An improved Constant-Factor Approximation Algorithm for Planar Visibility Counting Problem

Sharareh Alipour Mohammad Ghodsi Amir Jafari

February 22, 2016

Abstract

Given a set S of n disjoint line segments in $\mathbb{R}^2$, the visibility counting problem (VCP) is to preprocess S such that the number of segments in S visible from any query point p can be computed quickly. This problem can trivially be solved in logarithmic query time using $O(n^4)$ preprocessing time and space. Gudmoonsson and Morin proposed a 2-approximation algorithm for this problem with a tradeoff between the space and the query time. They answer any query in $O_\epsilon(n^{1-\alpha})$ with $O_\epsilon(n^{2+2\alpha})$ of preprocessing time and space, where α is a constant $0 \leq \alpha \leq 1$, $\epsilon > 0$ is another constant that can be made arbitrarily small, and $O_\epsilon(f(n)) = O(f(n)n^\epsilon)$.

In this paper, we propose a randomized approximation algorithm for VCP with a tradeoff between the space and the query time. We will show that for an arbitrary constants $0 \leq \beta \leq \frac{2}{3}$ and $0 < \delta < 1$, the expected preprocessing time, the expected space, and the query time of our algorithm are $O(n^{4-3\beta}\log n)$, $O(n^{4-3\beta})$, and $O(\frac{1}{\delta^3}n^\beta \log n)$, respectively. The algorithm computes the number of visible segments from p, or m_p, exactly if $m_p \leq \frac{1}{\delta^3}n^\beta \log n$. Otherwise, it computes a $(1+\delta)$-approximation m'_p with the probability of at least $1 - \frac{1}{\log n}$, where $m_p \leq m'_p \leq (1+\delta)m_p$.

Keywords. computational geometry, visibility, randomized algorithm, approximation algorithm, graph theory.

1 Introduction

Problem Statement

Let $S = \{s_1, s_2, ..., s_n\}$ be a set of n disjoint closed line segments in the plane contained in a bounding box, $\mathbb{B}$. Two points p and q in the bounding box are visible to each other with respect to S, if the open line segment $\overline{pq}$ does not intersect any segments of S. A segment $s_i \in S$ is also said to be visible from a point p, if there exists a point $q \in s_i$ such that q is visible from p. *The visibility counting problem (VCP)* is to find m_p, the number of segments of S visible from a query point p. We know that *the visibility polygon of a given point* $p \in \mathbb{B}$ is defined as

$$VP_S(p) = \{q \in \mathbb{B} : p \text{ and } q \text{ are visible}\},$$

and *the visibility polygon of a given segment* s_i is defined as

$$VP_S(s_i) = \bigcup_{q \in s_i} VP_S(q).$$

Consider the $2n$ end-points of the segments of S as vertices of a geometric graph. Add a straight-line-edge between each pair of visible vertices. The result is *the visibility graph of* S or $VG(S)$. We can extend each edge of $VG(S)$ in both directions to the points that the edge hits some segments in S or the bounding box. This creates at most two new vertices and two new edges. Adding all these vertices and edges to $VG(S)$ results in a new geometric graph called *the extended visibility graph of* S or $EVG(S)$. $EVG(S)$ reflects all the visibility information from which the visibility polygon of any segment $s_i \in S$ can be computed [11].

Relaterd Work

$VP_S(p)$ can be computed in $O(n \log n)$ time using $O(n)$ space [4, 14]. Vegter proposed an output sensitive algorithm that reports $VP_S(p)$ in $O(|VP_S(p)| \log(\frac{n}{|VP_S(p)|}))$ time, by preprocessing the segments in $O(m \log n)$ time using $O(m)$ space, where $m = O(n^2)$ is the number of edges of $VG(S)$ and $|VP_S(p)|$ is the number of vertices of $VP_S(p)$ [15].

$EVG(S)$ can be used to solve VCP. $EVG(S)$ can optimally be computed in $O(n \log n + m)$ time [9]. If a vertex is assigned to any intersection point of the edges of $EVG(S)$, we have a planar graph, which is called the planar arrangement of the edges of $EVG(S)$. All points in any face of this arrangement have the same number of visible segments and this number can be computed for each face in the preprocessing step [11]. Since there are $O(n^4)$ faces in the planar arrangement of $EVG(S)$, a point location structure of size $O(n^4)$ can answer each query in $O(\log n)$ time. But, $O(n^4)$ preprocessing time and space is high. We also know that for any query point p, by computing $VP_S(p)$, m_p can be computed in $O(n \log n)$ with no preprocessing. This has led to several results with a tradeoff between the preprocessing cost and the query time [3, 5, 10, 13, 16].

