Posing Polygonal Objects in the Plane by Pushing

Srinivas Akella
Matthew T. Mason
The Robotics Institute and School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 USA

Abstract

This paper studies the use of pushing actions to orient and translate objects in the plane. The authors 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. The authors show that a set of linear normal pushes can always be found to move any polygonal object from any initial configuration to any goal configuration in the obstacle-free plane. The object configuration is specified by its pose; that is, its position and orientation. The authors formulate the search for such a sequence of pushes as a linear programming problem. They then describe an implemented 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. This planner is proven to be complete and to have polynomial time complexity. The planner, which uses an analysis of the mechanics of pushing an object, generates open-loop plans that do not require sensing. The authors describe experiments that demonstrate the validity of the generated plans.

1. Introduction

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.

The work reported in this paper is concerned with the automatic synthesis of sequences of linear pushing actions to position and orient an object, 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 1986; Brost 1988; Peshkin and Sanderson 1988b). Sequences of such actions have been used primarily to orient parts (Mani and Wilson 1985; Peshkin and Sanderson 1988a). Here, we develop a method to find plans to move a polygonal object 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 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 have proven the planner complete; it is guaranteed to generate open-loop plans to move an arbitrary polygonal object 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. Henceforth, we refer to the combined position and orientation of the object as the *pose* of the object. See Figure 1 for an example plan.

This paper focuses on three issues:

Authors' Note. Srinivas Akella is currently at the Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

The International Journal of Robotics Research,
Vol. 17, No. 1, January 1998, pp. 70–88,

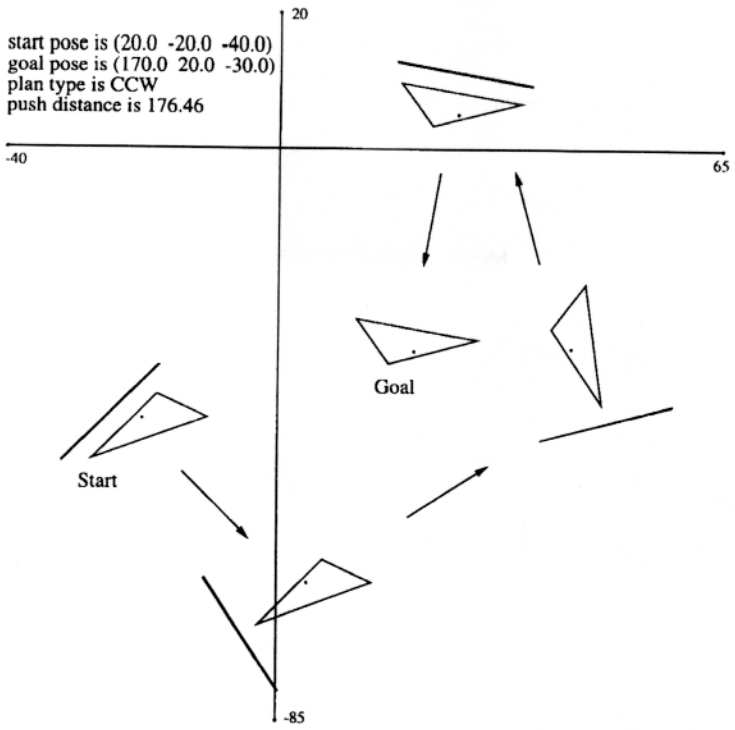


Fig. 1. A plan generated by the pose planner to move the triangle. The arrows indicate push directions.

1. Generating sequences of linear normal pushes for the problem of simultaneously orienting and positioning polygonal objects.
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 that 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.

The paper is organized as follows. Section 2 discusses related work, primarily in the areas of pushing and non-prehensile manipulation. Section 3 states the problem and outlines the solution. Section 4 discusses the relevant results on the mechanics of pushing an object. Section 5 describes the selection of a feasible set of pushes using a linear programming formulation. Section 6 outlines the pose planner, and Section 7 describes experimental validation of the generated plans. The final section concludes with a summary of our results and suggestions for future work.

2. Related Work

The use of task mechanics is a powerful solution technique in robotic manipulation tasks; we list a few early examples. Inoue (1974) used force information to achieve peg insertion despite inaccuracies in manipulator positioning that exceeded assembly tolerances. Simunovic (1975) analyzed the contact forces during the insertion of a peg in a hole and identified conditions under which jamming occurs. Whitney (1982) and colleagues studied assembly operations and developed the remote center compliance to overcome uncertainties during assembly insertions. Erdmann and Mason (1988) analyzed the mechanics of the sliding motion of a planar object in a tiltable tray and developed a planner to find a sensorless sequence of tilts to orient the object. Our work in this paper builds on previous work on the mechanics and planning of pushing operations, which we discuss next.

2.1. *Pushing*

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 (1982, 1986) 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 (1985) used the pushing rules of Mason (1982) 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 (1988) 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 (1982), he developed the push-stability diagram to describe the possible motions of an object being pushed by a fence. Peshkin and Sanderson (1988b) 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 (1988a) 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 (1993) designed curved fence sections to eliminate uncertainty in the orientations of parts being oriented by the fence. Pham, Cheung, and Yeo (1990) analyzed the initial motion of a rectangular object being pushed or pulled and its dependence on the magnitude of the exerted force. Goyal, Ruina, and Papadopoulos (1991a, 1991b) 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 (1993) derived bounds on the center of rotation locus over all possible support friction distributions for objects with known support regions. Balorda and Bajd (1994) used a two-finger tool to reduce the positional uncertainty of an object by pushing it. They discussed the effect of finger configurations on accurate positioning of the object. Goldberg and Mason (1990) described a Bayesian framework for planning multistep grasps with a frictionless parallel-jaw gripper to orient objects. In subsequent work, Goldberg (1993) 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 (1988) model for parallel-jaw grasping by using only pushes normal to the face of the gripper. In the work reported in this paper, we use linear normal pushes to orient and translate the object. This work was reported earlier in Akella and Mason (1992).

Work by Lynch and Mason (Lynch 1992; Lynch and Mason 1995b) is most closely related to our work. Lynch (1992) 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 (1995b) discussed issues of controllability and planning in the use of stable pushes. They described a pushing path planner, based on the nonholonomic motion planner of Barraquand and Latombe (1993), 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, whereas 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, whereas 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; the linear programming formulation of our planner provides a computationally efficient method to find exact solutions. Brost (1992) presented 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 (1994) used 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 (1996) included 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 determined the initial pose of the object.

Work on pushing by mobile robots dates back to work by Nilsson (1969). Okawa and Yokoyama (1992) described 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 (1995) used a team of mobile robots to reorient furniture by pushing and analyzed the information requirements of the task. They presented several algorithms that differ in the amounts of global control, communication, and synchronization. Brown and Jennings (1995) presented 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.

