Trajectory Planning for an Articulated Probe[⋆]

Ka Yaw Teo[†], Ovidiu Daescu, Kyle Fox

Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA.

Abstract

We consider a new trajectory planning problem involving a simple articulated probe. The probe is modeled as two line segments ab and bc, with a joint at the common point b, where bc is of fixed length r and ab is of arbitrarily large length. Initially, ab and bc are collinear. Given a set of obstacles in the form of n line segments and a target point t, the probe is to first be inserted in straight line, followed possibly by a rotation of bc, so that in the final configuration c coincides with t, all while avoiding intersections with the obstacles. We prove that a feasible probe trajectory can be determined in $O(n^2 \log n)$ time using $O(n \log n)$ space (in fact, our algorithm finds a set of "extremal" feasible configurations). We also show that, for any constant $\delta > 0$, a feasible probe trajectory with a clearance δ can be determined in $O(n^2 \log n)$ time using $O(n^2)$ space. Furthermore, we demonstrate that our algorithms can be extended to the case of h polygonal obstacles, where we can find a feasible solution in $O(n^2 + h^2 \log n)$ time using $O(n \log n)$ space when there is no clearance requirement or $O(n^2)$ space otherwise. In the process of describing the algorithms, we address and solve some other interesting problems, such as circular sector emptiness queries and a special case of circular arc ray shooting queries for line segments and circular arcs in the plane.

Keywords: articulated probe trajectory, motion planning, circular sector intersection, circular arc intersection, circular sector emptiness, clearance

[⋆]Preliminary versions of this work were presented at the 30th Annual Canadian Conference on Computational Geometry [1] and the 28th Annual Fall Workshop on Computational Geometry [2].

[†]Corresponding author

Email addresses: `ka.teo@utdallas.edu` (Ka Yaw Teo), `ovidiu.daescu@utdallas.edu` (Ovidiu Daescu), `kyle.fox@utdallas.edu` (Kyle Fox)

1. Introduction

Consider the following *trajectory (or motion) planning problem*. An articulated needle-like probe is modeled in $\mathbb{R}^2$ as two line segments, ab and bc, joined at point b. Line segment bc may rotate around point b. The length of line segment ab can be arbitrary large, while line segment bc has a fixed length r (e.g., unit length). A two-dimensional workspace is defined as the region bounded by a circle S, which encloses a set P of n disjoint line segment obstacles (see Figure 1). Let t be a given point in the free space (i.e., inside S and outside P). Without loss of generality, let S be a circle centered at t.

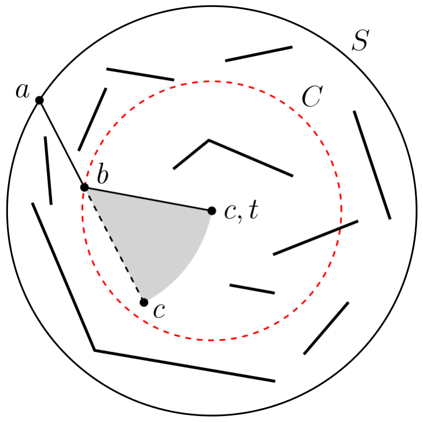


Figure 1: Trajectory planning for an articulated probe. After a straight insertion of line segment abc, in order to reach point t in the midst of obstacles, line segment bc may be required to rotate from its intermediate position (dashed line) to the final position (solid line).

In the beginning, the probe assumes a straight *unarticulated* configuration; that is, line segments ab and bc are collinear, and point b lies between points a and c. Starting from outside S, the unarticulated probe, represented by straight line segment abc, may be inserted into S as long as no obstacle is intersected by abc. After the insertion has completed, line segment bc may be rotated around point b up to $\pi/2$ radians in either direction, provided that line segment bc does not collide with any obstacle. If a rotation of line segment bc is performed, then the probe is said to have an *articulated* configuration. An *intermediate* configuration of the probe is defined as the probe configuration after inserting straight line segment abc into S and before rotating line segment bc.

A *feasible probe trajectory* consists of an initial insertion of straight line segment abc, possibly followed by a rotation of line segment bc around point b, such that point c ends at the target point t, while avoiding the obstacles in the process of insertion and rotation.

The objective of the problem is to determine a feasible probe trajectory, if one exists. As far as the authors are aware, no previous geometric-combinatorial algorithm is known for this problem (except [1, 2]).

Since line segment bc may only rotate as far as $\pi/2$ radians in either direction, it is easy to observe that, for any feasible probe trajectory, point b is the first intersection between segment ab and a circle C of radius r centered at point t. As illustrated in Figure 1, when line segment bc rotates around point b, the area swept by line segment bc is a sector of a circle (i.e., a portion of a disk enclosed by two radii and a circular arc) with its center located on C, a radius r, and the endpoint of one of its bounding radii located at point t. For conciseness, the center of the circle on which a circular sector is based is referred to herein as the *center of the circular sector*.

Related work

The motion of a linkage – that is, a sequence of fixed-length edges connected consecutively through joints – has been formerly studied from various perspectives, ranging from basic properties and questions (e.g., reachability, reconfiguration, and locked decision) with strong geometric and topological aspects [3, 4] to application-driven robotic arm modeling and motion planning problems [5, 6].

The overall objective of a robotic motion planning problem is to determine a plan specifying a sequence of actions to be taken by a robot, starting from an initial state, in order to arrive at a desired final state without colliding into obstacles. In the case of a rigid polygonal robot in a workspace populated with polygonal obstacles, it is relatively practical to compute a feasible path by using a combinatorial (i.e., *complete*) planning approach without resorting to approximation [7, 8, 9]. However, inexact approaches based on sampling [10, 11] and subdivision [12, 13, 14] have been commonly used to obtain solutions for path finding problems involving complex robots with a high-dimensional configuration space, which are typically challenging to solve in a reasonable time by using exact geometric approaches in the continuous domain.

A sampling-based algorithm, such as Probabilistic Roadmap (PRM) [10] and Rapidly-Exploring Random Tree (RRT) [11], constructs a solution path in the free space by randomly sampling the configuration space without explicitly computing the boundary of the configuration space obstacles. A sampling-based planner may fail to find a feasible path even when one exists. A sampling-based method has been shown at best, under certain assumptions, to be *probabilistically complete* – that is, the planner *eventually* finds a solution path if one exists. In addition, sampling-based planners have been known to be associated with two critical issues – i) the difficulty of finding feasible paths through narrow passages in the free space [15, 16, 17], and ii) the incapability of detecting the non-existence of any collision-free path [10].

On the other hand, a subdivision-based (or approximate cell decomposition) algorithm [12, 13, 14] divides the configuration space into cells using an octree or a lattice, and classifies each cell as "full" if it lies entirely in the configuration space obstacles, "empty" if it lies within the free space, or "mixed" otherwise. The algorithm then performs a search to find a collision-free path. Clearly, a subdivision-based method is *resolution complete* – that is, the planner is complete only at the resolution of the chosen decomposition (lattice).

These approximate methods are generally effective in solving a wide range of motion planning problems, but not in planning manipulation tasks for multi-link robots with multiple constraints. Spatial constraints on the links or joints of a manipulator impose restrictions on its free configuration space, and it remains challenging for approximate methods to find valid paths in the sub-manifold describing the restricted free configuration space [18].

Our paper is concerned with finding a collision-free path of motion for a two-bar linkage constrained to an ordered sequence of motions – namely, a straight insertion (of the linkage) followed by a rotation (at the joint). Furthermore, one of the links is considered to be unbounded in length. Thus, the problem is different in structure from previous work. Given the problem, as one shall see later, has such a rich geometric and combinatorial structure, it is important to examine the problem in detail from a rigorous mathematical point of view and explore its intrinsic structure for algorithmic improvements. Heuristics and approximations may be necessary in practice; nevertheless, in order to advance or take full advantage of their efficacy, the exact solution and its limitations must be well understood.

Motivation

The problem setting described in the current study has practical relevance in the field of robotics, particularly in minimally invasive robotic surgery [19], where the plane of insertion for a surgical probe can be defined based on various medical imaging techniques. In minimally invasive surgical approaches, a small incision is made, and the surgical operation is performed by using specialized tools inserted through the incision.

Most conventional surgical devices are straight, rigid, or flexible. A simple articulated probe such as one defined herein could be useful in minimally invasive surgery for reaching previously unattainable targets by circumventing critical structures, and for reaching multiple targets from a single insertion site while minimizing healthy tissue damage.

Results and contributions

We at first describe an algorithm that finds a feasible probe trajectory in $O(n^2 \log n)$ time using $O(n \log n)$ space. In fact, our algorithm finds a set of so-called "extremal" feasible probe configurations. In such a configuration, one or two obstacle endpoints are tangent to the probe (see Figures 2 and 3, and Lemma 2.1).

In addition, we address a more general version of the problem, which asks for a feasible probe trajectory of a given clearance δ from the obstacles, for any constant $\delta > 0$ (a *δ-clearance probe trajectory* for short). A feasible probe trajectory is claimed to have a clearance δ from the obstacles if and only if every point of the trajectory is of at least distance δ from its nearest obstacle. We propose an algorithm that finds a δ-clearance probe trajectory in $O(n^2 \log n)$ time using $O(n^2)$ space.

