Path Planning for Pushing a Disk using Compliance*

Dennis Nieuwenhuisen A. Frank van der Stappen Mark H. Overmars
Institute of Information and Computing Sciences
Utrecht University, The Netherlands
Email: {dennis,frankst,markov}@cs.uu.nl

***Abstract*— We consider the path planning problem for a robot that pushes a disk shaped object in an environment among obstacles. Instead of only allowing the object to move through the free space, we also allow the object to slide along the boundaries of the environment using compliance, extending the possibilities for the robot to find a push path.**

We present an exact algorithm that, given a path for the object consisting of k sections, preprocesses the environment consisting of n non-intersecting line segments in $O(n^2 \log n)$ and reports a push path in $O(kn \log n)$ time or reports failure if no path exists. Under the weak assumption of low obstacle density, the query time is reduced to $O((k+n)\log n)$.

***Index Terms*— pushing, compliance, exact, disk**

I. Introduction

Manipulation refers to a wide variety of changes that can be applied to an object. One form of manipulation is moving objects. Objects can be moved in many different ways, roughly divided in two classes, prehensile (using a form or force closure grasp) and non-prehensile. Prehensile manipulation includes grasping (for an overview see the book of Mason [19]) and squeezing [13]. Non-prehensile manipulation [17] includes pushing, rotating the support surface [12], rolling [2] and even throwing [21]. Also more passive forms of manipulation can be used, for example placing fences along a conveyor belt [7].

The objective of these manipulation actions can, for example, be to change the orientation of objects for industrial part feeding or navigating an object through an environment. In these cases, the object itself is passive and can only be manipulated if an external force is used. This force is applied on the boundary of the object or on the whole object (using e.g. gravitational forces). The result of the forces is that the object rotates, translates, or both. The exact motion of the object depends on a number of variables: the mass distribution of the object, the center of friction of the object, the point where the object is pushed and the friction. Friction can occur between the object and the surface, between the object and the manipulator or between the object and the environment (e.g. the walls). Because of the many variables, the resulting motion is usually difficult to predict and therefore often simplifications are used.

In the class of non-prehensile manipulation, pushing (see [9], [20]) has received the most attention. Here a pusher P pushes an object O from an initial configuration to a goal configuration while avoiding obstacles. If P moves such that the configuration of O is altered, this is called a *transfer* path. From time to time it may be necessary for P to change its contact point. A motion from one contact point to another while O does not move, is called a *transit* path. An important aspect of pushing is *controllability* [18]: is it possible to push an object from an initial state to a goal state? Because of the variables mentioned before, solving such problems can be very difficult. The problem shown in Fig. 2a for example requires many changes of the position of P in order to push O to its goal.

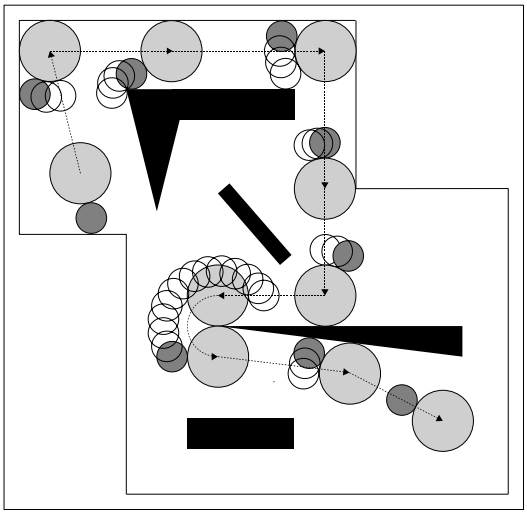

Fig. 1. An example created by an implementation of our algorithm where the pusher (the disk in dark gray) pushes the object (the disk in light gray) while making use of compliance. Also noncompliant sections are used. The desired path of the object is shown as the dotted line. At certain interesting points, the motions of the pusher are shown as sequences of circles. Without using compliance, it is impossible to push the object to its goal.

The path planning problem for a robot pushing an object using stable pushes [17] imposes *nonholonomic constraints* on the motion of the object (see e.g. [17], [18]). Because of these constraints, for example, once an object is pushed in one direction, it cannot be pulled in the reverse direction by reversing the motion of the pusher.

If an object is manipulated by pushing, the most important variable to take into account is the contact point. To reach the desired contact point, P needs to maneuver around O. Sometimes however, there is not enough room for P to reach the desired contact point or the contact point is blocked by an obstacle and no push path is found (see Fig. 2b). Also if the space for P to maneuver is limited, it may need many

*This research was supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2001-39250 (MOVIE - Motion Planning in Virtual Environments). Part of this research has been funded by the Dutch BSIK/BRICKS project.

transits to achieve the desired motion of O (see Fig. 2c). These problems can often be solved by exploiting the boundaries of the environment. The rationale behind this is that as soon as O touches one of the boundaries of the environment, they cause O to change its direction of motion to a path parallel to the boundary. The contact point changes to a set of contact points, which all cause O to follow the same path parallel to the boundary. Such a motion of the object is called a *compliant motion*. An example of a push path using compliance is shown in Fig. 1.

