The Visibility Center of a Simple Polygon

Anna Lubiw
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Anurag Murty Naredla
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Abstract

We introduce the *visibility center* of a set of points inside a polygon – a point c_V such that the maximum geodesic distance from c_V to see any point in the set is minimized. For a simple polygon of n vertices and a set of m points inside it, we give an $O((n+m)\log(n+m))$ time algorithm to find the visibility center. We find the visibility center of *all* points in a simple polygon in $O(n \log n)$ time.

Our algorithm reduces the visibility center problem to the problem of finding the geodesic center of a set of half-polygons inside a polygon, which is of independent interest. We give an $O((n+k)\log(n+k))$ time algorithm for this problem, where k is the number of half-polygons.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Visibility, Shortest Paths, Simple Polygons, Facility Location

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.65

Related Version *Full Version*: https://arxiv.org/abs/2108.07366

Funding *Anna Lubiw*: Supported by NSERC.

1 Introduction

Suppose you want to guard a polygon and you have many sensors but only one guard to check on the sensors. The guard must be positioned at a point c_V in the polygon such that when a sensor at any query point u sends an alarm, the guard travels from c_V on a shortest path inside the polygon to *see* point u; the goal is to minimize the maximum distance the guard must travel. More precisely, we must choose c_V to minimize the maximum, over points u, of the geodesic distance from c_V to a point that sees u. The optimum guard position c_V is called the *visibility center* of the set U of possible query points. See Figure 1. We give an $O((n+m)\log(n+m))$ time algorithm to find the visibility center of a set U of size m in an n-vertex simple polygon. To find the visibility center of *all* points inside a simple polygon, we can restrict our attention to the vertices of the polygon, which yields an $O(n \log n)$ time algorithm.

To the best of our knowledge, the idea of visibility centers is new, though it is a very natural concept that combines two significant branches of computational geometry: visibility problems [12]; and center problems and farthest Voronoi diagrams [5].

There is a long history of finding "center points", for various definitions of "center". The most famous of these is Megiddo's linear time algorithm [20] to find the center of a set of points in the plane (Sylvester's "smallest circle" problem).

Inside a polygon the relevant distance measure is not the Euclidean distance but rather the shortest path, or *geodesic*, distance. The *geodesic center* of a simple polygon is a point p that minimizes the maximum geodesic distance from p to any point q of the polygon, or equivalently, the maximum geodesic distance from p to any vertex of the polygon. Pollack, Sharir, and Rote [24] gave an $O(n \log n)$ time divide-and-conquer algorithm to find the geodesic center of a polygon. Our algorithm builds on theirs. A more recent algorithm by

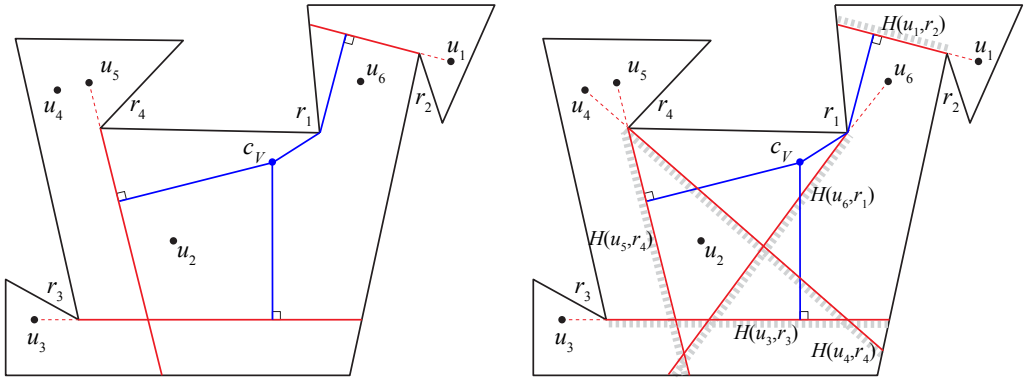

Figure 1 (left) Point c_V is the *visibility center* of points $U = \{u_1, \ldots, u_6\}$. Starting from c_V, the three points we need to travel (equally) farthest to see are u_1, u_3 and u_5. The shortest paths (in blue) to see these points must reach the half-polygons bounded by the chords (in red) emanating from the points. (right) Equivalently, c_V is the geodesic center of five half-polygons (each shown as a red boundary chord shaded on one side).

Ahn et al. [1] finds the geodesic center of a polygon in linear time. Another notion of the center of a polygon is the link center, which can be found in $O(n \log n)$ time [10].

Center problems are closely related to farthest Voronoi diagrams, since the center is (modulo degeneracies) a vertex of the corresponding farthest Voronoi diagram. Finding the farthest Voronoi diagram of points in the plane takes $\Theta(n \log n)$ time – thus is it strictly harder to find the farthest Voronoi diagram than to find the center. However, working inside a simple polygon helps for farthest Voronoi diagrams: the farthest geodesic Voronoi diagram of the vertices of a polygon can be found in time $O(n \log \log n)$ [23]. Generalizing the two scenarios (points in the plane, and polygon vertices), yields the problem of finding the farthest Voronoi diagram of m points in a polygon, which was first solved by Aronov et al. [3] with run-time $O((n+m)\log(n+m))$, and improved in a sequence of papers [23, 6, 22], with the current best run-time of $O(n + m \log m)$ [27].

Turning to visibility problems in a polygon, there are algorithms for the "quickest visibility problem" – to find the shortest path from point s to see point q, and to solve the query version where s is fixed and q is a query point [2, 26]. For a simple polygon [2], the preprocessing time and space are $O(n)$ and the query time is $O(\log n)$. We did not find these results useful in our algorithm to find the visibility center c_V, but they are useful afterwards to find the actual shortest path from c_V to see a query point.

A more basic version of our problem is to find, if there is one, a point that sees all points in U. The set of such points is the *kernel* of U. When U is the set of vertices, the kernel can be found in linear time [19]. For a general set U, Ke and O'Rourke [18] gave an $O(n + m \log(n+m))$ time algorithm to compute the kernel, and we use some of their results in our algorithm.