In the process of describing our solution, we solve some special cases of a number of fundamental problems of theoretical interest in computational geometry, such as circular sector intersection and circular sector emptiness queries. In particular, we present a data structure of near-linear size with logarithmic query time for solving a special instance of the circular arc intersection query problem (i.e., for a query circular arc with a fixed radius r and a fixed endpoint t).

Furthermore, we extend our algorithms for articulated probe trajectory planning to the case of polygonal obstacles, where we exploit output sensitive algorithms with respect to the number of polygons and the complexity of the visibility (to infinity) from a given point. We show that a feasible probe trajectory among h polygonal obstacles with a total of n vertices can be determined in $O(n^2+h^2 \log h)$ time using $O(n \log n)$ space. When a clearance δ from the obstacles is required, a feasible trajectory can be obtained in $O(n^2 + h^2 \log h)$ time using $O(n^2)$ space.

The current paper is organized as follows. In Section 2, we show and prove the existence of the so-called extremal feasible probe trajectories. In Section 3, we present an algorithm for finding an extremal feasible probe trajectory among obstacle line segments. Section 4 addresses the circular sector intersection query problem involved in finding a feasible probe trajectory. In Section 5, we extend the preceding algorithm by requiring to compute a probe trajectory with a given clearance δ from the obstacle line segments. In Section 6 and 7, we describe the extensions of the previous algorithms, with and without the consideration of a clearance δ, for the case of polygonal obstacles. We conclude in Section 8 by summarizing our results and stating some remaining open problems.

2. Extremal feasible probe trajectories

Depending on how the obstacles are arranged, there may be an infinite number of feasible probe trajectories. In the lemma below, we discuss how any of these trajectories may be perturbed, while remaining feasible, into one of a finite number of probe trajectories where one or two obstacle endpoints lie tangent to the probe. We refer to these trajectories as *extremal* (see Figures 2 and 3). It then suffices for our algorithm to test feasibility of only the extremal trajectories; assuming a feasible probe trajectory exists at all, our algorithm will find its perturbation.

Lemma 2.1. *There exists a feasible probe trajectory such that the probe assumes either I) an* ***unarticulated*** *final configuration (i.e., a straight line segment abc with $c = t$) that passes through an obstacle endpoint, or II) an* ***articulated*** *final configuration (i.e., line segments ab and bc are not collinear and $c = t$) that passes through an obstacle endpoint outside C and another obstacle endpoint inside or outside C.*

Proof. The existence of feasible probe trajectories for cases I and II can be proven using simple perturbation arguments.

For case I, suppose that a feasible probe trajectory T exists such that the final pose of the probe is unarticulated and point c coincides with point t. In other words, t has an unobstructed vision to some points on the bounding circle S. Let T' be the trajectory resulting from rotating T around point t

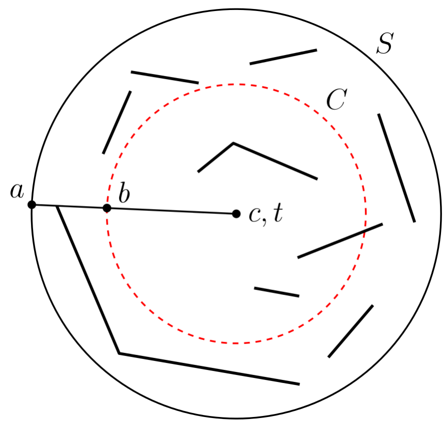


Figure 2: Extremal feasible *unarticulated* probe trajectory with a final configuration that is tangent to (at least) one obstacle endpoint.

in clockwise direction until T becomes tangent to an obstacle endpoint v. It follows that T' is also a feasible trajectory, and its articulation point b' is the intersection of segment at and circle C.

For case II, assume that a feasible probe trajectory T exists such that the final pose of the probe is articulated (i.e., line segments ab and bc are not collinear) and point c coincides with point t (Figure 4). Suppose probe trajectory T rotates segment bc clockwise around b to reach point t; the other case uses symmetric arguments. Let T' be the trajectory resulting from rotating line segment ab of T around point b in clockwise direction until line segment ab intersects an obstacle endpoint v_1 outside C. Given that the area swept by line segment bc of T' (to reach t) is within that of T (indicated by the shaded circular sectors in Figure 4), T' is also a feasible trajectory.

Now, let T'' be the trajectory obtained by moving b counter-clockwise along C while maintaining that v_1 lies on ab. The movement stops when either i) abt becomes a line segment (i.e., the final pose of the probe becomes unarticulated, where ab and bc are collinear and $c = t$), ii) ab intersects an obstacle endpoint that, immediately before the intersection, was to the left or right of ab's supporting line oriented from a to b, or iii) bt intersects some obstacle endpoint that was to the right of bt's supporting line oriented from b to t. Let v_2 denote this obstacle endpoint if either of the latter two cases applies. Let b'' be the new position of b after the movement stops. If abt becomes a line segment, we have achieved case I of the lemma. We now

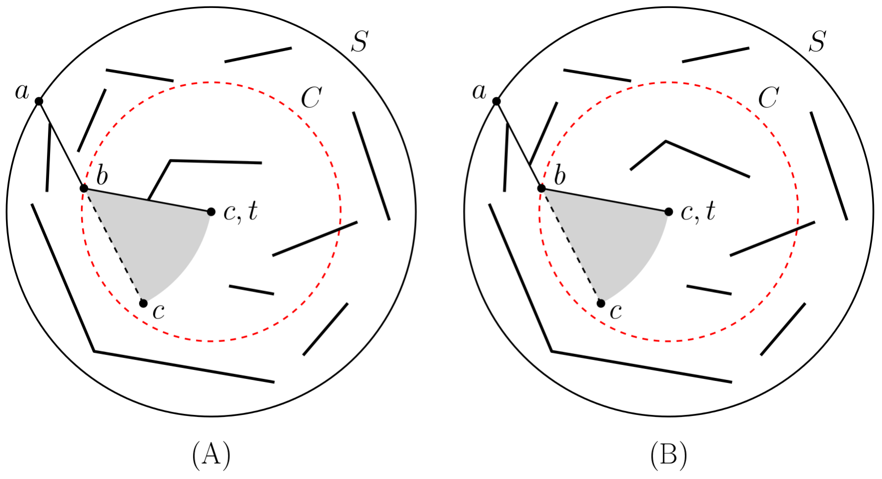


Figure 3: Extremal feasible *articulated* probe trajectory with a final configuration that is tangent to (A) one obstacle endpoint inside C and another outside C, or (B) two obstacle endpoints outside C.

assume otherwise.

Observe that every point of the circular sector centered at b'' lies to the side of the line through v_1 and b that contains t. They also lie to the side of the line through b'' and t that contains b. Therefore, these points either lie in the circular sector of radius r centered at t with arc endpoints at b and b'', or they lie in the wedge emanating from the circular sector centered at b. We know that the sector centered at t is empty because it was swept while constructing T''. We now argue that the remaining points of the sector centered at b'' not only lie in the wedge from b, but they actually lie in the circular sector centered at b. Since T' is a feasible probe trajectory, the sector at b and therefore the whole sector at b'' is empty as well, and T'' is a feasible probe trajectory.

Indeed, let x be a point of the sector centered at b'' that lies in the wedge at b. Let o be the intersection of the line segments bt and $b''x$ (Figure 5). By

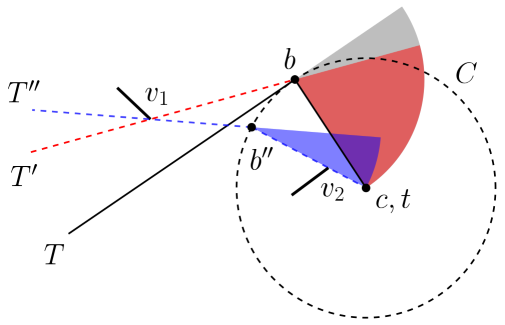


Figure 4: Case II of Lemma 2.1. T'' represents a feasible articulated probe trajectory such that the final configuration of the probe intersects an obstacle endpoint v_1 outside C and an obstacle endpoint v_2 inside C.

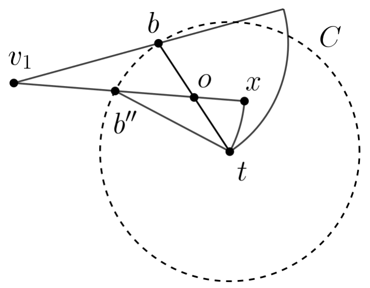


Figure 5: Point x lies inside the circular sector centered at b.

the triangle inequality,

$$\begin{aligned}|bx| &\leq |bo| + |ox| \\ &= |bt| - |ot| + |b''x| - |b''o| \\ &\leq |bt| + |b''x| - |b''t| \\ &= |b''x| \\ &\leq r\end{aligned}$$

If v_2 is inside circle C, then point b'' is the intersection between circle C and a ray emanating from point t through v_2. Otherwise, both v_1 and v_2 lie on the line segment ab''.

