Weak Visibility Counting in Simple Polygons

Mojtaba Nouri Bygi[a], Shervin Daneshpajouh[a], Sharareh Alipour[a], Mohammad Ghodsi[a,b]

[a] *Computer Engineering Department, Sharif University of Technology, Iran*
[b] *School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Iran*

Abstract

For a simple polygon P of size n, we define weak visibility counting problem (WVCP) as finding the number of visible segments of P from a query line segment pq. We present different algorithms to compute WVCP in sub-linear time. In our first algorithm, we spend $O(n^7)$ time to preprocess the polygon and build a data structure of size $O(n^6)$, so that we can optimally answer WVCP in $O(\log n)$ time. Then, we reduce the preprocessing costs to $O(n^{4+\epsilon})$ time and space at the expense of more query time of $O(\log^5 n)$. We also obtain a trade-off between preprocessing and query time costs. Finally, we propose an approximation method to reduce the preprocessing costs to $O(n^2)$ time and space and $O(n^{1/2+\epsilon})$ query time.

Keywords: Computational Geometry, Weak Visibility, Visibility Counting

1. Introduction

Two points inside a polygon are *visible* to each other if their connecting segment remains completely inside the polygon. *Visibility polygon* of a point p in a simple polygon P is the set of points in P that are visible from p.

In many problems, it is good to have a sense of the size of visible area before computing it. The visibility counting problem (VCP) in a polygonal domain is to find the number of objects (segments, edges, etc) that are visible from a point p. An inefficient solution would be to compute the visibility polygon of p and then report its size.

For a simple polygon, Bose *et al.* showed that by preprocessing the polygon in $O(n^3 \log n)$ time and building a data structure of size $O(n^3)$, one can answer VCP for a query point in $O(\log n)$ time [2]. For a set of n disjoint line segments, Suri and O'Rourke introduced the first 3-approximation algorithm for VCP, by representing the visibility polygon of the segments as a set of convex (triangular) regions [16]. Gudmundsson and Morin improved this result to a 2-approximation algorithm by using an improved covering scheme [11]. The same result achived by [1] and [13]. Alipour and Zarei proved that the number of visible end-points of the segments is a 2-approximation of VCP [1].

There are two other approximation algorithms for VCP by Fischer *et al.* [8, 9]. The first algorithm uses a data structure of size $O((m/r)^2)$ by which the queries are answered in $O(\log n)$ time, with an absolute error of r ($1 \leq r \leq n$). Here, m is the size of Visibility Graph, which is $O(n^2)$ [10]. The second algorithm uses a random sampling method to build a data structure of size $O((m^2 \log^{O(1)} n)/l)$ to answer any query in $O(l \log^{O(1)} n)$ time, where $1 \leq l \leq n$.

The visibility problem has also been considered for line segments. A point v is said to be *weakly visible* from a line segment pq if there exists a point $w \in pq$ such that w and v are visible to each other. The problem of computing the *weak visibility polygon* (or *WVP*) of pq inside a polygon P is to compute all points of P that are weakly visible from pq. If P is a simple polygon, $WVP(pq)$ can be computed in linear time [12]. Also, Nouri Bygi and Ghodsi showed that $WVP(pq)$ can be computed in output sensitive query time of $O(\log n + |WVP(pq)|)$, by building a data structure of size $O(n^3)$ in $O(n^3 \log n)$ time [14, 15].

1.1. Our Contributions

In this paper we consider the weak visibility counting problem (WVCP) for line segments in simple polygons. We define the weak visibility count of a line segment inside a simple polygon as the size of the weak visibility polygon of the line segment.

	Prep. Time	Size	Query Time
Naive	-	$O(n)$	$O(n)$
Exact	$O(n^7)$	$O(n^6)$	$O(\log n)$
Exact	$O(n^{4+\epsilon})$	$O(n^{4+\epsilon})$	$O(\log^5 n)$
Exact	$O(n^{3+\epsilon})$	$O(n^3)$	$O(n^{1/2+\epsilon})$
Approx	$O(n^2)$	$O(n^2)$	$O(n^{1/2+\epsilon})$

Table 1: A summary of our results on WVCP. The first part shows exact algorithms, and the second part shows the approximation result.

