Reaching a Polygon with Directional Uncertainty[*]

Otfried Cheong[†] René van Oostrum[‡]

Abstract

Assume a robot that, when directed to move in a particular direction, is guaranteed to move inside a cone of angle α centered at the specified direction. The robot has to reach a convex polygonal goal G, while avoiding polygonal obstacles of complexity n. We show that the complexity of the safe region, from where the robot can reach the goal with a single linear motion with uncertainty α, is $O(m+n)$, and can be computed in time $O((m+n)\log(m+n))$, if α is assumed constant.

1 Introduction

Most motion planning algorithms in the literature assume that the robot has perfect control over its motion: if the robot is commanded to move in a certain direction, these algorithms expect it to move exactly in the specified direction. In practice, however, a robot has to deal with uncertainty in the execution of its commanded motions.

A model for dealing with directional uncertainty was first proposed by Lozano-Pérez, Mason and Taylor [6] and was further developed by Erdmann [4]. A detailed discussion can be found in Latombe's book [5].

We follow the approach of de Berg et al. [2]. They assume a point robot—larger robots can be treated using Minkowski sum techniques—that wishes to reach a polygonal goal region while avoiding polygonal obstacles. It has to do so with a *directional uncertainty* α: Whenever a motion in a certain direction is commanded, we assume that the robot follows a straight path from its current position that falls into a cone of angle α centered around the commanded direction. We are interested in the complexity and computation of the *safe region*, that is the set of all points from which the robot can reach the goal without colliding with an obstacle.

De Berg et al. did not treat the problem in its full generality: They gave bounds and algorithms for two special cases: In the first situation, they assumed a single goal region, the "region at infinity", and a set of polygonal obstacles of complexity n. They showed that the complexity of the safe region is bounded by $O(n/\alpha^5)$. In the second situation, they considered a goal region consisting of a collection of k polygonal goal regions of total complexity m, but without any obstacles. The complexity of the region from which some goal region can be reached is in this case bounded by $O(mk^3)$.

We generalize their results by considering a convex polygonal goal G with m edges, and a set of polygonal obstacles of complexity n. We show that the complexity of the safe region is still linearly bounded if α is constant, namely by $O((m+n)/\alpha^5)$. The safe region can be computed in time $O((m+n)/\alpha^5 \log(m+n))$. Our proofs and algorithm are based on the ideas of de Berg et al., but are technically more complex.

We also consider how our results could be extended to non-convex goal regions. We give some evidence that we believe indicates that such a generalization no longer models directional uncertainty in robot motion planning in a useful way.

[*]This research was supported by HKUST Direct Allocation Grant 97/98.EG15, and partially by the ESPRIT IV LTR Project No. 21957 (CGAL).

[†]Dept. of Computer Science, Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong. Email: otfried@cs.ust.hk

[‡]Dept. of Computer Science, Utrecht University, Postbus 80.089, 3508 TB Utrecht, the Netherlands. Email: rene@cs.uu.nl

2 Definitions

We are given a convex goal polygon G of complexity m and a set S of obstacles. A point y in the plane is said to be *visible* from a point x if and only if the segment $\overline{xy}$ doesn't intersect the goal polygon or S (both considered topologically closed sets).

An α-cone is a cone of angle α. A point x in the plane is *safe* if an α-cone γ with apex x exists, such that all rays with origin x in the interior of the cone intersect the goal polygon G, and no such ray intersects S in a point that is visible from x. The region of all safe points is denoted by $\mathcal{R}_\alpha(G,S)$. This is the region from which a point robot can reach the goal polygon G in a single linear motion with directional uncertainty α, without hitting any obstacle. We are interested in the *complexity* of $\mathcal{R}_\alpha(G,S)$, which is defined as the number of edges and vertices of the boundary of $\mathcal{R}_\alpha(G,S)$.

We will model the obstacle set S as a set of n line segments. The line segments do not intersect in their interiors, but their endpoints may coincide; in this way, we can model obstacle polygons as well.

To simplify the notation, we will use P to denote the set of at most $2n$ endpoints of segments in S, and use V to denote the set of vertices of G throughout.

Observation 1 (de Berg et al. [2]) *A point on the boundary of $\mathcal{R}_\alpha(G,S)$ is either on a segment of S or on an edge of G, or is the apex of an α-cone γ that has two points $p,q \in P \cup V$ on its left and right rays. The apices of all cones that are determined by two fixed points p and q form two circular arcs. This implies that $\mathcal{R}_\alpha(G,S)$ is bounded by circular arcs and straight line segments.*

3 Combinatorial bounds

We start by establishing a simple $O((m+n)^4)$ upper bound on the complexity of $\mathcal{R}_\alpha(G,S)$ and we give a lower bound example of $\Omega(m+n^4)$. For the lower bound example, α has to be made very small if n grows. This motivates us to give a bound on the complexity of $\mathcal{R}_\alpha(G,S)$ in terms of α. We can prove a bound of $O((m+n)/\alpha^5)$, which implies that if α is bounded from below by a constant, then the complexity of $\mathcal{R}_\alpha(G,S)$ is only linear in n and m. We will prove this by first considering a directed version of the problem: for a fixed direction $\vec{u}$ we look at $\mathcal{R}_{\alpha,\vec{u}}(G,S)$, the region of all points x for which a safe α-cone exists that contains direction $\vec{u}$. Then we return to proving bounds on the complexity of $\mathcal{R}_\alpha(G,S)$ by merging a collection of directed safe regions. We will further simplify the problem for the directed version $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ by looking only at the safe region $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ induced by the set of endpoints P. We will show that its complexity is linear in m and n and that $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ is of the same complexity.

Theorem 1 *Given a convex polygon G of complexity m, a set S of n disjoint line segments, and an angle $\alpha < \pi$, the complexity of $\mathcal{R}_\alpha(G,S)$ is bounded by $O((m+n)^4)$. Furthermore, for every m and n there is a convex goal polygon G, a set S of n line segments and an angle $\alpha > 0$ (which decreases with n) such that $\mathcal{R}_\alpha(G,S)$ has complexity $\Omega(m+n^4)$.*

Proof: The circular arcs on the boundary of $\mathcal{R}_\alpha(G,S)$ lie on $O((m+n)^2)$ circles, since we can choose $O((m+n)^2)$ pairs of points from the endpoints of the segments in S and the vertices of G. A vertex of $\mathcal{R}_\alpha(G,S)$ is the intersection of two such circles or between such a circle and either a segment of S or an edge of G. It follows that the complexity of $\mathcal{R}_\alpha(G,S)$ is at most $O((m+n)^4)$.