Suri and O'Rourke [14] introduced the first approximation algorithm for VCP with the approximation factor of 3. Their algorithm is based on representing a region as the union of a set of triangles. Gudmundsson and Morin [11] improved this result to a 2-approximation algorithm using an improved covering scheme. Their method builds a data structure of size $O_\epsilon(m^{1+\alpha}) = O_\epsilon(n^{2(1+\alpha)})$ in $O_\epsilon(m^{1+\alpha}) = O_\epsilon(n^{2(1+\alpha)})$ preprocessing time, from which each query is answered in $O_\epsilon(m^{(1-\alpha)/2}) = O_\epsilon(n^{1-\alpha})$ time, where $0 < \alpha \leq 1$. This algorithm returns m'_p such that $m_p \leq m'_p \leq 2m_p$. The same result can be achieved from [2] and [12]. In [2], it is proven that the number of visible end-points of the segments in S, denoted by ve_p, is a 2-approximation of m_p, that is $m_p \leq ve_p \leq 2m_p$. There are two other approximation algorithms for VCP by Fischer *et al.* [7, 8]. One of these algorithms uses a data structure of size $O((m/r)^2)$ to build a (r/m)-cutting for $EVG(S)$ by which the queries are answered in $O(\log n)$ time with an absolute error of r compared to the exact answer $(1 \leq r \leq n)$. The second algorithm uses the 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 $1 \leq l \leq n$. In the latter method, the answer of VCP is approximated up to an absolute value of δn for any constant $\delta > 0$ (δ affects the constant factor of both data structure size and the query time).

Our Results

In this paper, we present a randomized $(1+\delta)$-approximation algorithm, where $0 < \delta \leq 1$. The expected preprocessing time and space of our algorithm are $O(m^{2-3\beta/2} \log m)$ and $O(m^{2-3\beta/2})$ respectively, and our query time is $O(\frac{1}{\delta^3} m^{\beta/2} \log m)$, where $0 \leq \beta \leq \frac{2}{3}$ is chosen arbitrarily in the preprocessing time.

In our proposed algorithm, a graph $G(p)$ is associated to each query point p; the construction of $G(p)$ is explained in Section 2. It will be shown that $G(p)$ has a planar embedding and this formula holds: $m_p = n - F(G(p)) + 1$ or $n - F(G(p)) + 2$, where $F(G(p))$ is the number of faces of $G(p)$.

Using Euler's formula for planar graphs, we will show that if p is inside a bounded face of $G(p)$, then $m_p = ve_p - C(G(p)) + 1$, otherwise $m_p = ve_p - C(G(p))$, where $C(G(p))$ is the number of connected components of $G(p)$. In Section 3 and 4, we will present algorithms to approximate ve_p and $C(G(p))$. This leads to an overall approximation for m_p.

Some detail of our algorithm is as follows: First, we try to calculate $VP_S(p)$ by running the algorithm presented in [15] for $\frac{1}{\delta^3} m^{\beta/2} \log m$ steps. If this algorithm terminates, the exact value of m_p is calculated, which is obviously less than $\frac{1}{\delta^3} m^{\beta/2} \log m$. Otherwise, our algorithm instead returns m'_p, such that $m_p \leq m'_p \leq (1+\delta) m_p$ with the probability of at least $1 - \frac{1}{\log n}$. Table 1 compares the performance of our algorithm with the best known result for this problem. Note that if we choose a constant number $0 < \delta < 1$, then our query time is better than [11], however our algorithm returns a $(1+\delta)$-approximation of the answer with a hight probability.

Table 1: Comparison of our method and the best known result for VCP. Note that β $(0 \leq \beta \leq \frac{2}{3})$ is chosen in the preprocessing time and $1+\delta$ $(0 < \delta \leq 1)$ is the approximation factor of the algorithm which affects the query time and $O_\epsilon(f(n)) = O(f(n)n^\epsilon)$, where ϵ is a constant number that can be arbitrary small.

Reference	Preprocessing time	Space	Query	Approx-Factor
[11]	$O_\epsilon(m^{2-3\beta/2})$	$O_\epsilon(m^{2-3\beta/2})$	$O_\epsilon(m^{3\beta/4})$	2
Our result	$O(m^{2-3\beta/2} \log m)$	$O(m^{2-3\beta/2})$	$O(\frac{1}{\delta^3} m^{\beta/2} \log m)$	$1+\delta$

2 Definitions and the main theorem

For each point $a' \in s_i$, let $\overrightarrow{pa'}$ be the ray emanating from the query point p toward a' and let $a = pr(a')$ be the first intersection point of $\overrightarrow{pa'}$ and a segment in S or the bounding box right after touching a'. We say that $a = pr(a')$ is covered by a' or the projection of a' is a. Also, suppose that $\overline{x'y'}$ is a subsegment of s_i and $\overline{xy}$ is a subsegment of s_j, such that $pr(x') = x$ and $pr(y') = y$ and for any point $z' \in \overline{x'y'}$, $pr(z') \in \overline{xy}$, then we say that $\overline{xy}$ is covered by $\overline{x'y'}$.

For each query point p, we construct a graph denoted by $G(p)$ as follows: a vertex v_i is associated to each segment $s_i \in S$, and an edge (v_i, v_j) is put if s_j covers one end-point of s_i (or vice-versa; that is, if s_i covers one end-point of s_j). Obviously, there are two edges between v_i and v_j, if s_j (or s_i) covers both end-points of s_i (or s_j). As an example, refer to Fig 1.(a) and (d). Note that the bounding box is not considered here.

