Trajectory Planning for an Articulated Probe

Ovidiu Daescu* Kyle Fox* Ka Yaw Teo*

Abstract

We present a geometric-combinatorial algorithm for computing a feasible solution for 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 (infinite) length. Initially, ab and bc are collinear. Given a set of obstacles represented as 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 line segments (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). In the process, 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 in the plane.

1 Introduction

Consider the following *trajectory (or motion) planning problem*. An articulated needle-like probe is modeled in $\Re^2$ as two line segments, ab and bc, joined at point b. Line segment bc may rotate at point b. The length of line segment ab can be arbitrarily large (infinitely long), 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 contains a set of n disjoint line segment obstacles P (see Figure 1). Let t be a point in the free space (i.e., inside S and outside the obstacles).

In the beginning, the probe assumes a straight configuration, that is, line segments ab and bc are collinear, with $b \in ac$. We call this an *unarticulated* configuration. 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, line segment bc may be rotated at point b up to $\pi/2$ radians in either direction, provided that line segment bc does not collide with any obstacle. If a rotation is performed, then we have an *articulated* configuration of the probe.

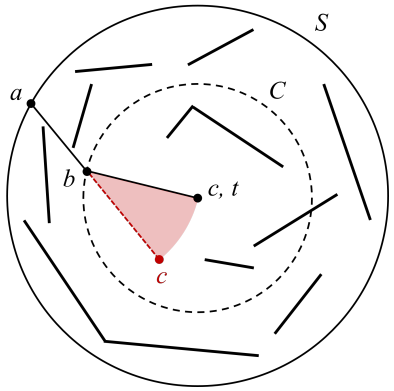

Figure 1: Trajectory planning for an articulated probe. After a straight insertion of line segment abc, in order to reach point t in the midst of obstacles, line segment bc may be required to rotate from its intermediate position (red dashed line) to the final position (black solid line).

A *feasible probe trajectory* consists of an initial insertion of straight line segment abc, possibly followed by a rotation of line segment bc at point b, such that point c ends at the target point t, while avoiding 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 for this problem is known. Possible extensions of the problem include reporting the space of all feasible probe trajectories and finding feasible probe trajectories of *maximum clearance*, although these extensions are beyond the scope of this extended abstract.

Because bc may only rotate up to $\pi/2$ radians, it is an easy observation that for any feasible probe trajectory, point b is the first intersection of segment ab with a circle C of radius r centered at point t. As illustrated in Figure 1, segment bc may rotate about point b, and the area swept by segment bc is a sector of a circle (a portion of a disk enclosed by two radii and an arc), with the center located on C, radius r, and the endpoint of one of its two 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*.

*Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA. {daescu, kyle.fox, ka.teo}@utdallas.edu

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 [4, 9] to application-driven robotic arm modeling and motion planning problems [10].

In contrast to those previous studies, 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.

Motivation

The problem setting described in the current study has practical relevance in the field of robotics, particularly in minimally invasive robotic surgery, 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 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 Lemma 1 below).

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 fixed endpoint t).

Our algorithm for articulated probe trajectory planning can be extended for polygonal obstacles, where we can exploit output sensitive algorithms with respect to the number of polygons and the complexity of the visibility (to infinity) from a given point.

2 Main observation

Our algorithm crucially depends upon the following observation. It immediately implies that it suffices to test a finite number of probe trajectories for feasibility. We refer to these trajectories as *extremal*.

Lemma 1 *There exists a feasible probe trajectory such that the probe assumes either I: an **unarticulated** final configuration (i.e., it is a straight line segment abc with $c = t$) that intersects an obstacle endpoint, or II: an **articulated** final configuration (i.e., line segments ab and bc are not colinear and $c = t$) that intersects an obstacle endpoint outside C and another obstacle endpoint inside or outside C.*

Proof. The existence of feasible probe trajectories for case I and II can be proven using simple perturbation arguments. The full proof of Lemma 1 is given in the Appendix. □

3 Solution approach

Based on the observation stated in Lemma 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 $n_{in} = |V_{in}|$, $n_{out} = |V_{out}|$. 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, and the set R of rays can be obtained by computing the visibility polygon from point t in $O(n \log n)$ time [1, 6, 15].[1]

Lemma 2 *The set of extremal feasible unarticulated probe trajectories can be determined in $O(n \log n)$ time.*

[1] For our case of disjoint line segments or even polygonal obstacles, we could use the *visibility complex* to compute R. The visibility complex is a two-dimensional subdivision in which each cell corresponds to a collection of rays with the same visibility properties [12]. For a simple scene of polygonal obstacles with a total of n vertices, the visibility complex can be computed in $O(n \log n + m)$ time using $O(m)$ space, where m is the size of the visibility complex (or the corresponding visibility graph) [13]. In the worst case, $m = O(n^2)$. After the visibility complex is built, one can compute the view from a point in time $O(k \log n)$, where k is the size of the computed view (from which the "visible" tangents or rays can be reported).