$\square$

3. Finding extremal trajectories

Based on the observation stated in Lemma 2.1, the set of extremal feasible probe trajectories can be obtained using the following approach. For the purpose of analysis and clarity, the line segments of P are divided into those lying inside C and those lying outside C. Since a line segment may intersect C at most two times, a line segment may be partitioned by C into at most three line segments. Let P_{in} (resp. P_{out}) be the set of line segments lying inside (resp. outside) C. In addition, let V, V_{in}, and V_{out} denote the set of endpoints of the line segments of P, P_{in}, and P_{out}, respectively. Let $n_{in} = |V_{in}|$ and $n_{out} = |V_{out}|$. We have $n_{in} + n_{out} = O(n)$.

Case I. Feasible unarticulated probe trajectory

We compute the set R of $O(n)$ rays, each of which originates at point t, passes through a vertex of V, and does not intersect any line segment of P. Each ray $\gamma \in R$ represents a feasible unarticulated probe trajectory.

The problem of finding the set R of rays (as well as some others that follow) can be reduced to the following radial visibility problem.

Problem 3.1. *Given a fixed point t and a real number r, let S be a circle of radius r centered at t. Given a set P of n obstacles of constant complexity inside S, return the portion of S visible from t.*

A simple algorithm for solving the radial visibility problem is described next. At first, find the two tangent lines from t to each obstacle in $O(n)$ time. Assume that t is located at (0, 0), and let θ be the angle of any said tangent line relative to the x-axis. Then, the pair of tangent lines to an obstacle defines a (possibly empty) occluded θ-interval $I = (\theta_s, \theta_f)$, within which circle S is invisible from t due to the obstruction by the obstacle, where $0 \leq \theta_s \leq \theta_f < 2\pi$. In order to ensure that all such occluded θ-intervals are defined over the range $[0, 2\pi)$, any θ-interval $I = (\theta_s, \theta_f)$ that contains $\theta = 0$ is divided into two corresponding θ-intervals $(\theta_s, 2\pi)$ and $[0, \theta_f)$.

The resulting $O(n)$ occluded θ-intervals, possibly overlapping, can be sorted by increasing value of θ_s in $O(n \log n)$ time, and can then be merged in $O(n)$ time to yield a set of non-overlapping occluded θ-intervals. The complement of these occluded θ-intervals is the set of radial visibility intervals that indicates the portion of S visible from t.

Lemma 3.1. *Given a fixed point t and a real number r, let S be a circle of radius r centered at t. Given a set P of n obstacles of constant complexity* *inside S, the portion of S visible from t can be determined in $O(n \log n)$ time using $O(n)$ space.*

The set R of rays is given by the endpoints of the radial visibility intervals. Thus, based on Lemma 3.1, the total time required to compute R is $O(n \log n)$.

Lemma 3.2. *The set of extremal feasible unarticulated probe trajectories can be determined in $O(n \log n)$ time.*

Case II. Feasible articulated probe trajectory

For ease of exposition, the two subcases of Case II, depending on whether an articulated final configuration intersects 1) an obstacle endpoint outside C and an obstacle endpoint inside C, or 2) two obstacle endpoints outside C, are treated separately.

Subcase 1. In order to find a feasible probe trajectory with an articulated final configuration that intersects an obstacle endpoint outside C and an obstacle endpoint inside C, we at first determine a feasible articulated *final* configuration in the following manner.

We compute the set R_{in} of rays, each of which originates at point t, passes through an endpoint of V_{in}, and does not intersect any line segment of P_{in}. By using the same algorithm for computing the set R of rays in Case I, R_{in} can be obtained in $O(n_{in} \log n_{in})$ time.

Recall, in any feasible probe trajectory, the line segment bc is rotated no more than $\pi/2$ radians from its initial configuration. Therefore, ab does not enter C, and it is unnecessary to check for intersections between ab and line segments of P_{in}. For each ray $\gamma_{in} \in R_{in}$, we i) find the intersection point b of γ_{in} and C in $O(1)$ time, and ii) compute the set R_{out} of rays, each of which originates at point b, passes through an endpoint of V_{out}, and does not intersect any line segment of P_{out}. This can be accomplished in $O(n_{out} \log n_{out})$ time.

A pair of rays $\gamma_{in} \in R_{in}$ and $\gamma_{out} \in R_{out}$ intersecting at a point b defines a feasible final configuration of an articulated probe trajectory, which intersects an endpoint outside C and an endpoint inside C. Given that the number of rays in R_{in} is bounded by $O(n_{in})$, the worst-case running time for finding the final configuration pairs of rays γ_{in} and γ_{out}, intersecting at a point b, is in the order of $O(n_{in} \log n_{in} + n_{in} n_{out} \log n_{out})$.

Subcase 2. In order to find a feasible probe trajectory with an articulated final configuration that intersects two obstacle endpoints outside C, we determine a feasible *intermediate* configuration using the following procedure.

We define the *reversal* of a ray γ originating at a point p as the ray originating at p but going in the opposite direction of γ. For each endpoint $v \in V_{out}$, we compute the set R of rays, each of which has the following properties: it originates at endpoint v, passes through an endpoint of $V_{out} \setminus \{v\}$, does not intersect any line segment of P, and its reversal intersects C and goes at least a distance r beyond the intersection point with C without intersecting any line segment of P. R can be obtained in $O(n \log n)$ time by computing the visibility polygon from v [20, 21, 22]. For each ray $\gamma \in R$, in $O(1)$ time, we find the first intersection point b of C with the reversal of γ.

A ray $\gamma \in R$ whose reversal intersects C at a point b and satisfies the obstacle free restriction above represents a feasible intermediate configuration of an articulated probe trajectory that intersects two vertices outside C. Since $|V_{out}| = n_{out}$, the worst-case running time for finding such a ray γ is $O(n_{out} n \log n)$.

An articulated probe trajectory with a feasible final or intermediate configuration is feasible if and only if the area swept by segment bc after the initial insertion (i.e., a circular sector) is not intersected by any obstacle. Thus, the remainder of Case II entails a circular sector intersection problem, detailed in the next section.

4. Circular sector intersection queries

The general line segment-circular sector intersection query problem can be formally stated as follows.

Given a set P of n line segments, preprocess it so that, for a query circular sector σ, one can efficiently determine whether σ intersects P.

For our purposes, it suffices to solve a special case of this problem where the radius of the circular sector is fixed to r and one endpoint of the circular arc of the sector is fixed at t. The intersection of a line segment and a circular sector can only occur as one of the three basic scenarios: (A) The segment intersects both radii of the sector. (B) The segment intersects both a radius of the sector and the sector's circular arc, or intersects the sector's circular arc twice. (C) At least one endpoint of the segment lies inside the sector. The three scenarios are depicted in Figure 6.

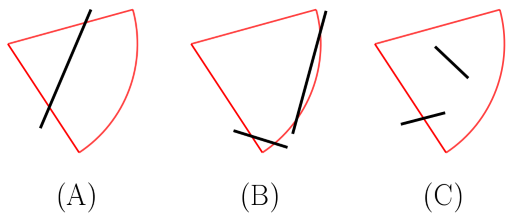


Figure 6: Basic scenarios of a line segment intersecting with a circular sector. (A) The segment crosses both radii of the sector. (B) The segment crosses a radius and the sector's circular arc, or crosses the sector's circular arc twice. (C) At least one endpoint of the segment lies inside the sector.

Recall that a feasible final or intermediate configuration for an articulated probe trajectory has been found in the previous section. Thus, one of the radii of the query circular sector is surely not intersected by any line segment of P. Therefore, scenario (A) can be eliminated from consideration.

Hence, our case of the circular sector intersection problem reduces to the following two problems: i) Circular arc intersection query – for detecting scenario (B). ii) Circular sector emptiness query – for detecting scenario (C). Note that a circular arc intersection query also rules out the instance of scenario (C) where the segment crosses the sector's circular arc once. Indeed, a circular arc intersection query addresses all scenarios where the segment crosses the sector's circular arc at least once.

4.1. Circular arc intersection queries

Consider the following circular arc intersection problem.

Problem 4.1. *Given a set P of n line segments, a fixed point t, and a fixed radius r, preprocess them so that, for a query circular arc γ of radius r that originates at t, one can efficiently determine if γ intersects P.*

Notice that since a query circular arc γ originates from a fixed point t and has a fixed radius r, the center of the circle supporting the circular arc γ is always located on a circle C of radius r centered at point t.

Let D be a circle of radius r centered at any point $p \in C$ (Figure 7). Note that circle D passes through the center t of circle C. Let θ be the angle of tp relative to the x-axis; D is uniquely determined by θ as p lies on C. We

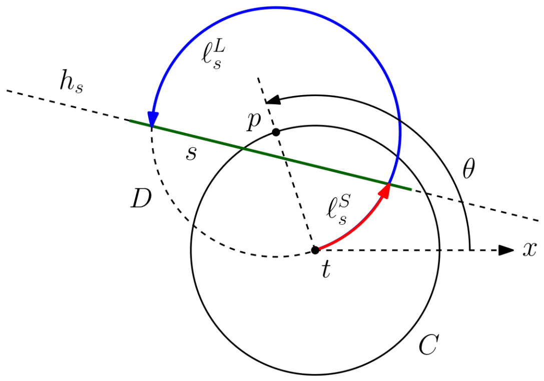


Figure 7: Counter-clockwise circular arcs ℓ_s^S and ℓ_s^L emanating from point t.

consider only query arcs that emanate counter-clockwise from t. The other case can be handled symmetrically.