Another problem somewhat similar to ours is the watchman problem [9, 11] – to find a minimum length tour from which a single guard can see the whole polygon. Our first step is similar in flavour to the first step for the watchman problem, namely, to replace the condition of "seeing" everything by a condition of visiting certain "essential chords".

Our Results

The *distance to visibility* from a point x to point u in P, denoted $d_V(x,u)$ is the minimum distance in P from x to a point y such that y sees u. For a set of points U in P, the *visibility*

radius of x with respect to U is $r_V(x, U) := \max\{d_V(x, u) : u \in U\}$. The *visibility center* c_V of U is a point x that minimizes $r_V(x, U)$. Our main result is:

▶ **Theorem 1.** *There is an algorithm to find the visibility center of a point set U in a simple n-vertex polygon P with run-time $O((n + m) \log(n + m))$ where m is the size of U.*

The key to our algorithm is to reformulate the visibility center problem in terms of distances to certain *half-polygons* inside the polygon. We illustrate the idea by means of the example in Figure 1 where the visibility center of the 6-element point set U is the *geodesic center* of a set of five half-polygons.

More generally, we will reduce the problem of finding the visibility center to the problem of finding a geodesic center of a linear number of half-polygons. The input to this problem is a set $\mathcal{H}$ of k half-polygons (see Section 2 for precise definitions) and the goal is to find a *geodesic center* c that minimizes the maximum distance from c to a half-polygon. More precisely, the *geodesic radius* from a point x to $\mathcal{H}$ is $r(x, \mathcal{H}) := \max\{d(x, H) : H \in \mathcal{H}\}$, and the *geodesic center* c of $\mathcal{H}$ is a point x that minimizes $r(x, \mathcal{H})$. Our second main result is:

▶ **Theorem 2.** *There is an algorithm to find the geodesic center of a set $\mathcal{H}$ of half-polygons in a simple n-vertex polgyon P with run-time $O((n + k) \log(n + k))$ where k is the size of $\mathcal{H}$.*

Our algorithm extends the divide-and-conquer approach that Pollack et al. [24] used to compute the geodesic center of the vertices of a simple polygon.

Our main motivation for finding the geodesic center of half-polygons is to find the visibility center, but the geodesic center of half-polygons is of independent interest. Euclidean problems of a similar flavour are to find the center (or the farthest Voronoi diagram) of line segments or convex polygons in the plane [7, 17]. These problems are less well-studied than the case of point sites (e.g., see [4] for remarks on this). The literature for geodesic centers is even more sparse, focusing almost exclusively on geodesic centers of points in a polygon. It is thus interesting that the center of half-polygons inside a polygon can be found efficiently. As a special case, we can find the geodesic center of the edges of a simple polygon in $O(n \log n)$ time.

The reduction from the visibility center problem to the geodesic center of half-polygons is in Section 3. The run time is $O((n + m) \log(n + m))$. The algorithm that proves Theorem 2 is in Section 4. Together these prove Theorem 1.

2 Preliminaries

We add a few more basic definitions to augment the main definitions given above. We work with a simple polygon P of n vertices whose boundary ∂P is directed clockwise. A *chord* of P is a line segment inside P that intersects ∂P only at its two endpoints. Any chord divides P into two *half-polygons*. A half-polygon is specified by its chord (p, q) with the convention that the half-polygon contains the path clockwise from p to q.

The *geodesic distance* $d(x, y)$ (or simply, *distance*) between two points x and y in P is the length of the shortest path $\pi(x, y)$ in P from x to y. For half-polygon H, the *geodesic distance* $d(x, H)$ is the minimum distance from x to a point in H.

Points x and y in P are *visible* (x "sees" y) if the segment xy lies inside P. The *distance to visibility* from x to u, denoted $d_V(x, u)$ is the minimum distance from x to a point y such that y sees u. If x sees u, then this distance is 0, and otherwise it is the distance from x to the half-polygon defined as follows. Let r be the last reflex vertex on the shortest path from x to u. Extend the ray $\overrightarrow{ur}$ from r until it hits the polygon boundary ∂P at a point p to

obtain a chord rp (which is an edge of the *visibility polygon* of u). Of the two half-polygons defined by rp, let $H(u, r)$ be the one that contains u. See Figure 1.

▶ **Observation 3.** $d_V(x, u) = d(x, H(u, r))$.

At their core, our methods depend on convexity properties of the distance functions. A basic result is the following which is proved in the full version of our paper.

▷ Claim 4. The set of points x with $r_V(x, U) \leq k$ [or with $r(x, \mathcal{H}) \leq k$] is *geodesically convex*, i.e., if two points x, y lie in the set then so does $\pi(x, y)$.

More detail on convexity properties can be found in the long version of our paper. These convexity properties allow us to prove that the visibility center of a set of points U and the geodesic center of a set of half-polygons $\mathcal{H}$ are unique except in very special cases. We explain this for the geodesic center of half-polygons, but the same argument works for the visibility center (or, alternatively, one can use the reduction from the visibility center to the geodesic center in Section 3). First of all, if the geodesic radius is 0 then any point in the intersection of the half-polygons is a geodesic center. So we assume that the geodesic radius r is positive. Then we have the following (proved in the long version):

▷ Claim 5. There is a set $\mathcal{H}' \subseteq \mathcal{H}$ of two or three half-polygons such that the set of geodesic centers of $\mathcal{H}$ is equal to the set of geodesic centers of $\mathcal{H}'$ and furthermore

1. if $\mathcal{H}'$ has size 3 then the geodesic center is unique (see Figure 1)
2. if $\mathcal{H}'$ has size 2 then either the geodesic center is unique or the two half-polygons of $\mathcal{H}'$ have chords that are parallel and the geodesic center consists of a line segment parallel to them and midway between them.