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 considered 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 endpoint inside C, we 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 visibility polygon of t used in Case I, R_{in} can be obtained in $O(n_{in} \log n_{in})$ time (alternatively, we can actually extract R_{in} directly from the visibility polygon constructed for Case I in $O(n)$ additional time).

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 done in $O(n_{out} \log n_{out})$ time (we could get rid of the $O(\log n)$ factor with some care, using duality, while treating all such b points at once).

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, that 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 $O(n_{in} n_{out} \log n_{out} + n_{in} \log n_{in})$.

Subcase 2. In order to find a feasible probe trajectory with an articulated final configuration that intersects two endpoints outside C, we determine a feasible intermediate configuration (i.e., the probe configuration after inserting straight line segment abc into S and before rotating line segment bc) using the following procedure.

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. Again, R can be obtained in $O(n \log n)$ time by computing the visibility polygon of v. 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 $O(n_{out} n \log n)$.[2]

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 some combination of the three basic scenarios depicted in Figure 2.

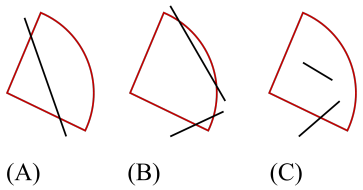

Figure 2: Basic scenarios of a line segment intersecting with a circular sector. (A) The segment intersects both radii of the sector. (B) The segment intersects the sector's circular arc at least once. (C) At least one segment endpoint lies inside the sector.

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, the basic scenario in Figure 2A can be eliminated from consideration.

Hence, our case of the circular sector intersection problem reduces to the following two problems:

1. Circular arc intersection query – for detecting the basic scenario in Figure 2B.
2. Circular sector emptiness query – for detecting the basic scenario in Figure 2C.

[2] As before, we can handle subcases 1 and 2 using the visibility complex approach.

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

Problem 1 *Given a set P of n line segments, 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 P.*

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

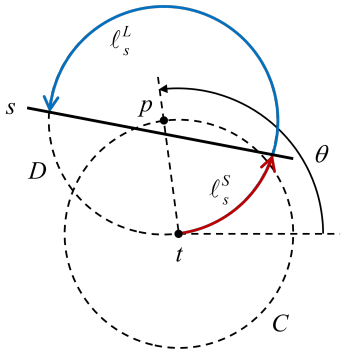

Figure 3: Counter-clockwise circular arcs ℓ_s^S and ℓ_s^L emanating from point t.

Let D be a circle of radius r centered at any point $p \in C$ (Figure 3). Note that circle D passes through the center t of circle C. We let θ be the angle of tp relative to the x-axis; D is uniquely determined by θ since we know p lies on C. We will 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 will define two partial functions $\ell_s^S, \ell_s^L : [0, 2\pi) \to \mathbb{R}_{\geq 0}$. Let $\theta \in [0, 2\pi)$ and let D be the circle for that θ as defined above. If D intersects h_s and the first intersection lies on s, let $\ell_s^S(\theta)$ be the length of the counter-clockwise arc from t to that first intersection. Otherwise, $\ell_s^S(\theta)$ is undefined. Similarly, if D has a second intersection with h_s on s, let $\ell_s^L(\theta)$ be the length of the arc to that intersection. We easily observe the following properties of ℓ_s^S (Figure 4). The same statements apply to ℓ_s^L as well.

Property 1 *Function ℓ_s^S is defined over at most two maximal contiguous subsets of $[0, 2\pi)$.*

Property 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 the value of θ for which D's shorter counter-clockwise arc ends at the intersection of s_i and s_j. Because s_i and s_j can only intersect at their endpoints, $\ell_{s_i}^S(\theta) = \ell_{s_j}^S(\theta)$ only at the endpoints of the maximal contiguous subsets of $[0, 2\pi)$ for which $\ell_{s_i}^S$ and $\ell_{s_j}^S$ are defined.*