2.2. *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 cited above can be viewed as examples of nonprehensile manipulation. Salisbury (1988) discussed 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 (1988)

presented a planner that uses both the palm surface and fingers of a frictionless gripper to achieve an enveloping grasp of an object. Brock (1988) 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 (1992) explicitly considered sliding of robot fingers on an object and planned finger motions so the object follows a desired trajectory. Sawasaki, Inaba, and Inoue (1989) used stand-up and tumbling operations with fingers to manipulate objects in contact with a support surface. Aiyama, Inaba, and Inoue (1993) used pivoting operations, where they manipulate an object supported on a surface by rotating the object about its vertices. Nagata (1994) 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 (1994) described a manipulation system that slides and rotates an object to enable subsequent grasping. Farahat, Stiller, and Trinkle (1995) analytically determined the position and orientation of a polygon moving in sliding and rolling contact with two or three position-controlled robots. Trinkle, Farahat, and Stiller (1995) analyzed systems of multiple objects and manipulators in contact and determined 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 (1995) analyzed the use of two frictionless fingers to stably support and rotate a polygonal object in a gravitational field. Akella et al. (1996) described a planar manipulation system consisting of a one-joint actuator positioned over a conveyor belt. They showed 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 (1996) developed a two-palm manipulation system that uses a sequence of nonprehensile operations such as sliding to rotate an object. Zumel and Erdmann (1996) used two frictionless palms to manipulate an object in a gravitational field and presented 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. (1994) used arrays of microelectromechanical actuators to perform sensorless manipulation of parts. By controlling the motions of the actuators in the array, they generated vector fields that produced desired part motions. Liu and Will (1995) simulated a parts manipulation surface consisting of microelectromechanical actuators and described various parts manipulation strategies. Böhringer, Bhatt, and Goldberg (1995) used 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 (1995) analyzed the motion of a part subjected to a vibratory juggling motion and indicated 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 (1989) developed a robot system that juggles a tennis ball in three dimensions and uses learning to improve its performance. Buehler, Koditschek, and Kindlmann (1994) described "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 (1993) developed a paddle juggler that performs spatial juggling of two balls. Zumel and Erdmann (1994) analyzed the juggling of a polygon and identified conditions under which the polygon can be brought to a stable trajectory. Mason and Lynch (1993) presented a taxonomy of manipulation models and analyzed 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 (1994) rotated a cube on a paddle and controlled its rotation rate by accelerating the paddle using a Puma robot. Lynch (1996) explored dynamic nonprehensile manipulation and demonstrated control of an object using a one-degree-of-freedom robot by sequencing phases of dynamic grasp, rolling, and free flight. Higuchi (1985) used impulsive forces generated by electromagnetic means to microposition objects. Huang, Krotkov, and Mason (1995) explored the use of striking to move a rotationally symmetric object to a desired position and orientation in the plane.

2.3. *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 problem described in this paper is that for the selected class of linear normal pushing 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 (1995). He gives examples of other solution-complete problems such as sensorless orienting of parts (Goldberg 1993) and controllability and motion planning for nonholonomic mobile robots in the obstacle-free plane (Barraquand and Latombe 1993). He describes a modular fixturing problem for polygonal parts (Zhuang, Wong, and Goldberg 1994) 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 1994). Characterizing problems in this manner and developing complete algorithms to solve them enables us to identify and guarantee capabilities of our robot systems.

3. The Planar Pose Problem

Given a polygonal object on a horizontal table at a known start position and orientation (the start pose), we are to find a pushing plan to move it to a specified goal position and orientation (the 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 object and the position of the center of mass of the object. The start and goal poses may be arbitrary.

3.1. Outline of the Approach

To make the analysis tractable, we partially decouple the problems of orienting and positioning the object. We first consider the required change in orientation and select a sequence of intermediate orientations, achieved by using *reorient pushes* to rotate the object. The intermediate orientations depend on whether the rotation is clockwise or counterclockwise and are selected based on an analysis of the mechanics of pushing an object with a fence. The reorient pushes do not usually combine to move the object to the goal position, so we need a set of *translation pushes* to translate the object without changing its orientation. These pushes can occur at the start, intermediate, and goal orientations of the object. The reorient and translation pushes produce a net translation of the object; 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.

We make the following assumptions for our analysis:

1. The object geometry and the location of its center of mass (COM) are known. The center of mass is in the interior of the object.
2. All objects 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 object does not contact the ends of the fence during pushes, and the object 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 object and the fence. Lynch and Mason (1995a) 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.
9. Support friction is uniform over the plane, and all support points lie inside the convex hull of the object.

4. The Mechanics of Pushing

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

4.1. Types of Pushes

The pusher motion and the contact conditions between the object and the pusher influence the motion of the pushed object. 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 object, the distance the fence moves in contact with the object, 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 object, 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 object motions relative to the pusher:

1. *No relative motion between the object and the pusher:* Here, there is no slip or rolling at the contacts between the pusher and the object, so the object has the same motion as the pusher. This class of motions occurs during the translation pushes defined in Section 4.2 and the stable rotational pushes of Lynch (1992).
2. *Known relative motion between the object and pusher:* When the pressure distribution is known, the object motion can be predicted. The pressure distribution is usually unknown, however, and the instantaneous relative motion between the object 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 4.2.

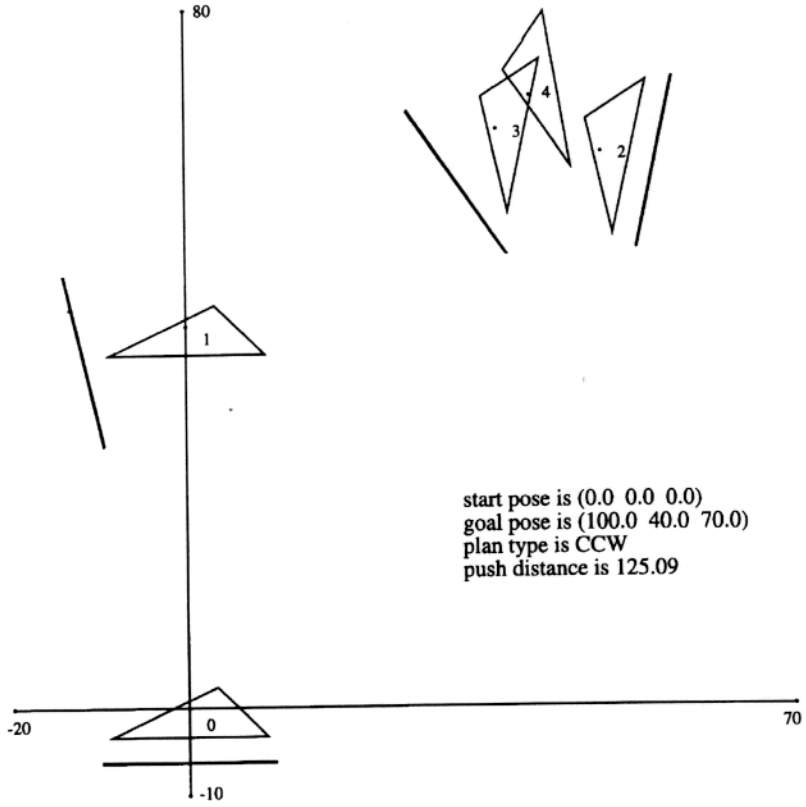


Fig. 2. An example pose plan for a triangle. The poses are numbered in sequence; the zeroth and fourth poses are the start and goal poses, respectively. The first and third poses are achieved by translation pushes, whereas the second and fourth poses are achieved by reorient pushes. The bold lines represent the fence. A pose is indicated by (orientation, COM-x, COM-y).

