Motion Planning for Vacuum Cleaner Robots*

(Extended Abstract)

Chantal Wentink† Otfried Schwarzkopf†

Abstract

A "vacuum cleaner robot" is a polyhedral robot in a three-dimensional, polyhedral environment that can translate in a horizontal plane and rotate around a vertical axis. Generalizing a previous result for the planar motion planning problem, it is shown that the motion planning problem for a k-faced vacuum cleaner robot in an environment of total complexity n can be solved in time $O(k^3n^3 \log kn)$.

1 Introduction

The planar motion planning problem considers a polygonal robot that is free to translate and rotate in a planar polygonal environment. This is usually considered a good model for realistic robots, even in our three-dimensional universe, since typical autonomous robots are restricted to move on a planar surface, and many environments—a factory floor, for instance—can be described well using a planar floor map.

Nevertheless, there are situations where this model is too crude. Imagine a robot vacuum cleaner that moves autonomously in an office. It may easily pass underneath a chair that is standing in the office, but it will not be able to pass it if the same chair has been tossed over. Or imagine a robot carrying a bulk object that sticks out beyond the robot. The robot may be able to carry its load over little obstacles lying in its way, even if it cannot pass these obstacles with its own body. In these situations the fact that robot and obstacles are living in a three-dimensional world cannot be ignored, even if the robot is restricted to stay on a two-dimensional surface of that world.

In this paper we will consider this situation, the case of a *vacuum cleaner robot*. In particular, we will show how an algorithm for planar motion planning by Avnaim et al. [1, 2] can be generalized to do motion planning for a vacuum cleaner robot.

The planar motion planning problem can be defined as follows: given a robot B, a simple polygon with k edges, a polygonal environment E consisting of n line segments, and a start and goal placement for the robot within this environment, find a continuous motion connecting the start and goal position during which the interior of B does not intersect any element of E. This problem had only been studied from a practical point of view when Schwartz and Sharir [6] gave the first exact solution, using a very general, but rather involved algorithm. For the case that B is a line segment (a "ladder") the algorithm takes time $O(n^5)$. For *convex* robots B Kedem and Sharir [3] gave an algorithm of running time about $O(n^2k^2)$. Finally, for the case of a general robot B, Avnaim and Boissonnat [1] gave an $O(n^3k^3 \log(kn))$ algorithm to compute the space of all legal configurations of the robot, and Avnaim et al. [2] showed how to compute a motion for the robot in time $O(n^3k^3)$ once this configuration space is known. (More results on exact and approximate motion planning algorithms can be found in Latombe's book [4].)

This is the algorithm we will generalize in the present paper. We start with a robot B, a k-faced polyhedron moving in a three-dimensional space cluttered with polyhedral obstacles with total complexity n. The robot has three degrees of freedom: It can translate in a fixed plane—for convenience, we will assume that this is the xy-plane—and it can rotate around the z-axis. We will show that the *configuration space* of this robot—the space of all placements where it does not intersect any obstacle—has complexity $O(n^3k^3)$, just like in the planar situation, and that it can be computed within $O(k^3n^3 \log(kn))$, using a generalization of Avnaim and Boissonnat's algorithm. Once the configuration space has been computed, the technique by Avnaim et al. can once again be used to compute a path connecting two given placements in time $O(k^3n^3)$.

While the planar motion planning problem becomes easier by one order of magnitude when the robot B is convex, this seems not to be true for the case of the vacuum cleaner robot. We give an example of a convex robot B with $k = 4$ and n point obstacles that induce a free configuration space of complexity $\Theta(n^3)$. Together

*This research was supported by the Netherlands' Organization for Scientific Research (NWO) and partially by the ESPRIT III Basic Research Action 6546 (PROMotion).

†Vakgroep Informatica, Universiteit Utrecht, Postbus 80.089, 3508 TB Utrecht, the Netherlands. E-mail: {chantal,otfried}@cs.ruu.nl

with a two-dimensional lower bound, we thus have a lower bound of $\Omega(n^3 + n^2k^2)$ and an upper bound of $O(n^3k^3)$ on the complexity of the free space. We leave it as an open problem to tighten or close this gap.

2 Definitions and problem setting

Let B be a rigid k-faced polyhedron (the robot), moving in a three-dimensional space, amidst a number of non-moving polyhedra (the obstacles), $A_1, A_2, ..., A_M$, having n faces altogether. (We assume that all polyhedra are already triangulated, so the total complexity of the obstacles is $O(n)$.)