Email addresses: nouribygi@ce.sharif.edu (Mojtaba Nouri Bygi), daneshpajouh@ce.sharif.edu (Shervin Daneshpajouh), alipour@ce.sharif.edu (Sharareh Alipour), ghodsi@sharif.edu (Mohammad Ghodsi)

Our approaches are divided into two categories: exact counting and approximate counting. In the first part of the paper, we present two algorithms that compute the exact size of the weak visibility polygon in sub-linear time, after preprocessing the polygon. In the second part, we show how to reduce the preprocessing costs of the exact algorithms, by approximating the counting problem. A summary of our results is given in Table 1.

The rest of this paper is organized as follows. In Section 2, we review some definitions and structures that will be used throughout the paper. In Section 3, we present three algorithms for computing WVCP. We also give an approximation algorithm based on random sampling in Section 4. Section 5 concludes the paper.

2. Preliminaries

Let P be a polygon with total vertices of n. Also, let p be a point inside P. The *visibility sequence* of a point p is the sequence of vertices and edges of P that are visible from p. A *visibility decomposition* of P is to partition P into a set of *visibility regions*, such that any point inside each region has the same visibility sequence (see Figure 1). This partition is induced by *critical constraint edges* (or critical edges), which are the lines in the polygon, each induced by two vertices of P, such that the visibility sequences of the points on its two sides are different.

The visibility sequences of two *neighboring* visibility regions which are separated by an edge, differ only in one vertex. This fact is used to reduce the space complexity of maintaining the visibility sequences of the regions [2]. By constructing a directed dual graph over the visibility regions, one can maintain the difference between the visibility sequences of the neighboring regions [2].

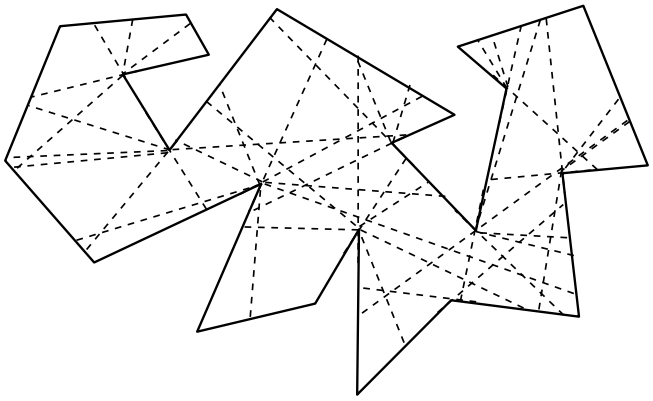

Figure 1: The visibility decomposition induced by critical constraints.

In a simple polygon with n vertices, the number of critical edges and visibility regions are $O(n^2)$ and $O(n^3)$, respectively [2].

2.1. Range Searching

Assume that there is a set P of n points in d-dimensional space. In range searching problems, one can report or count the points lying in a region R in this space. In this paper, we use the results of this problem when P is a set of points in the plane and R is a half-plane.

Chazelle *et al.* [4] introduced a simplex range searching method that answers queries in $O(n^{1-1/d+\epsilon})$ time by using $O(n^{1+\epsilon})$ preprocessing time and $O(n)$ space, for any arbitrary small positive constant ϵ. They also obtained a trade-off between storage and query time. They showed that one can build a data structure of size $O(m)$ in time $O(m^{1+\epsilon})$, where $n \leq m \leq n^d$, so that the query can be answered in $O(\frac{n^{1+\epsilon}}{m^{1/d}})$ time.

In another work [5], Chazelle showed that by using cuttings, a data structure of size $O(n^{d+\epsilon})$ can be used to answer the queries in $O(\log n)$ time. This structure can be constructed in $O(n^{d+\epsilon})$ time.