3. *Unknown relative motion between the object and pusher:* This occurs when the support pressure distribution is unknown and there are insufficient constraints on the motion of the object. The incomplete pushes described in Section 4.2 belong to this category.

4.2. 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 object's orientation, the orientation of the fence and the length of the push are then sufficient to describe a push. When a fence lined up parallel to an object edge executes a push, the object 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 an object can be classified as follows (Fig. 3):

1. *Translation push:* The fence and object edge are always parallel, and there is no change in the orientation of the object as the fence translates.
2. *Reorient push:* The push begins with an initial angle between the fence and object edge, and the object rotates so the edge is aligned with the fence by the end of the push. This push reorients the object to a known orientation. We refer to the minimum push distance guaranteed to align the object 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 (1988b) in Appendix A.
3. *Incomplete push:* The push begins with an initial angle between the fence and object edge, but unlike the reorient push, the object 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 object is not known.

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

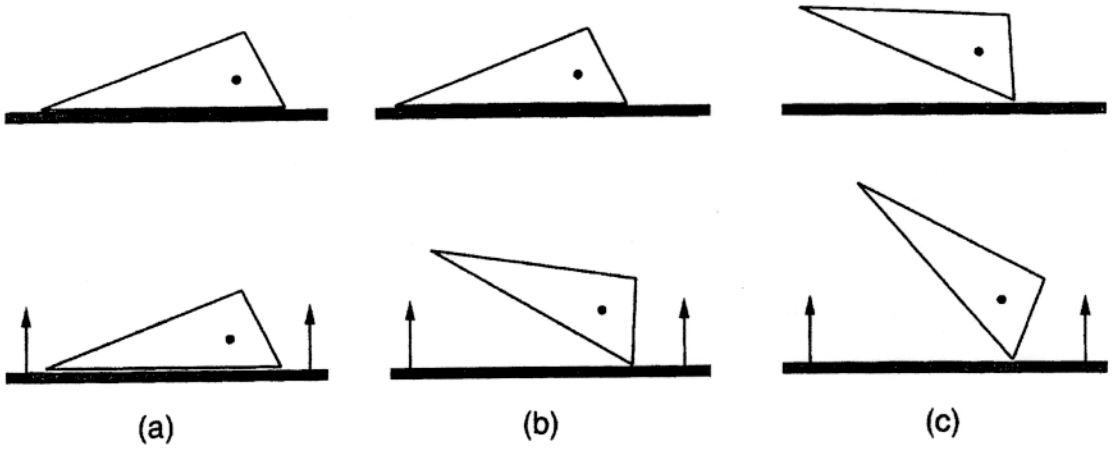


Fig. 3. Linear normal pushes. (a) Translation push. (b) Reorient push. (c) Incomplete push. The center of mass is indicated by the dot.

4.3. Object Classification

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

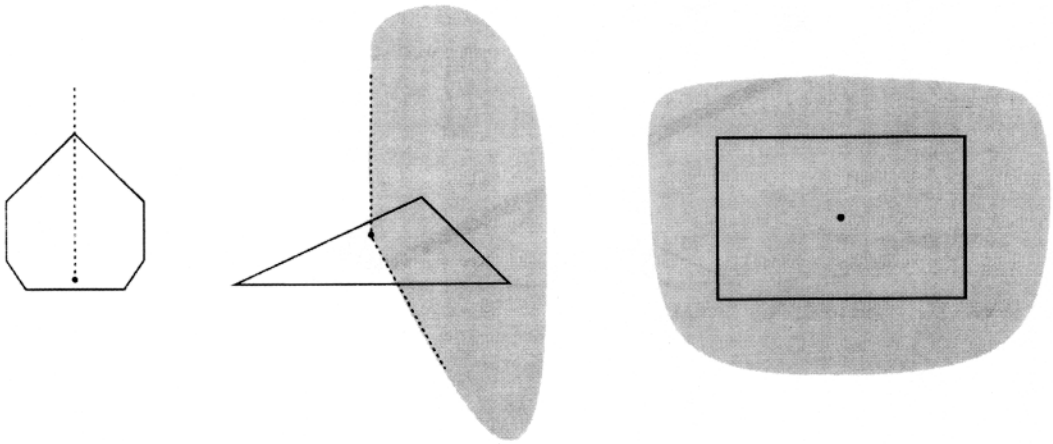
The *translation image* of an object at a given pose is the region of positions to which the object can be translated without a change in orientation. To reach the goal pose, the object should attain the goal orientation at a position whose translation image includes the goal position. The translation 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 objects subject to linear normal pushes:

1. *Unistables:* These are objects 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 objects 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 objects 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 objects is the entire plane.

The above classification is illustrated in Figure 4. Since only the spanning-stables have a translation image that spans the plane, unistables and biplus-stables cannot always be translated between any two positions without a change in orientation. Figure 5 illustrates this for a biplus-stable.

4.4. Stable Edges and the Angle-Eating Heuristic

To perform translation and reorient pushes on an object, we need to determine the stable edges of the object. 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 (1993) in obtaining this information from the *radius function* of the object (Fig. 6). The radius of a polygon is the perpendicular distance from a reference point in the polygon to a supporting line. The radius function $r : \mathcal{S}^1 \rightarrow \mathcal{R}^1$ is a plot of the radius as the supporting line is rotated. When the center of mass is the reference point and the fence is the supporting line, the radius function indicates the stable edges of the object 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 (1988) 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 (Fig. 6). To avoid introducing uncertainty in object 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.



(a) (b) (c)

Fig. 4. Examples of the three object classes with their translation images shown shaded. (a) Unistable. (b) Biplus-stable. (c) Spanning-stable.

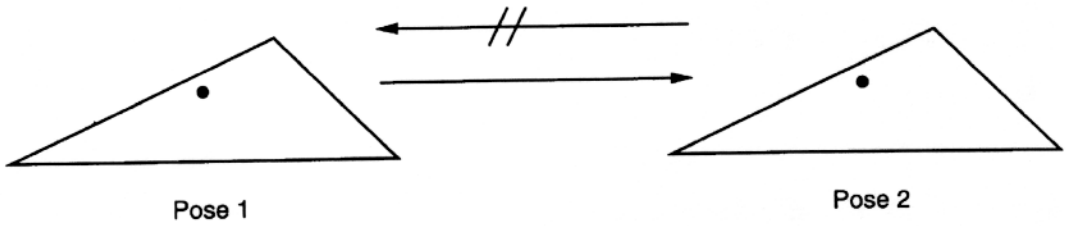


Fig. 5. The triangle of Figure 4 can be translated from pose 1 to pose 2, since pose 2 lies in the translation image of pose 1. However, the triangle cannot be translated from pose 2 to pose 1.

The *angle-eating heuristic*, used to generate reorient sequences, seeks to minimize the number of pushes. The reorientation strategy requires that the object 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 object 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 7 for sample reorient sequences.

5. Determining the Pushes

A pushing plan to pose an object 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 object reaches the goal position in the goal orientation.