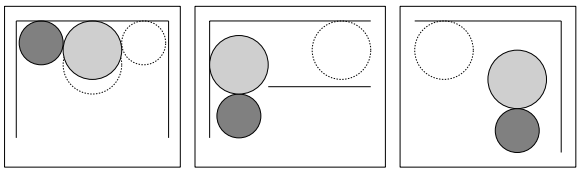

Fig. 2. The pusher P is shown in dark gray, the object O is shown in light gray, the destination of O is shown as a dotted disk. If no compliance is allowed, the push path may get very complicated or even impossible to create. (a) An example where the space of P to maneuver is very limited, resulting in a very complex pusher path with many transits (alternating between its current position and the dotted position). (b) Even though a simple path for O exists to reach its destination, no push path can be created without compliance. (c) Because P has little space to maneuver, it needs a complicated path to push O to its destination if no compliance is allowed.

In this paper, we present an exact algorithm that, given a path for O, calculates a path for P such that O is pushed along the given path or reports failure if no pusher path exists. Our algorithm takes $O(n^2 \log n)$ preprocessing time and reports a push plan in $O(kn \log n)$, where k is the complexity of the object path. If low obstacle density is assumed (Section V), the query time is reduced to $O((k+n)\log n)$. To create the path for O, any compliant motion planning technique can be used as long as the resulting paths consist of straight lines and circular arcs (the circular arcs can be used to rotate compliantly about a vertex). In addition noncompliant sections are allowed, for example to act as a bridge between two compliant sections. We assume both O and P are disks. Even for this restricted case it turns out the problem is complicated, for example the situation of Fig. 2a is difficult to solve even with compliance, while Figs. 2b+c benefit greatly from compliance.

Although pushing has received considerable attention over the years, using compliance as an aid to maneuver the object with a pusher, has not. Often compliance is used to compensate for uncertainty. In [16] the *preimage backchaining* approach is introduced. The idea is to compute the points from which the robot can reach the goal. Then the preimage is iteratively treated as a new goal until the initial robot configuration has been found. The concept of a *back projection* is introduced by [11] and [8] improves an algorithm introduced in [10] to find a trajectory from a start region to a goal region amidst planar polygonal obstacles where control is subject to uncertainty in $O(n^2 \log n)$ time. The coordinated motion planning problem of two independent robots in a plane using a cell decomposition is solved in $O(n^2)$ time in [22] and [3]. In our results the robots are not independent, their motions need to be coordinated such that one pushes the other.

II. Preliminaries and problem statement

Given disks O and P having radii r_o and $r_p < r_o$ in an environment of disjoint line segments L and a trajectory $\tau : [0,1] \rightarrow \mathbb{R}^2$ of O, we compute a push plan σ for P such that if P complies to this plan, it pushes O along τ or we report that no path for P exists. We allow the contact point to slide around the boundary of O. It is assumed that the friction between O and the supporting plane is large enough such that there is no motion of O after pushing ceases (quasistatic assumption). Finally no friction between O and the environment is assumed, although this restriction can be lifted easily (see Section VI).

A. Definitions

All angles used throughout this paper are defined in an absolute coordinate system (i.e. relative to a world frame) unless stated differently.

We denote the Euclidian distance between two objects a and b by $d(a,b)$. The disks O and P have radii r_o and r_p and are centered around c_o and c_p. For the rest of the definitions and of this paper, we assume that P is always *in contact* with O: $d(c_o, c_p) = (r_o + r_p)$. Thus we do not allow transit paths in which P loses contact with O. See the conclusions for some remarks about allowing such transit paths.

The object path τ consists of a set $I = \{1, .., k\}$ of k sections, each occupying a subsequent interval of $[0,1]$. Each $i \in I$ is either a straight line or a circular arc. The section containing position $s \in [0,1]$ is indicated by $I(s) : [0,1] \rightarrow \{1, .., k\}$. The start and endpoints of section $i \in I$ are called i_s and i_e. Every section is open in its endpoint, thus: $I(i_s) = i$ and $I(i_e) = I((i+1)_s) = i+1$. The end and start points of two subsequent sections are connected such that the path of O is C^0 continuous i.e. $\tau(i_e) = \tau((i+1)_s)$. Throughout this paper we will refer to a path section simply as *section*.

Definition II.1 (contact transit)**.** *A* contact transit *is a motion of P while O stands still, it is a circular shaped motion of radius $r_o + r_p$.*

Every section $i \in I$ is either compliant or noncompliant.

Definition II.2 (compliant)**.** *A section $i \in I$ defined on the domain $s \in [i_s, i_e]$ is called* compliant *with obstacle $l = (v_0, v_1) \in L$ if $\forall s \mid d(\tau(s), l) = r_o$. In this case l is called the* compliant edge*. If the closest point is either v_0 or v_1 for the whole domain, then this vertex is called the* compliant vertex.

A noncompliant section is a section where O does not touch any $l \in L$. From the definition it follows that there are two types of compliant sections: a straight line compliant section that is compliant with an obstacle edge and a circular compliant section that is compliant with an endpoint of an obstacle edge. Both types are shown in Fig. 3.

Since we assume that O and P are in contact for all $s \in [0,1]$, we define the push plan σ for P as $\sigma : [0,1] \rightarrow [0, 2\pi)$.

