation capabilities of a given robot. This characterization is especially important since robots are in widespread use in industrial tasks such as welding and painting, but not in tasks involving contact between the robot and its environment. This thesis is motivated by such manipulation tasks, and in particular, *parts transfer and orienting* tasks for flexible assembly. Automated assembly devices and robots need to receive parts in specified orientations at specified positions to assemble them into a product or place them in a pallet. Thus flexible *parts feeding* is a central problem for the development of automated assembly systems. This thesis develops techniques for robots to *transfer* parts from a known position and orientation to a goal position and orientation, and to *orient* parts by bringing them from an unknown initial orientation to a goal orientation.

The results of the manipulation operations used depend on the physics and geometry of the tasks. By exploiting our knowledge of the task mechanics and geometry and designing automatic planners that capture this knowledge, we can use simple and inexpensive hardware elements controlled by the planners to flexibly handle new parts. This reduces changeover costs and the time to market. Such systems can also allow designers to evaluate the feedability of parts during the early stages of design, enabling virtual prototyping. The implemented parts feeding systems described in this thesis use simple effectors that allow manipulation of a broad class of parts, and simple sensors that are robust and inexpensive. Such capabilities are becoming increasingly important in handling rapidly changing products and enabling agile manufacturing.

Characterization of the robot capabilities in terms of the mechanics and geometry of the tasks can lead to simpler hardware, and cheaper and more reliable systems. The main research issues are to identify a set of actions for the robot that is *complete* for the task and to develop *automatic planners* that share this completeness property. That is, the actions should enable the robot to successfully execute the task, and the planners should automatically generate such sequences of actions. In addition to the completeness of the action sets used and completeness of the planners, it is important to precisely characterize properties of the parts feeding systems such as the classes of parts the systems can handle, execution lengths of plans, and the complexity of the plan generation process.

In this thesis, the primary operation to manipulate parts is pushing using simple effectors such as straight-line fences. Pushing is a form of nonprehensile manipulation, where the part is manipulated without being grasped rigidly. As such, pushing is an example manipulation primitive, and the goal is to develop techniques that can then be applied to other manipulation primitives. Simple effectors such as fences allow us to handle classes of parts without the need for part-specific grippers. Using such effectors to manipulate objects by pushing means the traditional pick-and-place model of manipulation cannot be used — we have to develop new techniques to plan manipulation operations for parts transfer and orienting. I also illustrate the advantages of using simple and inexpensive sensors, such as photoelectric sensors, that provide only partial state information. The partial information from the sensors speeds up execution of orienting plans, and permits multiple part shapes to be manipulated using the same plan. The parts feeding systems in this thesis can be viewed as minimalist systems that use simple effectors and sensors. Note that the simpler hardware is often accompanied by an increase in complexity of the analysis and planning software.

To illustrate this approach, I describe a set of parts transfer and orienting tasks, their mechanics, planning techniques to solve the tasks, and the implemented systems. The tasks studied in this thesis are:

1. Parts transfer in the plane using pushing. The first example is a parts transfer system that automatically identifies a sensorless sequence of pushes for a robot to move any polygonal part to any goal position and orientation in the plane. The robot uses a fence to push the part, and needs at least three degrees of freedom to accomplish the task.
2. Parts transfer on a conveyor belt by a one degree of freedom robot. This second system demonstrates that a one-joint robot can transfer any polygon to a specified goal position and orientation by pushing it on a conveyor. This system uses a camera to determine the initial position and orientation of the part.
3. Parts orienting on a conveyor belt by a one degree of freedom robot. This is also a one-joint robot system, and it can perform sensorless orienting of parts.
4. Parts orienting with partial sensor information. This system, also for parts orienting, demonstrates the speedup resulting from using inexpensive photosensors in combination with actions. The sensors provide partial information on a part's orientation by measuring its width; the actions rotate the part to orientations the sensors can identify. This system can orient multiple part shapes with a single plan.
5. Parts orienting in the presence of shape uncertainty. I demonstrate that we can also orient parts with shape uncertainty arising from manufacturing tolerances, both with and without partial sensor information.

Planners for these systems have been implemented and experimentally demonstrated on industrial robots.

The parts transfer and orienting systems described in this thesis have been developed and analyzed for the manipulation of polygonal parts. While there is a large class of industrial parts that are flat or prismatic and that can be treated as polygons, this thesis is a preliminary step towards developing parts transfer and orienting systems for 3-D parts with complex shapes. The next challenge is to identify and develop effective manipulation operations for 3-D parts and to then extend the results of this thesis to the transfer and orienting of 3-D parts. The basic principles and techniques we use and illustrate, such as the use of simple effectors, knowledge of part geometry, use of simple sensors, and the nondeterministic effects of shape uncertainty can then be applied to a broader class of industrial parts and processes. In addition, to implement these techniques industrially, we have to consider all aspects of the parts feeding process. This includes developing techniques to perform singulation of parts, to ensure high system reliability, and to speed up the parts transfer and orienting process, possibly by using operations that incorporate dynamic effects.

1.1 Thesis Outline

This thesis concentrates on parts transfer and orienting for flexible assembly. *Parts transfer* refers to changing the known position and orientation of a part for an assembly or inspection step. *Parts orienting* refers to the process of bringing a part in an unknown initial orientation to a known orientation. Both these processes, commonly referred to as *parts feeding*, are important for flexible assembly and agile manufacturing. We use analyses of the mechanics and geometry of the tasks to develop effective solutions for such tasks.

We are interested in developing techniques and systems for flexible handling of new and changing products with minimal setup and changeover time. In addition to developing solution techniques, we seek to identify the capabilities of our systems in terms of the part classes they can handle, the completeness of the action sets, and completeness and complexity of the planners.

The next two chapters concentrate on parts transfer tasks.

Chapter 2 studies the use of pushing actions to perform planar parts transfer. The part configuration is specified by its pose, that is, its position and orientation. We use linear normal pushes, which are straight-line pushes in a direction normal to the pushing fence. These pushes are specified by the fence orientation and push distance. I prove that a sequence of linear normal pushes can always be found to move any polygonal object from any known initial configuration to any goal configuration in the obstacle-free plane. We formulate the search for such a sequence of pushes as a linear programming problem. We then describe an implemented polynomial-time Pose Planner that uses this formulation to identify a sequence of linear normal pushes given any polygonal object, any initial pose, and any goal pose; we prove this planner complete. The planner, which uses an analysis of the mechanics of pushing an object, generates open-loop plans that do not require sensing. See the example plan in Figure 1.1. We describe experiments with a Puma robot that demonstrate the validity of the generated plans. This method requires a robot with at least 3 planar degrees of freedom.

Chapter 3 introduces a method of manipulating a planar part on a conveyor belt using a robot with just one joint. This approach, which we refer to as "One Joint Over a Conveyor" or "1JOC" for short, replaces a SCARA robot by a fence with a single revolute joint. For a known initial position and orientation of the part, the fence positions and orients the part by a sequence of pushing operations, punctuated by drift along the conveyor. This approach has the potential of offering a simple and flexible method for feeding parts in industrial automation applications. We describe our approach, develop some of its theoretical properties, present a planner for the robot, and outline our implementation. See Figure 1.2 for an example plan on the 1JOC.

The remaining chapters of the thesis concentrate on parts orienting tasks.

Chapter 4 explores a sensorless variant of the 1JOC for orienting planar parts. This variant, called the *Sensorless 1JOC*, can orient and feed polygonal parts up to symmetries in the underlying mechanics, without sensors and without knowing the initial part configuration. This system, consisting of a single joint robot that uses no sensing, is a striking illustration that a minimalist system can perform effective manipulation. We describe a planning algorithm and experimental results with our implemented system.

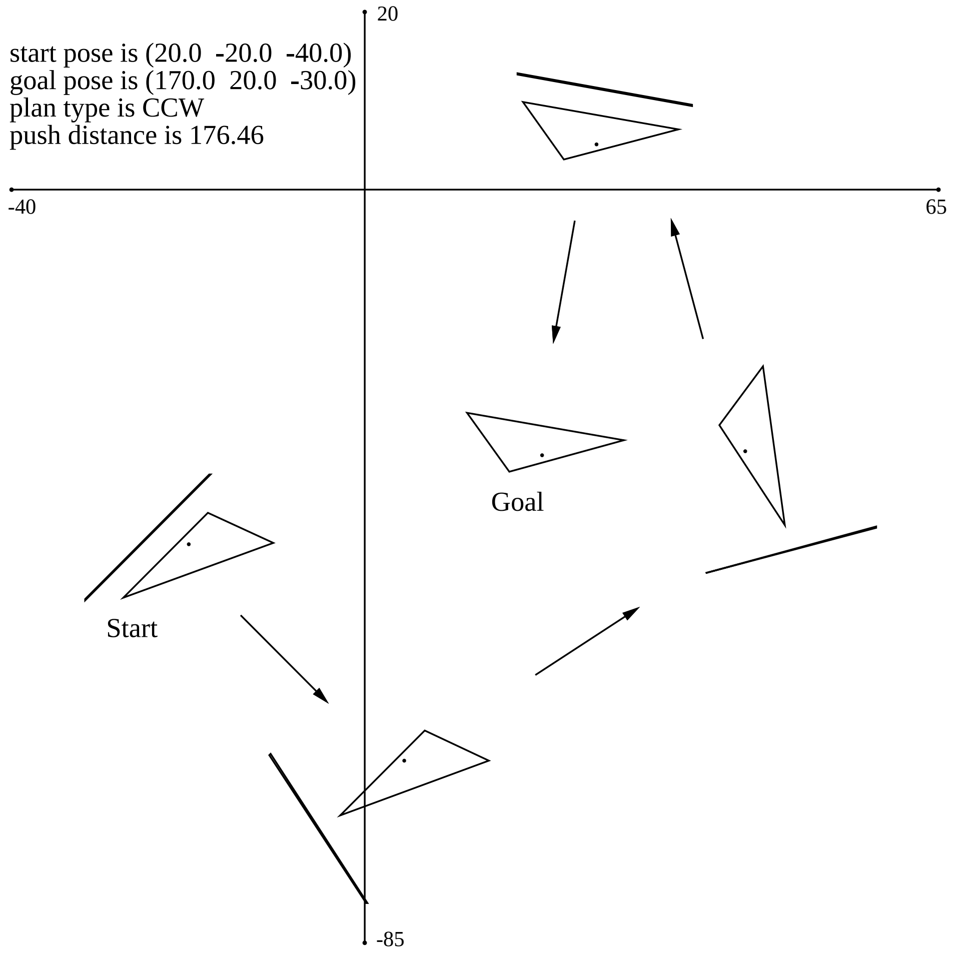

20
start pose is (20.0 -20.0 -40.0)
goal pose is (170.0 20.0 -30.0)
plan type is CCW
push distance is 176.46
-40
65
Goal
Start
-85

Figure 1.1: A plan generated by the Pose Planner to move the triangle from the start pose to the goal pose. The bold lines represent the pushing fence and the arrows indicate push directions.

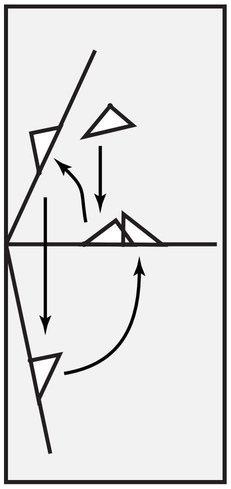


Figure 1.2: Example feeding plan for a triangle with the 1JOC system. The rotational fence motion and linear conveyor motion are used to transfer parts. Conveyor motion is “downwards”.

Chapter 5 analyzes the advantages of using partial sensor information to bring parts in an unknown initial orientation to a known final orientation. The part configuration is specified by the orientation of the part. The robot orients parts on a conveyor belt as shown in Figure 1.3 and uses a simple photoelectric sensor that measures part width to obtain partial information on the part orientation. The sensors reduce the execution length of plans and enable multiple parts to be oriented with the same plan. We describe an implemented system to generate and execute orienting plans. See Figure 1.4 for an example plan.

Chapter 6 explores the effects of variations in part shape on the orienting process. A parts orienting system must be robust to variations in part shape arising from manufacturing tolerances. We extend the parts orienting results of Chapter 5 to handle uncertainty in the shape of parts. For the model of shape uncertainty illustrated in Figure 1.5, we identify a class of orientable parts and describe a planner that generates both sensor-based and sensorless orienting plans that are robust to bounded shape uncertainty. See the example plan of Figure 1.6.

Chapter 7 summarizes the contributions of the thesis and identifies open problems for future work.

There is thus a progression in the problems described in the thesis from parts transfer to parts orienting, known initial part state to unknown initial part state, fixed part shape to uncertain part shape.

1.2 Related work

1.2.1 Uncertainty, Mechanics, and Manipulation Planning

Much effort in robotics has been devoted to overcoming uncertainties arising from robot control errors, sensor errors, and variations in the environment. The use of task mechanics is a powerful solution technique in robotic manipulation tasks; we list a few early examples. Inoue [95] used force information to achieve peg insertion despite inaccuracies in manipulator positioning that exceeded

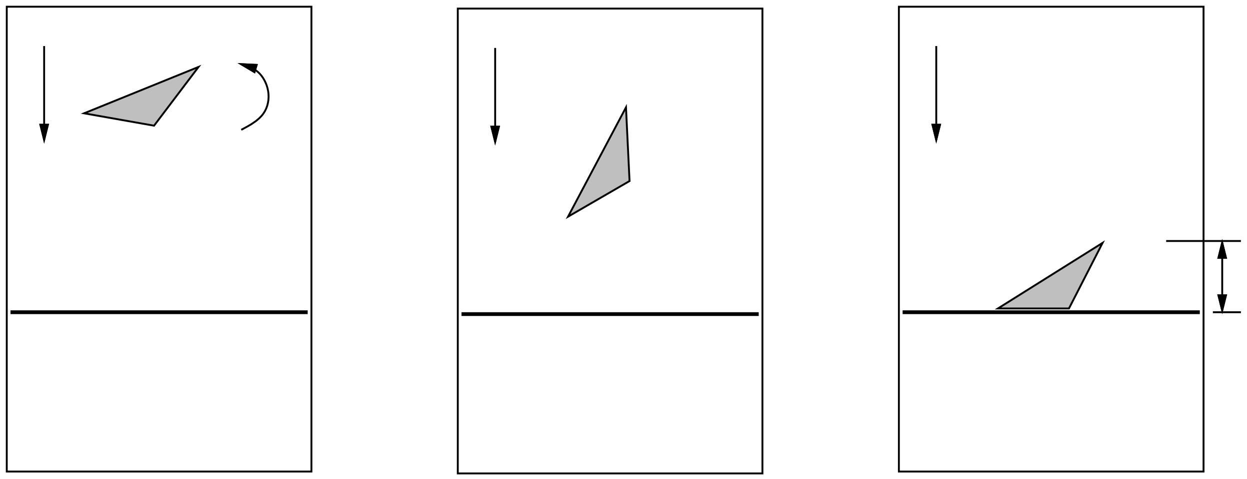


Figure 1.3: Schematic overhead view of a part on a conveyor during a push-align operation for parts orienting. The robot rotates the part through a specified angle before releasing it to drift into contact with the fence on the conveyor. A sensor measures the diameter of the part after it is aligned with the fence. The sensor reading provides partial information on the orientation of the part. A sequence of push-align operations is executed until the part's orientation is determined.

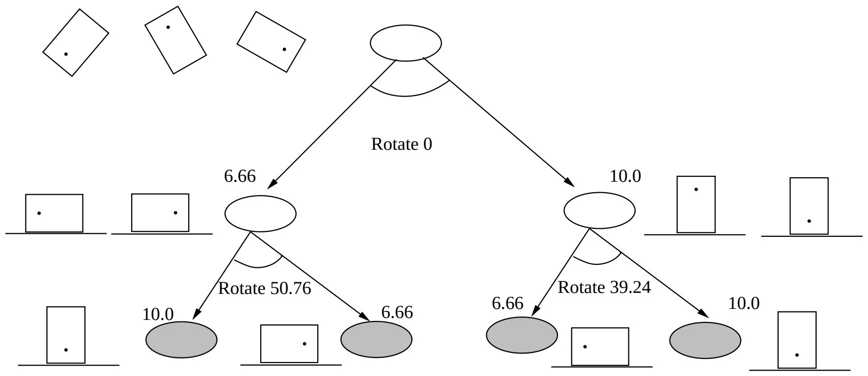


Figure 1.4: A conditional plan to orient a rectangle in an unknown initial orientation. The sensed diameter value shown at a node indicates the possible orientations at that node. Branching during execution occurs based on the sensor value. Goal nodes, shown shaded, have a single orientation.

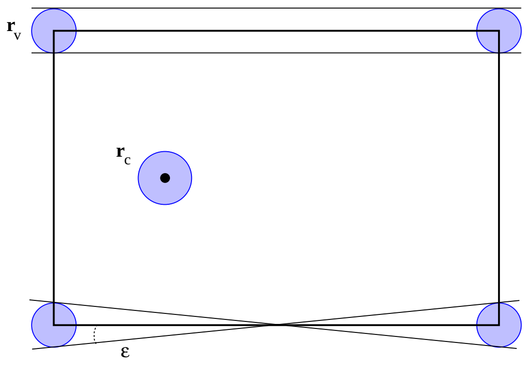


Figure 1.5: Shape uncertainty model indicating the uncertainty bounds on the locations of the center of mass and vertices. The nominal center of mass and edge locations are drawn bold.

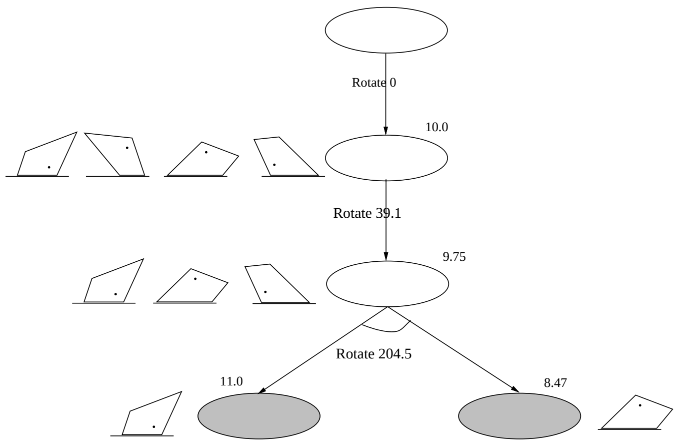


Figure 1.6: A plan using nondeterministic and deterministic actions to orient the quadrilateral in the presence of shape uncertainty. Shape uncertainty causes variations in the diameter values. The average diameter value of the possible states is indicated at each node.

assembly tolerances. This illustrates use of the task mechanics and sensor data to overcome uncertainty and enlarge the set of solvable tasks. Simunovic [168] analyzed the contact forces during the insertion of a peg in a hole and identified conditions under which jamming occurs. Taylor [175] describes a task-level manipulator planning system that generates skeleton plans from which the possible positional uncertainties are estimated, and discusses the use of sensory operations to bound this uncertainty. Brooks [28] extended this approach by using symbolic means to provide error estimates and identify points in a plan where sensing operations are necessary to overcome the uncertainty. Whitney and colleagues [185, 186, 134] have analyzed assembly operations and developed devices, notably the Remote Center Compliance for insertions, to overcome uncertainties during assembly. Compliant motions have been used to overcome positional uncertainties between contact surfaces; see Mason [119] for an overview of issues related to compliant motion.

One approach to reducing uncertainty is to use the stable states resulting from a manipulation operation to constrain the object configurations. Hanafusa and Asada [87] consider grasping an object stably by using a three-fingered robot hand. They use a potential function based on the object geometry and finger spring constants to determine stable grasp states of the object. To aid the design of automated feeding and orienting systems, Boothroyd *et al.* [24] analyze the statistical distribution of the natural rest states of objects. They discuss both the static case (when the part directly falls into a rest state after impact) and the dynamic case (when bouncing and rolling of the part occur after impact). Erdmann and Mason [66] developed an automatic sensorless tray tilting system based on the task mechanics.

1.2.2 Pushing

Pushing is a common manipulation operation, and there has been much work aimed at understanding the mechanics of pushing. Our work builds on previous work on the mechanics and planning of pushing operations. The motion of a pushed object depends on its geometry, the pressure distribution between the object and the support surface, the nature of contact between the pusher and the object, and the motion of the pusher. Mason [120, 121] analyzed the mechanics of robotic pushing operations. He developed a procedure to determine the instantaneous motion of a pushed object with a known support pressure distribution, and derived rules to predict the rotation direction of a pushed object with an unknown pressure distribution. Mani and Wilson [118] used the pushing rules of Mason to derive an Edge Stability Map for straight line pushes and developed a system to orient parts with a sequence of fence pushes at different angles. Brost [29] developed an algorithm to plan grasps with a parallel-jaw gripper that are robust to bounded uncertainties in object orientation. As an intermediate result, using the pushing rules of Mason, he developed the push stability diagram to describe the possible motions of an object being pushed by a fence. Peshkin and Sanderson [141] found the locus of centers of rotation of a pushed object for all possible pressure distributions over an enclosing circle centered at the object center of mass. These centers of rotation provide bounds on the rate of rotation of an object being pushed. From the center of rotation corresponding to the slowest rotation, they calculated the minimum push distance guaranteed to align the object with the fence. Using these results, Peshkin and Sanderson [142] described the orienting effect of a fence in terms of its configuration map, which maps all initial orientations of the object to all possible resulting orientations. They used these configuration maps to find a sequence of fences to automatically orient a sliding part. Brokowski, Peshkin, and Goldberg [27] designed curved fence sections to eliminate uncertainty in the orientations of parts being oriented by the fence. Wiegley et al. [187] developed a complete algorithm to find the shortest sequence of frictionless curved fences to orient a polygonal part. Pham, Cheung, and Yeo [143] analyzed the initial motion of a rectangular object being pushed or pulled and its dependence on the magnitude of the exerted force. Mason [122] considered the problem of pushing a block along a wall. Goyal, Ruina, and Papadopoulos [82, 83] assumed a known support pressure distribution and developed a limit surface description of the relation between the frictional forces and object motion. Alexander and Maddocks [9] derived bounds on the center of rotation locus over all possible support friction distributions for objects with known support regions. Balorda and Bajd [13] used a two-finger tool to reduce the positional uncertainty of an object by pushing it. They discuss the effect of finger configurations on accurate positioning of the object. Goldberg and Mason [78] described a Bayesian framework for planning multi-step grasps with a frictionless parallel-jaw gripper to orient objects. In subsequent work, Goldberg [76] developed a backchaining algorithm to generate sensorless orienting plans for polygonal objects using the frictionless gripper. This algorithm is guaranteed to return the shortest plan to orient any object up to symmetry. In their grasping work, Goldberg and Mason simplified Brost's [29] model for parallel-jaw grasping by using only pushes normal to the face of the gripper. In the work reported in Chapter 2, we use linear normal pushes to orient and translate the object. This work was reported earlier in Akella and Mason [6].

Work by Lynch and Mason ([112]; [116]) is most closely related to our work on parts transfer. Lynch [112] studied the kinematic constraints on object motion consistent with a pusher-object contact configuration, and the force constraints on object motion to ensure the resulting frictional forces could be balanced by the contact forces. He combined these constraints to define a new manipulation primitive, the stable rotational push, which guarantees that no relative motion between the pusher and the object occurs during a push. Lynch and Mason [116] discuss issues of controllability and planning in the use of stable pushes. They describe a pushing path planner, based on the nonholonomic motion planner of Barraquand and Latombe [14], that uses stable pushing motions to synthesize paths in the presence of obstacles. Our planner differs from this planner in several ways, the most obvious being the class of pushes used. Our planner assumes an obstacle-free plane while Lynch and Mason's planner can generate plans in the presence of obstacles. Interestingly, the presence of obstacles can reduce the search space and speed up plan generation while the absence of obstacles can slow it down. Lynch and Mason's planner is not exact in that it finds a path to a neighborhood of the goal configuration whereas the linear programming formulation of our planner provides a computationally efficient method to find exact solutions. Brost [31] presents a numerical integration procedure that returns the initial configurations that guarantee the linear pushing motion of a polygon will bring a contacting polygon to a goal equilibrium configuration. Kurisu and Yoshikawa [104] use an optimal control formulation to generate trajectories to push an object to a goal configuration with a point contact pusher in the presence of obstacles. Jia and Erdmann [99] include dynamics in their analysis of the motion of an object pushed by a finger. By sensing contact positions of the object on the finger, they determine the initial pose of the object.

Work on pushing by mobile robots dates back to work by Nilsson [137]. Okawa and Yokoyama [138] describe a mobile robot for pushing a box to a goal position through a specified set of via points. When the object mass and size are greater than that of the robot, multiple robots may be required to push the object. Donald, Jennings, and Rus [58] use a team of mobile robots to reorient furniture by pushing and analyze the information requirements of the task.

They present several algorithms that differ in the amounts of global control, communication, and synchronization. Brown and Jennings [36] present a two-robot system to manipulate objects by pushing. One robot steers while the other robot pushes the object and the steering robot; there is no explicit communication between the robots.

1.2.3 Nonprehensile Manipulation

Nonprehensile manipulation, where an object is manipulated without being rigidly grasped, takes several forms including pushing, whole arm manipulation, juggling, throwing, and striking. Much of the work on pushing cited above can be viewed as examples of nonprehensile manipulation. The 1JOC approach of Chapters 3 and 4 is an example of using a simple effector to perform nonprehensile manipulation. Salisbury [158] discusses whole arm manipulation, where manipulator link surfaces can contact objects to manipulate them. Nonprehensile manipulation occurs during a grasp when the object slips or rolls relative to the gripper. Trinkle, Abel, and Paul [178] presented a planner that uses both the palm surface and fingers of a frictionless gripper to achieve an enveloping grasp of an object. Brock [25] analyzed the permissible motions of an object grasped in a robot hand and used changes in forces to cause controlled slip of the object. Kao and Cutkosky [101] explicitly consider sliding of robot fingers on an object and plan finger motions so the object follows a desired trajectory. Sawasaki, Inaba, and Inoue [161] used stand-up and tumbling operations with fingers to manipulate objects in contact with a support surface. Aiyama, Inaba, and Inoue [3] use pivoting operations, where they manipulate an object supported on a surface by rotating the object about its vertices. Nagata [130] developed a parallel-jaw gripper with a turntable on each jaw and used it to perform operations such as sliding a block on a plane. Terasaki and Hasegawa [177] describe a manipulation system that slides and rotates an object to enable subsequent grasping. Farahat, Stiller, and Trinkle [71] analytically determine the position and orientation of a polygon moving in sliding and rolling contact with two or three position-controlled robots. Trinkle, Farahat, and Stiller [179] analyze systems of multiple objects and manipulators in contact and determine contact states that are stable to small variations in forces. When the contact state causes the object velocity to be uniquely determined from the manipulator velocity, these first-order stability cells can be used to plan whole-arm manipulation tasks. Abell and Erdmann [1] analyze the use of two frictionless fingers to stably support and rotate a polygonal object in a gravitational field. Akella *et al.* [4] describe a planar manipulation system consisting of a one joint actuator positioned over a conveyor belt. They show that using the conveyor drift field and pushing motions of the actuator, the robot can feed any polygonal part to some selected goal position and orientation. Erdmann [65] developed a two-palm manipulation system that uses a sequence of nonprehensile operations such as sliding to rotate an object. Zumel and Erdmann [191] use two frictionless palms to manipulate an object in a gravitational field, and present a planner that uses stable and unstable transitions to reorient the object.

Objects can be also be manipulated using a very large number of manipulators or by subjecting them to force fields. Böhringer *et al.* [22] use arrays of microelectromechanical actuators to perform sensorless manipulation of parts. By controlling the motions of the actuators in the array, they generate vector fields that produce desired part motions. Liu and Will [110] simulate a parts manipulation surface consisting of microelectromechanical actuators and describe various parts manipulation strategies. Böhringer, Bhatt, and Goldberg [21] use a vibrating plate to position and orient parts sensorlessly. The vibrations generate a force field, and parts move to nodes of the vibratory force field. Swanson, Burridge, and Koditschek [174] analyze the motion of a part subjected to a vibratory juggling motion and indicate conditions under which all initial orientations of the part acquire a unique stable motion.

Juggling, throwing, and striking provide dramatic examples of nonprehensile manipulation. Aboaf, Drucker, and Atkeson [2] developed a robot system that juggles a tennis ball in three dimensions and uses learning to improve its performance. Buehler, Koditschek, and Kindlmann [38] describe "mirror" algorithms to control a one degree of freedom robot system that can stably juggle one or two pucks simultaneously in a vertical plane. Rizzi and Koditschek [157] developed a paddle juggler that performs spatial juggling of two balls. Zumel and Erdmann [190] analyze the juggling of a polygon and identify conditions under which the polygon can be brought to a stable trajectory. Mason and Lynch [124] present a taxonomy of manipulation models and analyze the throwing of a club as an example of dynamic manipulation. Acceleration forces during the carry phase are used to hold the club in a dynamic grasp. Arai and Khatib [10] rotate a cube on a paddle and control its rotation rate by accelerating the paddle using a Puma robot. Lynch [113] explores dynamic nonprehensile manipulation, and demonstrates control of an object using a one degree of freedom robot by sequencing phases of dynamic grasp, rolling, and free flight. Higuchi [90] used impulsive forces generated by electromagnetic means to microposition objects. Huang, Krotkov, and Mason [93] explore the use of striking to move a rotationally symmetric object to a desired position and orientation in the plane.

1.2.4 Parallel-jaw grasping

Grasping, typically with a parallel-jaw gripper, is probably the most common mode by which robots manipulate objects. Brost [29] extended Mason's results to develop an analysis of the behavior of an object being pushed and grasped by a parallel-jaw gripper. In particular, he was able to describe the results of single step grasping operations in the presence of bounded uncertainty in object orientation. Diameter functions were used by Jameson [97] to determine grasp stability for a part grasped in the jaws of a parallel-jaw gripper. Goldberg [75, 76] extended Brost's results by analyzing probabilistic multi-step strategies to orient an object by grasping. By modifying a gripper to be frictionless, and using the diameter function of objects to predict the results of grasp operations, he was able to generate optimal strategies to orient an object. He was able to show that these plans are $O(n^2)$ in length, where n is the number of edges of the object. More recently, Chen and Ierardi [41] have shown these plans to reorient an object by grasping are $O(n)$ in length. An interesting aspect of Goldberg's planner is that it is provably complete, that is, it is guaranteed to generate a plan to orient any polygonal object (up to symmetry). Rao and Goldberg [150] show that for the class of planar polygonal parts, the shape of a part cannot be uniquely recovered from its diameter function. For any polygonal part, there is an infinite set of parts with the same diameter function. They also describe optimal and suboptimal strategies to recognize a part from a known set of parts by grasping it and sensing the jaw diameter. In [147], they present algorithms for orienting curved planar parts. They [148] further extend these results to the case of a gripper with friction between the object and the gripper, and show that under certain conditions, this is similar to the problem of orienting a curved polygonal part with a frictionless gripper.

1.2.5 Parts Feeding

Our long-term goal is to develop parts feeders with practical use in industry. Much of our work was inspired by industrial parts feeders such as vibratory bowl feeders, the SONY APOS system (Hitakawa [91]), and the Adept FlexFeeder. Riley [156] provides a good introduction and survey of the area of industrial automatic assembly.

Vibratory bowl feeders have been in widespread industrial use since their introduction in the 1940s. Boothroyd *et al.* [23] describe parts feeding and orienting devices for automated assembly, including bowl feeders. Singer and Seering [169] analyzed the effects of impacts and oscillatory motions on a part and determine conditions for it to reorient. Their goal was to identify impacts and oscillatory motions that can change the initially unknown state of the part to a desired goal state. Caine [39] considers the design of interacting part shapes to constrain motion and applies it to a vibratory bowl feeder track. Berkowitz and Canny [16] use a dynamic simulator to perform a series of simulated experiments to select bowl feeder gate parameters. Christiansen *et al.* [44] exploit genetic algorithms to identify sequences of bowl feeder gates that improve feeding efficiency.

A more recent and successful industrial parts feeding system is the SONY APOS system (Shirai and Saito [164], Hitakawa [91]). The APOS system relies on vibration and shape to orient parts. Parts are made to flow over a vibrating pallet with nests designed to trap only parts in the correct orientation. The remaining parts are recirculated. Brost [31, 30] demonstrates the use of shape constraints to orient a part by designing a nest to “catch” the part. Krishnasamy, Jakiela, and Whitney [103] analyze the effect of shape and vibration parameters on the energy of parts and hence their efficiency of entrapment in an APOS-like vibration system.

Research on orienting parts using the task mechanics, with and without sensing, goes back to Grossman and Blasgen [86], whose system brought objects to a finite number of orientations in a tilted tray, where their orientation was determined by a tactile probe. Erdmann and Mason [66] explored the use of sensorless manipulation strategies to eliminate uncertainty in the configuration of a part in a tray by repeated tilts of the tray. They implemented a planner that used a knowledge of the mechanics of sliding to predict the results of actions. Natarajan [131] focused on the computational issues related to the automated design of sensorless parts feeders. By assuming deterministic transitions of an orienting device and monotonicity conditions, he was able to derive polynomial time algorithms for certain classes of object orienting problems. Eppstein [61] extended Natarajan's results by presenting a more efficient algorithm that is guaranteed to find the shortest reset sequences for monotonic automata. He also extended his technique to certain classes of nonmonotonic automata. Erdmann, Mason, and Vanecek [67] describe a polynomial-time planner for the problem of orienting a polyhedral part by tilting a tray with infinite friction surfaces. Its output would be a sequence of tilt angles, linear in the number of faces of the polyhedron, that is guaranteed to reorient the polyhedron.

Peshkin and Sanderson [142] used results on the motion of a pushed object to find a sequence of fences to automatically orient a sliding part. Brokowski, Peshkin, and Goldberg [27] designed curved fence sections to eliminate uncertainty in the orientations of parts being oriented by the fences. Goldberg [76] developed a backchaining algorithm to orient polygonal parts up to symmetry using a frictionless parallel-jaw gripper. Wiegley *et al.* [187] developed a complete algorithm to find the shortest sequence of frictionless curved fences to orient a polygonal part.

1.2.6 Minimalism

Minimalism in robotics has been studied by Donald *et al.* [56], Canny and Goldberg [40], and Böhringer *et al.* [20]. Raibert [146] and McGeer [126] constructed simple, elegant machines that use dynamics for stable locomotion.

The 1JOC system of Chapter 3 is an underactuated system with a drift field. Such systems have been heavily studied in nonlinear control theory; see, for instance, (Brockett [26]; Crouch [49]). A good introduction to nonlinear control is given by Nijmeijer and van der Schaft [136], and nonholonomic robotic systems are discussed in the texts by Latombe [105] and Murray *et al.* [129]. Hermann and Kremer [89] provide a good introduction to the main issues in nonlinear controllability and observability.

Other examples of manipulation by an underactuated effector include slipping and rolling within a grasp (Bicchi and Sorrentino [19]; Brock [25]; Rao *et al.* [151]) and dynamic tasks such as snatching, rolling, and throwing (Arai and Khatib [10]; Lynch [113]).

The 1JOC can be likened to an underactuated manipulator in a gravity field, where the proximal "shoulder" (fence pivot) is directly actuated, but the distal degrees of freedom (represented by the object) are not. Much of the work on underactuated manipulation has exploited dynamic coupling among freedoms. Research on the controllability of such serial link manipulators has been carried out by Oriolo and Nakamura [139] and Arai and Tachi [11]. Sørdalen *et al.* [171] recently developed a nonholonomic gear which allowed them to construct a controllable n-link planar arm with only two motors.

Arai and Khatib [10] demonstrated rolling of a cube on a paddle held by a PUMA. Their motion strategy was hand-crafted with the assumption of infinite friction at the rolling contact. Lynch and Mason [124, 113, 117] report automatic planning of rolling, throwing, and snatching tasks using a single degree of freedom robot.

1.2.7 Sensor-based manipulation

In early work, Pingle *et al.* [144] demonstrated simple sensing and alignment steps to reduce uncertainty in object positions for programmable assembly examples. Grossman and Blasgen [86] describe a system that orients parts by a combination of tilt, vibration, and probing operations. The manipulator drops the part into a tilted tray which is then vibrated. The part comes to rest in one of a finite set of orientations, which are discriminated using mechanical probing operations. The probe points were specified by the operator. Sanderson [159] used positional entropy measures to determine the relative efficiency of mechanical operations and sensor operations for mechanical assembly tasks. Taylor *et al.* [176] approached automatic planning of sensor-based manipulation programs as a game-theory problem where the goal of the robot is to achieve a specified task state. The robot chooses actions, nature chooses sensor readings, and planning is performed by searching the game tree. Uncertainty in the robot's world model, control errors, and noisy sensors are explicitly modeled. Using an AND/OR search tree, they applied this framework to the task of orienting an object by tray tilting, and the task of orienting and grasping an object by a parallel-jaw gripper. Connell [48] illustrates the effective use of local binary sensors in conjunction with manipulator actions to pick up a soda can. Canny and Goldberg [40] advocate the use of simple, inexpensive sensors and actuators for industrial automation tasks. Hasegawa *et al.* [88] use manipulation primitives called skills to accomplish contact between objects, face alignment, and insertions. These skills use sensor data to achieve desired states despite errors in positioning. Morrow, Nelson, and Khosla [127] describe the use of vision and force based sensorimotor primitives to perform insertions of electrical connectors.

Feedback loops are a particular class of sensor-based operations where the instantaneous sensor data is sufficient to select an action. Kalman filters [125] are used when the uncertainty in sensor information results from system and measurement errors rather than ambiguities in interpretation of the sensor information. Dickmanns and Zapp [52] use Kalman filters to robustly estimate road parameters from vision data, and use this information as input to a digital controller to guide a high speed autonomous vehicle on a road. An important issue in these problems, assuming availability of complete sensors, is the relative difference in the servo rate and the sensor update rate. Papanikolopoulos *et al.* [140] use feedback control algorithms in conjunction with vision data from a camera to demonstrate real-time robotic tracking of objects in 2-D space. They overcome uncertainty due to noisy data and slow processing of images by a combination of stochastic controllers and interpolation techniques. Erdmann [69, 70] explores the advantages of using sensors tailored to the task and actions so they can be used in feedback loops.

1.2.8 Probabilistic manipulation operations

Since many manipulation operations are nondeterministic due to uncertainties in locations, geometries, and contact properties, there is a body of work devoted to analyzing probabilistic techniques. Erdmann [68, 63, 64] has explored the use of randomization to handle uncertainty in robot manipulation tasks. He demonstrates how randomization can expand the class of solvable tasks, reduce the robot's knowledge requirements, and simplify the planning and execution process. In [64], he discusses an approach for synthesizing strategies in the presence of uncertainty that employ both deterministic and randomized motions. Goldberg [75] discusses the role of stochastically optimal plans in manipulation tasks. Goldberg *et al.* [79] use a backchaining algorithm to generate multi-step plans to orient an object, and then use probability and cost measures to identify the plan which maximizes expected feedrate.

Christiansen and Goldberg [45] explored automatic open-loop planning for task domains with stochastic actions. They describe two planners for a tray-tilting domain where the effects of actions cannot be predicted due to control error, friction, and unmodeled dynamics. The first planner uses a forward exhaustive search to find a plan most likely to produce the desired goal, and the second planner uses a backward best-first search to find a plan that maximizes the lower bound on the probability of reaching the goal. Christiansen [43] describes a system that learns a physical model of a task with nondeterministic actions. Brost and Christiansen [33] introduce the probabilistic backprojection as a construction for robust manipulation planning in the face of uncertainty. This extension of the LMT framework (see Section 1.2.9) requires knowledge of the error distributions of sources of task uncertainty.

1.2.9 Motion planning with uncertainty

Lozano-Perez *et al.* [111] suggested a framework for synthesizing compliant motion strategies under uncertainty in sensing and control. In this framework, commonly referred to as the LMT framework, they define the preimage of a goal as the set of configurations from which the goal can be attained in a single compliant motion. To generate a strategy, their algorithm considers the preimage of the goal region as a subgoal, and recursively computes the preimages of the subgoals until they include the initial region. Erdmann [62] separated the issues of goal reachability and recognizability and implemented a planner that used the backprojection of regions rather than the preimages. Buckley [37] developed a planner to synthesize conditional compliant motion strategies that succeed in the presence of uncertainty in robot sensing, control, and the initial configuration of the robot. He makes the planning problem tractable by considering a finite state space consisting of vertices, edges, and faces of the configuration space of the robot. The planner considers only translational motions and uses a combination of three motion termination predicates. The planner is complete up to the chosen state space decomposition. Donald [53] extended the framework to allow plans that succeed or fail recognizably and handle variations in part dimensions. See Latombe [105] for a discussion of this body of work.

Hutchinson [94] extends the LMT approach by using visual constraints to implement visual guarded moves and visual compliant motions. Lazanas and Latombe [108] use landmark regions, within which the robot has perfect control and perfect sensing, to tackle a reduced version of the motion planning under uncertainty problem. They present a complete planner for this problem, and illustrate how designing the environment can simplify the task substantially. Shekhar and Latombe [163] describe how the set of initial states from which the goal can be achieved (the preimage) is enlarged by using knowledge of initial and final state. These are examples of the characterization of the addition of sensing and internal state to improve robot performance.

Donald and Jennings [57] discuss the occurrence of perceptual equivalence classes in mobile robot tasks. By studying the reachability and recognizability graphs (RR-graphs) for a mobile robot navigation task and their variation as the robot moves around, they wish to generate solutions for problems with a similar sensory structure. For their task, the projection map is unstructured and it is not clear that solution synthesis is always possible.

Donald [54, 55] develops the idea of information invariants to characterize sensors, tasks, and the complexity of robotic operations. His goal is an understanding of the information necessary to accomplish a task, and the different ways it can be manifested (sensing, action, communication, history, internal state). This work is of particular relevance when evaluating the result of adding a sensor. Erdmann [69, 70] has developed techniques to determine an abstract sensor given a set of actions for a task, and uses this to identify the minimal sensor which can then be physically implemented. The data from this sensor is used in a feedback loop to achieve the goal. LaValle [107] has described a game-theoretic framework for robot motion planning problems in the presence of uncertainty. He has used it to model problems involving control and sensing uncertainty, partially-predictable variations in the environment, and coordination of multiple robots.

1.2.10 Pose determination and object localization

Much research has been devoted to the problem of recognizing an object from a known set of objects, and determining its pose. We restrict our discussion to cases where the sensor data is incomplete and hence, multiple sensory operations are required. Gaston and Lozano-Perez [74] introduced the Interpretation Tree as a structure to recognize and localize polyhedral objects in the plane using tactile sensors. Grimson and Lozano-Perez [85] use local measurements of three-dimensional positions and surface normals to recognize and locate objects with six degrees of freedom. The local constraints used, such as distance between points, and angles between the surface normal vectors, are very effective in pruning invalid interpretations. Grimson [84] extends this work so the robot can distinguish between ambiguous poses of an object. Ellis [60] finds paths for a tactile recognition probe to uniquely determine the pose of a 2-D object in the presence of sensed data error given initial tactile data. Faugeras and Hebert [72] describe a system to recognize and locate a 3-D object using range data. As part of the localization process, they conduct a tree search to match scene features with model primitives. They exploit rigidity constraints by computing the tranformation necessary to match corresponding scene and model features, and pruning matches that are inconsistent with computed transformations.

There is also related theoretical work on determining the shapes and poses of polygons by geometric probing. Cole and Yap [47] discuss the number of probes necessary to determine the shape and position of a polygon. Skiena [170] describes a variety of probing models and provides a list of open problems in probing. Alevizos *et al.* [8] contain results on determining the shape of a simple polygon by a minimal number of probes.

Wallack, Canny, and Manocha [183] discuss the use of cross beam sensors to determine the orientation of an object by measuring its diameter at three angles. Rao and Goldberg [149] consider the placement of fiduciary marks on an object for optimal distinguishability of a known set of poses. Jia and Erdmann [98] examine the issue of finding the minimum number of sensing points required to distinguish between a finite set of poses of a known set of polygonal shapes. Siegel [165] describes the use of joint angle and torque sensing to determine the pose of planar objects grasped by the Utah-MIT hand.

1.2.11 Artificial Intelligence Planning

AI search techniques such as breadth-first search, A* search, AND/OR tree search are discussed by Rich and Knight [155]. Buckley [37] uses a state-graph representation which is essentially an AND/OR graph representation for compliant motion planning. de Mello and Sanderson [51] use AND/OR graphs to represent all possible subassemblies and the sequence of assembly operations during assembly of a part. Taylor *et al.* [176] explored the use of AND/OR search trees in the manipulation domain.

Russell and Wefald [184] discuss the use of decision-theoretic techniques in conjunction with traditional AI planning techniques. Chrisman and Simmons [42] use static sensing policies to determine plans which are near cost optimal with respect to both action and sensing. Simmons [166] explores, by coupling symbolic constraints and probability estimates, the effect of changing the resolution and frequency of sensing operations on the expected cost of a plan. Stentz [172] describes a new dynamic path planning algorithm called D* for mobile robots operating in unknown environments. As the robot moves towards the goal, the sensors report obstacles, and the algorithm recomputes the optimal plan by propagating arc costs minimally. He shows that the implemented planner is efficient, optimal, and complete. Elfes [59] describes a mobile robot that can dynamically plan and control sensor activities using information-theoretic measures to characterize the different task requirements. Simmons and Koenig [167] use partially observable Markov decision process (POMDP) models to keep track of the location of an indoor mobile robot. The approach estimates the probability distribution of the robot location while accounting for environment, sensor, and actuator uncertainties.

1.2.12 Shape Uncertainty and Tolerancing

The need to manufacture parts specified by tolerances arose with the development of mass production methods that required parts that could work interchangeably [181]. Current industrial standards include the ANSI Y14.5M-1982 tolerancing and dimensioning standard [12].

Voelcker [181] surveys the area of tolerancing and describes the trend to move from tolerancing standards specified by figures and examples to those chosen on a mathematical basis. The goals of this mathematization are to avoid ambiguous interpretations, establish correct measurement techniques, and identify data structures and algorithms suitable for use in geometric modeling systems. Requicha [152, 153] developed a mathematical formalism for the analysis and design of toleranced parts. He used tolerance zones to describe the valid variational class of parts and offset operations to generate these tolerance zones. Requicha [154] surveys various approaches to tolerancing and uses the offset zone tolerancing technique to describe the advantages of a mathematical theory of tolerances. Neumann [133] describes a new dimensioning and tolerancing standard, ASME Y14.5M, and Walker and Srinivasan [182] describe a companion standard, ASME Y14.5.1M-199X, that provides a mathematical basis for dimensioning and tolerancing. Yap [188] advocates the use of computational geometry techniques and exact computation for tolerancing metrology. Such algorithms would estimate the deviations from tolerance of manufactured parts using a set of sampled measurements.

Perhaps the first work that explicitly considered shape uncertainty effects for a manipulation problem was by Donald [53]. He defined a "generalized configuration space", which has additional dimensions for the parametric variations in the part features. He then used this generalized configuration space to generate multiple-step compliant motion strategies. Brost [32] developed a shape uncertainty model and has used it ([35]) in the automatic design of fixtures for 3-D parts that are robust to variations in part shape. Latombe and Wilson [106] have studied the problem of assembly sequencing with toleranced parts. They present an algorithm to determine if an assembly sequence exists for all instances of the components within the specified tolerances, and to generate it when it exists. They assume part edges vary only in position and not in orientation, and consider the geometric clearance issues for assembly without considering changes in stability of the assemblies. Joskowicz, Sacks, and Srinivasan [100] compute the variations in kinematic behavior of mechanisms from the tolerance specifications of their parts. Toleranced parts in contact have multiple and changing contacts which makes their behavior complex. The shape uncertainty model used in this thesis was earlier described in Akella and Mason [7]. It can be viewed as a parametric tolerance model which permits variations in both the orientations and positions of part edges. Inui, Miura, and Kimura [96] analyze the variations in relative positions of two contacting parts when there are variations in their shapes.

1.2.13 Completeness

For a given task, we select a set of actions and try to determine what combination of the actions, if any, can solve instances of the task. One interesting aspect of the problems described in Chapters 2, 3, and 4 is that for the selected class of actions, a solution always exists, and further, we can always find a solution to the problem. Such problems, which always have a solution for any instance of the problem, have been termed solution-complete by Goldberg [77]. He gives examples of other solution-complete problems such as sensorless orienting of parts (Goldberg [76]), and controllability and motion planning for nonholonomic mobile robots in the obstacle-free plane (Barraquand and Latombe [14]). He describes a modular fixturing problem for polygonal parts (Zhuang, Wong, and Goldberg [189]) that is not solution-complete, but for which a complete algorithm, an algorithm guaranteed to return a solution if it exists and report failure otherwise, exists (Brost and Goldberg [34]). Characterizing problems in this manner and developing complete algorithms to solve them enables us to identify and guarantee capabilities of our robot systems.

1.3 Thesis Contributions

This thesis demonstrates the use of the task mechanics and geometry in developing algorithms and systems with provable properties for a variety of parts transfer and orienting problems. The thesis makes the following principal contributions:

- Proves the completeness of pushing for planar parts transfer. I prove that there always exists a sequence of linear normal pushes to move any polygonal part from any initial position and orientation to any goal position and orientation without using a sensor.
- Demonstrates that a robot with a single degree of freedom effector over a conveyor can perform parts transfer and orienting of any polygonal part.
- Introduces linear programming and nonlinear programming formulations that yield efficient automatic planners for parts transfer.
- Demonstrates the advantages of using partial sensor information with manipulation operations to speed up parts orienting and to handle multiple part shapes.
- Characterizes the effect of part shape variations on the orienting process and identifies a class of parts that can be oriented even with shape uncertainty.
- Experimentally demonstrates the implemented parts transfer and orienting systems.

1.4 Summary

Flexible parts transfer and orienting is a central problem for the development of automated assembly systems. By exploiting our knowledge of the mechanics and geometry of the tasks, we can develop flexible parts transfer and orienting systems. This thesis presents a set of parts transfer and orienting systems to demonstrate this. The systems described in this thesis use simple effectors such as straight-line fences and simple sensors such as photoelectric devices. Since the effectors permit only nonprehensile manipulation and the sensors provide only partial state information, we develop new techniques to use them effectively. The simple effectors allow handling of a broad class of parts and the partial information sensors reduce the execution length of plans.

The use of simple effectors and sensors shifts the burden from hardware to software. By developing automatic planners for parts transfer and orienting systems, we can generate parts feeding solutions for new parts. Further, the same tools can be used in the early stages of design to evaluate the feedability of parts and to enable virtual prototyping. While selecting action sets and developing planners for a task, we have sought to characterize the completeness properties of both the action sets and the planners. We have also characterized other properties such as the class of parts that can be handled, the length of plans, and computational complexity of the planners.

The remaining chapters of the thesis illustrates these themes by describing a set of planar transfer and orienting tasks, presenting solutions, and characterizing the developed systems.

Chapter 2

Posing Parts in the Plane by Pushing

Robots usually grasp objects rigidly to move them. Robots can also move objects without grasping them by a variety of nonprehensile techniques such as pushing, throwing, and striking. Pushing is a form of nonprehensile manipulation useful for planar manipulation of objects, particularly when the object cannot be grasped or when it is more efficient to move the object along the support surface. An example is planar parts transfer for assembly, where parts are to be moved from one position to another in the plane, often with a change in orientation. If a part is too heavy or too large for the gripper to grasp, it can be moved to different locations by a sequence of pushes. Another example is movement of large containers and pallets on a floor by a mobile robot. In these and similar cases, pushing actions are a suitable means to manipulate the objects.

In tasks involving contact between the robot and the environment, the mechanics of the task determine the nature of the interaction between the robot and the environment. Understanding the task mechanics helps us achieve the desired outcome. This typically involves identification of a set of actions, a description of the results of these actions, and knowledge of the conditions under which these actions are successful. Often, the task is impossible with a single action. In such cases it is necessary to determine how to chain a sequence of actions together to accomplish the task.

This chapter is concerned with the automatic synthesis of sequences of linear pushing actions for planar transfer of a part, given knowledge of the mechanics of a single pushing action. Previous work in the domain of pushing provides a basis for understanding the mechanics of such pushing actions (Mason [121]; Brost [29] ; Peshkin and Sanderson [141]). Sequences of such actions have been used primarily to orient parts (Mani and Wilson [118]; Peshkin and Sanderson [142]). Here we develop a method to find plans to move a polygonal part from a known initial position and orientation to a goal position and orientation using linear normal pushes executed by a fence. These plans are open-loop — they do not require sensing. A linear normal push is a straight-line push in a direction normal to the pushing fence, and is specified by the fence orientation and push distance. We show that a set of linear normal pushes can always be found to move any polygonal part from any initial configuration to any goal configuration in the obstacle-free plane. The part configuration is specified by its *pose*, that is, its position and orientation. We formulate the search for a sequence of pushes as a linear programming problem whose feasible solutions provide valid plans. We have used this formulation to implement a polynomial-time *Pose Planner*, and proven the planner complete; it is guaranteed to generate open-loop plans to move an arbitrary polygonal part from an arbitrary start position and orientation to an arbitrary goal position and orientation in the obstacle-free plane. We further describe experiments that demonstrate the validity of the generated plans. See Figure 2.1 for an example plan.

