Computing the k-Crossing Visibility Region of a Point in a Polygon

Yeganeh Bahoo[1], Prosenjit Bose[2], Stephane Durocher[1], and Thomas Shermer[3]

[1] University of Manitoba, Winnipeg, Canada {bahoo,durocher}@cs.umanitoba.ca
[2] Carleton University, Ottawa, Canada jit@scs.carleton.ca
[3] Simon Fraser University, Vancouver, Canada shermer@sfu.ca

Abstract. Two points p and q in a simple polygon P are k-crossing visible when the line segment pq crosses the boundary of P at most k times. Given a query point q, an integer k, and a polygon P, we propose an algorithm that computes the region of P that is k-crossing visible from q in $O(nk)$ time, where n denotes the number of vertices of P. This is the first such algorithm parameterized in terms of k, resulting in asymptotically faster worst-case running time relative to previous algorithms when k is $o(\log n)$, and bridging the gap between the $O(n)$-time algorithm for computing the 0-visibility region of q in P and the $O(n \log n)$-time algorithm for computing the k-crossing visibility region of q in P.

Keywords: Computational Geometry · Visibility · Radial Decomposition

1 Introduction

Given a simple n-vertex polygon P, two points p and q inside P are said to be mutually visible when the line segment pq does not intersect the exterior of P. Problems related to visibility are motivated by many applications that require covering a given region using a minimum number of resources, some of which refer to visual coverage (e.g., guarding with cameras [21, 16]) or to providing wireless connectivity coverage [19, 23]. Unlike the visible-light model, in which a viewer's line of sight typically terminates upon encountering a wall, radio transmissions can pass through some walls, suggesting a more general notion of visibility. Mouad and Shermer [20] introduced a generalized model of visiblity in polygons; this model was subsequently extended by Dean et al. [11] and Bajuelos et al. [4] to define *k-crossing visibility*. When p and q are in general position relative to the vertices of P (i.e., no vertex of P is collinear with p and q), p and q are mutually *k-crossing visible* when the line segment pq intersects the boundary of P in at most k points. Various applications require computing the region of P that is visible or k-crossing visible from a given query point q in P [1]. This region is called the *k-crossing visibility polygon* of q in P. See Figure 1.

Our goal is to design an algorithm that reduces the time required for computing the k-crossing visibility polygon for a given point q in a given simple polygon P. $O(n)$-time algorithms exist for finding the visibility polygon of q in

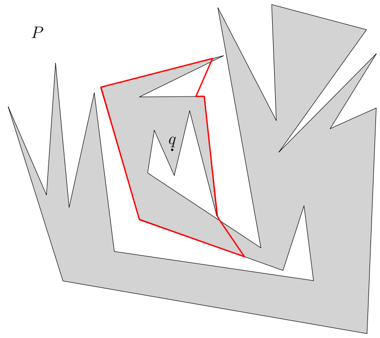


Fig. 1. a polygon P, a point q, and the k-crossing visibility polygon of q in P when $k = 2$

P (i.e., when $k = 0$) [13, 18, 17], whereas the best known algorithms for finding the k-crossing visibility polygon of q in P require $\Theta(n \log n)$ time in the worst case for any given k [3]. A natural question that remained open is whether the k-crossing visibility polygon of q in P can be found in $o(n \log n)$ time. In particular, can the problem be solved faster for small values of k? This paper presents the first algorithm parameterized in terms of k to compute the k-crossing visibility polygon of q in P. The proposed algorithm takes $O(nk)$ time, where n denotes the number of vertices of P, resulting in asymptotically faster worst-case running time relative to previous algorithms when k is $o(\log n)$, and bridging the gap between the $O(n)$-time algorithm for computing the 0-visibility polygon of q in P and the $O(n \log n)$-time algorithm for computing the k-crossing visibility polygon of q in P.

The paper begins with an overview of related work, followed by definitions, the presentation of the algorithm, and an analysis of its running time.

2 Related Work

Given a polygon P with n vertices and a query point q inside P, a fundemental problem in visibility is to compute the visibility polygon for q: the portion of P visible from q. This problem was first introduced by Davis and Benedikt [10], who gave an $O(n^2)$-time algorithm. The number of vertices of the visiblity polygon of q in P is proportional to the number of vertices of P in the worst case, i.e., $\Theta(n)$ [13, 18]. Algorithms for computing the visibility polygon for any given q and P in $O(n)$ time were given by Gindy and Avis [13], Lee [18], and Joe and Simpson [17].

