Generalized Visibility Kernel

Eyüp Serdar Ayaz[1] and Alper Üngör[1]

1 **CISE Department, University of Florida, Gainesville**
[ayaz,ungor]@cise.ufl.edu

Abstract

We propose a generalization of the concept visibility kernel which finds use in art gallery problems. For a point p in a simple polygon P and a subset X of P, we refer the set of points visible by every point in X that is seen by p as the generalized visibility kernel of p with respect to X. We present an $O(n + m \log m)$ time algorithm for computing the generalized visibility kernel, where n, m are the complexities of P and X respectively. As essential components of our approach, we also propose two efficient algorithms for computing the relative convex hull and the complete visibility polygon of a set of points.

1 Introduction

We consider a generalized version of the well known Art Gallery Problem. For a polygon P, and two subsets of it, $X, Y \subseteq P$, AGP(X, Y) asks to find the minimum subset of X as guard locations, such that all the points of Y are guarded [4]. A recent approach for solving art gallery problems involves use of witness sets [2–4]. A *witness set* W of a polygon P is defined as a set such that any set G that guards W also guards P. In [1], we extended the notion of witness sets for the parametrized version AGP(X, Y) and outlined an iterative refinement scheme for the art gallery problem. Formulation and computation of visibility kernels turns out to be an essential step of this refinement scheme. For a point $p \in P$, the set of points visible by every point in P that is seen by p is called the *visibility kernel* of p [3], denoted as $\mathcal{VK}(p)$. We present a generalization of this concept. For a point $p \in P$ and $X \subseteq P$, the set of points visible by every point in X that is seen by p is called the *generalized visibility kernel* of p with respect to X, denoted as $\mathcal{GVK}(p, X)$. Note that $\mathcal{VK}(p) = \mathcal{GVK}(p, P)$. While this generalization is seemingly simple, several algorithmic challenges arise.

Contribution. In Section 3, we present a $O(n+m \log m)$ time algorithm for computing generalized visibility kernel, where n, m are the complexities of P and X, respectively. Our algorithm for computing generalized visibility kernel involves computation of relative convex hulls. *Relative convex hull* of a set of points S is defined as the region with minimum circumference that includes S within a polygon P [10]. In Section 3.2, we present an efficient algorithm for computing the relative convex hull of a set of points in a visibility polygon of a point. Our algorithm works in $O(n + m)$ time when the result of Section 3.1 is given in $O(n+m \log m)$ time cumulatively. Previously, an algorithm is given to find a single viewpoint that sees a given relative convex hull in $O(n + m)$ time [6]. We present an algorithm that calculates the set of all viewpoints of a set of points in the same time complexity.

2 Preliminaries and definitions

The input for $AGP(X, Y)$ is a simple (non-convex) polygon P with n vertices and two sets $X, Y \subseteq P$. We assume that X is given as a union of non-intersecting sets of points, line segments, and convex polygonal regions within P. Let m denote the complexity of X. For any polygon Q, ∂Q denotes the boundary of Q and $ref(Q)$ denotes the reflex vertices of Q.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Two points $p, q \in P$ *see* each other if the whole line segment $\overline{pq}$ is in P. The set of points in P that can be seen from a point $p \in P$ is called the *visibility polygon* of p, denoted as $\mathcal{V}(p)$ (See Figure 1). The set of points that can see every point in $\mathcal{V}(p)$ is called the *visibility kernel* of p, denoted as $\mathcal{VK}(p)$. For a set of points $S \subseteq P$, the set of points that are visible from all points in S is called its *complete visibility polygon* [7]. It is denoted as $\mathcal{CV}(S)$ and can be formulated as $\bigcap_{p \in S} \mathcal{V}(p)$.