5.1. *The Pose Vector Equation*

To determine the magnitudes of the pushes, we treat the pushes as vectors in the plane. At a given orientation of the object, translation pushes can occur only in the directions of the inward unit normal vector to 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

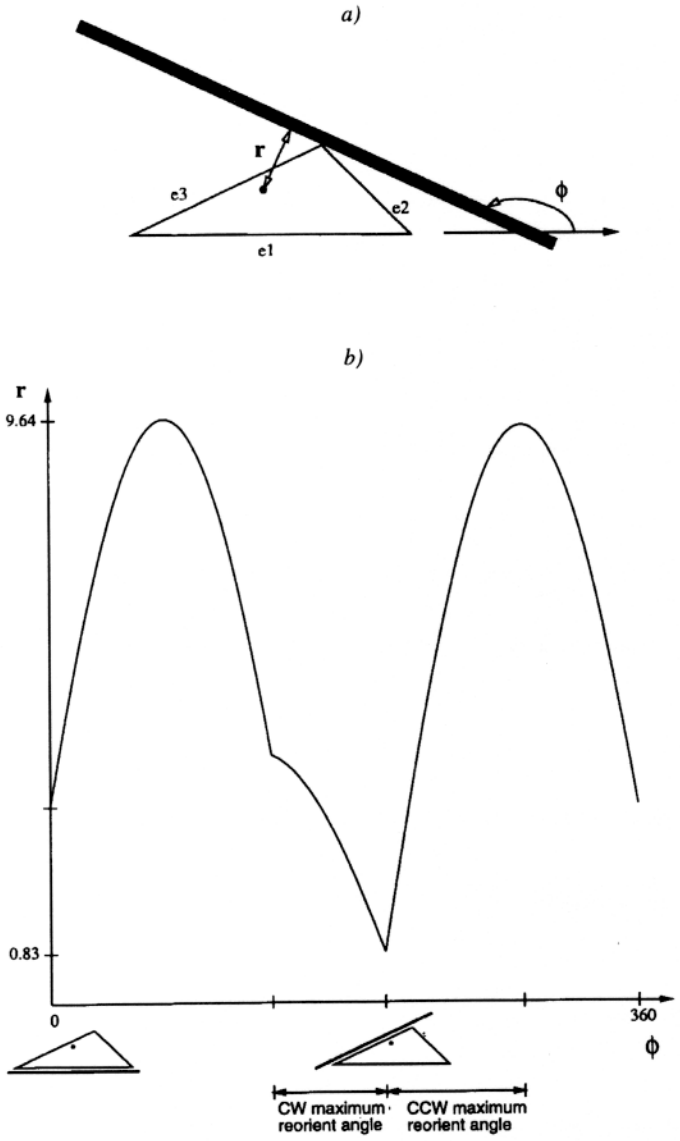


Fig. 6. The radius function for the triangle, based on Goldberg (1993). (a) The radius r of an object at a fence orientation ϕ is the perpendicular distance from the center of mass to the fence. (b) The radius function is the plot of the object radius as the fence orientation is varied. The orientation wraps around at 360°. The local minima of the radius function occur at stable edges of the object. Kinks in the radius function occur at unstable edges of the object and are nonminima of the radius function with a discontinuity in slope. This triangle is a biplus-stable because its radius function has only two minima. The clockwise (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 counterclockwise (CCW) maximum reorient angle is measured similarly in the rightward direction. The CW and CCW maximum reorient angles for edge e3 are shown.

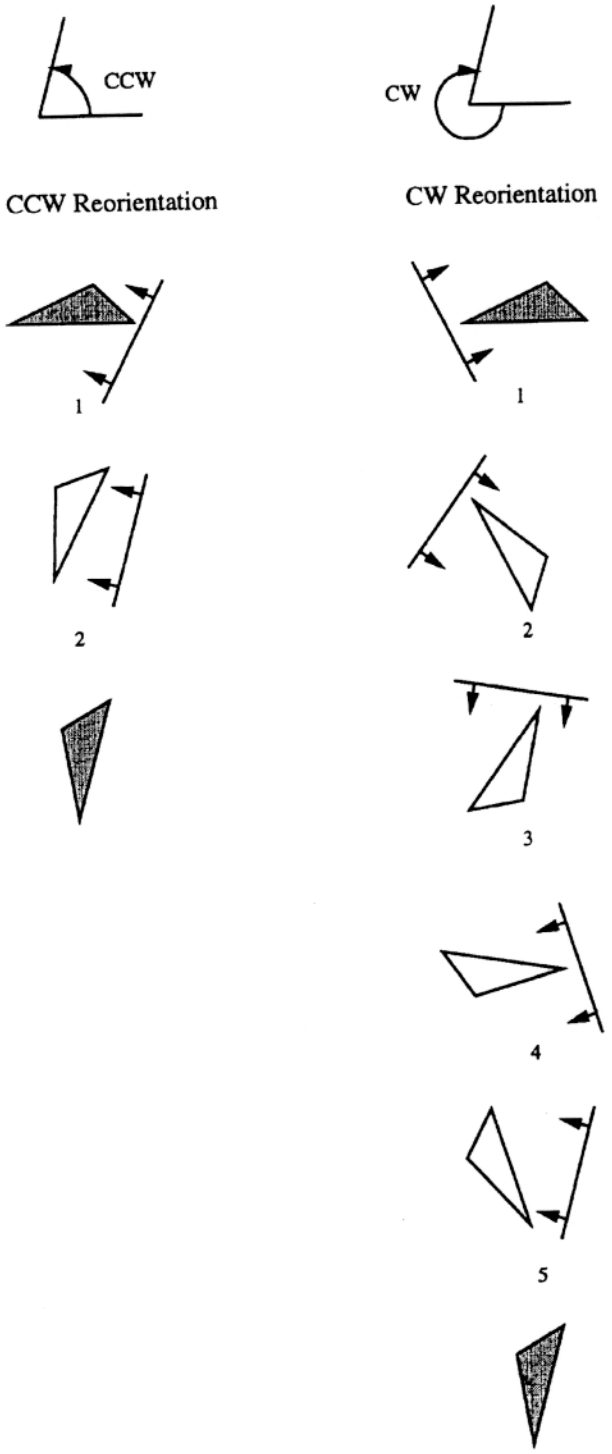


Fig. 7. Counterclockwise (CCW) and clockwise (CW) reorient push sequences to rotate a triangle. The longest edge of the triangle is the stable edge chosen for the reorient pushes. The start and goal orientations are shown shaded. The reorient pushes for each rotation direction are numbered in sequence. At each orientation, the arrows on the fence indicate the push direction to rotate the object to the next orientation.