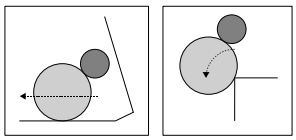

Fig. 3. The two types of compliant sections, shown as the dotted lines. O is shown in light gray, P in dark gray. (a) A straight line compliant section. (b) A circular shaped compliant section.

The push plan σ defines the position of P for a corresponding position on the trajectory τ.

Definition II.3 (push position)**.** *At position s on the path, the position of P is denoted by $\sigma(s)$. $\sigma(s)$ is an angle relative to the center of O. At position s, the world coordinates of P are:*

$$\tau(s) + \begin{pmatrix} \cos(\sigma(s)) \\ \sin(\sigma(s)) \end{pmatrix} (r_o + r_p).$$

For the desired object path τ our goal is to calculate a corresponding push plan σ such that if P complies to this path, it pushes the object along τ.

Definition II.4 (free push range)**.** *The* free push range $\text{FPR}(\tau(s))$ *is the set of push positions that do not collide with any $l \in L$ (see Fig. 4a).*

Definition II.5 (push range)**.** *For every $\tau(s)$ a set of push positions $\text{PR}(\tau(s))$ is specified, called* the push range*. $\text{PR}(\tau(s))$ is defined such that if P pushes O while $\sigma(s) \in \text{PR}(\tau(s))$ then O follows τ. The push range is a continuous set of angles from $\text{PR}_b(\tau(s))$ to $\text{PR}_e(\tau(s))$. The position of the pusher $\sigma(s) \in \text{PR}(\tau(s))$ iff $\sigma(s)$ is in the interval of the smallest rotation between $\text{PR}_b(\tau(s))$ and $\text{PR}_e(\tau(s))$.*

If $i \in I$ is a compliant section (see Fig. 4b) then:

- $\text{PR}_b(\tau(s)) = \arctan(\tau'(s)) + \pi$.
- $\text{PR}_e(\tau(s)) = \arctan(\tau'(s)) + \frac{\pi}{2}$.

This also holds for circular compliant sections. If $\sigma(s) \in \text{PR}(\tau(s))$, O slides along the compliant edge or rotates about a vertex. If i is noncompliant then:

- $\text{PR}_b(\tau(s)) = \arctan(\tau'(s)) + \pi$.
- $\text{PR}_e(\tau(s)) = \text{PR}_b(\tau(s))$.

This definition shows that for a noncompliant section, there is only one push position such that O is pushed along the desired path section.

Depending on the direction of $\tau(s)$ it may be necessary to subtract π from both $\text{PR}_b(\tau(s))$ and $\text{PR}_e(\tau(s))$. For a compliant section, the position of $\text{PR}_b(\tau(s))$ always moves toward $\tau(s)$ while the position of $\text{PR}_e(\tau(s))$ maintains a distance of $r_o + r_p$ from $\tau(s)$.

Definition II.6 (valid push range)**.** *The* valid push range *(Fig. 4c) consists of those push positions that are both free and within the push range:*

$$\text{VPR}(\tau(s)) = \text{FPR}(\tau(s)) \cap \text{PR}(\tau(s))$$

$\text{VPR}(\tau(s))$ may consist of multiple intervals, split up by obstacles if $I(s)$ is a compliant section. At most one of these intervals is reachable for P from its current position. Therefore we define the *reachable valid push range*.

Definition II.7 (reachable valid push range)**.** *The* reachable valid push range *(Fig. 4d),* $\text{RVPR}(\tau(s), \sigma(s))$ *is the set of push positions that is reachable for the pusher from its current position $\sigma(s)$ and object position $\tau(s)$ using a contact transit.*

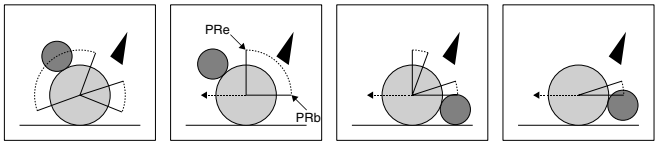

Fig. 4. Illustrations of the four ranges for a compliant section. P is shown in dark gray, O in light gray, obstacles are black. The ranges show the possible placements of the center of P. (a) The free push range FPR, these are all push positions for which the pusher is collision free. (b) The push range PR consists of all push positions that result in a motion of O in the desired direction (shown as the dotted arrow). (c) The valid push range $\text{VPR} = \text{PR} \cap \text{FPR}$. (d) The reachable valid push range RVPR; the part of VPR that is actually reachable for P from its current position by a contact transit.

B. Problem statement

Using the definitions of the previous section, we formally define our problem:

Given a collision free desired path τ for O consisting of k sections, that are allowed to be compliant, amidst a collection of disjoint obstacle line segments L, create a push plan σ for P such that if P complies to this plan, it pushes O along τ.

C. Preliminaries

The object path, given by the user, consists of k_c compliant sections and k_n noncompliant sections ($k_c + k_n = k$). The only restrictions on this path are that its noncompliant sections consist of straight lines and the compliant sections follow parts of the union boundary of the Minkowski sum $\cup_{l \in L}(l \oplus D)$ where D is a disk of radius r_o. The union boundary consists of line segments and circular arcs of radius r_o.

