Sweeping the Floor: Moving Multiple Objects with Multiple Disc-Shaped Robots*

Attawith Sudsang

Department of Computer Engineering, Chulalongkorn University, Bangkok 10330, Thailand
attawith@cp.eng.chula.ac.th

Abstract

This paper addresses the problem of transporting multiple objects in the plane with a team of disc-shaped robots. Using geometric properties of the objects, we present a method for computing positions of the robots that can kinematically constrain the objects to lie in a subset of the workspace. This computation is then used for deriving a motion plan of the robots for simultaneously pushing the objects to a given region. The approach is demonstrated in a simulation where a team of robots cooperatively sweep multiple objects that are scattered in a room to one of its sides.

1 Introduction

Consider a rectangular room with multiple objects scattered all over (Figure 1(a)). A representative of the goal of this work is to find a way to move the objects to one end of the room. Imagine if we could have a sufficiently long stick, we would be able to sweep all the objects at once by pushing the stick across the floor (Figure 1(b)). What the stick does is providing a kinematic constraint keeping the objects on one side of it. Of course, in the case where the objects are large, it may not be easy or practical to find a large stick and a robot powerful enough to push. Our idea is then to distribute the manipulation by replacing the stick with a team of disc-shaped robots (Figure 1(c)). Instead of the stick, the robots in a straight line formation cooperatively push the objects as they simultaneously move forward (Figure 1(d)). For this scheme to work, any gap between consecutive robots in the formation must not be too wide or some objects might be lost through the gap during the sweep. This condition requires us to solve two subproblems: (1) to compute how far a pair of consecutive robots can lie apart from each other while they still prevent the objects (with given geometries) from going through the gap, and (2) to synchronize the robots so that a specified interval of distance between consecutive robots be maintained during the entire sweeping motion. Before going into further detail, let us discuss some related works.

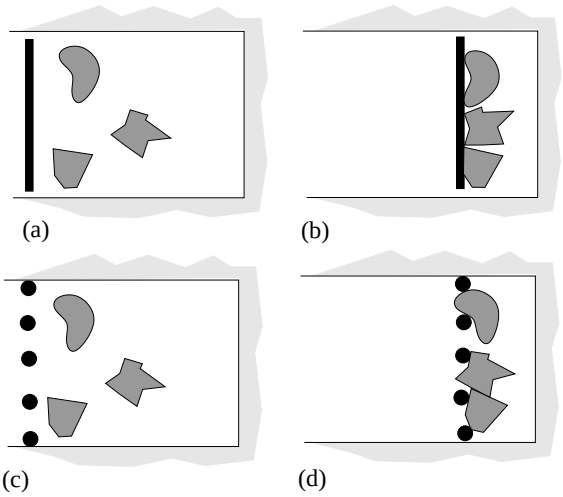

Figure 1: Moving multiple objects (a)-(b) using a stick, and (c)-(d) with multiple disc-shaped robots

The main idea of our work is based on using multiple robots, possibly together with the environment, to form kinematic constraint that can capture[1] the given objects in a bounded region of the workspace. By appropriately moving the robots while maintaining a capturing formation, the objects can be moved to a desired region. We do not need to know precisely where the objects are, but still can accomplish a desired transporting task. In other words, a capturing action provides a means with which uncertainty in the object's configuration can be handled. This also allows us to construct a simple manipulation plan without relying on the contact dynamics. Despite its usefulness, the process of capturing objects has so far received little attention in robotics. It was introduced in [14]

*The work is partially supported by a research grant from Ratchadaphisek Somphot Endowment

[1] an object is captured when it is restricted to stay within a bounded region of the workspace, i.e., there is no trajectory to bring the object to infinity

the concept of Inescapable Configuration Space (ICS) region, i.e., on the idea of characterizing the regions of configuration space in which the object is not immobilized but is constrained to lie within a bounded region of the free configuration space (see [12] for similar work in the two-finger case). This concept is used in [13] as a basis for computing a plan for manipulating polygonal objects using three disc-shaped robots. Although the contact dynamics can be safely ignored, polygonal object model is assumed and the resulting plans may contain many very short steps because the ICS region is often very small due to the computation that takes into account only three chosen edges. The ICS is defined in the combined object/robot configuration space while our capturing concept is purely derived in the workspace of the object. This difference in foundation results in a much simpler capturing condition than that of ICS and allows the entire boundary of the object to be taken into account.