For any segment $s \in S$, let $l(s)$ and $r(s)$ be the first and second end-points of s, respectively swept by a ray around p in clockwise order (Fig 1.(a)).

Lemma 2.1. *$G(p)$ has a planar embedding.*

Proof. Here is the construction. For each end-point $a \in s_i$ not visible from p, let $a' \in s_j$ such that $pr(a') = a$. Draw the straight-line $\overline{aa'}$. Doing this, we have a collection of non-intersecting straight-lines. For each s_i, we put a vertex v_i located very close to the mid-point of s_i. Also, for each segment $\overline{aa'}$, we connect a to v_i and a' to v_j. This creates an edge consisting of three consecutive straight-lines $\overline{v_i a}$, $\overline{aa'}$, and $\overline{a' v_j}$ that connects v_i to v_j. Obviously, none of these edges intersect. Finally, all the original segments are

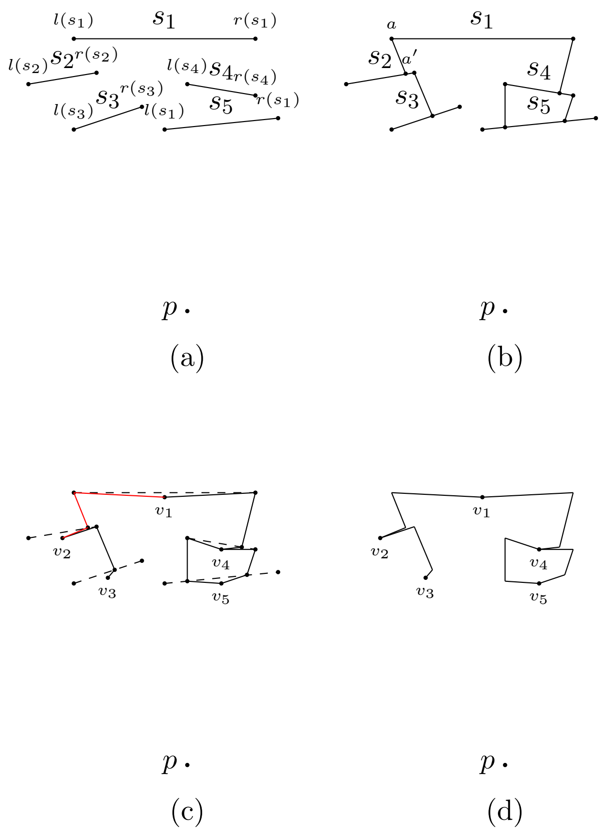

Figure 1: The steps to draw a planar embedding of $G(p)$. (a) The segments are $s_1, \ldots,$ and s_5 with their left and right end-points and a given query point is p. (b) For each end-point $a \in s_i$ not visible to p, if $a' \in s_j$ such that $pr(a') = a$, we draw $\overline{aa'}$. (c) Put a vertex v_i for each segment s_i in a distance sufficiently close to the middle of s_i. For each a and a' (described in (b)), connect a to v_i and a' to v_j. This creats and edge between v_i and v_j shown in red (d) Remove the segments and the remaining is the planar embedding of $G(p)$. Note that the final embedding has 5 vertices and 5 edges and each edge is drown as 3 consequence straight lines.

removed. The remaining is the vertices and edges of a planar embedding of $G(p)$ (These steps can be seen in Fig 1).

∎

From now on, we use $G(p)$ as the planar embedding of the graph $G(p)$. As we know the Euler's formula for any non-connected planar graph G with multiple edges is:

$$V(G) - E(G) + F(G) = 1 + C(G),$$

where $E(G)$, $V(G)$, $F(G)$, and $C(G)$ are the number of edges, vertices, faces, and connected components of G, respectively. The following theorem provides a method to calculate m_p, using $G(p)$.

Theorem 2.1. *The number of segments not visible from p is equal to $F(G(p)) - 2$ if p is inside a bounded face of $G(p)$, or is equal to $F(G(p)) - 1$, otherwise.*

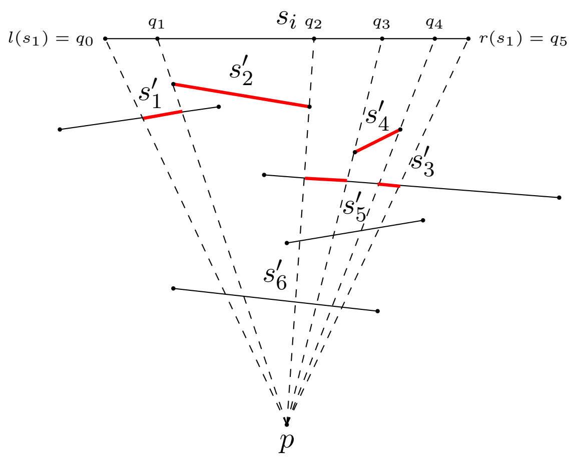

Figure 2: s_i is not visible from p. It can be partitioned into 5 subsegments $\overline{q_0q_1}, \overline{q_1q_2}, \overline{q_2q_3}, \overline{q_3q_4}$, and $\overline{q_4q_5}$, each is covered respectively by subsegment of s'_1, s'_2, s'_3, s'_4, and s'_3 shown above.