We will frequently use the union boundary of the Minkowski sum of $\cup_{l \in L}(l \oplus D)$ where D is a disk of radius r_p. This union boundary consists of all placements for the center of P where P is in touch with an obstacle. In the rest of this paper we shall denote this union boundary of the Minkowski sum by *the union boundary*. Since the Minkowski sums of each pair of line segments are pseudodisks and the obstacles together form a collection of pseudodisks, the complexity of the union boundary is $O(n)$ (see [6]).

III. Global approach

In order solve the problem stated in Section II-B, we need to create a push plan for P such that P avoids the obstacles. Without loss of generality, we divide the problem into four subproblems all representing one type of section. All 4 cases are illustrated in Fig. 5.

- creating a push plan for straight-line compliant sections
- creating a push plan for circular compliant sections
- creating a push plan for noncompliant sections
- finding a contact transit to reach $\text{VPR}(\tau(i_s))$ at the start of section $i \in I$.

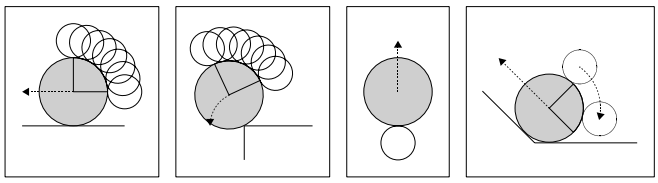

Fig. 5. The four possible sections. (a) For a straight line compliant section, we need to find a push plan in which the pusher is at one of the illustrated positions. (b) For a circular compliant section the push range rotates as O advances on its section. (c) For a noncompliant section of O, there is only 1 push position. (d) If the pusher is outside the push range at the start of a section, a contact transit is necessary.

A. Straight line compliant sections

For a straight line compliant path segment $i \in I$ on the domain $s \in [i_s, i_e]$, the corresponding push plan may consist of multiple push plan sections because P may need to avoid obstacles. We call a push plan monotonically descending when $\sigma(s)$ only moves in the direction of $\text{PR}_b(\tau(s))$ while pushing O along the section. A monotonically descending path exists if we assume that at the start of the section P is as close to $\text{PR}_e(\tau(i_s))$ as possible. We start by pushing until P hits an obstacle. Next, we let P follow the union boundary for as long as the path of P is descending (after that, P fits underneath the obstacle). Now, we again find the first obstacle that will be hit by P (which is now at a different position). We repeat this procedure until we have reached the end of the section. This approach ensures that P only changes its position if necessary. The path σ of P consists of straight line sections and circular sections. As each obstacle can be hit only once (in one object path section), in total at most n changes of the pusher position are required per section.

Note that we could also start as close to $\text{PR}_b(\tau(i_s))$ as possible but then no monotonically ascending or descending path for P can be guaranteed. The complexity of both approaches is the same.

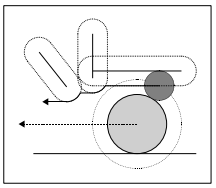

Fig. 6. An example of a pusher path (shown as the solid black arrow) avoiding two obstacles. The desired path of the object is shown as the dotted arrow. The dotted lines show $L \oplus D$ and $O \oplus D$, where D is a disk of radius r_p. For clarity the compliant edge at the bottom is not dilated.

B. Circular compliant segments

When the object is pushed compliant around an outer vertex in a certain section $i \in I$ on the domain $s \in [i_s, i_e]$, the section of O is circular shaped. If P is at the same relative position inside $\text{PR}(\tau(s))$ during the section, the path of the pusher is also circular. The radius of this pusher path depends on the exact position of P within $\text{PR}(\tau(s))$. If $\sigma(s) = \text{PR}_e(\tau(s))$ then the radius of the circular shaped path the pusher follows is $2r_o + r_p$ (Fig. 7a). If $\sigma(s) = \text{PR}_b(\tau(s))$, this radius is smaller. The smaller the radius of the path of the pusher, the smaller the combined sweep planes of O and P (because a larger part of the pusher moves inside the shadow region of the object which is collision free by definition).

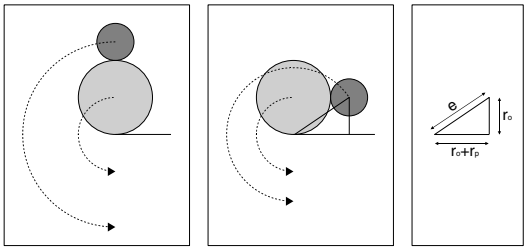

Fig. 7. Pushing O around a vertex. (a) If $\sigma(s) = \text{PR}_e(\tau(s))$, its path is circular shaped of radius $2r_o + r_p$. (b) The pusher path if $\sigma(s) = \text{PR}_b(\tau(s))$. (c) Calculating the radius of the pusher path of (b). It can easily be seen that $e = \sqrt{r_p^2 + 2r_p r_o + 2r_o^2}$.