The results of range searchings are given as disjoint union of some canonical subsets. One can further preprocess these canonical subsets, so that a series of range searchings can be answered on the original point set [7]. Using a range searching data structure in a multilevel fashion does not increase the required space, and the query time increases only by a logarithmic factor.

3. Exact Weak Visibility Counting

A naive approach to solve the weak visibility counting problem is to compute the weak visibility polygon of the query line segment and then, enumerate the size of the resulted polygon. As the computation of the WVP can cost $O(n)$ time, this approach costs $O(n)$ time.

In this section, we show how to preprocess a simple polygon P such that, given a query line segment pq in the query time, we can compute the size of $WVP(pq)$ in sub-linear time. Our approaches are based on the visibility decomposition of the polygon.

First, we show that it is sufficient to consider the visibility regions of the endpoints of the query line segment.

Lemma 1. *Suppose that the points p and p' are in the visibility region R, and q and q' are in the visibility region R'. We have $WVC(pq) = WVC(p'q')$.*

Proof. As p and p' are in the same visibility region, they see the same sequence of points and edges of P, and we have $SPT(p) = SPT(p')$ (see Figure 2). Here, $SPT(p)$ is the shortest path tree in P rooted at p. The same argument is valid for q and q'. We know that the algorithm of Guibas *et al.* [12] for computing the weak visibility polygon of a segment pq in a polygon P, is based on $SPT(p)$ and $SPT(q)$. Therefore, we can conclude that Guibas's algorithm returns the same result as the $WVP(pq)$ and $WVP(p'q')$.

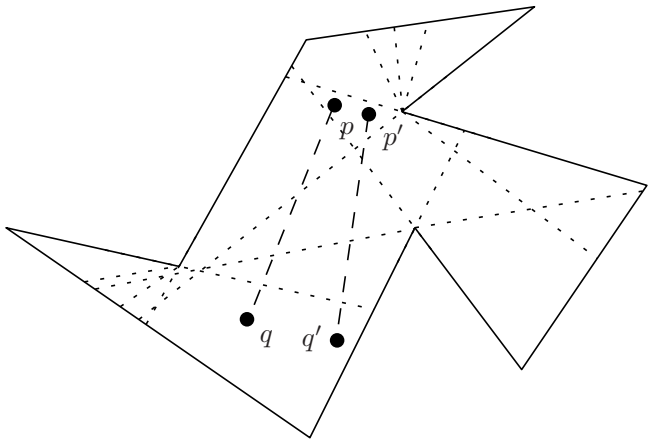

Figure 2: The weak visibility polygon of a query line segment is recognized by the regions of its endpoints.

Having Lemma 1, we focus on the visibility regions of the endpoints of the line segment. In the following subsections, we present two algorithms for computing the exact value of WVCP, which differ on preprocessing and query costs.

3.1. Optimal Query Time Algorithm

Consider the visibility decomposition of polygon P. We say that two visibility regions S_i and S_j are visible to each other if there is a point $s_i \in S_i$ and a point $s_j \in S_j$ such that s_i and s_j are visible. We can determine if two visibility regions are visible to each other and, if so, find a pair of (s_i, s_j) in $O(n)$ time.

As the weak visibility polygon can be identified by the visibility regions of the query line segment endpoints, the idea is to compute $WVP(s_i s_j)$ for each pair of regions (S_i, S_j) in $O(n)$ time and store its size. We also build a point location data structure for the visibility decomposition to find the corresponding regions of the query line segment.

As there are $O(n^6)$ pairs of regions, we need $O(n^7)$ time and $O(n^6)$ space to consider all the possible pairs. In the query time and upon receiving the query line segment pq, we find the regions of p and q and report the corresponding weak visibility size in $O(\log n)$ time.