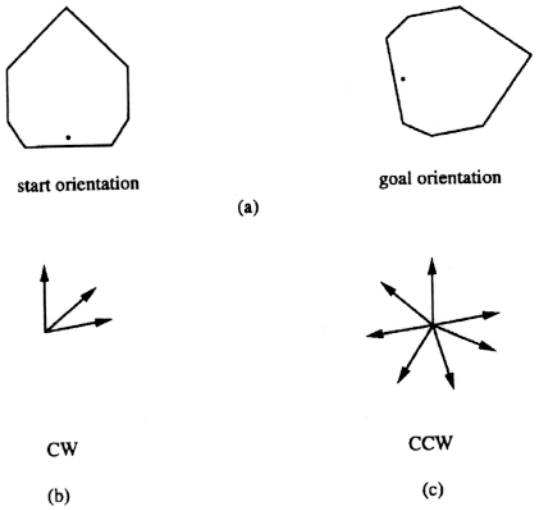


Fig. 8. Push vector diagrams for a unistable. The edge closest to the center of mass is the stable edge. (a) The start and goal orientations. (b) Clockwise (CW) push vector diagram. (c) Counterclockwise (CCW) push vector diagram.

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 8 for examples.

We have assumed that the object does not slip relative to the fence. So during a reorient push, the object rotates about its fixed contact point with the fence as the fence translates. As the object 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 object.

A solution to this equation is to be found that 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).

5.2. 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 (1983) 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

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

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

$$\text{Minimize} \sum_{i=0}^{r} \sum_{j \in S} a_{ij} + \sum_{i=1}^{r} b_i$$

subject to

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

$$\sum_{i=0}^{r} \sum_{j \in S} a_{ij} (\hat{\mathbf{n}}_{ij} \cdot \hat{\mathbf{t}}'_{\perp}) + \sum_{i=1}^{r} b_i (\hat{\mathbf{p}}_i \cdot \hat{\mathbf{t}}'_{\perp}) = 0$$

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

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

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 Appendix B for the MIP formulation that minimizes the plan execution time and an example plan.

5.3. Existence of Solutions

An object 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 object from an arbitrary start pose to an arbitrary goal pose always exist?

The pose vector equation can be transformed to

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

where $b_i = m_i + e_i$ such that $e_i \geq 0$ for $i = 1, \ldots, r$ 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 are obtained from the following theorem in Davis (1954): A set of vectors $\{\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_r\}$ positively spans $\mathcal{R}^n$ if and only if, for every nonzero 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 object rotation in the chosen rotation direction. There are two cases to consider:

1. *The reorientation is greater than or less than* 180°: Since an object 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 object 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 vector sets generated for the CW and CCW reorientations both positively span the plane.
- *Unistables:* For unistables, $\hat{\mathbf{n}}_{is} = \hat{\mathbf{p}}_i$ (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 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 exist. (Note that by rotating an object through an additional 360°, 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.

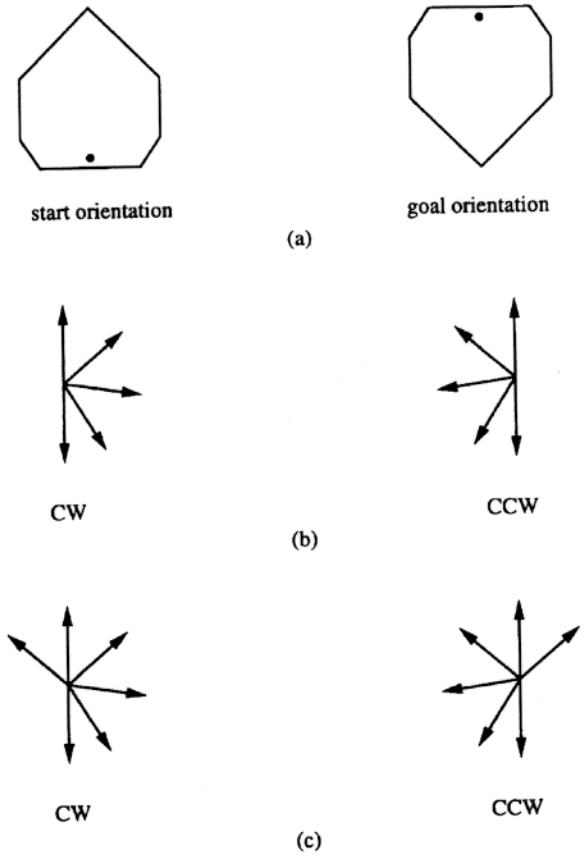


Fig. 9. A singularity case. (a) Start and goal orientations 180° apart. (b) Clockwise (CW) and counterclockwise (CCW) push vector diagrams: the push vectors do not positively span the plane. (c) CW and CCW push vector diagrams: the push vectors augmented by an additional reorient push in the opposite rotation direction positively span the plane.

6. The Pose Planner

The pose planner must generate a set of pushes that move an arbitrary object from an arbitrary start pose to an arbitrary goal pose. The planner attempts to generate plans involving CW rotation of the object; CCW rotation of the object; and, when the start and goal orientations are identical, no rotation of the object. 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 10.

When the required reorientation is 180° and the object is a unistable, sometimes neither the CW nor CCW plan is feasible. When such a singularity occurs, the planner uses the *CW* and *CCW singularity handlers* to generate 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, and similarly for the *CCW singularity plan*. Since these pushes positively span the plane, there is a guaranteed solution for each main

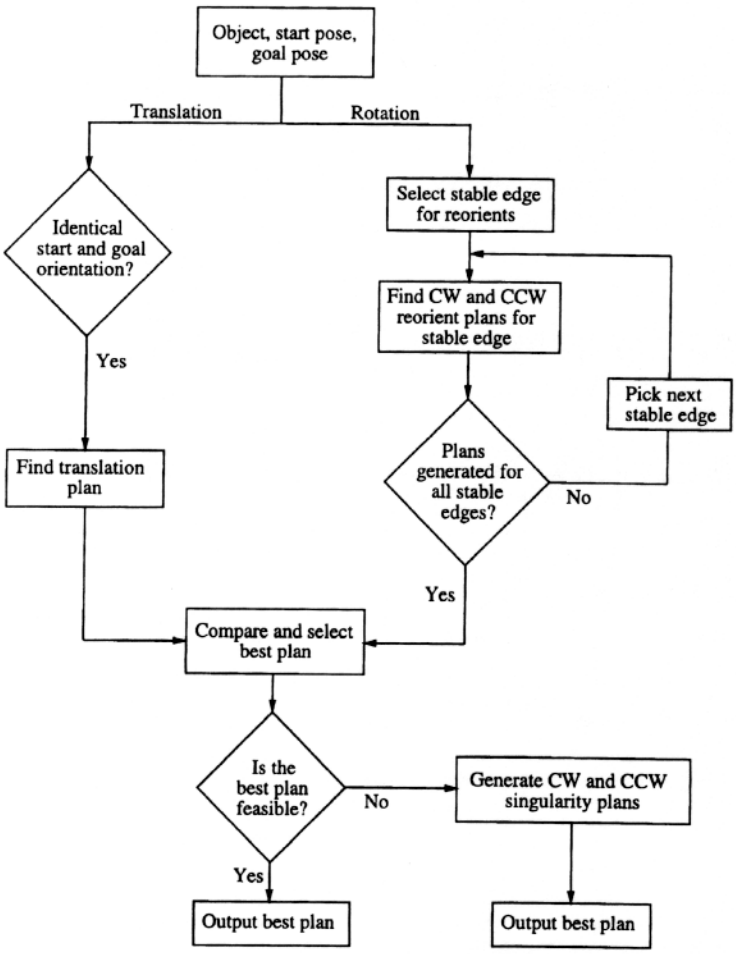


Fig. 10. A flowchart describing the organization of the pose planner. Given an object description and the start and goal poses, it outputs the best plan as defined by the objective function.

rotation direction and the better one is selected. An example singularity plan is shown in Figure 11.