Using this observation, we use the following approach to create a pusher path. First, we try to determine the smallest $s : \text{PR}_b(\tau(s)) \in \text{RVPR}(\tau(s), \sigma(s))$. If this s exists, we transit P to $\text{PR}_b(\tau(s))$ and maintain that position for the rest of the section. This preserves completeness because this position for P maximizes the part of the path of P that is within the shadow region of O. As soon as $\text{PR}_b(\tau(s)) \in \text{RVPR}(\tau(s), \sigma(s))$, the pusher can follow a path where $\sigma(s) = \text{PR}_b(\tau(s))$ for the rest of the section (provided that this path for P is collision free, else we report failure). If $\sigma(s) = \text{PR}_b(\tau(s))$ during a circular compliant section, then the radius of the path of P equals $\sqrt{r_p^2 + 2r_p r_o + 2r_o^2}$ (see Fig. 7b).

Before the pusher can follow a path for which $\sigma(s) = \text{PR}_b(\tau(s))$ however, it first needs to reach $\text{PR}_b(\tau(s))$. For this, we use the following approach, as illustrated in Fig. 8a..d. At the start of the section, we try to move P in a straight line perpendicular to the compliant edge. This will start pushing O around the vertex. If an obstacle is encountered, we follow the union boundary as long as P is moving toward the vertex of the compliant edge. In the meantime P pushes O along its desired path. The process is repeated until $\sigma(s) = \text{PR}_b(\tau(s))$ or $s = i_e$.

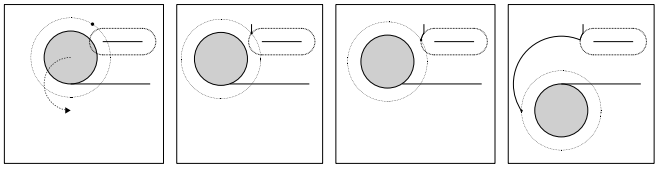

Fig. 8. An example of a pusher path that pushes O around a vertex. P is shown as a dot that represents its center. (a) The situation at the start of the section. The dotted arrow shows the trajectory of O. (b) First we push perpendicular to the compliant edge until an obstacle is encountered. (c) We follow the union boundary of the obstacles until $\sigma(s) = \text{PR}_b(\tau(s))$. (d) Now we push following a circular shaped path having radius $\sqrt{r_p^2 + 2r_p r_o + 2r_o^2}$.

Note that an exception occurs when at the start of the section, $\text{PR}_b(\tau(i_s))$ is reachable but an obstacle blocks the first part of the path (Fig. 9a). In this case, we partially follow the union boundary in order to avoid the obstacle (Fig. 9b).

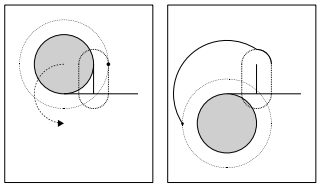

Fig. 9. (a) Even though $\text{PR}_b(\tau(i_s))$ is initially reachable for P, it is blocked a little later. (b) This is resolved by first following the union boundary.

C. Noncompliant sections

If a section $i \in I$ on the domain $s \in [i_s, i_e]$ is noncompliant, then $\text{PR}_b(\tau(s)) = \text{PR}_e(\tau(s))$. Thus for a noncompliant section the push range consists of only one single push position. If $\text{PR}_b(\tau(s)) \in \text{FPR}(\tau(s))$ on the entire domain, a push plan exists. Only if an obstacle is present in the wedge (see Fig. 10) between the pusher and the object when $s = i_s$ and the section is long enough such that P actually collides with this obstacle, then no push plan for this section exists and we report failure.

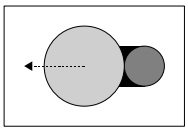

Fig. 10. For a noncompliant section, only if an obstacle is present in one of the wedges (shown in black) at the start of the section, no push plan may exist.

D. Contact transits

Contact transits are necessary at the start of section $i \in I$ if $\sigma(i_s) \notin \text{VPR}(\tau(i_s))$. In this case, we need to construct a contact transit for the pusher such that $\sigma(i_s) \in \text{VPR}(\tau(i_s))$. Such a situation can only occur at the start of a section.

When a contact transit is necessary, P has the choice between a clockwise or counter-clockwise path. If the current or next section is compliant, then one of these paths is always blocked by the compliant edge. During a contact transit, the path of P is always circular shaped having radius $r_p + r_o$. If both paths are blocked by an obstacle then the transit is not possible and failure is reported.

IV. The data structures

In the previous section we described the global approach. For actually computing the path, a number of collision queries must be answered. In this section we will describe the data structures that are needed for this.

A. Straight line compliant sections

A straight line compliant section $i \in I$ is defined on the domain $s \in [i_s, i_e]$. We now want to determine the first obstacle that forces P to change its position. For this, we use ray shooting. A ray is shot from position of the center of the pusher at $s = i_s$, parallel to the compliant edge in the direction of the section of O and the first element of the union boundary that the ray hits corresponds to the first obstacle that will be hit by P. Next, we push O until P actually hits the obstacle. Now P can follow the union boundary as long as this boundary is monotonically descending (as shown in Fig. 6). If the boundary ceases to descend monotonically, the pusher is at a position where it fits underneath the obstacle. Since the obstacles consist of line segments, we cannot encounter this obstacle again during section i. We establish the next obstacle that the pusher will hit, again we use ray shooting from the current position of P parallel to the compliant edge. This procedure is repeated until the object reaches the end of the section.