The robot's movements are confined to translations in the x- and y-directions and rotations around an axis parallel to the z-axis. This means, for instance, that a suitable reference point b for B remains in the xy-plane. We will call this type of robot a *vacuum cleaner robot.* As in the planar case, we can describe any placement of the robot using three parameters (x, y, θ)—the first two specify the position (x, y) of the reference point in the xy-plane, while θ specifies the orientation with respect to some standard orientation. The *free configuration space* FP is the set of all placements (x, y, θ) such that the interior of B does not intersect any of the obstacles. (Note that our definition allows motions where the boundary of the robot touches the obstacles.)

In this paper we will describe how to compute FP in *time $O(n^3k^3 \log kn)$. Once FP is given, the algorithm* by Avnaim et al. [2] can be used to actually compute a motion of the robot in time $O(n^3k^3)$.

The boundary of the free configuration space FP consists of placements where the robot is *in contact* with an obstacle. We distinguish three different types of contacts: face-vertex contacts (f, v) between a face f of the robot and a vertex v of an obstacle; vertex-face contacts (v, f) between a vertex v of the robot and a face of an obstacle; and edge-edge contacts (e_B, e_A) between a robot edge e_B and an obstacle edge e_A.

A *contact placement* associated with a certain contact C is a placement (x, y, θ) that satisfies one of the following conditions. (Here, $T(x, y)$ denotes a translation by vector (x, y), and R_θ denotes a rotation around the z-axis with angle θ.)

- $v \in T(x, y) \circ R_\theta(f)$ when $C = (f, v)$ is a face-vertex contact;
- $T(x, y) \circ R_\theta(v) \in f$ when $C = (v, f)$ is a vertex-face contact;
- $T(x, y) \circ R_\theta(e_B) \cap e_A \neq \emptyset$ when $C = (e_A, e_B)$ is an edge-edge contact.

A contact placement is said to be free if in addition it satisfies

$$T(x, y) \circ R_\theta(\text{int } B) \cap \bigcup_i A_i = \emptyset \qquad (1)$$

This means that no other part of the robot intersects with an obstacle feature at this contact placement. A contact region associated with a contact C is the closure of the set of all free contact placements associated with C. It is denoted by $R(C)$. These contact regions clearly contain the boundary of the free space FP. However, there are some contacts whose contact region is not on the boundary of FP, but lie in the interior of the free space. These are the face-vertex and vertex-face contacts where the face is horizontal, as well as the edge-edge contacts where both edges are horizontal. Although there may be free contact placements realizing such a contact, these placements are not necessarily on the boundary of FP, since there is no movement of the robot which turns the touch into a penetration, see Figure 1: a slight move of the robot (which could be both of the shown parts) does not result in an intersection between the robot and the obstacle. In the following, we will simply ignore all such contacts: a contact is a contact only if not both features are horizontal. Under that condition, we immediately have

Proposition 1 $\partial FP = \bigcup_C R(C)$

Accordingly, we can concentrate on the computation of $R(C)$ for all possible contacts. Later we will have to take the union over all these regions to compute a boundary representation of FP.

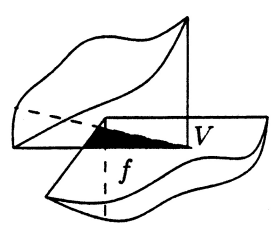

Figure 1: A contact placement not on the boundary of FP.

3 The algorithm

The algorithm presented here is a generalisation of the 2-dimensional algorithm as described by Avnaim et al. [1, 2]. First we give an exact description of the non-free placements of the robot and a description of the edges describing the contact placements. Then we intersect the edges describing the contact placements with the polygons describing an intersection between two faces to

obtain the contact regions. Finally the adjacency relationships between the resulting contact regions are produced. The result immediately gives us a description of the free configuration space, according to Proposition 1.

3.1 Non-free placements

First we want to give a description of the non-free configurations. These are exactly the configurations implying a face-face intersection. We make the following observation.

Observation 2 *There is an intersection between two polygons P_1 and P_2 in space if and only if either an edge of P_1 is intersected by P_2 or an edge of P_2 is intersected by P_1.*

When we want to describe the intersection configurations of an edge with a (triangular) face, the part of the face which needs to be considered is obtained by intersecting it with the planes $z = z_A$ and $z = z_B$ with z_A and z_B the z-coordinates of the vertices describing the edge involved. For simplicity, we divide the resulting polygon into triangles, having one horizontal edge each. Thus the problem is reduced to describing the intersection between an edge and a triangle having one horizontal edge.