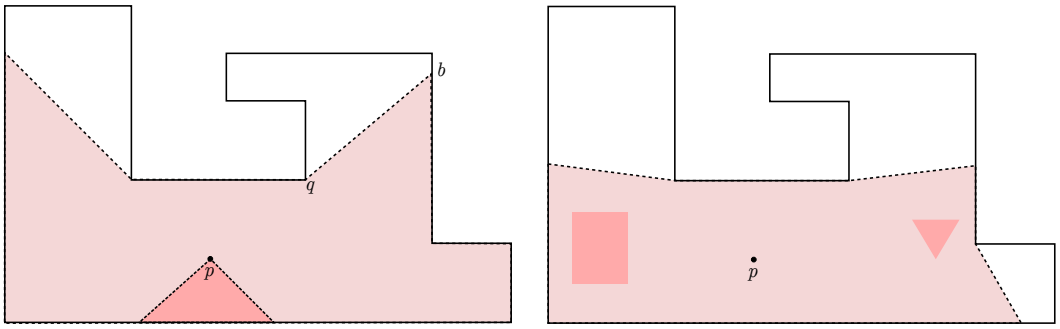

Figure 1 (Left) Visibility polygon $\mathcal{V}(p)$ (shaded area) and visibility kernel $\mathcal{VK}(p)$ (dark shaded area) of a point p. (Right) For a given set X (dark shaded area), $\mathcal{GVK}(p, X)$ is the shaded area.

The counter-clockwise (CCW) angle defined by ordered three points x, p, y is denoted as $\widehat{xpy}$. The shortest path within a polygon P between two points $p, q \in P$ is denoted as $SP(P, p, q)$. A *chain* is an ordered list of j line segments $s_1 ... s_j$ between $j + 1$ points $p_1 ... p_{j+1}$ such that $s_i = \overline{p_i p_{i+1}}$ for $1 \leq i \leq j$. A chain on a boundary of a polygon P between two points $q_1, q_2 \in \partial P$, is defined as the CCW chain on ∂P from q_1 to q_2, denoted as $chain(P, q_1, q_2)$. The half plane on the left of the ray $\overrightarrow{pq}$ is denoted as $HP(\overrightarrow{pq})$. The closure of the half plane is denoted as $CHP(\overrightarrow{pq})$. A *wedge* is determined by three points $l, p, r \in P$ and denoted as $wedge(l, p, r)$. If $\widehat{lpr} \leq \pi$, then $wedge(l, p, r) = \mathcal{V}(p) \cap HP(\overrightarrow{lp}) \cap HP(\overrightarrow{pr})$. Otherwise, $wedge(l, p, r) = \mathcal{V}(p) \cap (HP(\overrightarrow{lp}) \cup HP(\overrightarrow{pr}))$.

2.1 Generalized visibility kernel and its properties

From the definitions, we simply have $\mathcal{GVK}(p, X) = \mathcal{CV}(\mathcal{V}(p) \cap X)$. Also note that if a guard $g \in X$ sees p, then g is guaranteed to see all points in $\mathcal{GVK}(p, X)$. Indeed this very observation has been the motivation for us to propose a generalization of visibility kernels. Below, we list some more properties.

▶ **Lemma 2.1.** *Given two sets $X_1, X_2 \subseteq P$ such that $X_1 \subseteq X_2$. For any point $p \in P$ we have $\mathcal{GVK}(p, X_2) \subseteq \mathcal{GVK}(p, X_1)$.*

▶ **Lemma 2.2.** *For $p \in P$ and $X \subseteq P$, we have $\mathcal{VK}(p) \subseteq \mathcal{GVK}(p, X)$.*

▶ **Lemma 2.3.** *For $p \in P$ and $X \subseteq P$, we have $p \in \mathcal{GVK}(p, X)$.*

▶ **Lemma 2.4.** *The following statements are equivalent for $p, q \in P$ and $X \subseteq P$: (i) Any point $r \in X$ that sees p also sees q; (ii) $q \in \mathcal{GVK}(p, X)$; (iii) $\mathcal{GVK}(q, X) \subseteq \mathcal{GVK}(p, X)$.*

▶ **Theorem 2.5.** *A point set $W \subseteq P$ is a witness set for Y with respect to X if and only if $Y \subseteq \bigcup_{p \in W} \mathcal{GVK}(p, X)$.*

3 Algorithms

Our algorithm for computing $\mathcal{GVK}(p, X) = \mathcal{CV}(\mathcal{V}(p) \cap X)$ consists of four steps. Due to the space limitations, we present the summary of the algorithms in this extended abstract. The details of the algorithms can be found in the full version of the paper.

1. First, we use the linear-time algorithm by Joe and Simpson [8] to compute $\mathcal{V}(p)$.
2. Second, we present a angular plane sweep algorithm to find $\mathcal{V}(p) \cap X$ in Section 3.1.
3. Third, we calculate the relative convex hull of $\mathcal{V}(p) \cap X$ in Section 3.2.
4. Finally, we compute the complete visibility polygon of $\mathcal{V}(p) \cap X$ in Section 3.3 relying on the relative convex hull computed in Step 3.