Every regular polygon G with m vertices and every $\alpha > 0$ give rise to a lower bound example of $\Omega(m)$ (Figure 1, left): the vertices of the safe region lie on the supporting lines of the edges of G, and it's not difficult to verify that the number of vertices of the safe region is either m or $2m$. An $\Omega(n^4)$ lower bound example was given by de Berg et al. For completeness, we include the example here (Figure 1, right): We construct two rectangular obstacles with parallel walls, and poke $O(n)$ very small holes in the right walls of the rectangles. This creates $\Theta(n^2)$ thin safe regions in the inner rectangle: every pair of holes determines a "safe ray" if we position the rectangles near a large regular polygon of complexity m. We poke holes in the bottom walls in the same way. This gives us an arrangement of $\Theta(n^2)$ "safe rays" in the inner rectangle, resulting in a total complexity of $\Theta(n^4)$. $\square$

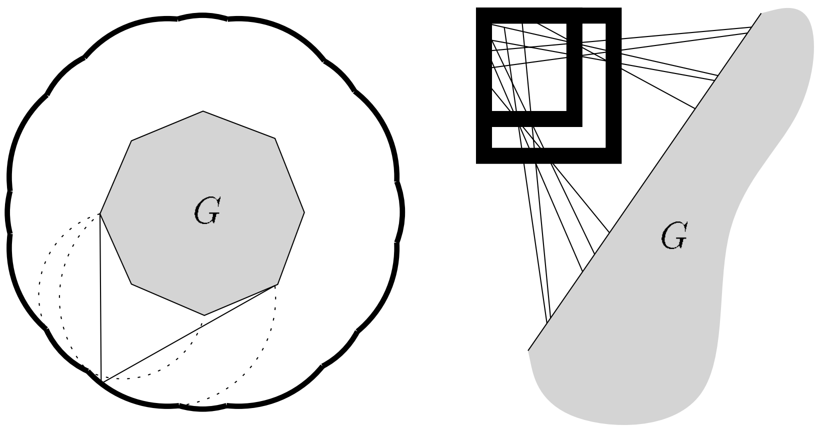

Figure 1: Lower bound example of $\Omega(m + n^4)$.

To construct the $\Theta(n^4)$ arrangement in the lower bound example we have to use a value of α that decreases very fast when n grows. For all practical purposes, it seems more useful to give bounds for the case where α is not allowed to vary with n. In the following, we will give an analysis of the complexity of $\mathcal{R}_\alpha(G, S)$ in terms of all n, m, and α. We are most interested in the case where α is a constant.

Closely following the work of de Berg et al. [2] we decompose the problem by considering first the following directed version: Let $\vec{u}$ be a direction vector, and let $\mathcal{R}_{\alpha,\vec{u}}(G, S)$ be the region of all points $x \in \mathbb{E}^2$ for which there exists a safe α-cone γ with apex x such that the ray with origin x and direction $\vec{u}$ lies in the closure of γ. To simplify the description, we assume that $\vec{u}$ is the upward vertical direction, that is the positive y-direction. We will show that the complexity of $\mathcal{R}_{\alpha,\vec{u}}(G, S)$ is $O(m + n)$.

Lemma 1 *The lower boundary β of $\mathcal{R}_{\alpha,\vec{u}}(G, S)$ is a chain with the property that its intersection with any vertical lineis a point or a segment).*

Proof: Suppose that α-cone γ with apex x on the lower boundary of $\mathcal{R}_{\alpha,\vec{u}}(G, S)$ is safe. Then a cone γ' with rays parallel to those of γ and apex x' on the vertical segment between x and G is also safe: Clearly, all rays with origin x' in γ' intersect G. If such a ray intersected a segment in S in a point s visible from x', then the ray with origin x through s, which is contained in γ, is also unsafe, a contradiction. ◻

We will call a chain with the property of Lemma 1 *semi-monotone*.

As we have seen before, the lower boundary β of $\mathcal{R}_{\alpha,\vec{u}}(G, S)$ consists of circular arcs (determined by two endpoints of S, two vertices of G or one endpoint of S and one vertex of G), line segments (edges of G and pieces of the segments of S), and vertical segments (below an endpoint of a segment of S).

We further break up the problem by considering only the set P of endpoints of the segments in S, and studying $\mathcal{R}_{\alpha,\vec{u}}(G, P)$ first. It is clear that the boundary of $\mathcal{R}_{\alpha,\vec{u}}(G, P)$ is confined to the vertical slab between the two vertical tangents of G: no safe α-cone with apex x outside this slab can contain the upward direction $\vec{u}$, since both rays of such a cone will have to hit G (Figure 2). It is also clear that an obstacle point p outside this slab cannot define a safe α-cone with its apex inside this slab: either p is obscured by G, or some of the rays in the α-cone will miss G. Furthermore, obstacle points above G cannot define safe α-cones. Therefore, we only need to consider points in P lying below the lower envelope of G.

To be able to bound the complexity of $\mathcal{R}_{\alpha,\vec{u}}(G, S)$ by the complexity of $\mathcal{R}_{\alpha,\vec{u}}(G, P)$ we state the following lemma. Again this is a slight generalization of a result by de Berg et al. [2]

Lemma 2 *$\mathcal{R}_{\alpha,\vec{u}}(G, S)$ is the intersection of $\mathcal{R}_{\alpha,\vec{u}}(G, P)$ with the region above the upper envelope of S.*

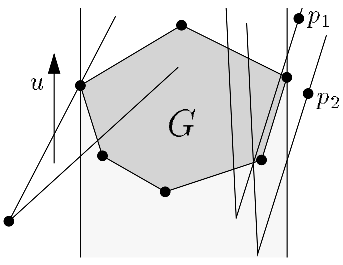

Figure 2: Impossible cones and obstacle points

Proof: Since P is the set of endpoints of the segments in S we have $\mathcal{R}_{\alpha,\vec{u}}(G,S) \subset \mathcal{R}_{\alpha,\vec{u}}(G,P)$. The upward vertical ray from any point below the lower envelope of S intersects a segment in S, so it cannot be the apex of a safe α-cone containing the vertical direction. Hence $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ must lie above the upper envelope of S.