Proof. We construct a bijection ϕ between the segments not visible from p to the faces of $G(p)$ except the unbounded face and the face that contains p. This will compelete the proof of our theorem.

Suppose that s_i is a segment not visible from p. Then, we can partition s_i into k subsegments, $\overline{q_0q_1}, \overline{q_1q_2}, \ldots,$ and $\overline{q_{k-1}q_k}$ such that $q_0 = l(s_i)$, $q_k = r(s_i)$, and for each $\overline{q_iq_{i+1}}$, there is a subsegment $\overline{q'_iq'_{i+1}} \in s_j$ that covers $\overline{q_iq_{i+1}}$. Let $s'_1, s'_2, ...,$, and s'_k be the set of segments such that $\overline{xy} \in s'_{i+1}$ covers $\overline{q_iq_{i+1}}$ (note that some segments may appear more than once in the above sequence) (Fig 2). We claim that the vertices $v_i, v'_1, v'_2, ...,$ and v'_k form a bounded face of $G(p)$ that does not contain p. In ϕ, we associate this face to s_i. Since v'_1 is the vertex associated to the first segment that covers $\overline{q_0q_1}$, s'_1 will cover $l(s_i)$ and hence v_i is adjacent to v'_1. Similarly, since s'_k covers $r(s_i)$, hence v_i is adjacent to v'_k. The next subsegment that covers a subsegment of s_i comes from s'_2. This means that $r(s'_1)$ is covered by s'_2 or $l(s'_2)$ is covered by s'_1. This implies that v'_1 is adjacent to v'_2. Similarly, we can show that v'_i is adjacent to v'_{i+1} for all $1 \leq i < k$. To complete the construction, we need to show that the closed path formed by $v_i \to v'_1 \to v'_2, ... \to v'_k \to v_i$ is a bounded face not containing p. Consider a ray around p in clockwise order. The area that this ray touches under s_i and above $s'_1, \ldots,$ and s'_k is a region bounded by $v_i, v'_1, v'_2, ...,$ and v'_k. Obviously, p is not inside this region.

Now, we show that our map ϕ is one-to-one and onto. The proof of one-to-oneness is easier. If $\phi(s_i) = \phi(s_j)$, then according to the construction of ϕ, a subsegment of s_i covers a subsegment of s_j and a subsegment of s_j covers a subsegment of s_i. This is a contradiction since these segments do not intersect. To prove the onto-ness, we need to show for any bounded face f that does not contain p, there is a vertex v_i corresponding to a segment s_i that is not visible to p such that $\phi(s_i) = f$.

To find s_i, we use the sweeping ray around p. Since f is assumed to be bounded and not containing p, the face f is between two rays from p; one from the left and the other from the right. If we start sweeping from left to right, there is a segment corresponding to the vertices of f whose end-point is the first to be covered by the other segments corresponding to the vertices of f. We claim that s_i is the desired segment .i.e. s_i is not visible to p and $\phi(s_i) = f$. For example in Fig 2, the closed path $v_i \rightarrow v'_1 \rightarrow v'_2 \rightarrow v'_3 \rightarrow v'_4, \rightarrow v'_3, \rightarrow v_i$ forms a face and s_i is the first segment among $\{s_i, s'_1, s'_2, s'_3, s'_4\}$ such that $l(s_i)$ is covered by one of the segments in $\{s_i, s'_1, s'_2, s'_3, s'_4\}$.

Obviously, $l(s_i)$ is not visible from p. v'_1 is adjacent to v_i which means that a subsegment of s'_1 covers a subsegment of s_i. Since v'_1 and v'_2 are adjacent, this means that a subsegment of s'_2 consecutively covers the next subsegment of s_i right after s'_1. Continuing this procedure, we conclude that a subsegment of each s'_i covers some subsegment of s_i continuously right after s'_{i-1}. v'_k and v_i are also adjacent, so $r(s_i)$ is not visible from p. We conclude that subsegments of $s'_1, s'_2 \ldots,$ and s_k completely cover s_i and hence s_i is not visible from p.

So, if p is in the unbounded face of $G(p)$, the number of segments which are not visible from p is $F(G(p)) - 1$, otherwise it is $F(G(p)) - 2$. ■

The Euler's formula is used to compute $F(G(p))$. Obviously, $V(G(p))$ is n. For each end-point not visible from p, an edge is added to $G(p)$; therefore, $E(G(p))$ is $2n - ve_p$ (ve_p was defined above as the number of visible end-points from p). The Euler's formula and Theorem 2.1 indicate the following lemma.