3.1 Calculating $\mathcal{V}(p) \cap X$

After computing $\mathcal{V}(p)$, we divide $\mathcal{V}(p)$ into wedges $W_i = wedge(r_i, p, r_{i+1})$, where r_i is the ith reflex vertex of $\mathcal{V}(p)$ in CCW order. Assume that $r_1 = r_{j+1}$ where j is the number of reflex vertices. Without loss of generality, we assume that $\widehat{r_j p r_1} \geq \widehat{r_i p r_{i+1}}$ for $1 \leq i < j$. Note that each wedge, except W_j in some cases, is convex. (See Figure 2).

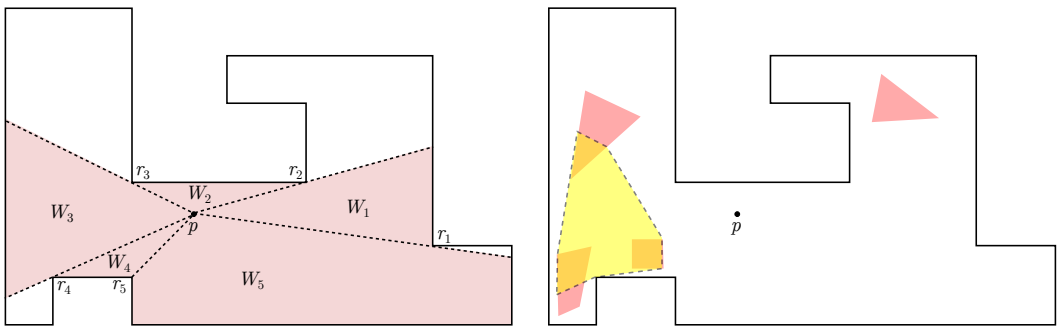

Figure 2 (Left) $\mathcal{V}(p)$ is partitioned into wedges with apices of p. (Right) For a given X (pink and orange shaded areas), $\mathcal{V}(p) \cap X$ is the orange shaded area. Yellow and orange shaded areas together shows $\mathcal{RCH}(\mathcal{V}(p) \cap X)$

.

Our sweepline is a ray anchored at p. We record the geometric objects in X that intersects the sweepline in a balanced binary search tree ($BBST$) in the order of the distance from p to the object. Since the objects in X are convex, the order of the nodes do not change. A node is active if p is closer to the corresponding object than ∂P in the direction of the sweepline. A node of the BBST records the geometric object in X, whether the object is active, and the event points where the node is activated and deactivated.

The event points in our angular plane sweep algorithm are the vertices of X and reflex vertices of $\mathcal{V}(p)$ in the angular order from p. For this, we sort the elements of X in CCW order with respect to the viewpoint p. The node corresponding to an object $x \in X$ is inserted to (deleted from) the BBST at the rightmost (leftmost) vertex of x with respect to p. The object x can be activated or deactivated at a reflex vertex $r \in ref(\mathcal{V}(p))$. We prune x through the sweepline when it is activated or deactivated. The output $\mathcal{V}(p) \cap X$ is recorded as a sequence of nodes sorted in the angular order of the appearance from p.

Each element of $x \in X$ is added to and removed from BBST once, and activated and/or inactivated at most once. Add, remove operates in $O(\log m)$ time, activation and inactivation takes $O(1)$ time per element. Calculating the cutting points of x takes an amortized

$O(1)$ time. Data structure operations in total takes $O(n + m \log m)$ time and sorting takes $O(m \log m)$. Therefore the total complexity of calculating $\mathcal{V}(p) \cap Y$ is $O(n + m \log m)$.

3.2 Calculating the relative convex hull of $\mathcal{V}(p) \cap X$

To calculate $\mathcal{CV}(\mathcal{V}(p) \cap X)$, we use the *relative convex hull* for a set of points S in a polygon P, denoted as $\mathcal{RCH}(S)$ [10]. $\mathcal{RCH}(S)$ is defined as the polygon that has the minimum circumference such that $S \subseteq \mathcal{RCH}(S) \subseteq P$. Note that $\mathcal{RCH}(S)$ can be a degenerate polygon (See Figure 3).

Ghosh [6] states that if a point p in a simple polygon P sees a set of points $S \subseteq P$ if and only if p sees $\mathcal{RCH}(S)$. Based on this, we can conclude that $\mathcal{CV}(\mathcal{V}(p) \cap X) = \mathcal{CV}(\mathcal{RCH}(\mathcal{V}(p) \cap X))$. Let us call $\mathcal{RCH}(\mathcal{V}(p) \cap X)$ as Q for the rest of the paper for brevity.

