α-Visibility*

Mohammad Ghodsi[1], Anil Maheshwari[†2], Mostafa Nouri-Baygi[1], Jörg-Rüdiger Sack[‡2], and Hamid Zarrabi-Zadeh[1]

[1]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.
[2]School of Computer Science, Carleton University, Ottawa, ON, Canada.

October 2, 2013

Abstract

We study a new class of visibility problems based on the notion of *α-visibility*. Given an angle α and a collection of line segments $\mathcal{S}$ in the plane, a segment t is said to be α-visible from a point p, if there exists an empty triangle with one vertex at p and the side opposite to p on t such that the angle at p is α. In this model of visibility, we study the classical variants of point visibility, weak and complete segment visibility, and the construction of the visibility graph. We also investigate the natural query versions of these problems, when α is either fixed or specified at query time.

1 Introduction

The study of visibility is at least 100 years old, when in 1913 Brunn [6] proved a theorem about the kernel of a set. By now, visibility has become one of the most studied notions in computational geometry. The reasons are two-fold: 1) such problems arise naturally in areas where computational geometry tools and algorithms find applications including: computer graphics, robotics, motion planning, geographic information systems, computer games, computer-aided architecture, and pattern recognition; and where 2) their solutions are required, or serve as building blocks in the development of solutions to other problems, such as shortest paths or motion planning problems. Many natural problem instances arise and have been extensively studied in two and higher dimensions. The reader is referred to the survey article by Asano *et al.* [4] and the book by Ghosh [13] for more details. We review some of that body of work, as relevant to this paper.

Previous Work. Given a polygonal scene $\mathcal{S}$, the *visibility polygon* of a point p, denoted by $\mathrm{VP}(p)$, is the set of all points inside the scene that are visible from p. When the scene is a simple polygon or a polygonal domain, several algorithms exist to compute the visibility polygon of a point with/without preprocessing. Previous results for point-visibility inside a scene are summarized in Table 1.

*A preliminary version of this paper was presented at the 13th Scandinavian Symposium on Algorithm Theory, Lecture Notes in Computer Science 7357: 1–12, 2012.

†Research supported by NSERC.

‡Research supported by NSERC, SUN Microsystems and HPCVL.

Scene	Prep. Time	Space	Query Time	Ref.
polygon	—	$O(n)$	$O(n)$	[4],[11],[21]
polygon	$O(n^3 \log n)$	$O(n^3)$	$O(\log n + m_p)$	[5]
polygon	$O(n^3)$	$O(n^3)$	$O(\log n + m_p)$	[15]
polygon	$O(n^2 \log n)$	$O(n^2)$	$O(\log^2 n + m_p)$	[1]
polygonal domain	—	$O(n)$	$O(n \log n)$	[2],[28]
polygonal domain	—	$O(n)$	$O(n + h \log h)$	[17]
polygonal domain	$O(n^2)$	$O(n^2)$	$O(n)$	[3]
polygonal domain	$O(n^2 \log n)$	$O(n^2)$	$O(m_p \log(n/m_p))$	[29]
polygonal domain	$O(n^3 \log n)$	$O(n^3)$	$O(\min\{h, m_p\} \log n + m_p)$	[30]
polygonal domain	$O(s \log(\sqrt{s}/n))$	$O(s)$	$O(n^2 \log(\sqrt{s}/n)/\sqrt{s} + m_p)$	[25]
polygonal domain	$O(n^2 \log n)$	$O(n^2)$	$O(\min\{h, m_p\} \log^2 n + h + m_p)$	[19]
polygonal domain	$O(T + \lvert E\rvert + n \log n)$	$O(\min(\lvert E\rvert, hn) + n)$	$O(m_p \log n + h)$	[19]
convex polygons	$O(n \log n)$	$O(n)$	$O(m_p \log n)$	[27]

Table 1: Summary of previous work on point-visibility. Here, n is the total complexity of the scene, h is the number of holes, s is a parameter satisfying $n^2 \leqslant s \leqslant n^4$, T is the time for triangulating the scene, m_p is the complexity of $\mathrm{VP}(p)$ for a query point p, and $|E|$ is the number of edges in the visibility graph of the scene.

Given a segment s, the *weak visibility polygon* $\mathrm{VP}(s)$ of s is the set of points in the scene that are visible from at least one point on s. Guibas *et al.* [16] showed how to compute the weak visibility polygon of a segment inside a simple polygon in $O(n)$ time. Suri and O'Rourke [28] established that the weak visibility polygon of a segment inside a polygon with holes has size $\Theta(n^4)$ in the worst case, but can be represented by a set of $O(n^2)$ triangles. They also gave an algorithm for computing the weak visibility polygon of a segment inside a polygon with holes which runs in $O(n^4)$ time.

The *visibility graph* of a polygon is the undirected graph of the visibility relation on the vertices of the polygon. The visibility graph construction is motivated, e.g., by Lozano-Perez's algorithm for finding a shortest path between two points which avoids all polygonal obstacles. Optimal algorithms for computing visibility graphs exist. Hershberger [18] showed how to construct the visibility graph in a simple polygon in $O(|E|)$ time. Whereas, Ghosh and Mount [12] established its construction in $O(n \log n + |E|)$ time for a polygon with holes. Here, $|E|$ is the number of edges in the resulting visibility graph.

The *weak visibility graph* of a set of segments is defined as the graph, with a node for each segment and an edge between any pair of weak visible segments, that have at least two mutual visible points. Ghosh and Mount [12] and Keil *et al.* [20] computed the weak visibility graph in $O(n \log n + |E|)$ time. Nouri-Baygi *et al.* [26] demonstrated how to detect the visibility between two query segments in $O(n^{1+\varepsilon})$ time, using $O(n^2)$ space and $O(n^{2+\varepsilon})$ preprocessing time, for any fixed positive ε. Gudmundsson and Morin [14] obtained a result for testing weak visibility between a query point and a segment. They give a data structure of size $O(k)$ that can test if a query point is visible from a segment s, known in advance, in $O(n^{\varepsilon} m_s/\sqrt{k})$ time. In the above bounds, m_s is the number of edges of the *extended visibility graph* of the scene incident on s, k is a parameter satisfying $m_s \leqslant k \leqslant m_s^2$, and ε is any fixed positive number.

The extended visibility graph of a scene $\mathcal{S}$, denoted by $\mathrm{EVG}(\mathcal{S})$, is obtained by adding edges and vertices to the visibility graph as follows: For each vertex v and each edge uv in the visibility graph, extend the segment uv in direction $\overrightarrow{uv}$ until it intersects an element of $\mathcal{S}$ at some point w,

and add w as a vertex and vw as an edge to EVG($\mathcal{S}$).