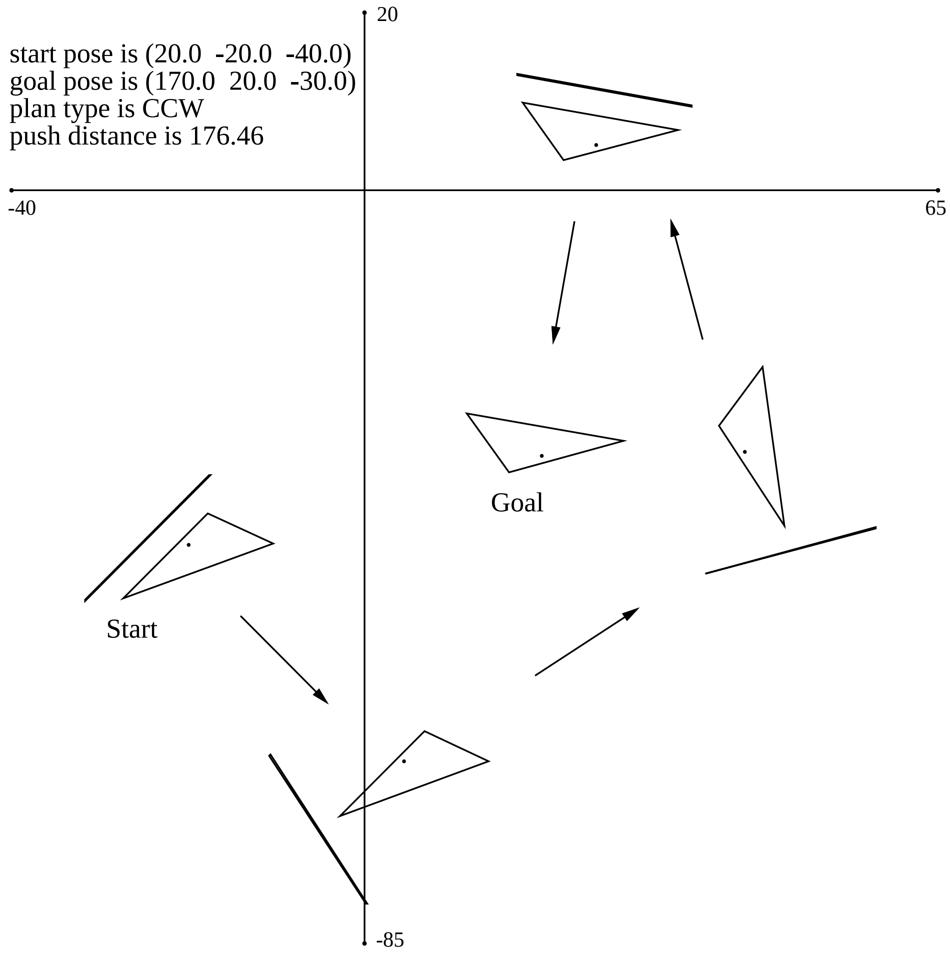

start pose is (20.0 -20.0 -40.0)
goal pose is (170.0 20.0 -30.0)
plan type is CCW
push distance is 176.46
20
-40
65
-85
Goal
Start

Figure 2.1: A plan generated by the Pose Planner to move the triangle from the start pose to the goal pose. The bold lines represent the pushing fence and the arrows indicate push directions.

This chapter focuses on three issues:

1. Generating sequences of linear normal pushes for the problem of simultaneously orienting and positioning polygonal parts.
2. Proving that the set of linear normal pushes is complete for this task. That is, there always exists a sequence of linear normal pushes to move any polygon from any start pose to any goal pose in the obstacle-free plane.
3. Showing the Pose Planner that generates the push sequences for this task is complete. That is, the Pose Planner always generates a push sequence guaranteed to move any polygon from any start pose to any goal pose.

This chapter is organized as follows. Section 2.1 states the problem and outlines the solution. Section 2.2 discusses the relevant results on the mechanics of pushing a part. Section 2.3 describes the selection of a feasible set of pushes using a linear programming formulation and proves completeness of the set of actions. Section 2.4 outlines the Pose Planner and presents completeness and complexity properties of the planner. Section 2.5 presents a linear mixed-integer programming formulation to minimize the execution time of plans. Section 2.6 describes experimental validation of the generated plans. The final section concludes with a summary of our results and suggestions for future work.

2.1 The Planar Pose Problem

Given a polygonal part on a horizontal table at a known start pose, we are to find a pushing plan to move it to a specified goal pose. We call this the *Planar Pose Problem*. The pushing actions are executed by means of a fence — a flat edge used as a pusher. The pose is described by the orientation of the part and the position of the center of mass of the part. The start and goal poses may be arbitrary.

2.1.1 Outline of the Approach

To make the analysis tractable, we partially decouple the problems of orienting and positioning the part. We first consider the required change in orientation, and select a sequence of intermediate orientations, achieved by using *reorient pushes* to rotate the part. The intermediate orientations depend on whether the rotation is clockwise or counterclockwise, and are selected based on an analysis of the mechanics of pushing a part with a fence. The reorient pushes do not usually combine to move the part to the goal position, so we need a set of *translation pushes* to translate the part without changing its orientation. These pushes can occur at the start, intermediate, and goal orientations of the part. The reorient and translation pushes produce a net translation of the part; the lengths of the pushes are chosen so this translation is equal to the desired translation from the start position to the goal position. Thus the plans generated by the Pose Planner consist of a sequence of reorient and translation pushes to accomplish the required pose transformation. An example plan is depicted in Figure 2.2.

We make the following assumptions for our analysis:

1. The part geometry and the location of its center of mass (COM) are known. The center of mass is in the interior of the part.
2. All parts are polygons. Nonconvex polygons are equivalent to their convex hulls.
3. All motions are in the horizontal plane, which is assumed infinite in extent and obstacle free.
4. The fence is of sufficient length that the part does not contact the ends of the fence during pushes, and the part does not roll off it.
5. The fence is position-controlled.
6. All motions are quasi-static. That is, inertial forces are assumed negligible compared to frictional and applied forces.
7. Coulomb's law of friction describes all frictional interactions.
8. There is no slip between the part and the fence. Lynch and Mason [115] have shown that slip can occur even with an infinite coefficient of friction. For the normal linear pushes used here, however, the no-slip assumption is consistent with an infinite coefficient of friction at the contact.

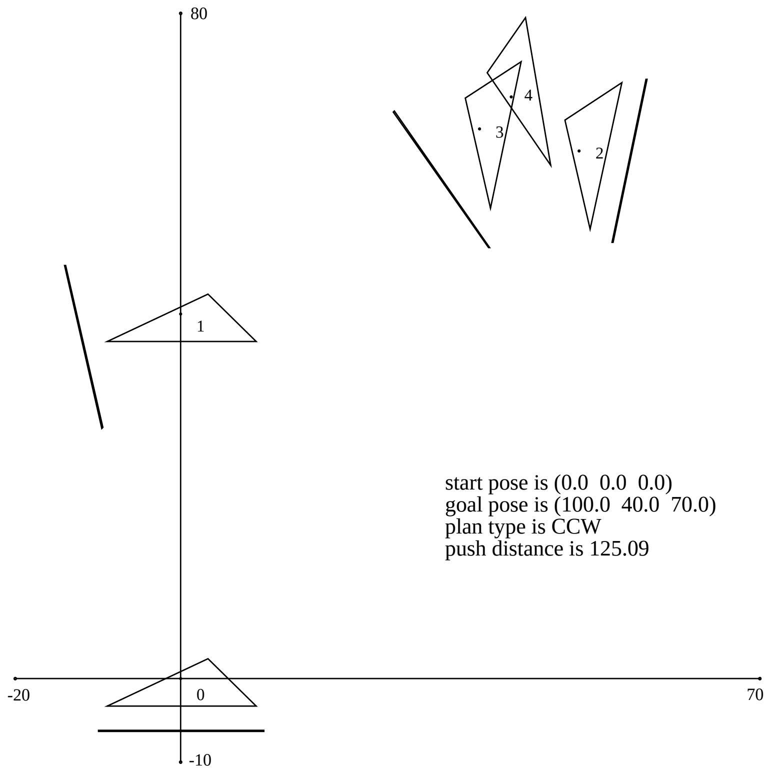

80
4
3
2
1
start pose is (0.0 0.0 0.0)
goal pose is (100.0 40.0 70.0)
plan type is CCW
push distance is 125.09
-20
0
70
-10

Figure 2.2: An example pose plan for a triangle. The poses are numbered in sequence; the zeroth and fourth poses are the start and goal poses respectively. The first and third poses are achieved by translation pushes while the second and fourth poses are achieved by reorient pushes. The bold lines represent the fence. A pose is indicated by (orientation, COM-x, COM-y).

9. Support friction is uniform over the plane, and all support points lie inside the convex hull of the part.

2.2 The Mechanics of Pushing

As a first step, it is necessary to understand the motion of a pushed part. We outline the different classes of pushes and the types of part motions that can result. We use the class of linear normal pushes, which can effect translations and known rotations of the part. For this class of pushes, we determine the translational and rotational motions that a part can undergo without introducing uncertainty in its pose.

2.2.1 Types of Pushes

The pusher motion and the contact conditions between the part and the pusher influence the motion of the pushed part. We identify the following classes of pusher motions:

1. *Linear pushes*: The pusher moves with a fixed orientation along a straight line in a specified direction for a specified distance. The result of a linear push depends on the angle between the fence and the part, the distance the fence moves in contact with the part, and the direction of the push relative to the fence.

2. *Rotational pushes*: The pusher rotates about a point, its center of rotation, through a specified angle. The result of the rotational push depends on the angle between the fence and the part, the rotation angle, and the center of rotation. A linear push may be viewed as a rotational push with the center of rotation at infinity.

Given a pusher motion, we identify the following classes of part motions relative to the pusher:

1. *No relative motion between the part and the pusher*: Here there is no slip or rolling at the contacts between the pusher and the part, and so the part has the same motion as the pusher. This class of motions occurs during the translation pushes defined in Section 2.2.2, and the stable rotational pushes of Lynch [112].

2. *Known relative motion between the part and pusher*: When the pressure distribution is known, the part motion can be predicted. The pressure distribution is usually unknown, however, and the instantaneous relative motion between the part and the pusher cannot be determined. In such cases, it is sometimes possible to determine the net relative motion. An example is the reorient push described in Section 2.2.2.

3. *Unknown relative motion between the part and pusher*: This occurs when the support pressure distribution is unknown and there are insufficient constraints on the motion of the part. The incomplete pushes described in Section 2.2.2 belong to this category.

2.2.2 Linear Normal Pushes

The class of push actions selected for our task influences the type and scope of the generated solutions. We use *linear normal pushes*, where the fence moves in a straight line normal to its face with a fixed orientation for a specified distance. Given the part's orientation, the orientation of

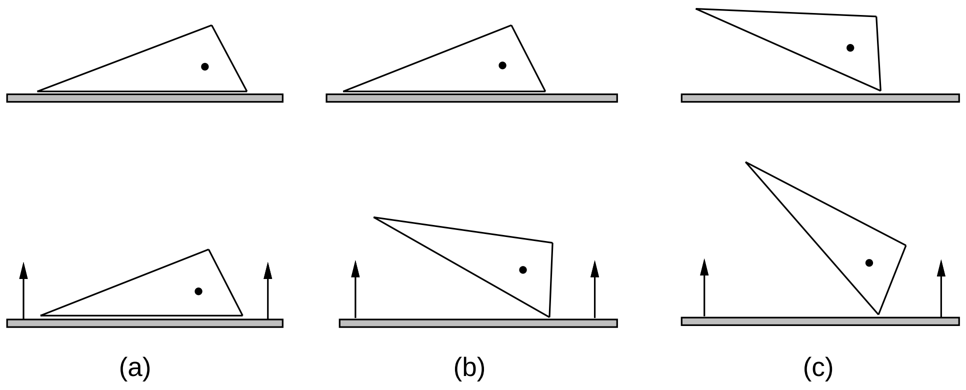


Figure 2.3: Linear normal pushes. (a) Translation push. (b) Reorient push. (c) Incomplete push. The center of mass is indicated by the dot.

the fence and the length of the push are then sufficient to describe a push. When a fence lined up parallel to a part edge executes a push, the part edge either remains aligned with the fence or rotates away. If the edge remains aligned with the fence, it is a *stable edge*. If it rotates away, it is an *unstable edge*. The location of the center of mass determines if an edge is stable or unstable; an edge is stable if the perpendicular projection of the center of mass to the edge lies in its interior. We permit pushes only on stable edges to ensure that once an edge is aligned with the fence, it does not rotate away. Linear normal pushes with a fence along a stable edge of a part can be classified as follows (Figure 2.3):

1. **Translation push**: The fence and part edge are always parallel and there is no change in the orientation of the part as the fence translates.

2. **Reorient push**: The push begins with an initial angle between the fence and part edge, and the part rotates so the edge is aligned with the fence by the end of the push. This push reorients the part to a known orientation. We refer to the minimum push distance guaranteed to align the part edge with the fence for a given initial angle as the *Peshkin distance* for that reorientation. We compute this distance from the results of Peshkin and Sanderson [141], as shown in Appendix A.

3. **Incomplete push**: The push begins with an initial angle between the fence and part edge, but unlike the reorient push, the part does not rotate sufficiently for the edge to be aligned with the fence by the end of the push. So the final orientation of the part is not known.

To avoid uncertainty in the part orientation, we use only translation pushes and reorient pushes.

2.2.3 Part Classification

The ease with which a part can be posed by linear normal pushes depends on the ease with which it can be translated in a given orientation. A relevant question is: Can a part be translated to any point in the plane without being rotated?

The *translation image* of a part at a given pose is the region of positions to which the part can be translated without a change in orientation. To reach the goal pose, the part should attain the

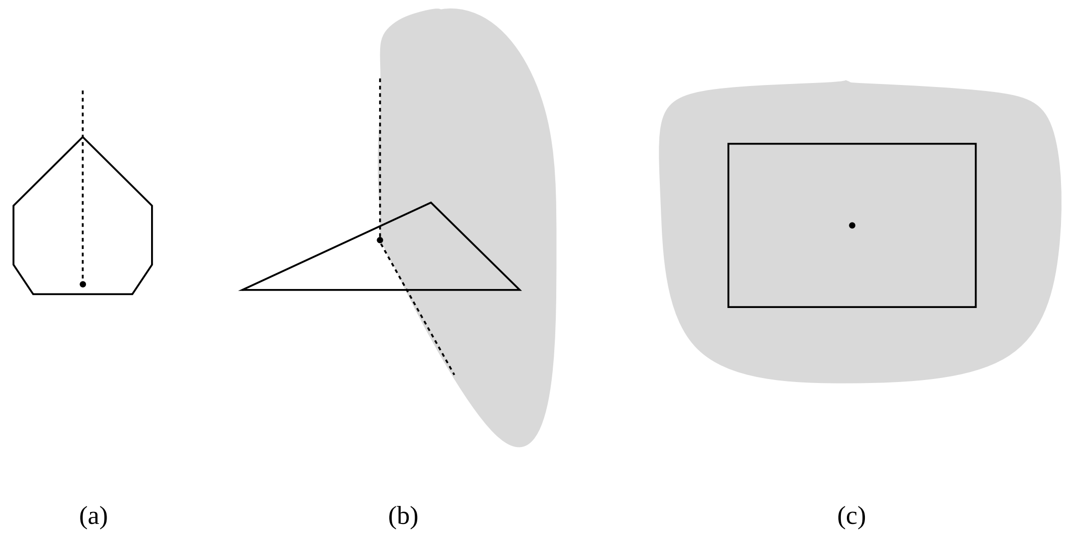


Figure 2.4: Examples of the three part classes with their translation images shown shaded. (a) Uni-stable. (b) Biplus-stable. (c) Spanning-stable.

goal orientation at a position whose translation image includes the goal position. The translation image is defined by the convex cone of the inward normals to the stable edges at the center of mass. It leads to the following classification of polygonal parts subject to linear normal pushes:

1. **Uni-stables**: These are parts with only one stable edge, and hence their translation image is a ray extending to infinity in one direction from their center of mass.

2. **Biplus-stables**: These parts have more than one stable edge such that the normals to the stable edges do not positively span $\mathcal{R}^2$, the plane. The translation image of a biplus-stable is a cone located at its center of mass.

3. **Spanning-stables**: These parts have at least three stable edges such that the normals to the stable edges positively span $\mathcal{R}^2$. Therefore the translation image of these parts is the entire plane.

The above classification is illustrated in Figure 2.4. Since only the spanning-stables have a translation image that spans the plane, uni-stables and biplus-stables cannot always be translated between any two positions without a change in orientation. Figure 2.5 illustrates this for a biplus-stable.

2.2.4 Stable Edges and the Angle-Eating Heuristic

To perform translation and reorient pushes on a part, we need to determine the stable edges of the part. For each stable edge, we have to find the largest angles between the edge and the fence for clockwise (CW) and counterclockwise (CCW) rotations that guarantee the edge will rotate into alignment with the fence during a reorient push. For linear normal pushes, we follow Goldberg [76] in obtaining this information from the *radius function* of the part (Figure 2.6). The radius of a polygon is the perpendicular distance from a reference point in the polygon to a support line. The

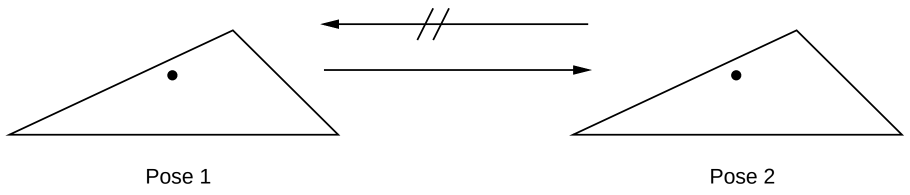


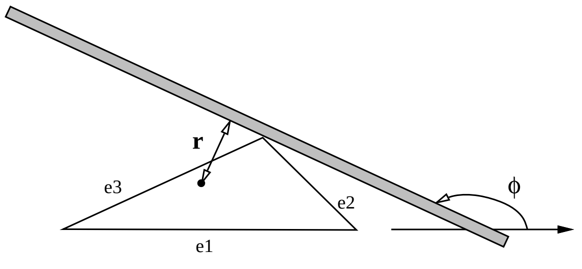
Figure 2.5: The triangle of Figure 2.4 can be translated from pose 1 to pose 2 since pose 2 lies in the translation image of pose 1. However the triangle cannot be translated from pose 2 to pose 1.

radius function $r : \mathcal{S}^1 \rightarrow \mathcal{R}^1$ is a plot of the radius as the support line is rotated. When the center of mass is the reference point and the fence is the support line, the radius function indicates the stable edges of the part for a linear normal push. Further, it gives the maximum angle, for each rotation direction, between any stable edge and the fence for which the edge will stably reorient onto the fence. (The push-stability diagram developed by Brost [29] gives us the same information for the broader class of linear pushes. In fact, we use the push-stability diagram in our implementation.) For a given rotation direction, the *maximum reorient angle* for a stable edge is the maximum angle we permit between the fence and edge for a reorient (Figure 2.6). To avoid introducing uncertainty in part pose, we permit only pushes involving stable edges and do not allow rolling from one edge to another. The maximum reorient angle we use is therefore always less than $90°$. For each stable edge, we find the CW and CCW maximum reorient angles permitted for CW and CCW rotations. For a successful reorient push, we can pick any angle less than the corresponding maximum reorient angle. This *step angle* is the magnitude of the reorientation achieved by each reorient push.

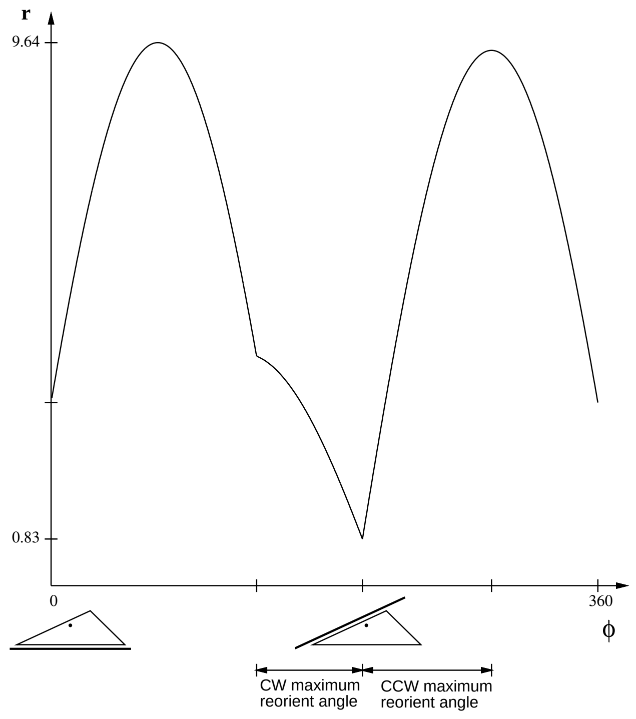
The *angle-eating heuristic*, used to generate reorient sequences, seeks to minimize the number of pushes. The reorientation strategy requires that the part go through a sequence of intermediate orientations to attain the goal orientation. For each rotation direction, these intermediate orientations depend on the magnitude of the corresponding step angle. The heuristic assumes that the larger the step angle, the smaller the number of pushes required to get the part to the goal orientation, and hence to the goal pose. The orientation change is divided into an integer number of step angle reorientations, and if necessary, a final smaller reorientation. See Figure 2.7 for example reorient sequences.

2.3 Determining the Pushes

A pushing plan to pose a part consists of a sequence of reorient and translation pushes. Once a stable edge to perform reorient pushes on and a rotation direction are chosen, the angle-eating heuristic provides a sequence of fence orientation angles that defines a corresponding sequence of reorient push directions. To guarantee that the reorients proceed to completion, the reorient pushes need to have a minimum magnitude (the Peshkin distance). These magnitude constraints coupled with the push directions partially determine the reorient pushes. At this stage, we need to select translation pushes and determine the magnitudes of the reorient pushes to ensure that the part reaches the goal position in the goal orientation.



(a)



(b)

Figure 2.6: The radius function for the triangle. Based on Goldberg [76]. (a) The radius r of a part at a fence orientation ϕ is the perpendicular distance from the center of mass to the fence. (b) The radius function is the plot of the part radius as the fence orientation is varied. The orientation wraps around at 360°. The local minima of the radius function occur at stable edges of the part. Kinks in the radius function occur at unstable edges of the part, and are nonminima of the radius function with a discontinuity in slope. This triangle is a biplus-stable since its radius function has only two minima. The *CW maximum reorient angle* for an edge is the angle from its stable orientation to the nearest leftward local maximum or kink orientation of the radius function. The *CCW maximum reorient angle* is measured similarly in the rightward direction. The CW and CCW maximum reorient angles for edge e_3 are shown.

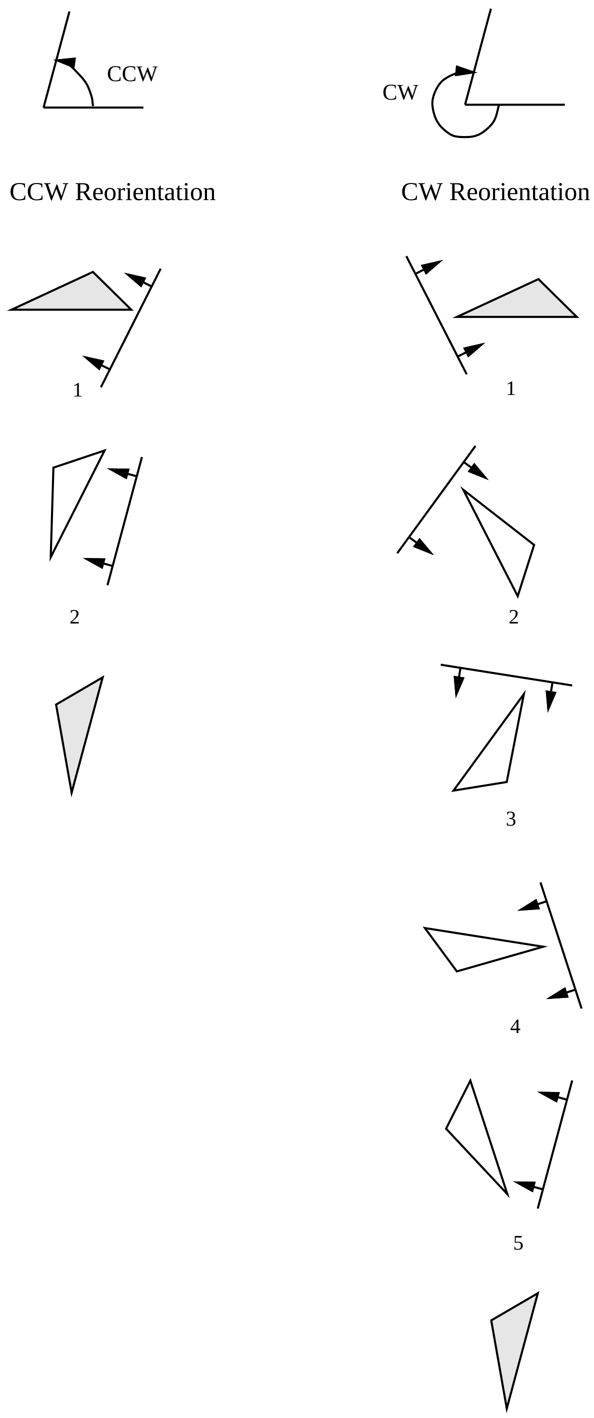


Figure 2.7: CCW and CW reorient push sequences to rotate a triangle. The longest edge of the triangle is the stable edge chosen for the reorient pushes. The start and goal orientations are shown shaded. The reorient pushes for each rotation direction are numbered in sequence. At each orientation, the arrows on the fence indicate the push direction to rotate the part to the next orientation.

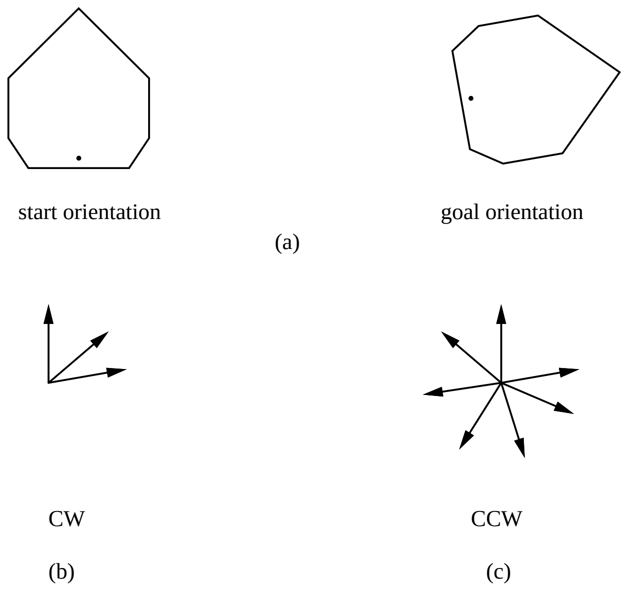


Figure 2.8: Push vector diagrams for a uni-stable. The edge closest to the COM is the stable edge. (a) The start and goal orientations. (b) CW push vector diagram. (c) CCW push vector diagram.

2.3.1 The Pose Vector Equation

To determine the magnitudes of the pushes, we treat them as vectors in the plane. At a given orientation of the part, translation pushes can occur only in the directions of the unit normals into the stable edges. Let this set of unit normal push vectors be denoted by $\{\hat{\mathbf{n}}_{ij}\}$, where $\hat{\mathbf{n}}_{ij}$ is the inward unit normal vector to the jth stable edge in the ith orientation. Let a_{ij} represent the length of the push along $\hat{\mathbf{n}}_{ij}$; it is nonnegative. Let the set of unit reorient push vectors, for the stable edge on which the reorient pushes are performed, be denoted by $\{\hat{\mathbf{p}}_i\}$, where $\hat{\mathbf{p}}_i$ is the unit vector along the push direction to achieve the ith orientation. Let b_i be the length of the push along $\hat{\mathbf{p}}_i$; the minimum magnitude of b_i is the Peshkin distance for the ith reorient.

The *push vector diagram* is a graphical representation of the unit push vectors. For a given reorient sequence, it is formed from the union of the $\hat{\mathbf{p}}_i$ vectors of the chosen stable edge and the $\hat{\mathbf{n}}_{ij}$ vectors of all stable edges at all orientations in the sequence. See Figure 2.8 for examples.

We have assumed that the part does not slip relative to the fence. So during a reorient push, the part rotates about its fixed contact point with the fence as the fence translates. As the part rotates, its center of mass moves relative to the fence. The *center of mass vector* $\vec{\mathbf{c}}_i$ describes this relative motion, and is determined from the $(i-1)$th and ith orientations. Thus the motion of the center of mass due to a reorient push is described by the combination of the reorient push vector $b_i\hat{\mathbf{p}}_i$ and the center of mass vector $\vec{\mathbf{c}}_i$.

The magnitudes of the push vectors, the a_{ij} and b_i terms, are to be determined so that all the vectors combine to obtain the *translation vector* $\vec{\mathbf{T}}$, which is the vector from the start center of mass position to the goal center of mass position. The equation relating all the relevant vectors is the *pose vector equation*

$$\sum_{i=0}^{r}\sum_{j\in S} a_{ij}\hat{\mathbf{n}}_{ij} + \sum_{i=1}^{r} b_i\hat{\mathbf{p}}_i + \sum_{i=1}^{r}\vec{\mathbf{c}}_i = \vec{\mathbf{T}}$$

where r is the number of reorients required to reach the goal orientation and S is the set of stable edges of the part.

A solution to this equation is to be found which satisfies the *nonnegativity constraints*

$$a_{ij} \geq 0 \quad \text{for } i = 0, \ldots, r \text{ and } j \in S$$

and the *Peshkin constraints*

$$b_i \geq m_i \quad \text{for } i = 1, \ldots, r$$

where m_i is the Peshkin distance for the ith reorientation (see Appendix A for details).

2.3.2 A Linear Programming Solution

The magnitudes of the push vectors, the a_{ij} and b_i terms, can be determined efficiently by developing a linear programming (LP) formulation. (See the text by Chvátal [46] for an introduction to linear programming.) Once a reorientation sequence is chosen, the $\vec{\mathbf{c}}_i$ vectors are determined, and the pose vector equation reduces to

$$\sum_{i=0}^{r}\sum_{j\in S} a_{ij}\hat{\mathbf{n}}_{ij} + \sum_{i=1}^{r} b_i\hat{\mathbf{p}}_i = \vec{\mathbf{T}}'$$

$$\text{where } \vec{\mathbf{T}}' = \vec{\mathbf{T}} - \sum_{i=1}^{r}\vec{\mathbf{c}}_i.$$

From the dot product of this vector equation with the unit vectors along and perpendicular to $\vec{\mathbf{T}}'$, $\hat{\mathbf{t}}'$ and $\hat{\mathbf{t}}'_{\perp}$ respectively, we obtain two scalar equations linear in a_{ij} and b_i. In addition, we have a set of magnitude constraints linear in a_{ij} and b_i.

We minimize the total push distance, which is the sum of the a_{ij} and b_i terms. So the LP formulation of the problem is:

$$\begin{array}{ll}
\text{Minimize} & \displaystyle\sum_{i=0}^{r}\sum_{j\in S} a_{ij} + \sum_{i=1}^{r} b_i \\
\text{subject to} & \displaystyle\sum_{i=0}^{r}\sum_{j\in S} a_{ij}(\hat{\mathbf{n}}_{ij}\cdot\hat{\mathbf{t}}') + \sum_{i=1}^{r} b_i(\hat{\mathbf{p}}_i\cdot\hat{\mathbf{t}}') = T' \\
& \displaystyle\sum_{i=0}^{r}\sum_{j\in S} a_{ij}(\hat{\mathbf{n}}_{ij}\cdot\hat{\mathbf{t}}'_{\perp}) + \sum_{i=1}^{r} b_i(\hat{\mathbf{p}}_i\cdot\hat{\mathbf{t}}'_{\perp}) = 0 \\
& a_{ij} \geq 0 \quad \text{for } i = 0, \ldots, r \text{ and } j \in S \\
& b_i \geq m_i \quad \text{for } i = 1, \ldots, r.
\end{array}$$

A solution to the above LP problem is a feasible pose plan and provides a push sequence that minimizes the push distance for the chosen set of intermediate orientations.

We can in fact minimize any linear function of the push distance and the number of pushes using a linear mixed-integer programming (MIP) formulation. See Section 2.5 for the MIP formulation that minimizes the plan execution time and an example plan.

2.3.3 Existence of Solutions

a part can be posed arbitrarily if there exists a sequence of translation and reorient pushes that satisfies the pose vector equation without violating the nonnegativity and Peshkin constraints. We now answer the question: *Does a set of pushes that can move an arbitrary part from an arbitrary start pose to an arbitrary goal pose always exist?*

The pose vector equation can be transformed to:

$$\sum_{i=0}^{r}\sum_{j\in S} a_{ij}\hat{\mathbf{n}}_{ij} + \sum_{i=1}^{r} e_i\hat{\mathbf{p}}_i = \vec{\mathbf{T}}''$$

$$\text{where } b_i = m_i + e_i \text{ such that } e_i \geq 0 \text{ for } i = 1,\ldots,r$$

$$\text{and } \vec{\mathbf{T}}'' = \vec{\mathbf{T}} - \sum_{i=1}^{r}\vec{\mathbf{c}}_i - \sum_{i=1}^{r} m_i\hat{\mathbf{p}}_i.$$

When this equation is satisfied by nonnegative values of the a_{ij} and e_i terms, the corresponding pose vector equation, and nonnegativity and Peshkin constraints are satisfied. This equation is satisfied for any value of $\vec{\mathbf{T}}''$ if the unit push vector sets $\{\hat{\mathbf{n}}_{ij}\}$ and $\{\hat{\mathbf{p}}_i\}$ positively span $\mathcal{R}^2$. The requirements for a set of unit push vectors to positively span the plane is obtained from the following theorem in Davis [50]: A set of vectors $\{\vec{\mathbf{a}}_1, ..., \vec{\mathbf{a}}_r\}$ positively spans $\mathcal{R}^n$ if and only if, for every non-zero vector $\vec{\mathbf{b}}$, there exists an $i \in \{1,\ldots,r\}$ such that $\vec{\mathbf{b}}\cdot\vec{\mathbf{a}}_i > 0$.

The unit push vectors positively span $\mathcal{R}^2$ if they are oriented in the push vector diagram such that for any vector in the plane, at least one of them forms an acute angle with it. The push vectors are chosen using the angle-eating heuristic with a step angle that is always less than 90°. By the above theorem, when the push vectors in the push vector diagram sweep out an angle greater than 180°, they positively span the plane. The angle swept out by the push vectors depends on the required part rotation in the chosen rotation direction. There are two cases to consider:

1. The reorientation is greater than or less than 180°: Since a part can be reoriented by rotating in either the CW or CCW direction, the orientation change in one of the rotation directions is always greater than 180°, and the corresponding set of unit push vectors positively spans the plane.

2. The reorientation is 180°: Here the CW and CCW reorientations are both 180°, and we examine the conditions for each part class to ensure that the push vectors do positively span the plane.

    - Spanning-stables: Since the normals to the edges $\{\hat{\mathbf{n}}_{ij}\}$ positively span $\mathcal{R}^2$, the pose vector equation is always satisfied.

- Biplus-stables: Since the normal vectors of a biplus-stable form a cone, when the desired reorientation is exactly 180°, the unit push vectors sets generated for the CW and CCW reorientations both positively span the plane.
- Uni-stables: For uni-stables, $\hat{\mathbf{n}}_{is} = \hat{\mathbf{p}}_i$ (subscript s refers to the single stable edge), and the pose vector equation can be replaced by
  $a_{0s}\hat{\mathbf{n}}_{0s} + \sum_{i=1}^{r}(a_{is} + b_i)\hat{\mathbf{p}}_i + \sum_{i=1}^{r}\vec{\mathbf{c}}_i = \vec{\mathbf{T}}$.
  The above vector equation is equivalent to
  $a_{0s}\hat{\mathbf{n}}_{0s} + \sum_{i=1}^{r}(a_{is} + e_i)\hat{\mathbf{p}}_i = \vec{\mathbf{T''}}$,
  where $\vec{\mathbf{T''}} = \vec{\mathbf{T}} - \sum_{i=1}^{r}\vec{\mathbf{c}}_i - \sum_{i=1}^{r} m_i\hat{\mathbf{p}}_i$.
  A singularity occurs when the reorientation is 180°. Since neither the CW nor CCW unit push vectors positively span the plane, a solution may not exist. By making an additional reorient push in the rotation direction opposite to the remaining reorient pushes, we obtain a set of unit push vectors that positively span the plane and guarantee existence of a feasible solution. See Figure 2.9 for an example.

We have thus shown that a set of unit push vectors that positively span the plane and satisfy the pose vector equation always exists. (Note that by rotating a part through an additional 360 degrees, we can trivially guarantee a set of pushes that positively span the plane.) A sequence of pushes to move any polygon from any start pose to any goal pose always exists, and the set of linear normal pushes is complete.

2.4 The Pose Planner

The Pose Planner must generate a set of pushes that move an arbitrary part from an arbitrary start pose to an arbitrary goal pose. The planner attempts to generate plans involving CW rotation of the part, CCW rotation of the part, and when the start and goal orientations are identical, no rotation of the part. These are the *CW*, *CCW*, and *Translation* pose plans respectively. These plans involve CW reorient pushes and translation pushes, CCW reorient pushes and translation pushes, and only translation pushes respectively. For a CW or CCW plan, the intermediate orientations selected using the angle-eating heuristic determine the reorient and translation push directions. This information is used to set up the corresponding linear programming problem which is then solved; each feasible solution is a valid plan. The planner attempts to generate CW and CCW pose plans for each stable edge. The plan with the lowest value of the objective function among all the feasible plans is selected as the best pose plan. The organization of the planner is depicted in Figure 2.10.

When the required reorientation is 180° and the part is a uni-stable, sometimes neither the CW nor CCW plan is feasible. When such a singularity occurs, the planner generates modified CW and CCW reorient sequences. A modified reorient sequence consists of the usual reorient sequence preceded by a reorientation in the opposite direction. The amount of rotation in the opposite direction is chosen to be the minimum of the CW and CCW step angles for the stable pushing edge. So a *CW singularity plan* consists of a CCW reorient push preceding a sequence of CW reorient pushes; similarly for the *CCW singularity plan*. Since these pushes positively span the plane, there is a guaranteed solution for each main rotation direction and the one that minimizes the objective function is selected. An example singularity plan is shown in Figure 2.11.

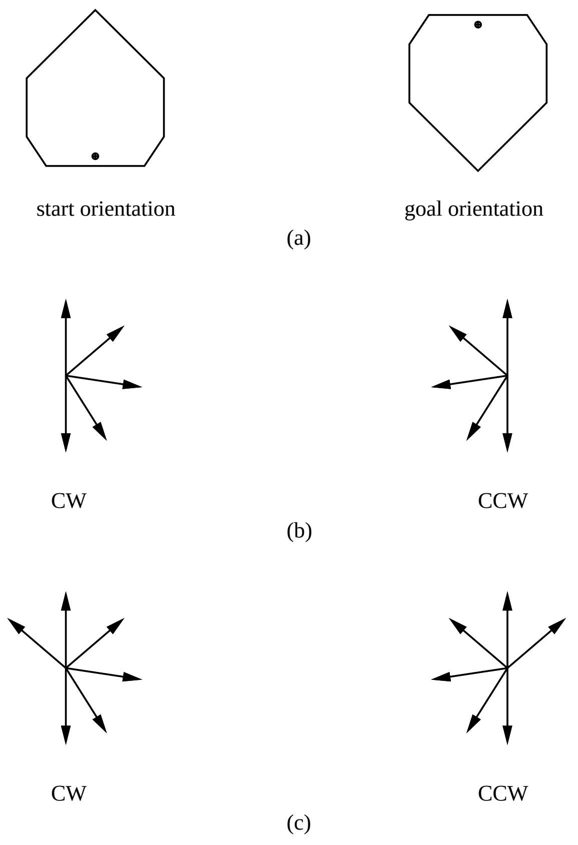


Figure 2.9: A singularity case. (a) Start and goal orientations 180° apart. (b) CW and CCW push vector diagrams: The push vectors do not positively span the plane. (c) CW and CCW push vector diagrams: The push vectors augmented by an additional reorient push in the opposite rotation direction positively span the plane.

2.4.1 Planner Completeness

We now show that the Pose Planner is complete, that is, that it can find a pose plan for an arbitrary pose problem for an arbitrary polygonal part. A proof of completeness is equivalent to a proof that for each part class, at least one sequence of translation and reorient pushes generated by the planner will satisfy the pose vector equation and the nonnegativity and Peshkin constraints. That is, the planner is complete if for each part class it generates at least one set of pushes whose unit vectors positively span the plane. As shown in Section 2.3.3, when the required reorientation is not 180°, the unit push vectors generated using the angle-eating heuristic positively span the plane in at least one of the reorient directions. When the reorientation is 180°, the unit push vectors for spanning-stables and biplus-stables positively span the plane. For uni-stables, the procedure to handle singularities guarantees two sets of unit vectors that positively span the plane when required for a 180° reorientation. So the Pose Planner is complete.

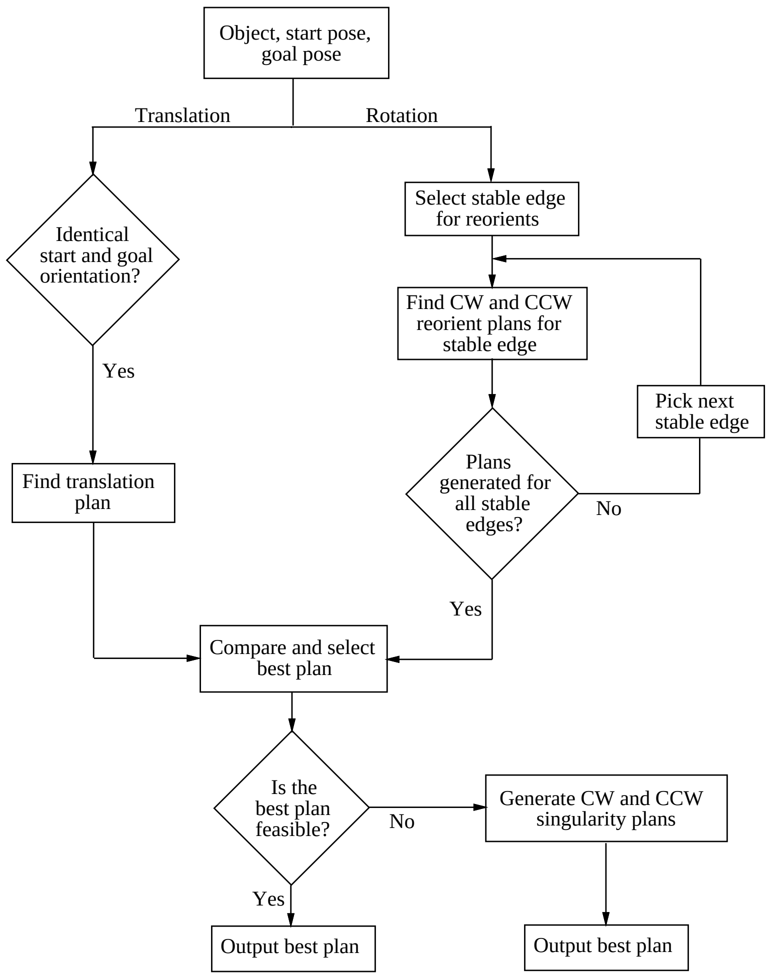

Object, start pose, goal pose
Translation
Rotation
Identical start and goal orientation?
Yes
Find translation plan
Select stable edge for reorients
Find CW and CCW reorient plans for stable edge
Pick next stable edge
Plans generated for all stable edges?
No
Yes
Compare and select best plan
Is the best plan feasible?
No
Yes
Generate CW and CCW singularity plans
Output best plan
Output best plan

Figure 2.10: A flowchart describing the organization of the Pose Planner. Given a part description and the start and goal poses, it outputs the best plan as defined by the objective function.

2.4.2 Planner Complexity

The planner has polynomial time complexity. For a part with n edges, the maximum reorient angles can be computed in $O(n)$ time. The LP formulation has $s(r+1)+r$ variables, two equality constraints, and $s(r+1)+r$ push magnitude constraints, where s is the number of stable edges and r is the number of reorients. This LP problem can be solved by the simplex method in $O(s^3r^3)$ time. The planner solves an LP problem for each stable edge for each rotation direction. There are $O(s)$ such LP problems to be solved, and their solutions can be compared in $O(s)$ time. If the maximum number of reorients is r_{max}, the total running time of the planner is bounded by $O(s^4r_{max}^3)$.

The dual of our LP problem has two variables and $s(r+1)+r$ constraints. It can therefore be solved by the simplex method in $O(s^2r^s)$ time, which means the total running time of a planner that uses the dual formulation is $O(s^3r_{max}^2)$.

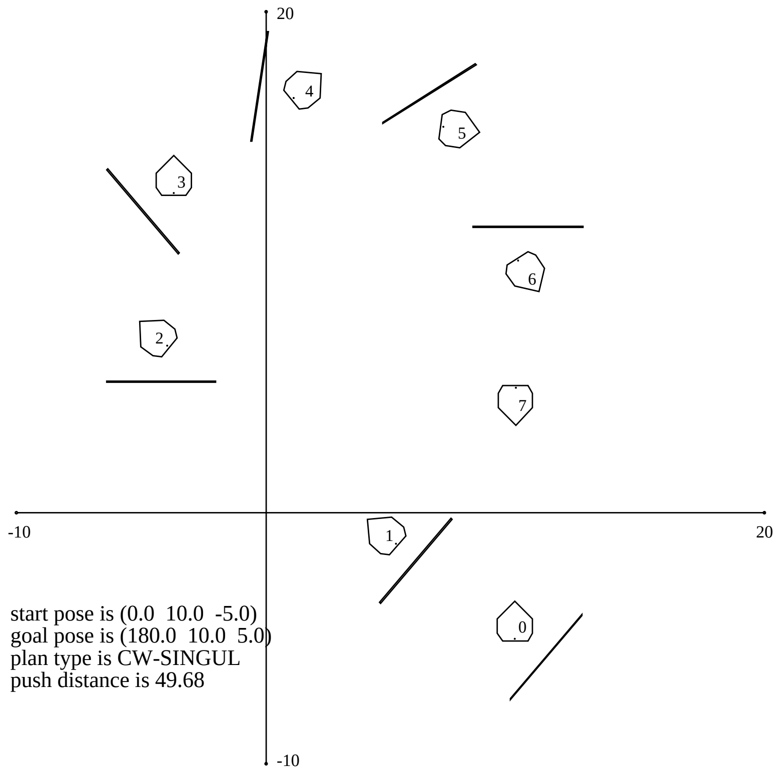


Figure 2.11: A plan to handle a uni-stable singularity case.

2.5 Minimizing Plan Execution Time

We have also developed a linear mixed-integer programming (MIP) formulation to minimize any linear function of the push distance and the number of pushes, and have implemented this formulation in our planner. (See Nemhauser and Wolsey [132] for an introduction to integer programming.) The MIP formulation shares the completeness property of the LP formulation.

We illustrate the MIP formulation by minimizing the time taken to execute a plan. Let the time taken to push a unit distance be k_d and the setup time for each push be k_p. The total number of pushes is the sum of the number of non-zero translation pushes and the number of reorient pushes. So the MIP formulation to minimize the execution time of a plan can be written as:

$$\begin{array}{ll}
\text{Minimize} & k_d\left(\sum_{i=0}^{r}\sum_{j\in S} a_{ij} + \sum_{i=1}^{r} b_i\right) + k_p\left(\sum_{i=0}^{r}\sum_{j\in S}\alpha_{ij} + r\right) \\
\text{subject to} & \sum_{i=0}^{r}\sum_{j\in S} a_{ij}(\hat{\mathbf{n}}_{ij}\cdot\hat{\mathbf{t}}') + \sum_{i=1}^{r} b_i(\hat{\mathbf{p}}_i\cdot\hat{\mathbf{t}}') = T' \\
& \sum_{i=0}^{r}\sum_{j\in S} a_{ij}(\hat{\mathbf{n}}_{ij}\cdot\hat{\mathbf{t}}'_{\perp}) + \sum_{i=1}^{r} b_i(\hat{\mathbf{p}}_i\cdot\hat{\mathbf{t}}'_{\perp}) = 0 \\
& a_{ij} \geq 0 \quad \text{for } i = 0,\ldots,r \text{ and } j \in S \\
& b_i \geq m_i \quad \text{for } i = 1,\ldots,r \\
& M\alpha_{ij} \geq a_{ij} \quad \text{for } i = 0,\ldots,r \text{ and } j \in S \\
& M a_{ij} \geq \alpha_{ij} \quad \text{for } i = 0,\ldots,r \text{ and } j \in S \\
& \alpha_{ij} \in \{0,1\} \quad \text{for } i = 0,\ldots,r \text{ and } j \in S
\end{array}$$

where M is a large positive number and the α_{ij} terms are binary variables for the corresponding translation pushes. The resulting plans minimize the time taken to execute a plan. Figure 2.12 depicts a plan that minimizes the execution time for the same start and goal poses as in Figure 2.1.

Using the MIP formulation, the planner can also generate plans that minimize the number of pushes; a second attribute such as the push distance is sometimes necessary to select the best plan. We can show that no more than two translation pushes are required in any plan that minimizes the number of pushes, the push distance, or a nonnegative linear combination of the number of pushes and the push distance. This property leads to a polynomial-time algorithm to generate such plans.

2.6 Implementation and Experiments

The Pose Planner was implemented in Common Lisp. The inputs to the planner are the start and goal poses, and a geometric description of the part, in terms of its vertices and center of mass. It uses the code to generate the push-stability diagram described by Brost [29], and a commercial linear programming package, LINDO (Schrage [162]). The planner takes between 1 and 12 seconds on a SPARCstation IPX to find the best plan for the example parts in this chapter.

When executing reorient pushes, using a step angle close to the maximum reorient angle can result in large Peshkin distances and sometimes, failure of the reorient push due to orientation errors; a reorient push is a failure when it does not result in the stable edge lining up with the

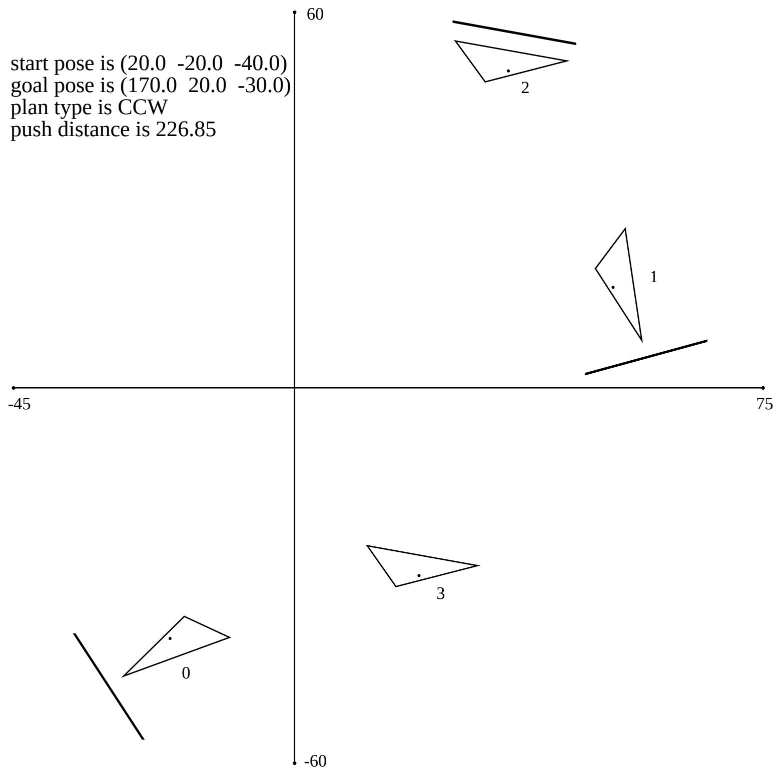

start pose is (20.0 -20.0 -40.0)
goal pose is (170.0 20.0 -30.0)
plan type is CCW
push distance is 226.85
60
-60
-45
75
0
1
2
3

Figure 2.12: A plan that minimizes the execution time. The push setup time k_p is 3 secs, and for a fence speed of 40 mm/sec, k_d is 0.025 secs/mm.

fence. To avoid this, we choose the step angle to be less than the maximum reorient angle by a *margin angle*. We also use a *push offset*, which is the offset distance of the fence from the part at the start of a push. We use a margin angle of 7° and a push offset of 30 mm to handle bounded uncertainty in the orientation and position of the part. The calculated Peshkin distances were sometimes observed to be insufficient to achieve reorientation, possibly due to inexact estimates of the center of mass location. They were therefore multiplied by a safety factor of 1.15.

With the above values of the margin angle and push offset, six plans were executed on a Puma 560 robot for a total of over 80 trials. The Puma, with a fence attached to its end-effector, executes pushing actions on a horizontal table. The fence is a piece of delrin coated with high friction sandpaper. Parts belonging to the three classes, including those in Figure 2.4, were constructed of delrin. They were made fairly small, about 15 to 30 mm in diameter, for the plans to be inside the Puma workspace. The plans always succeeded in bringing the part to the goal orientation. Position errors of 2 to 3 mm over a translation distance of 150 to 200 mm were however observed. These position errors were sometimes as large as 5 mm. The observed errors were a result of slip during reorients, motion of the part along the fence due to robot vibration, and the positioning inaccuracy of the Puma.

2.7 Conclusion

In this chapter we described the use of linear normal pushes to position and orient parts in the plane. We introduced a classification of polygonal parts subject to linear normal pushes, and used it to prove that the set of linear normal pushes is complete for the planar pose problem. Using a linear programming formulation, we have implemented a polynomial-time planner for this problem and proved its completeness. Sample plans executed by a robot demonstrate the feasibility of this method.

An important aspect of our approach is the use of a compact set of actions for the task, coupled with an efficient search mechanism. An alternative approach is to use a more extensive set of actions, such as non-normal linear pushes, or rotational pushes as in Lynch and Mason [116]. The selection of an action set should depend on the task domain; an environment with a mobile robot equipped to turn on the spot and move in straight lines is conducive to the use of linear pushes while one with a car-like mobile robot can better utilize rotational pushes.