Fix a line segment s, and let h_s be its supporting line. We define two partial functions $\ell_s^S, \ell_s^L : [0, 2\pi) \to \mathbb{R}_{\geq 0}$ that describe the lengths of (S)hort and (L)ong counter-clocwise arcs of D from t to intersections of the arcs with line segment s (more details to follow shortly). Specifically, let $\theta \in [0, 2\pi)$, and let D be the circle for that θ as defined above. If D does not intersect h_s, then neither $\ell_s^S(\theta)$ nor $\ell_s^L(\theta)$ are defined. Otherwise, there are two intersections between D and h_s. These intersections coincide if D lies tangent to h_s. Consider the first intersection with h_s when traveling along D counter-clockwise from t. If this intersection occurs on segment s, then $\ell_s^S(\theta)$ is the length of that counter-clockwise arc from t to the intersection. Otherwise, $\ell_s^S(\theta)$ is undefined. Similarly, if the second intersection with h_s occurs on s, then $\ell_s^L(\theta)$ is the length of the arc from t to the second intersection. Otherwise, it is undefined. We emphasize that functions ℓ_s^S and ℓ_s^L are defined based on the first and second intersections of the arc with h_s and whether those intersections occur on s (as opposed to the first and second intersections of the arc with s). We observe the following properties of ℓ_s^S (Figures 8 and 9). The same statements apply to ℓ_s^L as well.

Lemma 4.1. *Function ℓ_s^S is defined over at most two maximal contiguous subsets of* $[0, 2\pi)$.

Proof. For the sake of this proof, we extend the domain of ℓ_s^S to include all real numbers. By the definition of D, function ℓ_s^S is periodic with a period of 2π. Fix a value $\theta_0 \in [0, 2\pi)$ such that D does not intersect h_s, and consider the process of continuously increasing θ from θ_0 to $\theta_0 + 2\pi$ (see Figure 8). If D never intersects h_s, then ℓ_s^S is undefined for all $\theta \in [\theta_0, \theta + 2\pi)$. Otherwise, D remains disjoint from h_s until it becomes tangent at some value θ_1 for θ. Both intersections of D with h_s then travel monotonically along h_s until D becomes tangent with h_s again at some value θ_2 for θ. Finally, D remains disjoint from h_s until θ reaches $\theta_0 + 2\pi$. Function ℓ_s^S is defined during the (possibly empty) contiguous subset of $[\theta_1, \theta_2]$ for which the first intersection of D with h_s monotonically travels along s. We conclude there is at most one maximal contiguous subset of $[\theta_0, \theta_0 + 2\pi)$ during which ℓ_s^S is defined. Recalling ℓ_s^S is periodic with a period of 2π, we conclude the lemma. $\square$

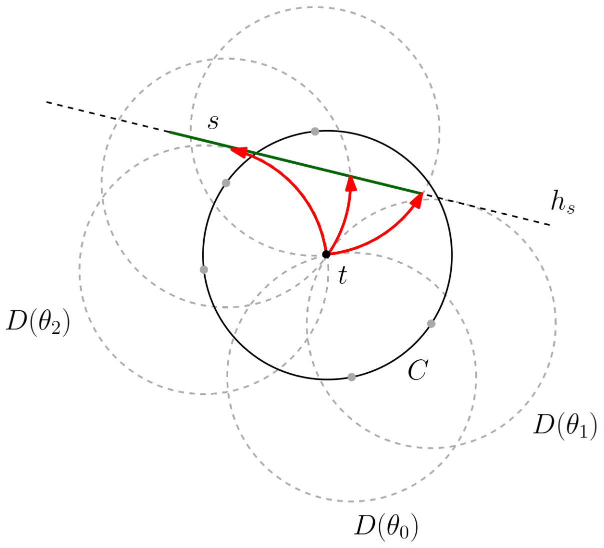
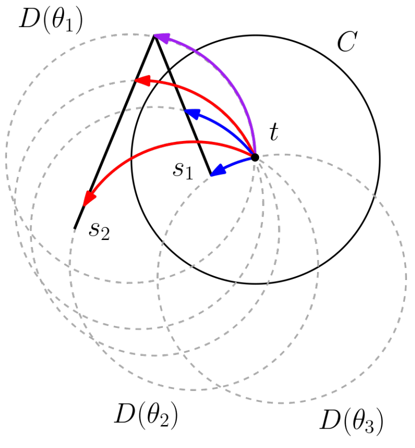


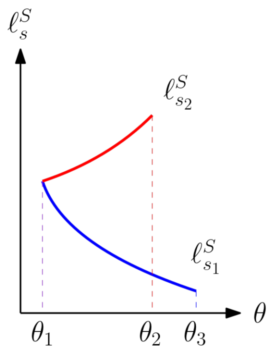
Figure 8: Illustration of Lemma 4.1. Circle D rotates counter-clockwise around t as θ increases from θ_0 to $\theta_0 + 2\pi$. Initially being disjoint from h_s at θ_0, D first becomes tangent to h_s at θ_1. D intersects h_s in two points when $\theta \in (\theta_1, \theta_2)$. D becomes tangent to h_s again at θ_2, after which D remains disjoint from h_s as θ increases to $\theta_0 + 2\pi$. ℓ_s^S is defined only when the first intersection of D with h_s occurs on s – that is, in this example, given by the lengths of the red arcs sweeping along s during a contiguous subset of $[\theta_1, \theta_2]$. Thus, ℓ_s^S is defined over at most one contiguous subset of $[\theta_0, \theta_0 + 2\pi)$.

Lemma 4.2. *Given two segments s_i, s_j, we have $\ell_{s_i}^S(\theta) = \ell_{s_j}^S(\theta)$ for at most one value of θ. Specifically, it is at the common endpoint of two maximal contiguous subsets of $[0, 2\pi)$ for which $\ell_{s_i}^S$ and $\ell_{s_j}^S$ are defined.*

Proof. We must have θ such that D's shorter counter-clockwise arc ends at the intersection of s_i and s_j. Segments s_i and s_j only intersect at their endpoints, implying the property (see Figure 9 for an illustration). $\square$



(A)



(B)

Figure 9: Illustration of Lemma 4.2. (A) The shorter counter-clockwise arcs of D sweep along s_1 and s_2 as θ increases. (B) The lengths of these sweeping arcs from t to their first intersections with the segments induce partially-defined piecewise continuous curves $\ell_{s_1}^S$ and $\ell_{s_2}^S$. Curves $\ell_{s_1}^S$ and $\ell_{s_2}^S$ intersect at θ_1 (and at most once) because their corresponding line segments s_1 and s_2 intersect at their endpoints.

The lower envelope of n segments in the plane has complexity bounded by the third-order Davenport-Schinzel sequence, which is $O(n\alpha(n))$, where $\alpha(n)$ is the inverse of the Ackermann function [23]. The lower envelope can be computed by a worst-case optimal divide-and-conquer algorithm running in $O(n \log n)$ time [24]. Let V^S be the lower envelope of the curves $\ell_{s_i}^S$ for all given line segments $s_i \in P$. Given any pair of curves $\ell_{s_i}^S$ and $\ell_{s_j}^S$ intersect at most once in their endpoints (as stated in Lemma 4.2), the size of lower envelope V^S is actually bounded by the second-order Davenport-Schinzel sequence, which is $O(n)$ in length, and we can compute it in $O(n \log n)$ time [23, 24]. We define and compute V^L similarly for the curves $\ell_{s_i}^L$.

In order to determine whether a query circular arc γ intersects P, the angle θ of center p of circular query arc γ from point t is looked up in V^S and V^L by using two binary searches that take $O(\log n)$ time. If the length of γ is less than $\ell_{s_i}^S(\theta)$ and $\ell_{s_i}^L(\theta)$ for all s_i, then γ does not intersect any line segment of P. Otherwise, it intersects the segment that lies on a lower envelope at θ. Thus, we obtain the following result, which can be easily shown to be worst-case optimal (see [22]).

Theorem 4.3. *A set P of n non-crossing line segments can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n)$ so that, for a query circular arc γ that originates at a fixed point t and has a fixed radius r, one can determine whether γ intersects P in $O(\log n)$ time.*

4.2. Circular sector emptiness queries

Our special case of the circular sector emptiness problem can be stated as follows.

Problem 4.2. *Given a set P of n points in the plane, a fixed point t, and a fixed radius r, preprocess them so that, for a query circular sector σ of radius r whose arc originates at t, one can efficiently determine whether σ contains any point of P.*

Circular sector σ can be partitioned into i) a triangle $\triangle bct$ and ii) a circular segment bounded by arc ct and the chord connecting the endpoints of the arc. Notice that circular sector σ is void of P if and only if both the triangle and circular segment are void of P. Thus, Problem 4.2 can be reduced to the following two subproblems – 1) restricted triangular emptiness query and 2) restricted circular segment emptiness query.