This paper focuses on finding the k-crossing visibility polygon of q in P without preprocessing P. A related problem is that of preprocessing a given polygon P to construct a query data structure that answers one or more subsequent visibility queries for points given at query time. Using an $O(n^3)$-space data structure precomputed in $O(n^3 \log n)$ time, the visibility polygon of any point q given at query time can be reported in $O(\log n + m)$ time, where m denotes the number of vertices in the output polygon [6]. Finally, an $O(n^2)$-space data structure precomputed in $O(n^2 \log n)$ time can report the visibility polygon of any point q given at query time in $O(\log^2 n + m)$ time [2].

Motivated by applications in wireless networks, in which a radio transmission can pass through some walls before the signal fades, the problem of k-crossing visibility has attracted recent interest. Mouad and Shermer [20] first introduced the concept of k-crossing visibility, in what they originally called the *Superman problem*: given a simple polygon P, a sub-polygon $Q \subseteq P$, and a point q outside P, determine the minimum number of edges of P that must be made opaque such that no point of Q is visible to q. Dean et al. [11] studied pseudo-star-shaped polygons, in which the line of visibility can cross one edge, corresponding to k-crossing visibility where $k = 1$. Bajuelos et al. [4] subsequently explored the concept of k-crossing visibility for an arbitrary given k, and presented an $O(n^2)$-time algorithm to construct the k-crossing visible region of q in P for an arbitrary given point q. Recently, Bahoo et al. [3] examined the problem under the limited-workspace mode, and gave an algorithm that uses $O(s)$ words of memory and reports the k-visiblity polygon of q in P in $O(n^2/s+n \log s)$ time. When memory is not constrained (i.e., $\Omega(n)$ words of memory are available) their algorithm computes the k-visiblity polygon in $O(n \log n)$ time.

Additional results related to k-crossing visibility include generalizations of the well-known Art Gallery problem to the setting of k-crossing visibility. A set of points W in a polygon P is said to guard P if every point in P is k-crossing visible from some point in W. Each point (guard) in W is called a *k-modem*. The Art Gallery problem seeks to identify a set of point of minimum cardinality that guards a given polygon P. Aichholzer et al. [1] showed that $\lfloor n/2k \rfloor$ k-modems are sometimes necessary and $\lfloor n/(2k+2) \rfloor$ are always sufficient for guarding monotone polygons. They also proved that a monotone orthogonal polygon can be guarded by $\lfloor n/(2k+4) \rfloor$ k-modems. Duque et al. [12] showed that at most $O(n/k)$ k-modems are needed to guard a simple polygon P; however, given a polygon P, determining the minimum number of modems to guard P is NP-hard [7]. k-crossing visibility can be considered in the plane with obstacles, where the goal is to guard the plane or the boundary of a given region. Ballinger et al. [5] developed upper and lower bounds for the number of k-modems needed to guard a set of orthogonal line segments and other restricted families of geometric objects. Finally, given a set of line segments and a point q, Fabila et al. [14] examined the problem of determining the minimum k such that the entire plane is k-crossing visible from q.

3 Preliminaries and Definitions

3.1 Crossings and k-Crossing Visibility

Two paths P and Q are *disjoint* if $P \cap Q = \varnothing$. To provide a general definition of visibility requires a comprehensive definition for a crossing between a line segment and a polygon boundary, in particular, for the case when points are not in general position.

Definition 1 (Weakly disjoint paths [Chang et al. (2014)[8]]) *Two paths P and Q are* weakly disjoint *if, for all sufficiently small $\epsilon > 0$, there are disjoint paths $\tilde{P}$ and $\tilde{Q}$ such that $d_{\mathcal{F}}(P, \tilde{P}) < \epsilon$ and $d_{\mathcal{F}}(Q, \tilde{Q}) < \epsilon$.*

$d_{\mathcal{F}}(A, B)$ denotes the Fréchet distance between A and B.

Definition 2 (Crossing paths [Chang et al. (2014)[8]]) *Two paths* cross *if they are not weakly disjoint.*

Definitions 1 and 2 apply when P and Q are Jordan arcs. We use Definition 2 to help define k-crossing visibility.

Definition 3 (k-crossing visibility) *Two Jordan arcs (or polygonal chains) P and Q* cross k times*, if there exist partitions $P_1, \ldots, P_k$ of P and $Q_1, \ldots, Q_k$ of Q such that P_i and Q_i cross, for all $i \in \{1, \ldots, k\}$. Points p and q in a simple polygon P are* k-crossing visible *if the line segment pq and the boundary of P do not cross k times.*

Given a simple polygon P, we refer to the set of points that are k-crossing visible from a point q as the *k-crossing visibility region of q with respect to P*, denoted $\mathcal{V}_k(P, q)$. When the polygon P is clear from the context, we simply refer to set as the k-crossing visibility region of q and denote it as $\mathcal{V}_k(q)$. Our goal is to design an efficient algorithm to compute the k-crossing visibility region of a point q with respect to a simple polygon P.