3 Reducing the Visibility Center to the Center of Half-Polygons

In this section we reduce the problem of finding the visibility center of a set of points U in a polygon P to the problem of finding the geodesic center of a linear number of "essential" half-polygons $\mathcal{H}$, which is solved in Section 4.

By Observation 3 (and see Figure 1) the visibility center of U is the geodesic center of the set of $O(mn)$ half-polygons $H(u, r)$ where $u \in U$, r is a reflex vertex of P that sees u, and $H(u, r)$ is the half-polygon containing u and bounded by the chord that extends $\overrightarrow{ur}$ from r until it hits ∂P at a point t. Note that finding t is a ray shooting problem and costs $O(\log n)$ time after an $O(n)$ time preprocessing step [16].

However, this set of half-polygons is too large. We will find a set $\mathcal{H}$ of $O(n)$ "essential" half-polygons that suffice, i.e., such that the visibility center of U is the geodesic center of the half polygons of $\mathcal{H}$. In fact, we give two possible sets of essential half-polygons, $\mathcal{H}_{\text{reflex}}$ and $\mathcal{H}_{\text{hull}}$, where the latter set can be found more efficiently. The bottleneck is still the algorithm for geodesic center of half-polygons, but it still seems worthwhile to optimize the reduction.

We first observe that any half-polygon that contains another one is redundant. For example, in Figure 1 $H(u_4, r_4)$ is redundant because it contains $H(u_5, r_4)$. At each reflex vertex r of P, there are at most two minimal half-polygons $H(u, r)$. Define $\mathcal{H}_{\text{reflex}}$ to be this set of minimal half-polygons. Note that $\mathcal{H}_{\text{reflex}}$ has size $O(n_r)$ where n_r is the number of reflex vertices of P.

Observe that for the case of finding the visibility center of *all* points of P, $\mathcal{H}_{\text{reflex}}$ consists of the half-polygons $H(v, r)$ where (v, r) is an edge of P, so $\mathcal{H}_{\text{reflex}}$ can be found in time $O(n + n_r \log n)$.

For a point set U, the set $\mathcal{H}_{\text{reflex}}$ was also used by Ke and O'Rourke [18] in their algorithm to compute the kernel of point set U in polygon P. (Recall from the Introduction that the kernel of U is the set of points in P that see all points of U.) They gave a sweep line algorithm ("Algorithm 2") to find $\mathcal{H}_{\text{reflex}}$ in time $O((n+m)\log(n+m))$. To summarize:

▶ **Proposition 6.** *The geodesic center of $\mathcal{H}_{\text{reflex}}$ is the visibility center of U. Furthermore, $\mathcal{H}_{\text{reflex}}$ can be found in time $O((n+m)\log(n+m))$.*

In the remainder of this section we present a second approach using $\mathcal{H}_{\text{hull}}$ that eliminates the $O(n \log n)$ term. This does not change the runtime to find the visibility center, but it means that improving the algorithm to find the geodesic center of half-polygons will automatically improve the visibility center algorithm. The idea is that $\mathcal{H}_{\text{reflex}}$ is wasteful in that a single point $u \in U$ can give rise to n_r half-polygons. Note that we really only need three half-polygons in an essential set, though the trouble is to find them!

We first eliminate the case where the kernel of U is non-empty (i.e., $r_V = 0$) by running the $O(n + m\log(n+m))$ time kernel-finding algorithm of Ke and O'Rourke [18]. Next we find $\mathcal{H}_{\text{hull}}$ in two steps. First make a subset $\mathcal{H}_0$ as follows. Construct R, the geodesic convex hull of U in P in time $O(n + m\log(m+n))$ [14, 25]. For each edge (u, r) of R where $u \in U$ and r is a reflex vertex of P, put $H(u, r)$ into $\mathcal{H}_0$. Note that $\mathcal{H}_0$ has size $O(\min\{n_r, m\})$ so ray shooting to find the endpoints of the chords $H(u, r)$ takes time $O(n + \min\{n_r, m\}\log n)$. Unfortunately, as shown in Figure 2, $\mathcal{H}_0$ can miss an essential half-polygon.

Next, construct a geodesic center c_0 of $\mathcal{H}_0$ using the algorithm of Section 4. (Note that the geodesic center can be non-unique and in such cases c_0 denotes any one point from the set of geodesic centers.) Then repeat the above step for $U \cup \{c_0\}$, more precisely, construct R', the geodesic convex hull of $U \cup \{c_0\}$ in P and for each edge (u, r) of R' where $u \in U$ and r is a reflex vertex of P, add $H(u, r)$ to $\mathcal{H}_0$. This defines $\mathcal{H}_{\text{hull}}$. Again, $\mathcal{H}_{\text{hull}}$ has size $O(\min\{n_r, m\})$ and ray shooting costs $O(n + \min\{n_r, m\}\log n)$.

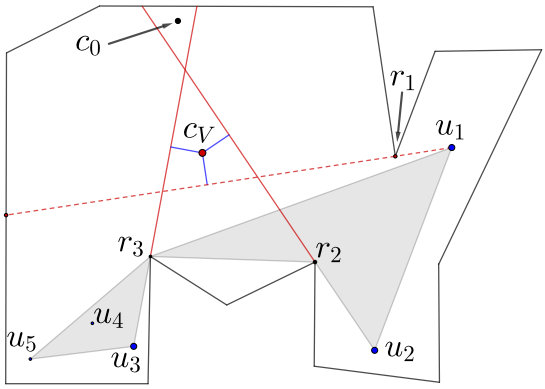

Figure 2 The geodesic convex hull of $U = \{u_1, \ldots, u_5\}$ is shaded grey. $\mathcal{H}_0$ consists of the two half-polygons $H(u_2, r_2)$ and $H(u_3, r_3)$ (with solid red chords), but misses $H(u_1, r_1)$, which is essential for the visibility center c_V. The point c_0 denotes a geodesic center of $\mathcal{H}_0$.