New model. In this paper, we study a new class of visibility problems based on the notion of *α-visibility* defined as follows. Let $\mathcal{S}$ be a set of n line segments in the plane, which are non-intersecting except possibly at their end-points. (Since each polygonal scene is composed of a set of segments, $\mathcal{S}$ can model polygonal scenes as well.) Let α be a positive real number.

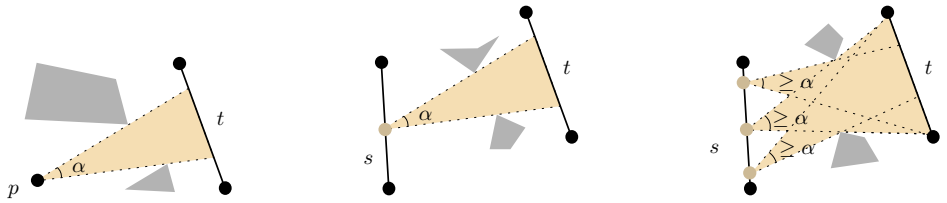

Figure 1: (a) Segment t is α-visible from p. (b) Segment t is weakly α-visible from segment s. (c) Segment t is completely α-visible from segment s.

- *Point-visibility.* A segment $t \in \mathcal{S}$ is said to be α-visible from a point p, if p can see t with an angle at least α; that is, if there exists an empty triangle with one vertex at p and side opposite to p on t such that the angle at p equals α (see Fig. 1a).
- *Segment-visibility.* A segment t is said to be *weakly α-visible* from a segment s, if there is a point on s from which t is α-visible (see Fig. 1b). A segment t is said to be *completely α-visible* from s, if for all points on s, t is α-visible (see Fig. 1c).
- *Visibility graph.* We define the *weak* (respectively *complete*) *α-visibility graph* of $\mathcal{S}$ as a directed graph G_α whose vertices are the segments of $\mathcal{S}$, and for any two segments $s, t \in \mathcal{S}$, there is a directed edge from s to t if t is weakly (respectively, completely) α-visible from s.

The notion of *α-visibility* appears to be natural. For example, the smallest angle we can observe directly is determined by the ratio of the wavelength of light to the diameter of the eye's pupil, which is lower bounded by a constant (10^{-4} radians). So, we can see a hair held at about arm's length, but would not be able to see it at double that distance. As such, objects that are too small, or too far, are not visible to human eyes. All optical/digital imaging devices have similar limitations, quantified by their resolutions. Our α-visibility model is not meant to model the eye or these devices accurately, but is inspired by these limitations and may provide a realistic alternative to the classical visibility models studied in the computational geometry literature. The value α could be also employed to approximate the inaccuracy of a device used to provide visibility-related measurements. For robot motion planning, De Berg *et al.* [9] and Cheong and van Oostrum [8] considered a model of uncertainty in which the direction of movement for the robot is confined to a cone with angle α centered around the specified direction.

In general, there is a wealth of literature on approximation algorithms for several geometric shortest path problems, and there is a very close connection between visibility and shortest path problems, but still there are essentially no results on the notion of approximate visibility. This paper lays a foundation in that respect, and will likely inspire further study of the notion of approximate visibility.

Our results. In this paper, we present some of the first results for several variants of the point/segment visibility problems in the α-visibility model. The main idea that we use is to round the set of all possible visibility directions into $O(1/\alpha)$ directions, and then use a combination of geometric tools, such as trapezoidal diagrams, shortest path maps, ray shooting, range searching, etc., to solve the visibility problem in each rounded direction. A summary of the results obtained in this paper is provided below. In the following, $\mathcal{S}$ denotes the set of input segments in the plane.

- We present efficient data structures that enable answering queries of the form "Is segment $t \in \mathcal{S}$ α-visible from a query point p in the plane?". When α is fixed, we preprocess $\mathcal{S}$ in $O(n \log n)$ time into a data structure of size $O(n)$ that answers aforementioned point-visibility queries in $O(\log n)$ time[1]. We also provide data structures for answering point-visibility queries when α is specified at query time.
- We show that the size of the (weak and complete) α-visibility graph of $\mathcal{S}$ is linear in n, and that, the weak/complete visibility graph can be computed in $O(n \log n)$ time. As a byproduct, we can answer queries of the form "Is $t \in \mathcal{S}$ weakly/completely α-visible from $s \in \mathcal{S}$?" in $O(1)$ time, after $O(n \log n)$ preprocessing time.
- We show how to preprocess $\mathcal{S}$ in $O(n \log n)$ time into a data structures of size $O(n)$ such that queries of the form "Is segment $t \in \mathcal{S}$ weakly α-visible from a query segment s in the plane?" can be answered in $O(\log n)$ time.

Note that one of the key differences between standard visibility (i.e., when $\alpha = 0$) and α-visibility lies in the size of the weak/complete visibility graph of the line segments in the plane. While the former has quadratic size, the latter is linear in size, which makes it appealing both theoretically and from an applied perspective when dealing with large data sets. Furthermore, unlike standard visibility, α-visibility is not symmetric, so the weak/complete α-visibility graph of a set of line segments is directed.

1.1 Paper Organization

The rest of the paper is organized as follows. In Section 2, we give brief explanations about required data structures and algorithms. In Section 3, α-visibility from a point is considered and two query problems are investigated. The weak and complete α-visibility from a segment are studied in Section 4 and 5, respectively. In Section 6, we show that our results on weak and complete α-visibility from a segment do not extend to 3D.

2 Preliminaries

In the section, we briefly introduce some of the geometric tools that will be used throughout the paper.

[1]The running times and space bounds of the data structures presented in this paper involve a factor $1/\alpha$, which is omitted when α is assumed to be a fixed constant.

Multi-Level Range Searching. A range searching problem in the plane has the following form: Given a set of n points, build a data structure that, for any query triangle R, reports (or counts the number of) points lying in R quickly. In this paper, we will use a nested range searching data structure to solve more complex problems. We employ the most recent result on multi-level range searching due to Chan [7]:

Theorem 1 (Chan [7]). *Given n points in $\mathbb{R}^d$, we can form $O(n)$ canonical subsets of total size $O(n \log n)$ in $O(n \log n)$ time, such that the subset of all points inside any query simplex can be reported as a union of disjoint canonical subsets C_i with $\sum_i |C_i|^{1-1/d} \leqslant O(n^{1-1/d} \log n)$ in time $O(n^{1-1/d} \log n)$ w.h.p.*

Here, w.h.p. means with probability at least $1 - 1/n^{c_0}$ for an arbitrarily large constant c_0. The above theorem can be applied repeatedly to solve other complex problems, while the complexities of preprocessing/query time and space increase by only a logarithmic factor per level.