To simplify the description of our algorithms, we assume that the query point q and the vertices of the input polygon P are in general position, i.e., q, p_i and p_j are not collinear for any vertices p_i and p_j in P. Under the assumption of general position, two points p and q are k-crossing visible if and only if the line segment pq intersects the boundary of P in fewer than k points. That is, Definition 3 is not necessary under general position. All results presented in this paper can be extended to input that is not in general position.

3.2 Trapezoidal and Radial Decompositions

A *polygonal decomposition* of a simple polygon P is a partition of P into a set of simpler regions, such as triangles, trapezoids, or quadrilaterals. Our algorithm uses trapezoidal decomposition and radial decomposition. A *trapezoidal decomposition* (synonymously, *trapezoidation*) of P partitions P into trapezoids and triangles by extending, wherever possible, a vertical line segment from each vertex p of P above and/or below p into the interior of P, until its first intersection with the boundary of P. A *radial decomposition* of P is defined relative to a point q in P. For each vertex p of P, a line segment is extended, wherever possible, toward/away from p into the interior of P on the line determined by p and q, until its first intersection with the boundary of P. A radial decomposition partitions P into quadrilateral and triangular regions. The number of vertices and edges in both decompositions is proportional to the number of vertices in P (i.e., $\Theta(n)$). Note that a trapezoidal decomposition corresponds to a radial decomposition when the point q has its y-coordinate at $+\infty$ or $-\infty$ (outside P). Chazelle [9] gives an efficient algorithm for computing a trapezoidal decomposition of a simple n-vertex polygon in $O(n)$ time.

4 k-Crossing Visibility Algorithm

4.1 Overview

Given as input an integer k, an array storing the coordinates of vertices whose sequence defines a clockwise ordering of the boundary of a simple polygon P, and a point q in the interior of P, our algorithm for constructing the k-crossing visibility polygon of q in P executes the following steps, each of which is described in detail in this section:

1. Partition P into two sets of disjoint polylines, corresponding to the boundary of P above the horizontal line ℓ through q, and the boundary of P below ℓ.
2. Nesting properties of Jordan sequences are used to close each set by connecting disjoint components to form two simple polygons, P_a above ℓ and P_b below ℓ.
3. The two-dimensional coordinates of the vertices of P_a and P_b are mapped to homogeneous coordinates, to which a projective transformation, f_q, is applied, with q as the center of projection.
4. Compute the trapezoidal decompositions of $f_q(P_a)$ and $f_q(P_b)$ using Chazelle's algorithm [9].
5. Apply the inverse tranformation f_q^{-1} on the trapezoidal decompositions to obtain radial decompositions of P_a and P_b.
6. Merge the radial decompositions of P_a and P_b to obtain a radial decomposition of P with respect to q.
7. Traverse the radial decomposition of P to identify the visibility of cells in increasing order from visibility 0 through visibility k, moving away from q and extending edges on rays from q to refine cells of the decomposition as necessary.
8. Traverse the refined radial decomposition to reconstruct and output the boundary of the k-crossing visibility region of q in P.

Steps 1–6 can be completed in $O(n)$ time and Steps 7–8 can be completed in $O(nk)$ time.

4.2 Partitioning P into Upper and Lower Polygons

We begin by describing how to partition the polygon P in two across the line ℓ, where ℓ denotes the horizontal line through q. By our general position assumption, no vertices of P lie on ℓ. Let ϵ denote the minimum distance between any vertex of P and ℓ. Let the *upper polygon*, denoted as P_a (respectively, the *lower polygon*, denoted P_b) refer to the closure of the region of the boundary of P that lies above (respectively, below) ℓ; see Figure 2. Let $\{x_1, \ldots, x_m\}$ denote the sequence of intersection points of ℓ with the boundary of P, labelled in clockwise order along the boundary of P, such that x_1 is the leftmost point in $P \cap \ell$. This sequence is a *Jordan sequence* [15]. We now describe how to construct P_a and P_b.

Between consecutive pairs (x_{2i-1}, x_{2i}) of the Jordan sequence, for $i \in \{1, \ldots, m/2\}$, the polygon boundary of P lies above ℓ. Similarly, between pairs (x_{2j}, x_{2j+1}), for $j \in \{1, \ldots, m/2 - 1\}$, and between (x_m, x_0), the boundary of P lies below ℓ. We call the former *upper pairs* of the Jordan sequence, and the latter *lower pairs*. These pairs possess the *nested parenthesis* property [22]: every two pairs (x_{2i-1}, x_{2i}) and (x_{2j-1}, x_{2j}) must either *nest* or be *disjoint*. That is, x_{2j-1} lies between x_{2i-1} and x_{2i} in the sequence if and only if x_{2j} lies between x_{2i-1} and x_{2i}.