▶ **Theorem 7.** *Suppose the kernel of U is empty. Then the geodesic center of $\mathcal{H}_{\text{hull}}$ is the visibility center of U. Furthermore $\mathcal{H}_{\text{hull}}$ can be found in time $O(n + m\log(n+m))$ plus the time to find the geodesic center of $O(\min\{n_r, m\})$ half-polygons.*

Proof. The run-time was analyzed above. Consider the visibility center c_V. By assumption, $r_V > 0$. We consider the half-polygons $H(u, r) \in \mathcal{H}_{\text{reflex}}$ such that $r_V = d(c_V, H(u, r))$.

By Claim 5 either there are three of these half-polygons, H_1, H_2 and H_3, that uniquely determine c_V, or there are two, H_1 and H_2, that determine c_V. Then c_V is the geodesic center of H_i $i = 1, 2, 3$ or $i = 1, 2$ depending on which case we are in. Let $H_i = H(u_i, r_i)$.

If all the H_i's are in $\mathcal{H}_0$, we are done. We will show that at least two are in $\mathcal{H}_0$ and the third one (if it exists) is "caught" by c_0. See Figure 2. Let h_i be the chord defining H_i and let $\overline{H}_i$ be the other half-polygon determined by h_i.

▷ Claim 8. If U contains a point in $\overline{H}_i$ then (u_i, r_i) is an edge of R so $H_i \in \mathcal{H}_0$.

Proof. Let u be a point in $\overline{H}_i$. Observe that $\pi(u_i, u)$ contains the segment $u_i r_i$. Thus r_i is a vertex of R. Furthermore $u_i r_i$ is an edge of R. (Note that H_i is extreme at r since we picked it from $\mathcal{H}_{\text{reflex}}$.) Thus H_i is in $\mathcal{H}_0$. ◁

▷ Claim 9. At least two of the H_i's lie in $\mathcal{H}_0$.

Proof. First observe that if two of the half-polygons are disjoint, say H_i and H_j, then they lie in $\mathcal{H}_0$, because $u_i \in H_i$ implies $u_i \in \overline{H}_j$ so by Claim 8, $H_i \in \mathcal{H}_0$, and symmetrically, $H_j \in \mathcal{H}_0$.

We separate the proof into cases depending on the number of H_i's. If there are two then they must be disjoint otherwise a point in their intersection would be a visibility center with visibility radius $r_V = 0$. Then by the above observation, they are both in $\mathcal{H}_0$.

It remains to consider the case of three half-polygons. If two are disjoint, we are done, so suppose each pair H_i, H_j intersects. Then the three chords h_i form a triangle. Furthermore, since $\bigcap \overline{H}_i$ is non-empty (it contains c_V), the inside of the triangle is $\bigcap \overline{H}_i$. Now suppose $H_1 \notin \mathcal{H}_0$. Then by Claim 8, $u_2, u_3 \in H_1$. This implies (see Figure 2) that $u_2 \in \overline{H}_3$ and $u_3 \in \overline{H}_2$, so by Claim 8, H_2 and H_3 are in $\mathcal{H}_0$. ◁

We now complete the proof of the theorem. We only need to consider the case of three H_i's, where one of them, say H_1, is not in $\mathcal{H}_0$. Our goal is to show that c_0, the geodesic center of $\mathcal{H}_0$, lies in $\overline{H}_1$ and thus H_1 is in $\mathcal{H}_{\text{hull}}$. Let $X = \{x \in P : d(x, H_2) \leq r_V \text{ and } d(x, H_3) \leq r_V\}$. Observe that $c_0 \in X$ (because the radius is non-increasing as we eliminate half-polygons). Now, c_V is the unique point within distance r_V of the half-polygons H_1, H_2 and H_3. If $c_0 \in H_1$, then c_0's distance to H_1 would be 0 which contradicts the uniqueness property of c_V. Thus $c_0 \in \overline{H}_1$. By the same reasoning as in Claim 8, this implies that $u_1 r_1$ is an edge of R', the geodesic convex hull of $U \cup \{c_0\}$. Thus H_1 is in $\mathcal{H}_{\text{hull}}$ by definition of $\mathcal{H}_{\text{hull}}$. ◀

4 The Geodesic Center of Half-Polygons

In this section, we give an algorithm to find the geodesic center of a set $\mathcal{H}$ of k half-polygons inside an n-vertex polygon P. We preprocess by sorting the half-polygons in cyclic order of their first endpoints around ∂P in time $O(k \log k)$. We assume that no half-polygon in $\mathcal{H}$ contains another – any non-minimal half-polygon is irrelevant and can be discarded. Note that the minimal half-polygons can be found in linear time from the sorted order.

We follow the approach that Pollack et al. [24] used to find the geodesic center of the vertices of a polygon. Many steps of their algorithm rely, in turn, on search algorithms of Megiddo's [20].

The main ingredient of the algorithm is a linear time *chord oracle* that, given a chord $K = ab$ of the polygon, finds the *relative geodesic center*, c_K (the optimum center point restricted to points on the chord), and tells us which side of the chord contains the center. We must completely redo the chord oracle in order to handle paths to half-polygons instead of vertices, but the main steps are the same. Our chord oracle runs in time $O(n + k)$. Pollack

et al.'s chord oracle was used as a black box in subsequent faster algorithms [1], so we imagine that our version will be an ingredient in any faster algorithm for the geodesic center of half-polygons.

Using the chord oracle, we again follow the approach of Pollack et al. to find the geodesic center. The total run time is $O((n+k)\log(n+k))$.

We give a road-map for the remainder of this section, listing the main steps, which are the same as those of Pollack et al., and highlighting the parts that we must rework.