Theorem 2. *Using $O(n^7)$ time to preprocess a simple polygon P and memory size of $O(n^6)$, it is possible to report the size of* WVP *for a query line segment in $O(\log n)$ time.*

3.2. Reducing Preprocessing Costs

Although Theorem 2 finds the output efficiently, its high preprocessing costs motivates us to find more efficient algorithms. In this section, we show how to use multi-level range searching to reduce the preprocessing costs, in expense of more query time.

To compute the weak visibility count of a query line segment pq, we start from the endpoint p and compute the initial visibility count of the viewpoint. Then, we move toward q, and maintain the visibility count as we proceed. When we arrive at q, we can report the maintained count as weak visibility count of pq.

The first issue is to compute the initial visibility count of the viewpoint at p. To do this, we use the following lemma:

Lemma 3. *[2] A simple polygon P can be preprocessed in $O(n^3 \log n)$ time and $O(n^3)$ space such that, given an arbitrary query point inside the polygon, it takes $O(\log n)$ time to give the number of visible vertices.*

As we move from p toward q, we may cross some of the critical constraints of the polygon. Crossing each critical constraint corresponds to a gain of visibility in one direction, and a loss in the opposite direction. We label the two sides of a critical constraint as $+$ or $-$, according to this gain or loss (see Figure 3). If we cross a critical constraint from its $+$ side, we call it a *pos*-cross, otherwise it is a *neg*-cross. As loosing a visibility does not affect our desired visibility count, we only need to consider *pos*-crosses. More precisely, each *pos*-cross corresponds to an additive value in the visibility count of pq.

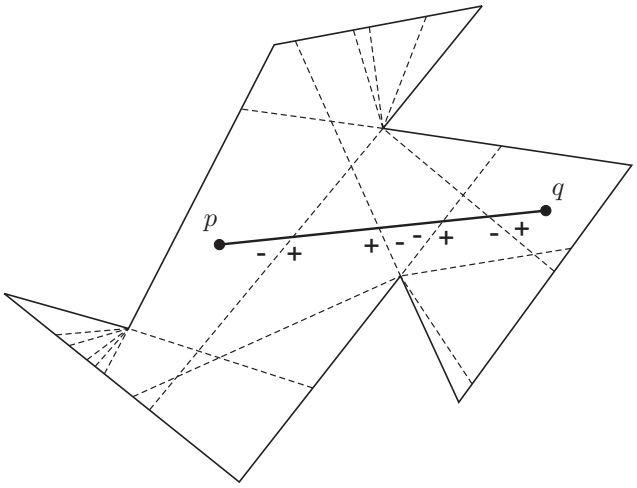

Figure 3: We can solve WVCP by adding the visibility count of p to the number of *pos*-crosses from p to q.

Lemma 4. *[2] Given a line segment ab in P, if a point z sees a and b, then z sees every point on the line segment ab.*

Lemma 5. *Assume that the line segment pq can see a vertex v of the polygon. Let $X = \{x \in pq | x \text{ can see } v\}$, then, X is a sub-segment ab of pq, and a is either p or an intersection of pq at a pos-cross (in $\overrightarrow{pq}$ direction), and b is either q or an intersection of pq at a neg-cross (in $\overrightarrow{pq}$ direction).*

Proof. This can be proved by using Lemma 4, and the fact that each critical constraint corresponds to a unique visibility change in its crossing sides.

According to Lemma 5, to compute the visibility count of a segment pq, we can add the initial visibility count of p to the number of *pos*-crosses in the path from p to q. We can compute the visibility count of the initial point p, using Lemma 3. Therefore, we have to count the number of critical constraints that intersect with pq and have p on their + sides. To solve this problem, we combine a series of half-plane range searching.

Assume that each critical constraint s is defined by its two endpoint s_{right} and s_{left}, and l is the supporting line of the query line segment pq. First, we find those critical constraints that intersect with l. To do this, we use half-plane range searching to find those segments whose left endpoint lies above l. In the result set, we should select those segments whose right endpoint lies below l. This can be done by running the half-plane range searching on the results of the first range searching. In the third level, we filter the result set by those segments that have p on their + sides.