Observation 3 *The configurations implying an edge-face intersection between an edge e and a face f are bounded by the configurations implying an intersection between e and the edges describing f.*

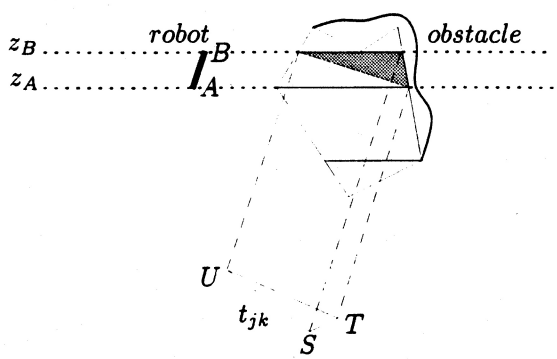

Figure 2: Projection of an obstacle face.

This means that we can describe the non-free placements by giving a description of three edge-edge intersections. The three segments describing the configurations resulting in edge-edge contacts form a triangle in the plane, which is the projection of the face involved in the direction of the line containing the edge of the robot e. This is illustrated in Figure 2. We construct such a triangle t_{jk} for all combinations of edges j and faces k of the robot and the obstacles.

3.2 Edge-edge intersection

Assume we have an edge AB of the robot and an edge EE' of an obstacle. We want a description of the contact placements. l is the line through AB.

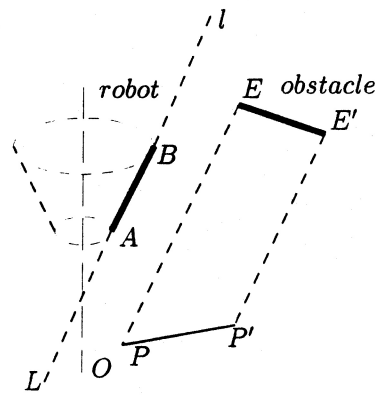

Figure 3: An edge edge intersection.

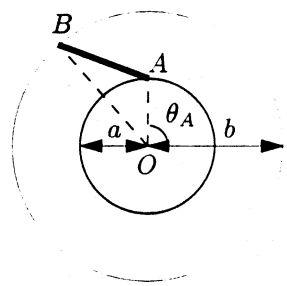

Figure 4: Top view on edge AB.

See Figure 3 and Figure 4 for a sketch of the situation and variable names. It is clear that the coordinates of A and B depend on the robots orientation θ, while E and E' do not. When θ_A is A's initial orientation and θ_B is B's initial orientation, the coordinates of A and B are given by

$$A = (a\cos(\theta + \theta_A), a\sin(\theta + \theta_A), z_A) \qquad (2)$$

$$B = (b\cos(\theta + \theta_B), b\sin(\theta + \theta_B), z_B) \qquad (3)$$

What we need is the projection of obstacle edge EE' onto the xy-plane in the direction of l. This is done by computing the intersection points of the lines parallel to l through E and E'. The resulting points are the endpoints of the segment, which describes all translations of L, the intersection point of l with the xy-plane, resulting in a contact between AB and EE'. This segment is then translated by the vector $\overrightarrow{L_\theta O}$, because we want a description of the configurations of the reference point, which was originally in O. It can be shown that the intersection points are:

$$P = (x_E + \lambda\,(b\cos(\theta+\theta_B) - a\cos(\theta+\theta_A)), y_E + \lambda\,(b\sin(\theta+\theta_B) - a\sin(\theta+\theta_A))) \text{ and}$$
$$P' = (x_{E'} + \mu\,(b\cos(\theta+\theta_B) - a\cos(\theta+\theta_A)), y_{E'} + \mu\,(b\sin(\theta+\theta_B) - a\sin(\theta+\theta_A))).$$

Note that $a, b, x_E, y_E, z_E, x_{E'}, y_{E'}, z_{E'}, \theta_A, \theta_B, z_A, z_B, \tilde{\lambda}$ and $\tilde{\mu}$ are constants and do not dependent on θ. The equation of the line PP' can now be written as

$$cs(\theta)X - cs(\theta)Y + cs(\theta)cs(\theta) = 0, \qquad (4)$$

where the $cs(\theta)$ are expressions of type $\alpha\cos(\theta) + \beta\sin(\theta) + \gamma$ (and different occurrences of $cs(\theta)$ denote different such expressions).