Consider the restricted triangular emptiness problem stated below.

Subproblem 4.2.1. *Given a set P of n points in the plane and a fixed point t, preprocess them so that, for a query triangle $\triangle$ with a vertex incident on t, one can efficiently determine whether $\triangle$ contains any point of P.*

As proposed by Benbernou et al. [25], Subproblem 4.2.1 can be solved as follows. The points of P can be at first sorted around point t in counter-clockwise order. Consider a wedge formed by two rays emanating from point t. Let i and j be the first and last points, respectively, within the wedge in counter-clockwise order. Points i and j can be determined for any given wedge in $O(\log n)$ time. Based on this observation, with $O(n \log n)$ preprocessing space and time, a restricted triangular emptiness query can be answered in $O(\log n)$ time. Daescu et al. [26] also used a similar idea to build a data structure for halfplane farthest-point queries.

The result for Subproblem 4.2.1 is summarized in the following lemma.

Lemma 4.4. *A set P of n points in the plane and a fixed point t can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n \log n)$ so that, for a query triangle $\triangle$ with a vertex incident on t, one can determine whether $\triangle$ contains any point of P in $O(\log n)$ time.*

The restricted circular segment emptiness problem is given as follows.

Subproblem 4.2.2. *Given a set P of n points in the plane and a fixed point t, preprocess them so that for a query circular segment s, bounded by a circular arc originating from t and the chord connecting the endpoints of the arc, one can efficiently determine if s contains any point of P.*

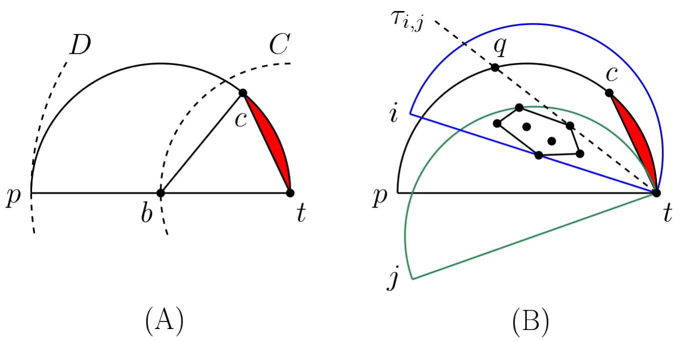


Figure 10: (A) Circular segment s_{ct} and its "enclosing" circular segment s_{pt}. (B) Circular segment s_{pt} and its corresponding event interval indicated by $[i, j]$.

Let s_{ct} be a query circular segment bounded by circular arc ct (of a circle C of radius r) and the chord connecting points c and t. In order to determine if s_{ct} contains any point of P, we begin by finding its corresponding "enclosing" circular segment (or semi-circle) s_{pt} as illustrated in Figure 10(A). s_{pt} is a circular segment bounded by arc pt and the chord connecting points p and t. Given circular arc ct emanating from point t and running counter-clockwise, s_{pt} can be determined by extending the arc until it intersects with a circle D of radius $2r$ centered at point t. The case of clockwise circular segments can be handled symmetrically.

Let $P_{pt} \subseteq P$ be the set of points in s_{pt}, and let $CH(P_{pt})$ be the convex hull of P_{pt}. As shown in Figure 10(B), at most two tangent lines on the convex hull pass through point t. Let q be the intersection point between arc pt and the first of the two tangents, denoted as τ, in counter-clockwise order. If point c is located on arc qt, then s_{ct} is empty of P.

We now describe a preprocessing procedure based on the earlier observations. At first, observe that, as s_{pt} rotates around point t counter-clockwise, a point of P may enter and leave s_{pt}. Each of these point-entering and -leaving events can be determined in $O(1)$ time by computing the intersections between the boundary of s_{pt} and each point of P. Since a point of P can enter and leave s_{pt} at most once, the total number of point-entering and -leaving events is bounded by $2n$. These events can be sorted in counter-clockwise order in $O(n \log n)$ time.

Let s_{it} and s_{jt} be the circular segments associated with any two consecutive events in sorted order, where i and j are the endpoints of the bounding arcs (emanating from point t) for s_{it} and s_{jt}, respectively (see Figure 10(B)). Notice that the set P_{pt} of points in s_{pt} remains constant within this event interval. For each of these event intervals $[i, j]$, the set P_{pt} of points of interest, their convex hull, and ultimately tangent line $\tau_{i,j}$ can be determined by using a dynamic convex hull data structure [27, 28], which requires $O(n)$ space, $O(n \log n)$ preprocessing time, $O(\log n)$ time per update operation, and $O(\log n)$ time for tangent queries. A simple $O(\log n)$ query-time data structure of linear size can then be built to store tangent lines $\tau_{i,j}$ for all event intervals $[i, j]$.

Thus, given a query circular segment s_{ct}, point p can be computed in $O(1)$ time, followed by a look-up of the event interval $[i, j]$ that contains p in $O(\log n)$ time. We then compute the intersection point q between arc pt and tangent line $\tau_{i,j}$. If the endpoint c of s_{ct} is located within arc qt, then s_{ct} does not contain any point of P.

Lemma 4.5. *For a fixed point t, a set P of n points in the plane can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n)$ so that, given a query circular segment s, bounded by a circular arc originating from t and the chord connecting the endpoints of the arc, one can determine whether s contains any point of P in $O(\log n)$ time.*

Altogether, the following result is obtained for Problem 4.2.

Theorem 4.6. *For a positive number r and a fixed point t, a set P of n points in the plane can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n \log n)$ so that, given a query circular sector σ with a fixed radius r and an endpoint of its arc located at t, one can determine whether σ contains any point of P in $O(\log n)$ time.*

Remark. *We can solve the general circular sector emptiness problem without a fixed radius or point t on the arc using a multi-level data structure similar to one by Matoušek [29] for counting points in the intersection of halfspaces (see also [26] for a similar approach on a related problem). Specifically, the first level is constructed for halfplane range queries to select the points of P lying on the σ side of the line supporting bc, and the second level is for halfplane range queries on the resulting points to select those lying on the σ side of bt. Thus, these two levels are used to find the points inside the wedge centered at b. Each subset of P on the second level is further preprocessed for nearest neighbor queries by computing its Voronoi diagram and augmenting it for point location. At query time, we can locate b in this data structure in logarithmic time. If the closest point is within distance r of b, then the circular sector is not empty of P. By following the strategy outlined in the first half of [29, Theorem 6.2], we can create a trade-off between space and time usage by our data structure.*

Theorem 4.7. *Let P be a set of n points. For any $\epsilon > 0$ and m such that $n^{1+\epsilon} \leq m \leq n^2$, set P can be preprocessed into a data structure of size $O(m)$ in $O(m \log n)$ time so that, for a query circular sector σ of radius r centered at point p, one can determine whether σ contains any points of P in $O(n/m^{1/2} \log^{5/2} n)$ time.*

Finishing up.

According to Theorems 4.3 and 4.6, the result for our case of the circular sector intersection problem can be stated as follows.

Theorem 4.8. *For a positive value r and a fixed point t, a set P of n line segments can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n \log n)$ so that, given a query circular sector σ with a fixed radius r whose circular arc has an endpoint at t, one can determine whether σ intersects P in $O(\log n)$ time.*

Recall, V denotes the set of endpoints of line segments of P. Let n_s be the number of endpoints in V within distance $2r$ from point t. Then, in Case II, given that $O(n_{in}n_{out} + n_{out}^2)$ queries are to be processed in the worst case and we only need to worry about endpoints lying sufficiently close to t, the following result is obtained.

Lemma 4.9. *A feasible articulated probe trajectory can be determined in $O((n_{in}n_{out} + n_{out}^2) \log n_s)$ time using $O(n_s \log n_s + n)$ space.*

Given that the space/time complexity of Case II (Lemma 4.9) is dominant over that of Case I (Lemma 3.2), the solution approach proposed herein for finding a feasible probe trajectory leads to the following theorem.

Theorem 4.10. *A feasible probe trajectory can be determined in* $O((n_{in}n_{out}+n_{out}^2)\log n_s)$ *time using* $O(n_s \log n_s + n)$ *space.*

Recall that $n_{in}, n_{out}, n_s \leq n$. Thus, the running time and space usage are bounded by $O(n^2 \log n)$ and $O(n \log n)$, respectively.

5. δ-clearance probe trajectory

Our algorithm for finding a δ-clearance probe trajectory follows the general framework of the solution approach described in Sections 3 and 4. We begin with the following observation, which is a generalization of Lemma 2.1.