As shown by Hoffmann et al. [15], the nested parenthesis property for the upper pairs determines a rooted tree, called the *upper tree*, whose nodes correspond to pairs of the sequence. The nodes in the subtree rooted at the pair (x_{2i-1}, x_{2i}) consist of all nodes corresponding to pairs that are nested betweeen x_{2i-1} and x_{2i} in the Jordan sequence order. The leaves of the tree correspond to pairs that are consecutive in the sorted order. If a node (x_{2j-1}, x_{2j}) is a descendant of a node (x_{2i-1}, x_{2i}) in the tree, then the points x_{2j-1} and x_{2j} are nested between x_{2i-1} and x_{2i}. The *lower tree* is defined analogously.

If the boundary of P intersects ℓ in more than two points, the resulting disconnected components must be joined appropriately to form the simple polygons P_a and P_b. To build the lower polygon P_b, we replace each portion of the boundary of P above ℓ from x_{2i-1} to x_{2i} with the following 3-edge path: x_{2i-1}, u, v, x_{2i}. The first edge (x_{2i-1}, u) is a vertical line segment of length $\epsilon/2d_i$, where d_i denotes the depth of the node (x_{2i-1}, x_{2i}) in the tree. The next edge (u, v) is a horizontal line segment whose length is $||x_{2i-1} - x_{2i}||$. The third edge (v, x_{2i}) is a vertical line segment of length $\epsilon/2d_i$. See Figure 2.

The nesting property of the Jordan sequence ensures that all of the 3-edge paths cross are similarly nested and that none of them intersect. Consider two pairs (x_{2i-1}, x_{2i}) and (x_{2j-1}, x_{2j}). Either they are disjoint or nested. If they are disjoint, then without loss of generality, assume that $x_{2i-1} < x_{2i} < x_{2j-1} < x_{2j}$. Their corresponding 3-edge paths cannot cross since the intervals they cover are disjoint. If they are nested, then without loss of generality, assume that $x_{2i-1} < x_{2j-1} < x_{2j} < x_{2i}$. The only way that the two paths can cross is if the horizontal edge for the pair (x_{2j-1}, x_{2j}) is higher than for the pair (x_{2i-1}, x_{2i}). However, since (x_{2j-1}, x_{2j}) is deeper in the tree than (x_{2i-1}, x_{2i}), the two paths do not cross. Thus, we form the simple polygon P_b by replacing the portions of the boundary above ℓ with these three edge paths. Sorting the Jordan sequence, building the upper tree, computing the depths of all the pairs and adding the 3-edge paths can all be achieved in $O(n)$ time using the Jordan sorting algorithm outlined by Hoffmann et al. [15]. The upper polygon P_a is constructed analogously. We conclude with the following lemma.

Lemma 1. *Given a simple n-vertex polygon P and a horizontal line ℓ that intersects the interior of P such that no vertices of P lie on ℓ, the upper and lower polygons of P with respect to ℓ can be computed in $O(n)$ time.*

4.3 Computing the Radial Decomposition

The two-dimensional coordinates of the vertices of each polygon P_a and P_b are mapped to homogeneous coordinates, to which a projective transformation, f_q, is applied with q as the center of projection. These transformations take constant time per vertex, or $\Theta(n)$ total time. Chazelle's algorithm [9] constructs trapezoidal decompositions of $f_q(P_a)$ and $f_q(P_b)$ in $\Theta(n)$ time, on which the inverse transformation, f_q^{-1} is applied to obtain radial decompositions of P_a and P_b. Merging the radial decompositions of P_a and P_b gives a radial decomposition of the original polygon P without requiring any additional edges. All vertices $x_1, \ldots, x_m$ of the Jordan sequence, all vertices of the three-edge paths, and their adjacent edges are removed. The remaining edges are either on the boundary of P, between two points on the boundary on a ray through q, or between the boundary and q. The entire process for constructing the radial trapezoidation takes $\Theta(n)$ time. This gives the following lemma.