Lemma 2.2. *If p is inside a bounded face of $G(p)$, then $m_p = ve_p - C(G(p)) + 1$, otherwise, $m_p = ve_p - C(G(p))$.*

In the rest of this paper, two algorithms are presented; one to approximate ve_p and the other to approximate $C(G(p))$. By applying Lemma 2.2, an approximation value of m_p is calculated. The main result of this paper is thus derived from the following theorem. The proof is given in the Appendix.

Theorem 2.2. *(Main theorem) For any $0 < \delta \leq 1$ and $0 \leq \beta \leq \frac{2}{3}$, VCP can be approximated in $O(\frac{1}{\delta^3} m^{\beta/2} \log m)$ query time using $O(m^{2-3\beta/2} \log m)$ expected preprocessing time and $O(m^{2-3\beta/2})$ expected space. This algorithm returns a value m'_p such that with the probability at least $1 - \frac{1}{\log m}$, $m_p \leq m'_p \leq (1+\delta)m_p$ when $m_p \geq \frac{1}{\delta^3} m^{\beta/2} \log m$ and returns the exact value when $m_p < \frac{1}{\delta^3} m^{\beta/2} \log m$.*

3 An approximation algorithm to compute the number of visible end-points

In this section, we present an algorithm to approximate ve_p, the number of visible end-points. In the preprocessing phase, we build the data structure of the algorithm presented in [15] which calculates $VP_S(p)$ in $O(|VP_S(p)|\log(n/|VP_S(p)|))$ time, where $|VP_S(p)|$ is the number of vertices of $VP_S(p)$. In [15], the algorithm for computing $VP_S(p)$, consists of a rotational sweep of a line around p. During the sweep, the subsegments visible from p along the sweep-line are collected. In the preprocessing phase, we choose a fixed parameter β, where $0 \leq \beta \leq \frac{2}{3}$. In the query time we also choose a fixed parameter $0 < \delta \leq 1$ which is the value of approximation factor of the algorithm.

We use the algorithm presented in [15] to find the visible end-points, but for any query point, we stop the algorithm if more than $\frac{2}{\delta^3}m^{\beta/2}\log m$ of the visible end-points are found.

If the sweep line completely sweeps around p before counting $\frac{1}{\delta^3}m^{\beta/2}\log m$ of the visible end-points, then we have completely computed $VP_S(p)$ and we have $|VP_S(p)| \leq \frac{2}{\delta^3}m^{\beta/2}\log m$. In this case, the number of visible segments can be calculated exactly in $O(\frac{1}{\delta^3}m^{\beta/2}\log m)$ time. Otherwise, $ve_p > \frac{2}{\delta^3}m^{\beta/2}\log m$ and the answer is calculated in the next step of algorithm, that we now explain.

The visibility polygon of an end-point a is a star shaped polygon consisting of $m_a = O(n)$ non-overlapping triangles [4, 14], which are called *the visibility triangles of* a denoted by $VT_S(a)$. Notice that m_a is the number of edges of $EVG(S)$ incident to a. The query point p is visible to an end-point a, if and only if it lies inside one of the visibility triangles of a. Let VT_S be the set of visibility triangles of all the end-points of the segments in S. Then, the number of visible end-points from p is the number of triangles in VT_S containing p. We can construct VT_S in $O(m\log m) = O(n^2\log n)$ time using $EVG(S)$ and $|VT_S| = O(m) = O(n^2)$[11].

We can preprocess a given set of triangles using the following lemma to count the number of triangles containing any query point.

Lemma 3.1. *Let Δ be a set of n triangles. There exists a data structure of size $O(n^2)$, such that in the preprocessing time of $O(n^2\log n)$, the number of triangles containing a query point p can be calculated in $O(\log n)$ time.*

Proof. Consider the planar arrangement of the edges of the triangles in Δ as a planar graph. Let f be a face of this graph. Then, for any pair of points p and q in f, the number of triangles containing p and q are equal. Therefore, we can compute these numbers for each face in a preprocessing phase and then, for any query point locate the face containing that point. There are $O(n^2)$ faces in the planar arrangement of Δ, so a point location structure of size $O(n^2)$ can answer each query in $O(\log n)$ time. Note that the number of triangles containing a query point differs in 1 for any pair of adjacent faces. ■

3.1 The algorithm

Here, we present an algorithm to approximate ve_p. We use this algorithm when $m_p > \frac{1}{\delta^3}m^{\beta/2}\log m$. In the preprocessing phase we take a random subset $RVT_1 \subset VT_S$ such that each member of VT_S is chosen with the probability of $\frac{1}{m^\beta}$.

Lemma 3.2. $E(|RVT_1|) = O(m^{1-\beta})$.