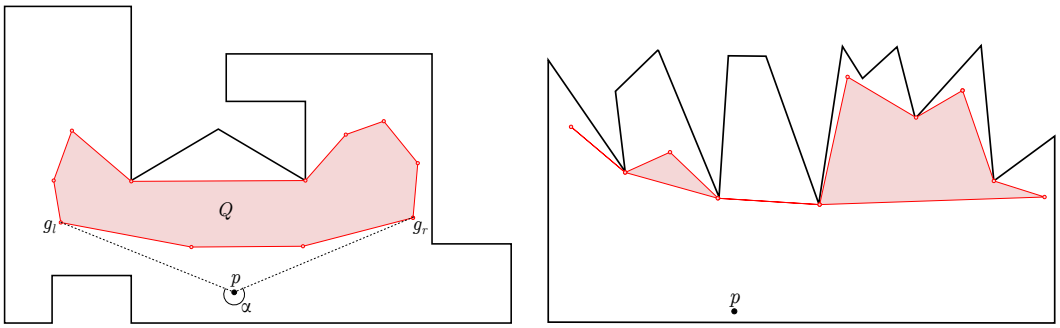

Figure 3 (Left) Q is the relative convex hull of $\mathcal{V}(p) \cap X$ for a given X. The near chain is $chain(Q, g_l, g_r)$, the far chain is $chain(Q, g_r, g_l)$, and α is the outer angle between the tangents from p to Q. (Right) A degenerate relative convex hull.

▶ **Lemma 3.1.** *For a point $p \in P$ and a set $X \subseteq P$, we have $\mathcal{RCH}(\mathcal{V}(p) \cap X) \subseteq \mathcal{V}(p)$.*

Let $g_l, g_r \in Q$ be the points that make the angle $\alpha = \widehat{g_l p g_r}$ maximal such that $wedge(g_l, p, g_r) \cap X = \emptyset$. (See Figure 3). In other words, if $p \notin Q$, then g_l and g_r are the tangent points from p to Q. We decide whether p is in Q based on α.

▶ **Lemma 3.2.** *The point p is in $\mathcal{RCH}(\mathcal{V}(p) \cap X)$, if and only if $\alpha \leq \pi$ or $p \in X$.*

To calculate Q, we define two chains based on whether p is in Q or not. If $p \notin Q$, then $chain(Q, g_r, g_l)$ is the *near chain* and $chain(Q, g_l, g_r)$ is the *far chain*. If $p \in Q$, then ∂Q is the *far chain*. Based on Lemma 3.2, if $\alpha > \pi$ and $p \notin X$, we calculate both the near chain and the far chain. Otherwise, we calculate only the far chain.

For each wedge W_i, we calculate the convex hull of $X \cap W_i$ by calculating the far and (if necessary) the near chains. From the previous calculation, we have $\mathcal{V}(p) \cap X$ in CCW order from the viewpoint of p. We traverse the points in the wedge by updating the far chain in CCW from p. While traversing, if the current point creates a right turn in the far chain, then the previous points are popped from the far chain until there is no right turn in the chain. The process is similar for the near chain.

After calculating the the far and (if necessary) the near chains for each wedge, we merge each of them to calculate Q. The merge is also done in CCW order from p, using the reflex vertices of $\mathcal{V}(p)$ as pivot points and draw tangents to the near and the far chains of the wedges. If $\alpha > \pi$ and $p \in X$, then we concatenate line segments $\overline{g_l p}$ and $\overline{g_r p}$ to the far chain to yield Q. If $\alpha > \pi$ and $p \notin X$, we concatenate the near chain and the far chain. If $\alpha \leq \pi$, only the far chain gives us Q.

The time complexity of calculating Q is $O(n+m)$, since there are $O(n+m)$ points to be considered in the convex hull and each vertex in $\mathcal{V}(p) \cup X$ and $ref(P)$ is traversed once and inserted to and/or removed from a chain at most once.

3.3 Calculating the complete visibility polygon of $\mathcal{RCH}(\mathcal{V}(p) \cap X)$

Here, we present an algorithm to calculate the complete visibility polygon of Q as a sub-method of calculating the $\mathcal{GVK}(p, X)$. However, p is not particularly relevant in our algorithm other than the assumption of the existence of at least one viewpoint for the relative convex hull.