Our approach to transporting objects applies cooperative pushing by a team of robots. In addition to the desire to capture multiple objects, the motivation comes from the difficulty in handling an object when it is too large to be grasped by a single robot, or when the available robot is not equipped with grasping capability. Influenced by [8], pushing has been recognized as a useful process in object manipulation [1, 7, 9]. The common approach of the works rely on the Coulomb friction model and the quasi-static assumption. Based on contact dynamics modeling which is a priori unverifiable, the assumption limits motion of the pusher to be slow enough that inertial forces can be ignored and requires that the contact between the pusher and the object be maintained during the entire manipulation. Avoiding these shortcomings, our approach to object manipulation bypasses the need for contact dynamics modeling by taking advantage of the ability to kinematically prevent the object from escaping. This is accomplished by computing trajectories of the robots such that the corresponding formation can always capture the object (as if the object is transported in a moving cage). The overall idea is similar to that of [13] except that our approach can handle multiple objects, can apply to objects other than polygons and extra robots may be easily added to a manipulation task for improving the robustness.

Recently, there have been a few works addressing the problem of manipulating multiple objects. An example is the distributed manipulation of multiple objects in small scale using a vibrating plate shown in [11]. Another interesting example is given in [3] where boxes are manipulated by wrapping ropes around them. This work shows a way to take advantage of the capturing constraint automatically provided by the wrapping protocol of a rope.

The remainder of the paper is organized as follows. We begin by considering a basic kinematic constraint that can be created by two robots. In particular, we present in Section 2 a sufficient condition guaranteeing that two pointed robots can form a gap for which the given object cannot pass through without colliding with the robots. Then in Section 3, this condition is used for arranging multiple robots in a capturing formation guaranteeing that the given set of objects surrounded in the formation are constrained in a bounded region of the workspace. Transporting multiple objects then roughly amounts to computing how to move the capturing formation to a desired region. We present the results from some simulation experiments in Section 4 and finally the conclusion in Section 5.

2 Impassable Gap

In this section, we present some background on a sufficient condition for capturing an object. The following content is mainly after our previous work in [**?**].

Two robots form an impassable gap for a given object if there exists no trajectory of the object that can bring it through the gap without colliding with the robots. Lemma 1, which is the main foundation of the work in this paper, states that two pointed robots form an impassable gap for a convex object when the distance between the robots is smaller than the *width* of the object (formally defined in Section 2.1). Then in Section 2.2, we extend Lemma 1 to derive Lemma 2 which describes a sufficient condition for two pointed robots to form an impassable gap for a given nonconvex object.

2.1 Impasse for a Convex Object

Let us consider a convex object $\mathcal{B}$.

Definition 1 *For a fixed orientation of $\mathcal{B}$, the parallel envelop $\mathcal{E}(\theta)$ is defined to be a pair of the closest parallel lines such that the angle between the lines and the x-axis is θ and the region bounded by the two lines contains the convex object $\mathcal{B}$ (see Figure 2). Also, let $d_{\mathcal{E}} : S^1 \mapsto I\!R$ be a function mapping an angle θ to the shortest distance between the two parallel lines of the envelop $\mathcal{E}(\theta)$.*

We follow [4] to call $d_{\mathcal{E}}$ as the diameter function. In that paper, the diameter function is used for computing the squeeze function for orientation planning of polygons using a frictionless gripper without sensors. The paper also presents an $O(n)$ algorithm for computing the diameter function of n-gon figures.

The following definition formally define the width of a convex object.

Definition 2 *The width of $\mathcal{B}$, denoted hereafter by W, is the minimum of the diameter function $d_{\mathcal{E}}(\theta)$ for all angles $\theta \in S^1$.*

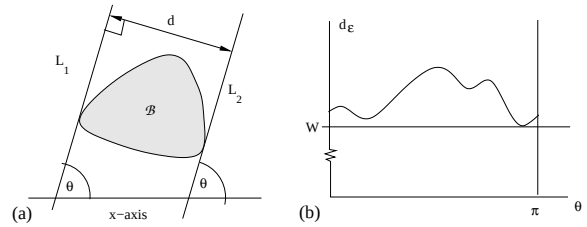

Figure 2: (a) Parallel lines L_1 and L_2 touch the convex $\mathcal{B}$ and form the parallel envelop $\mathcal{E}(\theta)$ with $d_{\mathcal{E}}(\theta) = d$, and (b) the corresponding function $d_{\mathcal{E}}$ in θ.