Translation of PP' in the xy-plane over the vector $\overrightarrow{L_\theta O}$ does not change the structure of the equation. When ST and TU are the projections of the non-horizontal edges as in Figure 2, we can make the following claim.

Proposition 4 *The equations of lines ST and TU are given by* $cs(\theta)X - cs(\theta)Y + cs(\theta)cs(\theta) = 0$.

Because US describes contact placements involving a horizontal obstacle edge, the equation of this line is different from ST and TU. The point on AB touching the horizontal edge EE' is always the same, because there is only one point on AB at the same height as EE'.

Proposition 5 *The equation of line US is given by* $aX - Y + cs(\theta) = 0$.

3.3 Contact placements

In order to compute the contact regions, we have to describe the contact placements associated with the diverse contacts C. The possible contacts in three dimensions are vertex-face, face-vertex and edge-edge contacts. As we saw before all contact placements associated with an edge-edge contact are on the segment $PP' = T(\overrightarrow{L_\theta O}) \circ p_{l_\theta}(EE')$ with $p_{l_\theta}(EE')$ the projection of EE' onto the xy-plane in the direction of l. The line through P and P' is described by the equation in Proposition 4.

The segment, describing all contact placements associated with a vertex-face contact (v, f) is obtained by first intersecting f with the plane $z = z_v$. Then the contacts between the resulting horizontal edge and v can be described in a similar way as the edge-edge contact placements, involving one horizontal edge. This leads us to an equation like the one in Proposition 5, for $PP' = T(\overrightarrow{v_\theta O})(EE')$.

A face-vertex contact (f, v) can be treated in a similar way as a vertex-face contact. The only thing that should be done in addition is symmetry in O: $PP' = (T(\overrightarrow{V_\theta O})(EE'))^*$. The equation for PP' is an expression like the one in Proposition 5.

We refer to the segments as described above as PP'. One such segment is associated with each possible contact C. Having an exact description of the contact placements and the non-free placements of the robot, we are able to compute the contact regions, associated with the different contacts.

3.4 Computation of the contact regions

As shown before we need all the contact regions, one per contact, to compute the boundary of the free space (see Proposition 1). A contact region can be computed as follows:

$$R(C) = PP'(\theta) \backslash \bigcup_{jk} t_{jk}(\theta) \qquad (5)$$

for all triangles t_{jk} as computed in section 3.1, j and k identifying the edge and face involved in contact C.

$PP'(\theta)$ is the segment describing all contact placements associated with contact C. Triangle $t_{jk}(\theta)$, describing non-free placements, intersects $PP'(\theta)$ in points that depend on θ. We call the intersection points $L_{jk}(\theta)$ and $M_{jk}(\theta)$. This is illustrated in Figure 5.

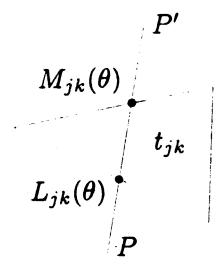

Figure 5: PP' intersecting a non free section.

The set theoretic difference between a segment and a family of triangles is a family of segments $S_1 \ldots S_r$. Each endpoint of S_i is either a point $L_{jk}(\theta)$ or a point $M_{jk}(\theta)$ for some j, k. In order to compute the free segments of $PP'(\theta)$, we associate two functions to each triangle t_{jk}. A function $\lambda(\theta)$ which defines the intersection point closest to P and $\mu(\theta)$ defining the intersection point closest to P', where the segment PP' is scaled to size 1, so when for example $\lambda(\theta) = 0.5$, the intersection point of t_{jk} with PP' closest to P is in the middle of the segment.

So, λ and μ range from 0 to 1. These two functions enable us to get a two-dimensional representation of a contact region. Using Propositions 4 and 5 it can be shown that

Proposition 6 *Let* $\lambda(\theta)$ *and* $\mu(\theta)$ *be the functions associated to triangle* $t(\theta)$ *and edge* PP'.