Let k be the number of reflex vertices of Q. Let us define $r_i \in ref(Q)$ as the reflex vertices of Q in the CCW order, where $1 \leq i \leq k$. Without loss of generality, let us assume that r_1 is the reflex vertex that maximizes the angle $\widehat{r_k p r_1}$ and i is an integer such that $1 \leq i \leq k$. We also assume that $r_{k+1} = r_1$. Now, we calculate $\mathcal{CV}(Q)$ by handling different cases depending on some properties of Q.

Case 1: $k = 0$, or $k = 1$. Ghosh [7] gave an algorithm to find the complete visibility polygon of a given convex chain within a simple polygon in linear time. When $k = 0$ i.e., Q is convex, $\mathcal{CV}(Q)$ can be calculated directly using his algorithm. If Q has only one reflex vertex, then we can represent ∂Q as a chain starting and ending at r_1 that has only left turns. This allows us to use the same algorithm to calculate $\mathcal{CV}(Q)$.

Case 2: $k \geq 2$. We have the following property:

▶ **Lemma 3.3.** *If $\widehat{r_i p r_{i+1}} \leq \pi$, then r_i and r_{i+1}, see each other.*

From Lemma 3.3, we yield that there can be at most one consecutive reflex vertex pair in Q that cannot see each other. Then, we have the following sub-cases:

Case 2.1: r_1 and r_k see each other. For each pair of consecutive reflex vertices $r_i, r_{i+1} \in ref(Q)$, we denote the polygon bounded by $chain(Q, r_i, r_{i+1})$ and $\overline{r_i r_{i+1}}$ as Q_i. We call the rest of Q after removing all Q_i's as Q_{core} (See Figure 4).

Case 2.1.1: Q_k is convex. For each Q_i, we define a superset P_i, which is bounded by $\overline{r_i r_{i+1}}$ and $chain(P, r_i, r_{i+1})$. We calculate the complete visibility polygon of Q_i inside P_i using [7] and call it C_i. Then, we create a polygon that is bounded by $chain(C_i, r_i, r_{i+1})$ for all i and call it as G. For each reflex vertex r_i, two incident vertices to r_i on ∂Q are denoted as t_i and u_i, such that t_i, r_i, u_i is the CCW order of these vertices on ∂Q. Then we have $\mathcal{CV}(Q) = G \cap \bigcap_{i=1}^{k} (CHP(\overrightarrow{t_i r_i}) \cap CHP(\overrightarrow{r_i u_i}))$.

We use a similar approach as Lee and Preparata's algorithm [9] to calculate the half plane intersections. Our algorithm starts with initializing a partially bounded intersection region K as $CHP(\overrightarrow{t_1 r_1}) \cap CHP(\overrightarrow{r_1 u_1})$. We initially assign the left and right tangents from r_1 to K as $\overrightarrow{t_1 r_1}$ and $\overrightarrow{u_1 r_1}$ respectively. Then, we traverse the reflex vertices of Q in CCW order starting from r_1. In the ith step, we trim K with the rays $\overrightarrow{r_i u_i}$, $\overrightarrow{t_{i+1} r_{i+1}}$ and $chain(G, r_i, r_{i+1})$ using [5] and update the tangents from r_i to K. After k steps, K yields us $\mathcal{CV}(Q)$.

Case 2.1.2: Q_k is not convex. In this case G will not be a simple polygon. However, we have $\mathcal{GVK}(Q) \subseteq C_k \subseteq CHP(\overrightarrow{r_1 t_1})$ assuming that r_1 is a reflex vertex of Q_k without loss of generality. We initialize C_k to K and iteratively trim K through the reflex vertices of Q using the half plane intersections.

Case 2.2: r_1 and r_k do not see each other. This case is similar to Case 2.1.2, except in this case, Q_k is defined as the polygon bounded by $chain(Q, r_k, r_1)$ and $SP(r_1, r_k)$. Similarly, P_k is bounded by $chain(P, r_k, r_1)$ and $SP(r_1, r_k)$. We calculate C_k as the intersection of the complete visibility polygon of $chain(Q, r_k, r_1)$ and P_k. Then, we use the same algorithm initializing $C_k \cap CHP(\overrightarrow{t_1 r_1}) \cap CHP(\overrightarrow{r_1 u_1})$ to K using [5].