Ray Shooting among Segments. A typical ray shooting problem in the plane has the following form: Given a set of n segments in the plane, build a data structure that, for any query ray r, reports the first segment intersected by r quickly. Chan [7] used his multi-level range searching data structure to obtain the following result:

Theorem 2 (Chan [7]). *Given a set of n line segments in the plane, there is a data structure requiring $O(n \log^3 n)$ preprocessing time and $O(n \log^2 n)$ space, such that one can find the first point of intersection between a query ray and the set in $O(\sqrt{n} \log^2 n)$ expected time.*

Simplified Trapezoidal Diagram. Given a set $\mathcal{S}$ of segments in the plane and a direction d, we define a subdivision of the plane such that, each region in the subdivision is the maximal region with the property that all points in that region see the same segment in direction d. We can construct this subdivision by drawing a line in the reverse direction of d, from each end-point of all segments of $\mathcal{S}$, until it meets another segment. We call this subdivision, the *simplified trapezoidal diagram* of $\mathcal{S}$ in direction d, and denote it by $\mathcal{T}_d(\mathcal{S})$ (see Fig. 2b).

Theorem 3. *Given a set $\mathcal{S}$ of n segments in the plane and a direction d, we can construct the simplified trapezoidal diagram $\mathcal{T}_d(\mathcal{S})$ by a plane-sweep in direction perpendicular to d in $O(n \log n)$ time and $O(n)$ space.*

Shortest Path Maps. For a point s in a simple polygon P, the *shortest path map*, SPM(s), is a partition of P into cells such that for all points t in a cell, the sequence of vertices of P along the shortest path from s to t is fixed. It is well-known that the complexity of SPM(s) is $O(n)$ and it can be built in $O(n)$ time [16], where n is the number of vertices in P. If we preprocess SPM(s) for point location, for a query point p we can find in $O(\log n)$ time, the last vertex of P in the shortest path from s to p, which can be thought of as the root associated to the cell containing p.

Ray Shooting in Splinegons. Splinegons (or informally curved polygons) are defined as generalizations of polygons [10]. A splinegon S is formed from a polygon P by replacing one or more edges of P with curved edges such that the region bounded by each curved edge and the segment joining its end-points is convex.

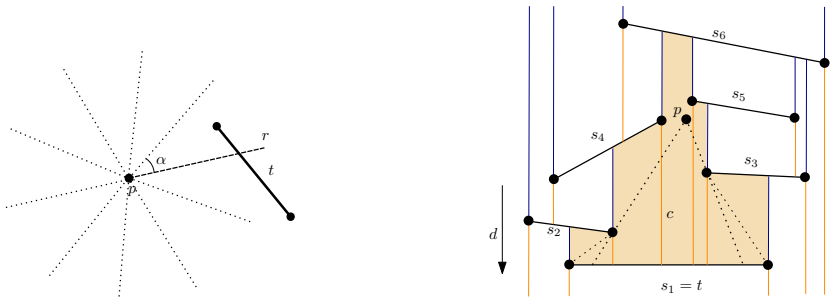

Figure 2: (a) Segment t is α-visible from p, and r intersects t. (b) A trapezoidation of a set of segments in direction d.

Theorem 4 (Melissaratos and Souvaine [23])**.** *Given a simple splinegon S with n edges, there is a data structure requiring $O(n)$ preprocessing time and $O(n)$ space, such that, for any query point p and a ray $\vec{r}$ emanating from p, the first intersection of $\vec{r}$ with the splinegon can be reported in $O(\log n)$ time.*

3 Point Visibility

3.1 Visibility testing for fixed α

Let $\alpha > 0$ be a fixed constant. In this section, we show how to build a data structure that efficiently determines, for a query point p in the plane and a query segment $t \in \mathcal{S}$, whether or not t is α-visible from p.

Theorem 5. *We can preprocess $\mathcal{S}$ into a data structure of size $O(n)$ in $O(n \log n)$ time, such that α-visibility testing can be carried out in $O(\log n)$ time.*

Proof. Assume that we have a set of $\lceil \frac{2\pi}{\alpha} \rceil$ rays emanating from p, as in Fig. 2a, so that the angle between any two consecutive rays is α (except possibly between a pair, where it is $\leqslant \alpha$). Let D denote the set of directions of these rays. If t is visible from p with an angle at least α, t must intersect one of the rays drawn from p as in Fig. 2a. Let r be a ray that intersects t and let d be the direction of r. Consider the trapezoidal diagram of $\mathcal{S}$ in direction d, as in Fig. 2b. Observe that p lies inside the trapezoid that sees t in direction d. Therefore, if t is α-visible from p, p must be inside a trapezoid that sees t, in the trapezoidal diagram drawn for $\mathcal{S}$, based on directions in D.

It remains only to be checked whether p sees t with an angle of at least α. Consider the simplified trapezoidal diagram $\mathcal{T}_d(\mathcal{S})$ in direction d. Note that p is inside a region c, whose visible segment in direction d is t (see Fig. 2b)[2]. For a point p in c, the shortest paths from p to the end-points of t, inside c, consist of two convex chains. The maximum visible part of t from p, including the intersection point of r with t, is determined by extending the first edge of each of the shortest paths. Therefore, to compute the maximum visible part of t from p, it is sufficient to find the first turning points on the shortest paths from p to the end-points of t, in c.

The complexity of the trapezoidal map in direction d is $O(n)$ and can be computed in $O(n \log n)$ time. We can use the trapezoidal map to locate the trapezoid containing p and find the visible

[2]Note that the region c is essentially a simple polygon.

segment in direction d. The shortest path map of the end-points of each segment t, in the corresponding cell c, has complexity proportional to the size of c. Since, the sum of the complexities of cells in the simplified trapezoidal map is $O(n)$, all shortest path maps can be computed in $O(n \log n)$ time using $O(n)$ space. We repeat this construction for all directions of D. To answer a query, for each direction d, we locate the trapezoid in which p lies, in the trapezoidal map of $\mathcal{S}$. This can be done in $O(\log n)$ time. The trapezoid gives us a segment that is visible in direction d. If the segment is not t we proceed to the next direction. Otherwise, based on the shortest path map associated to the cell of the simplified trapezoidal map, we can find the maximum portion of t that is visible from p, and check if that portion forms an angle of at least α with t. If so, we report "yes", otherwise we check the next direction. We can locate the first turning point in the shortest path from p to each end-point of t in $O(\log n)$ time, by finding in which region of the shortest path map of that point, p lies. Since the number of directions is $O(1/\alpha)$ (a constant), the total query time is $O(\log n)$. □