The following lemma is the key foundation of the work presented here. Let us imagine two rooms separated by an infinitely long straight wall with one open door. The lemma essentially states that a convex object cannot go from one room to the other if the door is narrower than the width of the object. The open door form a gap; imagining that two pointed robots are placed at the boundary of the gap (P_1 and P_2 in Figure 3), the lemma allows us to conclude that the two pointed robots form an impassable gap when the distance between the robots is smaller than the width of the convex object. A proof of the lemma can be found in [**?**]. The proof relies heavily on convexity of the object. It traces the two inward normals at the two intersection points between the object's boundary and a fixed vertical line as the object moves from one side of the line to the other. The proof shows that no matter which trajectory is chosen, the object always, at a certain moment, intersects the vertical line in a segment that is not smaller than the width of the object.

Lemma 1 *Let G be a vertical line with a gap of length d. If $d < W$, no trajectory can bring the convex object $\mathcal{B}$ from being entirely in one half plane completely to the other without colliding with G (the two half planes are separated by the line supporting G; see Figure 3(a)).*

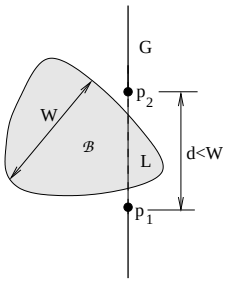

Figure 3: the object is confined in the left half plane by a gaped wall.

2.2 Impasse for a Nonconvex Object

The discussion up to this point has been concerned only with convex objects. In the following lemma, we give a sufficient condition for two robots to form an impassable gap for a nonconvex object. This lemma straightforwardly extends Lemma 1 using an intuitive fact that when part of a rigid object cannot go through a gap, neither the whole object can.

Lemma 2 *Let $\mathcal{A}$ be a nonconvex rigid object with a convex subset $\mathcal{B}$. Two pointed robots form an impassable gap for $\mathcal{A}$ when the distance between them is smaller than W where W denotes the width of $\mathcal{B}$. (Figure 4).*

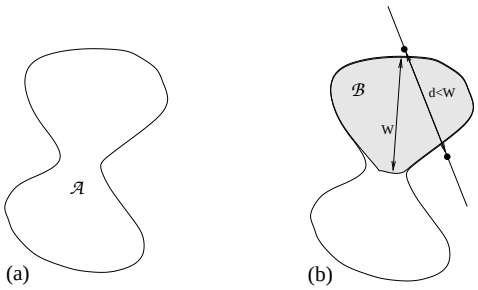

Figure 4: (a) a nonconvex object, (b) the object is captured when its convex subset (shaded region) is captured.

To form an impassable gap according to Lemma 2 for a nonconvex object, a convex subset of the object is needed. In the following, we describe how a convex subset could be obtained. Note here that the lemma holds for any convex subset but it is desirable to have a subset with larger width (to maximize the distance between the robots).

To find a convex subset of an object, first we find an inner polygonal approximation of the object. An inner polygonal approximation of a given figure is a polygon that is completely inside, and approximates the figure. The inner polygonal approximation can be found using a variation in two dimension of the polygonization technique in [6]. Clearly the resulting polygon is a subset of the given object. Next, we compute a convex subset of the polygon. Any subset of this polygon is obviously a subset of the given object as well. To find a convex subset of the polygon, we resort to a computational geometry technique for convex partitioning [10]. Convex partitioning problem asks how a polygon can be partitioned into a small number of convex pieces (Figure 5(b)). An algorithm giving the smallest number of partitions is presented in [2]. Using this heuristic makes sense because having fewer partitions usually implies larger partitions with larger width value. For each partition obtained, its width can be computed by applying the algorithm for computing the diameter function of a polygon in [4].