Lemma 2. *The radial decomposition of a simple n-vertex polygon P around a query point q can be computed in $\Theta(n)$ time.*

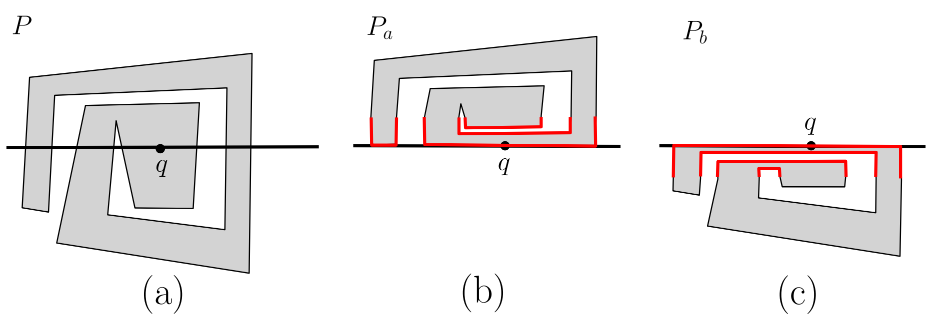


Fig. 2. (a) a polygon P, a point q, and the horizontal line ℓ through q; (b)–(c) the upper polygon P_a and lower polygon P_b of P with the additional 3-edge paths highlighted.

4.4 Reporting the k-Crossing Visible Region

The 0-visibility region of q in P, denoted $\mathcal{V}_0(q)$, is a star-shaped polygon with q in its kernel. A vertex of $\mathcal{V}_0(q)$ is either a vertex v of P or a point x on the boundary of P that is the intersection of an edge of P with a ray emanating from q through a reflex vertex r of P. In the latter case, (r, x) is an edge of $\mathcal{V}_0(q)$ that is collinear with q, called a *window* or *lid*, because it separates a region in the interior of P that is 0-visible from q and an interior region that is not 0-visible. The reflex vertex r is the *base* of the lid and x is its *tip*. There are two types of base reflex vertices. The reflex vertex r is called a *left base* (respectively, *right base*) if the polygon edges incident on r are to the left (respectively, right) of the ray emanating from q through r.

We now describe the algorithm to compute the k-crossing visible region of q in P, denoted $\mathcal{V}_k(q)$. The algorithm proceeds incrementally by computing $\mathcal{V}_{i+1}(q)$ after computing $\mathcal{V}_i(q)$. We begin by computing $\mathcal{V}_0(q)$ in $O(n)$ time using one of the existing linear-time algorithms, e.g. [13, 18, 17]. Label the vertices of $\mathcal{V}_0(q)$ in clockwise order around the boundary as $x_0, x_1, \ldots, x_m$. Triangulate the visibility polygon by adding the edge (q, x_i) for $i \in \{0, \ldots, m\}$; this corresponds to a radial decomposition of $\mathcal{V}_0(q)$ around q.

If x_i is a left base vertex, then notice that the triangle $\triangle(qx_ix_{i+1})$[4] degenerates to a segment. Similarly, if x_i is a right base vertex, then $\triangle(qx_ix_{i-1})$ is degenerate. If we ignore all degenerate triangles, then every triangle has the form $\triangle(qx_ix_{i+1})$, where (x_i, x_{i+1}) is on the boundary of P. The union of these triangles is $\mathcal{V}_0(q)$. To compute $\mathcal{V}_1(q)$, we show how to compute a superset of triangles whose union is $\mathcal{V}_1(q)$.

We start with an arbitrary triangle $\triangle(qx_ix_{i+1})$ of $\mathcal{V}_0(q)$, where (x_i, x_{i+1}) is on the boundary of P. Note that (x_i, x_{i+1}) is either an edge of P or a segment within the interior of an edge of P. It is this segment (x_i, x_{i+1}) of the boundary that blocks visibility. We show how to compute the intersection of $\mathcal{V}_1(q)$ with the cone that has apex q and bounding rays $\boldsymbol{qx_i}$ and $\boldsymbol{qx_{i+1}}$, denoted $\mathcal{C}(q, x_i, x_{i+1})$. We call this process *extending the visibility* of a triangle. We have two cases to consider. Either at least one of x_i or x_{i+1} is a base vertex or neither is a base vertex. We start with the latter case where neither is a base vertex.

Let Y be the set of vertices of the radial decomposition that lie on the edge (x_i, x_{i+1}). If Y is empty, then (x_i, x_{i+1}) lies on one face of the decomposition in addition to $\triangle(qx_ix_{i+1})$ since neither x_i nor x_{i+1} is a base vertex. We show how to proceed in the case when Y is empty, then we show what to do when Y is not empty. Let f be the face of the decomposition on the boundary of which (x_i, x_{i+1}) lies. By construction, this face is either a quadrilateral or a triangle. In constant time, we find the intersection of the boundary of f excluding the edge containing (x_i, x_{i+1}) with $\boldsymbol{qx_i}$ and $\boldsymbol{qx_{i+1}}$. Label these two intersection points as x'_i and x'_{i+1}. Extending the visibility of $\triangle(qx_ix_{i+1})$ results in $\triangle(qx'_ix'_{i+1})$. Note that $\triangle(qx'_ix'_{i+1})$ is the 1-visible region of q in $\mathcal{C}(q, x_i, x_{i+1})$ and (x'_i, x'_{i+1}) is on the boundary of P.