§ 4.1 A Linear Time Chord Oracle

1. Test a candidate center point. Given a point x on the chord $K = ab$, is the relative geodesic center c_K to the left or right of x? Is the geodesic center c to the left or right of chord K?
2. Find shortest paths from a and from b to all half-polygons. The details of this step are novel, because we need shortest paths to half-polygons rather than vertices.
3. Find a linear number of simple functions defined on K whose upper envelope is the geodesic radius function. We must redo this from the ground up.
4. Find the relative center on K (the point that minimizes the geodesic radius function) using Megiddo's technique.

§ 4.2 Finding the Geodesic Center of Half-Polygons

1. Use the chord oracle to find a region of P that contains the center and such that for any half-polygon $H \in \mathcal{H}$, all geodesic paths from the region to H are combinatorially the same. We give a more modern version of this step using epsilon nets.
2. Solve the resulting Euclidean "intersection radius problem" – to find a smallest disk that contains given disks and intersects given lines. This is new because of the condition about intersecting given lines.

4.1 A Linear Time Chord Oracle

In this section we give a linear time chord oracle. Given a chord $K = ab$ the chord oracle tells us whether the geodesic center of $\mathcal{H}$ lies to the left, to the right, or on the chord K. It does this by first finding the relative geodesic center $c_K = \text{argmin}\{r(x, \mathcal{H}) : x \in K\}$, together with the half-polygons that are farthest from c_K. From this information, we can identify which side of K contains the geodesic center c in the same way as Pollack et al. by testing the vectors of the first segments of the shortest paths from c_K to its furthest half-polygons. This test is described in Subsection 4.1.1.

The chord oracle thus reduces to the problem of finding the relative geodesic center and its farthest half-polygons. The main idea here is to capture the geodesic radius function along the chord (i.e., the function $r(x, \mathcal{H})$ for $x \in K$) as the upper envelope of a linear number of simple functions defined on overlapping subintervals of K. In order to find the simple functions (Section 4.1.3) we first compute shortest paths from a and from b to all the half-polygons (Section 4.1.2). Finally we apply Megiddo's techniques (Section 4.1.4) to find the point c_K on K that minimizes the geodesic radius function.

4.1.1 Testing a Candidate Center Point

Given the relative geodesic center c_K on chord K and the first edges of the paths from c_K to its farthest half-polygons, we can test in constant time whether the geodesic center is equal to c_K or lies to the left or the right of K. We can also test, given a point x on K and the

first edges of the paths to its farthest half-polygons, whether c_K is equal to x or lies to the left or right of x. We illustrate the tests in Figure 3, and defer a more rigorous explanation to the long version of our paper.

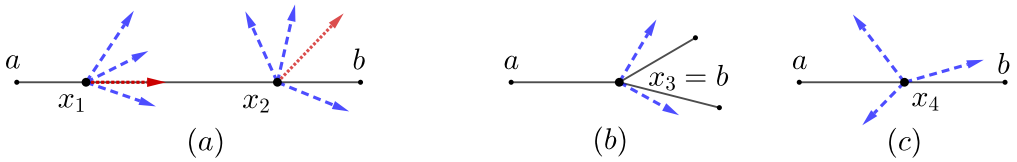

Figure 3 Points x_i on chord $K = ab$ with directions of paths to farthest half-polygons in dashed blue and the direction for improvement in dotted red. (a) x_1 is not a relative center. x_2 is a relative center but not the true geodesic center. (b,c) x_3 and x_4 are geodesic centers.

4.1.2 Shortest Paths to Half-Polygons

In this section we give a linear time algorithm to find the shortest path tree from point a on the polygon boundary to all the half-polygons $\mathcal{H}$. Recall that each half-polygon is specified by an ordered pair of endpoints on ∂P, and the half-polygons are sorted in clockwise cyclic order by their first endpoints. From this, we identify the half-polygons that contain a, and we discard them – their distance from a is 0. Let $H_1, \ldots, H_{k'}$ be the remaining half-polygons where H_i is bounded by endpoints p_iq_i, and the H_i's are sorted by p_i, starting at a.

The idea is to first find the shortest path map T_a from a to the set consisting of the polygon vertices and the points p_i and q_i. Recall that the shortest path map is an augmentation of the shortest path tree that partitions the polygon into triangular regions in which the shortest path from a is combinatorially the same (see Figure 4). The shortest path map can be found in linear time [13]. Note that T_a is embedded in the plane (none of its edges cross) and the ordering of its leaves matches their ordering on ∂P. Our algorithm will traverse T_a in depth-first order, and visit the triangular regions along the way.

Our plan is to augment T_a to a shortest path tree $\bar{T}_a$ that includes the shortest paths from a to each half-polygon H_i. Note that $\bar{T}_a$ is again an embedded ordered tree. We can find $\pi(a, H_i)$ by examining the regions of the shortest path map intersected by p_iq_i. These lie in the *funnel* between the shortest paths $\pi(a, p_i)$ and $\pi(a, q_i)$. Note that edges of the shortest path map T_a may cross the chord p_iq_i. Also, the funnels for different half-polygons may overlap. The key to making the search efficient is the following lemma:

▶ **Lemma 10.** *The ordering $H_1, H_2, \ldots, H_{k'}$ matches the ordering of the paths $\pi(a, H_i)$ in the tree $\bar{T}_a$.*

Proof. Consider two half-polygons $H_i = p_iq_i$ and $H_j = p_jq_j$, with $i < j$. We prove that $\pi(a, H_i)$ comes before $\pi(a, H_j)$ in $\bar{T}_a$. If H_i and H_j are disjoint, the result is immediate since the corresponding funnels do not overlap. Otherwise (because neither half-polygon is contained in the other) p_iq_i and p_jq_j must intersect, say at point x. See Figure 4. Let t_i and t_j be the terminal points of the paths $\pi(a, H_i)$ and $\pi(a, H_j)$, respectively. If t_i lies in p_ix and t_j lies in xq_j then the result follows since t_i and t_j lie in order on the boundary of the truncated polygon formed by removing H_i and H_j. So suppose that t_j lies in p_jx (the other case is symmetric). Then $\pi(a, t_j)$ crosses p_iq_i at a point z in p_ix. From z to t_j the path $\pi(a, t_j)$ lies inside the cone with apex x bounded by the rays from x through z and from x through t_j. Within that cone, the path only turns left. The angle α_j at t_j is $\geq 90°$ (it may be $> 90°$ if $t_j = p_j$), which implies that the angle α_i at z is $> 90°$. Therefore t_i lies to the left of z, as required. ◀