In order to find the colliding obstacle we need to perform a ray shooting query in a space consisting of circular arcs and line segments (that form the union boundary of the Minkowski sum of the obstacles and P). A procedure for ray shooting amidst possibly intersecting algebraic arcs in the plane is given in [15]. It describes a data structure that can be preprocessed in time $O(n^2 \log n)$ and has size (n^2). Ray shooting queries can then be solved in $O(\log n)$ time. Since, in worst case, an element of the union boundary can be encountered at every straight line compliant section, the total number of ray shooting queries is $O(k_c n)$.

Lemma IV.1. *We can preprocess the scene in $O(n^2 \log n)$ time such that a push plan for a straight line compliant section can be found in $O(n \log n)$ time. There are k_c compliant sections, resulting in a total query time of all straight line compliant sections of $O(k_c n \log n)$.*

B. Circular compliant sections

As stated before, the preferred position of P during a circular compliant section $i \in I$ is $\text{PR}_b(\tau(s))$. We try to find the smallest $s : s \in [i_s, i_e]$ for which $\text{PR}_b(\tau(s)) \in \text{RVPR}(\tau(s), \sigma(s))$. If this s exists, then $\sigma(s) = \text{PR}_b(\tau(s))$ and from then on the radius of the circular shaped path of P equals $\sqrt{r_p^2 + 2r_p r_o + 2r_o^2}$ (Fig. 7). The number of positions where a circular compliant section can occur, is $2n$ (at every vertex of the obstacles). We can preprocess the environment by finding all intersections of all possible circular shaped paths of P with the union boundary. For this we add circular arcs with radius $\sqrt{r_p^2 + 2r_p r_o + 2r_o^2}$ to the environment at every vertex. The total number of elements in this environment is $O(n)$. We find intersections by using a plane sweep algorithm. The original version of this algorithm is described in [5] by Bentley and Ottmann. Balaban [4] described an algorithm that also works for curved segments. His algorithm takes $O(n \log n + q)$ time and uses $O(n)$ space to list all q intersections. In worst case, if all obstacles are close together, $q = O(n^2)$, resulting in a time bound of $O(n^2)$ in our situation. The algorithm of Balaban requires the segments to have at most one intersection with any vertical line, which means that we may need to split certain circular shaped path sections of P into two separate pieces.

In order to reach $\text{PR}_b(\tau(s))$ (see Fig. 8), we proceed similar to Section IV-A. First, we shoot a ray from the current position of P to the compliant edge in a direction perpendicular to the compliant edge. Then, we push O until P collides. Next,

we follow the union boundary as long as P is moving toward the vertex of the compliant edge. Then, we shoot another ray. This procedure is repeated until $\sigma(s) = \text{PR}_b(\tau(s)) \vee s = i_e$. If $\sigma(s) = \text{PR}_b(\tau(s))$ we continue with a circular compliant section for P having radius $\sqrt{r_p^2 + 2r_p r_o + 2r_o^2}$. For the ray shooting we can reuse the structure of Section IV-A.

Lemma IV.2. *All circular compliant sections together cost $O(n^2)$ preprocessing time in order to find all intersections. For the ray shooting part, we reuse the data structure of Section IV-A. In worst case we encounter every obstacle at every circular section. Thus, the ray shooting queries for all circular sections together cost $O(k_c n \log n)$ time.*

C. Noncompliant sections

For a noncompliant section $i \in I$, there is only one valid push position. The only way to invalidate the pusher path is if an obstacle is present in one of the wedges shown in Fig. 10 and i is long enough for the collision to actually occur. We can check this by shooting a ray from the pusher position at the start of the segment in the direction of $\tau(i_s)$. If the ray collides with the union boundary before the end of the section, then no pusher path for this section exists. We can reuse the data structure created in Section IV-A. Since there are k_n noncompliant sections, the total query time is $O(k_n \log n)$.

Lemma IV.3. *The total query time used for the noncompliant sections together is $O(k_n \log n)$.*

D. Contact transits

Since the object path is known in advance, we also know all positions where contact transits may be necessary. Since the total number of sections of τ is k, so is the number of contact transits. The path of P during a contact transit is always circular shaped of radius $r_p + r_o$ (in the case of two subsequent noncomplaint sections there are two possible directions and thus two possible circular paths).

If the circular arc of the path of P intersects the union boundary, then no transit following this path is possible. We can again preprocess the environment using Balaban's algorithm by finding all intersections of the contact transit circular arcs with the union boundary as described in IV-B. In worst case we encounter every obstacle at the start of every section resulting in kn intersections.

Lemma IV.4. *In order to find all possible intersections during contact transits, in the query phase we use $O((k+n)\log(k+n) + kn)$ time.*

E. Run time analysis

For the two types of compliant sections, we need ray shooting. Also for the noncompliant sections, one ray shooting query is necessary. The preprocessing time for the algorithm is $O(n^2 \log n)$. We can also preprocess the plane sweep necessary for the circular compliant sections. According to Lemma IV.2, this preprocessing takes $O(n^2)$ time.

Combining Lemmas IV.1, IV.2 and IV.3 results in a total ray shooting query time of $O(kn \log n)$. For the contact transits we need a total query time of $O((k+n)\log(k+n) + kn)$ (Lemma IV.4).