1. *$\lambda(\theta)$ and $\mu(\theta)$ are defined on the same finite set of sub-intervals of $[0, 2\pi)$*

2. *On each of these sub-intervals, $\lambda(\theta)$ (resp.$\mu(\theta)$) is an expression of the following type*

$$\frac{cs(\theta)cs(\theta)cs(\theta)}{cs(\theta)cs(\theta)cs(\theta)}$$

3. *$\lambda(\theta)$ and $\mu(\theta)$ can be computed in constant time.*

For each triangle $t_{jk}(\theta)$ and contact C we can compute the associated functions $\lambda_{jk}(\theta)$ and $\mu_{jk}(\theta)$. These functions are defined on $[0, 2\pi)$, the range of the functions is $[0, 1]$. For every possible contact C we can draw these functions in a two-dimensional figure. Each point (θ, ϵ) in this planar graph represents a three-dimensional configuration of the robot involving contact C. We have obtained this two-dimensional representation by considering only one contact C at a time. Let ρ_{jk} be the region between two corresponding $\lambda_{jk}(\theta)$ and $\mu_{jk}(\theta)$. ρ_{jk} corresponds to the non-free placements at which contact C is made, but where two faces intersect at the same time.

We define the Φ-region associated to contact C as

$$\Phi(C) = \left([0, 2\pi) \times [0, 1]\right) \setminus \bigcup_{j,k} \rho_{jk}.$$

Each element $(\theta, \epsilon) \in \Phi(C)$ represents a free placement of $R(C)$. Now we have to transform this two-dimensional representation of the free contact placements associated with C into a description in the three-dimensional configuration space. To each element $(\theta, \epsilon) \in \Phi(C)$ corresponds a free placement $(\overrightarrow{OM}, \theta)$ on $R(C)$, such that $\overrightarrow{OM} = \overrightarrow{OP} + \epsilon\overrightarrow{PP'}$. This defines the one-to-one relation:

$$F : (\theta, \epsilon) \rightarrow (x_S, y_S, \theta) \qquad (6)$$

with $x_S = \frac{x_{P'} - x_P}{\epsilon} - x_P$, y_S can be calculated by substitution of the values of x_S and θ in the equation of line PP', because (x_S, y_S) is a translation to PP' at orientation θ.

In order to compute $R(C)$, we will compute $\Phi(C)$ and then apply F. For this purpose we can use a plane sweep algorithm, which is similar to the one described by Ottman [5]. With this algorithm we can compute a subdivision of Φ into ϕ_i's, where ϕ_i is bounded by two curves $\lambda_{jk}(\theta)$ and $\mu_{j'k'}(\theta)$ (keeping a constant analytic form) and possibly one line segment on $\theta = \alpha_i$ and one on $\theta = \alpha_{i+1}$. Where $[\alpha_0, \alpha_1, ..., \alpha_l]$ is the set of orientations either corresponding to an end point of a subinterval where $\lambda_{jk}(\theta)$ and $\mu_{j'k'}(\theta)$ keep a constant analytic form or to the orientation of an intersection point between two such functions.

Only, instead of working with straight line segments, we must consider pieces of curves described by the equations in Proposition 6. Because $\lambda(\theta)$ and $\mu(\theta)$ correspond to double-contacts (one of them being C) an intersection between two such curves implies a triple contact. These intersection points are the event points in the sweep line algorithm. In the full paper [7] it is shown that

Proposition 7 *$\Phi(C)$ consists of a finite (possibly empty) set of regions which can be computed in time $O((kn + t) \log kn)$ where t is the number of triple contacts involving contact C.*

Contact region $R(C)$ is now computed by taking the union of the images $F(\phi_i)$ of all subregions. $F(\phi_i)$ is a ruled surface, which can be computed in constant time from ϕ_i. We call it a *face* of $R(C)$.

Proposition 8 *$R(C)$ consists of a finite (possibly empty) set of faces, which can be computed in time $O((kn + t) \log kn)$, with t the number of triple contacts involving contact C.*

3.5 Computation of the free space

In order to compute a complete description of the boundary of FP (∂FP), we compute the set of all its faces by computing $R(C)$ for all the kn contacts C. As stated in Proposition 8 computing one contact region $R(C)$ takes $O((kn + t) \log kn)$ time, where t is the number of triple contacts involving C. Thus computing all the faces of ∂FP takes $(O(k^2n^2 + T) \log kn)$ time, where T is the total number of triple contacts. An example can be given where $T = \Omega(k^3n^3)$ [1]. In addition, we compute the adjacency relationships between the faces of ∂FP. These can be computed during the plane sweep algorithm as described in the previous subsection. Thus, we obtain the final result:

Proposition 9 *∂FP can be computed in time $O((k^2n^2 + T) \log kn)$ where T is the total number of (not necessarily free) triple contacts. In the worst case $T = \Theta(k^3n^3)$.*

Finally the algorithm described by Avnaim et al. [2] is used to find a free motion in FP. The time needed to find such a motion is $O(k^3n^3)$ in the worst case.