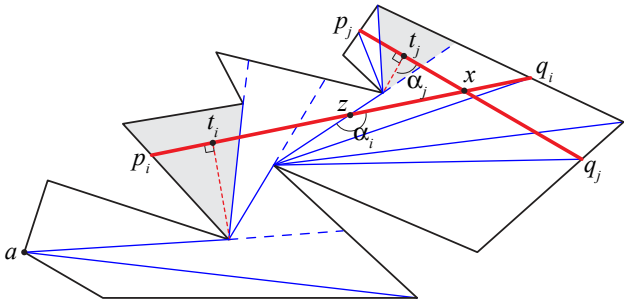

Figure 4 The shortest path map T_a (thin blue) and the augmentation (dashed red) to include shortest paths to the two half-polygons bounded by chords p_iq_i and p_jq_j (thick red).

Based on the Lemma, the algorithm traverses the regions of the shortest path map T_a in depth first search order, and traverses the half-polygons H_i in order $i = 1, 2, \ldots, k'$. It is easy to test if one region contains the shortest path to H_i (either to p_i, or to q_i, or reaching an internal point of p_iq_i at a right angle); if it does, we increment i, and otherwise we proceed to the next region. The total time is $O(n + k)$.

4.1.3 Functions to Capture the Distance to Farthest Half-Polygons

In this section we give a linear time algorithm to find a linear number of simple functions defined on the chord $K = ab$ whose upper envelope is the geodesic radius function $r(x, \mathcal{H})$ for $x \in K$. Specifically, we use the shortest path trees $\bar{T}_a$ and $\bar{T}_b$ constructed in the previous section to build a set of $O(n + k)$ pairs f, I where:

- I is a subinterval of K and f is a function defined on domain I,
- each function has the form $f(x) = d_2(x, s) + \kappa$ where κ is a constant, d_2 is Euclidean distance, and s is a point or a line,
- for any point $x \in K$ the maximum of $f(x)$ over intervals I containing x is equal to $r(x, \mathcal{H})$.

For intuition, see Figure 5a, which shows several intervals and their associated functions. Note that we deal separately with the two pieces of the polygon on each side of K. There is an obvious set of $O(nk)$ pairs f, I with the above properties, one for each $H \in \mathcal{H}$ and $v \in P$, but we want a set of linear size.

The crucial property is that there are a limited number of farthest half-polygons associated with each vertex, and we can restrict our attention (mostly) to longest paths in the trees $\bar{T}_a$ and $\bar{T}_b$. In particular, consider a point $x \in K$, a half-polygon H, and the shortest path $\pi(x, H)$. Suppose $\pi(x, H)$ has at least two edges, say $\pi(x, H) = x, u, v, \ldots, t$, where t is the terminal point on H. If $\pi(x, H)$ turns left at u, then the part of the path from u to t is part of the tree $\bar{T}_a$, and symmetrically for right turns and $\bar{T}_b$. See Figure 5b. Furthermore, if H is a farthest half-polygon from x, then we can show (in the full version) that the part of the path from v to t is the longest path in $\bar{T}_a$ descending from v to any half-polygon. Note that using longest paths from u rather than v is not correct – see Figure 5b.

For any node u in $\bar{T}_a$ let $\ell_a(u)$ be the maximum length of a path in $\bar{T}_a$ descending from u to a node representing a terminal point on some half-polygon. Define $\ell_b(u)$ similarly. We can compute these functions in linear time in leaf-to-root order. In the situation described above, where H is a farthest half-polygon from x and the path $\pi(x, H) = x, u, v, \ldots, t$ turns left at u, we have $d(x, H) = |xu| + |uv| + \ell_a(v)$, which is a simple function of the desired form.

The algorithm will output these functions for all pairs u, v where the following conditions hold: u is a vertex visible from a point on the chord ab (equivalently, u has different parents

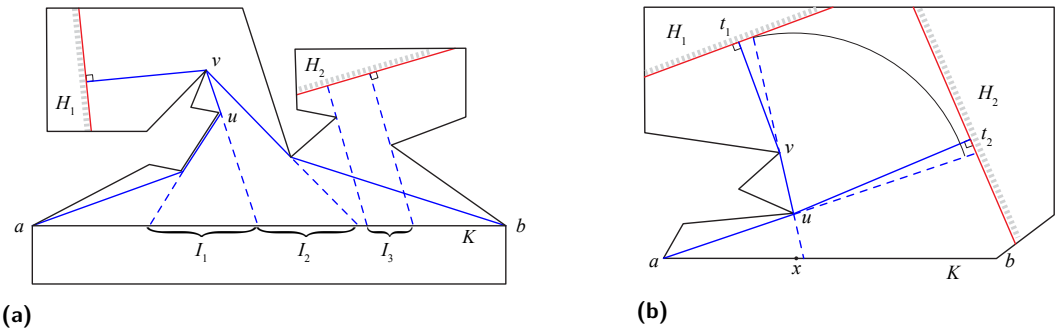

Figure 5 (a) An illustration of functions and intervals. For x in interval I_1, $d(x, H_1) = d_2(x, u)+\kappa_1$. For x in I_2, $d(x, H_1) = d_2(x, v) + \kappa_2$. For x in I_3, $d(x, H_2) = d_2(x, H_2)$. (b) From point x the farthest half-polygon is H_1 via the path x, u, v, t_1. This matches the longest path in $\bar{T}_a$ descending from v (which goes to H_1) but does not match the longest path descending from u (which goes to H_2).

in $\bar{T}_a$ and $\bar{T}_b$); v is a child of u in one of the trees; and x lies in an appropriate interval on ab such that u, v can be the start of a geodesic path from x. We deal separately with shortest paths that go from a point $x \in ab$ to a half-polygon without going through any vertices. Observe that the number of such functions is $O(n + k)$.