Next, we select those segments of the result set which supporting lines intersect with pq. This can be mapped to two additional range searchings in dual space. Remember that in dual space, the supporting lines of the critical constraints will map to points, and pq will map to a double wedge (see Figure 4). Therefore, we must answer two range searchings in the dual space. More precisely, we find those dual points that lie between the dual lines of p and q.

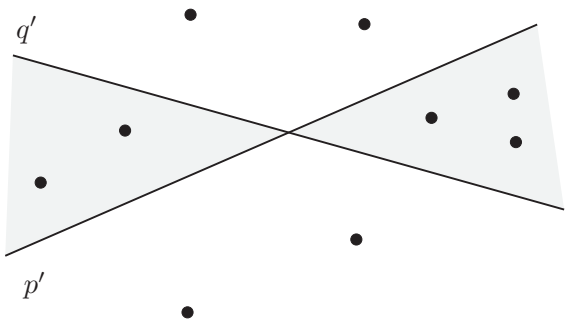

Figure 4: In dual space, the lines that intersects pq are mapped to points inside the double wedge $p'q'$. These points are those below p' and above q', or above p' and below q'.

Notice that we must repeat the same procedure for those critical constraints whose right endpoint lies above l and left endpoint lies below l.

As discussed in Section 2.1, this 5-level searching can be solved in $O(\log^5 k)$ time, using $O(k^{2+\epsilon})$ preprocessing time and space, where k is the number of critical constraints. As the number of critical constraints in a simple polygon of size n is $O(n^2)$, the preprocessing time and space needed by the multi-level range searching are $O(n^{4+\epsilon})$. We also need to use the algorithm of Lemma 3 to compute the visibility count of p. The preprocessing time and space needed by this algorithm are $O(n^3 \log n)$ and $O(n^3)$, respectively.

In the query time, we need $O(\log n)$ time to retrieve the visibility count of p, and $O(\log^5 n)$ time to compute the number of *pos*-crossings of pq with the critical constraints. In general we have:

Theorem 6. *Using $O(n^{4+\epsilon})$ time and space to preprocess a simple polygon P, it is possible to report the size of WVP for a query line segment in $O(\log^5 n)$ time.*

3.3. Trade-off between Preprocessing and Query Time Costs

Using the trade-off in range searching data structures [4] (see Section 2.1), we can obtain a trade-off between preprocessing costs and query time in our problem. If there are k points in the plane, one can spend storage of size $O(m)$, where $k \leq m \leq k^2$, to build a range searching data structure, such that the query can be answered in $O(\frac{k^{1+\epsilon}}{\sqrt{m}})$ time. Also, performing a multi-level range searching can be done with the same bounds [4].

In our scenario, k is the number of critical constraints and is $O(n^2)$. We also need to use Lemma 3. Therefore, we have the following result:

Theorem 7. *Using $O(n^3 \log n + m^{1+\epsilon})$ time and $O(n^3 + m)$ space to preprocess a simple polygon P, it is possible to report the size of WVP for a query line segment in $O(\frac{n^{2+\epsilon}}{\sqrt{m}})$ time, where $n^2 \leq m \leq n^4$.*

As a result, the minimum preprocessing costs achieve when we select $m = n^3$. In this case, the preprocessing time and space are $O(n^{3+\epsilon})$ and $O(n^3)$, respectively, and the query time is $O(n^{1/2+\epsilon})$.

4. Approximation of Weak Visibility Counting

Although the query time of the algorithms presented in Section 3 satisfy our requirements, preprocessing costs of these algorithms are hight. Therefore, in this section, we seek for approximate solutions.

For computing the approximate value of WVCP, we need to be sure that the weak visibility polygon is large enough, so that the random sampling approach is actually working. For this, we need to compute WVP in an output sensitive way such that, as soon as we find out that it is large enough, we stop the algorithm.