Theorem IV.1. *A push plan such that the object is pushed along a path consisting of both compliant and noncompliant sections can be calculated in $O(kn \log n)$. The preprocessing takes $O(n^2 \log n)$ time. The complexity of the pusher plan is $O(kn)$.*

V. Low obstacle density

In practical settings, the complexity of the free space tends to remain far below the theoretical worst-case complexity bounds. A realistic assumption about the complexity of the free space is *low obstacle density*. Motion planning in environments having low obstacle density has been well studied, see e.g. [23]. Before we formally define low obstacle density, we first devise a useful property of the object path.

If a straight line compliant section $i \in I$ is longer than some threshold d and O has been pushed at least a distance d on the current section, we are certain that on the domain $s \in [i_s + d, i_e]$ the following holds: $\text{PR}_b(\tau(s)) \in \text{RVPR}(\tau(s), \sigma(s))$.

Thus, if $s = i_s + d$ we are certain to be able to transit P to $\text{PR}_b(\tau(i_s + d))$ and stay at that position for the rest of the section. This can be seen as follows. Since it is not possible for an obstacle to enter the shadow region O after the start of the section, an obstacle that blocks $\text{PR}_b(\tau(s))$ from being in $\text{RVPR}(\tau(s), \sigma(s))$ already does so at the start of the section, see Fig. 11 for the calculation of d.

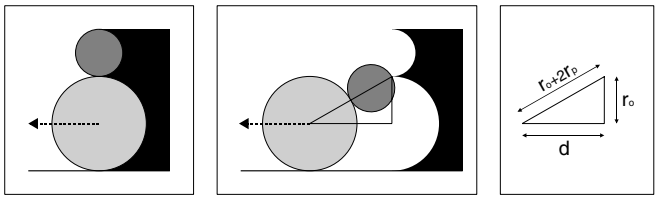

Fig. 11. Calculating the minimum distance d that O has to move to guarantee that $\text{PR}_b(\tau(i_s+d)) \in \text{RVPR}(\tau(i_s+d), \sigma(i_s+d))$. The black area is the union of all possible obstacles. (a) The start of the object path section. (b) The first chance for P to reach $\text{PR}_b(\tau(i_s + d))$. (c) The calculation of d. It is easy to see that $d = \sqrt{4r_p^2 + 4r_p r_o}$.

Lemma V.1 (Reachability of PR_b)**.** *If $(i_e - i_s) > \sqrt{4r_p^2 + 4r_p r_o}$ for compliant segment i, the following holds for $s \in [i_s + \sqrt{4r_p^2 + 4r_p r_o}, i_e]$:*

$$\text{PR}_b(\tau(s)) \in \text{RVPR}(\tau(s), \sigma(s))$$

Usually low obstacle density is defined as follows:

Definition V.1 (Low obstacle density)**.** *Let $\mathbb{R}^2$ be a space with a set Γ of obstacles. Then $\mathbb{R}^2$ is said to be a* low obstacle density *space if any region with minimal enclosing circle of radius λ intersects at most a constant number of objects $E \in \Gamma$ with minimal enclosing circle of radius at least $c\lambda$ for some constant $c > 0$.*

The total sweep plane of P and O together in a circular compliant section always fits within a disk of radius $3r_o$

(recall that $r_p < r_o$). For a straight line compliant section, using Lemma V.1 it is easy to see that the total sweep plane of P and O together on the domain $s \in [i_s, i_s + \sqrt{4r_p^2 + 4r_p r_o}]$ also fits within a disk of radius $3r_o$.

Let λ be the length of the smallest obstacle $l \in L$. If our environment L with n non-intersecting line segments satisfies the low obstacle density property, then at most a constant number of obstacles intersects with a disk shaped region of radius $c\lambda$ for some $c \geq 0$.

If we assume that $r_o < c\lambda$ for some $c \geq 0$, then a disk shaped region of radius r_o intersects at most a constant number of obstacles. This implies that a disk shaped region of radius $3r_o$ also intersects at most a constant number of obstacles. Since each obstacle can be encountered at most once every section, we know that P can encounter at most a constant number of obstacles in a compliant section. Therefore at most a constant number of ray shooting queries is needed for a compliant segment if the environment satisfies the low obstacle density.

We now partially restate Theorem IV.1.

Theorem V.1. *If the environment satisfies the low obstacle density property and if the radius of O is at most a constant times the size of any obstacle, then a push plan such that O is pushed along a path consisting of both compliant and noncompliant sections can be calculated in $O((k+n)\log n)$ time. The complexity of the pusher plan is $O(k)$.*

VI. Conclusions and future work

In this paper we have studied the contact pushing problem for two disks where we are allowed to make use of compliance in order to extend the number of pushable paths. Exploiting the boundaries increases the possibilities of finding a push plan. The problem can be solved in $O(kn \log n)$ time and, using a realistic assumption about the environment, it can even be solved in $O(k \log n)$ time. Fig. 1 shows an example of a path created by an implementation of our planner.

We have ignored the friction between O and the environment but this can easily be incorporated by narrowing the push range according to the friction cone. Since compliance is used, friction between O and P is allowed because it does not affect the direction of motion of O at a compliant section.