6.1. *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 object. A proof of completeness is equivalent to a proof that for each object 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 object class it generates at least one set of pushes whose unit vectors positively span the plane. As shown in Section 5.3, 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 unistables, the singularity handler guarantees two sets of unit vectors that positively span the plane when required for a 180° reorientation. So, the pose planner is complete.

6.2. *Planner Complexity*

The planner has polynomial time complexity. For an object 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 $O(s^4r_{max}^3)$.

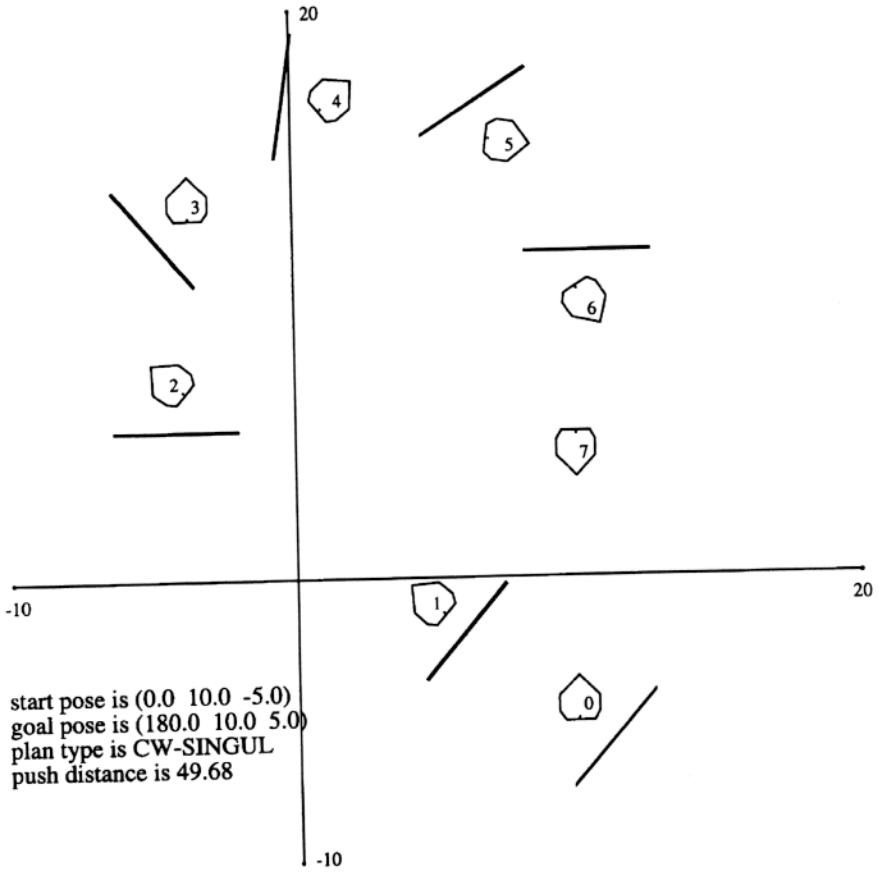


Fig. 11. A plan to handle a unistable singularity case.

7. 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 object, in terms of its vertices and center of mass. It uses the code to generate the push-stability diagram described by Brost (1988) and a commercial linear programming package, LINDO (Schrage 1981). The planner takes between 1 and 12 seconds on a SPARCstation IPX to find the best plan for the example objects in this paper.

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 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 object 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 object. 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 more than 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. Objects belonging to the three classes, including those in Figure 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 object to the goal orientation. However, position errors of 2 to 3 mm over a translation distance of 150 to 200 mm were observed. These position errors were sometimes as large as 5 mm. The observed errors were a result of slip during reorients, motion of the object along the fence due to robot vibration, and the positioning inaccuracy of the Puma.

8. Conclusion

In this paper, we described the use of linear normal pushes to position and orient objects in the plane. We introduced a classification of polygonal objects subject to linear normal

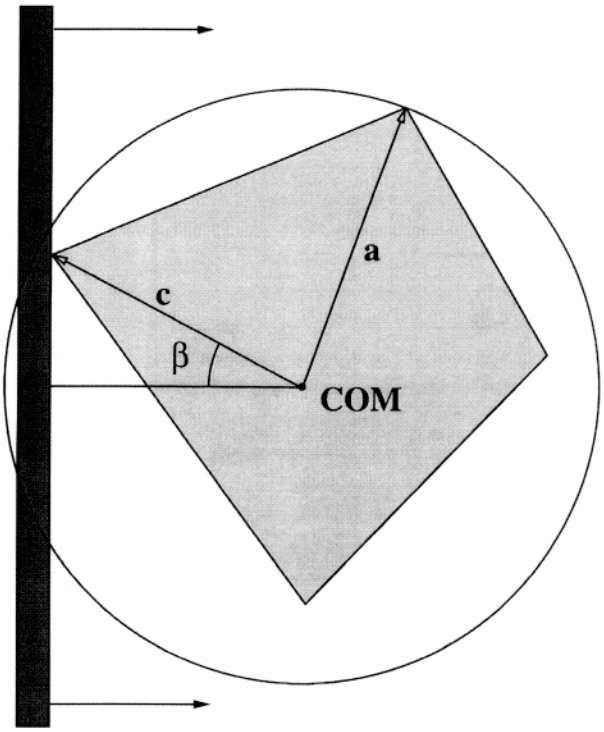


Fig. 12. Variables to calculate the Peshkin distance, based on Peshkin and Sanderson (1988b).

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 nonnormal linear pushes, or rotational pushes as in Lynch and Mason (1995b). 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, whereas one with a carlike mobile robot is conducive to the use of 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 object 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 an object. A combination of pushes and grasps with a parallel-jaw gripper, the use of a specially shaped fence (Brost 1992), or a sequence of centering operations (Mottaez and Goldberg 1993) are ways to reduce the pose uncertainty of the object. We have made the analysis tractable by separating object orientation from object 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 object 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 nonnormal 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 object shapes and to determine if multiple objects can be posed simultaneously.

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 (1988b). Since we assume that 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 Fig. 12).