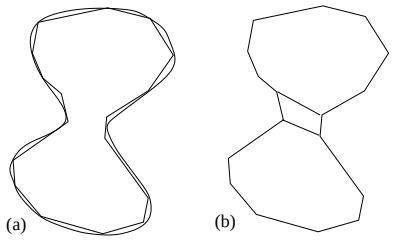

Figure 5: (a) inner polygonal approximation of the curve object shown in Figure 4(a), and (b) its convex partition.

So far we have assumed dimensionless pointed robots. For two disc-shaped robots with the same nonnegative radius r, it is easy to see that the shortest distance between the robots when their centers are far apart by distance d is equal to $d - 2r$. Taking this into account, to handle disc-shaped robots with a common radius $r \geq 0$, we generalize Lemma 2 by replacing W with $W + 2r$. This ensures that there exists a pair of points (one for each robot) such that the distance between the two points is less than W. With this consideration, without loss of generality, from now on, we will assume pointed robots.

3 Sweeping the Floor

We are now ready to apply the ideas discussed thus far to deal with the problem that is introduced at the beginning. Let us consider a rectangle R with one open end and whose sides are parallel to the x and y axes (Figure 6). Let G be a line segment parallel to the y-axis and passing through the rectangle. The segment G clearly divides the plane into two regions: bounded and unbounded. Let us call the bounded region by H and unbounded region by $\bar{H}$.

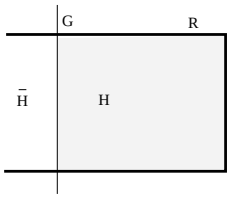

Figure 6: Regions H and $\bar{H}$ created by G and R

Suppose that there are n objects $\mathcal{B}_i, i = 1...n$ lying completely in the region H and let $W^* = \min\{W_i\}$, $1 \leq i \leq n$ where W_i denotes the width of the widest impassable gap according to Lemma 2 for the object $\mathcal{B}_i$. By Lemma 2, it is clear that if we place pointed robots along G by arranging that all gaps are narrower than W^*, we can guarantee that no object can pass through G to lie completely in the region $\bar{H}$ (Figure 7(a)).

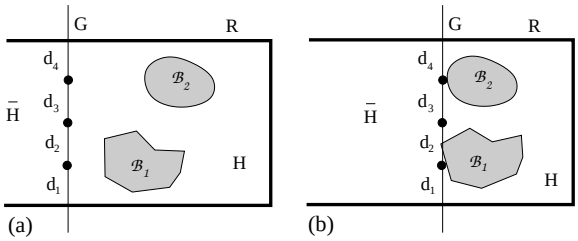

Figure 7: Moving objects using multiple robots attached to the line segment G (a) at the initial configuration, and (b) as G moves to the right

Let us attach the robots to the line segment G and consider the motion of the robots as G is translated to the right (Figure 7(b)). It is easy to see that the robots will simultaneously move to the right and the width of every gap remains unchanged during the entire motion of the robots. This means that if all the gaps are narrower than W^* before the motion, the objects will not be able to pass through G during the entire motion and they will be pushed into the shrunk bounded region H when the robots stop. In short, to move the objects to the right, simultaneously move the robots to the right. Because the evolving kinematic constraint created by the robots and the wall performs the manipulation, a simple transporting strategy is obtained. With this simplicity, there are, however, two critical issues: (1) what precision of the robot synchronization is needed?, and (2) how far can the robots continue to push?

Regarding the first issue, it is one of the main advantages of our approach that some deviation from perfect control and synchronization can be tolerated and this tolerance can be increased by adding extra robots. More precisely, the robots do not need to maintain fixed positions with respect to one another during the motion; they need to maintain only that the width of every gap is smaller than a specified value. By adding extra robots, we can distribute the robots more densely on the formation. This allows each robot to be able to independently move in a wider region relative to the others while still maintain that its associated gap is narrower than the specified width. This idea is illustrated in an example shown in Figure 8. In this example, the objects to be moved are three discs with the same diameter; we can then set the value of W^* to the common diameter. In Figure 8(a), only three robots are used and they are arranged evenly on G having the width of each gap equal to W^*. It is easy to see that to move the robots to the right while ensuring that the constraint on the gap's width is satisfied, the robots have to be perfectly synchronized so that they remain fixed with respect to each other. With two extra robots (Figure 8(b)), requirement on synchronization is relaxed; each robot may

move independently in a neighborhood (drawn as circle around the robot) while satisfying the gap's width constraint guaranteeing no object can escape.