We defer further details of the algorithm to the long version of our paper. Besides enhancing the method of Pollack et al. [24] to deal with half-polygons, our aim is to give a clearer and easier-to-verify presentation.

4.1.4 Finding the Relative Geodesic Center on a Chord

The last step of the chord oracle is exactly the same as in Pollack et al. [24]. Given the set of $O(n + k)$ simple convex functions whose upper envelope is the geodesic radius function on chord K (from Section 4.1.3), and given the test of whether the relative center is left/right of a point on K (from Section 4.1.1) we want to find the relative center, c_K, that minimizes the radius function. Pollack et al. use a technique of Megiddo's to do this in $O(n + k)$ time. The idea is to pair up the functions, find the intersection and domain end-points of each pair, and test medians of those in order to eliminate a constant fraction of the functions in each round. Further details are in the long version of this paper. Finally, to find the paths from the relative center c_K to its farthest half-polygons, use the linear time shortest path algorithm (Section 4.1.2) on each side of K.

4.2 Finding the Geodesic Center of Half-Polygons

In this section we show how to use the $O(n + k)$ time chord oracle from Section 4.1 to find the geodesic center of the k half-polygons in $O((n + k) \log(n + k))$ time. The basic structure of the algorithm is the same as that of Pollack et al. [24].

In the first step we use the chord oracle to restrict the search for the geodesic center to a small region where the problem reduces to a Euclidean "intersection radius problem". In the second step we solve the resulting problem, which involves some new ingredients to handle our case of half-polygons. Each step takes $O((n + k) \log(n + k))$ time.

4.2.1 Finding a Region that Contains the Geodesic Center

Triangulate P in linear time [8]. Run the chord oracle on a chord of the triangulation that splits the polygon into balanced pieces and recurse on the appropriate subpolygon. In $O(\log n)$ iterations, we narrow our search down to one triangle T^* of the triangulation. This step takes $O((n+k)\log n)$ time.

Next, we refine T^* to a region R that contains the center and such that R is *homogeneous*, meaning that for any $H \in \mathcal{H}$ the shortest paths from points in R to H have the same combinatorial structure (the same sequence of polygon vertices along the path).

The idea is to subdivide T^* by $O(n+k)$ lines so that each cell in the resulting line arrangement is homogeneous, and then to find the cell containing the center. Construct the shortest path trees to $\mathcal{H}$ from each of the three corners of triangle $T^* = (a^*, b^*, c^*)$ using the algorithm of Section 4.1.2. For each edge (u, v) of each tree, add the line through uv if it intersects T^*. (In fact, we do not need all these lines. The situation is similar to that shown in Figure 5a, with a^*, b^* in place of a, b. We need the dashed lines shown in the figure. In particular, it suffices to use tree edges (u, v) such that u is visible from an edge of T^*.)

We add three more lines for each half-polygon $H \in \mathcal{H}$, specifically, the chord h that defines H, and the two lines perpendicular to h through the endpoints of h.

The result is a set of $O(n+k)$ lines that we obtain in time $O(n+k)$. It is easy to prove that the resulting line arrangement has homogeneous regions.

All that remains is to find the cell of the arrangement that contains the geodesic center. It is simpler to state the algorithm in terms of ϵ-nets instead of the rather involved description of Megiddo's technique used by Pollack et al. [24]. Informally, to find a homogeneous region, we will look at a range space on ground set L whose ranges consist of the subsets of L that intersect some triangle. Such a range space can be shown to have constant sized ϵ-nets [15]. By using the $O(n+k)$ time chord oracle a constant number of times (on a constant sized ϵ-net) we can restrict the search space to a region that intersects only a fraction of the original lines. Repeating this step for $O(\log(n+k))$ times, we arrive at a region R with the required properties. The details are deferred to the long version due to space constraints. The total runtime for this step is $O((n+k)\log(n+k))$.

4.2.2 Solving an Unconstrained Problem

At this point, we know that the polygonal region R contains a geodesic center of the set $\mathcal{H}$ of half-polygons in P. Furthermore, R is homogeneous. We can pick a point p in R and find the shortest path tree from p to all half-polygons. If $\pi(p, H)$ reaches an internal point of the chord h defining H then $d(x, H) = d_2(x, h)$ for all $x \in R$. And if the first segment of $\pi(x, H)$ reaches a vertex u, then $d(x, H) = d_2(x, u) + \kappa$ for all $x \in R$, where κ is a constant. Our goal is to find the point $x \in R$ that minimizes the maximum over $H \in \mathcal{H}$ of $d(x, H)$. Since this point must lie in the region R (a guarantee we have from the earlier steps), we can completely disregard the underlying polygon P in solving the problem.

In the Euclidean plane, the problem may be reinterpreted in a geometric manner. We wish to find the circle of smallest radius that *contains* $O(n)$ disks and *intersects* $O(k)$ lines. The disk constraints in the new interpretation correspond to half-polygons H where $d(x, H) = d_2(x, u) + \kappa$ – the disk is centered at u with radius κ. The line constraints correspond to half-polygons H where $d(x, H) = d_2(x, h)$.

This is a combination of problems referred to as the spanning circle problem [21] or the intersection radius problem [7] in the literature. We will call it the *intersection radius*

problem for disks and lines, although the name is not entirely accurate. In the long version of this paper we prove:

▶ **Lemma 11.** *The intersection radius problem for disks and lines can be solved in linear time.*

We outline the method. The problem for disks alone was solved by Megiddo [21] using an ingenious idea. The two-dimensional problem is modified to a problem in three dimensions and the constraints modified in such a way that the bisector between the constraints for two disks becomes a plane in three dimensions. Thereafter, techniques from linear-time algorithms for linear programming are used to prune away disks that do not define the final answer [20]. The prune-and-search technique prunes away a constant fraction of those disks in linear time and repeating the process reduces the number of disks to some constant number, after which a brute force method may be employed.