On the other hand, consider a point x in the intersection of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ with the region above the upper envelope of S. Since $x \in \mathcal{R}_{\alpha,\vec{u}}(G,P)$ there is a α-cone γ with apex x that is safe with respect to G and does not contain any visible endpoint of a segment in S in its interior. Moreover, since x lies above the upper envelope of S and γ contains the vertical direction, no segment in S can completely cross γ. Hence, γ is safe with respect to G and S. ❏

Lemma 3 *The complexity of $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ is $O(m+n)$.*

Proof: Since the upper envelope of S has complexity n and both the upper envelope of S and $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ are semi-monotone, this follows from Lemma 2 if we can prove that the complexity of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ is $O(m+n)$.

The upper boundary of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ is the lower boundary of G; its complexity is $O(m)$. The lower boundary of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ consists of circular arcs and vertical segments. A vertex of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ on its lower boundary either lies below a point of P—there are at most $4n$ such vertices, namely two for each point of P—or is the apex of an α-cone with at least three points of $P \cup V$ on its bounding rays. We proceed to count the vertices defined by cones with at least two points of $P \cup V$ on their right ray—the case of vertices with at least two points on the left ray are counted symmetrically.

So consider a vertex x of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ defined by an α-cone γ with two or more points on its right ray. We call the point closest to x p_1, the next closest point p_2.

Let's first assume that $p_1 \in V$. Note that cones where $p_2 \in P$ do not form vertices of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$, since p_2 is not visible from x (see Figure 3, right). Hence $p_2 \in V$ (see Figure 3, left). The number of such cones is at most m, since p_1 and p_2 are then adjacent vertices of G.

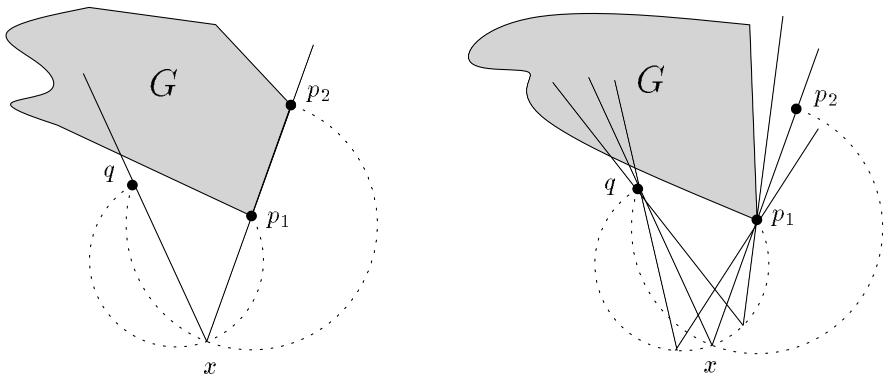

Figure 3: α-cones with two points on their right rays

Having treated this case, we can from now on assume that $p_1 \in P$. We show that a point $p \in P$ cannot play the role of p_1 for more than one vertex of $\mathcal{R}_{\alpha,\vec{u}}(G, P)$.

For a contradiction, suppose that there are two distinct α-cones γ_1, γ_2 with apices x_1, x_2, both of with have an obstacle point $p \in P$ as the leftmost point on their right ray, and another obstacle point or goal vertex q_1 and q_2 as the second point on the right ray.

Assume first that $q_1 \neq q_2$. Then either γ_1 contains q_2 or γ_2 contains q_1 in its interior. By chosing the notation accordingly, we assume the first case. Figure 4, shows the two possible configurations.

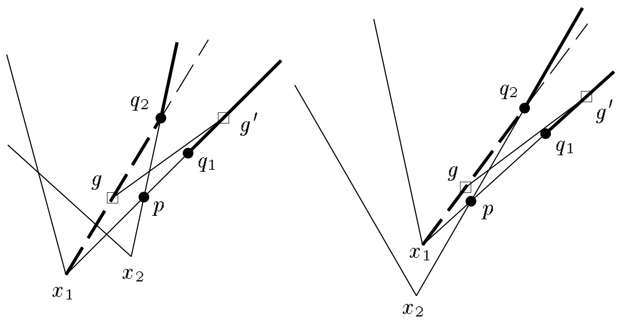

Figure 4: α-cones sharing the leftmost point on their right rays

Since γ_1 is a safe cone, this is only possible if q_2 is not visible from x_1, which means that the goal polygon G must intersect the line segment $\overline{x_1 q_2}$. Let g be a point in this intersection. Furthermore, the rightmost ray of the (safe) cone with apex x_1 intersects G, while the segment $\overline{x_1 q_1}$ does not meet G. Let g' be a point in the intersection of G and the ray with origin q_1 and direction $\vec{x_1 q_1}$. By convexity of G, the segment $\overline{gg'}$ is contained entirely in G. But this segment intersects the segment $\overline{x_2 q_2}$, which means that p_2 is not visible from x_2, hence x_2 cannot be a vertex of $\mathcal{R}_{\alpha,\vec{u}}(G, P)$ defined by q_2 (amongst others).

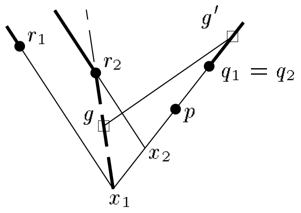

Figure 5: Two α-cones defined by the two same points on their right ray

It remains to consider the possibility that $q_1 = q_2$, see Figure 5. Denote the point on the left ray of γ_1 and γ_2 by r_1 and r_2. Again without loss of generality, we can assume that r_2 lies in the interior of γ_1. This is only possible if r_2 is not visible from x_1, so there is a point $g \in G$ on the segment $x_1 r_2$. Similar to the above, there's another point $g' \in G$ on the common right ray of γ_1 and γ_2, and the segment gg' blocks r_2 from being visible from x_2, and hence from being a defining point of the cone γ_2.

It follows that there are at most $O(m + n)$ vertices of $\mathcal{R}_{\alpha,\vec{u}}(G, P)$ that are apices of α-cones with at least three points on their bounding rays. This proves Lemma 3. ❑

Lemma 3 states that the complexity of $\mathcal{R}_{\alpha,\vec{u}}(G, S)$, a directed version of the safe region $\mathcal{R}_\alpha(G, S)$, is linear in m and n. The undirected safe region, $\mathcal{R}_\alpha(G, S)$, can be described as the union of $O(1/\alpha)$ directed safe regions (by chosing $O(1/\alpha)$ evenly spread directions). We need to bound the complexity of this union.