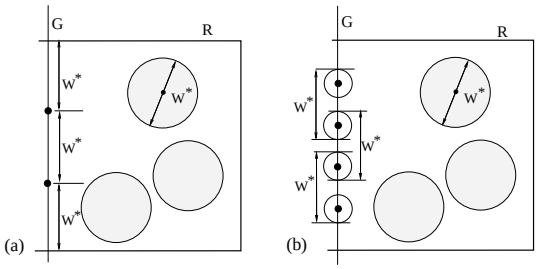

Figure 8: Robots shown as solid dot cooperatively moving three discs. Tolerance to error in synchronization and control can be improved by adding extra robots (a) no error in synchronization can be tolerated, and (b) each robot can lie independently in its neighborhood while still guarantee that the objects cannot escape

The second issue is crucial to the execution as we want to know how far we can command the robots to push. In practice, the robots could be stopped from moving further by the effect of jamming. Here we consider two classes of jamming. The first one, hard jamming, is when it is kinematically impossible for the robots to continue pushing. The robots and the objects interlock in such a way that the robots totally immobilize the objects and any motion of the robots in the intended direction would result in penetration. Figure 9(a) shows an example of hard jamming where the robots cannot continue pushing to the right. The second one, soft jamming, is when an object cannot be pushed further because it is in an equilibrium even though there exists a collision free trajectory for the object. An example of soft jamming is shown in Figure 9(b). In this illustration, the forces at the contact points P and Q of the object B cancel each other resulting in an equilibrium. We can see that in this case the equilibrium is independent of how hard the robot pushes; the wall will be able to react with an exact opposite force because the friction cone at P covers Q and likewise the friction cone at Q covers P. However, this does not mean that the object will always be stuck when it is at this particular configuration. The object may escape from this equilibrium by the dynamic effects such as impact and inertial forces.

In general, accurately predicting when hard jamming will occur is difficult; it requires simulating contact dynamics which is in a priori unverifiable. For soft jamming, the situation is less restricted as there are ways to avoid or reduce the possibility of soft jamming. A typical approach is to reduce the friction between the objects and the robots and the friction between the objects and the wall. The friction between the objects and the supporting floor is not to be concerned because it can be overcome by the robots

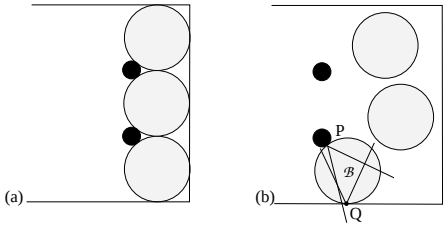

Figure 9: Two types of jamming: (a) hard jamming, and (b) soft jamming

pushing harder as it has a constant maximum magnitude (e.g., assuming the weight of the objects is fixed). In conclusion, due to the effect of jamming, it is difficult to tell in advance whether the robots will be able to push to a commanded distance. We need to rely on an online detection of jamming so a modified plan can be constructed. This issue is crucial to experiments with real robots and currently under our investigation.

In the case discussed so far, the wall of R is also needed to provide part of the capturing constraint. In fact, assuming that the initial configurations of the objects are given, without relying on the environment, the robots can form a cycle surrounding the objects to completely provide a capturing formation, again, by maintaining that the distance between consecutive robots is smaller than W^* (Figure 10(a)) . To move the objects to a new region, we command the robots to move accordingly while making sure that the distance contraint is satisfied during the entire motion. An advantage of this approach is that it is less likely to suffer from jamming than when the wall is involved. Nevertheless, it might seem that a large number of robots is needed; this is the price to pay for the simplicity in the manipulation planning. However, in general, some robots might not be necessary as the objects tend to shift to the capturing side opposite to the velocity direction of the moving formation. This is illustrated in Figure 10(b). This observation might be used as a heuristic for reducing the number of robots needed in the manipulation.