The major directions for future work are to increase the robustness of the plans to uncertainty and to extend the scope of the planner. To generate robust plans, it is important to model uncertainty in the initial pose of the part and in the pushing actions. Plans using the actions described here are sensitive to uncertainty since a single push can eliminate rotational uncertainty, but not pose uncertainty of a part. A combination of pushes and grasps with a parallel jaw gripper, the use of a specially-shaped fence (Brost [31]), or a sequence of centering operations (Mottaez and Goldberg [128]) are ways to reduce the pose uncertainty of the part. We have made the analysis tractable by separating part orientation from part position to the extent possible. When treating pose uncertainty, it may be necessary to consider the orientation and position together. Determining if "nearby" start poses behave similarly to a given plan could be useful in obtaining bounds on the permissible pose uncertainty for the plan.

When generating each plan, our planner selects a single stable edge to perform reorient pushes on and uses the angle-eating heuristic to select intermediate orientations. The best plan found with these conditions is not guaranteed to be globally optimal. For global optimality, we would have to extend our formulation in two directions. First, we must allow the intermediate orientations to be variables whose values are determined during the optimization process. This changes some of the linear constraints to nonlinear constraints. Second, we must permit reorient pushes to be performed on any of the stable edges of the part during a plan. This increases the search space since the planner has to enumerate all possible sequences of edges to perform reorient pushes on.

An important issue is to see if plans can be generated in the presence of obstacles. Since the linear programming formulation can have multiple solutions, we can look for solutions that do not violate the obstacle constraints. Enlarging the set of actions to include non-normal linear pushes may make it possible to find plans in the presence of obstacles.

It would be interesting to consider extensions of our method to handle more general part shapes and to determine if multiple parts can be posed simultaneously.

Chapter 3

Parts Transfer on a Conveyor with a One Joint Robot

The most straightforward approach to planar parts transfer is to use a rigid grasp and a robot with at least three joints, corresponding to the three motion freedoms of a planar rigid object. Alternatively we can use a robot with at least three joints to transfer a part by pushing, as in the previous chapter. But three joints are not really necessary to manipulate an object in the plane. This chapter[1] explores a method of manipulating a planar rigid body on a conveyor belt using a robot with just one joint. We demonstrate effective control of all three planar motion freedoms using a one joint robot working over a constant speed conveyor belt.

A central issue in this work is to develop a precise notion of "effective control" that is suited to the parts-feeding application. Our robot cannot impart arbitrary motions to a part on the conveyor, but it does have a set of actions sufficient to orient and position a wide class of shapes. To make this precise we define the *feeding property*:

> A system has the *feeding property* over a set of parts $\mathcal{P}$ and set of initial configurations $\mathcal{I}$ if, given any part in $\mathcal{P}$, there is some output configuration $\mathbf{q}$ such that the system can move the part to $\mathbf{q}$ from any location in $\mathcal{I}$.

One of the goals of this chapter is to demonstrate that a one joint robot can possess this property over a useful set of initial configurations and a broad class of part shapes.

The key to our approach is to use a single revolute joint to push the parts around on a constant speed conveyor belt. This approach, which we refer to as "1JOC" (one-joint-over-conveyor, pronounced "one jock") was initially conceived as a variation on the Adept Flex Feeder (see Figure 3.1). The Flex Feeder uses a system of conveyors to recirculate parts, presenting them with random orientation to a camera and robotic manipulator. Those parts that are in a graspable configuration may then be picked up by the robot and assembled into a product, placed in a pallet, or otherwise processed.

The question addressed in this chapter is whether, at least in parts feeding applications, we could replace the SCARA robot with a simpler and more flexible robot, and also replace the Flex Feeder's servoed programmable conveyor with a fixed speed conveyor. Figures 3.2 and 3.3 show a possible variation on the Flex Feeder, where the SCARA robot has been replaced by a fence with a

[1]This chapter represents joint work with Wesley H. Huang, Kevin M. Lynch, and Matthew T. Mason.

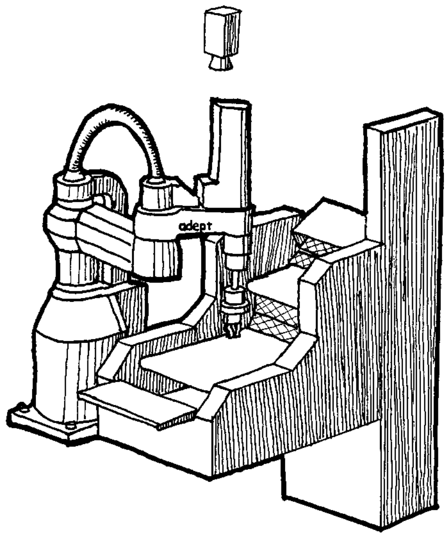


Figure 3.1: The Adept Flex Feeder System. A SCARA robot picks parts off the middle of three conveyors. These three conveyors, along with an elevator bucket, circulate parts; an overhead camera looks down on the back-lit middle conveyor to determine the position and orientation of parts.

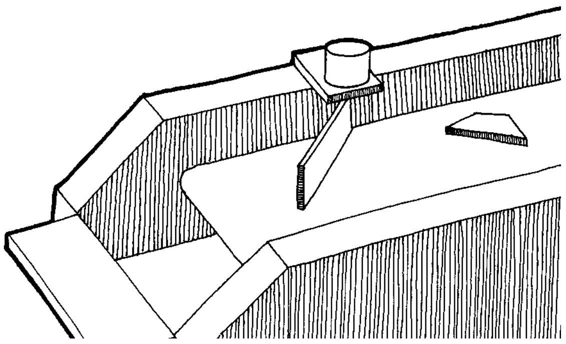


Figure 3.2: The Flex Feeder with a rotatable fence.

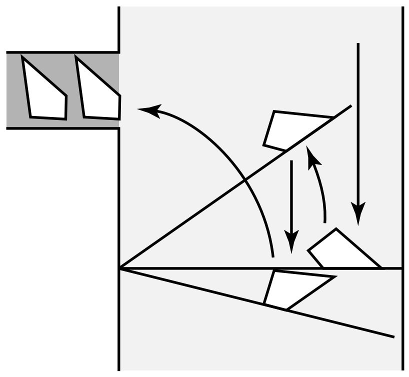


Figure 3.3: We can feed a part by alternately pushing it with the fence and letting it drift along the conveyor.

single revolute joint. By a sequence of pushing operations, punctuated by drift along the conveyor, the fence positions and orients a part and directs it into the entry point of a feeder track which carries the part to the next station.

There are many variations on this basic idea: a 2JOC, multiple 1JOCs working in parallel, curved fences, and so on. However, the purpose of this chapter is an initial study of the fundamental characteristics of the idea, so we will focus on the simplest version as described above and in Figures 3.2 and 3.3. The main result is:

> It is possible to move an arbitrary polygon from a broad range of initial configurations to a specific goal; and that goal can be chosen from a broad range of possible goals. (This is a generalization of the feeding property.)

The remainder of the chapter is organized as follows. After reviewing terminology from nonlinear control theory, we develop a progression of models leading to the 1JOC. We prove the feeding property for the 1JOC model and describe the implementation of a planner as well as experimental results.

3.0.1 Terminology

The 1JOC has only one controlled degree-of-freedom, yet we would like to control three degrees-of-freedom of a part on the conveyor — the system is underactuated. The natural question is whether the 1JOC is "rich" enough to manipulate parts, or whether the two constraints on the motion of the fence (pivot is fixed) overly constrain the set of reachable configurations of the part.

To be precise, we borrow definitions of reachable sets from nonlinear control theory (Sussmann [173]). A part's configuration $\mathbf{q} = (x, y, \phi)^T$ is *controllable from* $\mathbf{q}$ if, starting from $\mathbf{q}$, the part can reach every configuration in the configuration space. The part is *controllable to* $\mathbf{q}$ if $\mathbf{q}$ is reachable from every configuration. The part is *accessible from* $\mathbf{q}$ if the set of configurations reachable from $\mathbf{q}$ has nonempty interior in the configuration space.

We can also define local versions of these properties. The part is *small-time accessible from* $\mathbf{q}$ if, for any neighborhood U of $\mathbf{q}$, the set of reachable configurations without leaving U has nonempty interior. The part is *small-time locally controllable from* $\mathbf{q}$ if, for any neighborhood U of $\mathbf{q}$, the set of reachable configurations without leaving U contains a neighborhood of $\mathbf{q}$.

The phrases "from **q**" and "to **q**" can be eliminated in these definitions if they apply to the entire configuration space.

In these terms, if the configuration of the part is accessible, we know that the 1JOC has "enough" degrees of freedom: it can transfer the part from the initial configuration to a three-dimensional subset of the configuration space. Better yet, if the part is controllable, the 1JOC can transfer the part from any configuration to any other configuration. Small-time local controllability, an even stronger condition, implies that the part can follow any path arbitrarily closely. This is a useful property for repositioning parts under time and workspace constraints.

Although controllability and small-time local controllability guarantee the ability to feed parts, they are not absolutely necessary. The minimum we require of a parts feeding system is the feeding property: the parts must be controllable to a single configuration from the set of initial configurations.

One way to reason about the controllability of a system is to write it as a system of drift and control vector fields and look at the dimension of the distribution defined by these vector fields. We can do this for an idealized model of the 1JOC (Section 3.1); however, this is difficult to do with the full pushing mechanics because the motion of a pushed object has no closed form. In addition, the motion of the part depends on the distribution of support forces between the part and the conveyor, which is usually unknown. The precise motion of the part is therefore unpredictable.

When we consider the full pushing mechanics in Section 3.2, we will limit our study to pushing motions with predictable outcomes. Despite this limited set of actions, we find that the 1JOC possesses the feeding property.

3.1 A Progression of Models

The 1JOC approach arises from a desire to explore the simplest mechanisms to accomplish a task. Planar objects have three degrees of freedom, suggesting the use of a robot with three or more actuated joints. However, it seems that two joints should suffice, by analogy with planar mobile robots. It is well known that a car can be arbitrarily positioned in the plane even if the steering wheel has only two settings. (For an introduction to motion planning for cars, and nonholonomic motion planning in general, see Latombe's text [105].) Each setting of the steering wheel defines a rotation center of the car. In the case of the car these two rotation centers move with the car. To see that two *fixed* rotation centers suffice to produce arbitrary motions of the car, imagine that the car is fixed and the wheels drive the plane around.

Obviously a planar object cannot be moved to an arbitrary position using a single fixed rotation center. Thus one might conclude that two controlled freedoms are necessary. However, a drift field, plus a single controlled freedom, suffices.

The problem of pushing a part across a moving conveyor is complicated by the mechanics of pushing. Some insights can be gained by first considering an idealized model: the part matches the fence's motion whenever desired, and matches the conveyor's motion otherwise. In this section we examine a progression of models leading to the 1JOC system.

One Rotation Center First we consider an idealization of a rotating fence. We envision an infinite turntable to which the part can be affixed. This turntable can give the part an arbitrary angular velocity about its pivot.

From any initial point in the part configuration space, the reachable set is one dimensional.

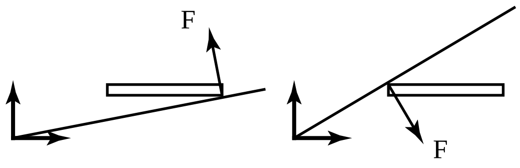


Figure 3.4: The thin rod is not small-time locally controllable from this configuration — it can only be rotated with a positive angular velocity.

If we consider the full pushing mechanics for a rotating fence, the reachable set is also just one dimensional. If the fence can swing all the way around and push on the other side, however, the part's configuration may be accessible. But we will not consider this case further.

Two Rotation Centers Now consider a pair of overlapping (yet independent) ideal turntables. At any given instant, the part is affixed to one of the two turntables and rotates about the center of that turntable. We assume that we can instantaneously switch the part from one turntable to the other.

There are two cases to consider:

- Both turntables are bidirectional. If each turntable can be driven in either forward or reverse, the situation is similar to driving a car with two different steering angles. As is well known, this system is small-time locally controllable and can approximate an arbitrary trajectory as closely as desired.

- One or both turntables are unidirectional. Now the situation is analogous to a car without reverse. The system is controllable, but not small-time locally controllable. It may be necessary to take a large excursion to accomplish a small motion.

We can model the conveyor as the limiting case of a turntable whose pivot approaches infinity. As above, the system is small-time locally controllable if both the turntable and conveyor are bidirectional, and it is simply controllable if either or both is unidirectional.

For this simple system, we can study controllability by considering the vector fields $X_1 = (0,1,0)^T$ and $X_2 = (-y,x,1)^T$ corresponding to the conveyor and the turntable, respectively. When the part tracks the conveyor, the part motion is given by $\dot{\mathbf{q}} = vX_1$, where v is the velocity of the conveyor in the y direction. When the part tracks the turntable, the part motion is given by $\dot{\mathbf{q}} = \omega X_2$, where ω is the angular velocity of the turntable and the origin is at the center of the turntable. The Lie bracket $[X_1, X_2]$ of these two vector fields is given by:

$$[X_1, X_2] = \frac{\partial X_2}{\partial \mathbf{q}} X_1 - \frac{\partial X_1}{\partial \mathbf{q}} X_2 = (-1,0,0)^T$$

The new vector field $[X_1, X_2]$ is linearly independent of X_1 and X_2, yielding a third controlled freedom (provided both v and ω can be nonzero). The system therefore satisfies the Lie Algebra Rank Condition, and it is small-time accessible. If both the conveyor and the turntable are bidirectional (both v and ω can be positive or negative), the system is also small-time locally controllable (Nijmeijer and van der Schaft [136]). If either or both is unidirectional, the system is not small-time locally controllable, but it can be shown to be controllable by a simple constructive argument (Brockett [26]; Lynch and Mason [114]).

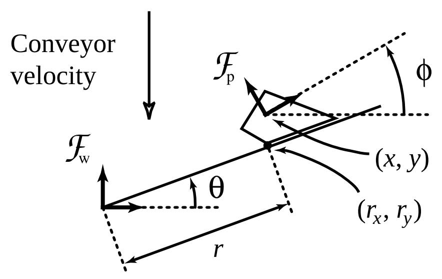


Figure 3.5: Notation.

The 1JOC system consisting of a conveyor and a rotating fence resembles the system above with a unidirectional conveyor. One difference is that rotating the part about the pivot is only stable on a subset of the configuration space, specifically when an edge of the part is aligned with the fence.

Detailed Pushing Model We now consider pushing mechanics in the model with a conveyor and a rotating fence. We assume that the friction coefficient at the pushing contact and the distribution of support forces are known, and we place no restriction on the fence motions. Although this model makes unrealistic assumptions about the available information, any negative results will also apply to less detailed models.

Figure 3.4 shows that the system is not small-time locally controllable for the case of a thin rod. If the fence pushes from below, the rod's angular velocity is positive. If the fence swings around and pushes the part from above, the angular velocity is still positive. There is no maneuver combining these actions and conveyor drift that can locally achieve negative rotations.

Practical Pushing Models The detailed pushing model is difficult to use because it requires full knowledge of the pressure distribution. We can define more abstract models by restricting the allowable actions. In particular, we would like to use actions with predictable outcomes that do not impose unrealistic demands for information.

Any abstraction of the detailed pushing model will inherit its limitations, thus it is immediately clear that small-time local controllability is impossible.

3.2 The Feeding Property

In this section we focus on a particular model for the 1JOC and prove that it can transfer any polygonal part from any valid input configuration to a feasible goal configuration. To prove this property, we utilize a subset of the actions available to the 1JOC and show that they are sufficient to demonstrate the feeding property.

3.2.1 The 1JOC Model

The fence is a line that pivots about a fixed point on the line. The origin of a fixed world frame $\mathcal{F}_w$ coincides with the pivot point. The conveyor is the half-plane $x > 0$, with a constant drift velocity v in the $-y$ direction. The fence angle θ is measured with respect to the x axis of $\mathcal{F}_w$, and its angular velocity is given by ω. The fence angular velocity ω is our single control input.

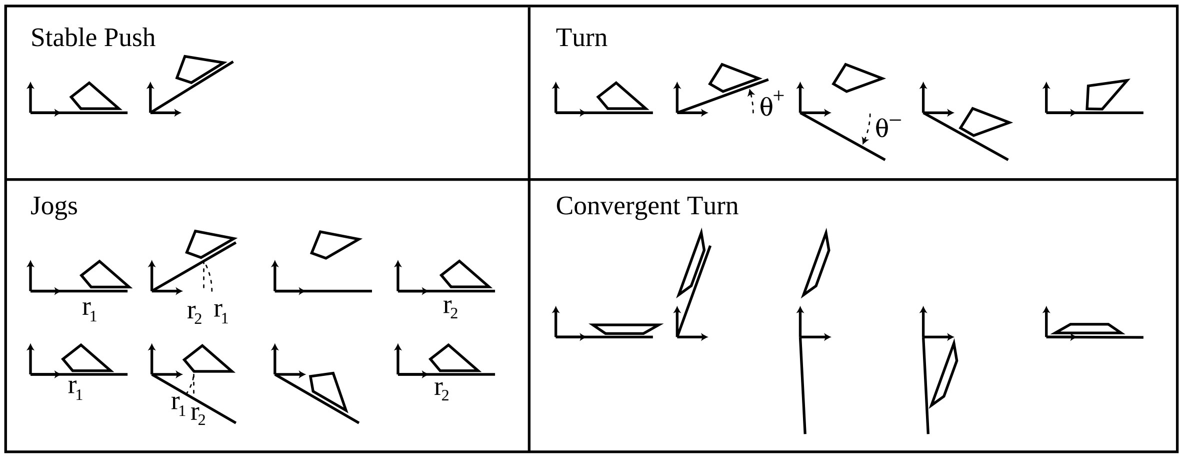


Figure 3.6: Step by step illustrations of the 1JOC primitives.

The part can be any polygon, but because the manipulator is a line, the part can always be treated as its convex hull. The center of mass of the part lies in the interior of the convex hull and the coefficient of friction between the fence and the part is nonzero. When we refer to a "part" in the rest of the chapter, we assume these properties.

We assume quasistatic mechanics. As the fence pushes the part over the conveyor, the motion of the part in the world frame $\mathcal{F}_w$ is sufficiently slow that inertial forces are negligible compared to the frictional forces. The support friction acting on the part during pushing is determined by the motion of the part relative to the conveyor, not the world frame $\mathcal{F}_w$.

It is convenient to represent the linear and angular velocity of the part relative to the conveyor in a frame $\mathcal{F}_p$ attached to the center of mass of the part. A velocity direction is simply a unit velocity vector. Velocity directions may also be represented as rotation centers in the frame $\mathcal{F}_p$.

The configuration of the part frame $\mathcal{F}_p$ in the world frame $\mathcal{F}_w$ is given by $(x, y, \phi) \in \mathcal{R}^2 \times \mathcal{S}^1$. When an edge of the part is aligned with the fence, the contact radius r defines the distance from the fence's pivot point to the closest point on the edge (the *contact vertex*), and the configuration of the part can be given in the polar coordinates (r, θ). The location of the point at r on the fence is (r_x, r_y) in the world frame. See Figure 3.5.

3.2.2 1JOC Primitives

We now describe the basic 1JOC primitives which yield the feeding property. We assume the fence is initially held at 0 degrees, perpendicular to the conveyor velocity, until the part contacts and comes to rest against the fence. See Figure 3.6 for an illustration of the primitives.

Stable Pushes

A stable push occurs when one edge of the part is aligned with the fence and the motion of the fence keeps the part fixed against it. The simplest example of a stable push occurs when the fence is held at 0 degrees, and a part on the conveyor drifts into contact and comes to rest on the fence. When the part is at rest, the fence is executing a stable push. The pushing direction, as seen by the part, is opposite the conveyor's velocity direction.

Of course, a stable push may also occur while the fence is rotating. Lynch and Mason [114] describe the procedure STABLE that finds a set of stable pushing directions $\mathcal{V}_{stable}$ for a given pushing edge, pushing friction coefficient, and center of mass of the part. This set of pushing directions is fixed in the part frame $\mathcal{F}_p$. The fence is guaranteed to execute a stable push if the edge is aligned with the fence and the motion of the fence and the conveyor combine to yield a pushing direction in $\mathcal{V}_{stable}$. See Figure 3.7 for details.

For simplicity, we will only consider the edges of the part that yield a stable push while the fence is held at 0 degrees (the pushing direction is a translation normal to the edge). The existence of these stable edges is indicated by the following lemma (Lynch and Mason [114]):

Lemma 3.1 *All polygonal parts have at least one edge such that the normal translational pushing direction, along with a neighborhood of this pushing direction, belongs to $\mathcal{V}_{stable}$. Such edges are called* stable edges.

Remark: The object may also have *metastable edges* — edges such that the center of mass lies directly above an edge endpoint when the fence is held at 0 degrees. If the fence rotates any nonzero amount in the "wrong" direction, the edge will become unstable. We will avoid these edges in this chapter. If the part initially comes to rest on the fence on one of these edges, we can perturb the fence slightly to bring the part to a stable edge.

Lemma 3.1 says that a stable edge is also stable for a neighborhood of fixed fence angles around 0 degrees. In addition, if the contact radius r is sufficiently large, the fence pivot will be inside the set of stable rotation centers $\mathcal{V}_{stable}$ fixed in the part frame $\mathcal{F}_p$. If the fence angular velocity ω is large enough relative to the conveyor velocity v, the combined pushing motion is nearly a pure rotation about the pivot, and the pushing motion is stable. Therefore, it is possible to stably push the part from any θ_0 to any θ_1 in a counterclockwise (CCW) direction, where $90° > \theta_1 > \theta_0 > -90°$. Figure 3.8 illustrates a method for finding the *minimum stable radius*: the minimum contact radius r such that the push is stable for all fence angles in the range $[-90, 90]$ for a given fence angular velocity ω and conveyor velocity v.

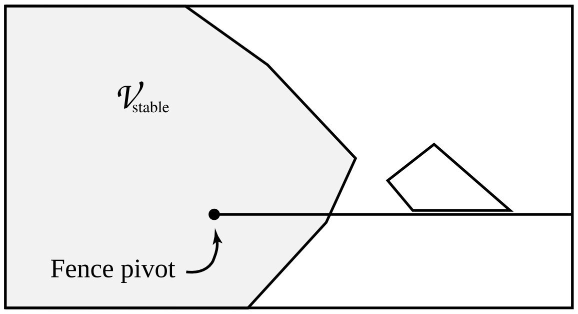
For a push to be stable when the fence is rotating CCW, the contact radius r must be sufficiently large. We can change r by performing jogs, described below.

Jogs

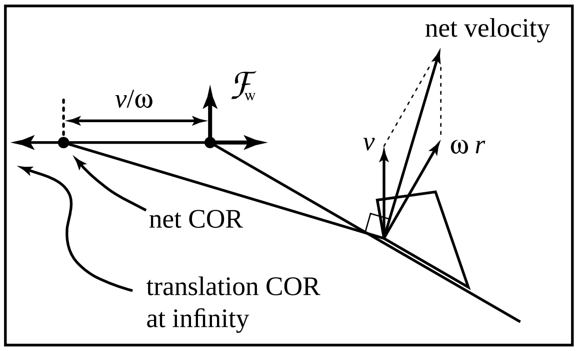
Jogs are simple maneuvers that allow us to change the contact radius r from the fence pivot to the part without changing the stable resting edge.

Lemma 3.2 *A part resting on a stable edge can be moved from any initial contact radius r on the fence to any desired r in the range $(0, \infty)$ by a series of jogs.*

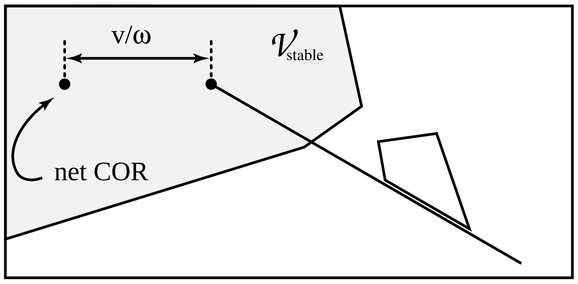
Proof: We use the fact that a neighborhood of pushing directions about the normal translation is stable for a stable edge. In fact, there is a range $[\theta^{min}, \theta^{max}]$ of fixed fence angles about 0 degrees that are also stable. The fence does not have to be motionless to execute a stable push, however; if the fence angular velocity ω is small enough with respect to the conveyor velocity v, then the fence can execute a stable push while moving in the angle range $[\theta^{min}, \theta^{max}]$, regardless of the contact radius r.



For a given edge contact, we can determine $\mathcal{V}_{stable}$, the set of pushing rotation centers that keep the object fixed to the pusher. Note that $\mathcal{V}_{stable}$ is fixed in the part frame, not the world frame.



The conveyor velocity v and the rotational velocity ω about the fence pivot combine to form a net velocity at each point. This can be expressed as a net center of rotation (COR), which lies on the line through the fence pivot and perpendicular to the conveyor velocity.



Provided the net COR is contained in $\mathcal{V}_{stable}$, the part will remain stationary with respect to the fence for that combination of conveyor and fence velocities. Note that as the fence rotates, the conveyor velocity changes direction with respect to the part, and the net COR rotates about the fence pivot at a radius of v/ω.

Figure 3.7: Computing the COR region for stable pushes on a conveyor.

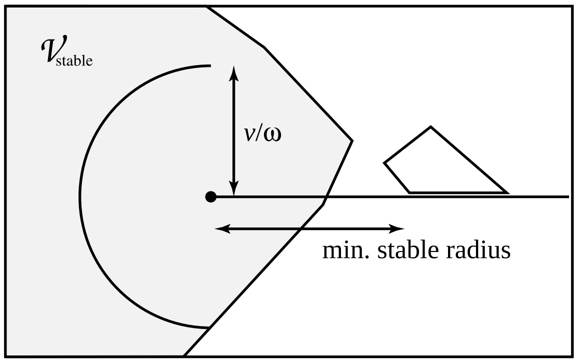


Figure 3.8: The locus of net COR form a semicircle about the fence pivot as the fence swings from -90 to 90 degrees. Given a fence angular velocity ω and a conveyor velocity v (which determine the radius of this semicircle), we pick a minimum stable radius so that this semicircle lies completely within $\mathcal{V}_{stable}$. Recall that $\mathcal{V}_{stable}$ is fixed with respect to the part, so increasing the contact radius moves the semicircle deeper into $\mathcal{V}_{stable}$. This conservative estimate of the minimum stable radius guarantees stable pushing in the counterclockwise direction at any fence angle in $[-90°, 90°]$.

To decrease the contact radius r, the fence is raised to an angle θ^+ ($\theta^{max} \geq \theta^+ > 0$), keeping the part fixed to the fence by a stable push. This changes the value of r_x to $r_x \cos\theta^+$. The fence then drops to 0 degrees, immediately releasing the part. The part drifts back into contact with the fence and settles on the same stable edge. We assume there is no slip between the part and the fence. (The no-slip assumption is important for open-loop feeding plans.) The result of the jog is to decrease the contact radius from r to $r \cos\theta^+$.

To increase the contact radius r, the fence is quickly lowered to an angle θ^- ($\theta^{min} \leq \theta^- < 0$), releasing the part. The part drifts back into contact with the fence and settles on the same stable edge. The fence is then raised to 0 degrees again, pushing the part with a stable push. The contact radius of the part has changed to $r/\cos\theta^-$.

A series of jogs, either increasing or decreasing r, can bring the part to any r in the range $(0, \infty)$. $\square$

A jog is directly analogous to the Lie bracket motion for the ideal turntable plus conveyor system analyzed in Section 3.1. The difference from the ideal system is that the part only tracks the fence when it is in stable contact. Nonetheless, we can now show that the configuration of the part is accessible with just stable pushes, jogs, and the conveyor drift.

Theorem 3.3 *The configuration of a polygonal part in stable edge contact with a fence held at 0 degrees is small-time accessible using stable pushes, jogs, and conveyor drift.*

Proof: Given any neighborhood of the part configuration while it is in stable edge contact with the fence at 0 degrees, it is possible to jog the part a nonzero distance in both directions without leaving this neighborhood. The effect of the jog is to change the x coordinate of the part configuration, exactly like the Lie bracket motion of Section 3.1. The other two vector fields of Section 3.1 can be obtained directly by conveyor drift and stable pushes. $\square$

We call $\mathcal{I}$ the set of initial part configurations that drift to stable edge contact with the fence at 0 degrees. From a stable edge contact, Theorem 3.3 indicates that the configuration of the part is small-time accessible. Therefore, the configuration of the part is accessible from the set $\mathcal{I}$.

We still have not proven the feeding property. To get this property, we need the ability to turn a part to a new stable edge.

Turns

Turns allow us to change the edge of the part aligned with the fence, subject to the constraint that the initial and final edges both be stable edges. A turn consists of raising the fence to an angle $\theta^+ \geq 0$ by executing a stable push; dropping the fence to an angle $\theta^- \leq 0$, immediately releasing the part; and reacquiring the part on the new edge with a stable push, raising the fence back to 0 degrees. The final push commences at the moment the new edge contacts the fence.

Lemma 3.4 *Any polygonal part has at most one stable edge from which it is impossible to turn the part to the next CW stable edge.* Exception: *a part has two such stable edges if they are the only stable edges and they are parallel to each other.*

Proof: The fence can execute a stable push to any angle less than 90 degrees, and it can reacquire the part with a stable push at any angle greater than -90 degrees. This indicates that it is possible to reach any final stable edge less than 180 degrees CW of the initial edge. If the part has two or more stable edges, there can only be one stable edge from which the next CW stable edge is greater than 180 degrees removed. For "exception" parts, there are two stable edges which are exactly 180 degrees removed from each other. □

There are a few important points to make about turns:

- The contact radius r, for both the initial and final edges, must be large enough that the pushing directions always remain in $\mathcal{V}_{stable}$.
- The quantity $\theta^+ - \theta^-$ is determined by the part geometry, but the actual values of θ^+ and θ^- are not. To some extent, the contact radius r after the turn can be controlled by the choice of these values. If the angle difference between the initial and final edge is 90 degrees or more, r can be changed to any value in the range $(0, \infty)$ during the turn.
- This primitive requires precise knowledge of the conveyor's motion — the final stable push must begin precisely when the new edge contacts the fence.

Consider the case where turning a part to a new stable edge requires raising the fence to $90 - \delta$ and catching the part at $-90 + \epsilon$, where δ and ϵ are small positive values. It may be necessary to move the part to a large contact radius r before executing the turn to ensure that the part remains completely on the conveyor ($x > 0$) during the initial push.

Convergent turns

To perform a turn between two stable edges which are parallel, the fence must, in principle, be raised to 90 degrees, pushing the part off the conveyor. To turn such parts, we can introduce a convergent turn: raise the fence to $90 - \delta$, allow the part to drift down, and begin pushing it again at $-90 + \epsilon$. Assuming no slip at the pushing contact, the part will converge to the new stable edge as the fence is raised to 0 degrees. Convergent turns allow us to strengthen Lemma 3.4 to apply to all polygonal parts, with no exceptions. In this chapter, the sole function of convergent turns is to handle parts with only two stable edges, which are parallel.

3.2.3 The Feeding Property

Using the lemmas proven above, we can now demonstrate the feeding property for the 1JOC.

Theorem 3.5 *The 1JOC possesses the feeding property:*

- *for all polygonal parts*
- *for the three-dimensional space $\mathcal{I}$ of initial configurations such that the part initially comes to rest on a stable edge, completely on the conveyor, against the fence fixed at 0 degrees, and*
- *using only stable pushes, jogs, turns, convergent turns, and conveyor drift.*

Furthermore, the goal configuration can be chosen from a three-dimensional subset of the configuration space.

Proof: If the part has a stable edge such that it cannot be turned to a new stable edge, then this edge must be chosen as the goal edge. Otherwise, any stable edge can be chosen as the goal edge. By Lemma 3.4 (strengthened by the convergent turn of Section 3.2.2), this goal edge can be reached using turns. Once at the goal edge, the part can be jogged to any contact radius r in the range $(0, \infty)$ (Lemma 3.2). By Lemma 3.1, the part is stable against the fence for a range of fence angles θ. The goal edge therefore allows a two-dimensional set of final stable configurations of the part in the (r, θ) space. If the fence is quickly lowered to 90 degrees, releasing the part, this set drifts to a set of reachable configurations with nonempty interior in the world configuration space. Any configuration in this set can be chosen as the goal. □

If the application of the 1JOC is to stuff parts into a feeder track, then the fence can be rotated 90 degrees, pushing the part off the conveyor into the feeder track, instead of releasing it to continue on the conveyor.

3.3 A Feeding Planner

The feeding property means that there exists a sequence of jogs and turns to feed any polygonal part, assuming infinite fence length and an infinite conveyor half-plane. A 1JOC with finite fence and conveyor dimensions may not be able to feed some parts. For a given part however, we can always find a fence and conveyor with finite dimensions to feed it. We describe how to find a sequence of jogs and turns to feed a given part with a fence and conveyor of known dimensions in the minimum amount of time.

We assume that we know the shape and center of mass of the part, the coefficient of friction between the fence and part, the conveyor velocity, and the fence and conveyor dimensions. The coefficient of friction between the part and conveyor is assumed constant, and we pick the fence angular velocity to be one of three discrete values: ω_{stable} (for stable pushes), ω_{drop} (when lowering the fence), and zero. We assume the fence receives parts that are singulated.

A part drifts down the conveyor until it comes to rest against the fence. The feeding problem consists of getting the part from this start configuration to the goal configuration (see Figure 3.9), where the configurations are specified by the edge aligned with the fence, and the contact radius (distance from pivot to the closest vertex of the resting edge). We assume that the initial fence orientation is zero, and, for convenience, that the final fence orientation is zero as well.

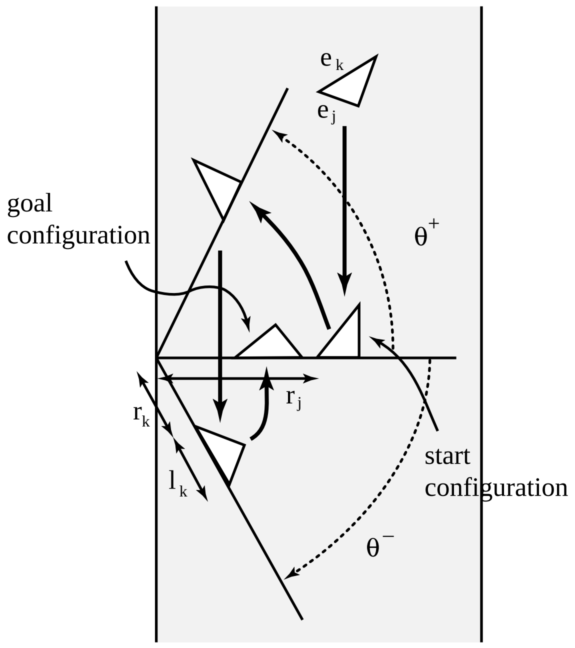


Figure 3.9: A feeding plan to move a triangle from start configuration (e_j, r_j) to goal configuration (e_k, r_k) by a turn with raise and drop angles of θ^+ and θ^-. Length of edge e_k is l_k.

Turns change the part orientation; they rotate the part CCW so the new edge aligned with the fence is CW to the initial edge. Turns thus enable transitions to stable edges in the CW direction. We consider all CW sequences of edges beginning at the start edge e_s and ending at the goal edge e_g which do not require rotating the part more than 360 degrees. Take an example sequence $S = \{e_s, e_j, e_k, e_g\}$. If we can find a valid sequence of jogs and turns to move the part in sequence through the edges in S and change the contact radius by the required amount, we have a feasible plan. Of all feasible plans, we take the one which requires minimum time.

3.3.1 A Simple Example

Consider a part resting on edge e_j (orientation ϕ_j) at radius r_j which is to be moved to the neighboring CW stable edge e_k (orientation ϕ_k) at radius r_k (see Figure 3.9). Here the sequence of edges is $S = \{e_j, e_k\}$. We look for a feasible solution (and ignore minimizing the time). Assume r_j and r_k are greater than the minimum stable radii for the corresponding edges, and are close enough in magnitude that no jog is required. Here, finding a plan consists of determining the fence angles θ^+ and θ^- for the turn. From the geometry, we have:

$$r_k = r_j \frac{\cos\theta^+}{\cos\theta^-} - l_k \tag{3.1}$$

$$\theta^+ - \theta^- = \phi_k - \phi_j \tag{3.2}$$

Solving these equations, we find:

$$\theta^- = \tan^{-1}\left(\cot(\phi_k - \phi_j) - \frac{r_k + l_k}{r_j \sin(\phi_k - \phi_j)}\right) \tag{3.3}$$

$$\theta^+ = \phi_k - \phi_j + \theta^- \tag{3.4}$$

This simple example has a closed form solution that provides us with a plan. (Note that when $(\phi_k - \phi_j) \geq 90$, the radius change can be arbitrarily large or small.)

3.3.2 Nonlinear Programming Approach

For the general case, we must find a sequence of jogs and turns that effects the desired configuration change and minimizes time. For a given sequence of edges, we must determine the parameters for each turn while ensuring that each contact edge in a turn has a contact radius that is no less than its minimum stable radius. If the contact radius is less than the minimum radius, we must translate the part prior to executing the turn. Such a *translation* may consist of a jog or a series of jogs.

A feeding plan to transfer a part from its start configuration (e_s, r_s) to its goal configuration (e_g, r_g) consists of several stages, each of which accomplishes an edge transition. Each stage consists of a translation followed by a turn; the translation for a given stage may be zero. A plan with the edge transition sequence S will take $n-1$ stages, where $n = |S|$. A final translation may also be required after the goal edge has been reached.

Consider the ith stage in a plan (see Figure 3.10) to accomplish the transition from edge e_j to edge e_k. First, we perform a translation of g_i, which moves the part to a contact radius of r_i^+. Since we are about to perform a turn, this radius must be greater than the minimum stable radius for edge e_j, r_j^{stable}. We then perform a turn with a raise angle of θ_i^+ and a drop angle of θ_i^-. The stable push at the end of the turn brings the fence to $\theta = 0$ with the new contact edge e_k. For this stable push on edge e_k, the contact radius r_i^- must be greater than the minimum stable radius for e_k, r_k^{stable}. We also have to ensure that the fence raise and drop angles are in valid ranges, that the part is within fence and conveyor bounds, and that the fence contacts the part only at the end of the drift phase.

Assume the fence pivot is on the conveyor left edge, and that the conveyor half-length exceeds its width, which exceeds the fence length. The resulting constraints are (for $i = 1, \ldots, n-1$):

$$r_i^+ = r_{i-1}^- + g_i \tag{3.5}$$

$$r_i^- = \frac{(r_i^+ - d_j)\cos\theta_i^+}{\cos\theta_i^-} - (l_k + d_k) \tag{3.6}$$

$$\theta_i^+ - \theta_i^- = \phi_k - \phi_j \tag{3.7}$$

$$r_j^{max} \geq r_i^+ \geq r_j^{stable} \tag{3.8}$$

$$r_k^{max} \geq r_i^- \geq r_k^{stable} \tag{3.9}$$

$$90 > \theta_i^+ \geq 0 \tag{3.10}$$

$$0 \geq \theta_i^- > -90 \tag{3.11}$$

$$(r_i^+ + a_j^v)\cos\theta_i^+ - p_j^v \sin\theta_i^+ > 0, \forall v \in V \tag{3.12}$$

$$\omega_{drop} r_i^+ \cos\theta_i^+ > v_c \tag{3.13}$$

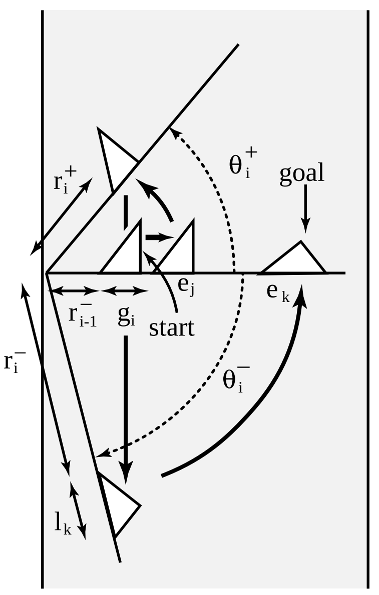


Figure 3.10: The ith transition from edge e_j to edge e_k with a translation of g_i, start and end turn radii r_i^+ and r_i^-, and raise and drop angles θ_i^+ and θ_i^-.

$$\frac{(r_i^+ - d_j)\sin\theta_i^+ - (r_i^- + l_k + d_k)\sin\theta_i^-}{v_c} > \frac{\theta_i^+ - \theta_i^-}{\omega_{drop}} \tag{3.14}$$

where l_k is the length of edge k, ϕ_j and ϕ_k are the orientations of edges e_j and e_k, v_c is the conveyor velocity, r_j^{max} and r_k^{max} are maximum valid contact radii for edges e_j and e_k, V is the set of part vertices, and a_j^v and p_j^v are the components along and perpendicular to edge e_j of the vector from the contact vertex to a part vertex v. If e_j and e_k are neighboring edges, $d_j = d_k = 0$. Else, d_j and d_k are the distances from the (virtual) intersection vertex of (extended) edges e_j and e_k to the contact vertices of these edges. Constraint 3.12 ensures that the part does not get pushed off the conveyor during the raise phase of a turn. During a turn, Constraint 3.13 ensures that the fence downward velocity is greater than the conveyor drift velocity and Constraint 3.14 ensures that the fence drop time is less than the part drift time.

The start and goal constraints are:

$$r_0^- = r_s \tag{3.15}$$

$$r_g = r_{n-1}^- + g_n \tag{3.16}$$

These define a set of nonlinear constraints over the variables g_i, θ_i^+, θ_i^-, r_i^+, and r_i^-. (We could also eliminate the equality constraints and solve for fewer variables.) Since the radius at the start of a turn depends on the radius at the end of the previous turn, an optimal solution must consider all variables simultaneously.

We want a solution which minimizes the total time to feed the part. The time taken for each stage of the plan is the sum of the jog time, fence raise time, drift time on the conveyor, and fence recovery time. We approximate the total time by the objective function we minimize:

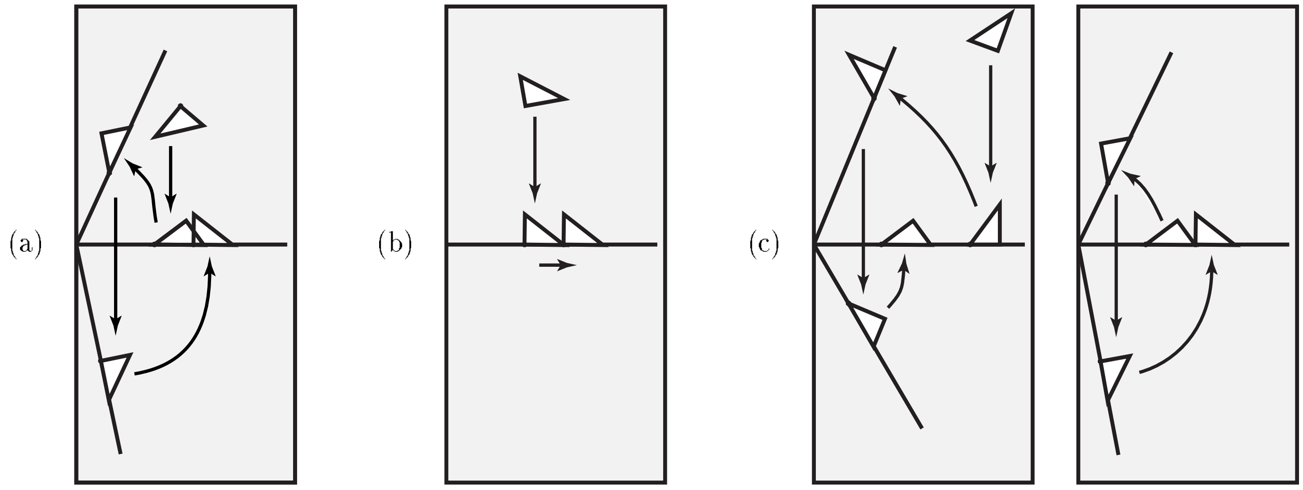


Figure 3.11: Example feeding plans for a triangle. The goal configuration ($\phi_g = 269.34, r_g = 150$) is the same, while the start configurations are different: (a) ($\phi_s = 126.49, r_s = 100$) (b) ($\phi_s = 269.34, r_s = 100$) (c) ($\phi_s = 0, r_s = 200$).

$$t_n g_n^2 + \sum_{i=1}^{n-1} \left[t_i g_i^2 + \frac{\theta_i^+}{\omega_{stable}} - \frac{\theta_i^-}{\omega_{stable}} + \frac{(r_i^+ - d_j)\sin\theta_i^+ - (r_i^- + l_k + d_k)\sin\theta_i^-}{v_c} \right] \tag{3.17}$$

The approximate time cost coefficient of the ith jog series is $t_i = k\cos\alpha_i / v_c(r_i^+ + r_{i-1}^-)$ where k is a constant chosen to be 1000, and $\alpha_i = \min(|\theta_i^{min}|, |\theta_i^{max}|)$ is a safe jog angle for the ith jog series.

Our problem as stated above is a nonlinear programming problem whose feasible solutions yield valid feeding plans. (See Bazaraa and Shetty's text [15] for an introduction to nonlinear programming.) Once we have a solution for the g_i, θ_i^-, θ_i^+, r_i^+, and r_i^- values which satisfies the constraints (3.5)–(3.16) and minimizes our objective function, we must still determine a series of jogs to execute the translations g_i.

An inward translation distance $g_i < 0$ requires a series of decreasing jogs, such that $r_i^+ = r_{i-1}^-(\cos\gamma_i)^m \cos\rho_i$ where m is a non-negative integer, and $\theta_i^{max} \geq \gamma_i \geq \rho_i \geq 0$. So this radius decreasing translation is accomplished by a series of m jogs of angle γ_i followed by a jog of ρ_i. Similarly, an outward translation distance $g_i > 0$ requires a series of increasing jogs, such that $r_{i-1}^- = r_i^+(\cos\gamma_i)^m \cos\rho_i$ where m is a non-negative integer, and $0 \geq \rho_i \geq \gamma_i \geq \theta_i^{min}$.

A feasible solution to the above problem always exists if the fence and conveyor are large enough that each stable edge has a valid contact radius greater than its minimum stable radius. When this condition is satisfied, a plan with a maximum of three stages to get the part from the start to the goal configuration exists, although it may not be optimal.

3.3.3 Generating Feeding Plans

We now outline the process of generating feeding plans. From the part description and coefficient of friction, we compute the minimum stable radii for the stable edges and the transition angles between these edges. For the start and goal configurations, we generate candidate edge transition sequences. For each such sequence, the corresponding objective function and constraints are input to a commercial nonlinear programming package called GINO [109] to solve the problem. A feasible solution to the nonlinear programming formulation provides us with a valid feeding plan. See the example plans in Figure 3.11.

Currently we use minimum stable radius values empirically determined to be robust. A conservative estimate of the minimum stable radius can be determined as illustrated in Figure 3.8. Also, the planning procedure can be extended to use convergent turns to handle parts with only two stable edges which are parallel.

3.4 Implementation

We have implemented several feeding plans on a conveyor with the fence being (somewhat ironically) actuated by one joint of an Adept 550 SCARA robot. The conveyor speed and fence rotational speed have been selected to be 20 mm/s and 30 degrees/s respectively. The fence is covered with a foam material to avoid slip between the part and the fence.

Figure 3.11 illustrates three plans we ran on our setup which achieve the same goal configuration from different starting configurations. Plan (a) requires a single turn to reach the goal, plan (b) requires just a translation, and plan (c) requires two turns. These plans had position errors at the goal configuration ranging from 0 mm to 3 mm. The errors are sensitive to part shape measurements, fence turn angles, and timing in the turns. Occasional slip also causes some error.

Evan Shechter and I developed the 1JOC planner, which has been implemented in C++. We provide the planner with the part shape, center of mass position, empirically estimated minimum stable radii for the stable edges, and the start and goal part edges and contact radii. The planner computes the necessary part information and for each feasible sequence of edges traversed using turns, generates the nonlinear constraints automatically and runs the nonlinear programming package GINO on the constraints. It then identifies the best plan as defined by the objective function. Currently the planner can automatically generate one and two stage plans. The average planner run times were 1.27 seconds for single stage plans and 1.62 seconds for two stage plans over a set of 60 trials each on a Sun Sparcstation20. The minimum times taken were 0.85 seconds and 1.25 seconds respectively.

We have also implemented software so an overhead camera can take a picture to determine the pose of a singulated part along the fence, run the planner to find a plan, and then generate the appropriate V+ program for the Adept robot to execute. The system takes 5-10 seconds from the time the camera takes an image to the time the robot starts executing the generated plan. This time can be significantly reduced with a few optimizations. Note that the feeding plan to get the part to the goal configuration requires no sensing after the initial camera snapshot.

3.5 Sensitivity to errors in geometry

During our experiments we observed that systematic errors in the contact radius of the part indicated errors in the fence turn angles or measurement of part geometry. We now analyze the sensitivity of the system to errors in fence angles and part geometry. The positioning of the part along the fence after a turn depends on the accuracy with which the fence is rotated. The contact radius after a turn is given by:

$$r_i^- = \frac{(r_i^+ - d_j)\cos\theta_i^+}{\cos\theta_i^-} - (l_k + d_k) \tag{3.18}$$

We compute the following partial derivatives to determine the sensitivity of the 1JOC system to errors in the contact radius and turn angles.

$$\frac{\partial r_i^-}{\partial r_i^+} = \frac{\cos\theta_i^+}{\cos\theta_i^-} \tag{3.19}$$

$$\frac{\partial r_i^-}{\partial \theta_i^+} = \frac{-(r_i^+ - d_j)\sin\theta_i^+}{\cos\theta_i^-} \tag{3.20}$$

$$\frac{\partial r_i^-}{\partial \theta_i^-} = \frac{(r_i^+ - d_j)\cos\theta_i^+ \sin\theta_i^-}{(\cos\theta_i^-)^2} \tag{3.21}$$

These partial derivatives show that the error in contact radius resulting from a turn is more sensitive to errors in θ_i^+ and θ_i^- than errors in r_i^+. Thus errors in fence turn angles cause positioning errors of the part. Since θ_i^+ and θ_i^- are related by the part geometry, errors in measurement of part geometry also translate into positioning errors of the part.

3.6 Variations on a Theme

The 1JOC approach uses a fixed velocity conveyor in combination with a single servoed joint to obtain the diversity of motions required for planar manipulation. We have shown by proof and demonstration that the 1JOC is capable of useful planar manipulation: any polygon is controllable from a broad range of initial configurations to any goal chosen from a broad range of goal configurations.

For this chapter we designed the system and the set of actions to simplify analysis and planning. There are many variations on the approach which may be more suitable in different contexts. Some variations on the system configuration are: a curved fence; a prismatic fence in place of the revolute fence; a rotary table in place of the conveyor; or use of gravity rather than a conveyor. Some variations on the actions are: optimization of fence rotation rates instead of using fixed values; allowing objects to slip along the fence; complete revolutions of the object to reduce jogs; and speeds high enough to require dynamic analysis instead of quasi-static.

When addressing potential industrial applications, it is important to consider the whole system, including interactions with surrounding equipment. Several different scenarios have occurred to us: a 1JOC to pose objects followed by a simple pick-and-place device; a 1JOC to singulate parts for a SCARA; or two or three 1JOCs pipelined to singulate and feed parts. In the next chapter, we will describe a sensorless 1JOC system.

Chapter 4

Sensorless Parts Orienting with a One-Joint Manipulator

This chapter[1] explores a sensorless technique for orienting planar parts using a single controlled joint in combination with a constant-velocity conveyor. We extend the *1JOC* approach described in the previous chapter to demonstrate that a variation called the *Sensorless 1JOC* can orient and feed polygonal parts up to symmetries in the underlying mechanics, without knowing the initial part location and without sensors. This result suggests that in planar parts transfer and orienting applications, we can replace a three-degree-of-freedom robotic manipulator with a simpler single-degree-of-freedom device. The main result of this chapter is that the sensorless 1JOC can orient polygons, up to symmetry in the underlying mechanics. A preliminary description of our work with the sensorless 1JOC appeared as [5].

4.1 Sensorless 1JOC Model

This section describes the sensorless 1JOC, defines the primitive actions employed, and addresses the main issue, which is the system's ability to orient polygons. The main argument is to show that we can use the sensorless 1JOC to implement Goldberg's *normal push* actions [76]. It follows that the sensorless 1JOC can orient polygons up to symmetry in the push function.

We adopt some conventions to simplify the presentation. We assume an origin at the center of the fence, and we assume the belt's velocity v is in the $-y$ direction. The fence angle is measured with respect to the x axis. We also have a part frame $\mathcal{F}_p$ fixed to the part. The fence is finite length, and has a "stop" at each end (see Figure 4.1). The pivot is not a simple revolute joint. Rather, the fence rolls without slipping on a circle, so that each point on the fence follows an evolute. If this pivot circle has radius r, then when the fence has rotated 180 degrees CW (CCW), it will also have moved up the belt by $2r$ and to the left (right) by πr. Moving up the belt by $2r$ gives the space necessary for the fence to push a part to 180 degrees, release it, and then rotate back to 0 degrees and catch the part as it drifts down the conveyor.

We assume the fence is frictionless. The part is a rigid polygon, of known shape and with a known center of mass. We assume Coulomb's law applies to the contact between polygon and conveyor, with a uniform coefficient of friction. (The *center of friction* is therefore coincident with

[1]This chapter represents joint work with Wesley H. Huang, Kevin M. Lynch, and Matthew T. Mason.