To extend this to handle our line constraints, we add constraints that ensure that the distances to the lines are less than the radius of the final disk. The bisectors between two lines are an angle bisector pair. These bisectors become vertical planes in the transformed three dimensional version of the problem. We thus have a set of planes in three dimensions that are bisectors between pairs of lines or pairs of disks. Using prune-and-search in two phases per iteration and a few other ideas ([7]), we modify Megiddo's technique for disks to solve our problem in linear time.

5 Conclusions

We introduced the notion of the visibility center of a set of points in a polygon and gave an algorithm with run time $O((n+m)\log(n+m))$ to find the visibility center of m points in an n-vertex polygon. To do this, we gave an algorithm to find the geodesic center of a given set of half-polygons inside a polygon, a problem of independent interest. We conclude with some open questions.

Can the visibility center of a simple polygon be found more efficiently? Note that the geodesic center of the vertices of a simple polygon can be found in linear time [1]. Our current method involves ray tracing and sorting, which are serious barriers. A more reasonable goal is to find the visibility center of m points in a polygon in time $O(n + m\log m)$.

Is there a more efficient algorithm to find the geodesic center of (sorted) half-polygons? In forthcoming work we give a linear time algorithm for the special case of finding the geodesic center of the *edges* of a polygon (this is the case where the half-polygons hug the edges).

How hard is it to find the farthest visibility Voronoi diagram of a polygon? Finally, what about the 2-visibility center of a polygon, where we can deploy two guards instead of one?

References

1 Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, Matias Korman, and Eunjin Oh. A linear-time algorithm for the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 56(4):836–859, 2016. doi:10.1007/s00454-016-9796-0.

2 Esther M Arkin, Alon Efrat, Christian Knauer, Joseph S B Mitchell, Valentin Polishchuk, Günter Rote, Lena Schlipf, and Topi Talvitie. Shortest path to a segment and quickest visibility queries. *Journal of Computational Geometry*, 7:77–100, 2016. URL: https://jocg.org/index.php/jocg/article/view/3001.

3 Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(3):217–255, 1993. doi:10.1145/73393.73417.

4 Franz Aurenhammer, Robert L Scot Drysdale, and Hannes Krasser. Farthest line segment Voronoi diagrams. *Information Processing Letters*, 100(6):220–225, 2006. doi:10.1016/j.ipl.2006.07.008.

5 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific Publishing Company, 2013.

6 Luis Barba. Optimal algorithm for geodesic farthest-point Voronoi diagrams. In *35th International Symposium on Computational Geometry (SoCG 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10416.

7 Binay K Bhattacharya, Shreesh Jadhav, Asish Mukhopadhyay, and J-M Robert. Optimal algorithms for some intersection radius problems. *Computing*, 52(3):269–279, 1994. doi:10.1007/bf02246508.

8 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991. doi:10.1007/bf02574703.

9 Wei-Pang Chin and Simeon Ntafos. Shortest watchman routes in simple polygons. *Discrete & Computational Geometry*, 6(1):9–31, 1991. doi:10.1007/bf02574671.

10 Hristo N Djidjev, Andrzej Lingas, and Jörg-Rüdiger Sack. An $O(n \log n)$ algorithm for computing the link center of a simple polygon. *Discrete & Computational Geometry*, 8(2):131–152, 1992.

11 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph SB Mitchell. Touring a sequence of polygons. In *Proceedings of the thirty-fifth Annual ACM Symposium on Theory of Computing (STOC '03*, pages 473–482, 2003. doi:10.1145/780542.780612.

12 Subir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

13 Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987. doi:10.1007/bf01840360.

14 Leonidas J Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989. doi:10.1016/0022-0000(89)90041-x.

15 David Haussler and Emo Welzl. ϵ-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, 1987.

16 John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *Journal of Algorithms*, 18(3):403–431, 1995. doi:10.1006/jagm.1995.1017.

17 Shreesh Jadhav, Asish Mukhopadhyay, and Binay Bhattacharya. An optimal algorithm for the intersection radius of a set of convex polygons. *Journal of Algorithms*, 20(2):244–267, 1996.

18 Yan Ke and Joseph O'Rourke. Computing the kernel of a point set in a polygon. In *Workshop on Algorithms and Data Structures*, pages 135–146. Springer, 1989.

19 Der-Tsai Lee and Franco P Preparata. An optimal algorithm for finding the kernel of a polygon. *Journal of the ACM (JACM)*, 26(3):415–421, 1979. doi:10.1145/322139.322142.

20 Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983. doi:10.1137/0212052.

21 Nimrod Megiddo. On the ball spanned by balls. *Discrete & Computational Geometry*, 4(6):605–610, 1989. doi:10.1007/bf02187750.

22 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. *Discrete & Computational Geometry*, 63(2):418–454, 2020.

23 Eunjin Oh, Luis Barba, and Hee-Kap Ahn. The geodesic farthest-point Voronoi diagram in a simple polygon. *Algorithmica*, 82(5):1434–1473, 2020. doi:10.1007/s00453-019-00651-z.

24 Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(6):611–626, 1989. doi:10.1007/bf02187751.

25 Godfried T Toussaint. Computing geodesic properties inside a simple polygon. *Revue d'Intelligence Artificielle*, 3(2):9–42, 1989.

26 Haitao Wang. Quickest visibility queries in polygonal domains. *Discrete & Computational Geometry*, 62(2):374–432, 2019. doi:10.1007/s00454-019-00108-8.

27 Haitao Wang. An optimal deterministic algorithm for geodesic farthest-point Voronoi diagrams in simple polygons. In *International Symposium on Computational Geometry (SoCG 2021)*, 2021. URL: https://arxiv.org/pdf/2103.00076.pdf.