Proof. Let $VT_S = \{\Delta_1, \Delta_2, ...\Delta_{m'}\}$, where $m' = O(m) = O(n^2)$ and $X_i = 1$ if $\Delta_i \in RTV_1$, and $X_i = 0$ otherwise. We have,

$$E(|RVT_1|) = E(\sum_{i=1}^{m'} X_i) = \sum_{i=1}^{m'} E(X_i) = \sum_{i=1}^{m'} \frac{1}{m^\beta} = \frac{m'}{m^\beta} = O(m^{1-\beta}).$$

■

Suppose that in the preprocessing time, we choose $m^{\beta/2}$ independent random subsets $RVT_1, ...,$ and $RVT_{m^{\beta/2}}$ of VT_S. Using Lemma 3.1, for any query point p, the number of triangles of each RVT_i containing p denoted by $(ve_p)_i$, is calculated in $O(\log m)$ time by $O(m^{2-2\beta}\log m)$ expected preprocessing time and $O(m^{2-2\beta})$ expected space. Then, $ve'_p = m^\beta \frac{\sum_{i=1}^{m^{\beta/2}}(ve_p)_i}{m^{\beta/2}}$ is returned as the approximation value of ve_p.

3.2 Analysis of approximation factor

In this section the approximation factor of the algorithm is calculated. Let $X_i = m^\beta(ve_p)_i$.

Lemma 3.3. $E(X_i) = ve_p$

Proof. Suppose that $VT(p) = \{\Delta'_1, \Delta'_2, ..., \Delta'_{ve_p}\} \subset VT_S$ be the set of all triangles containing p. Let $Y_j = 1$ if $\Delta'_j \in RVT_i$, and $Y_j = 0$ otherwise. So, $(ve_p)_i = \sum_{j=1}^{ve_p} Y_j$ and $E((ve_p)_i) = E(\sum_{j=1}^{ve_p} Y_j) = \frac{ve_p}{m^\beta}$. $E(X_i) = E(m^\beta(ve_p)_i) = m^\beta E((ve_p)_i) = m^\beta \frac{ve_p}{m^\beta} = ve_p$. ■

In addition, we can conclude the following lemma:

Lemma 3.4. $E(\frac{\sum_{i=1}^{m^{\beta/2}} X_i}{m^{\beta/2}}) = ve_p$

So, $X_1, X_2, ..., X_{m^{\beta/2}}$ are random variables with $E(X_i) = ve_p$. According to Chebyshev's Lemma the following lemma holds

Lemma 3.5. *(Chebyshev's Lemma) Given $X_1, X_2, ..., X_n$ sequence of i.i.d.'s random variables with finite expected value $E(X_1) = E(X_2) = ... = \mu$, we have,*

$$P((|\frac{X_1+...+X_n}{n} - \mu|) > \varepsilon_1) \leq \frac{Var(X)}{n{\varepsilon_1}^2}.$$

Lemma 3.6. *With a probability at least $1 - \frac{1}{\log m}$ we have,*

$$(1-\delta)ve_p \leq ve'_p \leq (1+\delta)ve_p.$$

Proof. Using Lemma 3.5, we choose $\varepsilon_1 = \delta ve_p$. Here, δ indicates the approximation factor of the algorithm. Obviously, $Var(X_i) = m^{2\beta}(ve_p)(1-\frac{1}{m^\beta})\frac{1}{m^\beta}$. So,

$$\mathbb{P} = P(|ve'_p - ve_p| > \delta ve_p) \leq \frac{m^\beta ve_p}{m^{\beta/2}\delta^2(ve_p)^2}.$$

We know that $ve_p \geq \frac{1}{\delta^2}m^{\beta/2}\log m$, so

$$\mathbb{P} = P(|ve'_p - ve_p| > \delta ve_p) \leq \frac{1}{\log m}.$$

With the probability of at least $1 - \mathbb{P}$, we have,

$$(1-\delta)ve_p \leq ve'_p \leq (1+\delta)ve_p.$$

Also, for a large m, we have $\mathbb{P} \sim 0$. ■

3.3 Analysis of time and space complexity

In the first step of the query time, we run the algorithm of [15]. The preprocessing time and space for constructing the data structure of [15] are $O(m \log m)$ and $O(m)$, respectively, which computes $VP_S(p)$ in $O(|VP_S(p)| \log(n/|VP_S(p)|))$ time. As we run this algorithm for at most $\frac{1}{\delta^3} m^{\beta/2} \log m$ steps, the query time of the first step is $O(\frac{1}{\delta^3} m^{\beta/2} \log m)$.

According to Lemma 3.2, $E(|RVT_i|) = O(m^{1-\beta})$. Using Lemma 3.1, the expected preprocessing time and space for each RVT_i are $O(m^{2-2\beta} \log m)$ and $O(m^{2-2\beta})$ respectively, such that in $O(\log m)$ we can calculate $(ve_p)_i$. So, the expected preprocessing time and space are $m^{\beta/2} O(m^{2-2\beta} \log m) = O(m^{2-\frac{3}{2}\beta} \log m)$ and $m^{\beta/2} O(m^{2-2\beta}) = O(m^{2-\frac{3}{2}\beta})$ respectively.

In the second step, for each RVT_i the value of $(ve_p)_i$ is calculated in $O(\log m)$. Therefore, the query time is $O(\frac{1}{\delta^3} m^{\beta/2} \log m) + O(m^{\beta/2} \log m)$. So, we have the following lemma.