Lemma 8. *[6] A simple polygon P can be preprocessed in $O(n)$ time and space such that, given an arbitrary query line segment inside the polygon, $WVP(pq)$ can be computed in $O(k \log n)$ time, where k is the size of the output that is to be reported.*

Using this lemma, we can check whether the size of $WVP(pq)$ is larger than $n^{1/2+\epsilon}$ in $O(\sqrt{n} \log n)$ time. This result will be used in Section 4.2, when we take a sample set of size $n^{1/2+\epsilon}$ from the edges of the polygon.

4.1. Weak Visibility Testing

To approximate the value of WVCP, we also need to quickly check whether the query line segment pq can see an arbitrary edge of the polygon. The following theorem shows how to do this.

Lemma 9. *A simple polygon P can be processed in time $O(n^2)$ into a data structure of size $O(n^2)$ so that, for any query line segment pq, and an arbitrary edge e of P, we can detect whether pq and e are weakly visible in $O(\log n)$ time.*

PROOF. Consider $WVP(e)$, the weak visibility polygon of an edge e in P, i.e., the points of P that are weakly visible from e. This polygon has size $O(n)$ and can be computed in time $O(n)$. In the preprocessing time, we compute $WVP(e)$ for all the edges of the polygon. For each of these weak visibility polygons, we construct the ray shooting data structure [3] in $O(n)$ time and space, so that we can answer the ray shooting queries in $O(\log n)$ times. In query time, we perform two ray shooting queries from p and q to find out whether pq intersects with $WVP(e)$.

Computing $WVP(e)$ for all the edges of P and preparing a ray shooting data structure for each one of them can be done in $O(n^2)$ time and space. Also, the query time needed by the algorithm is $O(\log n)$.

4.2. Random Sampling

Using the algorithm of Section 4.1, we propose an algorithm to approximate the answer of WVCP. Our approach consists of two phases. In the first phase, we run the algorithm of Lemma 8 until we find $O(\sqrt{n})$ edges of the WVP. If the algorithm is finished, we have the exact value of m_p where m_p is the answer of WVCP. Otherwise, $k > O(\sqrt{n})$ and we choose a random subset $R_i \subset S$, such that each segment is chosen with the probability of $1/\sqrt{n}$. Next, for each segment $s \in R_1$, we check whether it is weakly visible from pq or not, using the algorithm of Section 4.1.

We choose t random subsets like R_1. Let X_i be the number of visible segments from pq in R_i. We report $m'_p = \frac{\sum_{i=1}^{t} \sqrt{n} X_i}{t}$ as the approximated value of m_p.

Lemma 10. *(Chebyshev's Lemma) Let $X_1, X_2, \ldots, X_t$ be any random variable with $E(X_i) = \mu$, and let $\epsilon > 0$ be any positive real number. Then*

$$P((|\frac{X_1 + X_2 + \ldots + X_t}{t} - \mu|) > \epsilon) \leq \frac{Var(X)}{t\epsilon^2}$$

Using lemma 10, we have

Lemma 11. *With a probability close to 1 we have*

$$(1-\delta)m_p \leq \sqrt{n}\frac{X_1 + X_2 + \ldots + X_t}{t} \leq (1+\delta)m_p$$

PROOF. In Lemma 10, we choose $\epsilon = \delta m_p$, where δ is a constant number which can be made arbitrarily small in the preprocessing time, and $t = 1/\delta$. We have $E(\sqrt{n}X_i) = m_p$, and $Var(\sqrt{n}X_i) = nm_p(1 - \frac{1}{\sqrt{n}})\frac{1}{\sqrt{n}}$. Therefore,

$$\mathbb{P} = P(|\sqrt{n}\frac{X_1 + X_2 + \ldots + X_t}{t} - m_p| > \delta m_p) \leq \frac{\sqrt{n}m_p}{\delta^2 m_p^2}$$