To prove an upper bound on the complexity of $\mathcal{R}_\alpha(G, S)$, we exploit this lemma and the results and techniques of de Berg et al. [2]. Observe that there is a collection U of $O(1/\alpha)$ different directions such that

$$\mathcal{R}_\alpha(G,S) = \bigcup_{\vec{u}\in U} \mathcal{R}_{\alpha,\vec{u}}(G,S).$$

Next we note that any vertex of $\mathcal{R}_\alpha(G,S)$ is a vertex of $\mathcal{R}_{\alpha,\vec{u}}(G,S) \cup \mathcal{R}_{\alpha,\vec{v}}(G,S)$ for some pair $\vec{u}, \vec{v}$ in U. We will show that the complexity of such a union is $O((m+n)/\alpha^3)$. This will prove an upper bound of $O((m+n)/\alpha^5)$ on the complexity of $\mathcal{R}_\alpha(G,S)$, since there are $O(1/\alpha^2)$ possible pairs of $\vec{u}$ and $\vec{v}$.

To prove that the complexity of $\mathcal{R}_{\alpha,\vec{u}}(G,S) \cup \mathcal{R}_{\alpha,\vec{v}}(G,S)$ for two fixed directions $\vec{u}$ and $\vec{v}$ is bounded by $O((m+n)/\alpha^3)$, we can apply the results of de Berg et al. [2] for the situation with a set S of n obstacle segments, and a goal region "at infinity", with some minor modifications. We summarize these results below; details can be found in de Berg et al. [2].

We can approximate the boundary β of $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ by two polygonal chains β' and β'' of the same complexity $O(m+n)$, such that β is enclosed between β' and β''. In the same way, we can approximate the boundary μ of $\mathcal{R}_{\alpha,\vec{v}}(G,S)$ by two polygonal chains μ' and μ''. De Berg et al. [2] have shown that the number of intersections between β and μ is bounded by the number of intersections between two of the chains β', β'', μ' and μ''. The total number of these intersections equals the complexity of all pairwise unions of $\mathcal{R}'_{\alpha,\vec{u}}(G,S)$, $\mathcal{R}''_{\alpha,\vec{u}}(G,S)$, $\mathcal{R}'_{\alpha,\vec{v}}(G,S)$ and $\mathcal{R}''_{\alpha,\vec{v}}(G,S)$, the regions above β', β'', μ' and μ'', respectively. It remains to prove that the complexity of all these pairwise unions is $O(m+n)/\alpha^3)$.

The combination lemma by Edelsbrunner et al. [3] states that the complexity of the union of two polygonal regions is bounded by the individual complexities of the two regions plus the number of holes in the union. Using a technique by van Kreveld [8] it can be shown that the polygonal regions can be covered with $O(n)$ *β-fat triangles*—triangles whose smallest angle is bounded from below by β—with $\beta \geq c\alpha$ for some constant c. Finally, we apply a result of Matoušek et al. [7], which states that the union of n β-fat triangles has at most $O(n/\beta^3)$ holes.

We have thus proven the following theorem:

Theorem 2 *Let G be a convex polygon of complexity m, let S be a set of n disjoint line segments, and let $\alpha < \pi$ be given. The complexity of $\mathcal{R}_\alpha(G,S)$ is bounded by $O((m+n)/\alpha^5)$.*

4 Computing the safe region

In this section we show how to compute $\mathcal{R}_\alpha(G,S)$. Like de Berg et al. [2], we do so in a manner similar to the structure of the proof of Theorem 2.

We will show below that given a convex goal polygon G with m edges, a set of n obstacle points P, and a direction $\vec{u}$, $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ can be computed in time $O((m+n)\log(m+n))$.

To compute $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ for a set of n segments S, we first compute $\mathcal{R}_{\alpha,\vec{u}}(G,P)$, where P is the set of endpoints of S, and then use a simple plane sweep algorithm to intersect $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ with the envelope (in direction $\vec{u}$) of S within the same time bounds.

It remains to combine the $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ for $O(1/\alpha)$ different directions $\vec{u}$. We use a divide and conquer algorithm on the set U of $O(1/\alpha)$ directions to do this, again closely following de Berg et al. [2]. The merging step can be done in time $O((m+n+K)\log(m+n))$ by a standard plane sweep algorithm, where K is the complexity of the merged region. If we denote the number of directions in U by s then we have $K = O(s^2\frac{m+n}{\alpha^3})$, by the combinatorial results of the previous section. We thus obtain the following recursion for $T(s)$, the time for computing the union of $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ for s different direction $\vec{u}$:

$$\begin{aligned} T(1) &= O((m+n)\log(m+n)), \\ T(s) &= 2T(s/2) + O(s^2\frac{m+n}{\alpha^3}\log(m+n)). \end{aligned}$$

which solves to $T(s) = O(s^2\frac{m+n}{\alpha^3}\log(m+n))$. Substituting $s = O(1/\alpha)$ leads to the main theorem.

Theorem 3 *Let G be a convex polygon of complexity m, S a set of n disjoint line segments, and let $\alpha < \pi$ be given. Then $\mathcal{R}_\alpha(G,S)$ can be computed in time $O(((m+n)/\alpha^5)\log(m+n))$.*

It remains to describe how to compute $\mathcal{R}_{\alpha,\vec{u}}(G,P)$. As before, we simplify the exposition by assuming that $\vec{u}$ is the upward vertical direction. We will construct the lower boundary β of $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ from left to the right. This is possible since β is a semi-monotone chain.

Imagine a point x moving along β, taking with it a safe cone $\gamma(x)$ with apex x. It starts in the leftmost point x_0 of β. The cone $\gamma(x_0)$ has a vertical left ray tangent to G in the leftmost vertex g_0, and its right ray passes through a point q that is either in P or a vertex of G. As we start to move x along β, $\gamma(x)$ rotates counterclockwise while keeping contact with g_0 and p. As observed before, x traces a circular arc determined by g_0 and p. The arc ends when a new point q appears on the bounding rays of $\gamma(x)$. The following edge of β is then determined by two different points—if x continues to move along β, $\gamma(x)$ will continue to rotate counterclockwise while maintaining contact with the two new defining points.