Corollary 1. *We can preprocess $\mathcal{S}$ into a data structure of size $O(n)$ in $O(n \log n)$ time, and report all α-visible segments from a query point p in $O(\log n)$ time. Furthermore, the number of such α-visible segments from any point p is a constant.*

Proof. For each direction d, we find the trapezoid in $\mathcal{T}_d(\mathcal{S})$ that contains p. In that trapezoid, we know the segment $s \in \mathcal{S}$ that is visible in direction d. Using the shortest path maps related to the region containing p in $\mathcal{T}_d(\mathcal{S})$, we check if s is visible with an angle at least α. If so, s is a segment in the answer. Because there are $O(1/\alpha)$ directions, the number of reported segments is $O(1/\alpha)$, which is a constant. Detecting α-visible segment in direction d takes $O(\log n)$ time, therefore the total time is $O(\log n)$. □

3.2 Visibility testing for non-fixed α

In this section we construct a data structure to answer the following type of query. The query consists of a point p, a segment $t \in \mathcal{S}$ and an angle $\alpha > 0$. We need to answer whether t is α-visible from p?

Theorem 6. *We can preprocess $\mathcal{S}$, in $O(n \log^3 n)$ time, into a data structure of size $O(n \log^2 n)$, such that we are able to detect α-visibility of query segment $t \in \mathcal{S}$ from a query point p in $O(\sqrt{n} \log^2 n)$ expected time.*

Proof. Note that here α is specified at query time. Like for the proof of Theorem 5, we assume that we have a set of $\lceil \frac{2\pi}{\alpha} \rceil$ rays emanating from p, and the angle between any two consecutive rays is at most α (Fig. 2a). If t is α-visible from p, then it is visible from p along the direction of at least one of these rays. Let r be such a ray. To check if the visible part around the intersection point x of r with t constitutes an angle $\geqslant \alpha$, we need to find the maximum visible part of t from p around x. Therefore, we need to solve the following two sub-problems: *i*) Which is the first segment of $\mathcal{S}$ that the ray r, originating at p, intersects? *ii*) If we rotate r around p, when does the visibility from p along r change?

Problem (*i*) is ray shooting among segments, which has already been discussed in Theorem 2. To solve Problem (*ii*), we use the simplex range searching data structure in Theorem 1 as follows. Let the end-points of segment t be a and b. Consider the two simplices, $\triangle pxa$ and $\triangle pxb$. Using Theorem 1, we can obtain the points in these two simplices as a collection of $O(\sqrt{n} \log n)$ canonical

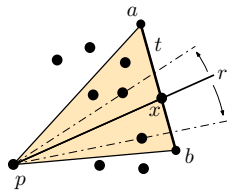

Figure 3: Finding the first point intersected by r while rotating counterclockwise.

subsets. As part of preprocessing, for each canonical set we will pre-compute the convex hull of its points. Hence, for each canonical set, using the convex hull, we can find the first point visited from that set, while we rotate r counterclockwise around p. The final result is the point that is visited first among all such points. Therefore, it can be found in $O(\sqrt{n}\log^2 n)$ expected time. The total complexity is now derived from the complexities of the two sub-problems. □

Remark: Using techniques introduced by [22], one can achieve a space/query-time tradeoff for the problem. Using $O(m)$ space, for any $n\log^2 n \leqslant m \leqslant n^2$, one obtains a query time of $O((n/\sqrt{m})\operatorname{polylog} m)$.

4 Segment Visibility

In this section, we turn our attention to problems related to segment visibility. We first study the weak α-visibility graph, G_α, for $\mathcal{S}$ defined as follows. Each segment of $\mathcal{S}$ is associated to a unique vertex in G_α. Furthermore, for any two segments $s, t \in \mathcal{S}$, if t is weakly α-visible from s, then there is a directed edge in G_α from the vertex corresponding to s to the vertex corresponding to t.

Lemma 1. *The weak α-visibility graph G_α of $\mathcal{S}$ has linear size.*

Proof. Let D be a set of $O(1/\alpha)$ uniformly distributed directions in $\mathbb{R}^2$. We say that $t \in \mathcal{S}$ is weakly visible from $s \in \mathcal{S}$ in direction $d \in D$, if there is point $p \in s$ that sees t in direction d. Clearly, if t is weakly visible from s in direction d, then s appears in the region bounded by t in the simplified trapezoidal map $\mathcal{T}_d(\mathcal{S})$ of $\mathcal{S}$, and hence, the number of such segments s is proportional to the size of the region (see Figure 2(b)). As a result, the total number of pairs (s, t) such that t is weakly visible from s in direction d is equal to the complexity of $\mathcal{T}_d(\mathcal{S})$, which is linear in n. Now, by definition, $t \in \mathcal{S}$ is weakly α-visible from $s \in \mathcal{S}$, if there is a point $p \in s$ that sees t with an angle at least α. Since the angle between any two adjacent directions in D is at most α, there is a $d \in D$ that lies inside the angle of view from p, and hence, t is weakly visible from s in direction d. Therefore, the total number of pairs (s, t) such that t is weakly α-visible from s is $O((1/\alpha)n) = O(n)$. □

Theorem 7. *We can preprocess $\mathcal{S}$ into a data structure of size $O(n)$ in $O(n\log n)$ time, so that weak α-visibility testing for two query segments $s, t \in \mathcal{S}$ can be carried out in $O(1)$ time.*

Proof. We start by computing G_α. Recall the set-up in the proof of Lemma 1. Assume that $t \in \mathcal{S}$ is weakly α-visible from a point p on segment $s \in \mathcal{S}$. Let d be the direction in D, that is inside the angle of view of p and r be the segment connecting p to t in direction d. Now slide r without changing its direction until it meets an end-point of a segment. The region in which r freely slides without intersecting any other segment and without changing its direction, specifies a strip w. This strip has the following property: Every point on s, on one side of w, sees a point on t, on the

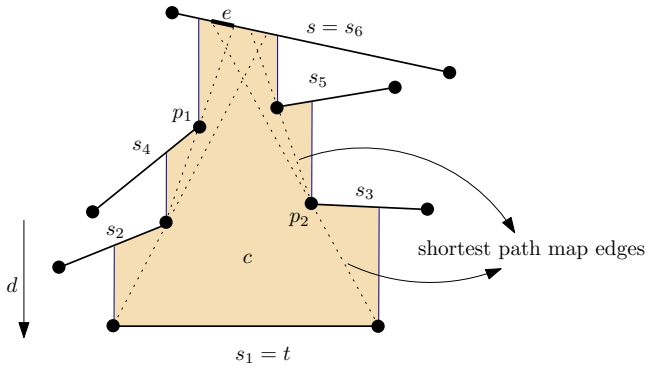

Figure 4: Segment s is partitioned into sub-segments.

opposite side of w, in direction d. Observe that w is a trapezoid in the trapezoidal map of $\mathcal{S}$, in direction d. We denote the nodes associated with s and t in the weak α-visibility graph as s^* and t^*, respectively. Therefore, the first condition that must be met for an edge from s^* to t^* is that s and t are two facing (i.e., opposite) edges of a trapezoid in one of the trapezoidal maps constructed for the different directions in D.