Lemma 5.1. *There exists a feasible probe trajectory such that the probe assumes either I) an* ***unarticulated*** *final configuration (i.e., a straight line segment abc with* $c = t$*) that is tangent to an obstacle, or II) an* ***articulated*** *final configuration (i.e., line segments ab and bc are not collinear and* $c = t$*) that is tangent to an obstacle outside* C *and another obstacle inside or outside* C*.*

Proof. The lemma can be proven by using the same set of perturbation arguments as in Lemma 2.1. We previously argued that, amidst a set of obstacle line segments, a feasible probe trajectory, if one exists, can be perturbed into another that intersects one or two obstacle endpoints. Recall that any perturbation operation performed in proving Lemma 2.1 always involves rotating a segment of the probe (with respect to some point) until the segment intersects with an obstacle endpoint. Obviously, in the midst of other types of geometric obstacles (such as those whose boundaries consist of line segments and circular arcs), a similar conclusion can also be reached by applying the same set of perturbation operations. The only differences are as follows: i) In addition to rotating around a given point, a perturbation operation may rotate a segment of the probe along a circular arc boundary of an obstacle (i.e., changing the slope of the segment while keeping it tangent to the arc). ii) Each rotational perturbation step stops when the probe becomes tangent to an obstacle (instead of intersecting with an obstacle endpoint specifically). □

Based on Lemma 5.1, the set of extremal feasible probe trajectories with a given clearance δ can be obtained using the following approach.

For each obstacle line segment s of P, we define $H(s, \delta) = s \oplus B_\delta$ to be the dilation of s by a distance δ, where B_δ is a closed disk of radius δ, and $s \oplus B_\delta$ denotes the *Minkowski sum* of s and B_δ. A dilated line segment $H(s, \delta)$ has the shape of a *stadium* – a rectangle with two semi-circles attached to its sides (see Figure 11). Let $Q = \{H(s, \delta) | s \in P\}$ denote the resulting set of dilated line segments. Notice that the total number of vertices (and edges) of the dilated line segments of Q is $O(n)$.

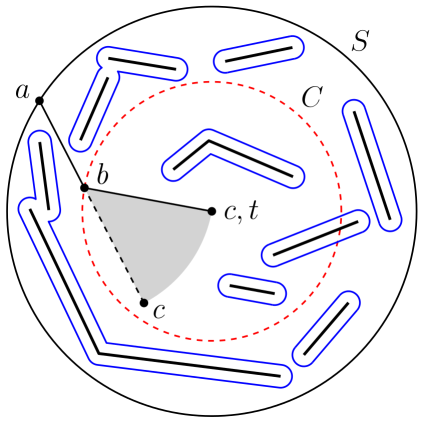


Figure 11: Planning a trajectory of a given clearance from obstacles for an articulated probe. Each obstacle line segment is “dilated” by a distance $\delta > 0$ using Minkowski sum to ensure that any computed feasible probe trajectory has a clearance of at least δ from the obstacles.

For the purpose of analysis and clarity, the dilated line segments of Q are divided into those lying inside C and those lying outside C. Since the boundary of a dilated line segment may intersect C at most four times, a dilated line segment may be partitioned by C into at most four open (piece-wise) curves, each of which may consist of line segments and circular arcs. Let Q_{in} be the set of curves lying inside C with $n_{in} = |Q_{in}|$, and let Q_{out} be the set outside C with $n_{out} = |Q_{out}|$. Note that $n_{in} + n_{out} = O(n)$.

The complexity of each dilated line segment of Q is $O(1)$; that is, the tangent line from a point to a dilated line segment, as well as the common tangent lines of two dilated line segments, can be computed in $O(1)$ time. For the sake of brevity, a dilated line segment obstacle is henceforth referred to as simply an *obstacle*.

The main added difficulty, when compared to the preceding case without clearance, is that after enlarging the obstacle line segments with a disk of radius δ to account for the required clearance, we have to work with obstacles with circular arc edges. That imposes a change on the data structures needed to handle various operations, particularly for visibility computation and circular arc queries, as detailed next.

Case I. δ-clearance unarticulated probe trajectory

As in the case without clearance, we compute the set R of $O(n)$ rays, each of which i) originates at point t, ii) is tangent to an obstacle of Q, and iii) does not intersect any obstacle of Q. Each ray $\gamma \in R$ represents an extremal δ-clearance unarticulated probe trajectory. According to Lemma 3.1, the set R of rays can be computed in $O(n \log n)$ time.

Lemma 5.2. *The set of extremal δ-clearance unarticulated probe trajectories can be determined in $O(n \log n)$ time.*

Case II. δ-clearance articulated probe trajectory

We consider separately the two subcases of Case II, providing whether an articulated final configuration is tangent to 1) an obstacle outside C and an obstacle inside C, or 2) two obstacles outside C.

Subcase 1. In order to find a feasible probe trajectory with an articulated final configuration that is tangent to an obstacle outside C and another obstacle inside C, we at first compute a feasible articulated *final* configuration by using the same method as the case without clearance. Recall that finding such a final configuration, represented by a pair of rays γ_{in} and γ_{out} intersecting at a point b, takes $O(n_{in} \log n_{in} + n_{in} n_{out} \log n_{out})$ time in the worst case.

After finding a probe trajectory with a feasible articulated final configuration, we examine the feasibility of its associated *intermediate* configuration.

For each computed point b, we consider a circle B of radius r centered at b, and find the radial visibility intervals in $O(n_b \log n_b)$ time (see Lemma 3.1), where n_b is the number of obstacles lying within B. Note that $n_b \leq n_s$ for any point $b \in C$, where n_s is the number of obstacles within distance $2r$ from point t. Recall that the size of R_{in} is bounded by $O(n_{in})$ (i.e., the upper bound on the number of distinct points b computed). Thus, the total time required to find the radial visibility intervals for all computed points b is $O(n_{in} n_s \log n_s)$.

After finding the radial visibility intervals for a point b, one can determine if a given radius of circle B intersects with any obstacle inside B in $O(\log n_s)$ time by using a binary search. Hence, it takes $O(\log n_s)$ time to determine if a segment bc (of an intermediate configuration) intersects with any obstacle. Since there could be $O(n_{in} n_{out})$ such segments bc, the worst-case running time for finding a feasible final configuration (that is tangent to an obstacle inside C and another obstacle outside C) with a feasible intermediate configuration is $O(n_{in} \log n_{in} + n_{in} n_{out} \log n_{out}) + O(n_{in} n_s \log n_s + n_{in} n_{out} \log n_s)$.

Subcase 2. In order to find a feasible probe trajectory with an articulated final configuration that is tangent to two obstacles outside C, we at first determine a feasible *intermediate* configuration in the following manner.

We compute the set R of rays, each of which i) originates from some point on circle S, ii) is a common tangent line between two obstacles of Q_{out}, iii) does not intersect any obstacle of Q_{out}, iv) intersects C, and v) goes at least distance r beyond the intersection point with C without intersecting any obstacle of Q.

R can be obtained by using the *visibility complex* of Q. The visibility complex is a two-dimensional subdivision in which each cell corresponds to a collection of rays with the same visibility properties [30]. For a simple scene of n obstacles with constant complexity, the visibility complex can be computed in $O(n \log n + k)$ time using $O(k)$ space, where k is the size of the visibility complex (or the corresponding tangent visibility graph). In the worst case, $k = O(n^2)$. After the visibility complex of Q is built, we can find the set R of rays (i.e., the set of bitangent lines that satisfy the obstacle-free restriction above) by simply traversing the cells of the visibility complex in $O(n^2)$ time.

After finding a probe trajectory with a feasible intermediate configuration, we determine if its associated *final* configuration is feasible.

Lemma 3.1 can be applied as follows in determining whether a segment bt (of a final configuration) intersects with any obstacle. The radial visibility intervals are computed with respect to circle C centered at point t in $O(n_{in} \log n_{in})$ time. Given that $O(n_{out}^2)$ such segments bt are to be examined (using binary searches), the worst-case running time for finding a feasible intermediate configuration (that is tangent to two obstacles outside C) with a feasible final configuration is $O(n^2) + O(n_{in} \log n_{in} + n_{out}^2 \log n_{in})$.

An articulated probe trajectory with both a feasible final configuration and a feasible intermediate configuration is feasible if and only if the area swept by segment bc after the initial insertion (i.e., a circular sector) is not intersected by any obstacle. Thus, the remainder of Case II involves a circular sector intersection problem.

Circular sector intersection queries. Similar as before, instead of addressing the general problem, it is sufficient to solve the special case in which the radius of the circular sector is fixed at r and one endpoint of the circular arc of the sector is incident at t.

Recall that we have previously found a pair of feasible final and intermediate configurations for an articulated probe trajectory. Thus, both radii of the query circular sector are certainly not intersected by any obstacle of Q. Therefore, an obstacle can only intersect with a query circular sector by i) intersecting the sector's arc, or ii) lying completely inside the sector. As a result, our case of the circular sector intersection problem can be reduced to the following two problems – i) circular arc intersection query, and ii) circular sector emptiness query.

Circular arc intersection queries. Consider the following circular arc intersection problem.

Problem 5.1. *Given a set Q of n line segments and semi-circular arcs, preprocess it so that, for a query circular arc γ that originates at a fixed point t and has a fixed radius r, one can efficiently determine if γ intersects Q.*

Problem 5.1 can be solved by using a similar data structure (i.e., lower envelopes) as constructed in the case without clearance (Section 4.1). Given that two semi-circular arcs (or an arc and a line segment, or two line segments) of Q can intersect at most twice, the size of the lower envelope is bounded by the fourth-order Davenport-Schinzel sequence, which is $O(n \cdot 2^{\alpha(n)})$. The lower envelope can be computed in $O(n\alpha(n)\log n)$ time [23, 24]. A binary search can be performed on the lower envelope to determine if a query circular arc γ intersects Q. Thus, the following result is obtained.