This process continues until x reaches a position where $\gamma(x)$ has a vertical right ray and therefore cannot rotate any further without losing the direction property (remember that a safe cone for $\mathcal{R}_{\alpha,\vec{u}}(G,P)$ has to contain the vertical upward ray). This can only happen when the right ray of $\gamma(x)$ is either tangent to G—in that case we have actually reached the end of β—or when the right ray contains an obstacle point $p \in P$. In the latter case, $\gamma(x)$ "wraps around" the point p—in other words, p switches from being the defining point on the right ray of $\gamma(x)$ to being the defining point on the left ray. We can imagine this as follows: We move the cone vertically upwards until its apex is in p. Then we rotate it clockwise, until the right ray hits an obstacle point or becomes tangent to G, or until the left ray becomes vertical. If the left ray becomes vertical, we then have to slide the cone vertically downwards, again until the right ray encounters an obstacle point or becomes tangent to G. The point p is now on the left ray of $\gamma(x)$, and if we let x continue on its journey along β, $\gamma(x)$ will continue to rotate counterclockwise.

Our algorithm constructs β from left to right. In every vertex of the curve, we perform certain queries on G and P that allow us to find the next vertex of the boundary curve. We maintain the safe cone and its two defining points during the process, so that we can easily produce a description of the edges of β.

To simplify the description of the queries, let's define $P_{<x}$ to be the subset of points $p \in P$ lying to the left of the point $x \in \mathbb{R}^2$. Sets $P_{\leq x}$, $P_{>x}$, and $P_{\geq x}$ are defined analogously.

We start by identifying the left endpoint x_0 of β. We can find x_0 by placing an α-cone at the leftmost vertex g_0 of G such that its left ray is vertical, and sliding it downwards until its right ray contains an obstacle point p, or becomes tangent to G in a point g. The two defining points are now g_0 and p (resp. g_0 and g). Note that during the sliding we have to ignore all obstacle points that are not visible from the apex of the resulting cone. We'll call this query a *sliding query*.

Assume now that we have constructed β up to a vertex v. The safe cone $\gamma(x)$, for x slightly to the right of v, is defined by a point $p \in P \cup V$ on its left and a point $q \in P \cup V$ on its right ray.

We start by considering the case where $\gamma(v)$'s right ray is not yet vertical. This means that we can find the next vertex v' of β by rotating $\gamma(x)$ counterclockwise, while keeping the points p and q on its bounding rays. Let's analyze the possible events:

1. Events on the left ray:

   (a) p is an obstacle point, and the left ray of $\gamma(x)$ hits another obstacle point p' (Figure 6a). Note that p' must lie beyond p on the ray. Furthermore, p' must be visible from x' as points behind the goal polygon G cannot play the role of p' (Figure 6b). The new defining points of $\gamma(x')$ will be p' and q.

   We can find p' by performing the following query: *Given a point $p \in P$, which point p' in $P_{<p}$ that is visible from p will be hit first by a counterclockwise rotating ray with origin p, starting at the upward vertical direction?.*

   (b) p is an obstacle point, and the left ray becomes tangent to G in a vertex g (Figure 6c). The new defining points will be g and q.

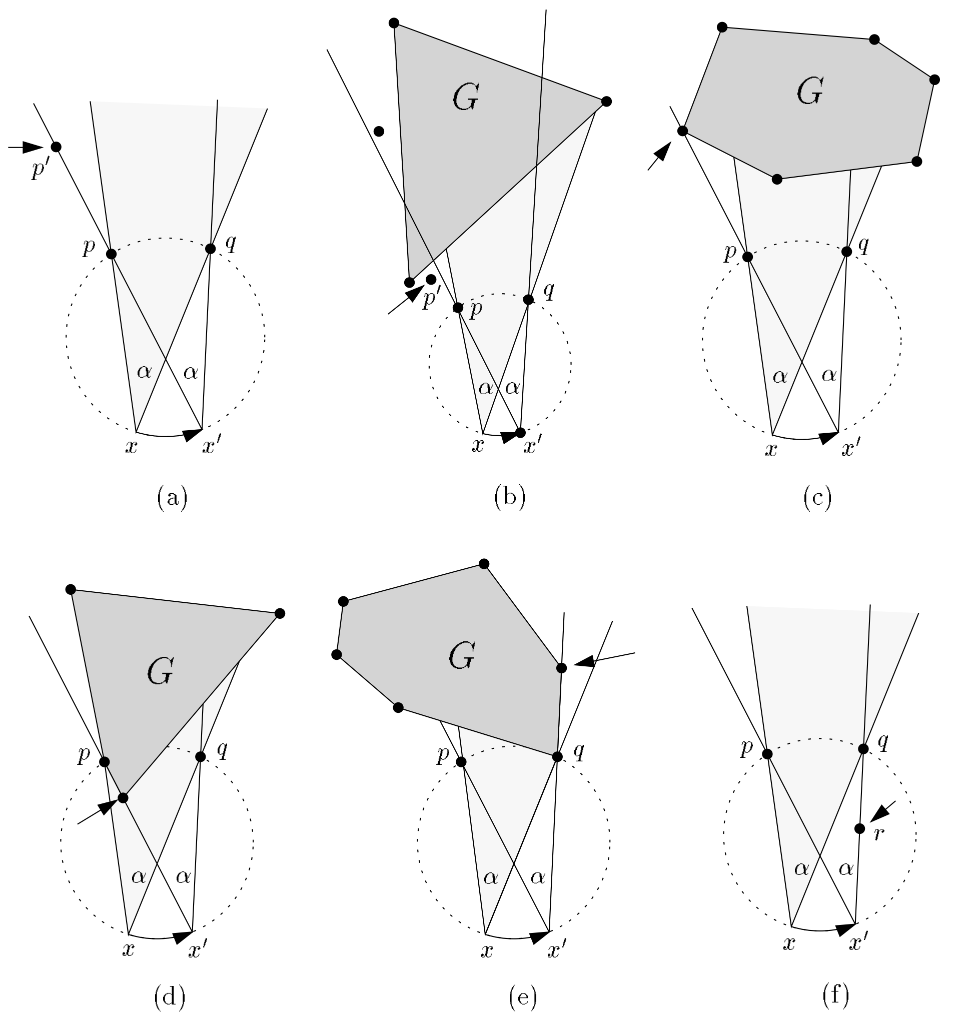

Figure 6: Finding the next vertex v' of β by rotating the α-cone

Note that pg must be the tangent to G from p, and can be computed in time $O(\log m)$ by binary search.