While this condition is necessary, it is not sufficient. To check whether t is actually weakly α-visible from s, we need to find a point on s that can see t with an angle at least α. If we find the point with the greatest view angle, and compare that angle with α, we can determine the α-visibility of t from s. We first partition s into sub-segments in which the boundary of the view angle passes through a unique pair of points. Then, for each sub-segment, we find the point with the greatest view angle which is one of these points.

Let c denote the region associated with t in $\mathcal{T}_d$. Let SPM(t_0) and SPM(t_1) denote the shortest path maps of the end-points of t in c. For each point p on s, the first turning points on the shortest paths to the end-points of t in c, form the view angle of p. We can partition s into sub-segments, such that for each sub-segment, the shortest paths to the end-points of t have the same set of turning points (i.e., combinatorially these shortest paths are identical). This is achieved by finding the intersection points of s with SPM(t_0) and SPM(t_1). The intersection points partition s into sub-segments. Let e be such a sub-segment on s (see Fig. 4). We want to locate the point on e that has the greatest view angle to t. Let p_1 and p_2 denote the first turning points in the shortest paths from any point on e to the end-points of t. The largest angle of view from each point on e towards t is determined by p_1 and p_2. Now, the problem reduces to finding a point on e with the maximum view through p_1 and p_2. This point is on the intersection of the smallest circle through p_1 and p_2 that intersects e. Therefore, if the circle through p_1 and p_2 which is tangent to the supporting line of e, is incident on the segment e itself, then that supporting point on e has the largest view. Otherwise, it is the end-point of e that is closest to the supporting point. If the view angle is no less than α, then add the edge from s^* to t^* in G_α. The above procedure is repeated for each sub-segment of s.

The total number of edges of all shortest path maps is $O(n)$. Each edge of each shortest path map in a cell c, intersects exactly one segment of $\mathcal{S}$, except if it intersects a boundary segment of c parallel to d, in which case the edge does not intersect any segment of $\mathcal{S}$. Therefore, the number of intersection points between the edges and segments of $\mathcal{S}$, and the number of sub-segments are $O(n)$. For each sub-segment, the first turning points in the shortest paths to the end-points of

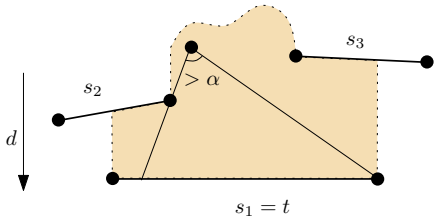

Figure 5: The α-visibility region in direction d for $t \in \mathcal{S}$ is shaded.

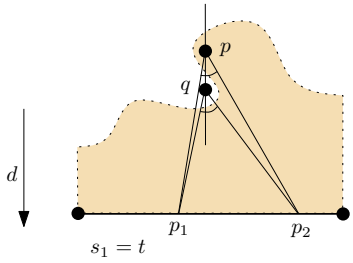

Figure 6: Points p and q are on a line in direction d. While p is in the α-visibility region in direction d for segment t, q is outside the region.

the visible segment are unique, which can be found in $O(\log n)$ time. Once we have the turning points, we can locate the point with the greatest angle of view in $O(1)$ time. Therefore, G_α can be computed in $O(n \log n)$ time.

For each direction $d \in D$, let $G_d = (V, E_d)$ be the subgraph of G_α with only edges $(s^*, t^*) \in E$ such that s and t are two facing edges in $\mathcal{T}_d$. Obviously G_d is planar, so we can store it using $O(n)$ space and check whether a pair of vertices are connected by an edge in constant time [24]. In order to check if t is weakly α-visible from s, we need to determine the existence of edge (s^*, t^*) in G_d for each $d \in D$. This requires $O(1/\alpha)$ time, which is a constant. □

4.1 One arbitrary query segment

Next, our objective is to build a data structure, such that given two query segments $s \notin \mathcal{S}$ and $t \in \mathcal{S}$, we can determine if t is weakly α-visible from s. As part of the preprocessing, for each segment $t \in \mathcal{S}$, we compute the set of all points in the plane from which t is α-visible. If t is weakly α-visible from s, then s must have a point in this set. First, we define the *α-visibility region in direction d for t*, as the region containing the points from which t is α-visible and d is inside the angle of view. Fig. 5 shows the α-visibility region in the vertical downward direction for $t \in \mathcal{S}$.

Lemma 2. *For any direction d and segment $t \in \mathcal{S}$, the boundary of the α-visibility region, say r_t, in direction d for t consists of two monotone curves with respect to the direction perpendicular to d.*

Proof. First notice that, because all points in r_t see t in direction d, r_t is a subset of the region associated to t in $\mathcal{T}_d$. Now assume r_t does not consist of two monotone curves in the direction perpendicular to d (see Fig. 6). Then, there is a line with direction d intersecting the boundary of r_t at three or more points. On this line, we can choose two points p and q, $p \in r_t$ and $q \notin r_t$, while

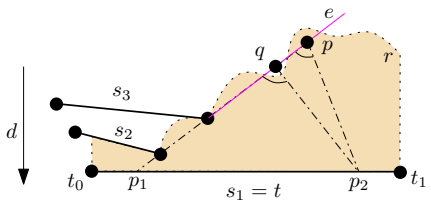

Figure 7: Points p and q are two points on e. While p is in the α-visibility region in direction d for segment t, q is outside the region.