Lemma 5.3. *A set Q of n line segments and circular arcs can be preprocessed in $O(n\alpha(n)\log n)$ time into a data structure of size $O(n \cdot 2^{\alpha(n)})$ so that, for a* *query circular arc γ that originates at a fixed point t and has a fixed radius r, one can determine whether γ intersects Q in $O(\log n)$ time.*

Circular sector emptiness queries. This query problem is exactly the same as that previously described in Section 4.2, and the result is summarized in Theorem 4.6.

Recall that n_s is the number of obstacles within distance $2r$ from point t. In Case II, since we need to process $O(n_{in}n_{out} + n_{out}^2)$ queries in the worst case and we are only concerned with obstacles lying sufficiently close to t, the following result is obtained.

Lemma 5.4. *A δ-clearance articulated probe trajectory can be determined in time $O(n_{in} \log n_{in} + n_{in}n_{out} \log n_{out}) + O(n_{in}n_s \log n_s + n_{in}n_{out} \log n_s) + O(n^2) + O(n_{in} \log n_{in} + n_{out}^2 \log n_{in}) + O((n_{in}n_{out} + n_{out}^2) \log n_s)$ using $O(n^2 + n_s \log n_s)$ space.*

Theorem 5.5. *A δ-clearance probe trajectory can be determined in $O(n^2 \log n)$ time using $O(n^2)$ space.*

6. Extremal feasible probe trajectory amidst polygonal obstacles

We can extend the algorithms presented in the prior sections to the case of simple polygonal obstacles. Let P be a set of h simple polygonal obstacles with a total of n vertices.

Case I. Feasible unarticulated probe trajectory

In the radial visibility problem, as previously defined in Problem 3.1, when P is a set of h polygonal obstacles with n vertices in total, $O(h)$ occluded intervals (each of which is delimited by a pair of tangent lines from t to an obstacle) can be computed in $O(n)$ time. These occluded intervals can then be sorted and merged in $O(h \log h)$ time to yield the visibility intervals. The following lemma is obtained as a result.

Lemma 6.1. *Given a fixed point t and a real number r, let S be a circle of radius r centered at t. Given a set P of h polygonal obstacles (with n vertices in total) inside S, the portion of S visible from t can be determined in $O(n + h \log h)$ time.*

Note that, after finding the visibility intervals, one can determine if a given radius of circle S intersects with any polygonal obstacle inside S in $O(\log h)$ time by using a binary search. Thus, the total time required for determining the set of extremal feasible unarticulated probe trajectories in the midst of h simple polygonal obstacles is $O(n + h \log h)$.

Case II. Feasible articulated probe trajectory

The polygonal obstacles of P can be divided into those lying inside C and those lying outside C. Let P_{in} (resp. P_{out}) be the set of simple polygons lying inside (resp. outside) C. Let $h_{in} = |P_{in}|$ and $h_{out} = |P_{in}|$. Note that $h_{in} + h_{out} = O(h)$. In addition, let n_{in} (resp. n_{out}) denote the number of vertices of the polygons in P_{in} (resp. P_{out}).

For the purpose of subsequent computations, a simple polygon $\Gamma \in P$ that intersects C can be pseudo-partitioned into a chain Γ_{in} (of line segments and possibly circular arcs) and a simple polygonal chain Γ_{out}, as illustrated in Figure 12.

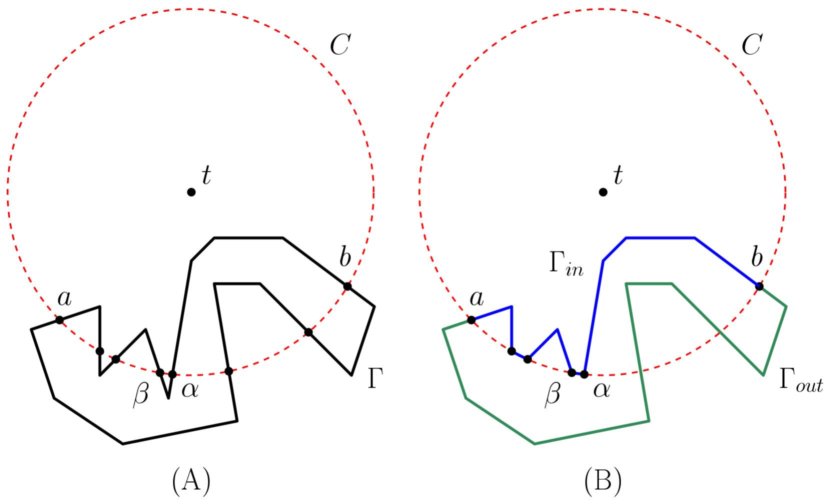


Figure 12: Pseudo-partitioning of (A) a simple polygon Γ intersecting C into (B) a chain Γ_{in} of line segments and circular arcs and a simple polygonal chain Γ_{out}.

Let polygon Γ be represented by a sequence of k vertices $v_1, v_2, ..., v_k$ ($v_{k+1} = v_1$) going counter-clockwise around the polygon. Let e_i be the i-th (directed) edge from vertex v_i to vertex v_{i+1} for $i = 1, ..., k$. An edge e_i may intersect C at most twice. When an edge e_i intersects C only once, the intersection between e_i and C is called an in-out (resp. out-in) intersection if v_i is inside (resp. outside) C and v_{i+1} is outside (resp. inside) C. When an edge e_i intersects C twice, e_i can be considered as two end-to-end collinear edge segments $v_i u$ and uv_{i+1}, where point $u \in e_i$ lies inside C, yielding a pair of out-in and in-out intersection points.

By using a brute-force method, one can find the set of all (in-out and out-in) intersection points between Γ and C in $O(k)$ time. As depicted in Figure 12, let a and b be the extreme in-out and out-in intersection points, respectively.

Let $\gamma[p_i, p_j]$ represent the polygonal subchain of Γ from point p_i to p_j in counter-clockwise direction. Γ_{in} is constructed from $\gamma[b, a]$ as follows. Let (α, β) denote a pair of consecutive in-out and out-in intersection points between $\gamma[b, a]$ and C (in counter-clockwise direction), if any. For each (α, β) between $\gamma[b, a]$ and C, replace the (counter-clockwise) polygonal chain of $\gamma[b, a]$ from α to β with the (counter-clockwise) circular arc of C from α to β. Γ_{in} is the resulting $\gamma[b, a]$ after chain modification. On the other hand, Γ_{out} is simply the polygonal subchain $\gamma[a, b]$ of Γ.

As a result of the pseudo-partitioning process, a polygon Γ that intersects C is "divided" into $\Gamma_{in} \in P_{in}$ and $\Gamma_{out} \in P_{out}$. Since the pseudo-partitioning of a simple polygon intersecting C results in two polygonal chains, each of which has at most $O(n)$ vertices, $|P_{in}|$ and $|P_{out}|$ are bounded by $O(n)$, and $n_{in} + n_{out} = O(n)$. Besides, the pseudo-partitioning ensures that the geometric procedure previously established for obstacle line segments (divided into those inside and outside C) continues to work for the case of polygonal obstacles. Recall that the only geometric operations that call for a partitioning of the obstacles are those used for finding feasible intermediate and final configurations in the first half of the algorithm (see Section 3 for instance). One can easily figure that those geometric operations, which mostly consist of finding radial visibility intervals with respect to some given point, remain valid in the case of polygonal obstacles after being pseudo-partitioned into those inside and outside C.

Subcase 1. We proceed similarly and use the same notations as in the case of obstacle line segments. By using Lemma 6.1, R_{in} can be obtained in $O(n_{in} + h_{in} \log h_{in})$ time, followed by the computation of R_{out} (for each ray $\gamma_{in} \in R_{in}$) in $O(n_{out} + h_{out} \log h_{out})$ time. Since $|R_{in}|$ is bounded by $O(h_{in})$, the worst-case running time for finding a feasible final configuration that is tangent to a polygonal obstacle inside C and another outside C is $O(n_{in} + h_{in} \log h_{in} + h_{in} n_{out} + h_{in} h_{out} \log h_{out})$.

Subcase 2. The algorithm previously described for obstacle line segments in Case II Subcase 2 can also be used herein for polygonal obstacles. The computation of the visibility polygon from a vertex (of a polygonal obstacle outside C) among h polygonal obstacles takes $O(n+h\log h)$ time [21]. Thus, the worst-case running time for finding a feasible intermediate configuration that is tangent to two polygonal obstacles outside C is $O(n_{out}n+n_{out}h\log h)$.

Circular sector intersection queries. The prior solution approach (described for obstacle line segments) for solving the circular sector intersection query problem remains valid for the case of polygonal obstacles. Hence, we obtain similar results as detailed in Theorems 4.3 and 4.6. As a result, a circular sector intersection query can be answered in $O(\log n)$ time with $O(n\log n)$ preprocessing space and time.