Appendix B: Minimizing Plan Execution Time

We have also developed a linear mixed-integer programming (MIP) formulation to minimize any linear function of the push

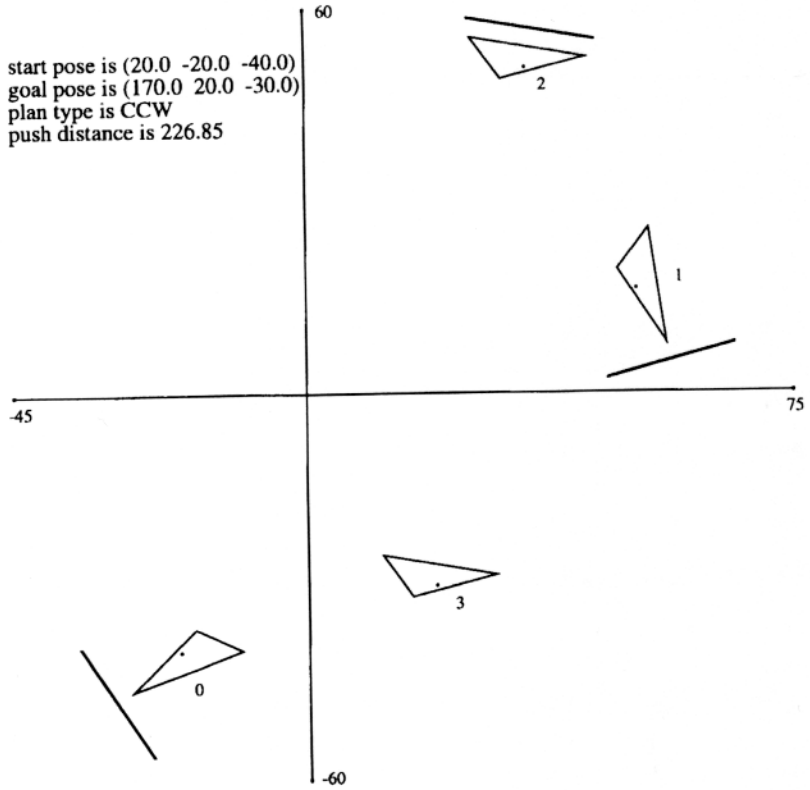


Fig. 13. A plan that minimizes the execution time. The push setup time k_p is 3 secs, and for a fence speed of 40 mm/sec, k_d is 0.025 secs/mm.

distance and the number of pushes, and have implemented this formulation in our planner. (See Nemhauser and Wolsey (1988) 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 nonzero translation pushes and the number of reorient pushes. So, the MIP formulation to minimize the execution time of a plan can be written as

$$\text{Minimize } k_d \left(\sum_{i=0}^{r} \sum_{j \in S} a_{ij} + \sum_{i=1}^{r} b_i \right) + k_p \left(\sum_{i=0}^{r} \sum_{j \in S} \alpha_{ij} + \right)$$

subject to

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

$$\sum_{i=0}^{r} \sum_{j \in S} a_{ij} (\hat{\mathbf{n}}_{ij} \cdot \hat{\mathbf{t}}'_{\perp}) + \sum_{i=1}^{r} b_i (\hat{\mathbf{p}}_i \cdot \hat{\mathbf{t}}'_{\perp}) = 0$$

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

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

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

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

$$\alpha_{ij} \in \{0, 1\} \quad \text{for } i = 0, \ldots, r \quad \text{and } j \in S,$$

where the α_{ij} terms are binary variables for the corresponding translation pushes and M is a large positive number. The resulting plans minimize the time taken to execute a plan. Figure 13 depicts a plan that minimizes the execution time for the same start and goal poses as in Figure 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.

Acknowledgments

Thanks to Randy Brost for generously sharing code and insights. Discussions with Alan Christiansen, Mike Erdmann,

Ken Goldberg, Kevin Lynch, N. R. Natraj, Fred Solomon, and Dafna Talmor helped shape the work. This work was supported by NSF Grant IRI-9114208 and The Robotics Institute, Carnegie Mellon University.

References

Abell, T., and Erdmann, M. 1995 (Pittsburgh, PA). Stably supported rotations of a planar polygon with two frictionless contacts. *Proc. IEEE/RSJ Int. Conf. on Robots and Systems*, Vol. 3, pp. 411–418.

Aboaf, E. W., Drucker, S. M., and Atkeson, C. G. 1989 (Scottsdale, AZ). Task-level robot learning: Juggling a tennis ball more accurately. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1290–1295.

Aiyama, Y., Inaba, M., and Inoue, H. 1993 (Yokohama, Japan). Pivoting: A new method of graspless manipulation of object by robot fingers. *Proc. IEEE/RSJ Int. Conf. on Robots and Systems*, pp. 136–143.

Akella, S., Huang, W. H., Lynch, K. M., and Mason, M. T. 1996 (Munich, Germany). Planar manipulation on a conveyor with a one joint robot. In G. Giralt and G. Hirzinger (eds.): *Robotics Research: The Seventh International Symposium*. London: Springer, pp. 265–276.

Akella, S., and Mason, M. T. 1992 (Nice, France). Posing polygonal objects in the plane by pushing. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2255–2262.

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

Arai, H., and Khatib, O. 1994 (Kobe, Japan). Experiments with dynamic skills. *Proc. 1994 Japan-USA Symp. on Flexible Automation*, pp. 81–84.

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

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

Böhringer, K.-F., Bhatt, V., and Goldberg, K. Y. 1995 (Nagoya, Japan). Sensorless manipulation using transverse vibrations of a plate. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1989–1996.

Böhringer, K.-F., Donald, B. R., Mihailovich, R., and MacDonald, N. C. 1994 (San Diego, CA). Sensorless manipulation using massively parallel microfabricated actuator arrays. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 826–833.

Brock, D. L. 1988 (Philadelphia, PA). Enhancing the dexterity of a robot hand using controlled slip. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 249–251.

Brokowski, M., Peshkin, M., and Goldberg, K. Y. 1993 (Atlanta, GA). Curved fences for part alignment. *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 3, pp. 467–473.

Brost, R. C. 1988. Automatic grasp planning in the presence of uncertainty. *Int. J. Robotics Research* 7(1):3–17.

Brost, R. C. 1992 (Nice, France). Dynamic analysis of planar manipulation tasks. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2247–2254.

Brost, R. C., and Goldberg, K. Y. 1994 (San Diego, CA). A complete algorithm for synthesizing modular fixtures for polygonal parts. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 535–542.

Brown, R. G., and Jennings, J. S. 1995 (Pittsburgh, PA). A pusher/steerer model for strongly cooperative mobile robot manipulation. *Proc. IEEE/RSJ Int. Conf. on Robots and Systems*, Vol. 3, pp. 562–568.

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

Chvátal, V. 1983. *Linear Programming*. New York: W. H. Freeman.

Davis, C. 1954. Theory of positive linear dependence. *Am. J. Mathematics* 76:733–746.

Donald, B. R., Jennings, J., and Rus, D. 1995. Information invariants for distributed manipulation. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson (eds.): *Algorithmic Foundations of Robotics*. Wellesley, MA: A. K. Peters, pp. 431–457.

Erdmann, M. 1996 (Munich, Germany). An exploration of nonprehensile two-palm manipulation: Planning and execution. In G. Giralt and G. Hirzinger (eds.): *Robotics Research: The Seventh International Symposium*. London: Springer, pp. 16–27.

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

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

Goldberg, K. Y. 1993. Orienting polygonal parts without sensors. *Algorithmica* 10 (2-3-4):201–225.

Goldberg, K. Y. 1995. Completeness in robot motion planning. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson (eds.): *Algorithmic Foundations of Robotics*. Wellesley, MA: A. K. Peters, pp. 419–429.