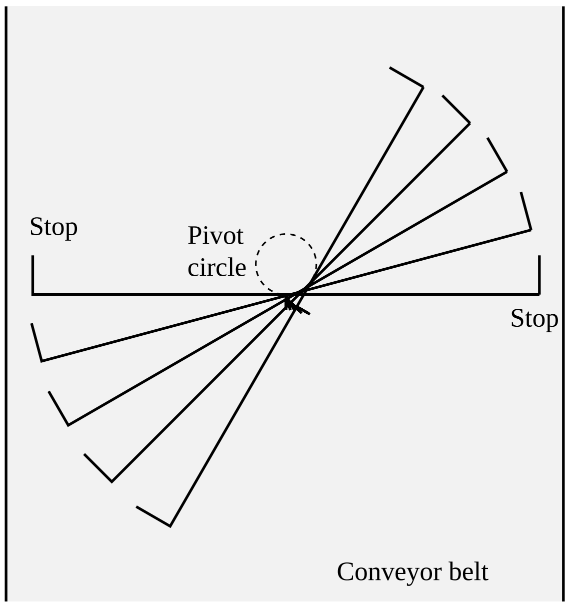


Figure 4.1: The sensorless 1JOC.

the center of mass.) We also assume quasistatic mechanics, so that the motion of the part is determined by a balance among contact forces and gravity. We assume singulated parts appear at predetermined time intervals.

The analysis depends on the definitions of *stable edge* and *metastable edge*. An edge of a polygon is a *stable edge* if the perpendicular projection of the center of friction to the edge is in the interior of the edge segment. An edge of a polygon is a *metastable edge* if the center of friction projects to a vertex. Note that when the fence is held horizontal, *i.e.* at zero degrees, a part resting on a stable edge will be stable. Metastable edges will require special treatment.

4.1.1 Primitives

There are three primitives for sensorless orienting of polygons: the *catch*, the *tilt*, and the *stable push* (see Figure 4.2):

- A *catch* begins with the part either above or on the fence. The fence is held at zero degrees until the part settles to a stable orientation on the fence.

- A *tilt* begins with the fence horizontal and the part resting on a stable edge. The fence is then tilted slightly CW (CCW) and the part slides without rolling to the right (left) stop.

- A *stable push* begins with the part on a stable edge at the left stop for a CW push, or at the right stop for a CCW push. The fence rotates, pushing the part, so that the part remains fixed relative to the fence.

We consider each primitive in detail to show that they behave as described above.

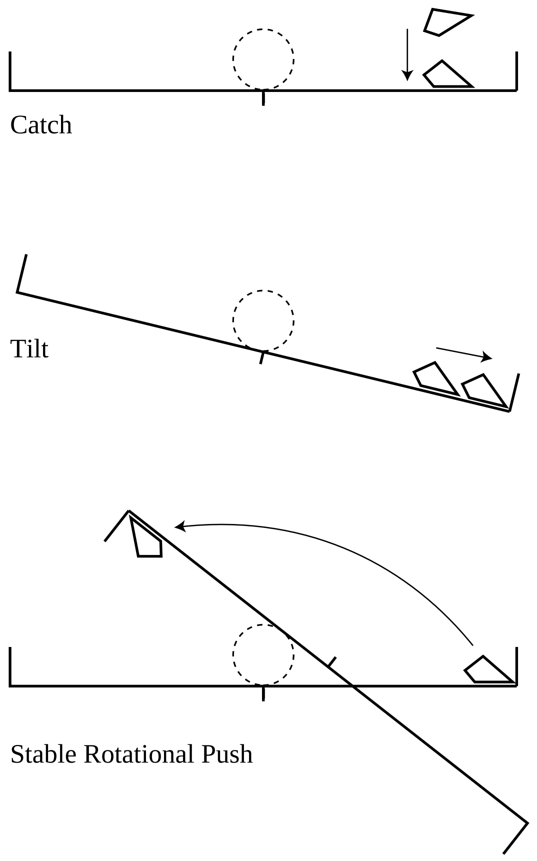


Figure 4.2: Primitives used for sensorless orienting.

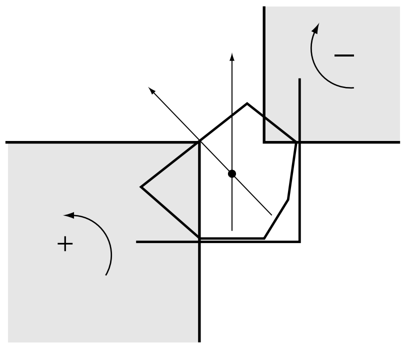


Figure 4.3: The maximum tilt angle can be derived using simple force-torque balance. Here the constraints are represented using moment-labeling regions (Mason [123]). The convex combination of the contact forces consists of the set of all forces that yield positive moment about the region labeled + and negative moment about the region labeled −. Two forces are shown—the force required for a tilt angle of zero, and a force which just begins to violate the constraint, defining the maximum tilt angle. A non-zero maximum always exists, because the polygon is on a stable edge.

The Catch

A catch occurs when a part on the conveyor contacts the fence held stationary at 0 degrees and rotates onto a stable edge.

There are a few subtleties that must be addressed:

- *Convergence time.* How long must the system wait for the part to converge to a stable edge? We need an upper bound on convergence time. In principle this upper bound can be obtained using Peshkin's and Sanderson's results [141]. However there is an exception, which occurs when a part is balanced on a vertex with its center of friction directly above. Our planning method will avoid these configurations, but there are two special cases discussed below.

- *Starting configuration.* If we start with an unoriented polygon above the fence, the part could arrive balanced on a vertex, or very nearly balanced. In that case there is no upper bound on convergence time. The easiest way to deal with it theoretically is to assume the part starts on the fence on a stable edge, i.e., just after the initial catch. The easiest way to deal with it practically is to determine a time that allows convergence with a very high probability.

- *Metastable edges.* Convergence time is also unbounded for a metastable edge. We need some method of perturbing the fence so that the polygon does not linger indefinitely on a metastable edge. It turns out that a very small rotation of the fence, opposite to the desired part rotation, moves the contact normal off of the center of mass, inducing a torque that causes the desired part rotation.

- *Stop interference.* As a part converges to a stable edge, it may come into contact with a stop, following a very small stable push. We know of no instance where this possibility affects the success of the plan, and we will simply assume that it is not a problem.

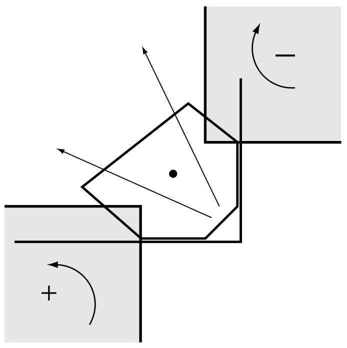


Figure 4.4: Frictionless contact forces through the edges can be represented as moment-labeling regions as in Figure 4.3.

The Tilt

A tilt is a CW or CCW rotation of the fence so that a part resting on a stable edge on the fence at 0 degrees slides without rotating to the right or left stop. The sensorless 1JOC uses a tilt before each stable push, and once more at the end to eliminate positional uncertainty of the part.

The part slides faster for steeper tilts, but there is a limit to how steep we can tilt the fence without rotating the part when it contacts the stop. Figure 4.3 illustrates a simple way of determining the maximum tilt angle.

The Stable Push

When the part is resting against the right (left) fence stop, a stable push allows the fence to rotate the part CCW (CW). The part remains fixed relative to the fence as it moves. To show that these pushes are stable, we must ensure that the contact forces provided by the fence and the stop can balance the support friction force for any pressure distribution of the part consistent with the center of friction. In this section we show that it is always possible to choose a pivot circle radius r, fence length, and ratio of the conveyor velocity v to the fence's pushing angular velocity ω to make the pushes stable.

Assume the part is resting against the right stop. The set of possible contact forces is given by the convex combination of the frictionless forces at each contact (Figure 4.4). For a set of contact forces, we can find a set of stable pushing motions using the method described by Lynch [112]. Figure 4.5 shows the CCW stable pushing motions $\mathcal{V}_{stable}$ for the example of Figure 4.4. The stable pushing motions are represented as rotation centers fixed in the part frame $\mathcal{F}_p$, relative to the conveyor it is sliding on. For these rotation centers, the contact forces can balance the support friction force for any pressure distribution of the part. Technically, to prove the motion is stable, we must also prove that no other motions of the part are possible. Here we simply assume it to be the case.

Now we assume that the conveyor velocity v is negligible. Then during a stable push, the rotation center in the part frame $\mathcal{F}_p$ is the point on the fence in contact with the pivot circle. Defining the zero location on the fence to be the point in contact with the circle when the fence is held at zero degrees, the rotation centers on the fence sweep the range $-r\pi/2$ to $r\pi$, where r is the radius of the pivot circle, as the fence rotates from $-\pi/2$ to π. (Note that $-\pi/2$ is the steepest

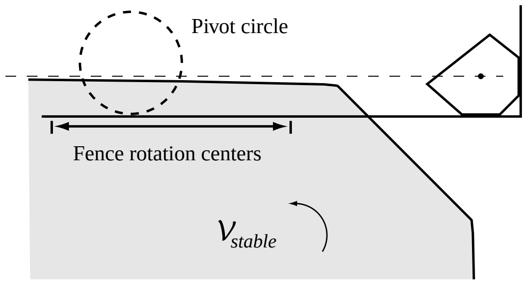


Figure 4.5: The set of CCW stable pushing motions $\mathcal{V}_{stable}$ represented as rotation centers. Also shown is an example pivot circle and the points on the fence that roll on the pivot circle as it rotates from $-\pi/2$ to π.

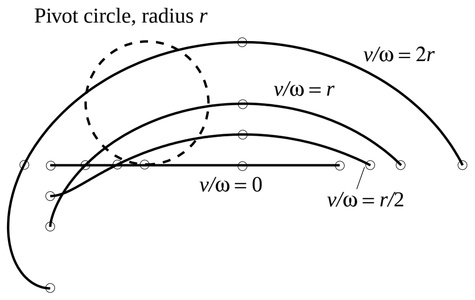


Figure 4.6: Curves of the rotation center of the part relative to the conveyor during stable pushing. The curves shown are for several different ratios of v/ω. Each curve is drawn by varying the fence angle θ in the range $[-\pi/2, \pi]$. Circles are drawn around the rotation centers on each curve at fence angles $-\pi/2, 0, \pi/2, \pi$.

possible tilt angle.) We can see in the example of Figure 4.5 that the push is stable for all angles of the fence.

We can always choose the fence to be long enough that the push is stable for all fence angles. When the rotation center is an infinite distance away along the fence, the required contact force $\mathbf{f}$ is normal to the fence and through the center of friction. As the rotation center moves in along the fence, the required contact force smoothly rotates counterclockwise because the part's support friction is "above" the fence. This counterclockwise force is provided by the stop (see Figure 4.4). More precisely, we can define the largest ball $\mathcal{B}$ of forces about $\mathbf{f}$ such that all forces in $\mathcal{B}$ and counterclockwise of $\mathbf{f}$ can be provided by the contacts. Because $\mathcal{B}$ always has nonzero radius, the distance to the nearest stable rotation center is finite, implying that we can always choose a sufficiently long fence to make pushing stable if the conveyor velocity v is negligible.

If the conveyor velocity v is not negligible with respect to the pushing angular velocity ω, the rotation center of the part relative to the conveyor during stable pushing can be written as a function of the ratio v/ω. For a fence angle θ, the rotation center location is (x, y), $x = \theta r + \frac{v}{\omega}\cos(\pi - \theta)$, $y = \frac{v}{\omega}\sin(\pi - \theta)$, where the x axis is aligned with the fence and the origin is the point on the fence in contact with the pivot circle at $\theta = 0$. The rotation center traces out a curve in the part frame $\mathcal{F}_p$ (Figure 4.6).

For a push to be stable, the curve must stay inside $\mathcal{V}_{stable}$. For a given pivot circle, fence length,

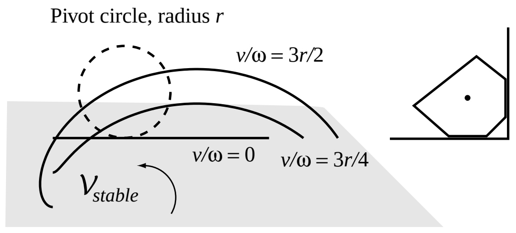


Figure 4.7: Rotation center curves for different ratios v/ω. For this example, the maximum conveyor velocity v that guarantees stable pushes is $3r\omega/4$. For larger values, the curve leaves $\mathcal{V}_{stable}$.

and pushing velocity ω (perhaps limited by the quasistatic assumption), we would like to find the highest possible conveyor velocity v. To do this, we increase v until the curve no longer remains in $\mathcal{V}_{stable}$ (Figure 4.7). By doing this for each possible configuration of a given part, we can find a maximum conveyor velocity guaranteed to give stable pushes for all configurations. Maximizing the conveyor speed maximizes the feeder's throughput.

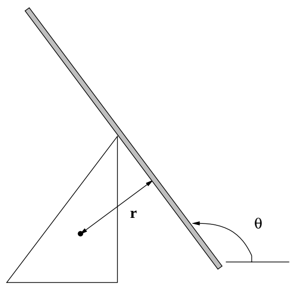
4.2 Planning

The sensorless 1JOC can use the primitives described above to orient parts sensorlessly. A sensorless orienting plan begins with a catch to acquire the part, followed by a sequence of stages to bring a known part edge to rest on the fence. Each stage consists of a tilt, a stable push, and a catch. The final step of the plan is a tilt to bring the part to a known stop of the fence. At the end of plan execution, the part is in a known orientation at a known position on the fence.

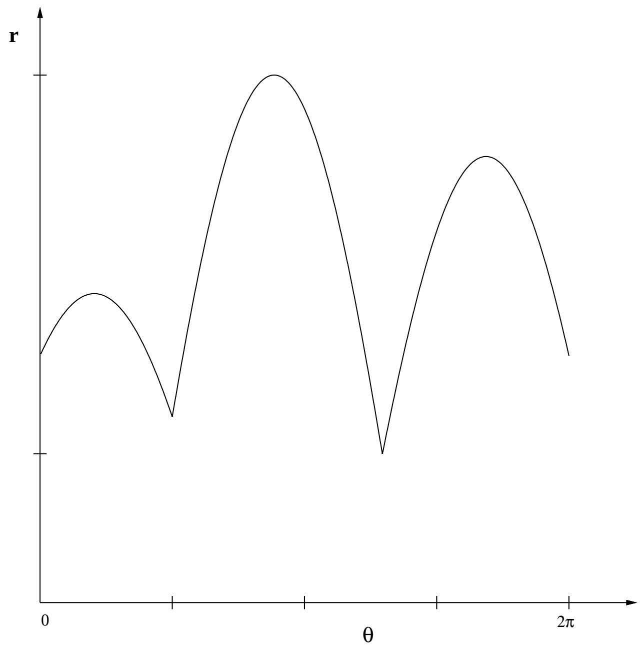
4.2.1 Predicting Part Rotation

The fence can rotate a part by any angle in the range $[0, 2\pi)$ using a stable push. Each stable push is followed by a catch when the part drifts into contact with the fence. Viewed from a frame fixed in the belt, each catch is a *linear normal push*, meaning that the fence is pushing the part in a direction normal to the fence face. To determine the rotation of the part during a catch, we follow Goldberg [76] in using the *radius function* (Figure 4.8). The radius of a part is the perpendicular distance from a reference point in the part to a support line. The radius function $r : \mathcal{S}^1 \to \mathcal{R}^1$ is a plot of the radius as the support line is rotated, and has a period of 2π. When the center of friction is the reference point and the fence is the support line, the local minima of the radius function occur at stable edges of the part. A part being pushed by a fence rotates to achieve a minimum radius. Each local minimum in the radius function determines a convergent orientation, and each local maximum determines a divergent orientation. Hence a normal push has the net effect of mapping the entire interval between two divergent orientations to the enclosed convergent orientation.

The *push function* [76], obtained from the radius function, is an alternative representation of the effect of a normal push. It is a mapping $p : \mathcal{S}^1 \to \mathcal{S}^1$ from the initial relative orientation of a part to the resulting orientation (Figure 4.9), and has a period of 2π. Symmetry in part shape with respect to the center of friction leads to a period less than 2π, and we say the part has a symmetric push function.



(a)



(b)

Figure 4.8: The radius function for a triangle with its center of friction indicated by the black dot. (a) The radius r of a part at a fence orientation θ is the perpendicular distance from the center of friction to the fence. (b) The radius function is the plot of the part radius as the fence orientation is varied. Based on [76].

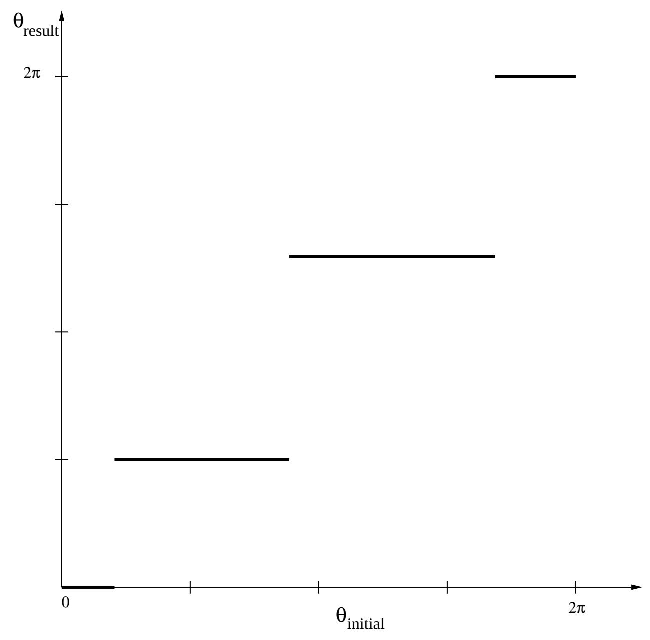


Figure 4.9: The push function for the triangle. The horizontal and vertical axes are the initial and resulting relative orientations of the part.

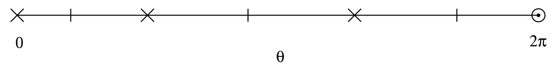


Figure 4.10: A compact representation of the push function for the triangle. The x's and the vertical bars indicate the convergent and divergent fence orientations respectively, and the circle indicates wraparound at 2π.

A compact representation of the push function is shown in Figure 4.10.

4.2.2 Plan Existence and Generation

The push function is a monotonic step function which wraps around at 2π. This means that Goldberg's backchaining algorithm [76] can be used to find the shortest sequence of fence rotations to orient a part. Orienting a part with n stable edges takes $O(n)$ stages. Further, the algorithm can generate a sequence of rotations to orient any part up to symmetry in its push function.

In our implementation, we use breadth-first search in the space of representative actions to find the shortest sequence of fence rotations to orient the part. We use the push function to find equivalence classes of actions—action ranges with the same effect. The representative actions chosen from the middle of each range provide a finite discretization of the action space that also covers it. See Figure 4.11 for an example plan.

The sequence of fence rotations specifies the sequence of stable pushes, each of which is followed by a catch. Stable pushes in the range $(-\pi, \pi)$ are sufficient to rotate the parts as desired. We have to precede each stable push with an appropriate tilt. Prior to a CCW stable push in the range $[0, \pi)$, we tilt the fence CW to bring the part to the right stop. Prior to a CW stable push in the

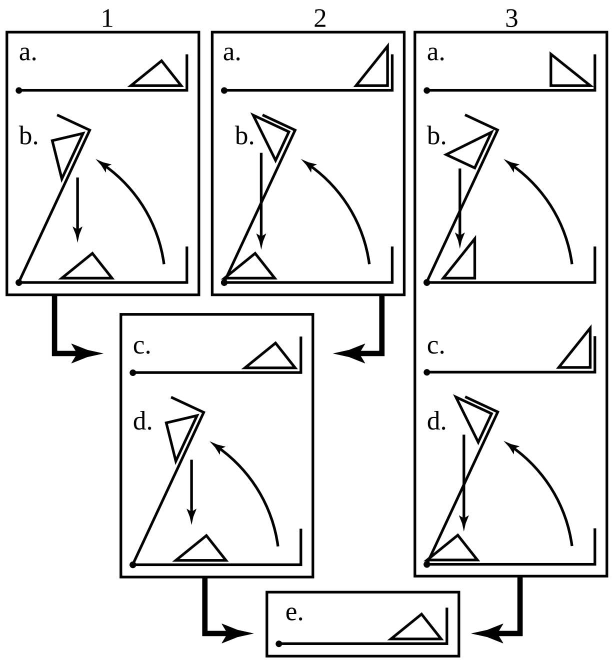


Figure 4.11: Example sensorless feeding plan. Three different initial orientations converge to a single final orientation. The tilts are not shown, and only the right half of the sensorless 1JOC is used in this plan.

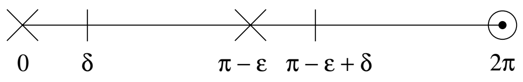


Figure 4.12: This push function requires the fence to push the part at least $\pi - \delta - \epsilon$ radians before releasing it and catching it. This allows the fence to change its orientation relative to the part by an angle $-\pi + \delta + \epsilon$, the minimum required to distinguish between the two stable edges.

range $(-\pi, 0]$, we tilt the fence CCW to bring the part to the left stop. Once the tilt direction is determined, the tilt angle is computed for the set of possible part orientations for the stage.

To ensure that the sensorless 1JOC can orient all parts up to symmetry in the push function, the fence must be capable of stable pushing anywhere in the range $(-\pi, \pi)$ before releasing and catching. To see that this is necessary, consider the push function of Figure 4.12. There are two stable edges at angles 0 and $\pi - \epsilon$, where ϵ is a small positive value, so the push function is not symmetric. The angle range from both stable angles to the counterclockwise limits of their convergence regions is δ, where δ is a small positive value.

Because the distances to both counterclockwise limits are equal, we must distinguish between the stable edges by using the different distances to their clockwise limits. The fence must move clockwise relative to the part by a magnitude of at least $\pi - \delta - \epsilon$. This implies a counterclockwise stable push (before the release and catch) of the same magnitude. As $\delta \to 0, \epsilon \to 0$, the required push angle approaches π.

Figure 4.13 constructs a part with the push function of Figure 4.12, requiring a stable push of up to π to orient it. The reflection of this part requires a stable push of up to $-\pi$, indicating that

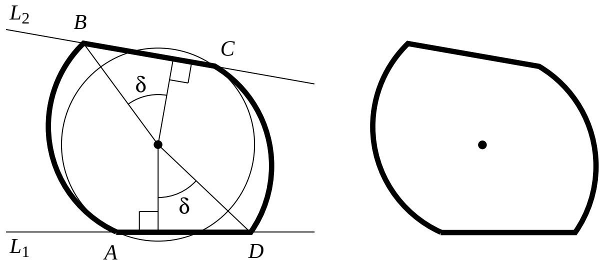


Figure 4.13: Constructing a part with the push function of Figure 4.12. Start with a circle centered at the center of friction of the part. Cut the bottom of the circle with the horizontal line L_1, and cut the top of the circle with the line L_2, which is at an angle $-\epsilon$ to the horizontal. The intersections of these lines with the circle define the points A and C as shown. Draw the normals to L_1 and L_2 which pass through the center of friction. Pivot these normals an angle δ about the center of friction and intersect them with the lines L_1 and L_2, giving the points B and D as shown. Now construct a curve from A to B which spirals away from the center of friction. Construct a similar curve between C and D. The part is defined by the curve from A to B, the stable edge from B to C, the curve from C to D, and the stable edge from D to A. By making the cuts, ϵ, and δ arbitrarily small and replacing the curves with an arbitrarily large number of line segments (none of these edges is stable), we create a polygonal part that requires a stable push of up to π to orient it.

the sensorless 1JOC must be capable of stable pushes in the range $(-\pi, \pi)$ to orient all parts up to symmetry in the push function.

Most parts can be oriented with an angle range much less than $(-\pi, \pi)$, and we are interested in characterizing such parts. For example, it is easy to show that all triangles are orientable using the angle range $(-\pi/2, \pi/2)$. There exist quadrilaterals, however, that cannot be oriented up to symmetry using this angle range. Goldberg and Overmars [80] showed that $(-\pi/2, \pi/2)$ is sufficient if parts possess no "partial" symmetries—the distances from each stable angle to each edge of its convergence range are unique.

4.3 Implementation

Our implementation of the sensorless 1JOC uses rotation about a fixed pivot point since most parts do not require rotations close to π. We used a plexiglass fence and delrin parts to test the plans. We generated plans to test the right triangle and 8-gon shown in Figure 4.14. See the two stage plan to orient the triangle in Figure 4.11. We used a conveyor velocity of 20 mm/sec and fence angular velocities of 30 degrees/sec and 6 degrees/sec for the stable pushes and tilts respectively. We used tilt angles of 45 degrees. The times for catches and tilts to proceed to completion were found empirically. As a practical issue, we note that stable push actions selected from smaller action ranges are less robust than those selected from larger action ranges. In fact, we specify a minimum action range size of 5 degrees for valid actions.

We ran 21 trials for the triangle starting in different configurations along the fence. All 21 trials were successful. We ran 20 trials on the 8-gon for different start configurations, of which 16 were successful. The failures occurred during tilts when the polygon rotated away from the stable edge it

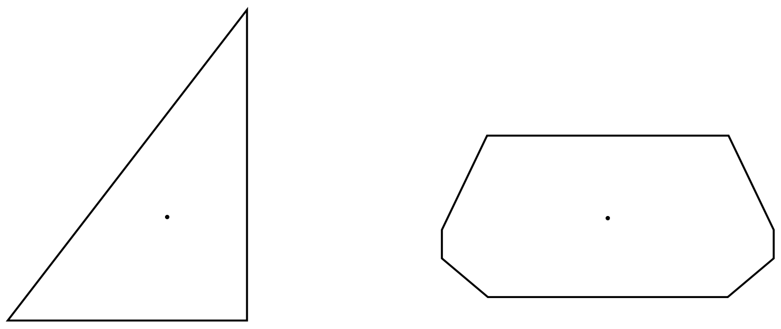


Figure 4.14: Example parts used in experiments.

was sliding on, probably due to the nonzero friction coefficient of the fence. From our tests on this and other parts, we observed that edges which are nearly unstable sometimes cause undesired part rotations, particularly when a part slides on such an edge during a tilt. Such edges also sometimes require small tilt angles, which may not allow sliding to proceed to completion. Another occasional failure mode occurs when a part contacts the fence stop in the middle of a tilt, leading to undesired part rotation as the tilt progresses.

4.4 Conclusion

This chapter demonstrates a system with one controlled degree of freedom that can orient and feed polygons without using sensors. Our implementation shows some unresolved difficulties, but suggests that our modeling assumptions are reasonably faithful to the behavior of the real world. The approach is mechanically simple, yet is readily adapted to new parts. The 1JOC approach admits many variations, offering some promise that remaining practical issues (singulation of the part, part feed rate, out of plane motion) may also be resolved.

Chapter 5

Parts Orienting with Partial Sensor Information

A parts orienting system has to bring randomly oriented parts to desired orientations for subsequent assembly. Vibratory bowl feeders are the most commonly used systems in industry today for orienting parts (Boothroyd *et al.* [23]; Riley [156]). While very effective, bowl feeders are not reconfigurable and require skilled craftsmen to design them. It can take a couple of months to design a bowl feeder for a new part or even a slightly modified part, or to discover that the part cannot be fed. So there is a pressing need for flexible parts orienting systems, especially systems whose capabilities can be characterized in advance. Using our knowledge of the mechanics and geometry of the manipulation processes during parts orienting, we can flexibly generate orienting strategies for new parts with simple hardware elements and planning software. This can reduce changeover costs and time to market for new products. Further, designers can use such tools during the design stage to assess the orientability of parts for assembly, leading to virtual prototyping.

In this chapter we explore another form of parts orienting on a conveyor where partial sensor information on part orientation is used to eliminate uncertainty in the orientation of the part. Here the robot has three degrees of freedom (rotation and translation along the vertical axis, and horizontal translation along the direction of conveyor motion), and can pick up the part using an electromagnet or suction device.

Most orienters and feeders use a variety of mechanical operations to change the state of the part. These include shaking the parts (bowl feeders, APOS), and using sequences of fences and cutouts to align or reject parts (bowl feeders). Sometimes these mechanical operations are augmented with a vision system. Broadly speaking, sequences of uncertainty reducing mechanical operations can be used to orient parts. The class of such operations includes pushing the parts with a moving fence, aligning them against stationary fences on a conveyor, parallel-jaw grasping, or shaking parts into stable orientations. Besides understanding the mechanics of the operations, the central issues for such systems are establishing the existence of sequences of operations to orient any part, generating such sequences, and finding bounds on the length of the orienting sequences.

We focus on a particular implementation where a part on a moving conveyor belt contacts a stationary fence and rotates into alignment with it. The fence acts to push and align the part. The part is made to repeatedly align itself against the fence by being picked up, rotated, and dropped off upstream by a robot with a suction device or electromagnet. A sequence of such *push-align operations*, where the robot rotates the part prior to letting it be aligned by the fence, is executed until the part's orientation is determined (Figure 5.1). We identify the set of stable states the part can assume, and then find a sequence of push-align operations to bring the part to a known state. We study these operations both with and without sensing of the resting diameter of the part. The principal questions to answer for this system are:

1. Does a sequence of operations to orient any part always exist? Can we find these sequences?
2. What is the maximum number of operations to orient a given part?
3. Can a part be oriented without the use of sensors? What are the advantages of using partial information sensors?

We show that it is possible to orient parts by sensorless and sensor-based push-align operations, and determine the number of operations required in each case. We identify planning methods and establish characteristics such as solution existence, planner completeness, plan length, and planning complexity. Focusing on a particular model and implementation of orienting allows us to explore the basic issues and develop solution techniques. These principles and techniques can then be applied to other part classes and parts orienting processes, such as parallel-jaw grasping or pushing by passive or actuated fences on a conveyor, leading to practical applications.

A preliminary version of this work appeared in Akella and Mason [7]. Our work builds on the work of Taylor *et al.* [176] who performed automatic planning of sensor-based manipulation programs using AND/OR search. They applied this framework to the task of orienting an object by tray tilting, and the task of orienting and grasping an object by a parallel-jaw gripper. Our work is also related to the work of Rao and Goldberg [150] on orienting multiple parts by parallel-jaw grasping. They describe the characteristics of the diameter function and its implications for the uniqueness of part shapes and orientability and recognizability of parts.

5.1 Assumptions

In analyzing our system, we make the following assumptions:

1. All parts are polygons. Since the parts contact a flat fence, nonconvex polygons are treated by considering their convex hulls.
2. The part shape is known and the center of mass of the part is at a known position in its interior.
3. All motions are in the plane and are quasi-static.
4. The fence is normal to the conveyor motion direction.
5. All frictional interactions are described by Coulomb friction.
6. The coefficient of friction between the part and the conveyor surface is uniform.
7. There is zero friction between the part and the fence.
8. All bodies in contact are perfectly rigid.

5.2 Action Model

We assume a singulated part in an unknown initial orientation drifts on the conveyor belt until it contacts a fence placed perpendicular to the direction of motion (see Figure 5.1). When the

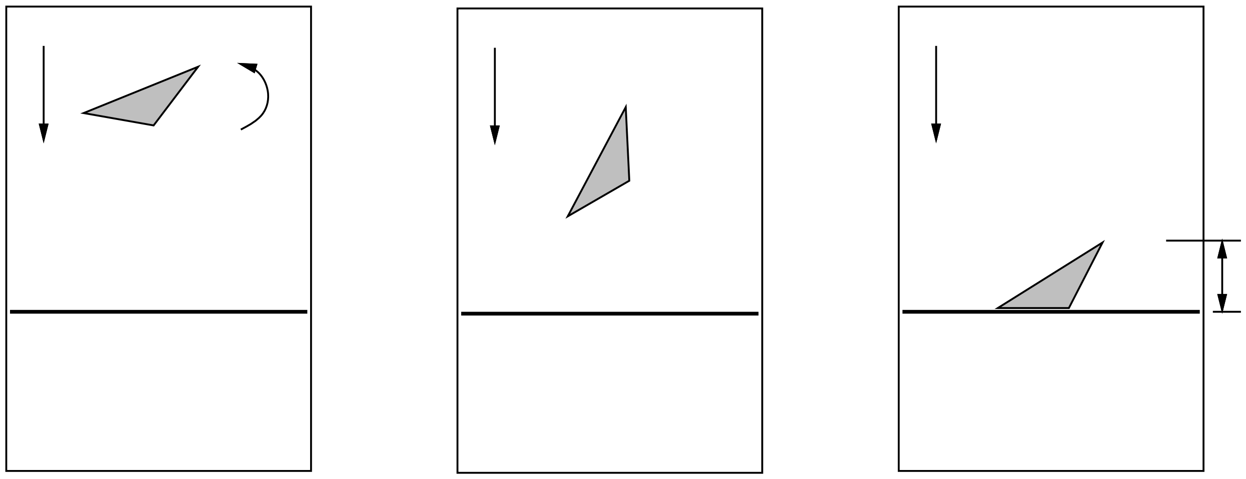


Figure 5.1: Schematic overhead view of a part on the conveyor during a push-align operation. The conveyor motion is "downward".

part contacts the fence, it rotates and possibly slides until it comes to rest with one of its stable edges aligned against the fence. We have an LED sensor mounted perpendicular to the fence that can measure the resting diameter, or width, of the part perpendicular to the fence. This diameter measurement provides only partial information on the orientation of the part. Our goal is to find a sequence of operations for the robot to execute to orient the part uniquely. We use a sequence of *push-align operations* — each operation consists of an action followed by a sensor measurement. The action consists of the robot picking up an aligned part at the fence, translating upstream from the fence, rotating the part through a chosen angle, and then releasing the part back on the conveyor for it to undergo alignment at the fence. The sensor measures the diameter of the part after it is aligned. The robot picks up the part using either a suction cup or an electromagnet if the part is ferromagnetic. By orienting a part, we mean determining the edge of the part that is aligned against the fence.

5.2.1 Mechanics

The result of a push depends on the part geometry, support pressure distribution, fence-part contact friction, length and direction of push ([121, 141]). The support pressure distribution is usually unknown and is constantly changing. The location of the center of friction determines the rotation direction resulting from a push. For a uniform coefficient of support friction, the center of friction coincides with the center of mass.

When a part on the moving conveyor contacts the stationary fence, it is being pushed normal to the fence face. Viewed from a frame fixed in the belt, this is exactly a *linear normal push*, where a moving fence pushes a part in a direction normal to the fence face. The part rotation due to such a push can be predicted using the radius function ([76]). The part comes to rest on the fence with only stable edges aligned with the fence. The center of mass location determines which part edges are stable; if the projection of the center of mass onto the edge lies in the interior of the edge, the edge is stable.

When the part contacts the fence, it takes a certain amount of time to rotate into alignment with the fence. We use a frictionless fence, so the part may also slide along the fence during the alignment phase. For a known constant conveyor velocity, we can find an upper bound on the time required for the rotation using the results of Peshkin and Sanderson [141]. They determine the minimum distance a part has to be pushed by a fence to rotate it into alignment with the fence.

There are a couple of additional subtleties during the push-alignment phase. During the initial alignment, if the part center of mass lies exactly on the fence normal during the push, the alignment time is unbounded. A practical solution is to empirically determine a time bound within which the part is aligned with high probability. If the center of mass projects to one of the edge vertices, the edge is metastable. When the part may come to rest on a metastable edge, we can perform an additional small rotation of the part to perturb its orientation and cause alignment on a neighboring stable edge. These perturbational rotations may be in one or both directions.

5.2.2 Radius Function, Push Function, and Push-diameter Function

To determine the result of a push-align operation, we follow Goldberg [76] in using the *radius function*. The radius r of a polygon at an orientation ϕ is the perpendicular distance from a reference point in the polygon to a support line of the polygon with orientation ϕ. The radius function describes the variation in the radius as the support line is rotated (see Figure 5.2).

Definition 5.1 *The* **radius function** *of a polygon is a mapping from the orientation ϕ of a support line of the polygon to the perpendicular distance r from a reference point in the polygon to the support line. We specify this reference point to be the center of mass of the polygon.*

Definition 5.2 *A* **kink** *in the radius function is a point with a discontinuity in slope that is not a local minimum of the radius function. A kink corresponds to an unstable edge of the part. It can also correspond to a metastable edge of a part, for which it has coincident local maxima and minima.*

When the center of mass is the reference point and the fence is the support line, the local minima of the radius function correspond to stable edges of the part. Kinks in the radius function correspond to unstable and metastable edges of the part. A part being pushed against a fence rotates to achieve a minimum radius. Each local minimum determines a convergent orientation and each local maximum determines a divergent orientation. A push has the net effect of mapping the entire interval between two divergent orientations to the enclosed convergent orientation. The convergent orientation provides the resulting part orientation and the resulting minimum radius is the perpendicular distance of the center of mass from the fence. The radius function has a period of 360 degrees. Symmetry in part shape leads to symmetry in the radius function and a period less than 360 degrees.

The *push function* [76], obtained from the radius function, is an alternative representation of the effect of a normal push. It is a mapping $p : \mathcal{S}^1 \rightarrow \mathcal{S}^1$ from the initial relative orientation of a part to the resulting orientation (Figure 5.3). It is a monotonic step function and has a period of 360 degrees. Symmetry in part shape with respect to the COM leads to a period less than 360 degrees, and we say the part has a symmetric push function. As we will see later, we can use the push function to plan a sequence of actions to orient a part sensorlessly.

The *push-diameter function* (Figure 5.4) is another representation of the result of a push-align operation. It is a mapping $d : \mathcal{S}^1 \rightarrow \mathcal{R}^1$ that describes the part diameter resulting from a push for a

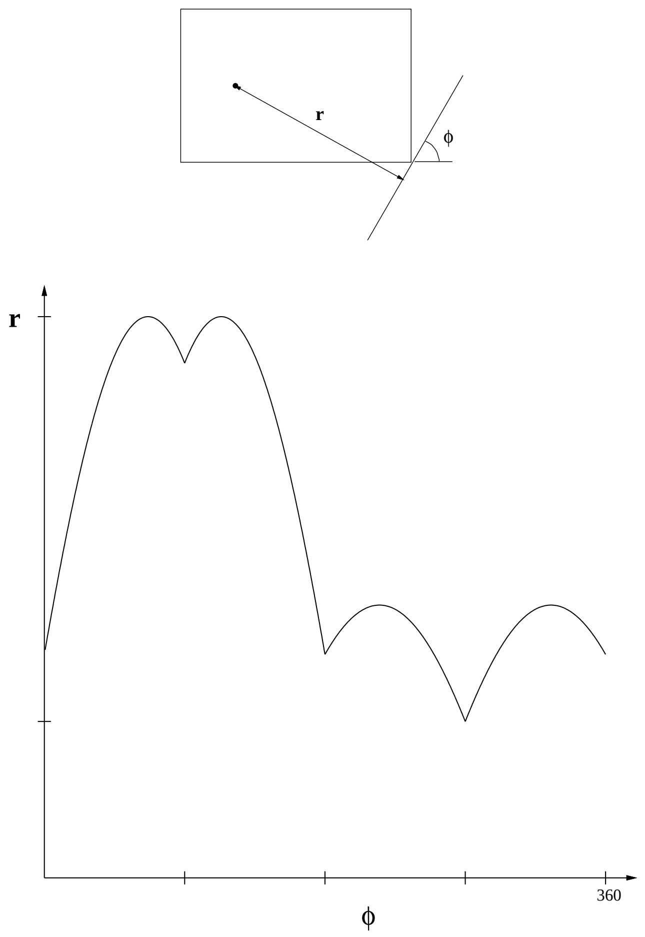


Figure 5.2: The radius function for the rectangle with its center of mass indicated by the black dot. (a) The radius r of a part at a fence orientation ϕ is the perpendicular distance from the center of mass to the fence. (b) The radius function is the plot of the part radius as the fence orientation is varied. The orientation wraps around at 360°. The rectangle has four stable edges since its radius function has four minima. Based on Goldberg [76].

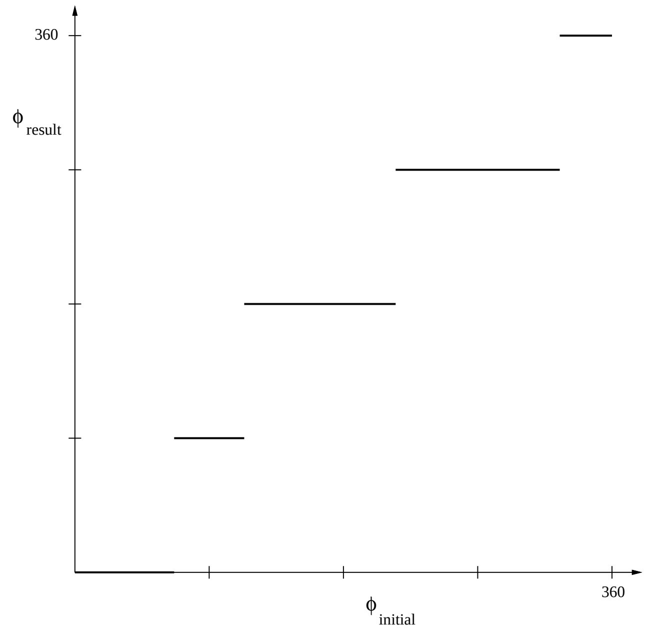


Figure 5.3: The push function for the rectangle of Figure 5.2. The horizontal and vertical axes represent the initial and resulting relative orientations respectively. Based on [76].

given initial relative orientation of the part. The diameter is the maximal part dimension measured normal to the fence when the part is stably aligned against the fence. A simple sensor consisting of a series of LEDs can measure this part diameter. If the stable diameters are all unique, the sensor can distinguish them and determine the orientation of the part in a single step. However when the resulting diameter cannot be distinguished by the sensor, a sequence of operations is necessary to orient the part. The push-diameter function is a step function with a period of 360 degrees. If the push-diameter function of a part has a period less than 360 degrees, we say the part has a symmetric push-diameter function. Symmetry of the push-diameter function implies symmetry of the push function.

5.2.3 The Radius Function and Part Shape

Since the results of push-align operations are determined from the radius function, it is important to determine the properties of the radius function in relation to part shape. We prove that every convex polygon uniquely defines a radius function, where the polygon's shape is specified by the relative locations of its vertices and center of mass. Note that all nonconvex parts with the same convex hull have the same radius function.

We call a radius function *valid* if it is the radius function of some polygon. A valid radius function is a piecewise sinusoidal function of period 360 degrees. To see this, consider the variation in radius as the support line is rotated from an edge e_i to its CCW neighboring edge e_{i+1} (Figure 5.5). Let the distance from the center of mass to their common vertex v_{i+1} be d_{i+1} and the orientation of the line joining the center of mass and the common vertex be δ_{i+1}. The radius r at an orientation ϕ is then given by $r = d_{i+1} \sin(\phi - \delta_{i+1})$, $\phi_i \leq \phi \leq \phi_{i+1}$. Changes in the sinusoids occur only at

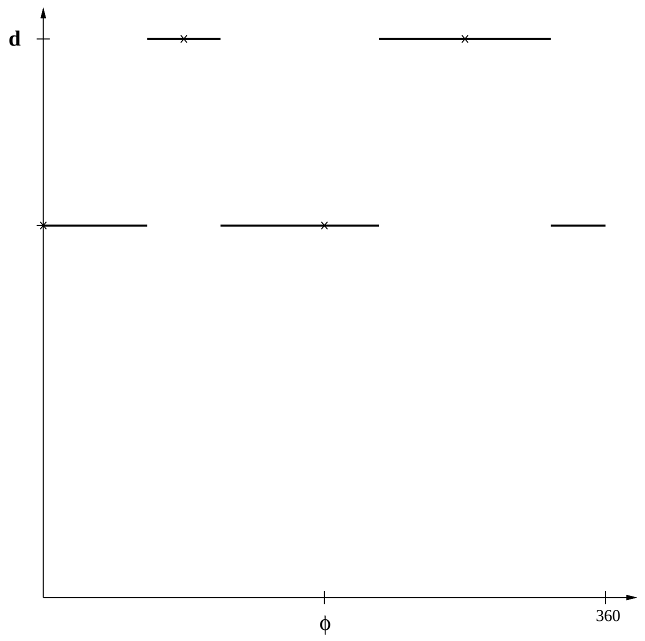


Figure 5.4: The push-diameter function for the rectangle of Figure 5.2. The function plots the stable diameter value resulting from a push-align operation for each initial relative orientation of the part. The steps indicate stable ranges and x's indicate stable orientations of the part.

minima and kinks of the radius function. Minima correspond to stable edges and kinks correspond to unstable and metastable edges of the part. This suggests that we can recover the shape of a part from the minima and kinks of its radius function. A valid set of minima and kinks is a set of minima and kinks associated with a valid radius function.

Lemma 5.1 *If two valid radius functions have the same set of minima and kinks, then they are identical.*

Proof: It is sufficient to show that the orientations and radius values of the minima and kinks determine the value of the radius function everywhere. Each minimum or kink is specified by its orientation ϕ_i and radius r_i. Order the minima and kinks by their orientations and consider pairs of successive minima and kinks. Each such minimum-minimum, minimum-kink, or kink-minimum pair specifies the endpoints of a sinusoidal segment of the radius function. Consider a pair $[(r_i, \phi_i), (r_{i+1}, \phi_{i+1})]$ corresponding to the endpoints of a sinusoidal segment. As shown above, the radius r of the sinusoidal segment at an orientation ϕ is given by $r = d_{i+1}\sin(\phi - \delta_{i+1})$, $\phi_i \leq \phi \leq \phi_{i+1}$. So

$r_i = d_{i+1}\sin(\phi_i - \delta_{i+1})$ and
$r_{i+1} = d_{i+1}\sin(\phi_{i+1} - \delta_{i+1})$.

We solve for d_{i+1} and δ_{i+1} to obtain:

$$\delta_{i+1} \quad = \quad \tan^{-1}\left(\frac{r_i \sin\phi_{i+1} - r_{i+1}\sin\phi_i}{r_i\cos\phi_{i+1} - r_{i+1}\cos\phi_i}\right)$$

$$d_{i+1} = \frac{r_i}{\sin(\phi_i - \delta_{i+1})}$$

From the polygon geometry, $|\phi_i - \phi_{i+1}| \leq 180$, and $\phi_i \leq \delta_{i+1} \leq \phi_{i+1}$. This implies there is a unique solution for δ_{i+1} and d_{i+1}. Therefore the pair $[(r_i, \phi_i), (r_{i+1}, \phi_{i+1})]$ uniquely defines a sinusoidal segment. We can perform this computation for each pair of successive minima and kinks to uniquely determine the corresponding radius function. □

Theorem 5.2 *If there are two convex polygons whose radius functions are identical, the polygons are identical.*

Proof: We show that a valid radius function is consistent with a single polygon. We present a constructive proof in the form of a procedure to construct a convex polygon from a valid radius function. The part shape is defined by the relative locations of the vertices of the polygon and reference point, the center of mass, in the interior of the polygon. First choose a reference point as the center of mass (see Figure 5.6). Then determine the minima and kinks of the radius function. The minima of the radius function give us the orientations of the lines along which the stable edges lie and their distances from the reference point. Kinks of the radius function give us the orientations and distances of the lines along which unstable and metastable edges of the part lie. Draw these lines, each of which contains a polygon edge. Since the edge orientations provide the ordered sequence of edges, we can compute the intersections of lines containing adjacent edges to find the vertices of the polygon. Since each pair of adjacent lines intersects at a single point, by construction each set of minima and kinks yields a single polygon. By Lemma 5.1, a valid radius function is uniquely determined by a valid set of minima and kinks, and hence each valid radius function maps to a single convex polygon. □

We now generalize Theorem 5.2 towards proving that two convex polygons have the same radius function if and only if they have the same shape. That is, we wish to treat all polygons generated by rotating a given polygon as the same shape. Two parts P_A and P_B with radius functions $r_A(\phi)$ and $r_B(\phi)$ are said to have the same radius function if there exists an angle α such that $r_A(\phi) = r_B(\phi + \alpha)$. If $r_A(\phi) = r_B(\phi + \alpha)$, let the part P_A rotated counterclockwise by α degrees have the radius function $r'_A(\phi)$. Then $r'_A(\phi) = r_B(\phi)$. So without loss of generality we assume that α is zero.

Theorem 5.3 *Two convex polygons have the same radius function if and only if they have the same shape.*

Proof: The "if" part of the above theorem is obvious. That is, two convex polygons have the same radius function if they have the same shape.

We prove the "only if" part using proof by contradiction. Assume there are two convex polygons P_A and P_B with radius functions $r_A(\phi)$ and $r_B(\phi)$ respectively, such that P_A and P_B have different shapes and $r_A(\phi)$ and $r_B(\phi)$ are identical radius functions. Using the radius function $r_A(\phi)$, construct P_A as described above. Similarly construct P_B from $r_B(\phi)$. Since $r_A(\phi)$ and $r_B(\phi)$ are the same, the sets of minima and kinks of the radius functions for the two parts are the same, and the two constructions yield the same shape, leading to a contradiction.

Therefore two convex polygons with different shapes cannot have the same radius function. □

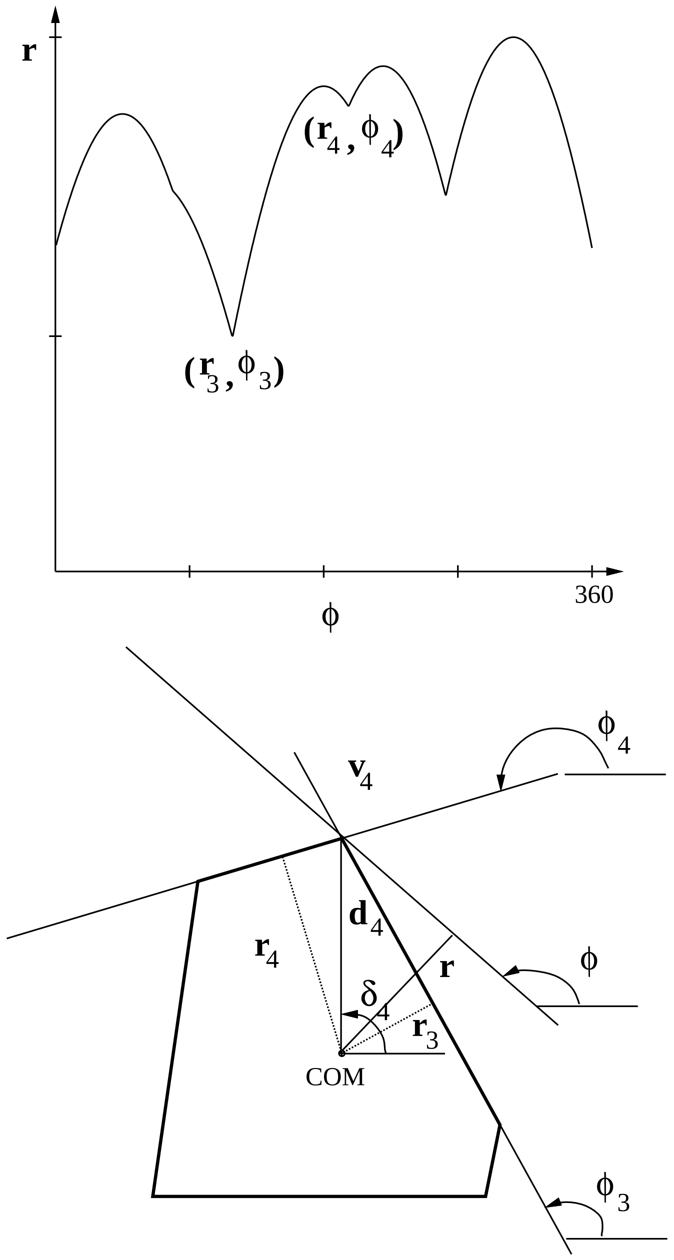


Figure 5.5: Variation in the radius as the fence is rotated. Minima of the radius function correspond to stable edges, and kinks correspond to unstable and metastable edges. The set of minima and kinks uniquely determines the radius function. So the radius function can be reconstructed from the set of minima and kinks.

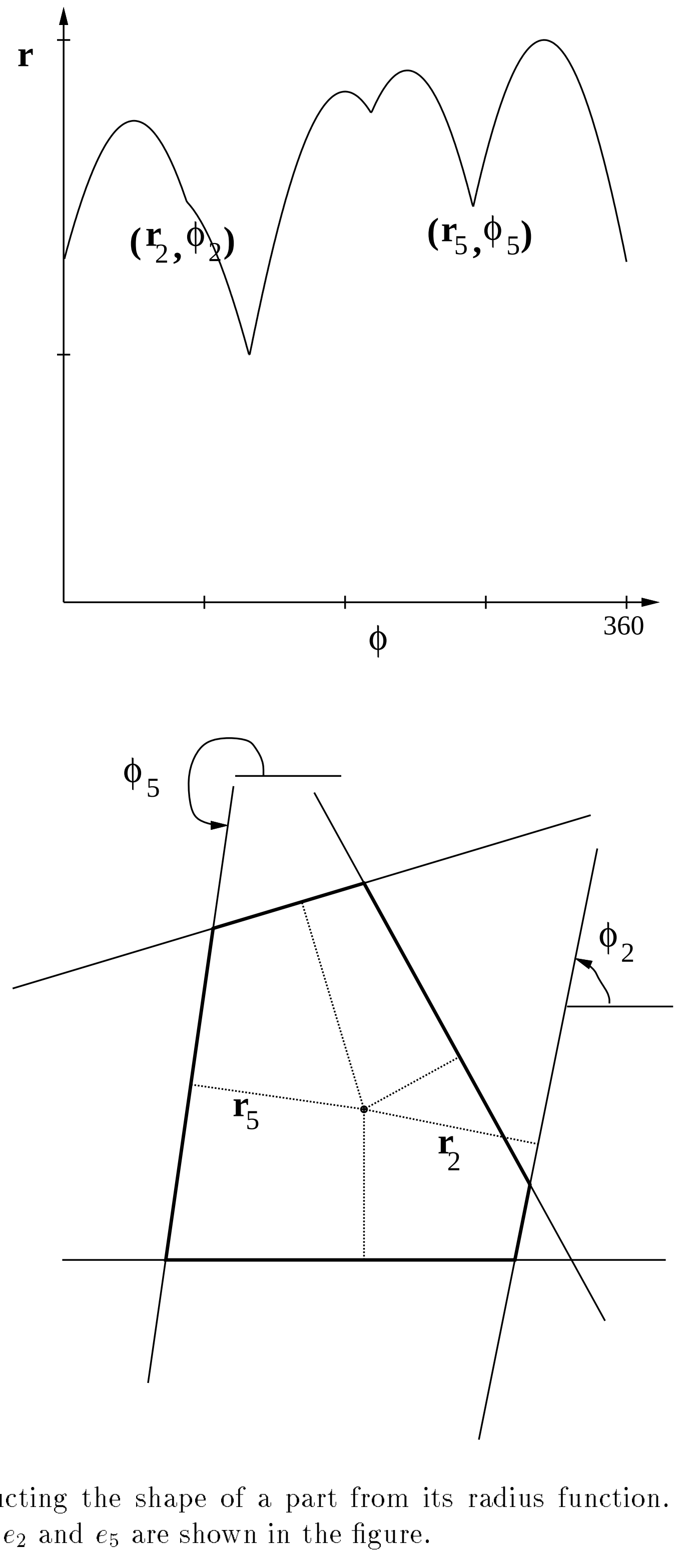


Figure 5.6: Reconstructing the shape of a part from its radius function. The orientations and radius values of edges e_2 and e_5 are shown in the figure.