We now show how to extend the visibility of $\triangle(qx_ix_{i+1})$ when Y is not empty. Label the points of Y as y_j for $j \geq 1$ in the order that they appear on the edge (x_i, x_{i+1}) from x_i to x_{i+1}; see Figure 3. Each y_j is an endpoint of an edge of

[4] All indices are computed modulo the size of the corresponding vertex set: $m + 1$ in this case.

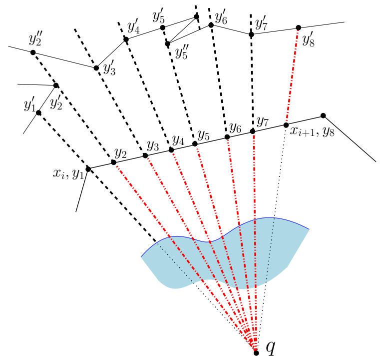


Fig. 3. Edges of the radial decomposition are extended where critical vertices cast a shadow. Portions of the polygon in the blue region that were processed in previous iterations are omitted from the figure.

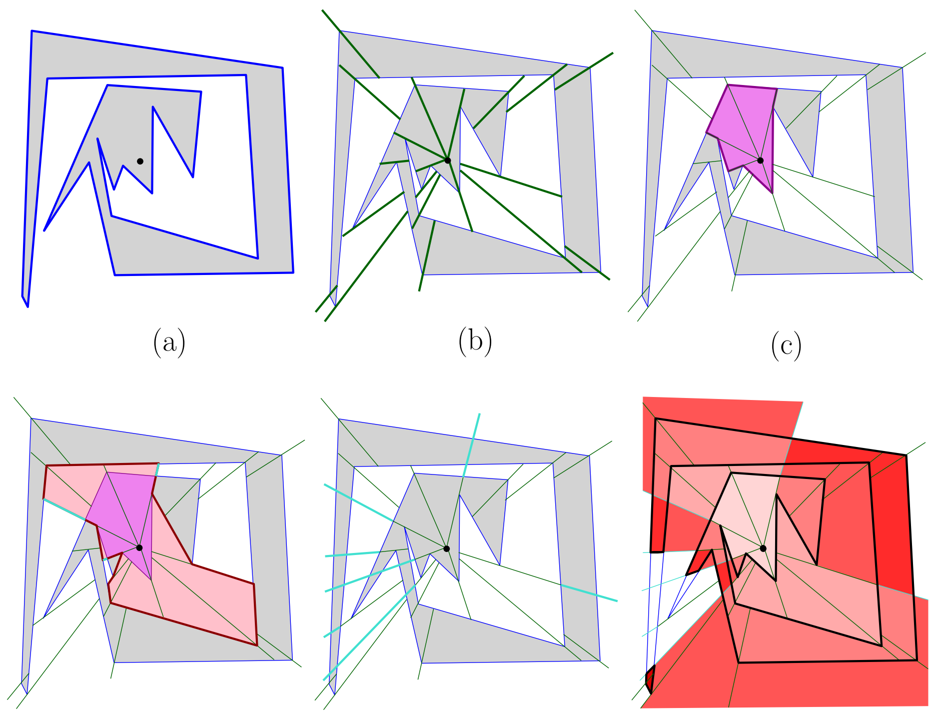
the radial decomposition. Since y_j is a point on the boundary of P, there are 2 faces of the radial decomposition with y_j on the boundary. Let y'_j be the other endpoint of the face on the left of y_j and y''_j be the endpoint for the face on the right. Either $y'_j = y''_j$ or $y'_j \neq y''_j$. In the former case, we simply ignore y''_j. In the latter case, we note that either y'_j is a left base of $\mathcal{V}_0(y_j)$ or y''_j is a right base. See Figure 3 where y'_2 is a left base and y''_5 is a right base.

Thus, the edges of the radial composition that intersect segment (x_i, x_{i+1}) are of the form (y_j, y'_j) or (y_j, y''_j). Note that y_1 is either x_i or the point closest to x_i on the edge. For notational convenience, if $y_1 \neq x_i$, relabel x_i as y_0. Let f be the face of the radial decompostion on the boundary of which (y_0, y_1) lies. Let y'_0 be the intersection of $\boldsymbol{qy_0}$ with the boundary of f excluding the edge of f containing (y_0, y_1). We call this operation *extending* x_i. Similarly, if $y_j \neq x_{i+1}$, relabel x_{i+1} as y_{j+1} and compute the edge (y_{j+1}, y'_{j+1}), i.e. *extend* x_{i+1}.