Lemma 3.7. *There exists an algorithm that for any query point p, approximates ve_p in $O(\frac{1}{\delta^3} m^{\beta/2} \log m)$ query time using $O(m^{2-3\beta/2} \log m)$ expected preprocessing time and $O(m^{2-3\beta/2})$ expected space ($0 \leq \beta \leq \frac{2}{3}$). This algorithm returns the exact value of ve_p when $ve_p < \frac{1}{\delta^2} m^{\beta/2} \log m$. Otherwise, a value of ve'_p is returned such that with the probability of at least $1 - \frac{1}{\log m}$, we have $(1-\delta) ve_p \leq ve'_p \leq (1+\delta) ve_p$.*

4 An approximation algorithm for computing the number of components of $G(p)$ and the proof of Theorem 2.2

In this section, we explain an algorithm to compute the number of connected components of $G(p)$, each is simply called a component of $G(p)$.

Let c be a component such that p is not inside any of its faces. It is easy to see that there exists a ray emanating from p that does not intersect any segments corresponding to the vertices of c. We start sweeping this ray in a clockwise direction. Let $l(c)$ (left end-point of c) be the first end-point of a segment of c and $r(c)$ (right end-point of c) be the last end-point of a segment of c that are crossed by this ray. (Fig 3). This way every component c has $l(c)$ and $r(c)$ except the component containing p.

As said, the bounding box is not a part of $G(p)$, but $G(p)$ is contained in the bounding box.

Lemma 4.1. *For each component c, except the one containing p, the projections of $l(c)$ and $r(c)$ both belong either to the same segment or the bounding box.*

Proof. Assume that $pr(l(c))$ belongs to a segment $s \in S$. Since $l(s)$ is on the left of $l(c)$, s can not be among the segments of c. We claim that $r(s)$ is on the right of $r(c)$. Obviously, if this claim is true then, if $pr(r(c)) \in s'$, then $l(s')$ is on the left of $l(c)$. Clearly, if $s \neq s'$, then these two should intersect, which is impossible. Also, this implies that if $pr(l(c))$ is on the bounding box, then $pr(r(c))$ should to be on the bounding box as well. The claim is proven by contradiction. Assume that $r(s)$ is on the left of $r(c)$. Since, $r(s)$ is not visible from p, then there should exist a segment s' that covers $r(s)$. Since, s is not in c and s' is connected to s, s' can not be in c, so $l(s')$ is to the right of $l(c)$ and hence is not visible. Therefore, there should exist a different segment s'' that covers $l(c)$ and with the same argument s'' can not be in c and $l(s'')$ should be covered by another segment. This process can not be continued indefinitely since the number of segments is finite and therefore we will reach a contradiction. ∎

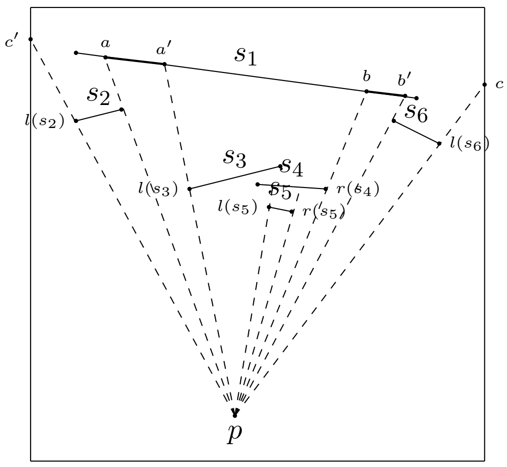

Figure 3: $\overline{aa'}$ and $\overline{bb'}$ are the visible subsegments of s_1. The bounding box has one visible part from c to c'. $G(p)$ has three components; $\{s_1, s_2, s_6\}$, $\{s_3, s_4\}$, and $\{s_5\}$. $l(s_2)$, $l(s_3)$, and $l(s_5)$ are the left end-points of these components, respectively. $r(s_6)$, $r(s_4)$, and $r(s_5)$ are the right end-points of these components, respectively.

Let s'_1, s'_2, s'_3, and s'_4 be the segments of the bounding box. According to Lemma 4.1, we can associate a pair of adjacent visible subsegments or a connected visible part of the bounding box for each component of $G(p)$. For example, in Fig 3, s_1 has two visible subsegments which are associated to the component composed of s_3 and s_4. If we can count the number of visible subsegments of each segment and the number of visible parts of the bounding box, then we can compute the exact value of $C(G(p))$. Because each pair of consecutive visible subsegments of a segment and each visible part of the bounding box are associated to a component. Let c' be the number of visible parts of the bounding box. If $c' > 0$, then p is in the unbounded face. So, if each segment s_i has c_i visible subsegments, then $C(G(p)) = c' + \sum_{i=1}^{n} \max\{(c_i - 1), 0\}$. For example in Fig 3, $c_1 = 2$, $c_2 = 1$, $c_3 = 1$, $c_4 = 2$, $c_5 = 1$ and $c_6 = 1$, also $c' = 1$. This implied that $C(G(p)) = 3$. If $c' = 0$, then p is in a bounded face and this face is contained in a component with no left and right end-point, so in this case $C(G(p)) = 1 + \sum_{i=1}^{n} \max\{(c_i - 1), 0\}$.