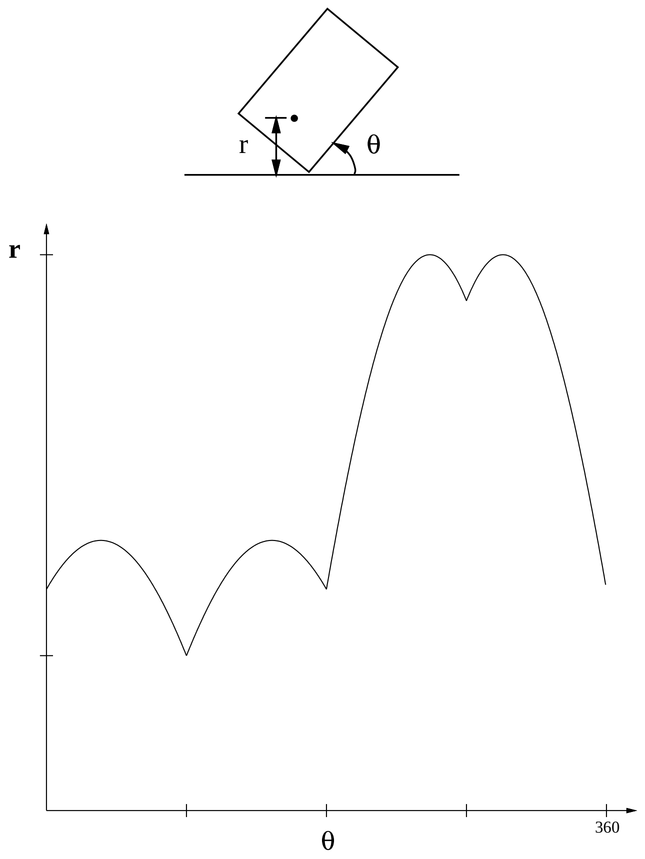


Figure 5.7: A plot of the radius of the rectangle of Figure 5.2 as a function of its orientation relative to a fixed support line, the fence.

The radius function is a support function of a polygon. (See Valentine [180] for an introduction to support functions of convex sets.) Prior results (Santaló [160] for example) show that there is a 1-to-1 mapping between a smooth convex shape and its support function.

5.2.4 Resting Ranges

The radius function gives the polygon radius as a function of the orientation of the support line. A more intuitive representation for push-align operations is to plot the polygon radius as a function of the orientation of the polygon relative to a fixed line, the fence. (See Figure 5.7.)

The *resting range* of a stable orientation of a part is the set of initial orientations for which the part comes to rest in that stable orientation. The stable orientations correspond to the minima of the above function and the resting ranges are defined by the enclosing left and right maxima of the stable orientations on the above function. For convenience, we capture this information in the *resting range diagram* (see Figure 5.8). The x's indicate stable orientations and the vertical bars indicate the limits of the resting range for each stable orientation. The resting range diagram corresponds to a slice along the 90 degree pushing direction of the push stability diagram of Brost [29].

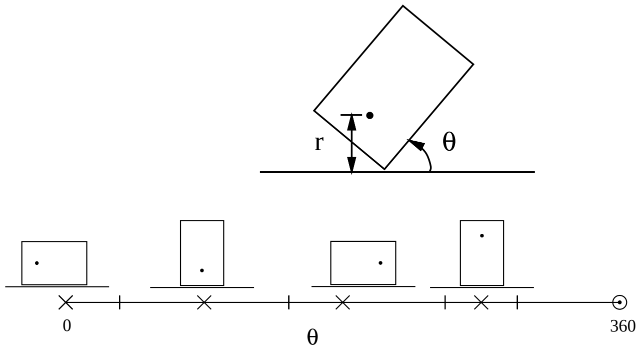


Figure 5.8: Resting range diagram for the rectangle of Figure 5.2 plots the set of initial orientations for which the part comes to rest in each stable orientation. The x's indicate stable orientations of the rectangle for each resting range and the vertical bars indicate the limits of each resting range. The resting range diagram corresponds to a slice of the push stability diagram of Brost [29] along the 90 degree pushing direction.

5.2.5 Finding Representative Actions

Action Ranges for a State

We have to find a set of push-align actions that can be used to generate an orienting plan for a given set of states. A push-align action is indexed by the angle the part is rotated through before being pushed. An action corresponds to a rotation in the range $[0, 360)$. Searching through this continuous space of actions would be difficult. Fortunately, it is also unnecessary. The mechanics of the task dictate that certain ranges of actions are equivalent, that is, all member actions of a class have the same result. Since the actions can be grouped into equivalence classes, we can discretize the action space and use a finite number of representative actions that cover it. Searching in the space of these actions gives us a plan.

Definition 5.3 *An* **action range** *is a contiguous interval of rotations that forms an equivalence class of actions. That is, every action that belongs to an action range results in the same final state for a given initial state.*

Let the stable orientations of states s_i and s_j be ψ_i and ψ_j respectively. Let the left and right limits of the range of stable resting angles for state s_j be λ_j and ρ_j respectively (see Figure 5.9). Each push-align action is identified by the angle the robot rotates the part by. A clockwise rotation of $\psi_j - \psi_i$ enables a transition from state s_i to state s_j. In fact, the entire range of rotations from the enclosing maximum at λ_j to the other enclosing maximum at ρ_j enables this transition. So the action range specified by the open interval $(\lambda_j - \psi_i, \rho_j - \psi_i)$ is an equivalence class for the transition from state s_i to state s_j. Proceeding in this manner and taking care to handle angle wraparound at 360 degrees, we can determine the action ranges for the transitions between every pair of states (Figure 5.10).

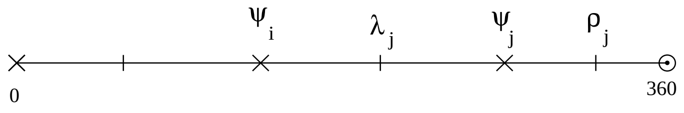


Figure 5.9: The action range for the transition from state s_i to state s_j is computed from the resting range. The action range is $(\lambda_j - \psi_i, \rho_j - \psi_i)$, where ψ_i is the stable orientation of state s_i, and λ_j and ρ_j are the left and right limits of the resting range of state s_j.

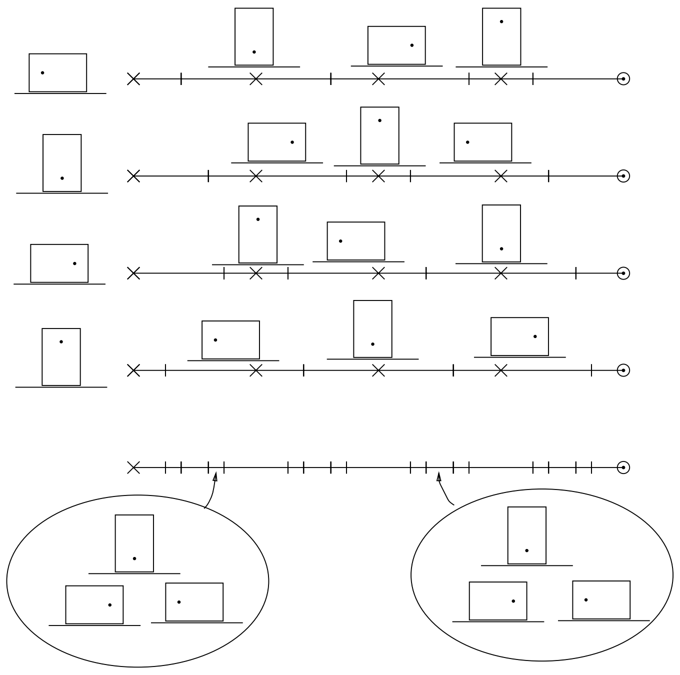


Figure 5.10: Action ranges for the stable states of the rectangle. For each stable state, we indicate its action ranges and the resulting stable states. The overlap ranges for the entire set of stable states are shown at the bottom. For clarity, stable orientations are not shown on the overlap range diagram. The set of possible resulting stable orientations for two selected ranges are also shown.

Action Ranges for a Set of States

From the action ranges of the individual states, we can determine the action ranges for a given set of states. If we overlap the set of intervals corresponding to the action ranges from the individual states, we obtain another set of intervals, the *overlap ranges* (see Figure 5.10). If the part has n stable states and we are computing action ranges for k of them, there are kn action ranges and since the action ranges are contiguous, there are kn overlap ranges. The overlap ranges are also equivalence classes. For the given set of initial orientations, any action chosen from an overlap range results in the same set of orientations. For each overlap range, we can select a representative action, and we thus have kn actions to choose from. The actions contained inside an equivalence class are deterministic. The actions corresponding to endpoints of intervals are nondeterministic since they can have multiple results. To ensure that the representative actions are deterministic, we select the representative action from an equivalence class to be in the middle of the angle interval.

This set of actions covers the space of state transitions $[0, 360)$. There is always an action to get from any state to any other state. This property implies that if a finite length plan exists, a search-based planner can always find it and is hence complete. We will show in Section 5.5 that there are in fact finite bounds on the plan length. A part cannot be oriented uniquely when there is symmetry in the part's push-diameter function, and in such cases a plan always exists to orient the part up to symmetry.

Note that the set of actions covers the space of state transitions $[0, 360)$ except for the finite number of overlap range endpoints. As long as each overlap range has non-zero extent, a deterministic action can be found to the left or right of the endpoint and this does not affect the covering of the action space. In fact, to guarantee robustness of the actions we can specify a minimum size for the overlap ranges for usable actions. Overlap ranges smaller than the minimum size do not yield usable actions. For such cases, we cannot guarantee that there is always an action to get from any state to any other state, or that a plan exists.

We now compute the time taken to generate the representative actions for a set of k states. For each state we determine the action ranges to each of the n stable states. This involves finding the difference of the resting range limits of the destination state and the stable orientation of the start state. Therefore this takes $O(n)$ time for each of the k states. The action ranges can be viewed as a list of ordered range limits, and the overlap ranges can be computed by merging the lists for the k states and sorting them. Since there are kn overlap ranges, a naive implementation takes $O(kn \log kn)$ time. So the time to find the representative actions for a set of k states, $T_a(k, n)$ is $O(kn \log kn)$.

5.2.6 Graph Representation of the Orienting Problem

We can represent the orienting problem for a given part by a directed graph, the *state transition graph* (Figure 5.11). The vertices of the graph are the stable orientations of the part. Each directed arc contains the action range for the state transition from the state at its tail to the state at its head. The sensed value at each vertex depends on the sensor resolution and sensor noise. The planning problem consists of searching this graph to find a sequence of actions to determine the orientation of the part. As before, we can use the action ranges to determine overlap ranges and select representative actions (see Figure 5.12).

For sensor-based orienting, we have to determine the current state of the part based on the executed actions and measured sensor values. For sensorless orienting, we have to find a sequence

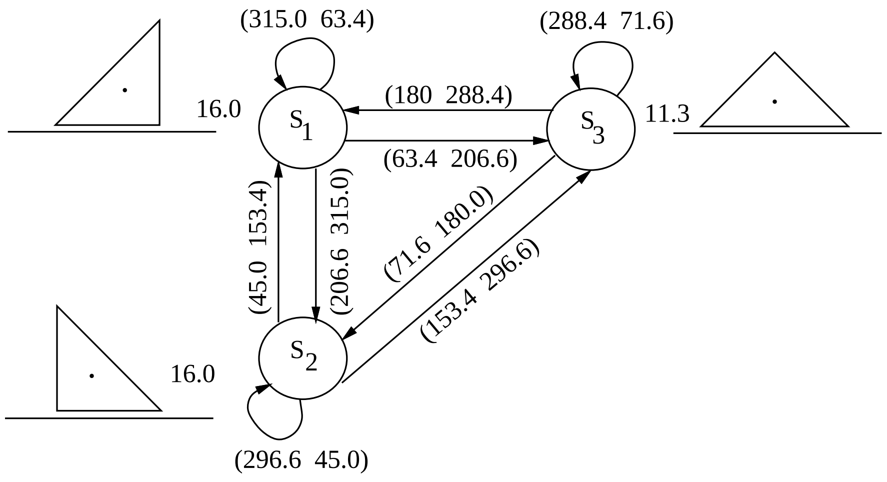


Figure 5.11: State transition graph for an isosceles right triangle. The state transition graph is a directed graph whose edges correspond to action ranges and vertices correspond to stable orientations of the part.

of actions to bring any initial part state to the same final part state.

An alternate representation of the problem is as a finite state machine (FSM) where the machine outputs a signal based on the state it transitions to. In fact, the problem of finding the minimal length orienting plan is the same as finding the minimal length *adaptive homing sequence* of a finite state machine. See the textbook by Kohavi [102] for an introduction to finite state machines.

5.3 Sensor Model and State Distinguishability

To generate sensor-based plans, we have to explicitly consider the information the sensor provides and its ability to distinguish the stable orientations of the part. Our sensor measures the diameter of the aligned part, and these diameter measurements provide only partial information on the part orientation.

The sensor consists of a linear array of LEDs arranged perpendicular to the fence at the side of the conveyor, with a corresponding parallel array of phototransistors on the opposite side of the conveyor (Figure 5.13). See Appendix B for a description of the sensor. The resolution of the sensor is determined by the spacing of the LEDs. When a part is aligned with the fence, we can determine its diameter up to the resolution of the sensor by identifying which LED–phototransistor pairs are blocked by the part. Two stable orientations of the part that have diameters that are beneath the resolution of the sensor are indistinguishable.

In general, sensors are not perfect — they do not give us precise readings. The discriminability of the sensors depends on their resolution and noise characteristics. However by determining the range of sensor values consistent with each state, we can capture the effect of sensor resolution and sensor noise. Two states with overlapping sensor ranges are considered *indistinguishable* and two states whose sensor ranges do not overlap are *distinguishable*. Using this model for the sensor, we can group states with overlapping sensor ranges.

Definition 5.4 *An* **indistinguishable set** *is a set of states where each state is indistinguishable*

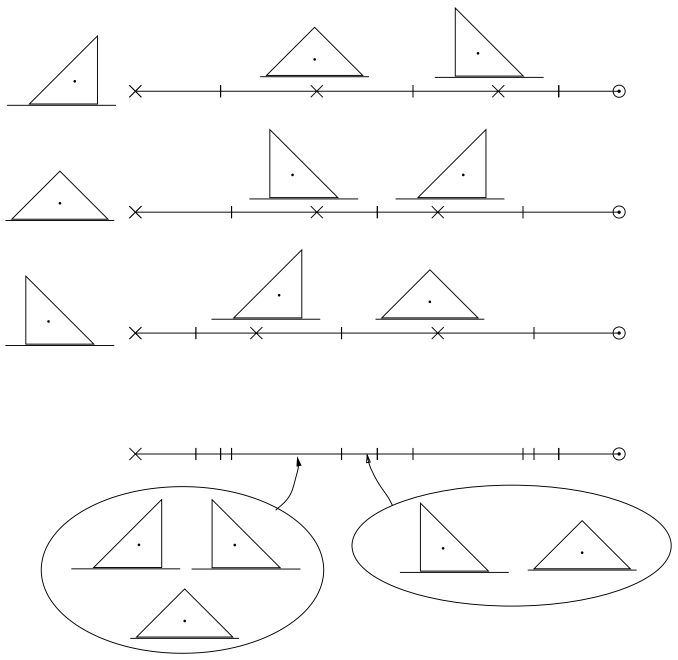


Figure 5.12: Action and overlap ranges for the triangle of Figure 5.11.

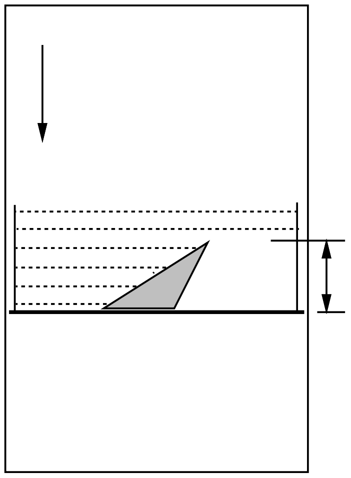


Figure 5.13: Sensor setup. We determine the diameter of the aligned part from the LED-phototransistor pairs it blocks.

from at least one other state. We refer to a set of indistinguishable states with k elements as a k-indistinguishable set.

Note that some of the states in a k-indistinguishable set may be pairwise distinguishable. This is because the indistinguishability relation is not transitive due to sensor noise. Consider three states a, b, and c with increasing diameter values such that a and b are indistinguishable, and b and c are indistinguishable. These states form a 3-indistinguishable set. However a and c will be distinguishable if their diameter values differ sufficiently.

5.4 Sensorless Orienting

Sensorless orienting can be viewed as a form of sensor-based orienting with a very coarse resolution or noisy sensor which cannot distinguish any states. To orient a part without sensors, we must find a sequence of push-align actions that brings all initial orientations of the part to the same final orientation.

5.4.1 Search-based Planning

For sensorless orienting, the initial set of possible states is the set of all stable states. A sensorless plan is a sequence of actions that transforms any initial state to the same final state. An action consists of rotating the part by a chosen angle before it is made to contact the fence. We perform breadth-first search to find a minimum-length sensorless plan (Figures 5.15 and 5.16). (See Rich and Knight [155] for an introduction to search techniques.) We have implemented this planner in Common Lisp.

The search uses two lists, OPEN and CLOSED. Search is initialized with the root node containing the set of all stable states and the root node on the OPEN list. A node is labeled SOLVED when it contains exactly one state. An explored node is one to which all representative actions of its constituent states have been applied. Every new unexplored node is placed at the tail of the OPEN list. At each stage, a node is popped off the head of the OPEN list and moved to the CLOSED list once it has been explored. When a node is labeled SOLVED, this information is transmitted up through its ancestors all the way to the root node. Search terminates when the root subset is labeled SOLVED or the OPEN list becomes empty.

The representative actions for a set of k stable states are obtained from the overlap ranges over the k states (see Figure 5.14). A set with k states has kn representative actions and the time taken to find the representative actions, $T_a(k,n)$, is $O(kn \log kn)$. To determine the result of an action, for each possible orientation of the part, we find its orientation after the specified rotation by the robot. We then compare this part orientation before it contacts the fence with the precomputed resting ranges of the n stable states until we find the resting range that it is a member of and determine the resulting orientation. This is an $O(n)$ operation for each action and each state. So the time taken to find the result of an action on k states, $T_r(k,n)$, is $O(kn)$.

For a part with n stable states, there are 2^n possible subsets of states. There are C^n_k subsets with k states. To find an action sequence to bring all states to one unique state, in the worst case we have to consider every possible subset and apply all of its representative actions. Therefore the total time to evaluate the results of the actions over all possible subsets is $\sum_{k=1}^{n} C^n_k[T_a(k,n) + (kn)T_r(k,n)]$. Therefore the asymptotic running time to find a sensorless plan is $O(n^4 2^n)$.

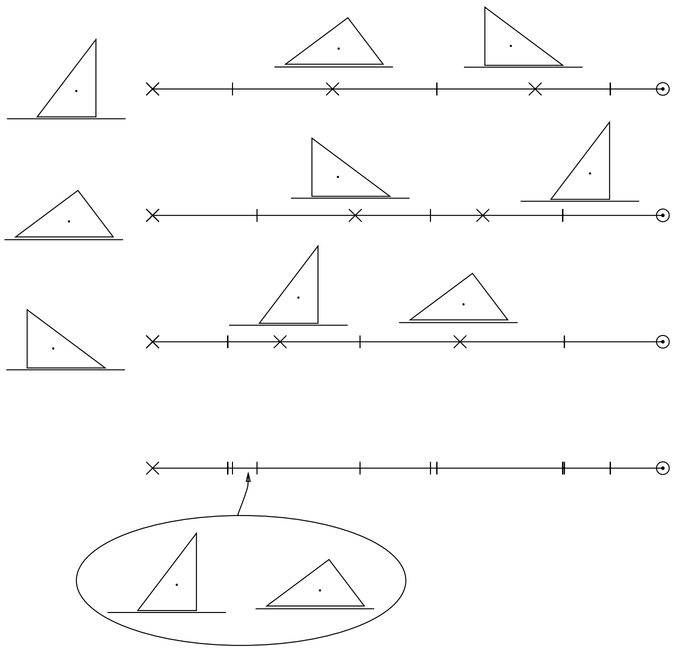


Figure 5.14: Action ranges for each stable state and the overlap ranges for the entire set of stable states for a triangle. An action from the indicated overlap range is used in the plan in Figure 5.15.

A conservative upper bound on the length of the plan is obtained by assuming that the actions cause transitions through all possible subsets before the part is brought to a unique state. Therefore the maximum length of the plan is 2^n. We will see next that the length of a plan is in fact $O(n)$.

5.4.2 Backchaining Algorithm

A natural question is: Is there a more efficient way to generate a sensorless plan? In fact there is a polynomial-time algorithm to find the shortest sequence of push-align actions to bring all initial orientations to the same final orientation. It is the *backchaining algorithm* developed by Goldberg [76] to find the shortest sequence for sensorless orienting of a polygon by frictionless parallel-jaw grasping. As mentioned earlier, the push function is another representation of the push-align operation that describes the resulting part orientation as a function of its initial orientation (Figure 5.3). The push function is a monotonic step function with a period of 360 degrees. Discontinuities in the push function occur at maxima of the radius function. The similar structure of the push function and the squeeze function used in grasping means the backchaining algorithm can be applied to the push function to derive a minimum length sensorless plan for orienting the part by push-aligning. Chen and Ierardi [41] showed that for a part with n stable edges, the backchaining algorithm is guaranteed to find a solution of $O(n)$ steps in $O(n^2)$ time. When the push function has a period of 360 degrees, the part can be oriented uniquely. If the period is less than 360 degrees, the part can be oriented only up to symmetry.

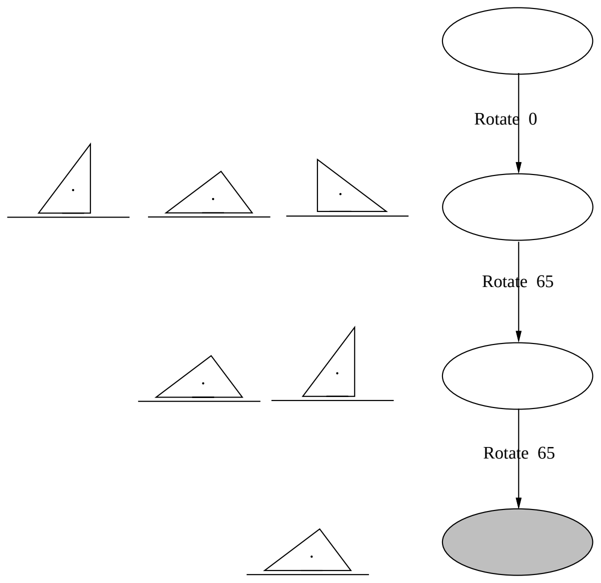


Figure 5.15: Sensorless plan to orient the triangle. Each node contains a set of stable states and each arc represents an action. The goal node, shown shaded, contains a single state.

5.5 Sensor-based Orienting

We now consider generating an orienting plan when the diameter of the part is measured after every push. When the part has a unique diameter value for every edge, a single push-align operation can determine the part orientation (Figure 5.17). When the part has several orientations with the same sensed diameter value (Figure 5.18), we need a plan consisting of a conditional sequence of push-align operations to orient the part (Figure 5.19). Branching during plan execution occurs based on the sensed diameter value. The sensor data reduces the set of possible states to those consistent with the sensed value and the sequence of actions that have been performed. For each k-indistinguishable set of same diameter-valued states to be distinguished, we find the representative actions as described in Section 5.2.5 (see Figure 5.20). For a k-indistinguishable set, there are kn action ranges for the different pairs of states. Therefore there are kn overlap intervals of these action ranges and kn representative actions.

The number of orientations the robot eliminates after each action depends on the actual part orientation, the number of orientations with the same diameter value, and the applied action. Whether the part can be oriented uniquely depends on the characteristics of its push-diameter function (Figure 5.4). If the part has at least one diameter with a unique value or its push-diameter function has a period of 360 degrees, it can be oriented uniquely. If the period of the push-diameter function is less than 360 degrees, the part can be oriented only up to symmetry. If the part has only one diameter value for all its stable edges, the sensor-based plan is then effectively a sensorless plan. The depth of the search tree in the sensor-based case is never greater than that for the sensorless case. In fact we will show that the depth of the search tree is never greater than m, the maximum number of indistinguishable states.

We seek to find plans that are *optimal* in the sense that they minimize the worst-case, or maximum, number of actions to orient a part. We describe algorithms to find such optimal plans

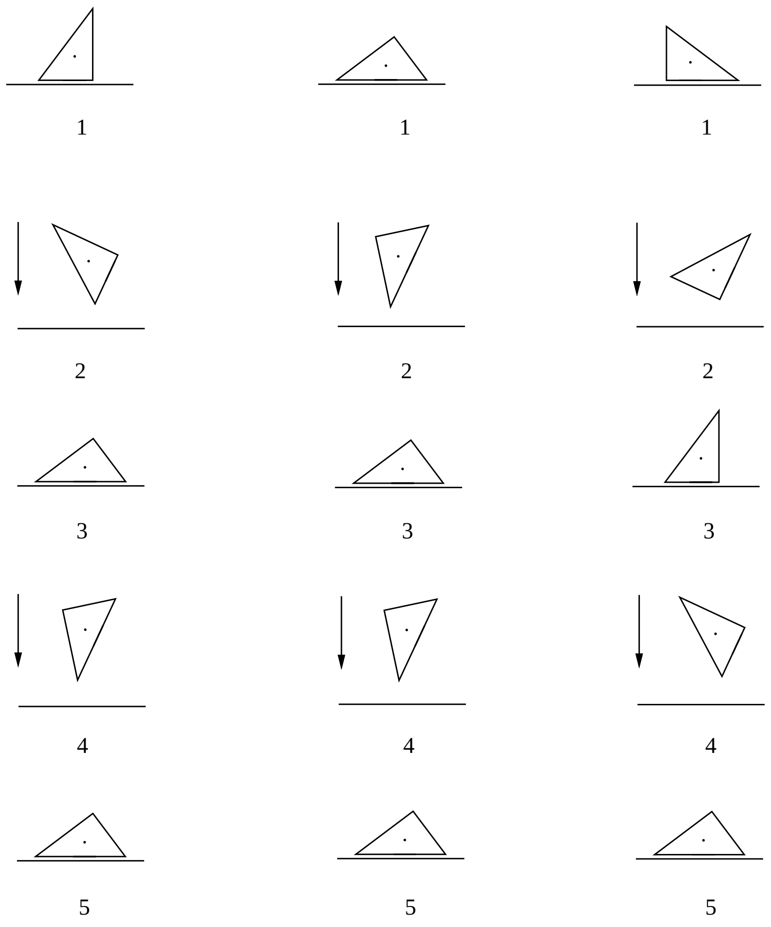


Figure 5.16: Execution trace of the sensorless plan in Figure 5.15.

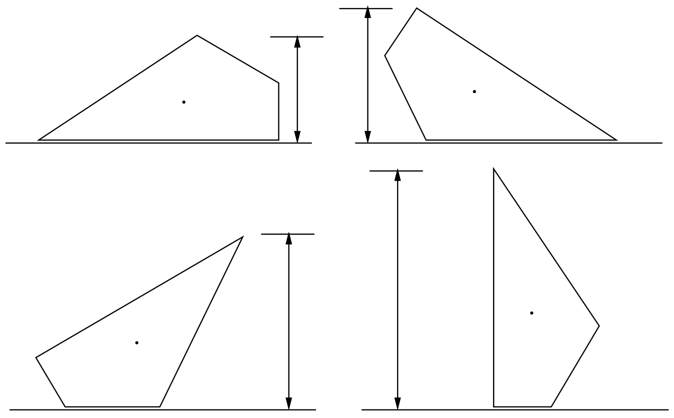


Figure 5.17: A part with unique diameter values at its stable orientations can be oriented in a single step.

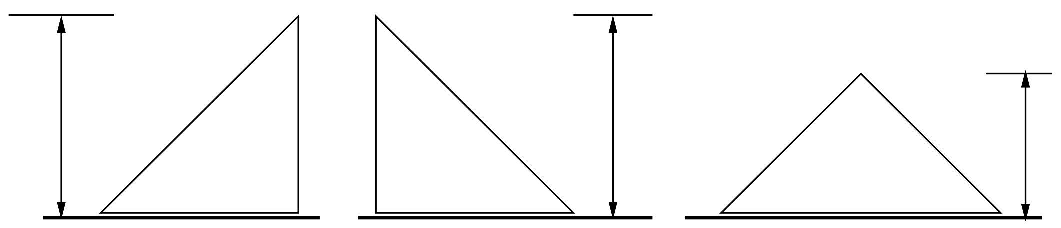


Figure 5.18: The isosceles right triangle has non-unique diameter values at its stable orientations and cannot always be oriented in a single step.

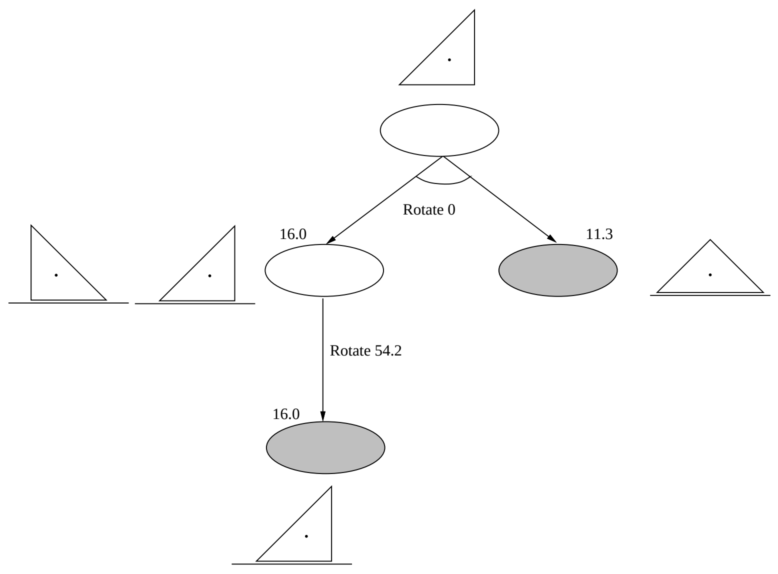


Figure 5.19: A sensor-based plan to orient the isosceles right triangle. Each node contains a set of indistinguishable states. Goal nodes, shown shaded, have a single orientation. The sensor value shown at a node indicates the diameter value for the set of indistinguishable states at that node. For clarity, we indicate the range of sensor values corresponding to these states by the average of the sensor values.

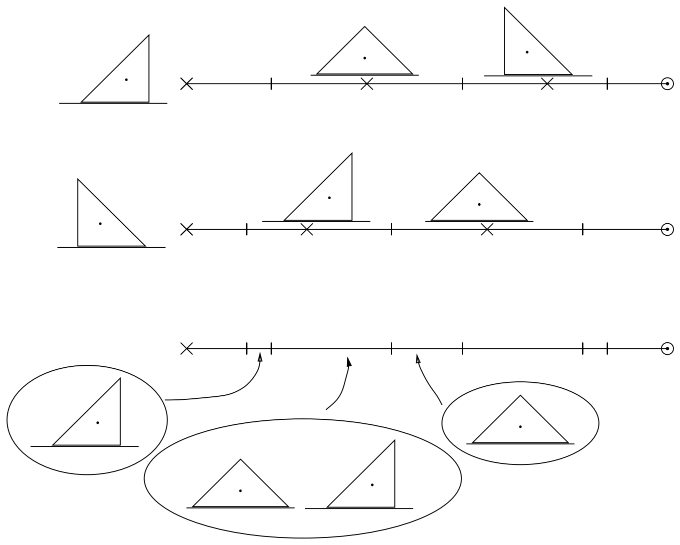


Figure 5.20: Action ranges and overlap ranges for subset of states with the same sensed diameter value for the isosceles right triangle of Figure 5.11.

that minimize the maximum execution length. There are other optimality criterion that can be used. Minimizing the expected number of actions to orient a part can be useful, particularly since the expected number of actions can be significantly smaller than the worst-case number of actions. The expected number of actions depends on the initial distribution of part orientations and the shape of the part. In particular, the sizes of the resting ranges determine which initial stable orientations are most likely. Some of our current algorithms can be modified in a straightforward manner to generate plans that minimize the expected plan length. Such plans require as input an estimate of the initial probability distribution of the part.

We will present three algorithms to find sensor-based orienting plans. The first uses breadth-first AND/OR search, the second is a greedy heuristic based algorithm, and the third is the bottom-to-top algorithm. The search-based algorithm and the bottom-to-top algorithm are exponential-time algorithms that find optimal plans, where an optimal plan is an orienting plan with the shortest maximum length. The greedy algorithm takes polynomial time, but returns potentially suboptimal plans.

5.5.1 AND/OR Search

To find an orienting plan, we perform a breadth-first AND/OR search. (See Rich and Knight [155] for a description of AND/OR graphs and the similar AO* algorithm.) The root node corresponds to the set of all possible orientations of the part. A node in the search tree contains the set of orientations consistent with the push-align operations along the path from the root to the node. All links are AND links and correspond to a push-align action followed by a sensor reading. When an action is applied at a node, all orientations that can result are generated and classified into sets of indistinguishable orientations. The AND link from a node for a given operation points to a set of child nodes where each child node contains a set of indistinguishable orientations.

To group the resulting states into indistinguishable sets, we sort the states by their diameter

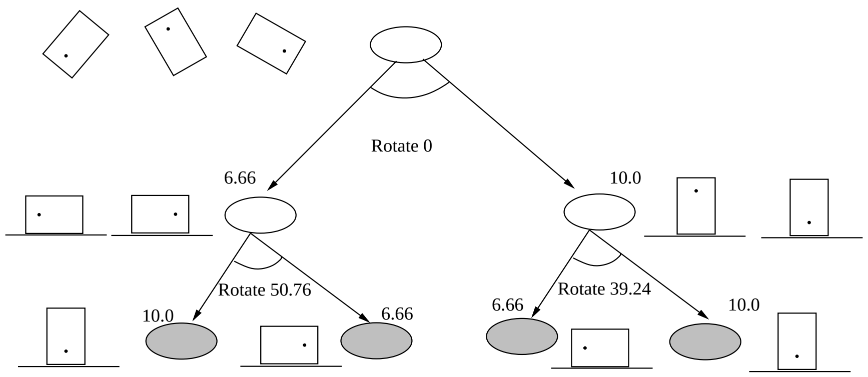


Figure 5.21: A plan generated by the AND/OR search planner to orient the rectangle of Figure 5.2.

values. The first indistinguishable set is initialized to consist of the first state on the ordered list. We then traverse the entire sorted list. If a state is identical to its successor on the list, discard the successor. If the state and its successor are not identical and are indistinguishable, add the successor to the current indistinguishable set and set the successor to be the current state for comparison. If a state and its successor are distinguishable, the current indistinguishable set is complete. We then create a new indistinguishable set whose first member is the successor state. For k states, this can be accomplished in $O(k \log k)$ time. since it takes $O(k \log k)$ time to sort the states by diameter values and $O(k)$ time to traverse the ordered list of states. So the time to group resulting states into indistinguishable sets, $T_g(k)$, is $O(k \log k)$.

For search we use an OPEN and CLOSED list. The OPEN list is initialized to the root node and the CLOSED list is empty. Search begins at the root node, which contains the set of all possible initial part orientations. The first push-align operation results in a set of nodes corresponding to the indistinguishable sets of stable orientations. Any node with a single orientation is a goal node and is labeled solved. The search proceeds in a breadth-first manner by popping a node off the head of the OPEN list and applying the representative actions for its constitutent states, and generating the corresponding set of child nodes. Each node that is generated by an action is placed at the tail of the OPEN list if it has not been generated before. Nodes that have been explored are placed on the CLOSED list. When a node is labeled solved, this information is passed up to its parent nodes. A node is labeled solved if it is a goal node or when all its child nodes from an operation are labeled solved. Search terminates when the root node is solved or the OPEN list becomes empty. The plan recovered from tracing the solved nodes corresponds to a conditional sequence of operations to determine part orientation where the sensed diameter values determine branching (Figure 5.21). The search process is exhaustive, but needs to be performed once off-line for each part.

5.5.2 Plan Length

The plan length and depth of the search tree are determined by the worst-case number of operations required to orient a part. This depends on the part geometry, and in particular, the uniqueness of the diameter values of the stable states and the degree of symmetry of the part. We will use the term *diameter value* to refer to the part diameter when the part is in a stable orientation on the fence and use it interchangeably with sensor value. It is instructive to classify polygonal parts as follows (see Figure 5.22):

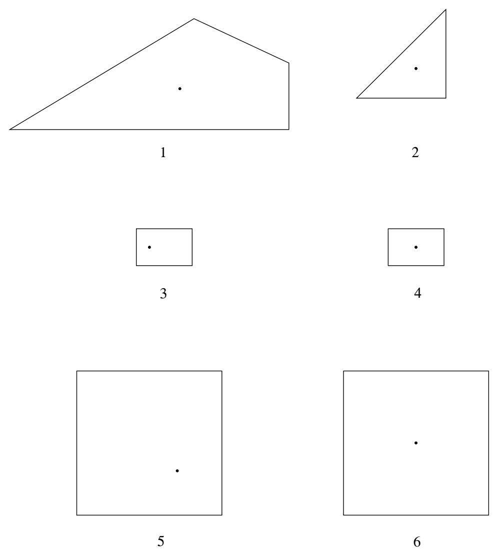


Figure 5.22: Polygons with different worst-case plan length characteristics.

1. Parts with all states having unique diameter values.
2. Parts with some states with unique diameter values.
3. Parts with multiple diameter values, none of which are unique, and an asymmetric push-diameter function.
4. Parts with multiple diameter values and a symmetric push-diameter function.
5. Parts with all states with the same diameter value and an asymmetric push function.
6. Parts with all states with the same diameter value and a symmetric push function.

For each of the above cases, we can determine bounds on the length of the plan in terms of m, the maximum number of stable orientations with the same sensed diameter value, and n, the number of stable edges of the part. When the part has multiple diameter values and an asymmetric push-diameter function, the length of the plan is no greater than m. When all stable orientations have the same sensed diameter value, the plan length is the same as the length of the sensorless plan, which has an upper bound of $2n - 1$ ([41]). When the push function of the part has symmetry, the part can be oriented only up to symmetry. Table 5.1 gives bounds on the plan lengths for different part geometries. We now prove these bounds on the length of the worst-case plan for each case.

1. *All states have unique diameter values*: Any state the part comes to rest in can be identified uniquely from its sensed diameter value. Therefore a single step is sufficient to orient such a part.
2. *Some states with unique diameter values*: Let there be at least one state with a unique diameter value. Group states with the same sensed diameter value into indistinguishable sets. Let the maximum number of states with the same sensed diameter value be m. Assume that

State distinguishability (from sensed stable diameters)	worst case
1. All unique diameter values	1
2. Some unique diameter values	m
3. Multiple diameter values, none unique, asymmetric push-diameter function	m
4. Multiple diameter values, symmetric push-diameter function	m/p
5. Single diameter value, asymmetric push function (sensorless case)	$2n-1$
6. Single diameter value, symmetric push function (sensorless case)	$2n/p-1$

Table 5.1: Plan length for different part geometries: m is the maximum number of stable orientations with the same sensed diameter value, n is number of stable edges of the polygon, p is the number of periods of the push-diameter function.

after the initial alignment, the part is in a state that belongs to an m-indistinguishable set. Consider the action ranges for each candidate state, which give the resulting part orientation and diameter value for each action (Figure 5.20). Moving along the horizontal axis to an angle θ for each state gives us the result of performing an action of θ degrees for that state. If we sweep a line along the action ranges, it is guaranteed there is some value of θ for which at least one of the resulting states is distinguishable from the others. This is because at some $0 \leq \theta \leq 360$, the sweep line will be in the resting range of a unique diameter valued state for at least one of the initial states. Using this value of θ enables us to reduce the number of possible states by at least one. So we have remaining, in the worst case, a set of $m-1$ states to be distinguished. We can recursively continue this process on the resulting set of states, grouping them by sensed diameter value, and for each indistinguishable set, eliminating at least one state with each action, until we are left with a single state. Since with each action we reduce the number of states by at least one, the maximum number of actions to orient the initial set of states is no greater than m. Since every other initial set of states is of size m or smaller, the worst-case number of steps to orient a part is m.

The actual maximum length of a plan depends on the part shape and can be less than m. Note that the number of steps to orient a part at execution time can range from one up to m, depending on the initial resting edge.

3. *Multiple diameter values, none unique, asymmetric push-diameter function*: Assume that the initial state belongs to the largest indistinguishable set of size m. Look at the overlap ranges of states with the same diameter value (Figure 5.23). Using the sweep-line argument shows that there is always some action for which we have different outcomes since the push-diameter function is asymmetric. Since we have multiple diameter values, we can reduce the size of the largest resulting indistinguishable set by at least one. This property, applied recursively to the resulting set of states, shows that the maximum number of steps to orient a part in this case is no greater than m.

   The minimum number of steps to orient a part is always greater than 1 since there is no state with a unique diameter value.

4. *Multiple diameter values, none unique, symmetric push-diameter function*: Let p be the number of periods of the push-diameter function over 360 degrees. Since we can orient the part only up to symmetry, for each indistinguishable set of size k, we effectively have k/p

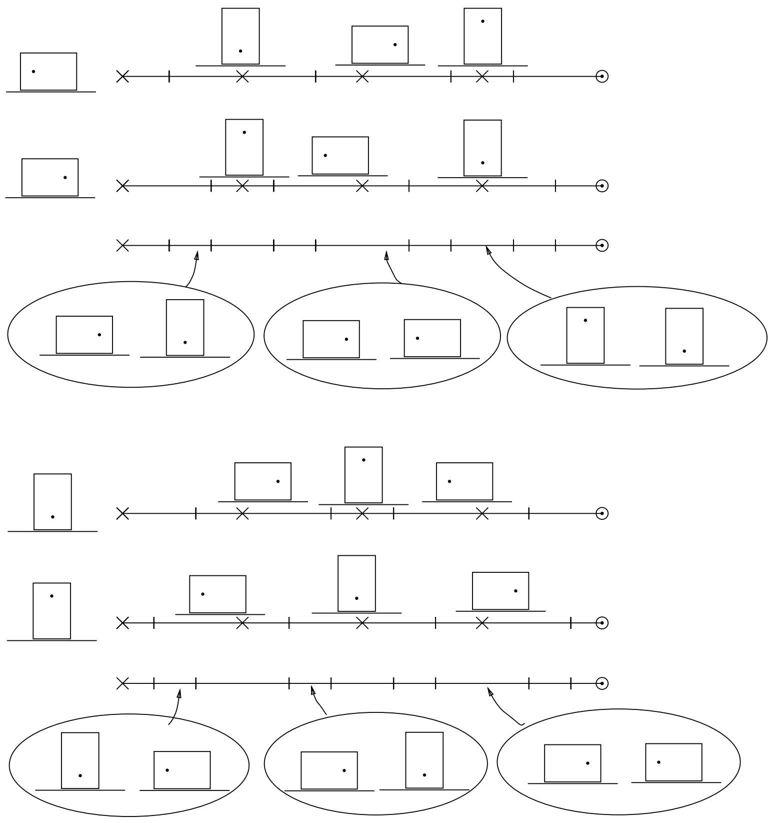


Figure 5.23: Action and overlap ranges for the indistinguishable states of the rectangle. Distinguishing a set of states with the same sensed diameter value for the rectangle of Figure 5.2.

states to consider. We can transform our problem to the case of an asymmetric push diameter function of period $360/p$ with $1/p$ the number of states. It follows that the maximum number of steps to orient the part up to symmetry is m/p.

5. *Same diameter value, asymmetric push function*: When all states have the same sensed diameter value, the sensor does not provide any useful information and the problem is identical to the sensorless orienting problem. Using the backchaining algorithm [76] we can show that a plan always exists for any part with an asymmetric push function. The maximum length of the plan is $2n-1$ steps ([41]). Here the minimum number of steps, expected number of steps, and the worst-case number of steps to orient the part are identical.

6. *Same diameter value, symmetric push function*: This case again corresponds to the sensorless orienting case. Since there is symmetry in the push function, the part can be oriented only up to symmetry. If p is the number of periods of the push function, we can transform the problem to one with a push function of period $360/p$ and n/p states. The maximum number of steps to orient the part up to symmetry is then $2n/p-1$.

5.5.3 Completeness

The search-based planner is complete. For each indistinguishable set explored during search, the procedure to generate representative actions (Section 5.2.5) returns a set of actions that cover the action space. The search procedure of Section 5.5.1 guarantees all representative actions of an explored indistinguishable set are applied to it. The planner performs breadth-first search, and the termination conditions ensure that the planner either finds the solution when one exists or indicates failure when none exists. These conditions guarantee completeness of the planner. The search procedure is exponential in complexity. The maximum depth of the search tree is m, the size of the largest indistinguishable set, since no more than m actions are required to orient a part.

5.5.4 Greedy Algorithm

The search based planner returns the minimum length plan, but takes exponential time. A natural question is: Is there a polynomial-time algorithm that can give us an orienting plan?

Assume we have a part with multiple diameter values. For any k-indistinguishable set, there always exists an action to eliminate at least one of the states. We can use this property to develop a polynomial-time greedy algorithm to generate orienting plans. Let the maximum number of states with the same sensed diameter value be m. Starting with m-indistinguishable set, we can find an action to eliminate at least one state and yield an $(m-1)$-indistinguishable set. Continuing this process of eliminating at least one state with each action, we need no more than m actions to orient a part in any initial orientation. A k-indistinguishable set has kn representative actions. To select an action, we could use one of the following heuristics: maximize the number of indistinguishable sets, minimize the total number of distinct possible resulting states, minimize the size of the largest indistinguishable set. We use the heuristic that maximizes the number of indistinguishable sets to select the action. Ties are broken by selecting the action that minimizes the number of distinct states. This heuristic guarantees that the indistinguishable sets resulting from the selected action are always of smaller size than the initial set. However it is not perfect. For example, if an action transforms all initial states to a single state, the heuristic will not select this action. The plans generated using the greedy heuristic are not guaranteed to be minimum length plans.

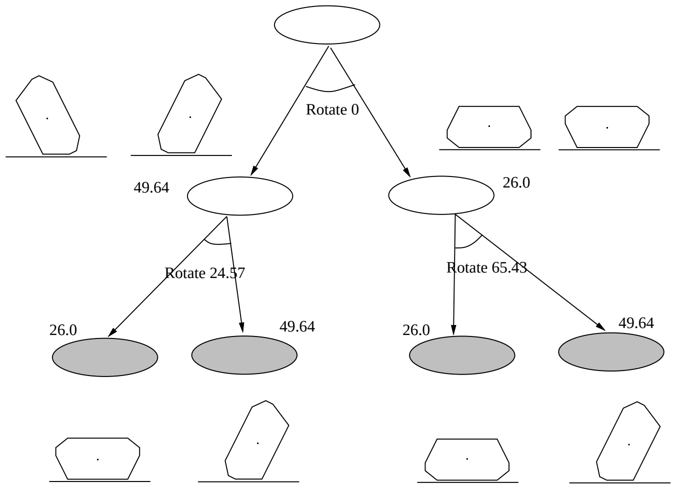


Figure 5.24: Plan generated by the greedy algorithm for a connector. The worst case, best case, and expected lengths are the same for this part.

The plan generation procedure proceeds as follows. We group the set of possible initial states into indistinguishable sets. For each set, we find the set of representative actions to apply. For each representative action, we find the resultant set of states. Using the heuristic, we select the best action for the indistinguishable set under consideration. We recursively continue this process for each indistinguishable set until all initial orientations lead to known part orientations.

We now find the time $T(k)$ to generate a distinguishing sequence of actions for an indistinguishable set of size k. We generate the representative actions for the indistinguishable set in time $T_a(k, n)$ and for each of the kn representative actions, compute the result in time $T_r(k, n)$ and group the resulting states in time $T_g(k)$. The comparison time to select the best action using the heuristic is $O(kn)$. Assume that with each action we eliminate only one state from consideration, resulting in an indistinguishable set of size $k - 1$. So the recurrence is

$$T(k) = T_a(k, n) + kn\{T_r(k, n) + T_g(k)\} + O(kn) + T(k - 1).$$

The time to generate the representative actions for a set of k states, $T_a(k, n)$ is $O(kn \log kn)$. The time to determine the results of an action for a set of k states, $T_r(k, n)$, is $O(kn)$. The time to group resulting states into indistinguishable sets, $T_g(k)$, is $O(k \log k)$. We solve the recurrence to find $T(k)$ is $O(n^2k^3)$. The total running time is $\sum_i O(n^2 n_i^3)$ where n_i is the number of states with the ith sensor value. The running time is clearly bounded by $O(n^3m^3)$.

When all stable states of the part share the same sensed diameter value, the heuristic that seeks to maximize the number of indistinguishable sets is not effective. The heuristic that seeks to minimize the number of states is more effective although it does not guarantee that the child indistinguishable sets are always of smaller size and can result in ties. Ties can be broken by randomly selecting an action that does not result in a set of states that is an ancestor of the current set of states.

5.5.5 Bottom-to-top Algorithm

We now present another algorithm that returns the shortest length plan to orient a part. As before, we refer to a set of indistinguishable states with k elements as a k-indistinguishable set. We find the best plan by finding the best actions to distinguish every possible indistinguishable set, and using them to select the best sequence of actions for each of the initial indistinguishable sets. Let there be n_i states which share the ith sensed diameter value. There are 2^{n_i} indistinguishable sets with the ith value to be distinguished. We begin by finding the best actions to distinguish the smallest indistinguishable sets, with 2 elements. The best action for an indistinguishable set is the one that minimizes the maximum length over the resulting child indistinguishable sets. (In case of a tie, we select the action with a smaller rotation.) The length associated with an indistinguishable set is the worst-case number of actions to distinguish it, and is one plus the maximum length of its child indistinguishable sets from its best action. We find the best actions for all 2-indistinguishable sets over all sensor values, avoiding actions that lead to child indistinguishable sets of the same size. We then find the best actions for all 3-indistinguishable sets. We continue this process of sequentially working on larger indistinguishable sets until we find the best actions for all the indistinguishable sets. Since the child indistinguishable sets are smaller or of the same size as their parent indistinguishable set, by proceeding from the smallest to the largest indistinguishable sets, at each level we have available the best actions and lengths for all the child indistinguishable sets. We store the best action as well as pointers to the resulting child indistinguishable sets during this process. Once all the initial indistinguishable sets have been processed, we work backwards, noting the actions and the resulting indistinguishable sets to create a conditional plan.

Consider the n_i states corresponding to the ith sensed diameter value. A subset of k of these states forms a k-indistinguishable set, with kn representative actions. We generate the representative actions for the indistinguishable set in time $T_a(k,n)$ and for each of the kn representative actions, compute the result in time $T_r(k,n)$ and group the resulting states in time $T_g(k)$. The comparison time to select the best action is $O(kn)$. The time $T(k)$ to find the best action for a k-indistinguishable set is $T_a(k,n) + (kn)(T_r(k,n) + T_g(k)) + O(kn)$. The time to find the best actions for all k-indistinguishable sets with the ith sensed diameter value is $\{T_a(k,n) + (kn)[T_r(k,n) + T_g(k)] + O(kn)\}C_k^{n_i}$.

The time to compute the best actions over all subsets over all sensor values is

$$\sum_i \sum_{k=2}^{n_i} \{T_a(k,n) + (kn)[T_r(k,n) + T_g(k)] + O(kn)\}C_k^{n_i}$$

$$= \sum_i \sum_{k=2}^{n_i} \{O(kn \log kn) + kn(O(kn) + O(k \log k)) + O(kn)\}C_k^{n_i}$$

$$= \sum_i O(n^2 n_i^2 2^{n_i}).$$

Since m is the maximum number of states with the same diameter value, the time complexity of the algorithm is bounded by $O(n^3 m^2 2^m)$.

We have implemented this algorithm in Common Lisp. See example plan in Figure 5.25. As we will see in Section 5.7, this algorithm can be extended to work on multiple parts.

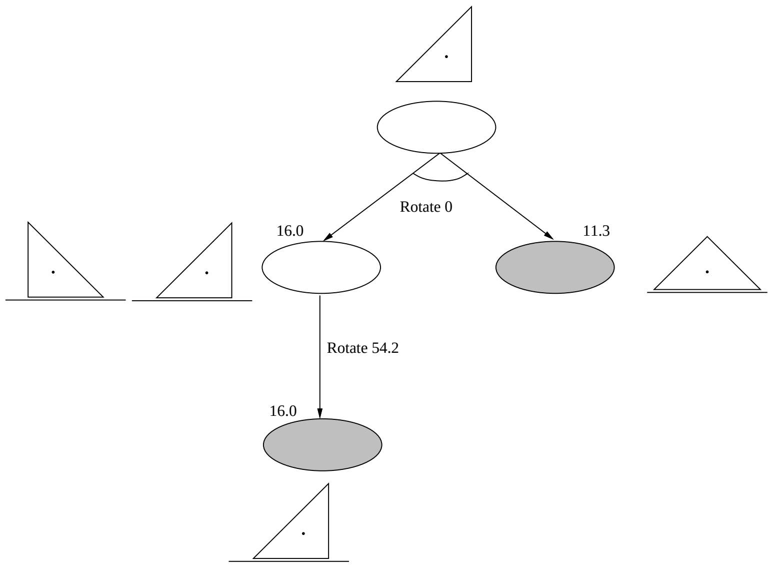


Figure 5.25: Plan generated by the bottom-to-top algorithm for the isosceles right triangle.