Given that $O(h_{in}h_{out}+h_{out}^2)$ queries are to be processed in the worst case, a feasible articulated probe trajectory can be determined in time $O(n_{in}+h_{in}\log h_{in}+h_{in}n_{out}+h_{in}h_{out}\log h_{out}+n_{out}n+n_{out}h\log h+(h_{in}h_{out}+h_{out}^2)\log n_s)$ using $O(n_s\log n_s+n)$ space, where n_s is the number of vertices within distance $2r$ from point t.

Given that $h_{in}, h_{out}\leq h$, $n_s\leq n$, and $h\leq n$, the final result can be stated as the following theorem.

Theorem 6.2. *An extremal feasible probe trajectory amidst h simple polygonal obstacles with n vertices can be determined in $O(n^2+h^2\log n)$ time using $O(n\log n)$ space.*

7. δ-clearance probe trajectory amidst polygonal obstacles

As with dilated line segment obstacles, a dilated simple polygonal obstacle is a convex region whose boundary consists of line segments and circular arcs. Since a vertex of a simple polygon, after dilation, gives rise to at most two vertices and a circular arc, the number of vertices or (line-segment and circular-arc) edges remains in the order of $O(n)$.

The solution approach previously described for dilated obstacle line segments can also be applied to the case of dilated polygonal obstacles. We can partition the dilated polygonal obstacles into those inside and outside C using the same pseudo-partitioning procedure as in Section 6, and proceed with the geometric operations for finding feasible trajectories as in Section 5.

In the process, we can also exploit the algorithm with respect to the number of polygonal obstacles (see Lemma 6.1). Given the similarity in analysis, we omit the details herein. The resulting space and time complexities of the subroutines involved are summarized below.

- Case I. Feasible unarticulated probe trajectory: $O(n + h \log h)$ time.
- Case II. Feasible articulated probe trajectory:
  - Subcase 1: $O(n_{in} + h_{in} \log h_{in}) + O(h_{in} n_{out} + h_{in} h_{out} \log h_{out}) + O(h_{in} n_s + h_{in} h_s \log h_s) + O(h_{in} h_{out} \log h_s)$ time.
  - Subcase 2: $O(n^2) + O(n_{in} + h_{in} \log h_{in}) + O(h_{out}^2 \log h_{in})$ time.
  - Circular arc intersection queries: $O(n_s 2^{\alpha(n_s)})$ space, $O(n_s \alpha(n_s) \log n_s)$ preprocessing time, $O(\log n_s)$ query time.
  - Circular sector emptiness queries: $O(n_s \alpha(n_s) \log n_s)$ preprocessing space/time, $O(\log n_s)$ query time.

Recall that, in the worst case, the number of queries to be processed is bounded by $O(h_{in} h_{out} + h_{out}^2)$. Hence, after some simplification, the final result can be stated as the following theorem.

Theorem 7.1. *A δ-clearance probe trajectory amidst h simple polygonal obstacles with n vertices can be determined in $O(n^2 + h^2 \log n)$ time using $O(n^2)$ space.*

8. Conclusion

We have presented efficient geometric-combinatorial algorithms for several variants of a novel trajectory planning problem involving a simple articulated probe. Specifically, we can determine a feasible probe trajectory, with or without a given clearance, among line segment obstacles in $O(n^2 \log n)$ time. In addition, the algorithms have been extended to the case of polygonal obstacles, where we can find a feasible solution in $O(n^2 + h^2 \log n)$ time. In any case, our main algorithm has essentially reduced to special cases of the circular sector intersection problem, for which we have provided near-linear time solutions. Notice that, while the proposed algorithm for finding a feasible probe trajectory of a given clearance from obstacles has the same time complexity as when no clearance is required, the space usage increases from $O(n \log n)$ to $O(n^2)$.

A number of open problems remain: (1) Our main algorithm works by enumerating over a set of possible "extremal" solutions. Can it be sped up, possibly by skipping some of these solutions? (2) We believe that it is possible to reduce the working space of the visibility complex to $O(n)$ by using topological sweep. That being so, can the space usage of the special circular sector intersection queries be reduced to $O(n)$ as well? We conjecture that our result would then be optimal. (3) Can we find an efficient general data structure for circular arc ray shooting queries among (disjoint or intersecting) line segments or circular arcs?

References

[1] O. Daescu, K. Fox, K. Y. Teo, Trajectory planning for an articulated probe, in: 30th Annual Canadian Conference on Computational Geometry, 2018, pp. 296–303.

[2] O. Daescu, K. Fox, K. Y. Teo, Computing trajectory with clearance for an articulated probe, in: 28th Annual Fall Workshop on Computational Geometry, 2018.

[3] R. Connelly, E. D. Demaine, Geometry and topology of polygonal linkages, Handbook of Discrete and Computational Geometry (2017) 233–256.

[4] J. Hopcroft, D. Joseph, S. Whitesides, Movement problems for 2-dimensional linkages, SIAM Journal on Computing 13 (3) (1984) 610–629.

[5] S. M. LaValle, Planning algorithms, Cambridge University Press, 2006.

[6] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, S. Thrun, Principles of robot motion: theory, algorithms, and implementation, MIT Press, 2005.

[7] F. Avnaim, J. Boissonnat, Practical exact motion planning of a class of robots with three degrees of freedom, in: Proceedings of the Canadian Conference on Computational Geometry, 1989, p. 19.

[8] K. Kedem, M. Sharir, An automatic motion planning system for a convex polygonal mobile robot in 2-dimensional polygonal space, in: Autonomous robot vehicles, Springer, 1990, pp. 349–362.

[9] D. Halperin, Robust geometric computing in motion, The International Journal of Robotics Research 21 (3) (2002) 219–232.

[10] L. Kavraki, P. Svestka, J. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580.

[11] S. M. LaValle, J. J. Kuffner Jr, Randomized kinodynamic planning, The International Journal of Robotics Research 20 (5) (2001) 378–400.

[12] R. A. Brooks, T. Lozano-Perez, A subdivision algorithm in configuration space for findpath with rotation, IEEE Transactions on Systems, Man, and Cybernetics SMC-15 (2) (1985) 224–233.

[13] B. R. Donald, Motion planning with six degrees of freedom, Tech. rep., MIT Artificial Intelligence Lab (1984).

[14] D. Zhu, J. Latombe, Constraint reformulation in a hierarchical path planner, in: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE, 1990, pp. 1918–1923.

[15] J. Barraquand, L. Kavraki, J. C. Latombe, R. Motwani, T. Y. Li, P. Raghavan, A random sampling scheme for path planning, The International Journal of Robotics Research 16 (6) (1997) 759–774.

[16] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, D. Vallejo, OBPRM: An obstacle-based PRM for 3D workspaces, in: Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics, 1998, pp. 155–168.

[17] D. Hsu, J. Latombe, R. Motwani, Path planning in expansive configuration spaces, International Journal of Computational Geometry & Applications 9 (4 & 5) (1999) 495–512.

[18] J. H. Yakey, S. M. LaValle, L. E. Kavraki, Randomized path planning for linkages with closed kinematic chains, IEEE Transactions on Robotics and Automation 17 (6) (2001) 951–958.

[19] N. Simaan, R. M. Yasin, L. Wang, Medical technologies and challenges of robot-assisted minimally invasive intervention and diagnostics, Annual Review of Control, Robotics, and Autonomous Systems 1 (2018) 465–490.

[20] E. Arkin, J. Mitchell, An optimal visibility algorithm for a simple polygon with star-shaped holes, Tech. rep., Cornell University Operations Research and Industrial Engineering (1987).

[21] P. J. Heffernan, J. S. Mitchell, An optimal algorithm for computing visibility in the plane, SIAM Journal on Computing 24 (1) (1995) 184–201.

[22] S. Suri, J. O'Rourke, Worst-case optimal algorithms for constructing visibility polygons with holes, in: Proceedings of the Second Annual Symposium on Computational Geometry, ACM, 1986, pp. 14–23.

[23] M. Sharir, P. K. Agarwal, Davenport-Schinzel sequences and their geometric applications, Cambridge University Press, 1995.

[24] J. Hershberger, Finding the upper envelope of n line segments in $O(n$ log $n)$ time, Information Processing Letters 33 (4) (1989) 169–174.

[25] N. M. Benbernou, M. Ishaque, D. L. Souvaine, Data structures for restricted triangular range searching, in: 20th Annual Canadian Conference on Computational Geometry, 2008, pp. 15–18.

[26] O. Daescu, N. Mi, C. Shin, A. Wolff, Farthest-point queries with geometric and combinatorial constraints, Computational Geometry 33 (3) (2006) 174–185.

[27] G. S. Brodal, R. Jacob, Dynamic planar convex hull, in: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002, pp. 617–626.

[28] J. Hershberger, S. Suri, Off-line maintenance of planar configurations, Journal of Algorithms 21 (3) (1996) 453–475.

[29] J. Matoušek, Range searching with efficient hierarchical cuttings, Discrete & Computational Geometry 10 (2) (1993) 157–182.

[30] M. Pocchiola, G. Vegter, The visibility complex, International Journal of Computational Geometry & Applications 6 (3) (1996) 279–308.