(c) p is a vertex of G and the left ray of $\gamma(x')$ hits the counterclockwise next vertex g of G (Figure 6d). The new defining points are then g and q.

The vertex g can be found in constant time from p.

2. Events on the right ray:

(a) The right ray of $\gamma(x)$ becomes vertical. If q is the rightmost vertex of G, we are finished. Otherwise, q is an obstacle point, and we will see below how to let the safe cone "wrap around" q.

We can easily find the point where this happens in constant time.

(b) q is a vertex of G and the right ray of $\gamma(x')$ hits the counterclockwise next vertex g of G (Figure 6e). The new defining points are p and g.

The vertex g can be found in $O(1)$ time from q.

(c) q is either an obstacle point of P or a vertex of G, and the segment $\overline{x'q}$ hits an obstacle point r of $P_{>x}$ (Figure 6f). The new defining points are p and r.

Finding the first point hit by the segment $\overline{x'q}$ requires a little more work than the other queries. We start with a vertical upward ray ρ with origin q and rotate it counterclockwise. No point in $P_{>x}$ will be hit by ρ before it reaches x, since the α-cone $\gamma(x)$ is empty. When we continue rotating ρ, it might encounter a point r' that would not be met by the segment $\overline{x'q}$. If this happens, we continue querying with the set $P_{>r'}$. This process is continued until we find the point r on the segment $\overline{x'q}$ or until ρ becomes vertical.

We now observe that every point in P can be found at most once in this manner, since once found the point will never appear in subsequent sets of the form $P_{>x}$. We can therefore charge the cost of this operation to the points encountered by the rotation of ρ.

It remains to implement the query: *Given points $q \in P \cup V$ and $x \in \mathbb{R}^2$, which point p' in $P_{>x}$ will be hit first by a counterclockwise rotating ray with origin q, starting at the upward vertical direction?*

We still have to explain how to "wrap around" the cone when its right ray has become vertical. In other words, we are at a vertex v of β, the right ray of $\gamma(v)$ is vertical and contains an obstacle point q, and we need to find a new vertex v' below q such that q lies on the left ray of $\gamma(v')$. The boundary curve will thus contain a vertical segment vv'. Note that v and v' can—by chance—happen to coincide. We still prefer to make a distinction, since $\gamma(v)$ and $\gamma(v')$ are quite distinct cones.

As we sketched before, we find $\gamma(v')$ by moving $\gamma(v)$ upwards until its apex is q. We then rotate the cone clockwise until its right ray encounters a point of P visible from q or becomes tangent to G, or until the left ray becomes vertical. Tangency to G is reached in the vertex of g where the tangent from q rests—we can find this in time $O(\log m)$ using binary search. The visible obstacle point will be found using a query of the form: *Given a point $q \in P$, which point p' in $P_{>q}$ visible from q will be hit first by a clockwise rotating ray with origin q, starting at the upward vertical direction?*

If the left ray becomes vertical, we continue by sliding the cone vertically downwards, until again its right ray reaches an obstacle point $p \in P$ or becomes tangent to G. Tangency to G is reached in the vertex of G extreme in direction $-\alpha$ (there is only one such vertex and we can simply precompute it at the beginning in linear time.) The obstacle point p can again be computed using a *sliding query* on $P_{>v}$.

We have thus seen that in every vertex v of β, we can find the next vertex v' by inspecting a constant number of candidates, each of which can be determined by performing certain queries on

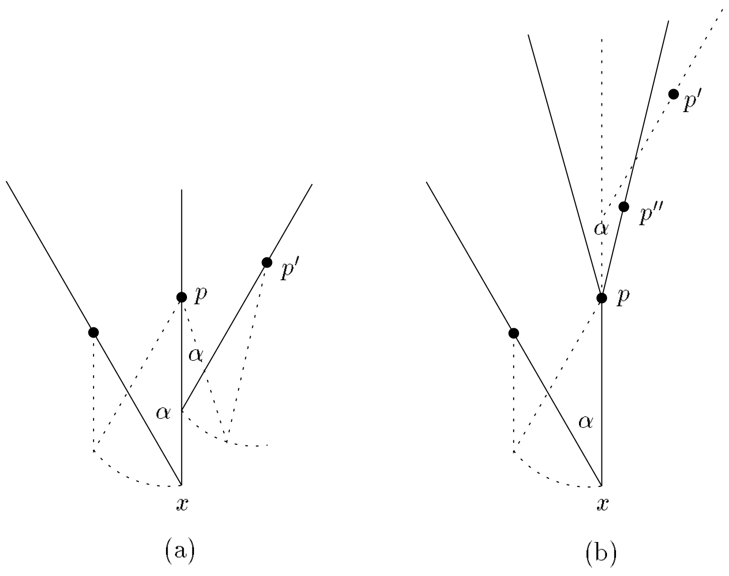

Figure 7: Finding the next edge on a vertical segment of β

G and P. The total number of queries is bounded by the number of points in P and the number of vertices of β, and hence is $O(m + n)$ by the results of the previous section.

In the next section we will show that all the queries needed for the algorithm can be implemented to run in $O(\log(m + n))$ time, with $O((m + n)\log(m + n))$ preprocessing. This allows us to state the following lemma:

Lemma 4 *$\mathcal{R}_{\alpha,\vec{u}}(G, S)$ can be computed in $O((m + n)\log(m + n))$ time.*

This finishes the proof of Theorem 3.

5 Implementing the queries

It remains to show how to implement the queries efficiently. We can do this by making use of *relative convex hulls*. A formal definition of relative convex hulls is given in de Berg's thesis [1, p. 116]. Informally, the convex hull of P relative to G is the shape an elastic rubber band, wrapped around the points in P, takes when it is released. It tries to take the shape of the convex hull of P, but it can be stopped by G in some points. Figure 8 shows the relative convex hull of $P_{\leq p}$ with respect to G. (All our relative convex hulls will be relative to the goal polygon G, so we omit that qualification.)

We will now show how all the queries from the previous section can be answered quickly, assuming we have precomputed the upper relative convex hulls of all the sets $P_{<x}$ and $P_{>x}$:

- *Given a point $q \in P$, which point p' in $P_{>q}$ visible from q will be hit first by a clockwise rotating ray with origin q, starting at the upward vertical direction?*

  This point p' is the right neighbor of q on the upper relative convex hull of $P_{\geq q}$.