5.6 Complexity of Orienting Problem

An important aspect of the orienting problem is the complexity of finding the shortest plan. Is there a polynomial-time algorithm to find the minimum length orienting plan? While we have not answered this question, looking at other versions of the problem suggest that the problem is NP-hard. (See Garey and Johnson [73] for an introduction to NP-completeness.)

Govindan and Rao [81] have worked on a very similar problem of determining the minimum length plan to recognize and orient a part by parallel-jaw grasping where the jaw diameter is sensed after each grasp. They suggest that the problem of finding the minimum length plan for their problem is NP-hard. This suggests that our problem of orienting by push-aligning is also NP-hard.

Consider a representation of the push-diameter function as a deterministic finite state machine. (See [102] for an introduction to finite state machines.) Each step of the push-diameter function can be viewed as a state encoded by its diameter value, angle range, and resultant orientation; the states have a cyclic arrangement, and only the diameter can be sensed. Identifying the shortest sequence of operations to determine the part orientation is the same as identifying a minimum length *adaptive homing sequence* for this finite state machine. The homing sequence for a finite state machine is an input sequence that permits the final state of the machine to be determined from its final output. In an adaptive homing sequence, the choice of each input depends on the previous machine outputs and permits the state of the machine to be determined from the machine output as the input string is processed. While the problem of finding the homing sequence of minimum length for an arbitrary finite state machine is NP-complete, determining if a polynomial-time algorithm exists to find the shortest adaptive homing sequence for this finite state machine is an open problem.

Consider a similar problem of character identification in a string. A simplified representation of the push-diameter function is one where the entire function is viewed as a cyclic string where

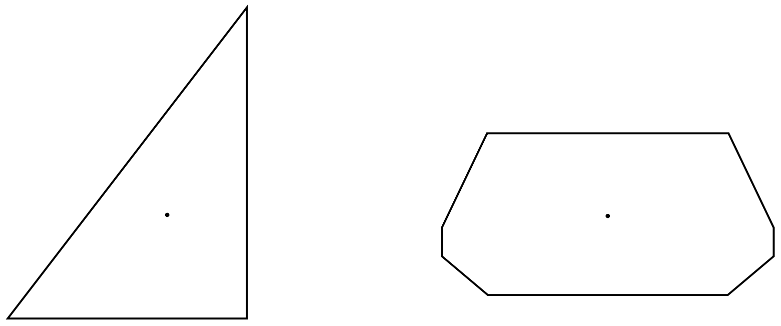


Figure 5.26: Can these two parts be oriented using the same plan?

each location corresponds to a stable state and where each character corresponds to a sensor value. Determining the orientation of the part corresponds to knowing the location of a character in a string with possible multiple repetitions of characters. Determining if there is a polynomial-time algorithm to find the shortest sequence of "jumps" to identify where in the string we are currently located is also an open problem.

5.7 Orienting Multiple Part Shapes

We now shift focus to the problem of orienting any part from a known set of parts using the same plan. During product changeovers or when different parts are processed on the same line, parts with different shapes may have to be oriented by the same feeder (Figure 5.26). Consequently we ask the question: Can our parts orienting system orient multiple part shapes with a single plan?

There are really two aspects to this problem. The first is orienting the input part, which means bringing each part shape to a selected orientation. The second is recognizing the input part, which is identifying the shape that has been oriented.

5.7.1 Properties of Push and Push-diameter Functions

The effects of an action for a part can be determined using its radius function. To orient multiple part shapes, we need to clarify the relation between part shape and the radius, push, and push-diameter functions. We proved in Section 5.2.3 that there is a 1-to-1 correspondence between the set of convex polygons and the set of radius functions of convex polygons. Does this mean that each push-diameter function maps to a single convex polygonal shape? Does each push function map to a single convex polygonal shape? Can any polygon from a known set of convex polygons be identified by a sequence of a push-align operations?

In contrast to the radius function, the push function of a convex polygon is not uniquely defined by the part's shape. Multiple parts, each with a different radius function, can all have the same push function. Further, multiple parts with different radius functions can all have the same push-diameter function.

Lemma 5.4 *Convex polygons with different radius functions can have the same push function.*

Proof: Proof by example. Consider a set of similar parts, where each part is a scaled copy of the other parts. Each part has a unique radius function, but all parts in the set have the same push function. □

We now prove a stronger result that an infinite set of non-similar convex polygons can have the same push function.

Theorem 5.5 *For every convex polygon, there exists another non-similar convex polygon having the same push function. In fact, there is an infinite set of non-similar convex polygons having the same push function.*

Proof: We present a constructive proof that given any convex polygon, we can always generate another non-similar convex polygon having the same push function. The proof is based on showing that we can always generate a new transition vertex from any transition vertex of the given polygon without altering the polygon push function. A *transition vertex* is a polygon vertex at which a local maximum of the radius function occurs; every polygon has at least one transition vertex. We show that we can generate an infinite set of new transition vertices for each original transition vertex, and can therefore generate an infinite set of convex polygons with the same push function.

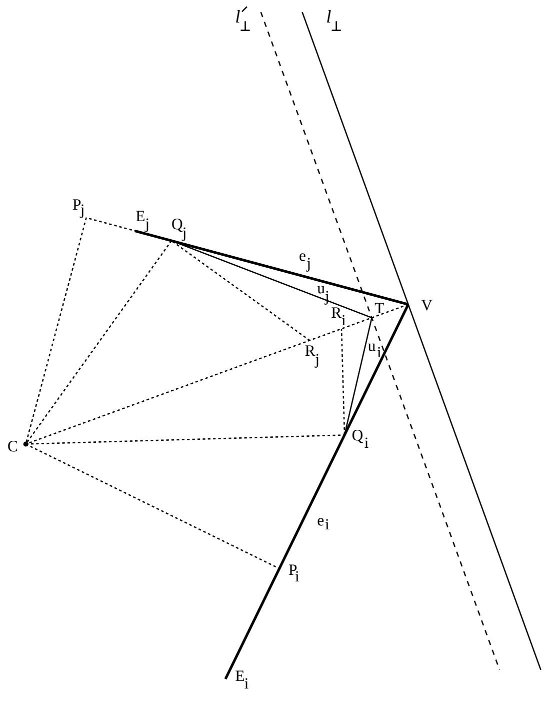


Figure 5.27: Generating a new polygon with the same push function as the given polygon. The edges e_i and e_j of the original polygon, drawn bold, meet at the transition vertex V. The construction splices in unstable edges u_i and u_j so the new vertices Q_i and Q_j and transition vertex T replace vertex V.

Consider a transition vertex V of a polygon with the incident edges e_i and e_j (Figure 5.27). Let the edges be defined by the line segments VE_i and VE_j respectively. Draw the line segment VC that connects V to the polygon center of mass C. Draw the perpendicular lines to the edges

e_i and e_j from C to obtain points P_i and P_j respectively. If P_i is in the interior of the segment VE_i, then e_i is a stable edge. When the perpendicular does not intersect the edge in its interior, the edge is unstable, as in the case of P_j and e_j. A local maximum of the radius function occurs at the transition vertex V when the support line is perpendicular to the line VC. Call this line $l_\perp$.

Our construction adds two new unstable edges to the convex polygon that meet at a new transition vertex such that the modified convex polygon has the same push function as the given polygon. Pick a point Q_i in the interior of both line segments VP_i and VE_i. Draw a perpendicular line through Q_i to intersect VC at the point R_i. R_i lies in the interior of VC since P_i and E_i lie on the same side of the line $l_\perp$. Any line segment with one endpoint in the interior of the segment VR_i and with Q_i as the other endpoint will form an edge whose included angle with CQ_i is greater than 90 degrees. Such an edge is unstable since the perpendicular projection of the center of mass C cannot lie in its interior. A similar construction gives us the points Q_j and R_j for the edge e_j, and another set of line segments that can form unstable edges.

We now pick a point T in the interior of the segment formed by the intersection of VR_i and VR_j to be the new transition vertex. The line segments TQ_i and TQ_j identify the new unstable edges u_i and u_j respectively. Since adding these unstable edges to the polygon does not change the stable orientations of the polygon, the stable orientations in the push function do not change.

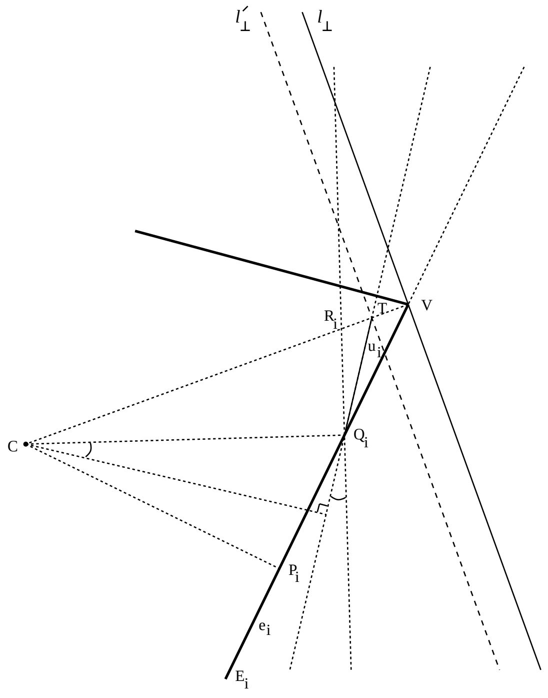


Figure 5.28: Establishing that a local maximum of the radius function occurs at the new transition vertex T.

We must also prove that there is no change in the divergent orientations of the push function, which correspond to local maxima of the radius function. When the support line is in contact with T, the maximum radius value occurs when the line is perpendicular to CT. This line $l'_\perp$ is parallel to $l_\perp$, and so this maximum occurs at the same orientation as the original divergent orientation corresponding to vertex V. To establish that T is a transition vertex with this divergent orientation, we prove that this maximum radius value, length(CT), is the locally maximal radius of the modified polygon. We must show that length(CT) is greater than the radius values of the edges e_i, e_j, u_i, and u_j, and that the radius values of edges u_i and u_j are greater than those of edges e_i and e_j respectively.

We prove the radius is locally maximal at vertex T with respect to edges e_i and u_i (see Figure 5.28). The radius value of edge e_i, r_i, is equal to length(CP_i). Let the radius of edge u_i be r_{u_i}. The radius of segment Q_iR_i is length(CQ_i). Consider a line rotating about the point Q_i, with its initial orientation along Q_iR_i, the perpendicular to the line CQ_i. The radius of the line is proportional to the cosine of the angle it is rotated by, with the maximum at its initial orientation. Therefore length$(CQ_i) > r_{u_i} > r_i$. The maximum radius at vertex T, r_{max}, is length(CT). From $\triangle CQ_iT$, length$(CT) >$ length(CQ_i), which implies $r_{max} > r_{u_i} > r_i$. Similarly we can show $r_{max} > r_{u_j} > r_j$. Therefore the vertex T corresponds to a local maximum of the radius function.

The new vertex T is a transition vertex while the vertices Q_i and Q_j are not. Therefore the new polygon with vertices Q_i, T, and Q_j replacing the vertex V is a convex polygon with the same push function as the given convex polygon. Since there is an infinite set of choices for Q_i, Q_j, and T, we can generate an infinite set of polygons having the same push function as the original polygon. □

See Figure 5.29 for an example of non-similar parts with the same push function. The part at the bottom was constructed from the part at the top using the above construction.

Theorem 5.6 *Convex polygons with different shapes can have the same push-diameter function.*

Proof: Proof by example. See Figure 5.30. □

Since multiple part shapes with different radius functions can share the same push-diameter function, given a valid push-diameter function we cannot always uniquely reconstruct the part from which the push-diameter function was generated. For two parts to share the same push-diameter function, the stable edges of the two parts must have the same orientations, and their divergent orientations, specified by the transition vertices at which radius maxima occur, must also be identical. The diameter values of the stable edges of a given part are fixed, which imposes additional constraints on the vertices that define the diameters of the stable edges.

Theorems 5.5 and 5.6 show that parts with different radius functions can share the same push function, and can also share the same push-diameter function. Therefore when orienting multiple part shapes, differences in part shape do not guarantee that the parts can be identified.

5.7.2 Orientability and Recognizability of Multiple Parts

We wish to orient and recognize any part from a known set of parts using a single plan. When can we guarantee that any part from a known set of parts can be oriented and recognized using the same plan? For example, two parts with the same push-diameter function can be oriented, but can they be distinguished? Here we identify conditions for a set of parts to be orientable and recognizable. We begin by considering pairs of parts and then extend our results to a set of parts.

Definition 5.5 *Two parts are said to be* **distinguishable** *if there exists an action sequence such that the output strings of stable orientation diameter values for the two parts differ.*

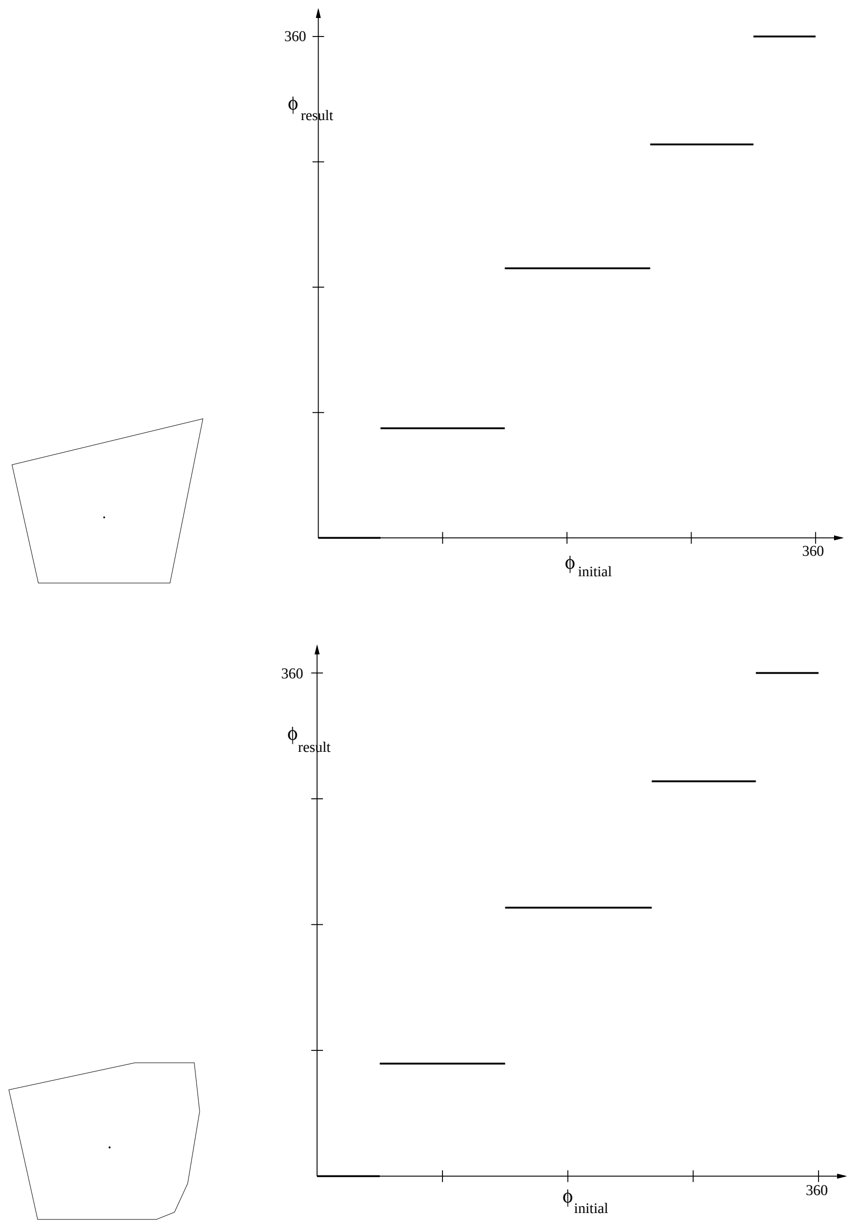


Figure 5.29: Two parts with the same push function. The push-diameter functions of these two parts differ.

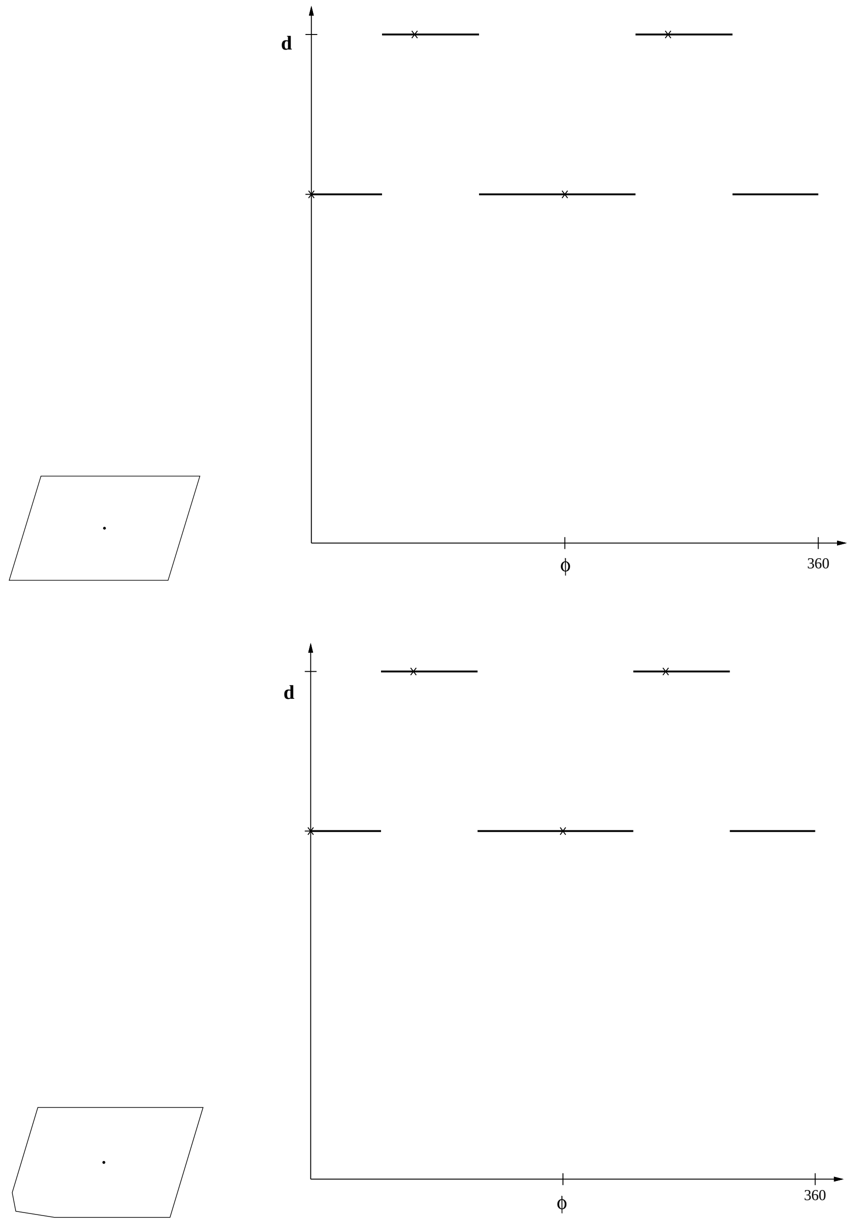


Figure 5.30: Two parts with the same push-diameter function. Their push functions are also identical.

Lemma 5.7 *Two parts with identical push-diameter functions are orientable but they are not distinguishable.*

Proof: Since the two parts have the same push-diameter function, their action ranges are identical and hence their mechanical behavior is identical. The same orienting plan can be used for both parts. However the diameter values of corresponding states are also identical. Therefore there is no action sequence that can distinguish the parts. □

When can two parts be oriented and distinguished? From Lemma 5.7, a necessary condition is that their push-diameter functions be different. When two parts with different push-diameter functions have the same diameter value for all their stable states, the sensor provides zero information and this is equivalent to the sensorless case. Even if we can orient the parts, we cannot tell them apart. A simple example is two squares of the same size with their centers of mass at different points. Therefore a second necessary condition is that either the parts have different diameter values or at least one of the parts has multiple sensed diameter values. A sufficient condition for two parts to be orientable and distinguishable is that each part have at least one unique diameter value.

Theorem 5.8 *Two parts with different push-diameter functions are always orientable and distinguishable if $|D| > 1$, where D is the set of distinct diameter values of all stable orientations of the two parts.*

Proof: Since $|D| > 1$, there are multiple distinct diameter values. Either both parts have the same set of multiple diameter values, or one or both parts have some diameter values that are unique to them.

When one or both parts have some diameter values unique to them, we can distinguish the parts by bringing them to the corresponding stable orientations.

Now consider the case when the two parts both have the same set of multiple diameter values. Since the push-diameter functions of the two parts differ, there is at least one pair of states whose action ranges differ. The most difficult distinguishability case occurs when the two parts have almost identical push-diameter functions. We will treat the case when each part has the same two diameter values (see Figure 5.31); the argument can be extended to parts with more than two identical diameter values. It is hard to tell the parts apart because every state in one part is paired with a corresponding state in the other part. Since the action ranges of at least one pair of states differs, there is some action for which the parts transition from this pair of states to two states that are not paired. This action causes the parts to go "out of phase", so to speak. Once the two parts are in states that are not paired, there is an action sequence guaranteed to bring the two parts to distinguishable states. □

Lemma 5.9 *Distinguishability of parts is not transitive.*

Proof: We prove this with an example. Consider three parts P_A, P_B, and P_C. Let the push-diameter functions of parts P_A and P_B be such that they are orientable and distinguishable. Similarly, let parts P_B and P_C be orientable and distinguishable. However parts P_A and P_C can have the same push-diameter function, in which case they are not distinguishable. □

Definition 5.6 *A set of known parts P is said to be* **recognizable** *if there exists an action sequence to distinguish every member of the set from every other member of the set.*

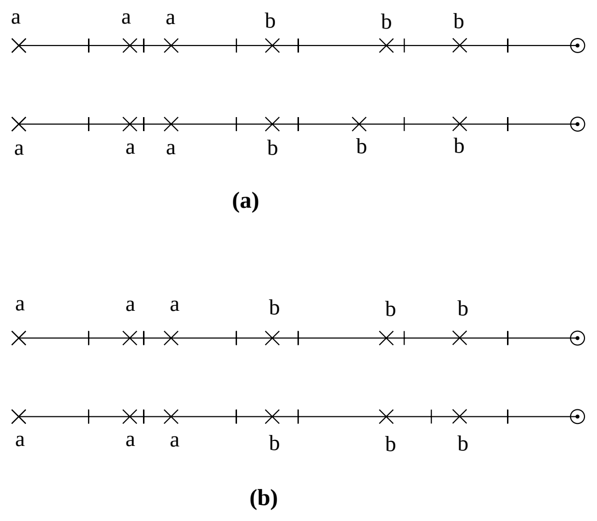


Figure 5.31: Resting range diagrams of two parts with almost identical push-diameter functions. All states with the same character are indistinguishable states. In both cases the parts can be distinguished. (a) Resting ranges of the two parts are identical, but stable orientations differ. There is one pair of states with different action ranges. (b) Resting ranges of the two parts differ, but stable orientations are identical. The action ranges of every pair of states differ.

Lemma 5.9 implies that a necessary condition for any set of parts to be recognizable is that every pair of parts be distinguishable.

Theorem 5.10 *For a given set of parts P, if every pair of parts is orientable and distinguishable, then any subset of parts $S, S \subseteq P$ is orientable and recognizable.*

Proof: Since every pair of parts can be oriented and distinguished, no two parts have the same push-diameter function. Therefore each part has a state whose action range differs from at least one state in every other part. As in Theorem 5.8, we can show that an action sequence always exists to rotate parts "out of phase" with the others one at time and use this property to recognize them. $\square$

5.7.3 Sensor-based Orienting and Recognizing of Multiple Parts

We have extended the bottom-to-top planner of Section 5.5.5 to generate orienting plans for multiple parts. There are two modifications for this. First, a state is now encoded by the part it belongs to in addition to its stable orientation and diameter value. Second, indistinguishable sets now consist of states with the same sensed diameter value over all parts. The representative actions are found from overlap ranges generated using the action ranges of states belonging to these indistinguishable sets. An example plan is shown in Figure 5.32.

5.7.4 Sensorless Orienting of Multiple Parts

A natural next question is: Can multiple parts be oriented by sensorless strategies? The answer is yes for certain cases. A trivial example is when the set of parts consists of scaled versions of the

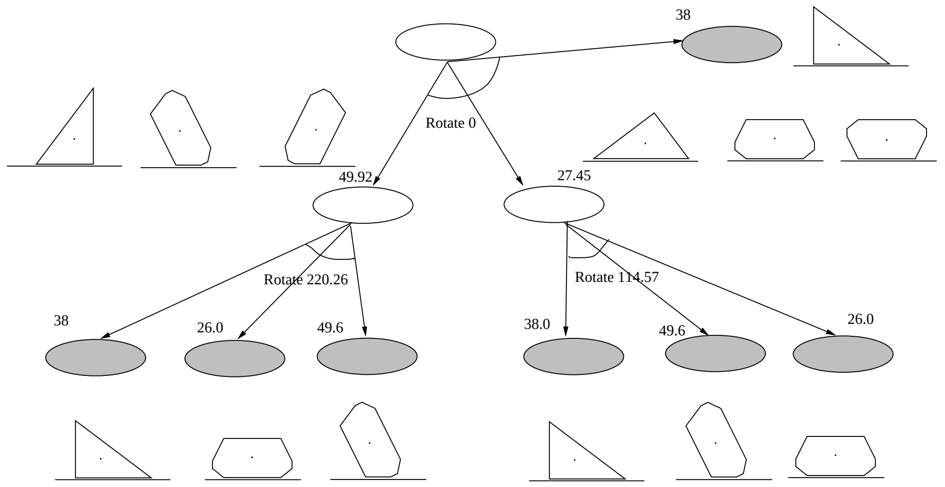


Figure 5.32: A sensor-based plan to orient two part shapes. It was generated by a modified version of the bottom-to-top planner.

same part and therefore have the same push function. More generally, when a set of parts all have the same push function (as in Figure 5.29), the same plan can orient all the parts. Note that we cannot recognize the parts however.

For parts with different push functions, a breadth-first search planner can be used to find a sensorless plan when one exists. An alternative approach is to generate individual plans for each part in the input set and to execute them sequentially. As long as all actions are chosen to be deterministic for all the parts, at the end of the complete sequence each part shape will be in a known orientation. However we cannot know which part we have oriented without subsequent sensing operations. Clearly such plans will be long.

5.7.5 Implementation

We have implemented planners in Common Lisp to generate orienting plans for convex polygonal parts using AND/OR search, the greedy algorithm, and the bottom-to-top algorithm. Given a part shape and sensor noise, the planners return a plan to orient the part uniquely when possible, and up to symmetry otherwise. These planners have been tested on several parts including those shown in Figure 5.33. Example plans generated by the AND/OR search planner, greedy planner, and bottom-to-top planner are shown in Figure 5.21, Figure 5.24, and Figure 5.25 respectively.

I timed the AND/OR search planner and bottom-to-top planner on a Sparc ELC. Traversing top to bottom and left to right in Figure 5.33, the AND/OR search planner took an average of 0.064 secs, 0.116 secs, 0.096 secs, 0.042 secs, 0.084 secs, and 0.142 secs respectively to generate a sensor-based plan. The bottom-to-top planner took an average of 0.060 secs, 0.112 secs, 0.078 secs, 0.048 secs, 0.096 secs, and 0.210 secs for the same set of parts.

We have implemented and demonstrated orienting of singulated parts using an Adept 550 robot and a conveyor belt. To pick up a part, the robot uses a suction cup (or an electromagnet for ferromagnetic parts). I implemented both sensor-based and sensorless orienting plans for four parts. The parts and the fence were made of delrin. I selected the two parts used in the sensorless 1JOC experiments (Figure 5.26) for further experimentation and ran 20 trials using a sensor-based

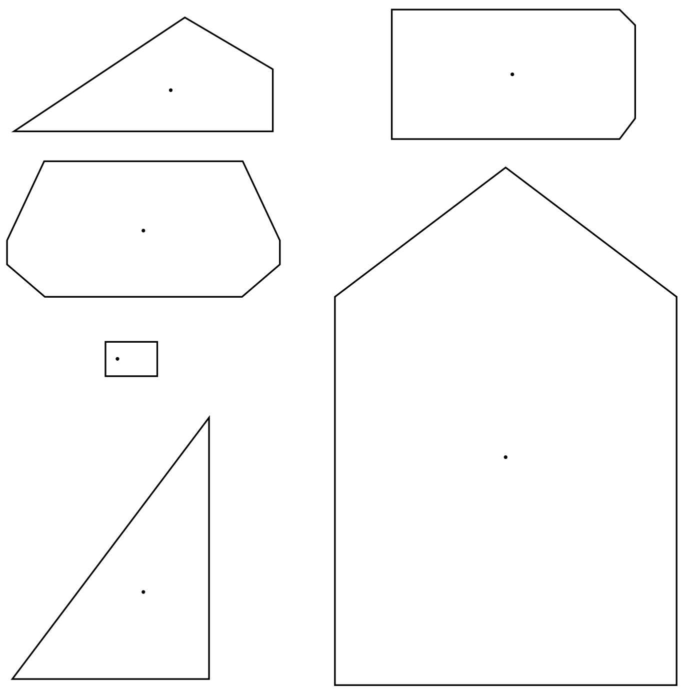


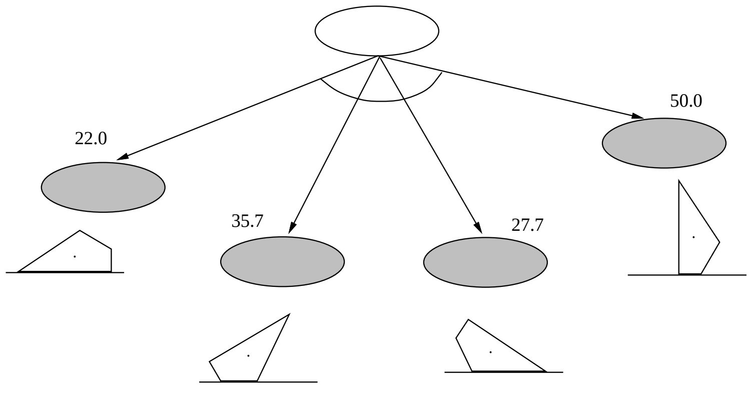
Figure 5.33: Sample part shapes that planners were tested on.

plan and 10 trials using a sensorless plan. For the right triangle, 17 of the 20 trials with the sensor-based plans and 8 of the 10 trials with the sensorless plan succeeded. For the 8-gon, 19 of the 20 trials with the sensor-based plan and all 10 trials of the sensorless plan succeeded. All observed failures occurred when the suction cup made insufficient surface contact with the part to pick it up. Since the robot picks up a part at an unknown point on the part, rotating the part causes bounded but unknown changes in the position of the part along the fence. To a lesser extent, the part also slides along the frictionless fence. Since the part position along the fence is not fully determined, this occasionally leads to the observed failures. The frequency with which such failures occur for a part depends on the sequence of rotations in the plan. An array of suction cups can be used to eliminate such failures.

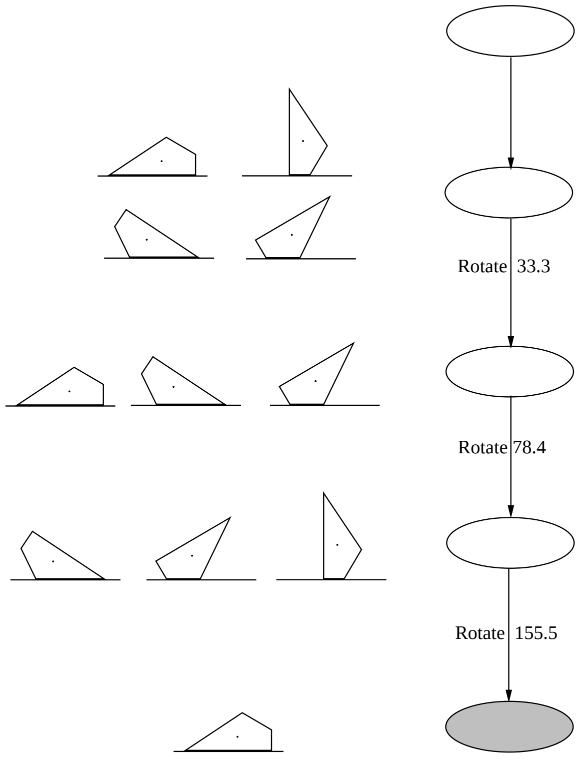
We also tested the example plan to orient and recognize the parts of Figure 5.26. We ran 20 trials, 10 with each part. The triangle was successfully oriented and recognized all 10 times and the 8-gon was successfully oriented and recognized 9 times. The single failure occurred due to a pickup failure with the suction cup.

5.8 Conclusion

In this chapter we described a parts orienting system that uses knowledge of the mechanics coupled with simple sensing operations to orient parts. The use of inexpensive and robust LED sensors with manipulation operations has several advantages. We showed that the sensors reduce the the number of orienting steps from $O(n)$ to m, the maximum number of states with the same diameter value (see Figure 5.34 for an example). This reduces execution time, and when the orienting process is



(a)



(b)

Figure 5.34: Using partial state information provided by sensors can significantly reduce plan length. (a) A one-step sensor-based plan. (b) A four-step sensorless plan.

pipelined, the number of pipelined stages. The sensors allow a feeder to orient multiple part shapes with a single plan. We identified conditions under which parts with different shapes can be oriented and recognized with a single plan, leading to greater flexibility in the parts orienting process. The sensors implicitly also increase robustness by verifying the results of operations.

The AND/OR search algorithm and the bottom-to-top algorithm both find the minimum length plan. They are both exponential-time planners. However for each new part, these algorithms have to run off-line only once. The greedy algorithm provides solutions in polynomial time, but the solutions may be longer than the minimum length plans.

Our work also has the potential to be applied to other tasks with similar mechanics, such as the sensorless 1JOC of Chapter 4. It would be straightforward to incorporate sensing of object diameter into the sensorless 1JOC system to substantially speed up the parts orienting process and to orient multiple parts.

Sensor-based plans can vary in execution time depending on the initial orientation of the part while sensorless plans always take the same amount of time to execute. While such variations in orienting time may appear to be a problem, note that the maximum execution time for sensor-based orienting is never greater than the execution time for sensorless orienting and is usually shorter. Therefore the use of sensors can significantly increase throughput.

In the next chapter we will extend this approach to perform parts orienting in the presence of shape uncertainty.

Chapter 6

Parts Orienting with Shape Uncertainty

Parts manufactured to tolerances have variations in shape. So far in this thesis we have assumed that the part shape is known exactly and does not vary. In fact, most work in robotic manipulation assumes parts have no shape variations. Devices such as bowl feeders frequently fail due to variations in part shape, particularly when multiple manufacturers supply the same part. Consequently it is important to devise systems that can orient parts in the presence of shape uncertainty. In this chapter, we study the effects of uncertainty in part shape on the orienting process. We identify conditions under which we can generate orienting plans for parts with shape uncertainty using the parts orienting system of the previous chapter. The variations in the shape of a part are characterized by the part's nominal shape and bounds on the shape uncertainty. The *variational class* (Requicha [152]) of parts for a given nominal part shape and a given model of shape uncertainty is the class of all part shapes that satisfy the shape uncertainty bounds. Given a nominal part shape and shape uncertainty bounds, we wish to generate orienting plans for the variational class of part shapes. As before, we define orienting a part to mean aligning a known edge of the part with the fence.

In the previous chapter, we saw the importance of part shape for the orientability of parts. There is an uncountably infinite set of part shapes that are valid instances of a given variational class. Planning in the presence of shape uncertainty is more difficult than planning for one shape or even a discrete set of shapes since we have to deal with this continuously varying class of shapes.

The main questions we tackle in this chapter are:

1. What are the effects of shape uncertainty on part orientability?

2. Can we orient parts with shape uncertainty, with and without sensors?

3. Can we develop a planner for parts orienting in the presence of shape uncertainty?

We first present our shape uncertainty model and discuss the nondeterminism in the parts orienting process that arises from shape uncertainty. We present algorithms to determine if each edge is stable, unstable, or possibly stable for the input shape uncertainty values. Focusing on the class of parts with a constant set of stable edges, we show that sensor-based and sensorless orienting plans can exist even with shape uncertainty. We present implemented planners that

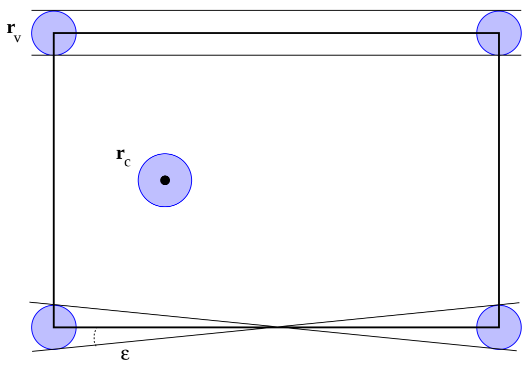


Figure 6.1: Shape uncertainty model indicating the uncertainty bounds on the locations of the center of mass and vertices. The nominal center of mass and polygon edges are drawn bold. The extremal positions and orientations of the longest edges are also indicated.

generate sensor-based and sensorless plans for this class of parts and describe properties of the planners. We outline a planner to handle parts whose set of stable edges may change due to shape uncertainty and conclude with a discussion of future work.

6.1 Shape Uncertainty Model

The need to manufacture parts specified by tolerances arose with the development of mass production methods that required parts that could work interchangeably (Voelcker [181]). Current industrial standards include the ANSI Y14.5M-1982 tolerancing and dimensioning standard [12]. The shape uncertainty model used in this chapter was earlier described in Akella and Mason [7]. It can be viewed as a parametric tolerance model which permits variations in both the orientations and positions of part edges.

We make the following assumptions in modeling the bounded uncertainty in the shape of a part (Figure 6.1):

- The nominal shape of the part is known. The part shape is defined by the positions of the vertices and center of mass.
- We consider only convex polygons that remain convex for all instantiations of the shape uncertainty values. That is, every member of the variational class is a convex polygon.
- Each vertex lies inside a circle of radius r_v centered at its nominal location. This radius is a specified input variable.
- The center of mass lies inside a circle of radius r_c centered at its nominal location. This radius can be computed as a function of r_v and the polygon shape, or can be specified as an independent input variable.
- The vertex uncertainty circles do not intersect each other and do not intersect the center of mass uncertainty circle. The uncertainty circle at each vertex intersects only the two edges incident at that vertex.
- The actual part edges are straight lines connecting the actual vertex positions.

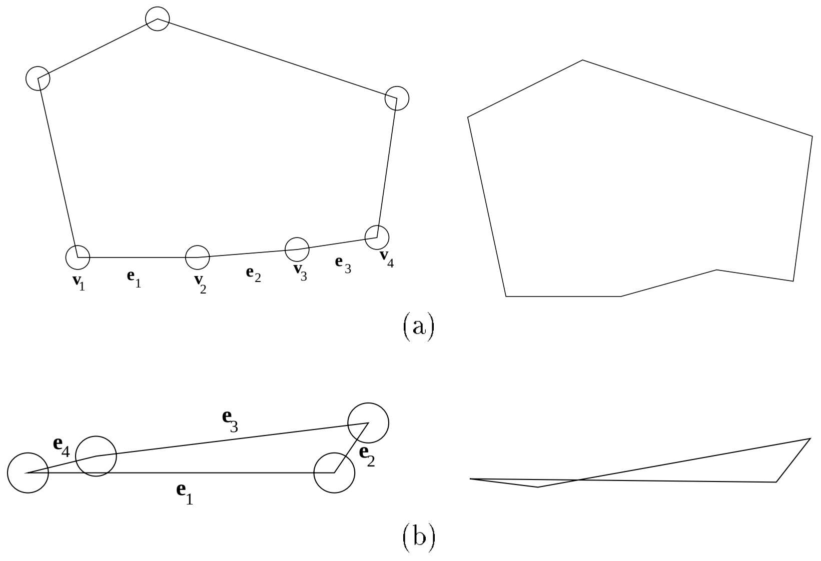


Figure 6.2: These parts do not satisfy the assumptions of our model. (a) The vertices v_2 and v_3 can become concave. This is shown for vertex v_3 at right. (b) Edges e_3 and e_4 of the quadrilateral can cross edge e_1, as shown at right.

Any part whose shape lies within the class of shapes defined by the nominal shape and the above shape uncertainty model belongs to the variational class of parts we wish to orient. Note that the center of mass location and its uncertainty circle radius can be specified as independent variables that are not functions of the vertex locations and uncertainty radii. We exclude from consideration polygonal parts which do not satisfy the above assumptions, such as those in Figure 6.2. The part in Figure 6.2 (a) has three adjacent edges e_1, e_2, and e_3, that are almost parallel. Vertices v_2 and v_3 can become concave vertices. The quadrilateral in Figure 6.2 (b) can have some of its edges cross each other.

6.1.1 Selecting a Shape Uncertainty Model

There are a couple of aspects to choosing a model of shape uncertainty. One is the ease of analysis that the model permits. The other is how closely it captures the actual variations in the shapes of the toleranced parts, which depend on the manufacturing process. We use our shape uncertainty model to characterize the effects of shape variations on part orienting in a manner independent of the manufacturing process. Our model enables us to quantify variations in action ranges and edge orientations in terms of the shape uncertainty parameters. Our ultimate long-term goal is to use shape uncertainty models which lend themselves to such analysis while incorporating true shape variations arising from manufacturing processes. This work represents a first step in that direction.

Our model is motivated by the need to bound the range of orientations for which a part can rotate to either of two adjacent stable edges. Since the vertices and center of mass uncertainty regions are explicitly bounded, we can compute these orientation ranges. Alternative models of shape uncertainty have been developed for other problems, including those of Requicha [152], Brost [32], and Latombe and Wilson [106]. Requicha's model specifies tolerance zones by sweeping out circles of specified radii along the polygon edges. Brost's model specifies tolerance zones by

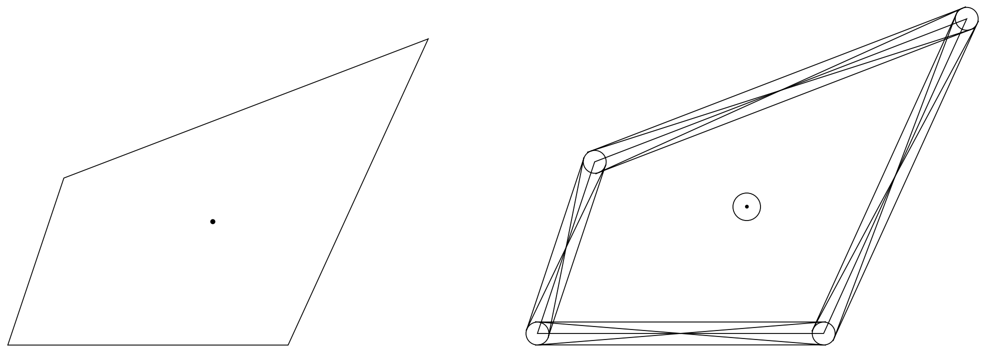


Figure 6.3: The nominal part shape and the range of permissible part shapes arising from shape uncertainty.

choosing an uncertainty polygon for each vertex and sweeping it along the two polygon edges that meet at the vertex. The models of Requicha and Brost place no restrictions on the shape of the edges and permit the vertices to lie outside their uncertainty regions provided the edges lie inside the tolerance zone. Latombe and Wilson's model specifies tolerance zones by sweeping out each edge of the polygon through a specified positional range. This therefore bounds the region of vertex locations. Note that they make the stronger assumption of the edges having perfect orientation. Any shape uncertainty model for which the vertices and center of mass uncertainty regions are explicitly bounded can be analyzed and treated by our method.

6.2 Effects of Shape Uncertainty

Shape uncertainty, under our model, causes the following effects (Figure 6.3):

1. Variations in orientations of part edges.
2. Variations in the diameter values associated with the stable orientations.
3. Variations in the action ranges of the parts, the range of angles over which a part transitions from one edge to another.
4. Variations in the center of mass location.
5. Variations in the set of stable edges.

Variations in edge orientations and diameter values can be computed directly from the shape uncertainty values. Consider an edge whose nominal endpoints, the vertices, are known. For a given value of the vertex uncertainty circle radius r_v the maximum and minimum orientations of the edge are determined from the tangents to the vertex uncertainty circles at the edge endpoints. Thus the edge orientation varies between $\psi - \epsilon$ and $\psi + \epsilon$ where ψ is the edge orientation, $\epsilon = \sin^{-1}(2r_v/l)$ and l is the nominal length of the edge. The diameter of the part normal to the fence when aligned against it can vary from the nominal size by $2r_v$. That is, for a nominal diameter value of d, the actual diameter lies in the range $[d - 2r_v, d + 2r_v]$. See Figure 6.1.

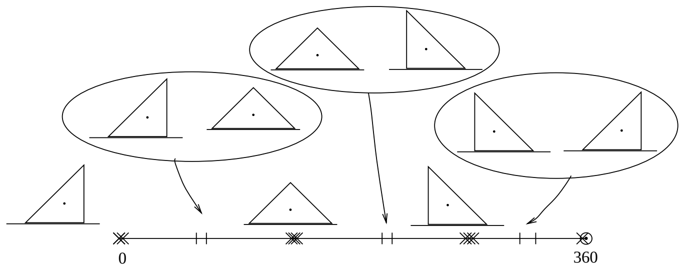


Figure 6.4: Resting ranges for the isosceles right triangle with shape uncertainty. Variations in the resting ranges and stable orientations are due to shape uncertainty. The uncertainty in the stable orientations and the resting ranges is indicated by the x's and vertical bars respectively drawn at their extremal orientations. r_v and r_c are 0.4 mm and the hypotenuse of the triangle has a length of 22.63 mm.

Finding the variations in the action ranges requires a more careful analysis. The action ranges can be obtained from the resting ranges as described in Section 5.2.5. Recall that the resting range of a stable edge is the range of initial part orientations for which the part comes to rest on the edge. Uncertainty in the positions of the vertices of the edge and the center of mass causes variations in the resting ranges (Figure 6.4). The uncertainty in the endpoints of the resting ranges can be determined from the angles defined by the common tangents to the center of mass uncertainty circle and the vertex uncertainty circles for the CW and CCW vertices of the edge. See Figure 6.1 for the relevant variables.

Given the nominal shape of the part, we can compute its nominal center of mass (COM) from its geometry. For a given shape uncertainty bound, computing the exact COM uncertainty locus is an open problem. We model the variation in the center of mass with a circle of radius r_c that bounds the locus of possible center mass locations. For a more detailed discussion of the estimation of the center of mass locus, see Appendix C.

The set of stable edges for a given instantiation of the part depends on the center of mass location and the edge positions and orientations. Therefore identifying all possible sets of stable edges that can occur requires a careful analysis of all combinations of stable edges that are consistent. We defer a more detailed discussion to Section 6.3.

6.2.1 Nondeterminism

The main operational effect of shape uncertainty is to introduce nondeterminism into the parts orienting process. The nondeterminism comes from variations in the outcomes of actions and variations in the set of stable edges.

The actions for a particular instantiation of a part shape are deterministic. Over the variational class of parts however, some of the actions are effectively nondeterministic actions. To see this, consider the two instantiations of the part shown in Figure 6.5 (b) and (c). A push-align action which brings the two instantiations to the same initial orientation results in a CCW rotation for the first instantiation and a CW rotation for the second instantiation. The action is deterministic for each instantiation, but since it has different outcomes for the two, we have to treat it as a nondeterministic action. More generally, actions which lead to different outcomes for different part instantiations have to be treated as nondeterministic actions over the variational class of part

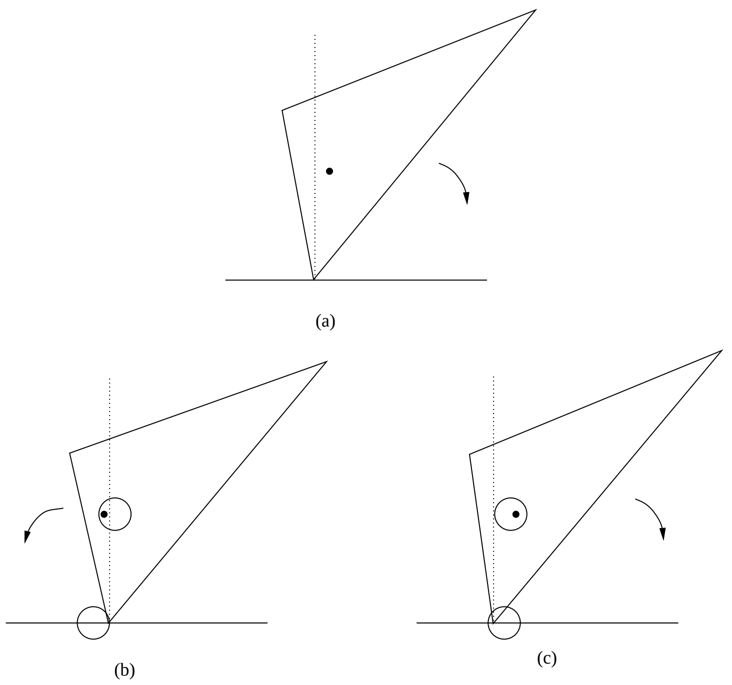


Figure 6.5: Shape uncertainty results in nondeterministic actions. The result of the push depends on the relative location of the center of mass and contact vertex. The contact normal at the vertex is indicated by the dotted line. (a) For the nominal part shape, the line from the center of mass to the vertex is CW to the normal, and the part rotates CW. (b) The part rotates CCW for this instantiation of the part shape. (c) The part rotates CW for this instantiation of the part shape.

shapes.

The set of stable edges for a particular instantiation of a part shape is constant. An edge is stable if the perpendicular projection of the center of mass onto the edge lies in the interior of the edge. Over the variational class of parts, the position of the center of mass relative to each edge varies, and the set of stable edges may therefore change. Edges of a part that are stable in its nominal shape may become unstable for some instantiations of the part, or unstable edges of the part may become stable (see Figure 6.6). These variations in the set of stable edges manifest themselves as nondeterministic behaviors since they cause variations in the action ranges as well.

These nondeterministic effects complicate the orienting problem and make planning harder. As a first step, we tackle the problem of planning with shape uncertainty when the set of stable edges does not change. While the size, position, and orientation of each stable edge can vary, these edges are always stable for all instances of the variational class, and no other edges become stable. In this chapter we present an implemented planner for polygons whose set of stable edges is constant and outline a planner for polygons whose set of stable edges may change.

6.3 Stability Regions

The set of stable edges depends on the location of the center of mass relative to the edges. To satisfy our assumption that the set of stable edges of the polygon does not change, we have to

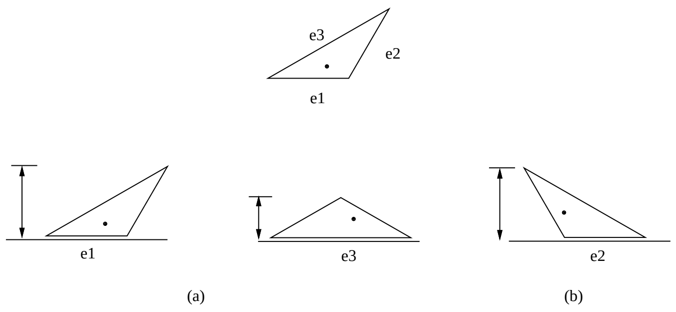


Figure 6.6: Edges of a part can become stable or unstable depending on the particular instantiation of the shape uncertainty values. (a) The isosceles triangle has stable edges e_1 and e_3 for its nominal shape. (b) With shape uncertainty, edge e_2 is stable for some instantiations of the shape.

ensure that the center of mass always lies in the appropriate region of the polygon. The *stability region* of a part edge is the region of locations of the COM for which the edge is stable. It is the region in the interior of the polygon for which the center of mass projects onto the interior of the edge segment. Uncertainty in the length, position, and orientation of an edge and in the position of the center of mass leads us to replace the stability region with the *shrunken stability region* and the *expanded stability region*. The shrunken stability region of an edge is the region of center of mass locations in the polygon interior for which the edge is always stable. The expanded stability region of an edge is the region of all center of mass locations in the polygon interior for which the edge may be stable. If the center of mass is inside the shrunken stability region of an edge, the edge is always stable. If the center of mass is outside the expanded stability region of an edge, the edge is always unstable. When the COM is outside the shrunken stability region and inside the expanded stability region, the edge can be stable or unstable depending on the instantiation of the shape uncertainty parameters. The shrunken stability region is in the interior of the stability region which is in turn in the interior of the expanded stability region. We compute these regions by performing region intersection and union operations. (See Nievergelt and Preparata [135] for a description of region intersection and union algorithms.)

6.3.1 Stability Region Computation

Stability Region of an Edge. The stability region of a part edge is the region of locations of the center of mass for which the edge is stable. For a part with known shape, it is the region in the interior of the polygon for which the perpendicular projection of the center of mass onto the edge is in the interior of the edge segment. The region is found using the inward-pointing normals to the edge at its vertices, n_{cw} and n_{ccw}. All points between these normals in the polygon interior constitute the stability region of the edge (Figure 6.7 (a)). We intersect the polygon with the region contained by the normals and the edge to obtain the stability region. Any center of mass location in the stability region not including points on its boundary guarantees stability of the edge.