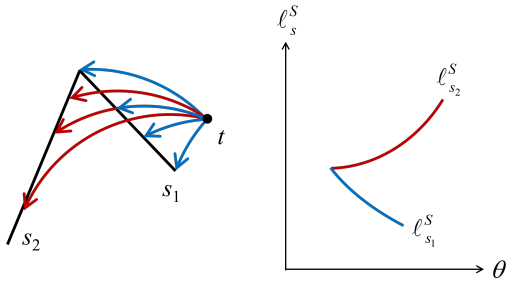

Figure 4: Illustration of the properties of function ℓ_s^S. A given line segment s induces a partially-defined piecewise continuous curve l_s^S. Two curves $\ell_{s_i}^S$ and $\ell_{s_j}^S$ intersect (at most once) if and only if their corresponding line segments s_i and s_j intersect (e.g., s_1 and s_2).

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 [14]. The lower envelope can be found by a worst-case optimal divide-and-conquer algorithm running in $O(n \log n)$ time [7]. Let V^S be the lower envelope of the curves $\ell_{s_i}^S$ for all given line segments $s_i \in P$. Given the properties of each $\ell_{s_i}^S$ (in particular that the curves intersect at endpoints), the size of lower envelope V^S is actually bounded by the second order Davenport-Schinzel sequence, which is $O(n)$, and we can again compute it in $O(n \log n)$ time. 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 which lies on a lower envelope at θ. Thus, we obtain the following result, which can be easily shown to be worst case optimal (see [15]).

Lemma 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.*

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

Problem 2 *Given a set P of n points in the plane preprocess it so that, for a query circular sector σ of fixed radius r whose arc contains a fixed point 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 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 1 *Given a set P of n points in the plane, preprocess it so that, for a query triangle $\triangle$ with a vertex incident on a fixed point t, one can efficiently determine whether $\triangle$ contains any point of P.*

As proposed by Benbernou et al. [2], Subproblem 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. [5] also used a similar idea to build a data structure for halfplane farthest-point queries.

The result for Subproblem 1 is summarized in the following lemma.

Lemma 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 2 *Given a set P of n points in the plane, preprocess it so that for a query circular segment s, bounded by a circular arc originating from a fixed point t and the chord connecting the endpoints of the arc, one can efficiently determine if s contains any point of P.*

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 5. 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.

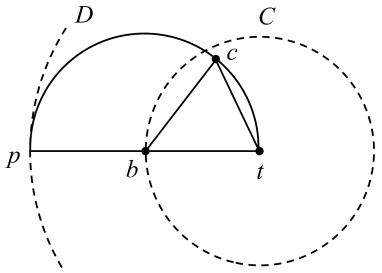

Figure 5: Circular segment s_{ct} and its "enclosing" circular segment s_{pt}.

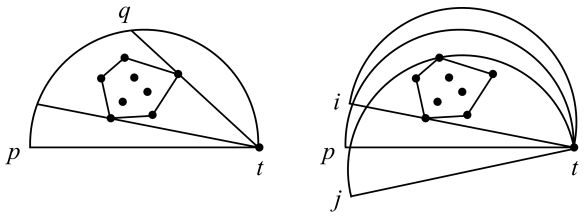

Figure 6: Circular segment s_{pt} and its corresponding event interval indicated by $[i, j]$.

Let $P_{pt} \subseteq P$ be the set of points in s_{pt}, and $CH(P_{pt})$ be the convex hull of P_{pt}. As shown in Figure 6, 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 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 about 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 6, right). Notice that, the set of points of P in s_{pt} remains constant within this event interval. For each of these event intervals, the set of points of interest, their convex hull, and ultimately point q can be determined by using a dynamic convex hull data structure [3, 8], 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 point q for each event interval.

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 and its associated point q in $O(\log n)$ time. If the endpoint c of s_{ct} is located within arc qt, then s_{ct} does not contain any point of P.

Lemma 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 2.

Lemma 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 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 more general circular sector emptiness problem without a fixed radius or point t on the arc using a multilevel data structure similar to one by Matoušek [11] for counting points in the intersection of halfspaces (see also [5] 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 [11, Theorem 6.2], we can create a tradeoff between space and time usage by our data structure.

Lemma 7 *A set P of n points 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 is void of P in $O(n/m^{1/2} \log^{5/2} n)$ time for any $n^{1+\epsilon} \leq m \leq n^2$.*

Finishing up. According to Lemmas 3 and 6, the result for our case of the circular sector intersection problem can be stated as follows.

Lemma 8 *For a positive value r and a fixed point t, a set of 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 radius r whose circular arc has an endpoint at t, one can determine whether σ intersects P in $O(\log n)$ time.*

Let n_s be the number of endpoints of 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 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 9) is dominant over that of Case I (Lemma 2), the solution approach proposed herein for finding a feasible probe trajectory leads to the following theorem.