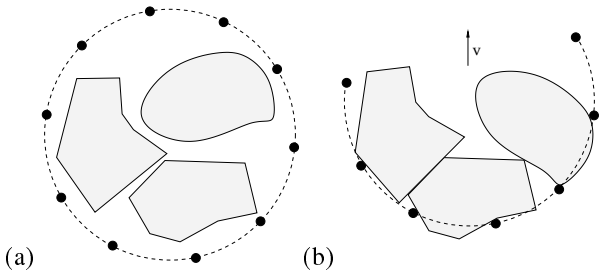

Figure 10: (a) A capturing cycle formation around the objects, where (b) some robots is not needed as the formation moves forward in the direction v

4 Simulation Results

We have implemented the algorithm for computing the width of an impassable gap for a given object. The program is written in the C++ programming language and runs on an 800 MHz PC. The current implementation can handle only polygonal objects. The program takes less than one second on each of the test polygons with over 30 vertices. For simulating the behavior of the objects when being pushed, we use MATLAB and Working Model 2D (see [5] about the use of the software for simulating rigid body motion) . In Figure 11, we show snapshots from a simulation run as the robots move to the right. The masses of the robots are set to be sufficiently large that the plan trajectory of the robots are not affected by collision with the objects.

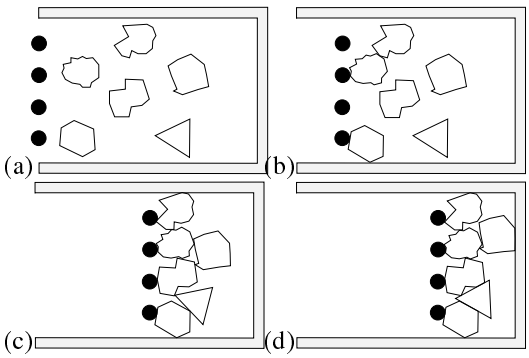

Figure 11: (a)-(d) snapshots from a simulation as the robots move to the right.

5 Conclusion and Future Works

We have presented a sufficient condition for two disc-shaped robots to form an impassable gap for a given object. This condition has been used in planning formation of multiple robots in transporting multiple objects with known geometries and initial configurations. Results from simulation experiments have been presented.

Future works include physical implementation using real robots. We are currently completing jam detection to be used in the implementation as discussed in Section 3. It is clear that when dealing with elongated objects, according to our approach, large number of robots are needed. We plan to explore how the number could be minimize or how planning strategy should be modified to specifically handle the presence of elongated objects. Other works include investigation of transportation planning with limited number of robots, and transportation planning for multiple 3D objects floating in space.

References

[1] S. Akella and M.T. Mason. Parts orienting by push-aligning. In *IEEE Int. Conf. on Robotics and Automation*, pages 414–420, Nagoya, Japan, May 1995.

[2] B. Chazelle and D. Dobkin. Optimal convex decomposition. In G. Toussaint, editor, *Computational Geometry*, Lecture Notes in Computer Sciences, pages 63–133. Noth-Holland, 1985.

[3] B. Donald, L. Gariepy, and D. Rus. Distributed manipulation of multiple objects using ropes. In *IEEE Int. Conf. on Robotics and Automation*, Sanfrancisco, CA, 2000.

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

[5] Russell C. Hibbeler. *Engineering Mechanics: Statics and Dynamics*. Prentice Hall, New York, 2001.

[6] W. Lorensen and H. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21:163–169, 1987.

[7] K.M. Lynch and M.T. Mason. Stable pushing: mechanics, controllability, and planning. In K.Y. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 239–262. A.K. Peters, 1995.

[8] M.T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986.

[9] M. Mataric, M. Nilsson, and K. Simsarian. Cooperative multi-robot box pushing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 556–561, Pittsburgh, PA, August 1995.

[10] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, NY, 1998.

[11] D. S. Reznik and J. F. Canny. C'mon part, do the local motion! In *IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea.

[12] E. Rimon and A. Blake. Caging 2D bodies by one-parameter two-fingered gripping systems. In *IEEE Int. Conf. on Robotics and Automation*, pages 1458–1464, Minneapolis, MN, April 1996.

[13] A. Sudsang and J. Ponce. A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane. In *IEEE Int. Conf. on Robotics and Automation*, Sanfrancisco, CA, 2000.

[14] A. Sudsang, J. Ponce, and N. Srinivasa. Grasping and in-hand manipulation: Geometry and algorithms. *Algorithmica*, 26(3):466–493, 2000.