Shrunken Stability Region. The shrunken stability region of an edge is the region of center of mass locations in the polygon interior for which the edge is always stable. The shrunken stability region differs from the stability region for three reasons. First, there is uncertainty in the length and location of the edge. Second, there is uncertainty in the orientation of the edge. Third, there is uncertainty in the location of the center of mass. The shrunken stability region is the intersection of the stability regions generated for every instance of the center of mass location and edge length, position, and orientation. The uncertainties therefore shrink the region of center of mass locations for which the edge is always stable.

First consider the case with no uncertainty in the orientation of the stable edge. Since we want the smallest region for which the COM projection lies in the interior of the edge segment, we first shrink the edge segment at each end by the vertex uncertainty radius r_v. To compensate for the uncertainty in the center of mass location, we further shrink this edge segment, the projection target, by the COM uncertainty radius r_c at each end. We also have to translate this shrunken edge segment into the polygon interior by the distance $r_v + r_c$ to ensure that any valid center of mass location is in the interior of the polygon and lies above the edge. The region obtained by erecting the inward-pointing normals at each end of the resulting edge segment indicates center of mass locations for which the edge is always stable. Since the center of mass must always lie in the interior of the polygon, we intersect this region with the polygon shrunk by r_v. The resulting region is the shrunken stability region for the edge with no uncertainty in its orientation. This region can be found more directly by first finding the three lines formed by moving the edge line and the two normal lines "inward" by the amount $r_v + r_c$. The intersection of the polygon shrunk by r_v with the "inward" halfspaces defined by these three lines gives the stability region of the edge with no orientation uncertainty.

Since there is uncertainty in the orientation of the stable edges, we have to find the stability region for each valid orientation of the edge. For a selected edge orientation, we sweep the edge line over the vertex uncertainty circles to obtain the "highest", "lowest", and "innermost" CW and CCW vertex positions for that orientation. We find the stability region for each of the corresponding edge instantiations and shrink it by the center of mass uncertainty radius r_c. The stability region for the selected edge orientation is the intersection of the four regions obtained in this manner. Once we have obtained this stability region for every possible edge orientation, we find the intersection of all these regions to generate the shrunken stability region.

Since the above procedure to compute the shrunken stability region involves region intersection operations for every valid edge orientation, we use a simpler procedure which provides us with a conservative approximation to the shrunken stability region (Figure 6.7 (b)). We first draw the two extremal common tangents to the vertex uncertainty circles of the edge e, t^{min} and t^{max}, which correspond to the minimum and maximum edge orientations respectively. Consider the tangent t^{min}. Erect the inward pointing normals n_{cw}^{min} and n_{ccw}^{min} to the tangent that are also tangent to the vertex uncertainty circles of the edge and closest to each other. Translate the nominal edge segment inwards by r_v and call it e'. Find the region R^{min} formed by intersecting the appropriate halfspaces defined by the lines n_{cw}^{min}, n_{ccw}^{min} and e'. Repeat this operation for the other tangent t^{max} to obtain the region R^{max}. Shrink the polygon by r_v and intersect it with the intersection of R^{min} and R^{max}. We shrink this resultant region along the lines e', n_{cw}^{max}, and n_{ccw}^{min} by the uncertainty in the center of mass position, r_c, to get the (approximate) shrunken stability region of the edge.

Closer inspection reveals that the CW boundary of the shrunken stability region can be obtained from n_{cw}^{max} and the CCW boundary of the shrunken stability region can be obtained from n_{ccw}^{min}. If we generate the lines e', n_{cw}^{max}, and n_{ccw}^{min} and shrink them "inwards" by the amount r_c, and intersect the polygon shrunk by r_v with the appropriate halfspaces defined by these lines, we obtain the (approximate) shrunken stability region.

This procedure also demonstrates that it generates a conservative approximation of the true shrunken stability region. There is no edge instantiation with normals that are clockwise to n_{cw}^{max}, and there is no edge instantiation with normals that are counterclockwise to n_{ccw}^{min}. Further, there is no edge instantiation which is above e' or with a smaller length than the segment formed by intersecting e' with n_{ccw}^{min} and n_{cw}^{max}.

Expanded Stability Region. The expanded stability region of an edge is the region of all center of mass locations in the polygon interior for which the edge may be stable. The expanded stability region also differs from the stability region because of uncertainty in the length and position of the edge, uncertainty in the orientation of the edge, and uncertainty in the location of the center of mass. In contrast to the shrunken stability region, it is the union of the stability regions generated for every instance of the COM location and edge length, position, and orientation. The uncertainties therefore serve to enlarge the region of center of mass locations for which the edge may be stable.

We compute a conservative approximation of the expanded stability region (Figure 6.7 (c)). We draw the two extremal common tangents to the vertex uncertainty circles of the edge e, t^{min} and t^{max}, which correspond to the extremal edge orientations. Consider the tangent t^{min}. Erect the inward pointing normals n_{cw}^{min} and n_{ccw}^{min} to the tangent that are also tangent to the vertex uncertainty circles of the edge and farthest from each other. (Note that these normals differ from those used in computing the shrunken stability region.) Move the edge line outwards by r_v and move the normals outwards by the amount r_c. Find the region R^{min} formed by the intersection of the appropriate halfspaces defined by these lines. Repeat this operation for the other tangent t^{max} to obtain the region R^{max}. Grow the polygon by r_v and intersect it with the union of R^{min} and R^{max}. The resulting region is the (approximate) expanded stability region of the edge.

Closer inspection reveals that the CW boundary of the expanded stability region can be obtained from n_{cw}^{min} and its CCW boundary can be obtained from n_{ccw}^{max}. If we move the edge line "outwards" by r_v, and move the normals n_{cw}^{min} and n_{ccw}^{max} "outwards" by the amount r_c, and intersect the appropriate halfspaces defined by these lines with the polygon expanded by r_v, we obtain the (approximate) expanded stability region.

Given a nominal center of mass location, we can perform point-inclusion tests to determine if the center of mass is in the interior of the shrunken stability and expanded stability regions. (See Preparata and Shamos' book [145] for a description of point-inclusion algorithms.) These tests identify an edge as always stable, always unstable, or sometimes stable.

We will first concentrate on parts whose center of mass is inside the shrunken stability region of a subset of edges and outside the expanded stability region of the remaining edges for the specified shape uncertainty values. So the set of stable edges is constant. We refer to such a part as a part with a *single qualitative shape*. Given a part with a single qualitative shape, we can identify representative actions that cover the space of actions and generate plans using breadth-first search. These actions may be deterministic or nondeterministic, and we next describe how to identify them.

6.4 Generating Actions

Shape uncertainty introduces nondeterminism in the actions. We therefore have to identify the deterministic and nondeterministic action ranges and select representative actions accordingly. To compute the deterministic and nondeterministic action ranges, for each start edge we have to find the smallest and largest rotations that are guaranteed to achieve the transition to each destination edge, and the smallest and largest rotations that can potentially achieve the transition

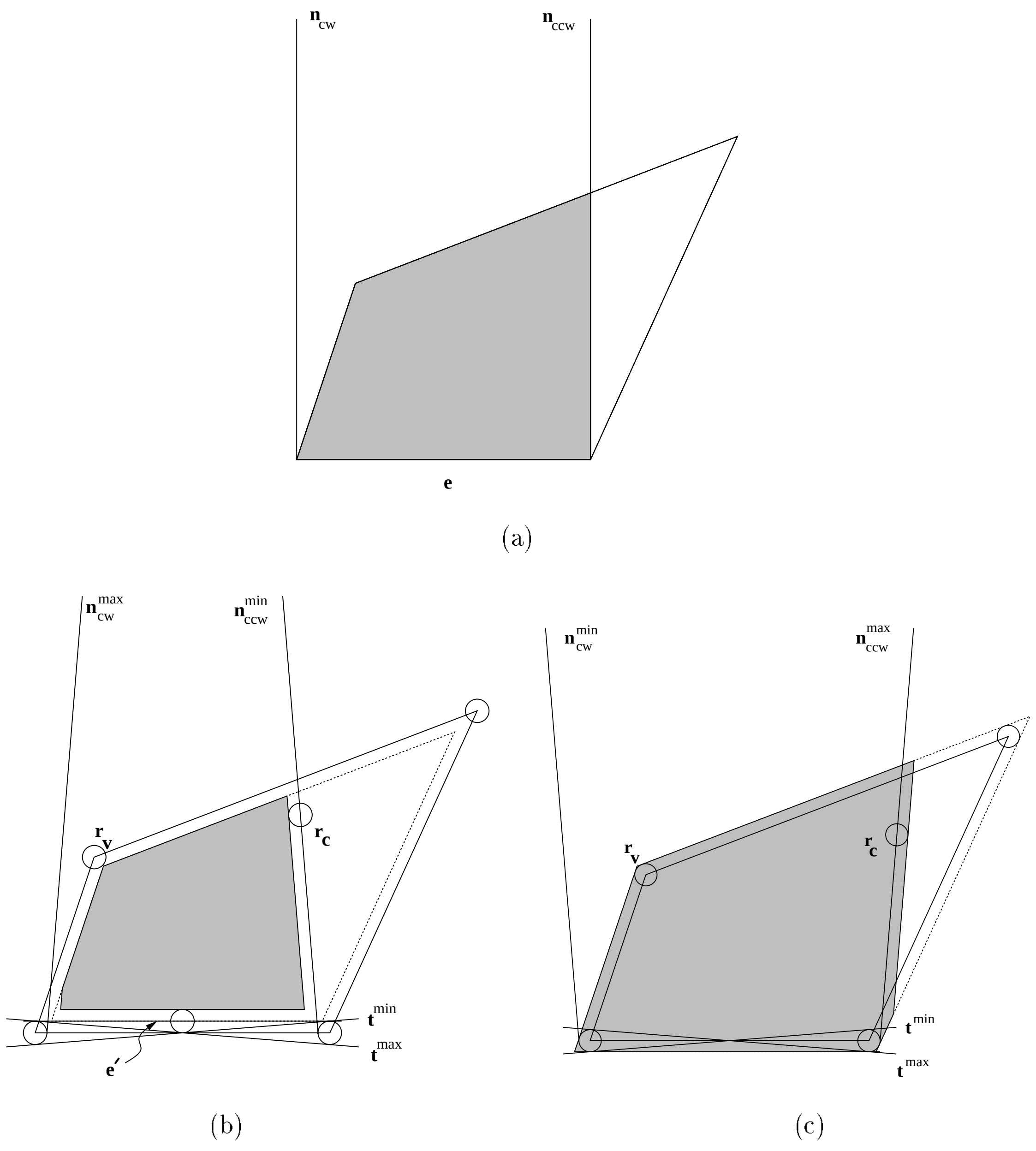


Figure 6.7: Stability regions of the bottom-most edge e of the polygon. (a) Stability region with no shape uncertainty. (b) Shrunken stability region. (c) Expanded stability region. The shrunken and expanded stability regions shown here are conservative approximations of the true shrunken and expanded stability regions.

to each destination edge. To compute these extremal actions for each edge, we will assume that the following can occur simultaneously: the start edge can be in any orientation in its allowed range, each vertex can be at any position in its uncertainty circle, and the center of mass can be at any position in its uncertainty circle. We will refer to this as the *independence assumption*. From the extremal action ranges we can find the representative actions for a given set of edges and search for an orienting plan in this space of actions. If a plan exists, it is guaranteed to work for any instance of the part since it works under the most extreme combination of COM, vertex, and edge placements.

6.4.1 Deterministic Action Ranges

Recall the case when there is no shape uncertainty and we wish to determine the range of actions to transition from start state s_i to state s_j . Let the stable orientation of state s_i be ψ_i. Let the right and left limits of the resting range of state s_j be ρ_j and λ_j. Then any action in the range $(\lambda_j - \psi_i, \rho_j - \psi_i)$ will cause a deterministic transition from state s_i to state s_j. We perform this computation for the transition from s_i to every other state, taking care to handle angle wraparound at 360 degrees. We can thus determine the deterministic action ranges for the transitions between every pair of states.

Now consider transitioning from state s_i to state s_j with the added element of shape uncertainty. There are variations in the stable orientation of each edge, and in the resting range of each edge, the range of initial orientations for which the part comes to rest on the edge (Figures 6.8 and 6.9). We can determine these variations from the positional uncertainty circles of the center of mass and the transition vertices. Recall that a transition vertex is a polygon vertex that corresponds to a local maximum in the radius function and thus specifies the limit of a resting range. The stable orientation of state s_i, ψ_i, varies between ψ_i^{min} and ψ_i^{max} where $\psi_i^{min} = \psi_i - \epsilon_i$, $\psi_i^{max} = \psi_i + \epsilon_i$, $\epsilon_i = sin^{-1}(2r_v/l_i)$, and l_i is the nominal length of edge e_i. The right limit of the resting range of state s_j varies between ρ_j^{min} and ρ_j^{max} and this variation is computed from r_v, r_c, and the distance d_j^{cw} from the center of mass to the CW transition vertex for the edge. The minimum limit ρ_j^{min} is given by $\rho_j - sin^{-1}((r_v + r_c)/d_j^{cw})$ and the maximum limit ρ_j^{max} is given by $\rho_j + sin^{-1}((r_v + r_c)/d_j^{cw})$. Similarly the variation in the left limit of the resting range of state s_j is computed from r_v, r_c, and the distance d_j^{ccw} from the center of mass to the CCW transition vertex for the edge. The minimum limit λ_j^{min} is given by $\lambda_j - sin^{-1}((r_v + r_c)/d_j^{ccw})$ and the maximum limit λ_j^{max} is given by $\lambda_j + sin^{-1}((r_v + r_c)/d_j^{ccw})$. The action range specified by the open interval $(\lambda_j^{max} - \psi_i^{min}, \rho_j^{min} - \psi_i^{max})$ guarantees a deterministic transition from s_i to s_j. A deterministic action range is an equivalence class of actions.

So uncertainty in part shape causes the deterministic action range to shrink from $(\lambda_j - \psi_i, \rho_j - \psi_i)$ to $(\lambda_j^{max} - \psi_i^{min}, \rho_j^{min} - \psi_i^{max})$. (See Figure 6.10.) In fact, deterministic action ranges can vanish for large enough values of uncertainty. This corresponds to the condition $(\lambda_j^{max} - \psi_i^{min}) \geq (\rho_j^{min} - \psi_i^{max})$. Intuitively, this occurs when the uncertainty in the stable orientation of the start edge is as large or larger than the nominal resting range of the edge it is transitioning to. When some of the action ranges vanish, the deterministic action ranges are no longer guaranteed to cover the action space of $[0, 360)$ and we may not have deterministic actions to get from every state to every other state. In such cases, a plan to orient the part using only deterministic actions may not exist.

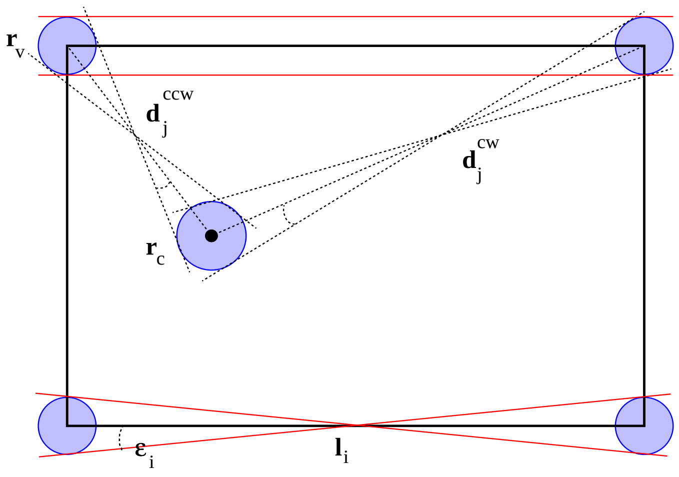


Figure 6.8: Uncertainty in stable orientations and resting ranges of edges can be computed using the indicated variables.

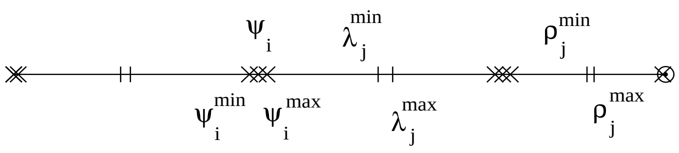
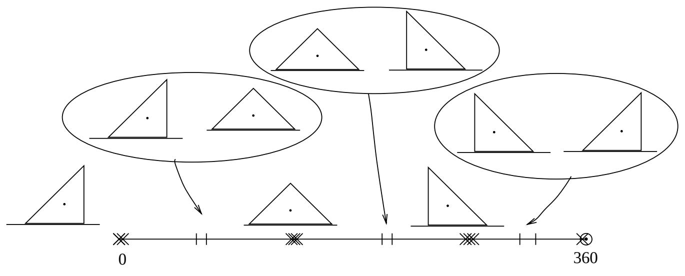
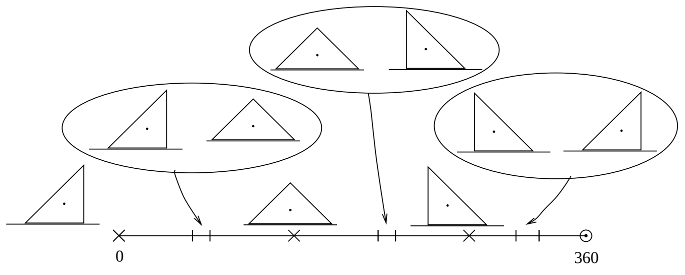


Figure 6.9: Resting range diagram indicating the uncertainty in stable orientations and resting ranges of edges due to shape uncertainty. The labeled elements are used to compute the action ranges for the transition from state s_i to state s_j.



(a)



(b)

Figure 6.10: (a) Resting ranges with shape uncertainty. (b) Corresponding action ranges with shape uncertainty for state s_1. Each range of actions that contains an x and is bracketed by a pair of successive vertical bars is a deterministic action range, and each range of actions that does not contain an x and is bracketed by a pair of successive vertical bars is a nondeterministic action range. Only the nominal stable orientations are shown for clarity.

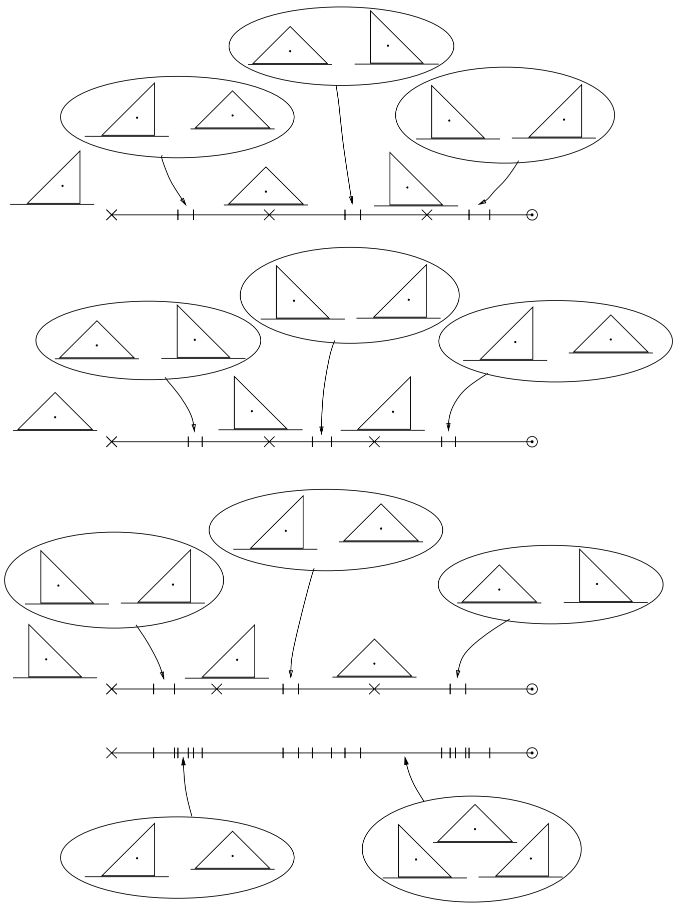


Figure 6.11: Deterministic and nondeterministic action ranges for the isosceles right triangle. An action from a deterministic action range causes a transition to a single outcome while while an action from a nondeterministic range can cause a transition to either of two states. The overlap ranges for the triangle are also indicated.

6.4.2 Nondeterministic Action Ranges

Nondeterministic action ranges are action ranges with multiple outcomes due to shape uncertainty. A nondeterministic action causes a transition from a start edge to a destination edge or the stable edges that are neighbors of the destination edge. We compute the nondeterministic action ranges from state s_i that can lead to state s_j or its CW or CCW stable neighbors, s_j^{cw} and s_j^{ccw} respectively. The action range specified by the closed interval $[\lambda_j^{min} - \psi_i^{max}, \lambda_j^{max} - \psi_i^{min}]$ is the range of actions for which the part transitions from state s_i to either state s_j or state s_j^{ccw} and $[\rho_j^{min} - \psi_i^{max}, \rho_j^{max} - \psi_i^{min}]$ is the range of actions for which the part transitions from state s_i to either state s_j or state s_j^{cw}. See Figure 6.11. A nondeterministic action range is not guaranteed to be an equivalence class of actions.

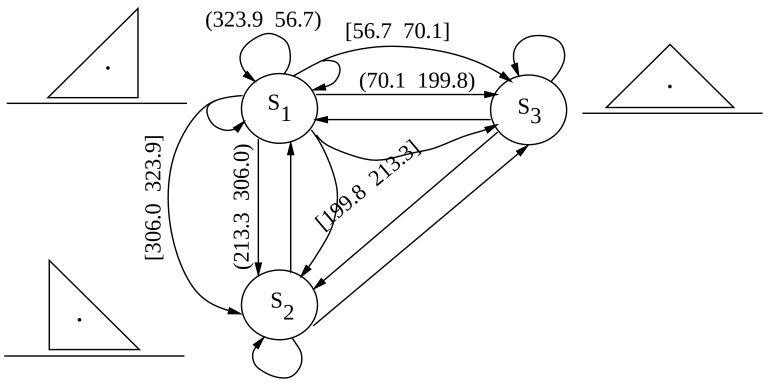


Figure 6.12: Transition graph of the isosceles right triangle with nondeterministic and deterministic actions to transform the orientation. For clarity, nondeterministic arcs and action ranges are shown only for transitions from state s_1. r_c and r_v are 0.4 mm.

6.4.3 Action Ranges for a Set of States

The deterministic and nondeterministic action ranges for transitions from a state to the other states cover the entire action space of $[0, 360)$, as illustrated in Figure 6.12. To find the deterministic and nondeterministic action ranges of a set of states, we first compute the deterministic and nondeterministic action ranges of each individual state that belongs to the set. As with the case with no shape uncertainty, we overlap the set of action range intervals from the individual states to obtain another set of intervals, the overlap ranges. Overlap ranges that are formed by the overlap of only deterministic action ranges are deterministic overlap ranges and overlap ranges that are formed by the overlap of at least one nondeterministic action range are nondeterministic overlap ranges. Representative actions are selected to be in the middle of each overlap range. A representative action selected from a deterministic action range is a deterministic action and a representative action selected from a nondeterministic action range is a nondeterministic action.

Consider a set of k indistinguishable states of a part with n stable edges. Each of the k states has n deterministic action ranges and n nondeterministic action ranges. So there are $2kn$ action ranges, which lead to $2kn$ overlap ranges and hence $2kn$ representative actions.

6.5 Sensor-based Orienting with Shape Uncertainty

Planning in the presence of shape uncertainty is a difficult problem since we have to find plans that work for a continuous range of part shapes that are all valid instances of the variational class. As in the case with no shape uncertainty, we can continue to use only deterministic actions when generating plans (see example plan in Figure 6.13). However the deterministic actions do not cover the space of actions, and searching solely in the space of deterministic actions may not yield a plan even when one exists (Figure 6.14). The use of deterministic and nondeterministic actions when generating plans guarantees that the action space is covered. The deterministic and nondeterministic action ranges and representative actions are determined for any set of states using the method described in the previous section. Knowing the representative actions, we can generate plans by breadth-first search of the AND/OR tree. See Figure 6.15 for another plan that uses

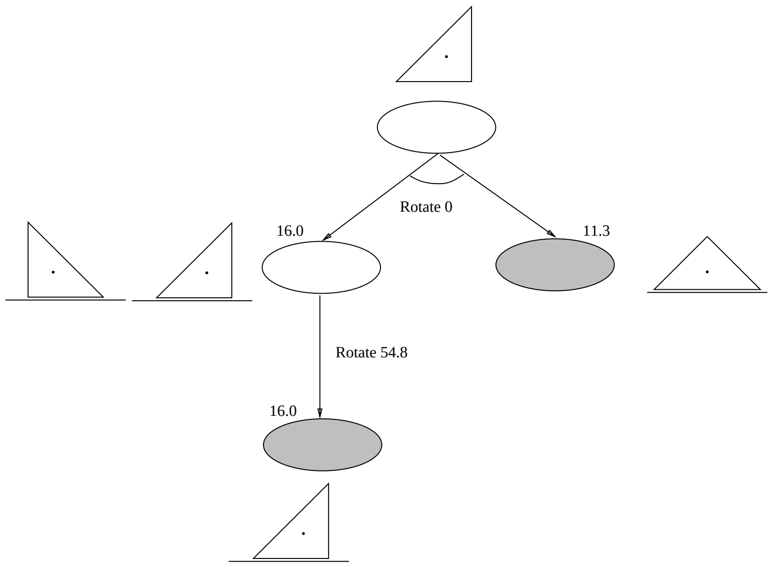


Figure 6.13: Plan to orient the isosceles right triangle in the presence of shape uncertainty. The actions are all deterministic. r_v and r_c are 0.1 mm.

nondeterministic actions.

Finding the result of an action with shape uncertainty requires a little preprocessing since this information cannot always be obtained from the radius function. For each state, we first store the resulting set of states for each of its deterministic and nondeterministic action ranges. To find the resulting set of states when a specified action is applied to a given set of initial states, we find the action range that the action belongs to for each of the initial states and retrieve the corresponding stored set of resulting states. The union of the resulting states over all the initial states is the resulting set of states for the specified action and set of initial states.

To deal with variations in the stable diameter values due to shape uncertainty, we extend the approach of the previous chapter in dealing with sensor noise and finite sensor resolution. That is, for each state we determine the range of diameter values it can assume and then determine the range of sensor values this range of diameter values can generate. Two states are considered distinguishable if their ranges of sensor values do not overlap, and two states with overlapping ranges of sensor values are considered indistinguishable.

6.6 Is the Planner Complete?

We first ask if there always exists a guaranteed plan to orient any part which satisfies our shape uncertainty model. An orienting plan is guaranteed to exist if the part has at least one stable edge with a unique value. To see this sufficient condition for plan existence, note that such a part can be repeatedly rotated until it comes to rest on the stable edge with the unique value, at which point its orientation is known. However there are parts that satisfy our shape uncertainty model but are not guaranteed to be orientable for some value of the shape uncertainty parameters. For example, consider a rectangle with its center of mass at a distance d from its geometric center and whose nominal breadth b, nominal height h, vertex uncertainty circle radius r_v, and center of mass

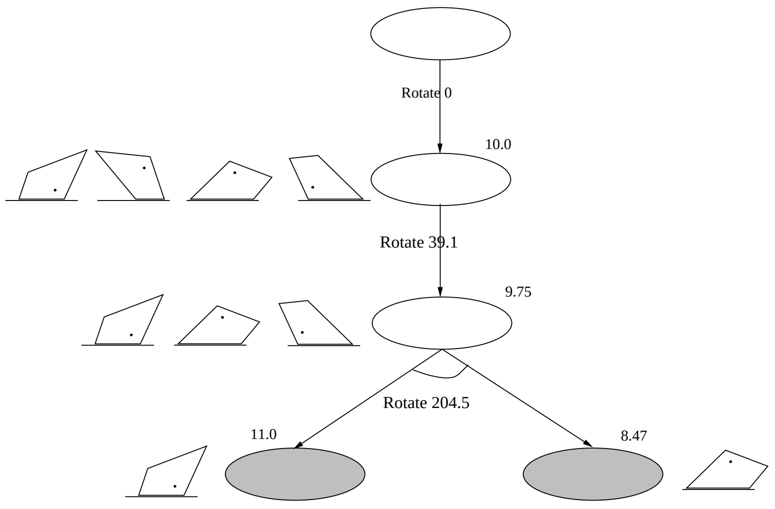


Figure 6.14: A plan using nondeterministic and deterministic actions exists for the quadrilateral when no plan using only deterministic actions exists. r_v and r_c are 0.5 mm and the longest polygon edge has length 13.93 mm. Note that for clarity, the sensor value we show at each node is the average of the diameter values for the corresponding indistinguishable set.

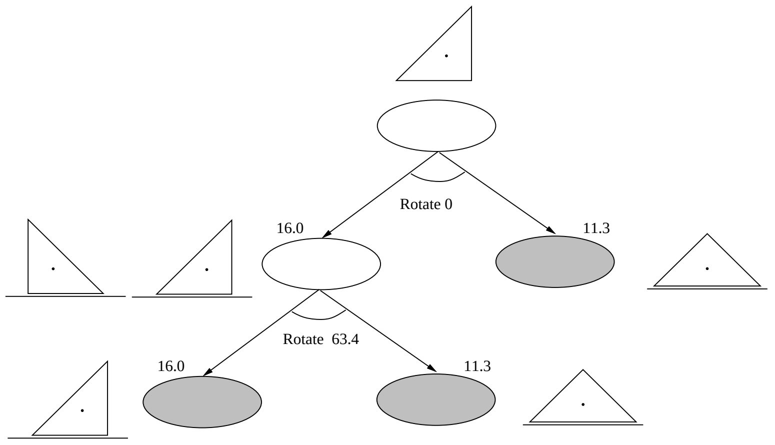


Figure 6.15: Plan to orient the isosceles right triangle when r_v and r_c are 0.4 mm. This plan uses a nondeterministic action.

uncertainty circle radius r_c are chosen such that $b > h$, $b - 2r_v \leq h + 2r_v$, and $d < r_c$. The nominal rectangle can be brought to a distinct orientation. A valid instantiation of this rectangle is a square with its center of mass at its geometric center, and clearly such a square can be oriented only up to symmetry.

Having demonstrated that the problem of parts orienting with shape uncertainty is not solution-complete, we address the question of planner completeness here: Given a nominal part shape with bounded shape uncertainty that satisfies our assumptions, does our planner always return an orienting plan when one exists and indicate failure when no such plan exists?

Using our shape uncertainty model, the planner can compute the deterministic and nondeterministic action ranges, which cover the range $[0, 360)$. The planner chooses representative actions from the overlap ranges obtained from these action ranges and generates plans by performing breadth-first search. The planner is complete if the computed action ranges are accurate. If the action ranges are overly conservative, the planner may not find a solution even when one exists.

The deterministic and nondeterministic action ranges for a stable edge depend on its orientation range and the allowed positions of the COM and transition vertices. When generating these action ranges, we have assumed that the orientation of each stable edge can assume any value in its allowed orientation range and that the center of mass and the transition vertices can be located anywhere in their respective uncertainty circles. We call this the independence assumption since we assume independence of the edge orientations and vertex positions. When a plan exists, it is guaranteed to work for any instance of the part since it works for the worst-case combination of conditions.

Each instantiation of the shape within the uncertainty bounds is a shape with constraints on the orientations and positions of the part edges. For example, the sum of the interior angles of an n-gon must sum to $(2n - 4)90$ degrees. Also, multiple edges cannot all be at their extremal orientations. Do these constraints mean the action ranges generated using the independence assumption are overly conservative?

Consider a stable edge, the start edge, for which we wish to find the action ranges. For a given orientation of the start edge, only the positions of its vertices are constrained. Any other vertex including the transition vertices can assume any position in its uncertainty circle for any given orientation of the start stable edge. So the action ranges from a stable edge to all stable edges other than its adjacent edges are accurately determined. For transitions from the stable edge to its adjacent edges, we can find the extremal position ranges of the vertices for each orientation of the edge and compute the corresponding action range. For example, when the edge is in an extremal orientation, there is exactly one set of positions its vertices can assume. If we compute the union of the action ranges over all orientations of the edge, we obtain the maximal action range for the transition to an adjacent edge. The action ranges to adjacent edges computed under our independence assumption are more conservative than those computed by this procedure.

The accuracy with which we determine the locus of the center of mass also determines the accuracy of the action ranges. We have assumed that the center of mass uncertainty locus is described by a circle. In fact this locus is not constrained to be a circle (see Appendix C). The circumscribing circle of the true locus we use may be an overly conservative approximation of the true locus.

These two sources of inaccuracy in the action ranges mean that our planner may not be complete. When the planner generates a plan, all instances of the part are guaranteed to be orientable. When the planner cannot find a plan however, it is unclear if there are some part instantiations which cannot be oriented.

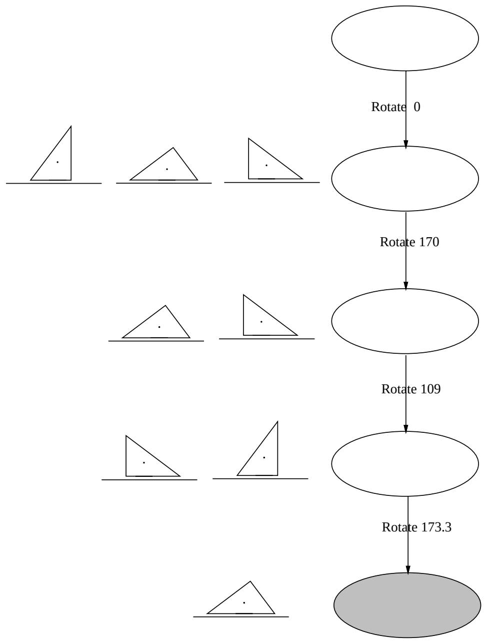


Figure 6.16: Sensorless plan to orient the triangle with shape uncertainty. r_c and r_v are 2 mm and the longest edge of the triangle has length 63.2 mm. This plan is one step longer than the sensorless plan with no shape uncertainty in Figure 5.15.

6.7 Sensorless Orienting with Shape Uncertainty

Can we perform sensorless orienting of a part with shape uncertainty? The answer is yes for certain parts. See Figure 6.16 for an example sensorless plan. Plans can be generated using breadth-first search just as for the case with no shape uncertainty and can use deterministic and nondeterministic actions. Completeness of a search-based sensorless planner, as with the sensor-based planner, depends on the accuracy of the action ranges. If a sensorless plan exists for a given set of uncertainty values, a sensor-based plan also exists for those values.

For a single qualitative shape, the set of stable edges does not change over the shape uncertainty values, and the steps of the push function change only in width and height. This suggests that if additional conditions are satisfied, we can apply Goldberg's backchaining algorithm [76] to generate a sensorless orienting plan with shape uncertainty. Further, this will provide a polynomial-time planning algorithm.

Consider the edge with the largest deterministic (i.e. guaranteed) resting range. If this range is greater than the maximum separation of the stable orientations of two adjacent stable edges, the part can be brought to the edge with the largest resting range from either of these two stable edges. If the combined deterministic range of these two stable edges is greater than the maximum separation of the stable orientations of two or more adjacent edges, we can ultimately bring the part to the edge with the largest deterministic range. We continue this backchaining process until the resting range of the set of stable edges is 360 degrees or the deterministic resting range of the set of stable edges is smaller than the maximum separation of the stable edges of the next larger set of adjacent edges. We can generate an orienting plan using the backchaining algorithm in the

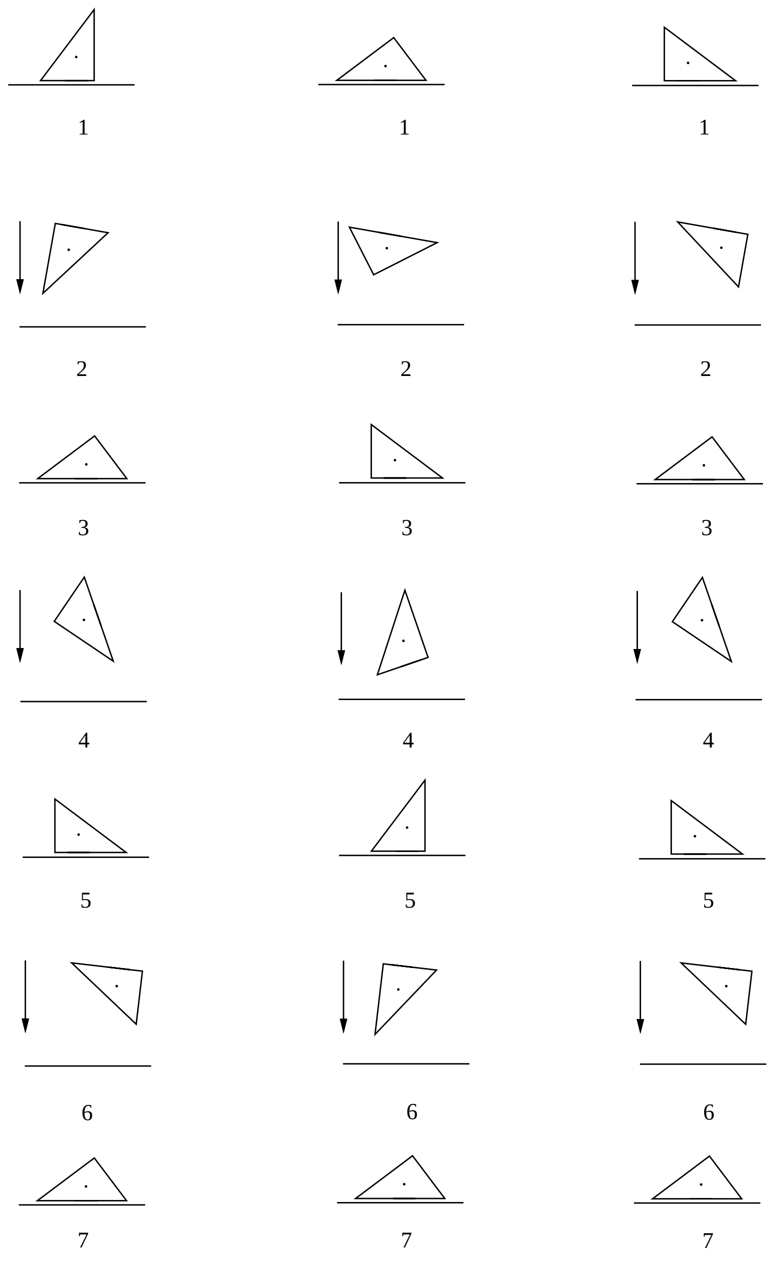


Figure 6.17: Execution trace of the sensorless plan in Figure 6.16 to orient the triangle with shape uncertainty. A comparison with the execution trace in Figure 5.16 of the sensorless plan with no shape uncertainty shows that the plan with shape uncertainty requires an additional action.

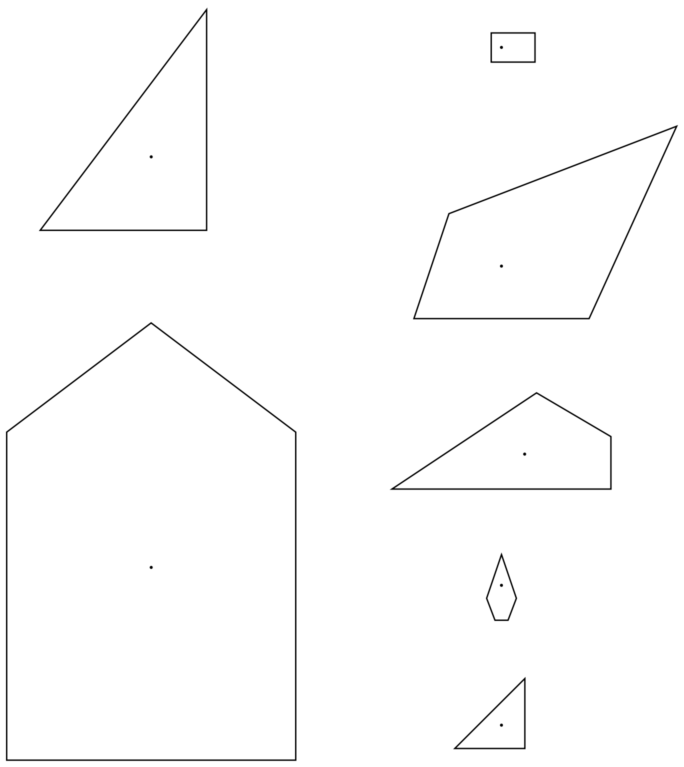


Figure 6.18: Example part shapes with shape uncertainty that planners were tested on.

former case, which also provides a sufficient condition for existence of a sensorless plan.

6.8 Implementation

I implemented sensor-based and sensorless planners in Common Lisp for orienting with shape uncertainty. These planners currently generate plans for parts that have a single qualitative shape. Given a nominal part shape along with radius values of the center of mass and vertex uncertainty circles and sensor noise, they return a plan when they can find one and indicate failure otherwise. Some example parts we tested the planners on are shown in Figure 6.18.

The sensor-based planner performs breadth-first AND/OR search to generate sensor-based orienting plans for parts with shape uncertainty. The implementation is more involved than for the case with no shape uncertainty since for each action we have to keep track of the state it is performed in, and the set of possible resulting states. An example sensor-based plan is shown in Figure 6.19. For the parts in Figure 6.18, going from top to bottom, left to right, the sensor-based planner took an average of 0.192 secs, 1.870 secs, 0.756 secs, 0.262 secs, 0.262 secs, 0.224 secs, and 0.188 secs respectively on a SPARC ELC.

The sensorless planner uses breadth-first search to find sensorless orienting plans. This planner, which is an extension of the search-based sensorless planner of Section 5.4.1, uses deterministic and nondeterministic actions. See Figures 6.16 and 6.17 for an example plan generated by the planner. Completeness of this planner, as with the sensor-based planner, depends on the accuracy of the

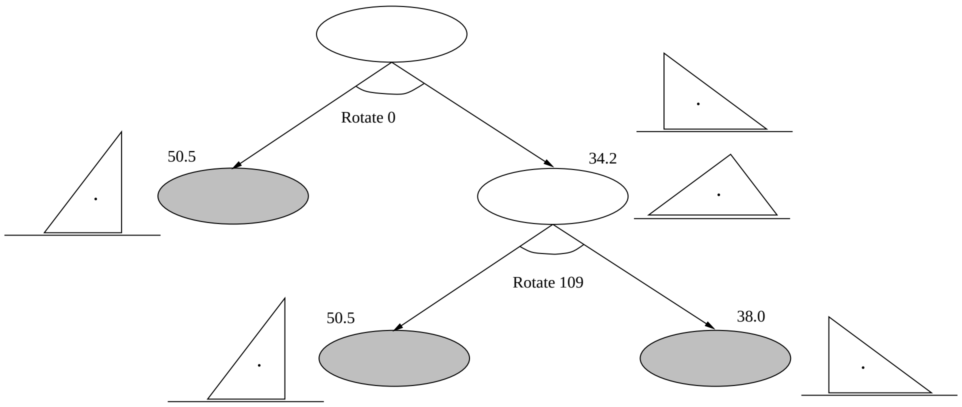


Figure 6.19: Sensor-based plan with shape uncertainty for a triangle. r_c and r_v are 2 mm and the longest edge of the triangle has length 63.2 mm.

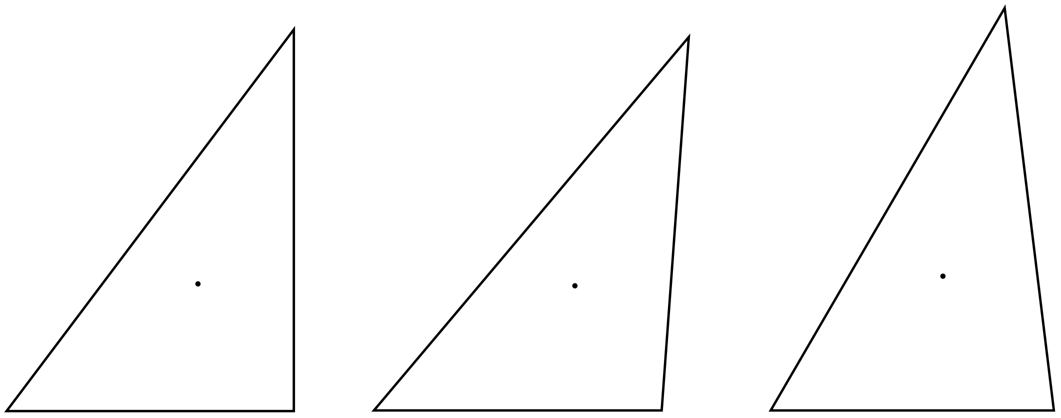


Figure 6.20: Part shapes used in experiments to test plans with shape uncertainty. Parts are shown actual size. Going left to right, the nominal triangle, triangle with r_v and r_c of 1 mm, and triangle with r_v and r_c of 2 mm. The nominal length of the longest edge is 63.2 mm.

action ranges. Not surprisingly, sensor-based orienting plans typically work for larger values of the shape uncertainty parameters than sensorless orienting plans.

I tested sensor-based and sensorless orienting plans with shape uncertainty using the Adept 550 robot and conveyor belt system described in Chapter 5. A test set of triangles we used is shown in Figure 6.20. I ran 30 trials of the sensor-based plan (Figure 6.19), with 10 trials for each part. I also ran 15 trials of the sensorless plan (Figure 6.16), with 5 trials for each part. Both plans succeeded on all trials. During the experiments I observed that nondeterministic actions typically take longer to execute since the center of mass is closer to the contact normal.

6.9 Multiple Part Shapes with Shape Uncertainty

Assume we have a set of multiple part shapes to be oriented, none of which undergo qualitative shape changes. Can we orient these multiple part shapes in the presence of shape uncertainty? We can identify sufficient conditions for existence of sensor-based orienting plans for certain cases. For example, if each part has a guaranteed stable orientation with a unique sensor reading in the presence of shape uncertainty, it is possible to orient any part from the set of parts. To see this, assume the part is repeatedly rotated until it comes to rest in a state with a unique diameter value. When some of the parts do not have orientations with unique sensor values, a plan is no longer guaranteed to exist. We can extend the AND/OR search planner to handle multiple part shapes and use it to check for plan existence. However we are not guaranteed that the planner is complete.

6.10 Qualitative Shape Changes

We have so far assumed that the set of stable edges does not change over all instantiations of the shape uncertainty parameters. When the set of stable edges can change due to shape uncertainty, a part can exhibit qualitatively different mechanical behaviors. For example, a part with two stable edges in its nominal shape can behave like a part with three stable edges for some instantiations of its variational class. This occurs when the center of mass is in the expanded stability region of at least one edge without also being in its shrunken stability region. We refer to a part whose set of stable edges can vary as a part with *multiple qualitative shapes* and refer to an edge that can be stable or unstable depending on the particular shape instantiation as a *sometimes-stable edge*. For a part with multiple qualitative shapes, a plan generated for one qualitative shape of the part may fail for another qualitative shape of the part (Figure 6.21).

Interestingly, plans that work even in the presence of qualitative shape changes can sometimes be found. See Figure 6.22 for such an example plan. In this section, we describe the issues to be dealt with in generating plans for parts with qualitative shape changes and suggest possible solutions.

The effect of qualitative shape changes on the orienting problem is that we now have multiple transition graphs for each part, where each transition graph represents a different qualitative shape of the part (Figure 6.23). Further we have to determine the angle ranges associated with each of the arcs of the graphs and perform planning in the space of these multiple graphs. A guaranteed orienting plan is one that can successfully orient every part shape corresponding to any of the graphs.

We classify parts based on the center of mass location and the qualitative shape changes caused by shape uncertainty for a given set of uncertainty values as follows:

1. Center of mass inside shrunken stability regions of all edges. There are no qualitative shape changes and plans can be generated as previously described.

2. Center of mass inside shrunken stability regions of stable edges and outside expanded stability regions of all other (unstable) edges. These parts do not undergo qualitative shape changes and plans can be generated as previously described.

3. Center of mass inside shrunken stability regions of stable edges, inside expanded stability regions of some unstable edges without being inside their shrunken stability regions, and

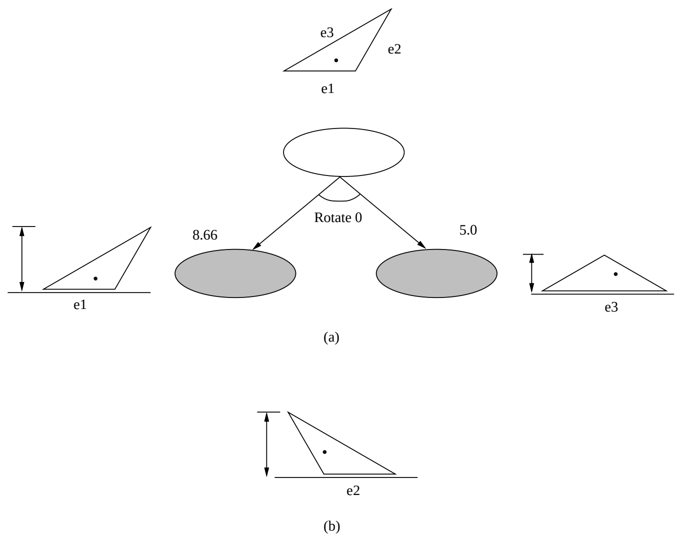


Figure 6.21: The effect of qualitative shape changes due to shape uncertainty. (a) An orienting plan for the nominal part shape. The edges e_1 and e_3 are stable and edge e_2 is unstable for the nominal shape. The longest edge e_3 has length 17.32 mm. (b) For shape uncertainty values r_v and r_c of 1 mm, the edges e_1 and e_3 are always stable and edge e_2 is stable for some instantiations of the shape and has the same diameter value as edge e_1. The plan fails since it cannot distinguish edges e_1 and e_2 for such instances.

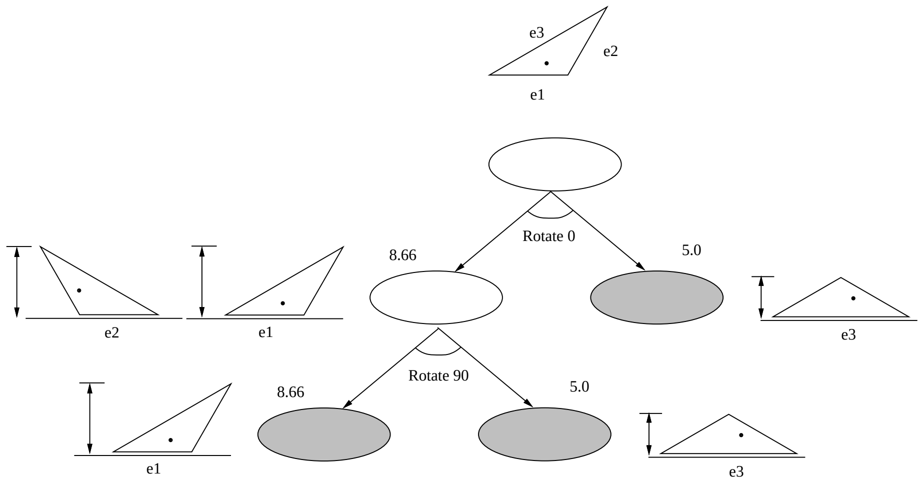


Figure 6.22: A plan that works for the triangle of Figure 6.21 even with qualitative shape changes. This plan was generated by hand.

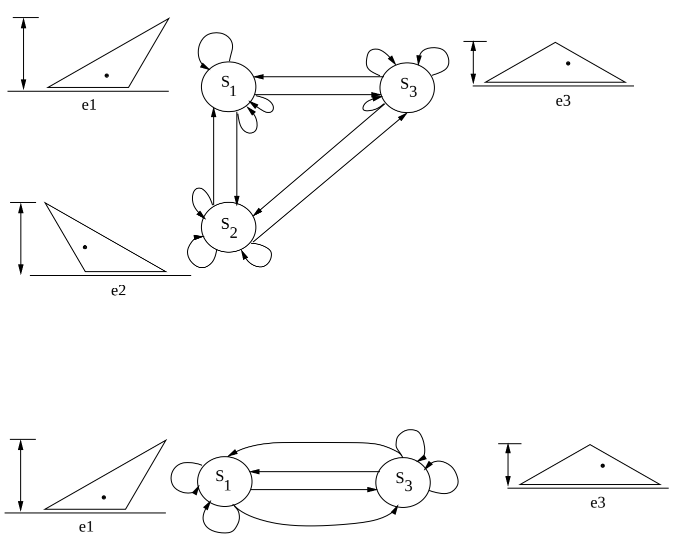


Figure 6.23: Transition graphs for the two qualitative shapes of the triangle. Some of the directed arcs have been omitted for clarity. Edges e_1 and e_3 are always stable, and edge e_2 is sometimes-stable. The number of transition graphs grows exponentially with the number of sometimes-stable edges.

outside expanded stability regions of all other unstable edges. These parts can undergo qualitative shape changes. See Figure 6.24 (a) for an example.

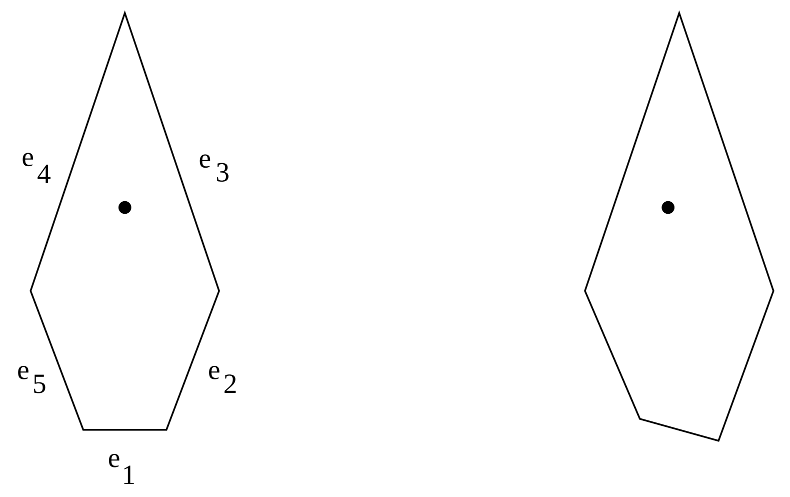
4. Center of mass outside shrunken stability regions of all edges, inside expanded stability regions of some edges, and outside expanded stability regions of remaining edges. These parts can undergo qualitative shape changes. See Figure 6.24 (b) for an example.