Goldberg, K. Y., and Mason, M. T. 1990 (Cincinnati, OH). Bayesian grasping. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1264–1269.

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

Goyal, S., Ruina, A., and Papadopoulos, J. 1991b. Planar sliding with dry friction. Part 2. Dynamics of motion. *Wear* 143:331–352.

Higuchi, T. 1985. Application of electromagnetic impulsive force to precise positioning tools in robot system. In H. Hanafusa and H. Inoue (eds.): *Robotics Research: The Second International Symposium*. Cambridge, MA: MIT Press, pp. 281–285.

Huang, W. H., Krotkov, E. P., and Mason, M. T. 1995 (Nagoya, Japan). Impulsive manipulation. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 120–125.

Inoue, H. 1974. Force feedback in precise assembly tasks. Technical Report AIM-308, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge. Reprinted in 1979 in P. H. Winston and R. H. Brown (eds.): *Artificial Intelligence: An MIT Perspective*. Cambridge, MA: MIT Press.

Jia, Y.-B., and Erdmann, M. 1996 (Minneapolis, MN). Pose from pushing. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 165–171.

Kao, I., and Cutkosky, M. R. 1992. Quasistatic manipulation with compliance and sliding. *Int. J. Robotics Research* 11(1):20–40.

Kurisu, M., and Yoshikawa, T. 1994 (Kobe, Japan). Trajectory planning for an object in pushing operation. *Proc. 1994 Japan-USA Symp. on Flexible Automation*.

Liu, W., and Will, P. 1995 (Pittsburgh, PA). Parts manipulation on an intelligent motion surface. *Proc. IEEE/RSJ Int. Conf. on Robots and Systems*, Vol. 3, pp. 399–404.

Lynch, K. M. 1992 (Nice, France). The mechanics of fine manipulation by pushing. *IEEE Int. Conf. on Robotics and Automation*, pp. 2269–2276.

Lynch, K. M. 1996. Nonprehensile robotic manipulation: Controllability and planning (Ph.D. thesis). Technical Report CMU-RI-TR-96-05, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Lynch, K. M., and Mason, M. T. 1995a. Pulling by pushing, slip with infinite friction, and perfectly rough surfaces. *Int. J. Robotics Research* 14(2):174–183.

Lynch, K. M., and Mason, M. T. 1995b. Stable pushing: Mechanics, controllability, and planning. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson (eds.): *Algorithmic Foundations of Robotics*. Wellesley, MA: A. K. Peters, pp. 239–262.

Mani, M., and Wilson, W.R.D. 1985 (Berkeley, CA). A programmable orienting system for flat parts. *Proc. North American Manufacturing Research Institute Conf. XIII*.

Mason, M. T. 1982. Manipulator grasping and pushing operations (Ph.D. thesis). Technical Report AI-TR 690, Massachusetts Institute of Technology Artificial Intelligence Lab, Cambridge. Reprinted in 1985 in M. T. Mason and J. K. Salisbury (eds.): *Robot Hands and the Mechanics of Manipulation*. Cambridge, MA: MIT Press.

Mason, M. T. 1986. Mechanics and planning of manipulator pushing operations. *Int. J. Robotics Research* 5(3):53–71.

Mason, M. T., and Lynch, K. M. 1993 (Yokohama, Japan). Dynamic manipulation. *Proc. IEEE/RSJ Int. Conf. on Robots and Systems*, pp. 152–159.

Mottaez, A., and Goldberg, K. Y. 1993 (Boston, MA). Positioning polygonal parts without sensors. *SPIE Conf. on Sensors and Controls for Automated Manufacturing Systems*.

Nagata, K. 1994 (San Diego, CA). Manipulation by a parallel-jaw gripper having a turntable at each fingertip. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1663–1670.

Nemhauser, G. L., and Wolsey, L. A. 1988. *Integer and Combinatorial Optimization*. New York: John Wiley & Sons.

Nilsson, N. J. 1969 (Washington, DC). A mobile automaton: An application of artificial intelligence techniques. *Proc. Int. Joint Conf. on Artificial Intelligence*, pp. 509–520.

Okawa, Y., and Yokoyama, K. 1992 (Nice, France). Control of a mobile robot for the push-a-box operation. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 761–766.

Peshkin, M. A., and Sanderson, A. C. 1988a. Planning robotic manipulation strategies for workpieces that slide. *IEEE J. Robotics and Automation* 4(5):524–531.

Peshkin, M. A., and Sanderson, A. C. 1988b. The motion of a pushed sliding workpiece. *IEEE J. Robotics and Automation* 4(6):569–598.

Pham, D. T., Cheung, K. C., and Yeo, S. H. 1990 (Cincinnati, OH). Initial motion of a rectangular object being pushed or pulled. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1046–1050.

Rizzi, A. A., and Koditschek, D. E. 1993 (Atlanta, GA). Further progress in robot juggling: The spatial two-juggle. *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 3, pp. 919–924.

Salisbury, K. 1988. Whole arm manipulation. In R. C. Bolles and B. Roth (eds.): *Robotics Research: The Fourth International Symposium*. Cambridge, MA: MIT Press, pp. 183–190.

Sawasaki, N., Inaba, M., and Inoue, H. 1989 (Tokyo, Japan). Tumbling objects using a multi-fingered robot. *Proc. 20th Int. Symp. on Industrial Robots and Robot Exhibition*, pp. 609–616.

Schrage, L. E. 1981. *User's Manual for LINDO*. Palo Alto, CA: Scientific Press.

Simunovic, S. 1975 (Chicago, IL). Force information in assembly processes. *Proc. 5th Int. Symp. on Industrial Robotics*, pp. 415–431.

Swanson, P. J., Burridge, R. R., and Koditschek, D. E. 1995 (Nagoya, Japan). Global asymptotic stability of a passive juggler: A parts feeding strategy. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1983–1988.

Terasaki, H., and Hasegawa, T. 1994 (Munich, Germany). Motion planning for intelligent manipulations by sliding and rotating operations with parallel two-fingered grip-

pers. *Proc. IEEE/RSJ Int. Conf. on Robots and Systems*, pp. 119–126.

Trinkle, J. C., Abel, J. M., and Paul, R. P. 1988. An investigation of frictionless enveloping grasping in the plane. *Int. J. Robotics Research* 7(3):33–51.

Trinkle, J. C., Farahat, A. O., and Stiller, P. F. 1995. First-order stability cells of active multi-rigid-body systems. *IEEE Trans. on Robotics and Automation* 11(4):545–557.

Whitney, D. E. 1982. Quasi-static assembly of compliantly supported rigid parts. *ASME J. Dyn. Sys. Meas. Control* 104:65–77.

Zhuang, Y., Wong, Y.-C., and Goldberg, K. Y. 1994 (San Diego, CA). On the existence of modular fixtures. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 543–549.

Zumel, N. B., and Erdmann, M. A. 1994 (San Diego, CA). Balancing of a planar bouncing object. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2949–2954.

Zumel, N. B., and Erdmann, M. A. 1996 (Minneapolis, MN). Nonprehensile two palm manipulation with non-equilibrium transitions between stable states. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3317–3323.