We are now in a position to describe the extension of the visibility of triangle $\triangle(qx_ix_{i+1})$ when neither x_i nor x_{i+1} is a base vertex. The set of triangles are $\triangle(qy'_ky'_{k+1})$ and $\triangle(qy''_ky'_{k+1})$ (when y''_k exists). The union of these triangles is the 1-visible region of q in $\mathcal{C}(q, x_i, x_{i+1})$. Furthermore, notice that each triangle $\triangle(qy'_ky'_{k+1})$ (respectively, $\triangle(qy''_ky'_{k+1})$) has the property that (y'_k, y'_{k+1}) (respectively, (y''_k, y'_{k+1})) is on the boundary of P. This is what allows us to continue incrementally since at each stage we extend the visibility of a triangle $\triangle(qab)$ where (a, b) is on the boundary of P.

Now, if x_i is a base vertex, then it must be a right base. Of the two edges of P incident on x_i, let e be the one further from q. The procedure to extend $\triangle(qx_ix_{i+1})$ is identical except that we only extend x_i when $x_{i+1} \in e$. Similarly, if x_{i+1} is a base vertex, then it must be a left base. Of the two edges of P incident on x_{i+1}, let e be the one further from q. Again, the procedure to extend $\triangle(qx_ix_{i+1})$ is identical except that we only extend x_{i+1} when $x_i \in e$.

The general algorithm proceeds as follows. At iteration i, the visibility region $\mathcal{V}_i(q)$ is represented as a collection of triangles around q with the property that



(a) (b) (c)

(d) (e) (f)

Fig. 4. (a) a simple polygon P and a query point q; (b) the radial decomposition of P; (c) the 0-visibility polygon, $\mathcal{V}_0(q)$, of q in P computed in the first iteration; (d) the 1-visibility polygon, $\mathcal{V}_1(q)$, of q in P computed in the second iteration, with extended edges highlighted in light blue; (e) the refined radial decomposition, with extended edges highlighted in light blue; (f) the 4-visibility polygon, $\mathcal{V}_4(q)$, of q in P computed in the fourth iteration, with the algorithm's output highlighted in black (two components of the boundary of $\mathcal{V}_4(q) \cap P$), and cells of the decomposition with depth ≤ 4 coloured by depth, as computed by the algorithm.

the edge of the triangle opposite q is on the boundary of P and it is the edge blocking visibility. We wish to extend past this edge to compute $\mathcal{V}_{i+1}(q)$ from $\mathcal{V}_i(q)$. To do this, we extend each triangle in $\mathcal{V}_i(q)$. There are at most $O(n)$ triangles at each level. Therefore, the total time to extend all the triangles in $\mathcal{V}_i(q)$ is linear. Thus, we can compute $\mathcal{V}_{i+1}(q)$ from $\mathcal{V}_i(q)$ in $O(n)$ time and computing $V_k(q)$ takes $O(nk)$ time since we repeat this process k times.

The algorithm can report either only the subregion of P that is k-crossing visible from q, i.e., $\mathcal{V}_k(q) \cap P$, or the entire region of the plane that is k-crossing visible from q, including parts outside P. To obtain the region inside P, it suffices to traverse the boundary of P once to reconstruct and report portions of boundary edges that are k-crossing visible. The endpoints of these sequences of edges on the boundary of P meet an edge of the refined radial decomposition through the interior of P that bridges to the start of the next sequence on the boundary of P. The entire boundary of P must be traversed since the k-crossing visible region in P can have multiple connected components (unlike the k-crossing visible region in the plane that is a single connected region). See Figure 4 for an example. We conclude with the following theorem.

Theorem 4. *Given a simple polygon P with n vertices and a query point q in P, the region of P that is k-crossing visible from q can be computed in $O(kn)$ time without preprocessing.*

5 Discussion

This paper presents the first algorithm parameterized in terms of k for computing the k-crossing visible region for a given point q in a given polygon P, resulting in asymptotically faster worst-case running time relative to previous algorithms when k is $o(\log n)$, and bridging the gap between the $O(n)$-time algorithm for computing the 0-visibility region of q in P [13, 18, 17], and the $O(n \log n)$-time algorithm for computing the k-crossing visibility region of q in P [3]. It remains open whether the problem can be solved faster. In particular, an $O(n \log k)$-time algorithm would provide a natural parameterization for all k. Alternatively, can a lower bound of $\Omega(n \log n)$ be shown on the worst-case time when k is $\omega(\log n)$?