Theorem 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 space usage and running time are bounded by $O(n \log n)$ and $O(n^2 \log n)$, respectively.

5 Conclusion

We presented an efficient geometric-combinatorial algorithm for a novel trajectory planning problem involving a simple articulated probe. Specifically, we can determine a feasible probe trajectory in $O(n^2 \log n)$ time. Our algorithm reduced to special cases of the circular sector intersection problem, for which we provided solutions.

A number of open problems remain: (1) Our algorithm works by enumerating over a set of possible "extremal" solutions. Can it be sped up, possibly by skipping some of these solutions? (2) Can the algorithm be extended to find a representation of all feasible probe trajectories? (3) Can the space usage of the special circular sector queries be reduced to $O(n)$? We conjecture that our result would then be optimal. (4) Can we find an efficient general data structure for circular arc ray shooting queries among (disjoint or intersecting) line segments?

Acknowledgements. The authors would like to thank Pankaj K. Agarwal and Carola Wenk for helpful discussions and pointers to related literature.

References

[1] E. Arkin and J. Mitchell. An optimal visibility algorithm for a simple polygon with star-shaped holes. Technical report, Cornell University Operations Research and Industrial Engineering, 1987.

[2] N. M. Benbernou, M. Ishaque, and D. L. Souvaine. Data structures for restricted triangular range searching. In *20th Annual Canadian Conference on Computational Geometry*, pages 15–18. Citeseer, 2008.

[3] G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 617–626. IEEE, 2002.

[4] R. Connelly and E. D. Demaine. Geometry and topology of polygonal linkages. *Handbook of discrete and computational geometry*, pages 233–256, 2017.

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

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

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

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

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

[10] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

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

[12] M. Pocchiola and G. Vegter. The visibility complex. *International Journal of Computational Geometry & Applications*, 6(03):279–308, 1996.

[13] S. Rivière. Topologically sweeping the visibility complex of polygonal scenes. In *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, pages 436–437. ACM, 1995.

[14] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge university press, 1995.

[15] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proceedings of the Second Annual Symposium on Computational Geometry*, pages 14–23. ACM, 1986.

Appendix

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 unobstructed vision to some points on the bounding circle S. Let T' be the trajectory resulting from rotating T about point t in clockwise direction until T intersects an obstacle endpoint v. It is apparent that T' is also a feasible trajectory, and its articulation point b' is the intersection of segment vt and circle C.

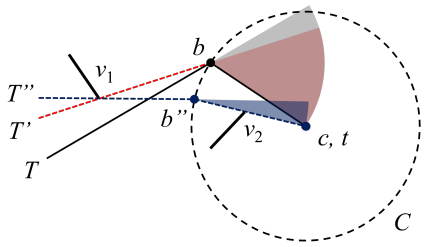

Figure 7: Case II of Lemma 1. T'' represents a feasible articulated probe trajectory such that the final configuration of the probe intersects an obstacle endpoint v_1 outside C and an obstacle endpoint v_2 inside C.

For case II, assume that a feasible 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 7). 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 about 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' is within that of T (indicated by the shaded circular sectors in Figure 7), T' is also a feasible trajectory.

Now, let T'' be the trajectory obtained by rotating line segment bc of T' about point t in counter-clockwise direction while simultaneously rotating ab around v_1 in the clockwise direction until either abc becomes a line segment or either ab or bc intersects some obstacle endpoint v_2. Note that as T' changes into T'' point b of T' slides on circle C in counter-clockwise direction into a new position b''. If abc becomes a line segment, we have achieved case I of the lemma. We now assume otherwise.

Observe that every point of the circular sector centered at b'' lies on the t side of the line through v_1 and b. They also lie on the b side of the line through b'' and t. 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 the sector centered at t is empty, because it was swept while constructing T''. We now argue 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. Because 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

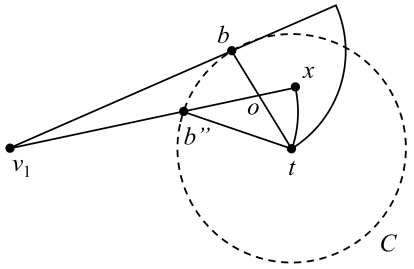

Figure 8: Point x lies inside the circular sector centered at b.

lies in the wedge at b. Let o be the intersection of the line segments bt and $b''x$ (Figure 8). By 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''.