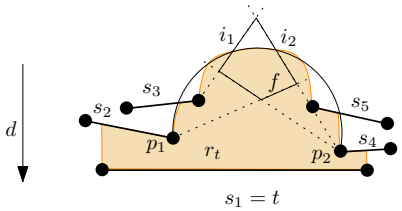

Figure 8: Points i_1 and i_2 are two consecutive points which are the intersections of the α-visibility region, in direction d for the segment t, with f.

the vector $\overrightarrow{pq}$ points towards t. The angle of view at p, β, is at least α, and $\overrightarrow{pq}$ is inside that angle. Let p_1 and p_2 be the two extreme points on t that are visible from p through β. It is easy to see that the angle $\angle p_1 q p_2 \geqslant \alpha$ and it does not intersect any segment in $\mathcal{S}$ other than t. Therefore, q can also see t with an angle at least α. This contradicts our assumption that $q \notin r$. This proves that r_t consists of two monotone curves in direction perpendicular to d. □

Using the proof of the above lemma, the following corollary is derived.

Corollary 2. *The α-visibility region for any $t \in \mathcal{S}$ and for any direction d, does not contain a hole.*

Lemma 3. *The total complexity of the α-visibility regions, in direction d, for all segments in $\mathcal{S}$, is linear.*

Proof. Let c denote the cell in $\mathcal{T}_d$ associated to t, and let r_t denote the α-visibility region in direction d for t. Obviously, $r_t \subseteq c$. Construct the shortest path maps of the end-points of t inside c.

Our first claim is that the boundary of r_t intersects any edge of the shortest path map at most once. By contradiction, as illustrated in Fig. 7, assume that for an end-point of t, say t_0, there is an edge e in the shortest path map of t_0, such that the boundary of r_t intersects e at least twice. We can choose two points p and q on e, such that $p \in r_t$ and $q \notin r_t$ and $\overrightarrow{pq}$ points towards t. Let p_1 and p_2 be the two extreme points on t visible from p through its angle of view. Let p_1 be closer to t_0 than p_2. Because p and q are on an edge of the shortest path map of t_0, both are visible from p_1. Moreover, p_1 is defined by the intersection of t and the supporting line of e. We know that $\angle p_1 p p_2 \geqslant \alpha$. Consider $\angle p_1 q p_2$. Observe that this is greater than $\angle p_1 p p_2$ and is empty. Therefore, q can see t with an angle no less than α, and because $q \in c$, it is also in r_t, which contradicts the assumption that $q \notin r_t$. Therefore, the boundary of r_t intersects each edge of the shortest path map at most once.

Consider the example depicted in Fig. 8. Let i_1 and i_2 be two consecutive intersection points of the boundary of r_t with the two shortest path maps of the end-points of t. Then, i_1 and i_2 are two points on the boundary of a cell, f, in the overlay of the two shortest path maps. Our second claim is that between i_1 and i_2, the boundary of r_t has complexity $O(|f|)$. The reason is that for all points in f, the combinatorial shortest path towards each end-point of t is unique. Therefore, for all points on the boundary of r_t, between i_1 and i_2, the largest view to t is determined by two fixed points, say p_1 and p_2. The set of all points which can see t through p_1 and p_2, with an angle $\geqslant \alpha$, are inside, or on the boundary of, the circle through p_1, p_2 having inscribed angle α lying on p_1p_2 clipped with the segment p_1p_2. Hence, the boundary of r_t between i_1 and i_2 is the intersection of that circle and f. This intersection has complexity at most $2 * |f|$, because any edge in f can be intersected by the circle at most twice. Thus, the boundary of r_t between two consecutive intersections with shortest path maps has complexity proportional to the size of f.

The complexity of r_t is equal to the number of intersections of its boundary with the shortest path maps and segments of $\mathcal{S}$. Thus, the size of r_t is $O(|c|)$. Since, the total complexity of the cells in $\mathcal{T}_d$ is linear, the total complexity of the α-visibility regions in direction d, for all segments, is $O(n)$. □

Lemma 4. *The α-visibility region in direction d, for all segments in $\mathcal{S}$, can be computed in $O(n \log n)$ time using $O(n)$ space.*

Proof. We first compute $\mathcal{T}_d$ for $\mathcal{S}$. For each segment $t \in \mathcal{S}$, with t_0 and t_1 as end-points, we construct the two shortest path maps from t_0 and t_1 in the cell c associated to t in $\mathcal{T}_d$. Let SPM(t_0) and SPM(t_1) denote the shortest path maps of t_0 and t_1, respectively, and O denote the subdivision produced by the overlay of the two shortest path maps in c.

Let r_t denote the α-visibility region in direction d for t. The lower boundary of r_t is t. Since the upper boundary of r_t is monotone in direction perpendicular to d, any line parallel to d intersects it at exactly one point. We compute the upper boundary of r_t incrementally, by sweeping a line l parallel to d from t_0 to t_1. The event points are the intersection points of r_t with the edges of O. By definition, the end-points of t are in r_t.

Let q denote the current point in the process, i.e., the intersection point of l with the upper boundary of r_t. Let p_1 and p_2 denote the first turning points of the shortest paths from q to t_0 and t_1, respectively. Initially, q is at t_0 and p_1 and p_2 are at t_0 and t_1, respectively.

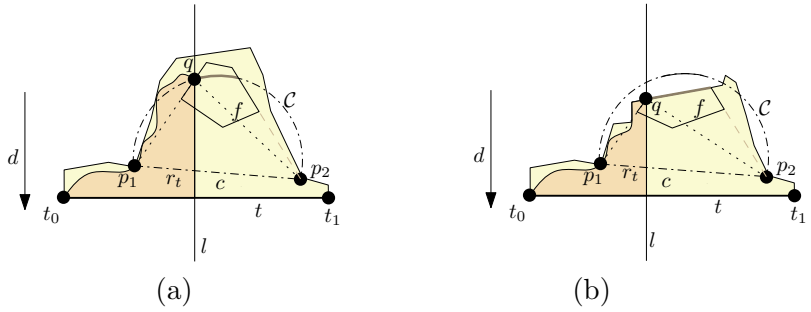

Figure 9: (a) Processing the event point q, while q is inside c. (b) Processing the event point q, while q is on the boundary of c.

Assume that we have constructed the upper boundary of r_t from t_0 to q. Assume further that r_t is entering the cell f in O, and q is on the boundary of f. The largest view angle of all points

in f to t are through p_1 and p_2. Let $\mathcal{C}$ denote the circle with inscribed angle α through p_1 and p_2. The set of points that can see t with angle at least α with view angle through p_1 and p_2 lies inside or on the boundary of $\mathcal{C}$ and above chord p_1p_2. We consider two cases: q is inside c or on the boundary of c. If q is not on the boundary of c (see Fig. 9a), because it is on the boundary of r_t, it sees t with view angle exactly equal to α. Therefore, q is on the boundary of $\mathcal{C}$. In this case, the boundary of $\mathcal{C}$ enters f and is the boundary of r_t, and we set the first intersection point of $\mathcal{C}$ with the boundary of f as the next event point. In the other case, if q is on the boundary of c (see Fig. 9b), q is inside $\mathcal{C}$ and the boundary of c is the boundary of r_t, and we set the first intersection point of the boundary of c with f and $\mathcal{C}$ as the next event point. In either case, we add the edge connecting q to the next event point (an arc in the first case or a line segment in the latter case) to the list of edges of r_t.