References

1. Aichholzer, O., Fabila-Monroy, R., Flores-Peñaloza, D., Hackl, T., Huemer, C., Urrutia, J., Vogtenhuber, B.: Modem illumination of monotone polygons. Computational Geometry **68**, 101–118 (2018)
2. Aronov, B., Guibas, L.J., Teichmann, M., Zhang, L.: Visibility queries and maintenance in simple polygons. Discrete & Computational Geometry **27**(4), 461–483 (2002)
3. Bahoo, Y., Banyassady, B., Bose, P., Durocher, S., Mulzer, W.: A time-space tradeoff for computing the k-visibility region of a point in a polygon. Theoretical Computer Science (2018)

4. Bajuelos, A.L., Canales, S., Hernández, G., Martins, M.: A hybrid metaheuristic strategy for covering with wireless devices. Journal of Universal Computer Science **18**(14), 1906–1932 (2012)
5. Ballinger, B., Benbernou, N., Bose, P., Damian, M., Demaine, E., Dujmović, V., Flatland, R., Hurtado, F., Iacono, J., Lubiw, A., et al.: Coverage with k-transmitters in the presence of obstacles. Journal of Combinatorial Optimization **25**(2), 208–233 (2013)
6. Bose, P., Lubiw, A., Munro, J.I.: Efficient visibility queries in simple polygons. Computational Geometry **23**(3), 313–335 (2002)
7. Cannon, S., Fai, T., Iwerks, J., Leopold, U., Schmidt, C.: Combinatorics and complexity of guarding polygons with edge and point 2-transmitters. arXiv preprint arXiv:1503.05681 (2015)
8. Chang, H.C., Erickson, J., Xu, C.: Detecting weakly simple polygons. In: Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA 2014). pp. 1655–1670 (2014)
9. Chazelle, B.: Triangulating a simple polygon in linear time. Discrete & Computational Geometry **6**(3), 485–524 (1991)
10. Davis, L.S., Benedikt, M.L.: Computational models of space: Isovists and isovist fields. Computer Graphics and Image Processing **11**(1), 49–72 (1979)
11. Dean, J.A., Lingas, A., Sack, J.R.: Recognizing polygons, or how to spy. The Visual Computer **3**(6), 344–355 (1988)
12. Duque, F., Hidalgo-Toscano, C.: An upper bound on the k-modem illumination problem. arXiv preprint arXiv:1410.4099 (2014)
13. El Gindy, H., Avis, D.: A linear algorithm for computing the visibility polygon from a point. Journal of Algorithms **2**(2), 186–197 (1981)
14. Fabila-Monroy, R., Vargas, A., Urrutia, J.: On modem illumination problems. Proc. XIII Encuentros de Geometria Computacional (EGC 2009) (2009)
15. Hoffmann, K., Mehlhorn, K., Rosenstiehl, P., Tarjan, R.E.: Sorting Jordan sequences in linear time using level-linked search trees. Information and Control **68**(1-3), 170–184 (1986)
16. Huang, H., Ni, C.C., Ban, X., Gao, J., Schneider, A.T., Lin, S.: Connected wireless camera network deployment with visibility coverage. In: Proc. IEEE International Conference on Computer Communications (INFOCOM 2014). pp. 1204–1212 (2014)
17. Joe, B., Simpson, R.B.: Corrections to Lee's visibility polygon algorithm. BIT Numerical Mathematics **27**(4), 458–473 (1987)
18. Lee, D.T.: Visibility of a simple polygon. Computer Vision, Graphics, and Image Processing **22**(2), 207–221 (1983)
19. Meguerdichian, S., Koushanfar, F., Qu, G., Potkonjak, M.: Exposure in wireless ad-hoc sensor networks. In: Proc. 7th ACM International Conference on Mobile Computing and Networking (MOBICOM 2001). pp. 139–150. ACM (2001)
20. Mouawad, N., Shermer, T.C.: The Superman problem. The Visual Computer **10**(8), 459–473 (1994)
21. Murray, A.T., Kim, K., Davis, J.W., Machiraju, R., Parent, R.: Coverage optimization to support security monitoring. Computers, Environment and Urban Systems **31**(2), 133–147 (2007)
22. Rosenstiehl, P.: Planar permutations defined by two intersecting Jordan curves. Graph Theory and Combinatorics pp. 259–271 (1984)
23. Wang, Y.C., Hu, C.C., Tseng, Y.C.: Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks. In: Proc. 1st IEEE Conference on Wireless Internet (WICON 2005). pp. 114–121 (2005)