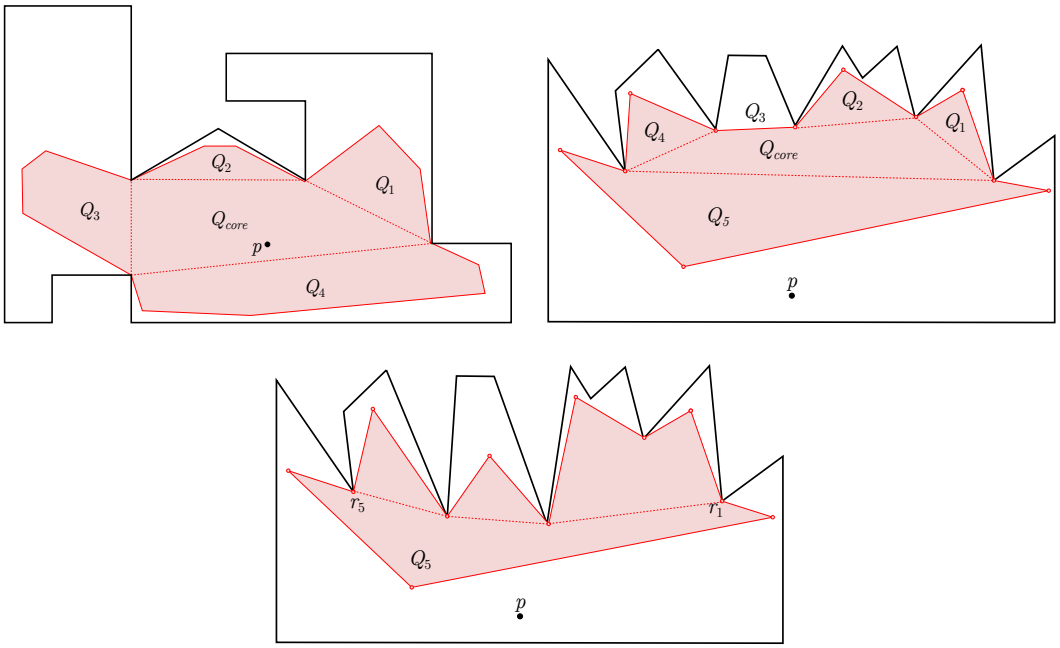

Figure 4 Partition of Q for Cases (Upper Left) 2.1.1, (Upper Right) 2.1.2, (Lower) 2.2.

These cases cover all the possible situations assuming that there exists at least one point in P that sees every point in Q. We can also spot whether $\mathcal{CV}(Q)$ is empty if there are more than one r_i, r_{i+1} pairs that do not see each other or K becomes empty at some step in the half plane intersections. All the submethods can be calculated in linear time with respect to the number of vertices in P, Q, Q_{core} and G all of which are in $O(n+m)$.

References

1 E. S. Ayaz and A. Üngör. An iterative refinement scheme of dominating guards and witnesses for art gallery problems. In *Canadian Conf. on Comp. Geom.*, pages 168–174, 2016.

2 E. S. Ayaz and A. Üngör. Minimal witness sets for art gallery problems. In *European Workshop on Computational Geometry (EuroCG)*, pages 195–198, 2016.

3 K. Chwa, B. Jo, C. Knauer, E. Moet, R. van Oostrum, and C. Shin. Guarding art galleries by guarding witnesses. *Int. J. Comput. Geometry Appl.*, 16(2-3):205–226, 2006.

4 P. J. de Rezende, C. C. de Souza, S. Friedrichs, M. Hemmer, A. Kröller, and D. C. Tozoni. Engineering art galleries. *CoRR*, abs/1410.8720, 2014.

5 S. K. Ghosh. A linear-time algorithm for determining the intersection type of two star polygons. In *Foundations of Software Technology and Theoretical Computer Science*, pages 317–330. Springer, 1984.

6 S. K. Ghosh. Computing a viewpoint of a set of points inside a polygon. In *Foundations of Software Technology and Theoretical Computer Science*, pages 18–29. Springer, 1988.

7 S. K. Ghosh. Computing the visibility polygon from a convex set and related problems. *Journal of Algorithms*, 12(1):75 – 95, 1991.

8 B. Joe and R. B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987.

9 D. T. Lee and F. P. Preparata. An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26(3):415–421, July 1979.

10 G. T. Toussaint. An optimal algorithm for computing the relative convex hull of a set of points in a polygon. In *Signal Processing: Theories and Applications*. North-Holland, 1986.