4 Convex obstacles

In most motion planning problems there are some special cases, which can be solved in less time than the general problem. One example of such a special case is the case of convex obstacles and a convex robot. In

the planar case, for example, Kedem and Sharir [3] presented a motion planning algorithm for a convex robot, running in $O(kn\lambda_6(kn)\log kn)$ time, where $\lambda_s(q)$ is an almost linear function of q, the maximum complexity of a (q,s)-Davenport-Schinzel sequence. This is significantly better than the running time of the algorithm by Avnaim and Boissonnat for a general robot. For the vacuum cleaner robot, however, such an improvement does not seem likely. In particular, the complexity of the free configuration space of the robot does not decrease as in the planar case. In the following we give an example of a vacuum cleaner robot where the free configuration space has complexity $\Omega(n^3)$. Even in the planar case, the complexity can be $\Omega(k^2n^2)$, so this gives us a lower bound of $\Omega(k^2n^2+n^3)$ for the complexity of FP for a convex robot. We do not know if the complexity could be $\Omega((kn)^3)$ as in the general case.

As we saw before, the complexity of the free space is determined by the number of triple contacts between the obstacles and the robot. In Figure 6 we give an example where the number of triple contacts is $\Omega(n^3)$. Here is

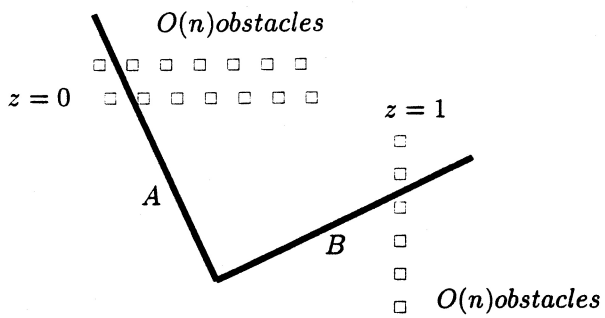

Figure 6: $\Omega(n^3)$ lower bound on triple contacts.

a description of the robot. We have a robot which has one horizontal edge A in the plane $z=0$ and another horizontal edge B in the plane $z=1$. The robot itself is a tetrahedron, the convex hull of these two segments.

There are two rows of $n/3$ obstacles parallel to the x-axis in the plane $z=0$ and one row of $n/3$ obstacles parallel to the y-axis in the plane $z=1$.

Edge A of the robot can now touch every combination of two obstacles, one from each row in the $z=0$ plane simultaneously, by changing its orientation. This way, there are $O(n^2)$ possible double contacts. If the edge A and B are long enough, at every one of these double contacts, all of the $O(n)$ obstacles in the $z=1$ plane can be touched by edge B. These contacts can be made by translation of the robot in the direction of the line through A at a particular orientation, associated with a double contact.

In the mean time no other part of the robot hits or intersects an obstacle, because no part of it, except A and B, is in one of the planes $z=0$ or $z=1$, the planes where the obstacles are.

So, with every $\Omega(n^2)$ double contact in $z=0$, $\Omega(n)$ contacts in $z=1$ are possible. This adds up to $\Omega(n^3)$ triple contacts in total.

References

[1] F. Avnaim and J.-D. Boissonnat. Polygon placement under translation and rotation. In *Proc. 5th Sympos. Theoret. Aspects Comput. Sci.*, volume 294 of *Lecture Notes in Computer Science*, pages 322–333. Springer-Verlag, 1988.

[2] F. Avnaim, J.-D. Boissonnat, and B. Faverjon. A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. In *Proc. 5th IEEE Internat. Conf. Robot. Autom.*, pages 1656–1661, 1988.

[3] K. Kedem and M. Sharir. An efficient motion planning algorithm for a convex rigid polygonal object in 2-dimensional polygonal space. *Discrete Comput. Geom.*, 5:43–75, 1990.

[4] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[5] T. Ottmann, P. Widmayer, and D. Wood. A fast algorithm for Boolean mask operations. *Comput. Vision Graph. Image Process.*, 30:249–268, 1985.

[6] J. T. Schwartz and M. Sharir. On the "piano movers" problem I: the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Commun. Pure Appl. Math.*, 36:345–398, 1983.

[7] C. J. Wentink. Motion planning for vacuum cleaners and related problems. Master's thesis, Utrecht University, August 1993.