$\mathcal{T}_d$ can be computed in $O(n \log n)$ time. SPM(t_0) and SPM(t_1) can be computed in $O(|c|)$ time. We need not compute O, because we only need the cells in O that are intersected by the boundary of r_t, which can be computed while we construct r_t. Computing the first intersection point of $\mathcal{C}$ with f takes $O(|f|)$ time. This yields a total time of $O(|c|)$ for all circles, because each edge of SPM(t_0) and SPM(t_1) is intersected by the boundary of r_t at most once. Computing the intersection points of each circle with c can also be done in $O(|c|)$ total time, as well. Therefore, given $\mathcal{T}_d$, the α-visibility region in direction d for a segment can be computed in $O(|c|)$ time and for all segments can be computed in $O(n)$ time using $O(n)$ space. □

Theorem 8. *We can preprocess $\mathcal{S}$ into a data structure of size $O(n)$ in $O(n \log n)$ time, such that weak α-visibility between two query segments $s \notin S, t \in \mathcal{S}$, can be tested in $O(\log n)$ time.*

Proof. As before, we first fix a set D of $O(1/\alpha)$ directions with the property that the angle between any two adjacent directions is at most α. If t is weakly α-visible from s, we know that there is a direction d in D, and a point q on s from which t is α-visible. Furthermore, q can see t in direction d inside its angle of view. It is easy to see that q is in the α-visibility region in direction d for t. So, the problem reduces to that of checking the intersection of s with each of the α-visibility regions computed for t, with respect to all directions in D.

We compute the α-visibility regions in all directions $d \in D$ for all segments $t \in \mathcal{S}$ and preprocess each region for ray shooting queries. An α-visibility region is a bounded region without any hole, and its boundary consists of straight line segments and circular arcs. Therefore, it is a splinegon and we can use Theorem 4 for ray shooting queries. Given two query segments $s \notin \mathcal{S}$ and $t \in \mathcal{S}$, we first find r_t (the α-visibility region in direction d for t). We need to know if s has any point in r_t. Let s_0 and s_1 be the end-points of s. We first check whether $s_0 \in r_t$. Because the ray shooting algorithm by Melissaratos and Souvaine [23] has point location as a basis, we use it to determine if $s_0 \in r_t$. If this is the case, t is α-visible from s_0 and weakly α-visible from s. If $s_0 \notin r_t$, we perform ray shooting on r_t (the splinegon) to find the first intersection point of the ray originating from s_0 in direction towards s_1. If the intersection point is on s itself, we know that s intersects r_t and therefore t is weakly α-visible from s, otherwise it is not.

Computing the α-visibility regions for $\mathcal{S}$ takes $O(n \log n)$ time and $O(n)$ space. Preprocessing the α-visibility regions for ray shooting queries takes the same time and space. Point location and ray shooting queries can be performed in $O(\log n)$ time. This proves the bound claimed in the theorem. □

5 Complete α-Visibility

In this section, we show how to build a data structure such that given two query segments $s, t \in \mathcal{S}$, one can efficiently determine if t is completely α-visible from s. A segment t is *completely α-visible from another segment s* if and only if t is α-visible from all points on s. Note that the complete α-visibility graph is a subgraph of the weak α-visibility graph, and hence its size is also linear. To compute the complete α-visibility graph, we can design a scheme similar to that of Theorem 7. Recall that in Theorem 7, we partitioned each segment s into sub-segments in which the boundary of the view angles passes through a unique pair of points. In order to make sure t is completely α-visible from s, for each sub-segment, we identify the part from which t is completely α-visible. If the union of all parts (for all directions) is equal to s, we add an edge from s to t in the complete α-visibility graph.

In the following theorem, we describe another approach using α-visibility regions to compute the complete α-visibility graph.

Theorem 9. *In $O(n \log n)$ time, we can preprocess $\mathcal{S}$ into a data structure of size $O(n)$, such that we can answer complete α-visibility queries in $O(1)$ time.*

Proof. We first fix a set, D, containing the $O(1/\alpha)$ directions. For each $d \in D$, we compute the α-visibility regions of all segments of $\mathcal{S}$ via Lemma 4. The boundary of the α-visibility region, r_t, of a segment $t \in \mathcal{S}$ consists of a set of arcs and segments. The segments on the boundary of r_t are segments of $\mathcal{S}$ or parts thereof, blocking the view of the points behind them. Segment t is α-visible from these portions of segments in $\mathcal{S}$. Therefore, if t is completely α-visible from a segment s, the union of the parts of s that appeared in the α-visibility regions of t in all directions, is the entire segment s.

Let $d \in D$ be a direction. For each segment $t \in \mathcal{S}$, we traverse all segments $e \in r_t$, and add e to the ordered set $I_{s,d}$ of intervals of the corresponding segment s. The ordered set $I_{s,d}$ stores intervals of s from which t is α-visible and their view angles contain d. After scanning the α-visibility regions of all segments in all directions, we have $O(1/\alpha)$ ordered sets of intervals for each segment $s \in \mathcal{S}$, such that each point in these sets sees t with angle of view $\geqslant \alpha$. If t is completely α-visible from s, then the union of all these intervals is s. The union of these intervals can be computed in $O(k \log(1/\alpha))$ time, where k is the total number of sub-segments of s that are used in the α-visibility regions. In the worst case, k could be $O(n)$. If the union of these sub-segments is s, we add an edge to the complete α-visibility graph of $\mathcal{S}$ from the vertex corresponding to s to the vertex corresponding to t. The total preprocessing time is $O(n/\alpha \log(1/\alpha))$. To answer the query, we now employ Theorem 7 to decompose the graph into $O(1/\alpha)$ planar graphs. This yields a query time of $O(1/\alpha)$, which is a constant. □

5.1 One arbitrary query segment

Now, we wish to build a data structure, so that for any two query segments $s \notin \mathcal{S}$ and $t \in \mathcal{S}$, we can determine if t is completely α-visible from s. To answer such queries, we use the α-visibility regions. If t is completely α-visible from s, then the union of the intersections of s with the α-visibility regions of t in all directions is equal to the segment s.