Since $m_p > O(n^{1/2+\epsilon})$, then $\mathbb{P} \sim 0$. This means that with the probability of at least $1 - \mathbb{P}$, we have

$$(1-\delta)m_p \leq m'_p \leq (1+\delta)m_p$$

Theorem 12. *WVCP can be approximated in $O(1/\delta n^{1/2+\epsilon})$ time, using $O(n^2)$ preprocessing time and space. The algorithm returns a value m'_p such that, if $m_p > \sqrt{n}$, with a high probability, $(1-\delta)m_p \leq m'_p \leq (1+\delta)m_p$. If $m_p \leq \sqrt{n}$, the algorithm returns the exact value of WVCP.*

PROOF. In the first phase of the algorithm we use the Lemma 8, which needs preprocessing time and space of $O(n)$ and query time of $O(n^{1/2+\epsilon})$. In the second phase, according to the Lemma 9, by using $O(n^2)$ preprocessing time and building a data structure of size $O(n^2)$, we can check whether pq and a random selected segment are weakly visible or not in $O(\log n)$ time. Since for each R_i we choose $O(\sqrt{n})$ random segments, and $t = 1/\delta$ is a constant value, the query time is $O(1/\delta n^{1/2+\epsilon} + \log n\sqrt{n}) = O(1/\delta n^{1/2+\epsilon})$ Therefore, the query time for both phases is $O(n^{1/2+\epsilon})$ and the preprocessing time and space are $O(n^{2+\epsilon})$.

5. Conclusion

In this paper, we studied the problem of computing the size of weak visibility polygon in simple polygons. We obtain two different exact solutions that solve the problem in sub-linear time. We also obtained a trade-off between preprocessing costs and query time costs, which can be used to reduce the preprocessing cost even more. Finally, we showed how to reduce the preprocessing time even more, by approximately computing the desired value via random sampling.

One can investigate WVCP in polygonal domains. In this case, WVP have $O(n^4)$ complexity [16]. Therefore, extending our results for polygonal domains would be challenging.

References

[1] S. Alipour, A. Zarei. Visibility Testing and Counting. *FAW-AAIM*, pages 343–351 , 2011.

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

[3] B. Chazelle and L.J. Guibas. Visibility and intersection problems in plane geometry. *Discrete & Computational Geometry*, 4, 551-581, 1989.

[4] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8(5&6), 407–429, 1992.
[5] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9:145–158, 1993.
[6] D. Z. Chen and H. Wang. Weak visibility queries of line segments in simple polygons. In *23rd International Symposium, ISAAC*, pages 609–618, 2012.
[7] D.P. Dobkin and H. Edelsbrunner. Space searching for intersecting objects. *J. Algorithms*, 8(3), 348361 1987.
[8] M. Fischer, M. Hilbig, C. Jahn, F. Meyer auf der Heide, and M. Ziegler. Planar visibility counting. *CoRR, abs/0810.0052*, 2008.
[9] M. Fischer, M. Hilbig, C. Jahn, F. Meyer auf der Heide, and M. Ziegler. Planar visibility counting. In *Proceedings of the 25th European Workshop on Computational Geometry (EuroCG 2009)*, pp. 203–206, 2009.
[10] S.K. Ghosh and D.M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM J. Comput.*, 20:888–910, 1991.
[11] J. Gudmundsson and P. Morin. Planar visibility: testing and counting. *Annual Symposium on Computational Geometry*, 77–86, 2010.
[12] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.
[13] M. Nouri and M. Ghodsi. Space/query-time trade-off for computing the visibility polygon. *Computational Geometry.* 46(3), 371–381, 2013.
[14] M. Nouri Bygi and M. Ghodsi. Weak visibility queries in simple polygons. In *Proc. 23rd Canad. Conf. Comput. Geom.*, 2011.
[15] M. Nouri Bygi and M. Ghodsi. Weak Visibility Queries of Line Segments in Simple Polygons and Polygonal Domains. *CoRR*, abs/1310.7197, 2013.
[16] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proceedings of the Second Annual Symposium on Computational Geometry (SCG 84)*, 14–23, 1984.