6.10.1 Identifying the Multiple Qualitative Shapes

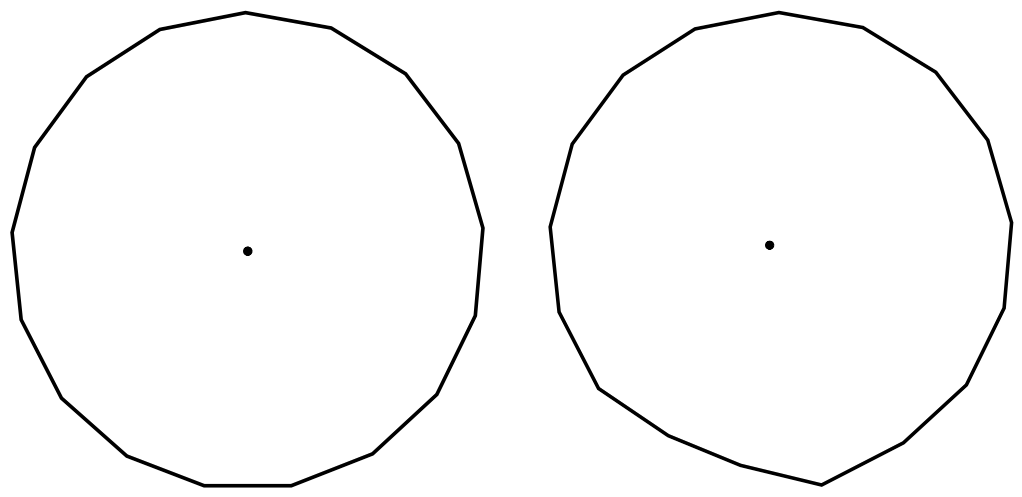
To identify all qualitative shapes of a part, we first identify the set of stable, unstable, and sometimes-stable edges by performing the point inclusion test for the center of mass with the shrunken and expanded stability regions of every edge as described in Section 6.3. Each qualitative shape is defined by all the stable edges and some subset of the sometimes-stable edges. We generate a qualitative shape corresponding to each subset of sometimes-stable edges.

6.10.2 Generating Plans for Multiple Qualitative Shapes

Qualitative shape changes mean that some edges are stable for certain values of the shape uncertainty parameters and unstable for other values of the parameters. Consequently we have to find plans that work for any valid combination of stable and unstable edges. Let there be s sometimes-stable edges, edges that can be stable or unstable. There are potentially 2^s combinations of these sometimes-stable edges. Therefore the number of qualitative shapes and the number of transition graphs for a part with s sometimes-stable edges is 2^s. Conceptually, we can treat it as a multiple part shape orienting problem, where each qualitative shape is a different part.



(a)



(b)

Figure 6.24: Example parts that can undergo qualitative shape changes. For each part, the nominal part shape is shown at left and a qualitatively different instantiation is shown at right. (a) A 5-gon with a single sometimes-stable edge. Edge e_1 is sometimes-stable, edges e_2 and e_5 are always unstable, and edges e_3 and e_4 are always stable for the specified shape uncertainty values. (b) A 17-gon with several sometimes-stable edges. It was created by replicating the triangular wedge formed by edge e_1 and the center of mass of the 5-gon sixteen times and adding an additional thinner wedge.

We sketch the procedure to generate plans when there are multiple qualitative shapes for a part. For each qualitative shape, we determine its resting ranges. Using the resting ranges, we determine the action ranges for each stable and sometimes-stable state in each qualitative shape. For every state that is stable or sometimes-stable, we find its action ranges over all qualitative shapes and compute its composite action ranges over all qualitative shapes in much the same way as we compute overlap ranges for parts with a single qualitative shape. Note that actions that belong to the composite action ranges can result in multiple outcomes — they are potentially nondeterministic. For a set of indistinguishable states, we compute the corresponding overlap ranges using the composite action ranges for the individual states. From the overlap ranges, we find representative actions and can then perform breadth-first AND/OR search to find a plan.

The principal details to be worked out in implementing this planning procedure for parts with multiple qualitative shapes relate to determining the resting and action ranges for every qualitative shape of the part. Perhaps the most straightforward approach is to compute the most conservative estimate of the resting ranges of each stable and sometimes-stable state.

6.11 Future Work and Open Issues

Much work remains in orienting parts with shape uncertainty, including determining center of mass loci accurately, computing the maximum amount of uncertainty for which a plan exists, and developing more realistic shape uncertainty models to faithfully capture shape variations in manufactured parts. We can obtain bounds on the maximum shape uncertainty values by setting the vertex or center of mass uncertainty circle radius to zero while varying the other radius value. To more faithfully model shape variations in manufactured parts, we might model an edge of a polygon as a set of multiple connected line segments. To model cases when only some edges of the polygon have shape uncertainty, we could specify vertex uncertainty circles with differing radii. An important objective for future work is to broaden the class of parts to which this analysis can be applied. Nonconvex parts, parts with curved edges, 3-D parts are prime candidates for such extensions.

Our definition of a good plan is a guaranteed plan and therefore conservative. It would be useful to develop a planner which generates plans that either orient a part or recognize when they cannot orient the part. Such plans, similar to Donald's [53] Error Detection and Recovery (EDR) plans in the context of motion planning under uncertainty, are especially useful when we cannot find a guaranteed plan for a part. EDR plans ought to handle larger values of shape uncertainty and a broader class of parts. Further, such plans can potentially be used as metrological tools to identify and eliminate parts that do not meet tolerance specifications.

The analysis here can be extended to other parts orienting operations such as the sensorless 1JOC (Chapter 4), parallel-jaw grasping [76], and tray tilting [66]. We have performed preliminary experiments with the sensorless 1JOC system which demonstrate that it can indeed orient parts with shape uncertainty. In fact, this approach can be applied to any class of operations whose deterministic and nondeterministic action ranges can be determined.

Identifying tolerance models that allow analysis of the effects of shape uncertainty on parts orienting and whose results can be transformed to other tolerance models would be useful. It would be interesting to apply our techniques to other tolerance models such as the least material, maximal material model of Requicha [152, 154], or the perfect orientation model of Latombe and Wilson [106]. Whenever the shape uncertainty model allows us to compute the transition ranges and variations in edge orientation, plan generation should be possible. In fact, assuming a perfect orientation model like that of Latombe and Wilson will simplify the orienting problem since there are no variations in the stable orientations of the edges.

It is tempting to model the parts orienting with shape uncertainty problem as a partially observable Markov decision process (POMDP), where the transition probabilities give the probability of transitioning between edges for a given action and the part diameter is the partially observable variable related to the orientation state variable. (See Howard [92] for a variety of Markovian models and solution techniques to model probabilistic dynamic systems and Bertsekas [18] for the use of dynamic programming techniques in stochastic control and Markov decision problems.) However a closer examination reveals that the orienting process is not Markovian. The transition probabilities of an edge are not independent of the transition probabilities of other edges since they are related by the part geometry. For example, if the resting range of an edge increases, the probability of transitioning to that edge from several other edges can simultaneously increase.

We can estimate the probability distributions of the shape uncertainty parameters from knowledge of the manufacturing process. We can then use this information to generate plans that minimize the expected execution length. For example, if an edge has a 90% probability of being the initial resting edge, the minimum expected length plan will be biased towards determining the orientation of that edge quickly. A plan that minimizes expected length can have a worst case length that is greater than that of the plan that minimizes the worst-case plan length.

Probabilistic modeling can be useful in modeling the nondeterminism of the actions. By observing the results of actions over a sample set of parts, we can estimate the transition probabilities of the actions. See Christiansen [43] and Brost and Christiansen [33] for examples of tasks whose action probabilities are empirically estimated. This approach can reduce the need for analytical computation of the results of actions. On-line learning of the relevant parameters of part geometry for each part by the orienting system is another approach. An important issue in this case is minimizing the number of steps to orient the part, which depends on the number of steps to learn the relevant part parameters.

Randomization during action selection is another approach. (See Erdmann [63] for a discussion of the use of randomization in manipulation tasks.) Since randomization works well when goal states are recognizable, it will be effective especially when there are several states with unique sensor values and large resting ranges. The randomized action selection will lead to one of the stable orientations that has a unique sensor value.

6.12 Conclusion

In this chapter we explored the effects of shape uncertainty on parts orienting and demonstrated that it is possible to orient parts in the presence of shape uncertainty. This work is among the first to explore the manipulation of parts with shape uncertainty. We have shown that the operational effect of shape uncertainty is to introduce nondeterminism. We have demonstrated that it is possible to generate reliable sensor-based and sensorless orienting plans in the presence of shape uncertainty and characterized a class of parts that can be oriented under shape uncertainty. We believe that similar analyses can be applied to other orienting tasks such as tray-tilting, parallel-jaw grasping, and the sensorless 1JOC.

Chapter 7

Conclusion

This thesis has presented algorithms and systems to automatically perform transfer and orienting of polygonal parts. The use of simple hardware elements coupled with the use of software that captures the task mechanics and geometry enables these systems to handle a broad class of parts.

I have considered a set of parts feeding tasks and for each task, modified the robot and task parameters to study how this influences the task. For the planar pose problem of Chapter 2, I assumed the robot has three degrees of freedom in the plane and uses a flat fence instead of a gripper to manipulate parts. In Chapter 3, we restricted the robot to have a single degree of freedom and by adding a constant-velocity conveyor, demonstrated that the robot could perform planar parts transfer. Chapter 4 continued this process of reducing the robot's hardware capabilities by eliminating the camera and showing that a one joint robot over a conveyor can perform sensorless orienting. In Chapter 5 I reintroduced sensors capable of providing partial state information and demonstrated that these sensors reduce plan execution length and permit multiple part shapes to be oriented with a single plan. Finally I relaxed the assumption of perfectly known part shape in Chapter 6 and identified a class of parts can be oriented even with shape uncertainty.

For each task, I have sought to characterize the capabilities and completeness properties of the action sets and the planners. I have proven the completeness of the actions and the planners for the parts transfer tasks, and developed linear and nonlinear programming formulations to solve them efficiently. I have shown completeness of the orienting systems and illustrated the advantages of using partial sensors during orienting. The sensors speed up the orienting process, permit orienting of multiple part shapes with a single plan, and allow orienting with larger values of shape uncertainty.

The tasks in this thesis are motivated by automated manufacturing applications. Parts transfer and orienting is an important component of automated assembly and there is pressing need for flexible systems that can handle new parts with minimum setup and changeover times. By using simple effectors that are not part specific and simple sensors that are robust and inexpensive, we can develop minimalist systems that are flexible and cost effective. Since the effectors do not permit the robot to grasp the part rigidly and the sensors do not provide complete state information, our analyses and planners are correspondingly more sophisticated to successfully accomplish the tasks. We use knowledge of the mechanics and geometry of the tasks to solve them. Pushing is the basic mechanical operation used to manipulate parts in these tasks. Pushing is a form of nonprehensile manipulation, and serves as an illustrative example for other manipulation operations. We have shown the importance of part shape for these parts transfer and orienting tasks and described the effects of uncertainty in part shape on orienting.

The work in this thesis has concentrated on flexible feeding of polygonal parts. While there is a large class of industrial parts that are flat or have prismatic shapes and can thus be treated as 2-D parts, there is an even larger class of 3-D parts with complex geometries. All six degrees of freedom of these parts have to be controlled, as opposed to the three degrees of freedom of polygonal parts. The next challenge is to develop effective manipulation techniques for 3-D parts that can be used to automatically generate feeding solutions for new parts. Systematic characterization of the mechanical operations used by current industrial systems such as bowl feeders and vibratory orienters is important in this context. The basic principles and techniques we have used and illustrated, such as the use of simple effectors and simple sensors, knowledge of the mechanics and part geometry, and characterization of the nondeterministic effects of shape uncertainty can then be applied to a broader class of industrial parts. For example, a part in a bowl feeder that encounters a wiper blade behaves in a similar manner to a part that is pushed by a fence. We have to consider the entire integrated system for parts feeding and develop techniques to perform robust singulation of parts and speed up the parts transfer and orienting process, possibly by using operations that incorporate dynamic effects.

7.1 Thesis Contributions

This thesis has demonstrated the use of the task mechanics and geometry in developing algorithms and systems with provable properties for a variety of parts transfer and orienting problems. My main contributions in this thesis are:

- Proved completeness of the class of linear normal pushes for planar parts transfer. I described a linear programming formulation and using it, implemented a polynomial-time complete planner.
- Demonstrated with the 1JOC system that a single revolute joint effector over a conveyor can perform planar parts transfer. I developed a nonlinear programming formulation to automatically generate manipulation plans for the 1JOC system.
- Showed that we can perform sensorless orienting of any polygonal part using a single revolute joint over a conveyor and implemented a planner to generate such plans.
- Demonstrated that using partial sensors with parts orienting operations can significantly reduce the plan length and permits orienting of multiple part shapes with a single plan. I implemented exponential-time planners that generate optimal plans and a polynomial-time greedy planner.
- Analyzed the nondeterministic effects of shape uncertainty on parts orienting, identified conditions under which parts can be oriented in the presence of shape uncertainty, and implemented a planner to generate these plans.
- Implemented and experimentally demonstrated the parts transfer and orienting systems using Puma and Adept robots.

7.2 Future Work

I present here some open problems and challenging extensions to the problems described in the thesis.

A significant challenge is extending our analyses to 3-D parts and parts with more general geometries, for example, parts with curved edges and faces. Some of our results on orienting multiple part shapes can be applied to polyhedral parts by considering each stable face of the polyhedron separately and treating the resulting convex hull projection as a different part. However we have to deal with out of plane forces and multiple contact profiles.

From an industrial viewpoint, speeding up these systems to accomplish faster parts feeding is essential. Industrial parts feeding systems are commonly required to feed parts at the rate of a part every 4–6 seconds. As we achieve these faster rates, it may become necessary to study the dynamic effects of the manipulation operations and to explicitly account for them in the planning process. Similarly, automated singulation of bulk-fed parts to be oriented is an important capability to make our parts orienting systems industrially viable.

We have used simple end-effectors, primarily flat fences, for our manipulation operations. Since part shape and fence shape are tightly linked in determining the results of operations, it would be worthwhile to investigate the design of fence shapes to suit the feeding task. This is related to the problem of selecting wiper designs for a bowl feeder. It would be interesting to develop a translational analog of the 1JOC system, using minimal translational degrees of freedom and an appropriately chosen fence geometry.

Determining the inherent computational complexity of the orienting task with partial sensing is an open problem. Finding polynomial-time algorithms that generate solutions that are within a known factor of the optimal solution would be useful.

We have to develop techniques to orient parts with qualitative shape changes so our system can orient a larger class of parts with shape uncertainty. We also have to determine accurate bounds on the center of mass locus for polygons with shape uncertainty since the center of mass location influences part motion during alignment. Treating manipulation operations with more complicated mechanics in the presence of shape uncertainty will be challenging. For example, parallel-jaw grasping involves contact of the two parallel planes of the gripper with the part. Similarly, during bowl feeding, the part makes multiple simultaneous contacts with the bowl features and all possible combinations of these contacts have to be analyzed to determine the motion of the body in the presence of shape uncertainty.

Our parts orienting with shape uncertainty planner relies on a worst-case analysis. It would be useful to generate plans that minimize the expected length, particularly if we can estimate the probability distributions of the shape uncertainty parameters from knowledge of the manufacturing process.

Exploiting the task mechanics is a very effective method to perform manipulation. Developing automatic task-level planning systems that rely on an analysis of the task mechanics and geometry, and identifying the capabilities of the planners enables the design of robust feeder systems. A logical next step is to integrate parts transfer and orienting systems with assembly and fixturing systems. Such integrated systems would enable designers to analyze all aspects of the integrated assembly process at the design stage, cutting down time to market and enabling virtual prototyping.

Appendix A

The Peshkin Distance

The minimum distance a fence must translate in contact with an object to ensure that an edge of the object rotates into alignment with the fence is determined by Peshkin and Sanderson [141]. For the case when there is no slip between the object and the fence, this distance corresponds to the center of rotation of the object consistent with sticking that causes the slowest rotation. Assuming sticking-slowest behavior, the Peshkin distance m_i for the ith reorientation is

$$m_i = \frac{a^2 + c^2}{2c} \left(\ln \left| \frac{1 - \cos \beta_{i2}}{1 + \cos \beta_{i2}} \right| - \ln \left| \frac{1 - \cos \beta_{i1}}{1 + \cos \beta_{i1}} \right| \right)$$

where a is the radius of the smallest circle centered at the center of mass that circumscribes the object, c is the distance from the center of mass to the point of contact, and β is the angle between the line of motion and the line from the point of contact to the center of mass. β_{i1} and β_{i2} are the initial and final values of β at the ith reorientation. (See Figure A.1.)

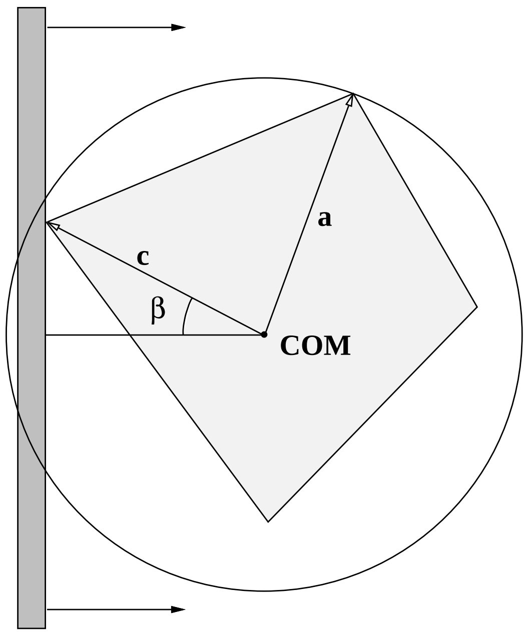


Figure A.1: Variables to calculate the Peshkin distance. From Peshkin and Sanderson [141].

Appendix B

LED Sensor to Measure Part Diameter

I constructed a simple sensor to measure the diameter of parts aligned against a fence. The sensor consists of a linear array of LEDs arranged perpendicular to the fence, with a corresponding parallel array of phototransistors at the other end of the fence. To measure the part diameter, we pulse the LEDs on in sequence and read the output of the opposite phototransistors. If the part is present between the LED and its corresponding phototransistor, it blocks the light to the phototransistor. Pulsing eliminates optical crosstalk between the LEDs and phototransistors, and a red filter avoids spurious readings from ambient light. We used high intensity red LEDS with narrow output beams (Hewlett Packard HLMP-8103) and wide angle phototransistors (Quality Technologies L14N2). It would have probably have been better to use narrow angle phototransistors to reduce optical crosstalk and the effects of ambient lighting. The LEDs and phototransistors were 7.5 inches apart on the conveyor. (We tested the sensor with the LED-phototransistor pairs up to 10 inches apart and found it to work reliably.)

When there is light incident on the phototransistor, its output voltage is low. When there is no light incident on it, its output voltage is high. The output of the phototransistor is sent to a comparator (National Semiconductor LM 139J) in an inverting comparator with hysteresis circuit, to convert it to a digital signal in an appropriate voltage range. The threshold voltage of the comparator is empirically selected for the LED-phototransistor distance by setting the resistors to appropriate values. The output of the comparator is fed to an op amp (LM 324) that acts as a unity gain voltage follower. The buffered output voltage of the op amp is fed to the digital I/O board of the Adept robot.

So the output of the sensor is a series of 1's and 0's, where a 1 corresponds to a blocked phototransistor and 0 to an unblocked phototransistor. The diameter of the part can be determined from the spacing of the LEDs, and the set of blocked phototransistors. This is a finite resolution sensor whose resolution is determined by the spacing of the LEDs and phototransistors. The resolution of the implemented sensor was 7/16 inch.

Appendix C

Estimating the Center of Mass Locus

Variations in part shape can result in uncertainty in the location of the center of mass of the part. Our shape uncertainty model does not assume parts have a uniform mass density and hence the center of mass location and its uncertainty circle radius can be specified as independent variables that are not functions of the vertex locations and uncertainty radii. We now consider the case of a polygon with uniform density whose center of mass locus depends on the vertex locations and the radii of the vertex uncertainty circles.

We first tried to analytically bound the possible locations of the center of mass. To do this, we triangulate the polygonal part and find the locus of center of mass locations of each constituent triangle. The center of mass of a triangle with vertices at (x_1, y_1), (x_2, y_2), and (x_3, y_3) is located at

$$\left(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3}\right).$$

This implies that the locus of the center of mass of a triangle with vertex uncertainty circles of radius r_v is a circle of radius r_v. The mass of a triangle with uniform density is proportional to its area. The area of the triangle, given by

$$\frac{x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)}{2},$$

also varies with the vertex locations.

From the expressions for the center of mass and the area of the constituent triangles, we obtain the expression for the center of mass of the polygon in terms of the polygon vertex coordinates. Unfortunately, the center of mass locus of an arbitrary convex polygon does not have a simple result similar to that of a triangle. Assuming the shape instantiations of the constituent triangles are independent, finding the extremal locations of the center of mass of the polygon amounts to solving a nonlinear programming problem whose objective function is the x or y coordinate of the polygon center of mass and whose constraints are the constraints on the possible locations of the vertices. We can use the smallest circle which encloses these extremal points to obtain a conservative bound on the locus of the center of mass. See Figure C.1.

Finding the exact bounding locus of the center of mass is an open problem. See Bern *et al.* [17] for related work on estimating the locus of the centroid of a set of points whose weights vary within known ranges.

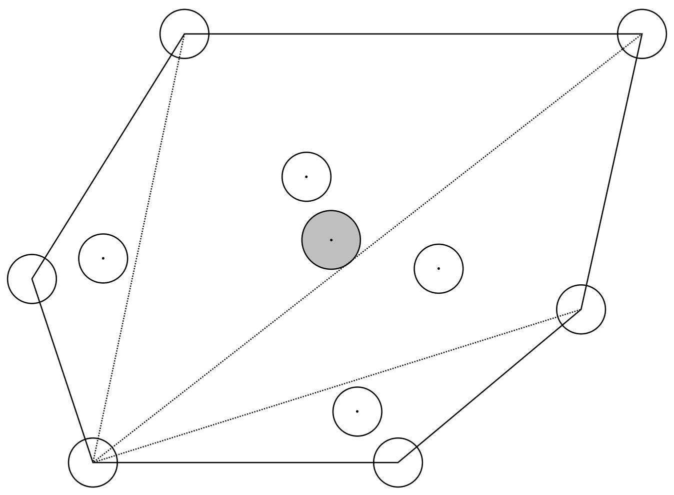


Figure C.1: Computation of the locus of the center of mass of a part with shape uncertainty. The center of mass loci of the constituent triangles are circles with the same radius as the vertex uncertainty circles. The smallest circle that encloses the extremal center of mass locations, represented by the shaded circle, provides a conservative bound on the center of mass locus of the polygon.

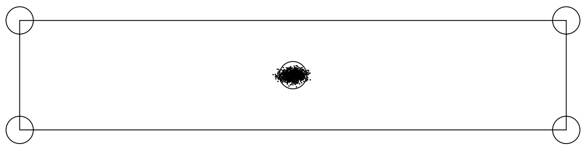


Figure C.2: A plot of the center of mass locations of 1000 randomly generated instances of a rectangle consistent with its shape uncertainty values. The circle of radius r_v drawn at the nominal center of mass location illustrates that the bounding circle for the center of mass locus can have a radius greater than r_v. r_v is the vertex uncertainty circle radius.

For convenience, we estimate center of mass variations by randomly sampling the variational class of valid part shapes. We allow each vertex to assume a random location within its uncertainty circle, determine the resulting uncertainty polygon, and compute its center of mass. By performing this random sampling over several thousand trials (100,000 in our tests), we obtain an indication of the bounds of the center of mass locus. Such sampling shows that the center of mass locus of a polygon (that is not a triangle) with shape uncertainty can lie outside the circle of radius r_v centered at the nominal center of mass. See Figure C.2 for an example. We identify the smallest enclosing circle from the extremal center of mass locations found during random sampling, and use the circle enlarged by a safety factor as the center of mass locus of the polygon.

Bibliography

[1] T. Abell and M. A. Erdmann. Stably supported rotations of a planar polygon with two frictionless contacts. In *IEEE/RSJ International Conference on Robots and Systems*, pages 3:411–418, Pittsburgh, PA, Aug. 1995.

[2] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson. Task-level robot learning: Juggling a tennis ball more accurately. In *IEEE International Conference on Robotics and Automation*, pages 1290–1295, Scottsdale, AZ, May 1989.

[3] Y. Aiyama, M. Inaba, and H. Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *IEEE/RSJ International Conference on Robots and Systems*, pages 136–143, Yokohama, Japan, 1993.

[4] S. Akella, W. H. Huang, K. M. Lynch, and M. T. Mason. Planar manipulation on a conveyor with a one joint robot. In G. Giralt and G. Hirzinger, editors, *Robotics Research: The Seventh International Symposium*, pages 265–276, London, UK, 1996. Springer.

[5] S. Akella, W. H. Huang, K. M. Lynch, and M. T. Mason. Sensorless parts feeding with a one joint robot. In *Second Workshop on the Algorithmic Foundations of Robotics*, Toulouse, France, July 1996.

[6] S. Akella and M. T. Mason. Posing polygonal objects in the plane by pushing. In *IEEE International Conference on Robotics and Automation*, pages 2255–2262, May 1992.

[7] S. Akella and M. T. Mason. Parts orienting by push-aligning. In *1995 IEEE International Conference on Robotics and Automation*, pages 414–420, May 1995.

[8] P. Alevizos, J. D. Boissonnat, and M. Yvinec. On the order induced by a set of rays: Application to the probing of non convex polygons. In *IEEE International Conference on Robotics and Automation*, 1989.

[9] J. C. Alexander and J. H. Maddocks. Bounds on the friction-dominated motion of a pushed object. *International Journal of Robotics Research*, 12(3):231–248, 1993.

[10] H. Arai and O. Khatib. Experiments with dynamic skills. In *1994 Japan-USA Symposium on Flexible Automation*, pages 81–84, Kobe, Japan, July 1994.

[11] H. Arai and S. Tachi. Position control of a manipulator with passive joints using dynamic coupling. *IEEE Transactions on Robotics and Automation*, 7(4):528–534, Aug. 1991.

[12] *Dimensioning and Tolerancing, ANSI Y14.5M-1982.* American Society of Mechanical Engineers, United Engineering Center, New York, 1982.

[13] Z. Balorda and T. Bajd. Reducing positioning uncertainty of objects by robot pushing. *IEEE Transactions on Robotics and Automation*, 10(4):535–541, Aug. 1994.

[14] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2/3/4):121–155, Aug. 1993.

[15] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms.* John Wiley and Sons., 1979.

[16] D. R. Berkowitz and J. F. Canny. Designing parts feeders using dynamic simulation. In *1996 IEEE International Conference on Robotics and Automation*, pages 1127–1132, Minneapolis, MN, Apr. 1996.

[17] M. Bern, D. Eppstein, L. J. Guibas, J. E. Hershberger, S. Suri, and J. Wolter. The centroid of points with approximate weights. In *Proc. 3rd Eur. Symp. Algorithms*, Lecture Notes in Computer Science, No. 979, pages 460–472. Springer-Verlag, 1995.

[18] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[19] A. Bicchi and R. Sorrentino. Dexterous manipulation through rolling. In *IEEE International Conference on Robotics and Automation*, 1995.

[20] K. Böhringer, R. Brown, B. Donald, J. Jennings, and D. Rus. Distributed robotic manipulation: Experiments in minimalism. In *International Symposium on Experimental Robotics*, 1995.

[21] K. F. Böhringer, V. Bhatt, and K. Y. Goldberg. Sensorless manipulation using transverse vibrations of a plate. In *IEEE International Conference on Robotics and Automation*, pages 1989–1996, Nagoya, Japan, May 1995.

[22] K.-F. Böhringer, B. R. Donald, R. Mihailovich, and N. C. MacDonald. Sensorless manipulation using massively parallel microfabricated actuator arrays. In *IEEE International Conference on Robotics and Automation*, pages 826–833, San Diego, CA, May 1994.

[23] G. Boothroyd, C. Poli, and L. E. Murch. *Automatic Assembly.* Marcel Dekker, 1982.

[24] G. Boothroyd, A. H. Redford, C. Poli, and L. E. Murch. Statistical distributions of natural resting aspects of parts for automatic handling. *Manufacturing Engineering Transactions, Society of Manufacturing Automation*, 1, 1972.

[25] D. L. Brock. Enhancing the dexterity of a robot hand using controlled slip. In *IEEE International Conference on Robotics and Automation*, pages 249–251, Philadelphia, PA, Apr. 1988.

[26] R. Brockett. Nonlinear systems and differential geometry. *Proceedings of the IEEE*, 64(1), Jan. 1976.

[27] M. Brokowski, M. Peshkin, and K. Y. Goldberg. Curved fences for part alignment. In *IEEE International Conference on Robotics and Automation*, pages 3:467–473, Atlanta, Georgia, May 1993.

[28] R. A. Brooks. Symbolic error analysis and robot planning. *International Journal of Robotics Research*, 1(4):29–68, 1982.

[29] R. C. Brost. Automatic grasp planning in the presence of uncertainty. *International Journal of Robotics Research*, 7(1):3–17, Feb. 1988.

[30] R. C. Brost. *Analysis and Planning of Planar Manipulation Tasks*. PhD thesis, School of Computer Science, Carnegie Mellon University, Jan. 1991. Technical Report CMU-CS-91-149.

[31] R. C. Brost. Dynamic analysis of planar manipulation tasks. In *IEEE International Conference on Robotics and Automation*, pages 2247–2254, Nice, France, May 1992.

[32] R. C. Brost. Personal communication, 1994.

[33] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks: A conceptual framework. *International Journal of Robotics Research*, 15(1):1–23, Feb. 1996.

[34] R. C. Brost and K. Y. Goldberg. A complete algorithm for designing planar fixtures using modular components. *IEEE Transactions on Robotics and Automation*, 12(1):31–46, Feb. 1996.

[35] R. C. Brost and R. R. Peters. Automatic design of 3-d fixtures and assembly pallets. In *IEEE International Conference on Robotics and Automation*, pages 495–502, Minneapolis, MN, Apr. 1996.

[36] R. G. Brown and J. S. Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. In *IEEE/RSJ International Conference on Robots and Systems*, pages 3:562–568, Pittsburgh, PA, Aug. 1995.

[37] S. J. Buckley. Planning compliant motion strategies. *International Journal of Robotics Research*, 8(5), Oct. 1989.

[38] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. Planning and control of robotic juggling and catching tasks. *International Journal of Robotics Research*, 13(2):101–118, Apr. 1994.

[39] M. Caine. The design of shape interactions using motion constraints. In *1994 IEEE International Conference on Robotics and Automation*, 1994.

[40] J. F. Canny and K. Y. Goldberg. RISC for industrial robotics: Recent results and open problems. In *IEEE International Conference on Robotics and Automation*, 1994.

[41] Y.-B. Chen and D. J. Ierardi. Oblivious plans for orienting and distinguishing polygonal parts. In *The 4th Canadian Conference on Computational Geometry*, 1992.

[42] L. Chrisman and R. Simmons. Sensible planning: Focusing perceptual attention. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 1991.

[43] A. D. Christiansen. *Automatic acquisition of Task Theories for Robotic Manipulation.* PhD thesis, School of Computer Science, Carnegie Mellon University, Mar. 1992. Technical Report CMU-CS-92-111.

[44] A. D. Christiansen, A. D. Edwards, and C. A. C. Coello. Automated design of parts feeders using a genetic algorithm. In *IEEE International Conference on Robotics and Automation*, pages 846–851, Minneapolis, MN, Apr. 1996.

[45] A. D. Christiansen and K. Y. Goldberg. Robotic manipulation planning with stochastic actions. In *1990 DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, 1990.

[46] V. Chvátal. *Linear Programming.* W. H. Freeman and Co., New York, 1983.

[47] R. Cole and C. K. Yap. Shape from probing. *Journal of Algorithms*, 8, 1987.

[48] J. H. Connell. A behavior-based arm controller. *IEEE Transactions on Robotics and Automation*, 5(6), Dec. 1989.

[49] P. Crouch. Spacecraft attitude control and stabilization: Applications of geometric control theory to rigid body models. *IEEE Transactions on Automatic Control*, 29(4):321–31, 1984.

[50] C. Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76:733–746, 1954.

[51] L. S. H. de Mello and A. C. Sanderson. AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2), Apr. 1990.

[52] E. D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision. In *IFAC 10th Triennial World Congress, Munich*, 1987.

[53] B. R. Donald. Planning multi-step error detection and recovery strategies. *International Journal of Robotics Research*, 9(1), Feb. 1990.

[54] B. R. Donald. Information invariants in robotics: Part I – State, Communication, and Side-Effects. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, 1993.

[55] B. R. Donald. Information invariants in robotics: Part II – Sensors and Computation. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, 1993.

[56] B. R. Donald. Toward a theory of information invariants for cooperating autonomous mobile robots. In *Sixth International Symposium on Robotics Research*, Oct. 1993.

[57] B. R. Donald and J. Jennings. Constructive recognizability for task-directed robot programming. *Robotics and Autonomous Systems*, 9:41–74, 1992.

[58] B. R. Donald, J. Jennings, and D. Rus. Information invariants for distributed manipulation. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson, editors, *Workshop on the Algorithmic Foundations of Robotics*, pages 431–457. A. K. Peters, Wellesley, Massachusetts, 1995.

[59] A. Elfes. Dynamics control of robot perception using stochastic spatial models. In *Proceedings of the International Workshop on Information Processing in Mobile Robots*, Mar. 1991.

[60] R. E. Ellis. Planning tactile recognition paths in two and three dimensions. *International Journal of Robotics Research*, 11(2), 1992.

[61] D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal of Computing*, 19(3):500–510, June 1990.

[62] M. Erdmann. Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1), 1986.

[63] M. Erdmann. Randomization in robot tasks. *International Journal of Robotics Research*, 11(5), October 1992.

[64] M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10(2/3/4):248–291, 1993.

[65] M. Erdmann. An exploration of nonprehensile two-palm manipulation: Planning and execution. In *Seventh International Symposium on Robotics Research*, Munich, Germany, Oct. 1995.

[66] M. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Journal of Robotics and Automation*, 4(4):369–379, Aug. 1988.

[67] M. Erdmann, M. T. Mason, and G. Vanecek. Mechanical parts orienting: The case of a polyhedron on a table. *Algorithmica*, 10(2/3/4):201–225, 1993.

[68] M. A. Erdmann. *On Probabilistic Strategies for Robot Tasks.* PhD thesis, Department of EECS, Massachusestts Institute of Technology, 1989.

[69] M. A. Erdmann. Action subservient sensing and design. In *1993 IEEE International Conference on Robotics and Automation*, 1993.

[70] M. A. Erdmann. Understanding action and sensing by designing action-based sensors. In *Sixth International Symposium on Robotics Research*, 1993.

[71] A. O. Farahat, P. F. Stiller, and J. C. Trinkle. On the geometry of contact formation cells for systems of polygons. *IEEE Transactions on Robotics and Automation*, 11(4):522–536, Aug. 1995.

[72] O. D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotics Research*, 5(3):27–52, Fall 1986.

[73] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[74] P. C. Gaston and T. Lozano-Perez. Tactile recognition and localization using object models: The case of polyhedra on a plane. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):257–266, May 1984.

[75] K. Y. Goldberg. *Stochastic Plans for Robotic Manipulation.* PhD thesis, School of Computer Science, Carnegie Mellon University, Aug. 1990. Technical Report CMU-CS-90-161.

[76] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2/3/4), 1993.

[77] K. Y. Goldberg. Completeness in robot motion planning. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson, editors, *Workshop on the Algorithmic Foundations of Robotics*, pages 419–429. A. K. Peters, Wellesley, Massachusetts, 1995.

[78] K. Y. Goldberg and M. T. Mason. Bayesian grasping. In *IEEE International Conference on Robotics and Automation*, pages 1264–1269, Cincinnati, OH, May 1990.

[79] K. Y. Goldberg, M. T. Mason, and M. Erdmann. Generating stochastic plans for a programmable parts feeder. In *IEEE International Conference on Robotics and Automation*, 1991.

[80] K. Y. Goldberg and M. Overmars. Personal communication, 1996.

[81] R. Govindan and A. S. Rao. Optimal strategies for recognizing polygonal parts. In *IEEE International Conference on Robotics and Automation*, 1994.

[82] S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction. Part 1. Limit surface and moment function. *Wear*, 143:307–330, 1991.

[83] S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction. Part 2. Dynamics of motion. *Wear*, 143:331–352, 1991.

[84] W. E. L. Grimson. Sensing strategies for disambiguating among multiple objects in known poses. *IEEE Journal of Robotics and Automation*, RA-2(4), Dec. 1986.

[85] W. E. L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, Fall 1984.

[86] D. D. Grossman and M. W. Blasgen. Orienting mechanical parts by computer-controlled manipulator. *IEEE Transactions on Systems, Man, and Cybernetics*, 5(5), September 1975.

[87] H. Hanafusa and H. Asada. Stable prehension by a robot hand with elastic fingers. In M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, editors, *Robot Motion.* MIT Press, 1982.

[88] T. Hasegawa, T. Suehiro, and K. Takase. A model-based manipulation system with skill-based execution. *IEEE Transactions on Robotics and Automation*, 8(5), Oct. 1992.

[89] R. Hermann and A. J. Krener. Nonlinear controllability and observability. *IEEE Transactions on Automatic Control*, AC-22(5):728–740, Oct. 1977.

[90] T. Higuchi. Application of electromagnetic impulsive force to precise positioning tools in robot system. In H. Hanafusa and H. Inoue, editors, *Robotics Research: The Second International Symposium*, pages 281–285, Cambridge, MA, 1985. MIT Press.

[91] H. Hitakawa. Advanced parts orientation system has wide application. *Assembly Automation*, 8(3):147–150, 1988.

[92] R. A. Howard. *Dynamic Probabilistic Systems.* John Wiley, 1971.

[93] W. H. Huang, E. P. Krotkov, and M. T. Mason. Impulsive manipulation. In *IEEE International Conference on Robotics and Automation*, pages 120–125, Nagoya, Japan, May 1995.

[94] S. Hutchinson. Exploiting visual constraints in robot motion planning. In *IEEE International Conference on Robotics and Automation*, 1991.

[95] H. Inoue. Force feedback in precise assembly tasks. Technical Report AI Lab Memo 308, Massachusetts Institute of Technology, Cambridge, MA, 1974.

[96] M. Inui, M. Miura, and F. Kimura. Positioning conditions of parts with tolerances in an assembly. In *IEEE International Conference on Robotics and Automation*, Minneapolis, MN, Apr. 1996.

[97] J. Jameson. *Analytic techniques for automated grasp.* PhD thesis, Department of Mechanical Engineering, Stanford University, June 1985.

[98] Y.-B. Jia and M. Erdmann. The complexity of sensing by point sampling. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson, editors, *Workshop on the Algorithmic Foundations of Robotics*, Wellesley, Massachusetts, 1995. A. K. Peters.

[99] Y.-B. Jia and M. Erdmann. Pose from pushing. In *IEEE International Conference on Robotics and Automation*, pages 165–171, Minneapolis, MN, Apr. 1996.

[100] L. Joskowicz, E. Sacks, and V. Srinivasan. Kinematic tolerance analysis. *Computer Aided Design*, 1996. To appear. Also in Second Workshop on the Algorithmic Foundations of Robotics.

[101] I. Kao and M. R. Cutkosky. Quasistatic manipulation with compliance and sliding. *International Journal of Robotics Research*, 11(1):20–40, Feb. 1992.

[102] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, 1978.

[103] J. Krishnasamy, M. J. Jakiela, and D. E. Whitney. Mechanics of vibration-assisted entrapment with application to design. In *IEEE International Conference on Robotics and Automation*, pages 838–845, Apr. 1996.

[104] M. Kurisu and T. Yoshikawa. Trajectory planning for an object in pushing operation. In *1994 Japan-USA Symposium on Flexible Automation*, Kobe, Japan, 1994.

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

[106] J.-C. Latombe and R. H. Wilson. Assembly sequencing with toleranced parts. In *Proceedings of the Third ACM Symposium on Solid Modeling and Applications*, pages 83–94, Salt Lake City, 1995.

[107] S. M. LaValle. Robot motion planning: A game-theoretic foundation. In *Second Workshop on the Algorithmic Foundations of Robotics*, Toulouse, France, July 1996.

[108] A. Lazanas and J.-C. Latombe. Landmark-based robot navigation. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, July 1992.

[109] J. Liebman, L. Lasdon, L. Schrage, and A. Waren. *Modeling and Optimization with GINO.* The Scientific Press, Palo Alto, CA, 1986.

[110] W. Liu and P. Will. Parts manipulation on an intelligent motion surface. In *IEEE/RSJ International Conference on Robots and Systems*, pages 3:399–404, Pittsburgh, PA, Aug. 1995.

[111] T. Lozano-Perez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1), Spring 1984.

[112] K. M. Lynch. The mechanics of fine manipulation by pushing. In *IEEE International Conference on Robotics and Automation*, pages 2269–2276, Nice, France, May 1992.

[113] K. M. Lynch. *Nonprehensile Robotic Manipulation: Controllability and Planning.* PhD thesis, The Robotics Institute, Carnegie Mellon University, Mar. 1996. Robotics Institute Technical Report CMU-RI-TR-96-05.

[114] K. M. Lynch and M. T. Mason. Controllability of pushing. In *IEEE International Conference on Robotics and Automation*, pages 112–119, May 1995.

[115] K. M. Lynch and M. T. Mason. Pulling by pushing, slip with infinite friction, and perfectly rough surfaces. *International Journal of Robotics Research*, 14(2):174–183, Apr. 1995.

[116] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson, editors, *Workshop on the Algorithmic Foundations of Robotics*, Wellesley, Massachusetts, 1995. A. K. Peters.

[117] K. M. Lynch and M. T. Mason. Dynamic underactuated nonprehensile manipulation. In *IEEE/RSJ International Conference on Robots and Systems*, Nov. 1996.

[118] M. Mani and W. R. D. Wilson. A programmable orienting system for flat parts. In *North American Manufacturing Research Institute Conference XIII*, Berkeley, California, May 1985.

[119] M. T. Mason. Compliant motion. In M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, editors, *Robot Motion.* MIT Press, 1982.

[120] M. T. Mason. *Manipulator Grasping and Pushing Operations.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1982. Also Artificial Intelligence Lab TR-690.

[121] M. T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986.

[122] M. T. Mason. Compliant sliding of a block along a wall. In *First International symposium on Experimental robotics*, June 1989.

[123] M. T. Mason. Two graphical methods for planar contact problems. In *IEEE/RSJ International Conference on Robots and Systems*, pages 443–448, Osaka, Japan, Nov. 1991.

[124] M. T. Mason and K. M. Lynch. Dynamic manipulation. In *IEEE/RSJ International Conference on Robots and Systems*, pages 152–159, Yokohama, Japan, July 1993.

[125] P. S. Maybeck. The Kalman Filter: An introduction to concepts. In *Stochastic Models, Estimation and Control.* Academic Press, 1979.

[126] T. McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9(2):62–82, 1990.

[127] J. D. Morrow, B. J. Nelson, and P. K. Khosla. Vision and force driven sensorimotor primitives for robotic assembly skills. In *IEEE/RSJ International Conference on Robots and Systems*, Pittsburgh, PA, Aug. 1995.

[128] A. Mottaez and K. Y. Goldberg. Positioning polygonal parts without sensors. In *SPIE Conference on Sensors and Controls for Automated Manufacturing Systems*, Boston, MA, Sept. 1993.

[129] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[130] K. Nagata. Manipulation by a parallel-jaw gripper having a turntable at each fingertip. In *IEEE International Conference on Robotics and Automation*, pages 1663–1670, May 1994.

[131] B. K. Natarajan. Some paradigms for the automated design of parts feeders. *International Journal of Robotics Research*, 8(6), Dec. 1989.

[132] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, New York, 1988.

[133] A. G. Neumann. The new ASME Y14.5M standard on dimensioning and tolerancing. *Manufacturing Review*, 7(1):9–15, Mar. 1994.

[134] J. L. Nevins and D. E. Whitney. Computer-controlled assembly. *Scientific American*, 238(2), Feb. 1978.

[135] J. Nievergelt and F. Preparata. Plane-sweep algorithmss for intersecting geometric figures. *Communications of the ACM*, 25(10):739–747, Oct. 1982.

[136] H. Nijmeijer and A. J. van der Schaft. *Nonlinear Dynamical Control Systems.* Springer-Verlag, 1990.

[137] N. J. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *International Joint Conference on Artificial Intelligence*, pages 509–520, Washington, D.C., May 1969.

[138] Y. Okawa and K. Yokoyama. Control of a mobile robot for the push-a-box operation. In *IEEE International Conference on Robotics and Automation*, pages 761–766, Nice, France, May 1992.

[139] G. Oriolo and Y. Nakamura. Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators. In *Proceedings of the 30th Conference on Decision and Control*, pages 2398–2403, 1991.

[140] N. Papanikolopoulos, P. K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9(1), Feb. 1993.

[141] M. A. Peshkin and A. C. Sanderson. The motion of a pushed sliding workpiece. *IEEE Journal of Robotics and Automation*, 4(6):569–598, Dec. 1988.

[142] M. A. Peshkin and A. C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4(5):524–531, Oct. 1988.

[143] D. T. Pham, K. C. Cheung, and S. H. Yeo. Initial motion of a rectangular object being pushed or pulled. In *IEEE International Conference on Robotics and Automation*, pages 1046–1050, Cincinnati, OH, May 1990.

[144] K. Pingle, R. Paul, and R. Bolles. Programmable assembly, three short examples. Film, Stanford Artificial Intelligence Laboratory, Oct. 1974.

[145] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[146] M. H. Raibert. *Legged Robots That Balance.* Cambridge: MIT Press, 1986.

[147] A. S. Rao and K. Y. Goldberg. Orienting generalized polygons. In *1992 IEEE International Conference on Robotics and Automation*, 1992.

[148] A. S. Rao and K. Y. Goldberg. On the relation between friction and part shape in parallel-jaw grasping. In *1993 IEEE International Conf. on Robotics and Automation.*, 1993.

[149] A. S. Rao and K. Y. Goldberg. Placing registration marks. In *1993 IEEE International Conference on Robotics and Automation*, 1993.

[150] A. S. Rao and K. Y. Goldberg. Shape from diameter: Recognizing polygonal parts with a parallel-jaw gripper. *International Journal of Robotics Research*, 13(1):16–37, Feb. 1994.

[151] A. S. Rao, D. Kriegman, and K. Y. Goldberg. Complete algorithms for reorienting polyhedral parts using a pivoting gripper. In *1995 IEEE International Conference on Robotics and Automation*, pages 2242–2248, May 1995.

[152] A. A. G. Requicha. Toward a theory of geometric tolerancing. *International Journal of Robotics Research*, 2(4):45–60, 1983.

[153] A. A. G. Requicha. Representation of tolerances in solid modeling: Issues and alternative approaches. In M. S. Pickett and J. W. Boyse, editors, *Solid Modeling by Computers: From Theory to Applications*, pages 3–22. Plenum Press, New York, 1984.

[154] A. A. G. Requicha. Mathematical definition of tolerance specifications. *Manufacturing Review*, 6(4):269–274, Dec. 1993.

[155] E. Rich and K. Knight. *Artificial Intelligence.* McGraw-Hill, 1991.

[156] F. J. Riley. *Assembly Automation: A Management Handbook.* Industrial Press, New York, 1983.

[157] A. A. Rizzi and D. E. Koditschek. Further progress in robot juggling: The spatial two-juggle. In *IEEE International Conference on Robotics and Automation*, pages 3:919–924, Atlanta, Georgia, May 1993.

[158] K. Salisbury. Whole arm manipulation. In R. C. Bolles and B. Roth, editors, *Robotics Research: The Fourth International Symposium*, pages 183–190, Cambridge, MA, Aug. 1988. MIT Press.

[159] A. C. Sanderson. Parts entropy methods for robotics assembly system design. In *1984 IEEE International Conference on Robotics and Automation*, 1984.

[160] L. A. Santaló. *Integral Geometry and Geometric Probability.* Encyclopedia of Mathematics and its Applications, Vol. 1. Addison-Wesley, Reading, MA, 1976.

[161] N. Sawasaki, M. Inaba, and H. Inoue. Tumbling objects using a multi-fingered robot. In *Proceedings of the 20th International Symposium on Industrial Robots and Robot Exhibition*, pages 609–616, Tokyo, Japan, 1989.

[162] L. E. Schrage. *User's manual for LINDO.* The Scientific Press, Palo Alto, CA, 1981.

[163] S. Shekhar and J.-C. Latombe. On goal recognizability in motion planning with uncertainty. In *IEEE International Conference on Robotics and Automation*, pages 1728–1733, 1991.

[164] M. Shirai and A. Saito. Parts supply in Sony's general-purpose assembly system "SMART". *Jpn. J. Adv. Automation Tech.*, 1:108–111, 1989.

[165] D. M. Siegel. Finding the pose of an object in a hand. In *1991 IEEE International Conference on Robotics and Automation*, 1991.

[166] R. Simmons. Determining sensing resolution and frequency. In *1992 AAAI Spring Symposium on Control of Selective Perception*, 1992.

[167] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1080–1087. Morgan Kaufmann, 1995.

[168] S. Simunovic. Force information in assembly processes. In *Proceedings, 5th International Symposium on Industrial Robotics*, pages 415–431, Chicago, IL, Sept. 1975.

[169] N. C. Singer and W. P. Seering. Utilizing dynamic stability to orient parts. *Journal of Applied Mechanics*, 54:961–966, Dec. 1987.

[170] S. S. Skiena. Problems in geometric probing. *Algorithmica*, 4, 1989.

[171] O. Sørdalen, Y. Nakamura, and W. Chung. Design of a nonholonomic manipulator. In *IEEE International Conference on Robotics and Automation*, pages 8–13, 1994.

[172] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical Report CMU-RI-TR-93-20, The Robotics Institute, Carnegie Mellon University, 1993.

[173] H. J. Sussmann. Lie brackets, real analyticity and geometric control. In R. W. Brockett, R. S. Millman, and H. J. Sussmann, editors, *Differential Geometric Control Theory*. Birkhauser, 1983.

[174] P. J. Swanson, R. R. Burridge, and D. E. Koditschek. Global asymptotic stability of a passive juggler: A parts feeding strategy. In *IEEE International Conference on Robotics and Automation*, pages 1983–1988, Nagoya, Japan, 1995.

[175] R. H. Taylor. *A Synthesis of Manipulator Control Programs from Task-Level Specifications.* PhD thesis, Artificial Intelligence Laboratory, Stanford University, 1976.

[176] R. H. Taylor, M. T. Mason, and K. Y. Goldberg. Sensor-based manipulation planning as a game with nature. In R. C. Bolles and B. Roth, editors, *Robotics Research: The Fourth International Symposium*, pages 421–429, Cambridge, MA, 1988. MIT Press.

[177] H. Terasaki and T. Hasegawa. Motion planning for intelligent manipulations by sliding and rotating operations with parallel two-fingered grippers. In *IEEE/RSJ International Conference on Robots and Systems*, pages 119–126, Munich, Germany, Sept. 1994.

[178] J. C. Trinkle, J. M. Abel, and R. P. Paul. An investigation of frictionless enveloping grasping in the plane. *International Journal of Robotics Research*, 7(3):33–51, June 1988.

[179] J. C. Trinkle, A. O. Farahat, and P. F. Stiller. First-order stability cells of active multi-rigid-body systems. *IEEE Transactions on Robotics and Automation*, 11(4):545–557, Aug. 1995.

[180] F. A. Valentine. *Convex Sets.* McGraw-Hill, New York, 1964.

[181] H. Voelcker. A current perspective on tolerancing and metrology. *Manufacturing Review*, 6(4):258–268, Dec. 1993.

[182] R. K. Walker and V. Srinivasan. Creation and evolution of the ASME Y14.5.1M standard. *Manufacturing Review*, 7(1):16–23, Mar. 1994.

[183] A. S. Wallack, J. F. Canny, and D. Manocha. Object localization using crossbeam sensing. In *IEEE 1993 International Conference on Robotics and Automation*, 1993.

[184] E. Wefald and S. J. Russell. *Do the right thing: studies in limited rationality.* MIT Press, 1991.

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

[186] D. E. Whitney. Remote Center Compliance. In R. C. Dorf, editor, *International Encyclopedia of Robotics: Applications and Automation, Vol. 3*, pages 1316–1324. John Wiley and Sons, New York, 1988.

[187] J. Wiegley, K. Y. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. In *IEEE International Conference on Robotics and Automation*, pages 1133–1139, Apr. 1996.

[188] C. K. Yap. Exact computational geometry and tolerancing metrology. In *Snapshots of Computational and Discrete Geometry, Volume 3*, Sept. 1995. McGill School of Computer Science Tech. Report No. SOCS-94.50.

[189] Y. Zhuang, Y.-C. Wong, and K. Y. Goldberg. On the existence of modular fixtures. In *IEEE International Conference on Robotics and Automation*, pages 543–549, San Diego, CA, May 1994.

[190] N. B. Zumel and M. A. Erdmann. Balancing of a planar bouncing object. In *IEEE International Conference on Robotics and Automation*, pages 2949–2954, San Diego, CA, 1994.

[191] N. B. Zumel and M. A. Erdmann. Nonprehensile two palm manipulation with non-equilibrium transitions between stable states. In *IEEE International Conference on Robotics and Automation*, pages 3317–3323, Minneapolis, MN, Apr. 1996.