Theorem 10. *We can preprocess $\mathcal{S}$ into a data structure of size $O(n)$ in $O(n \log n)$ time, such that given two query segments $s \notin S, t \in \mathcal{S}$, their complete α-visibility can be tested in $O(n)$ time.*

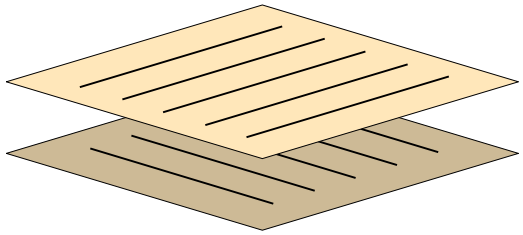

Figure 10: An arrangement of segments in 3D, whose weak α-visibility graph has $\Omega(n^2)$ edges.

Proof. In the preprocessing phase, we fix a set of $O(1/\alpha)$ directions D and for each direction $d \in D$, based on Lemma 4, we compute the α-visibility regions of all segments $t \in \mathcal{S}$. At query time, given a query segment s, for each direction $d \in D$, we store intervals of s from which t is α-visible in $I_{s,d}$. Then, we compute the union of these intervals. If this union is equal to s, t is completely α-visible from s, otherwise t is not completely α-visible from s.

The preprocessing phase can be carried out in $O(n \log n)$ time using $O(n)$ space. In the preprocessing we have $O(1/\alpha)$ ordered interval sets, whose union can be found in $O(k \log(1/\alpha))$, where k is the total number of intervals. In the worst case, again k could be $O(n)$. □

6 α-Visibility in 3D

We construct an example to show that there exists a set of n-line segments $\mathcal{S}$ in 3-d Euclidean space, so that the weak α-visibility graph of $\mathcal{S}$ has $\Omega(n^2)$ edges. Consider two parallel planes, each consisting of a set of $n/2$ parallel line segments (see Fig. 10). We can choose the distance between the two parallel planes and α such that all segments on one plane are weakly α-visible to the segments on the other plane. Therefore, Lemma 1 does not extend to three dimensions.

7 Conclusion

In this paper, we introduced a new model for approximating visibility between a point and a segment, called *α-visibility*. In this model, we showed how to approximate weak and complete visibility between segments and point-to-segment α-visibility. We proved that both the weak and the complete α-visibility graph in a polygonal scene has linear complexity and can be computed in $O(n \log n)$ time. This makes it appealing both from a theoretical as well as from an applied perspective, particularly when dealing with large data sets. We then extended our results and solved several query versions of weak and complete α-visibility problems. In particular, the data structures described in Section 4 might be usable for solving other types of visibility problems. Finally, we established that our results established in 2D for weak and complete α-visibility from a segment do not extend to 3D.

The problem of determining whether t is weakly/completely α-visible from s, when both s, t do not belong to $\mathcal{S}$ in $O(\log n)$ query time with $O(n \log n)$ preprocessing time is open. We currently study generalizing segment *α-visibility* to *α-visibility* of objects. It may be interesting to explore quantitative measures of *α-visibility* where, say segments, are not just *α-visible*, or not, but the proportion of the *α-visible* part vs. the invisible one inside the triangle is determined.

References

[1] B. Aronov, L. J. Guibas, M. Teichmann, and L. Zhang. Visibility queries and maintenance in simple polygons. *Discrete & Computational Geometry*, 27(4):461–483, 2002.

[2] T. Asano. An efficient algorithm for finding the visibility polygon for a polygonal region with holes. *Transactions of IECE of Japan*, E68(9):557–559, 1985.

[3] T. Asano, T. Asano, L. J. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1(1):49–63, 1986.

[4] T. Asano, S.K. Ghosh, and T.C. Shermer. Visibility in the plane. In *Handbook of Computational Geometry*, pages 829–876. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999.

[5] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. *Computational Geometry: Theory & Applications*, 23(3):313–335, 2002.

[6] H Brunn. Über Kerneigebiete. *Mathematische Annalen*, 73:436–440, 1913.

[7] T. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012.

[8] Otfried Cheong and René van Oostrum. Reaching a polygon with directional uncertainty. *International Journal of Computational Geometry and Applications*, 11(2):197–214, 2001.

[9] Mark de Berg, Leonidas J. Guibas, Dan Halperin, Mark H. Overmars, Otfried Schwarzkopf, Micha Sharir, and Monique Teillaud. Reaching a goal with directional uncertainty. *Theoretical Computer Science*, 140(2):301–317, 1995.

[10] David P. Dobkin and Diane L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5(3):421–457, 1990.

[11] H. A. ElGindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2(2):186–197, 1981.

[12] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20(5):888–910, 1991.

[13] S.K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

[14] J. Gudmundsson and P. Morin. Planar visibility: testing and counting. In *Proceedings of the 26th annual Symposium on Computational Geometry*, pages 77–86, 2010.

[15] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. *SIAM Journal on Computing*, 26(4):1120–1138, 1997.

[16] Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

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

[18] J. Hershberger. Finding the visibility graph of a simple polygon in time proportional to its size. In *Proceedings of the third annual Symposium on Computational Geometry*, pages 11–20, 1987.

[19] R. Inkulu and S. Kapoor. Visibility queries in a polygonal region. *Computational Geometry: Theory & Applications*, 42(9):852–864, 2009.

[20] M. Keil, D. M. Mount, and S. K. Wismath. Visibility stabs and depth-first spiralling on line segments in output sensitive time. *International Journal of Computational Geometry & Applications*, 10(5):535–552, 2000.

[21] D. T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207–221, 1983.

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

[23] E. A. Melissaratos and D. L. Souvaine. Shortest paths help solve geometric optimization problems in planar regions. *SIAM Journal on Computing*, 21(4):601–638, 1992.

[24] J. Ian Munro and Venkatesh Raman. Succinct representation of balanced parentheses and static trees. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[25] Mostafa Nouri-Baygi and Mohammad Ghodsi. Space/query-time tradeoff for computing the visibility polygon. *Computational Geometry: Theory & Applications*, 46(3):371–381, 2013.

[26] Mostafa Nouri-Baygi, Alireza Zarei, and Mohammad Ghodsi. Weak visibility of two objects in planar polygonal scenes. In *Proceedings of the 2007 International Conference on Computational Science and its Applications*, pages 68–81, 2007.

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

[28] 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, 1986.

[29] G. Vegter. The visibility diagram: a data structure for visibility problems and motion planning. In *Proceedings of the second Scandinavian Workshop on Algorithm Theory*, pages 97–110, 1990.

[30] A. Zarei and M. Ghodsi. Query point visibility computation in polygons with holes. *Computational Geometry: Theory & Applications*, 39(2):78–90, 2008.