At certain discretized positions along the path of O it may be necessary to perform a non-contact transit of P (moving to another interval of VPR) in order to find a path. We have found a polynomial algorithm to incorporate non-contact transits, and we are currently working to improve this algorithm.

A challenging open problem is to adapt the algorithm to work for non-disc shaped objects. In that case, PR is not only dependent on the compliant edge but also on the topology of the object. Also not all pushes will be stable, i.e. pushing at certain positions on the boundary of O will cause O to rotate despite the compliant edge.

In this paper we strictly follow the given path of O. We could however, given a start and goal configuration, create a path for O such that it is guaranteed we find a push plan (provided one exists). Using our algorithm we can verify the pushability of the compliant sections. Noncompliant sections can then be added to act as bridges between compliant parts of the path. In order to create these bridges, non-complete methods such as PRM [14] could be used.

An open problem is the maximum complexity of σ. We have proved that for pushers of radius $r_p < (3 - 2\sqrt{2})r_o$ the complexity of the push plan is $O(k)$ (without any assumption about the obstacle density), but for larger pushers this has yet to be proved.

References

[1] P. K. Agarwal, J.-C. Latombe, R. Motwani and P. Raghavan, Nonholonomic Path Planning for Pushing a Disk Among Obstacles, *Proc. IEEE International Conference on Robotics and Automation,* 1997

[2] H. Arai and O. Khatib, Experiments with Dynamic Skills, *Proc. of 1994 Japan-USA Symposium on Flexible Automation, 81-84,* 1994

[3] B. Aronov, M. De Berg, A. F. Van der Stappen, P. Ŝvestka, and J. Vleugels. Motion Planning for Multiple Robots, *Discrete and Computational Geometry, 22:505-525,* 1999

[4] I. L. Balaban, An Optimal Algorithm for Finding Segment Intersections, *Proc. 11th Annual ACM Symposium Computational Geometry 211-219,* 1995

[5] J. L. Bentley and T. A. Ottmann, Algorithms for Reporting and Counting Geometric Intersections, *IEEE Trans. Computing, C-28:643-647,* 1979

[6] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry, *2nd ed. Springer-Verlag, Berlin, Germany, 272-276,* 2000

[7] R.-P. Berretty, K. Y. Goldberg, M. H. Overmars, A. F. van der Stappen, Algorithms for Fence Design, *Robotics, the Algorithmic Perspective, 279-295, A.K. Peters* 1998

[8] A. J. Briggs, An Efficient Algorithm for One-Step Planar Compliant Motion Planning with Uncertainty, *Algorithmica, 8(3):195-208,* 1992

[9] R. Brost, Automatic Grasp Planning in Presence of Uncertainty, *Int. Journal of Robotics Research, 7(1):3-17,* 1988

[10] B. R. Donald, The Complexity of Planar Compliant Motion Planning under Uncertainty, *Proc. ACM Symposium on Computational Geometry,* June 1988

[11] M. A. Erdmann, On Motion Planning with Uncertainty, *Technical Report AITR-810 MIT Artificial Intelligence Laboratory,* 1984

[12] M. A. Erdmann, M. T. Mason, An Exploration of Sensorless Manipulation, *IEEE Journal of Robotics and Automation, 4:367-379,* 1988

[13] K. Goldberg, Orienting Polygonal Parts without Sensors, *Algorithmica, 10(2):201-225,* 1993

[14] L. Kavraki, P. Ŝvestka, J.-C. Latombe and M. H. Overmars, Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Space, *IEEE Trans. on Robotics and Automation, 12(4):566-580,* 1996

[15] V. Koltun, Segment Intersection Searching Problems in General Settings, *Proceedings of the seventeenth annual symposium on Computational geometry 197-206,* 2001

[16] T. Lozano-Peréz, M. T. Mason and R. H. Taylor, Automatic Synthesis of Fine-Motion Strategies for Robots *IEEE Trans. on Computers (C-32),108-120,* 1983

[17] K. Lynch, Nonprehensile Robotic Manipulation: Controllability and Planning, *PhD Thesis, The Robotics Institute, Carnegie Mellon University,* 1996

[18] K. M. Lynch and M. T. Mason, Stable pushing: Mechanics, Controllability, and Planning, *Int. Journal of Robotics Research, 15(6):533-556,* 1996

[19] M. T. Mason, Mechanics of Robotic Manipulation, *MIT Press,* 2001

[20] M. T. Mason, Mechanics and Planning of Manipulator Pushing Operations, *Int. Journal of Robotics Research, 5(3):53-71,* 1986

[21] M. T. Mason, K. M. Lynch, Dynamic Manipulation, *Proc. IEEE/RJS International Conference Intelligent Robots and Systems 152-159,* 1993

[22] M. Sharir and S. Sifrony, Coordinated Motion Planning for Two Independent Robots, *Proc. of the fourth annual symposium on Computational Geometry 319-328,* 1988

[23] A. F. van der Stappen, M. H. Overmars, M. de Berg and J. Vleugels, Motion Planning in Environments with Low Obstacle Density, *Discrete & Computational Geometry, 20:561-578,* 1998