- *Given a point $p \in P$, which point p' in $P_{<p}$ visible from p will be hit first by a counterclockwise rotating ray with origin p, starting at the upward vertical direction?*

  This point p' is the left neighbor of p on the upper relative convex hull of $P_{\leq p}$.

- The *sliding query*: Starting with an α-cone γ with apex in a point p (this is either the leftmost vertex of G or a point in P), what is the first point $p' \in P_{>p}$ encountered by the right ray of γ if it is shifted vertically downwards? Note that points that are not visible from the apex of the cone (when it encounters p') are not considered.

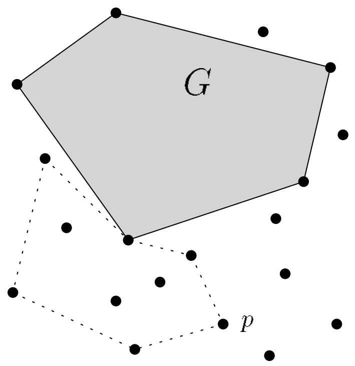

Figure 8: The convex hull of $P_{\leq p}$ relative to G.

The point p' is the leftmost vertex of the upper relative hull of $P_{>p}$ that is extreme in direction $-\alpha$.

- *Given points $q \in P \cup V$ and $x \in \mathbb{R}^2$, which point p' in $P_{>x}$ will be hit first by a counterclockwise rotating ray with origin q, starting at the upward vertical direction?*

  If q is an obstacle point in P, then p' is the left neighbor of q on the upper convex hull of $P_{>x}$.

  If q is a vertex of G, we can't use this idea, since not every vertex of G is on the upper relative convex hull of $P_{>x}$. We will show how to solve this problem after we have described how to construct relative convex hulls.

Of course we cannot afford to precompute and store the upper relative convex hulls for all the sets $P_{>p}$ and $P_{<p}$. What we do instead is to compute the upper convex hull of P, once by inserting the points from left to right, once by inserting the points from right to left, and to record the information necessary to answer all queries.

For the first query, we record the right neighbor of point $q \in P$ when we computed $P_{\geq q}$ while computing the upper relative convex hull from right to left. For the second query, we record the left neighbor of p when we computed $P_{\leq p}$ while computing the upper relative convex hull from left to right. For the sliding query, we keep track of the leftmost vertex extreme in direction $-\alpha$, and record it every time we add a point while computing the upper relative convex hull from right to left. It follows that these three queries can be answered in constant time, by reporting the recorded value.

The last query is a bit more complicated. Again we construct the upper relative convex hull of P from right to left, maintaining for every point q the moments when its left neighbor changes. We will see that there are at most $m + n$ changes, so maintaining these moments can be done in $O(m+n)$ time and space. The final structure consists of an array for every point $q \in P$ that stores the intervals and the corresponding neighbors. This structure can be used to answer the query in $O(\log(m + n))$ time.

It remains to demonstrate how to construct the upper convex hull of P relative to G. We will show how to compute this upper relative convex hull incrementally from left to right; a symmetric algorithm constructs it from right to left.

Note that due to the convexity of G, the upper relative convex hull of P relative to G has a special shape: It consists of a sequence of points in P, followed by a chain of consecutive vertices of G, and finally again a sequence of points in P.

We start by sorting the points in P in $O(n \log n)$ time. Then we process the points in P from left to right (Figure 9). Suppose we have constructed the upper relative convex hull of $P_{<p}$, and we want to insert p.

Let's first assume that at least one vertex of G is on the upper relative convex hull. We then connect p to the rightmost point p_i on the already constructed upper relative convex hull. If this makes p_i a reflex vertex, we remove p_i from the upper relative convex hull, and connect p to the left neighbor p_{i-1} of p_i. We repeat this process until we either reach a vertex p_j for which the insertion of the edge between p_j and p makes p_j a convex corner, or we reach a vertex g of G. In the latter case, we check in $O(1)$ time whether the edge gp intersects G. If it does, we add the counterclockwise neighbor g_i of g to the upper convex hull; we repeat this process until we find a vertex g_j for which the edge g_jp doesn't intersect G.

Note that due to the convexity of G, we do not need to test whether a newly inserted edge p_jp intersects G, if $p_j \in P$.

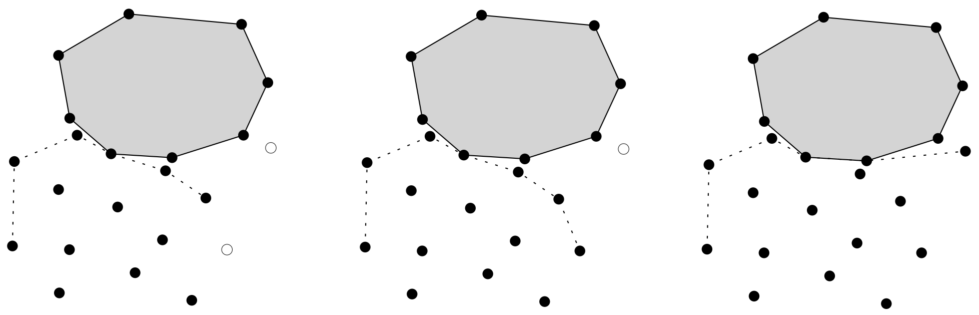

Figure 9: Constructing the upper convex hull of P relative to G.

Consider now the case where no vertex of G is on the current upper relative convex hull. After we connected p to a hull vertex p_j such that p_j forms a convex corner, we then have to check in $O(\log m)$ time whether p_jp intersects G (Figure 10). If this is the case, we determine the common tangent of G and the upper convex hull of $P_{<p}$ and the tangent to G through p. Using these tangents, we modify the upper relative convex hull as in Figure 10.

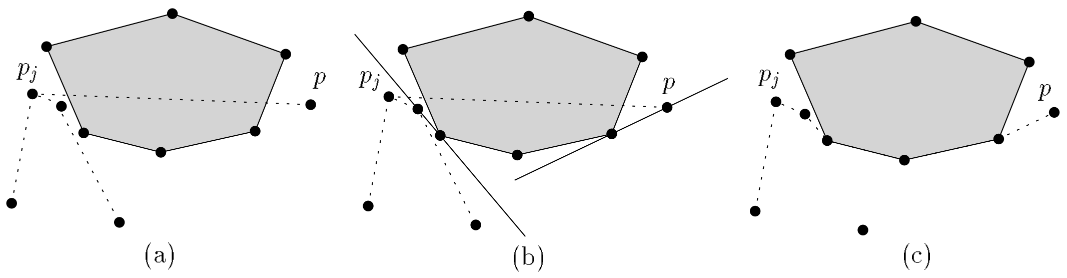

Figure 10: Determining the tangents

Sorting the points in P takes $O(n \log n)$ time. Determining whether the partially constructed relative upper convex hull intersects G takes $\log m$ time for every added point of P, until we actually find an intersection. Finding the tangents and making the necessary corrections to the upper relative convex hull after we have found an intersection can be done in $O(m+n)$ time and this has to be done only once. Every point $p \in P$ is added to the upper relative convex hull once, and removed from it at most once. Vertices of G are added at most once to the upper relative convex hull; once they have been added, they are never removed from it again. It follows that constructing the upper convex hull of P relative to G from the left to the right takes $O((m+n)\log(m+n))$ time.

The query *which point p' in $P_{>p}$ will be hit first by a leftward rotating ray with origin q, starting at the upward vertical direction* can't be solved using the upper convex hull of P relative to G, since not all vertices of G are on this hull. For the vertices of G we solve this query in a different way. First we note that for a vertex v_i, only the points in P which lie in the wedge formed by the supporting lines of the edges $v_{i-1}v_i$ and v_iv_{i+1} have to be considered (Figure 11): points above this wedge are not visible from v_i, and below this wedge v_i does not take part in the definition of vertices and edges of β, since v_{i+1} is visible.

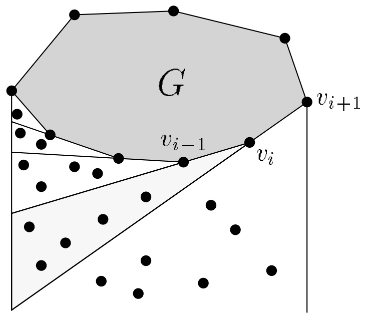

Figure 11: Wedges in the lower envelope of G

We now construct $O(m)$ convex hulls from the right to the left, one for each vertex in the lower envelope of G; for each vertex v we only consider the points of P that lie in the appropriate wedge. We process the points in P from the right to the left. For each point p we find the corresponding vertex v of G in time $O(\log m)$ by performing a binary search on the vertices of G. Again, for every vertex v we maintain the moments when its left neighbor changes. The total amount of time for constructing the $O(m)$ convex hulls and maintaining the necessary information is $O((m+n)\log(m+n))$.

The final structure consists of an array for every vertex v in the lower envelope of G that stores the intervals and the corresponding hit points. This structure can be used to answer the rotational queries in $\log(n)$ time.

6 A non-convex goal region

To round off our exhibition, we consider the case of a non-convex goal region. Surprisingly, in this case the complexity of the safe region turns out to be at least quadratic in m, even if the angle α is constant:

Theorem 4 *Given a simple polygon G of complexity m, a set S of n disjoint line segments, and an angle $\alpha < \pi$, the complexity of $\mathcal{R}_\alpha(G,S)$ is bounded by $O((m+n)^4)$. There is an example of a non-convex goal polygon G of complexity m, a set S of n line segments and an angle $\alpha > 0$ (which decreases with n, but independent of m) such that $\mathcal{R}_{\alpha,\vec{u}}(G,S)$ has complexity $\Omega(m^2+n^4)$.*

Proof: The proof of the upper bound and the $\Omega(n^4)$ lower bound are the same as in the proof of Theorem 1.

To prove the $\Omega(m^2)$ lower bound, we create the following construction (see Figure 12): for the goal polygon G, we start with a rectangle in which we make m rectangular "caves". In every cave we place an obstacle of constant complexity. The caves have very small openings. As a result, we can only see the obstacle in the cave when we are very close to the line that passes through the opening. By giving the openings different orientations, we obtain m intersecting lines on which we can see an obstacle while from any other position the obstacles are hidden. Hence, for any point in the interior of the large α-cone to the right of the goal polygon, we can find a safe α-cone (with its rays parallel to those of the large cone), except for points on the m lines. It is possible to arrange these lines such that they make an $\Omega(m^2)$ arrangement, yielding $\Omega(m^2)$ safe regions. $\square$

Note that in the example α can be chosen rather large, and does not need to vary as a function of m. The reason is that most edges of $\mathcal{R}_\alpha(G,S)$ are not determined by points on the bounding rays of the safe cone, but are edges of visibility—the locus of points where an obstacle becomes visible behind a corner of the goal region.

It seems that we have reached the limits of the usefulness of our model here. A robot has bounded precision, and it does not seem to matter whether an obstacle can be seen through an

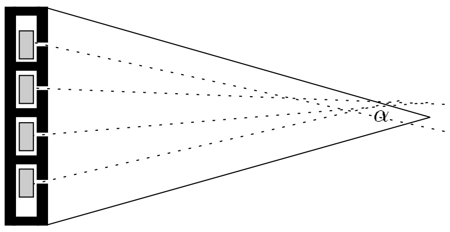

Figure 12: Lower bound example of $\Omega(m^2 + n^4)$.

infinitely small gap in the goal polygon. We believe that if non-convex goal regions are needed in practice, then a different model of uncertainty will have to be developed.

References

[1] M. de Berg. *Ray Shooting, Depth Orders and Hidden Surface Removal*, volume 703 of *Lecture Notes Comput. Sci.* Springer-Verlag, Berlin, Germany, 1993.

[2] Mark de Berg, Leonidas Guibas, Dan Halperin, Mark Overmars, Otfried Schwarzkopf, Micha Sharir, and Monique Teillaud. Reaching a goal with directional uncertainty. *Theoret. Comput. Sci.*, 140:301–317, 1995.

[3] H. Edelsbrunner, L. J. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and of segments. *Discrete Comput. Geom.*, 5:161–196, 1990.

[4] M. Erdmann. On motion planning with uncertainty. Technical report, AI Laboratory, MIT, 1984.

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

[6] T. Lozano-Pérez, M. T. Mason, and R. Taylor. Automatic synthesis of fine-motion strategies for robots. *Internat. J. Robot. Res.*, 3(1), 1984.

[7] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. *SIAM J. Comput.*, 23:154–169, 1994.

[8] Marc van Kreveld. On fat partitioning, fat covering, and the union size of polygons. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes Comput. Sci.*, pages 452–463. Springer-Verlag, 1993.