In the following we propose an algorithm to approximate the number of visible subsegments of each segment $s_i \in S \cup \{s'_1, s'_2, s'_3, s'_4\}$.

4.1 Algorithm

According to [11], it is possible to cover the visibility region of each segment $s_i \in S \cup \{s'_1, s'_2, s'_3, s'_4\}$ with $O(m_{s_i})$ triangles denoted by $VT(s_i)$. Here, $|VT(s_i)| = O(m_{s_i})$, where m_{s_i} is the number of edges of $EVG(S)$ incident on s_i. Note that the visibility triangles of s_i may overlap. If we consider the visibility triangles of all segments, then there is a set $VT_S = \{\Delta_1, \Delta_2, ...\}$ of $|VT_S| = O(m)$ triangles. We say Δ_i is related to s_j if and only if $\Delta_i \in VT(s_j)$. For a given query point p, m''_p, the number of triangles in VT_S containing p, is between m_p and $2m_p$. So, m''_p gives a 2-approximation factor solution for VCP [11]. Since the visibility triangles of each segment may overlap, some of the segments are counted repeatedly. In [11], it is shown that each segment s_i is counted c_i times, where c_i is the number of visible subsegments of s_i. In other words, there are c_i triangles related to s_i in VT_S which contain p.

A similar approach can be used to approximate $C(G(p))$. A random subset $RVT_1 \subset VT_S$ is chosen such that each member of VT_S is chosen with probability $\frac{1}{m^\beta}$. For a given query point p, let $c'_{i,1} \geq 1$ be the number of triangles related to s_i in RVT_1 containing p. We report $C_1 = \sum_{i=1}^{n}(m^{\beta}c'_{i,1} - 1)$ as the approximated value of $C(G(p))$ received by RVT_1. We choose $m^{\beta/2}$ random subsets $RVT_1, ..., RVT_{m^{\beta/2}}$ of VT_S. Let p be the given query point, for each RVT_j, $C_j = \sum_{i=1}^{n}(m^{\beta}c'_{i,j} - 1)$ is calculated. At last, $C'_p = \frac{\sum_{j=1}^{m^{\beta/2}} C_j}{m^{\beta/2}}$ is reported as the approximation value of $C(G(p))$.

4.2 Analysis of approximation factor

We show that with the probability at least $\frac{1}{\log m}$, if $C(G(p)) > \frac{1}{\delta^2}m^{\beta/2}\log m$, then C'_p is a $(1+\delta)$-approximation of $C(G(p))$.

Lemma 4.2. $E(C_j) = C(G(p))$.

Proof. $E(C_j) = E(\sum_{i=1}^{n} m^{\beta}c'_{i,j} - 1) = \sum_{i=1}^{n} E(m^{\beta}c'_{i,j} - 1) = \sum_{i=1}^{n} c_i - 1 = C(G(p))$. ■

Using Lemma 3.5, we have,

$$\mathbb{P} = P(|\frac{C_1 + ... + C_{m^{\beta/2}}}{m^{\beta/2}} - C(G(p))| > \delta C(G(p))) \leq \frac{Var(C_i)}{m^{\beta/2}\delta^2 C(G(p))^2}.$$

$Var(C_i) = m^{2\beta}C(G(p))(\frac{1}{m^{\beta}})(1 - \frac{1}{m^{\beta}})$. Since we have, $C(G(p)) > \frac{1}{\delta^2}m^{\beta/2}\log m$,

$$\mathbb{P} = P(|\frac{C_1 + ... + C_{m^{\beta/2}}}{m^{\beta/2}} - C(G(p))| > \delta C(G(p))) \leq \frac{1}{\log m}.$$

So, with the probability at least $1 - \mathbb{P}$,

$$(1-\delta)C(G(p)) \leq C'_p \leq (1+\delta)C(G(p)).$$

and for a large m, we have, $\mathbb{P} \sim 0$.

4.3 Analysis of time and space complexity

By Lemma 3.1, for each RVT_i, a data structure of expected preprocessing time and size of $O(m^{2-2\beta}\log m)$ and $O(m^{2-2\beta})$ is needed. RVT_i returns C_i in $O(\log m)$ for each query point p. So, the expected space for all $m^{\beta/2}$ data structures is $O(m^{2-2\beta+\beta/2}\log m)$ and the query time for calculating C'_p is $O(m^{\beta/2}\log m)$. So, we have the following lemma.

Lemma 4.3. *There exists an algorithm that approximates $C(G(p))$ in $O(\frac{1}{\delta^2}m^{\beta/2}\log m)$ query time by using $O(m^{2-3\beta/2})$ expected preprocessing time and $O(m^{2-3\beta/2})$ expected space $(0 \leq \beta \leq \frac{2}{3})$. For each query p, this algorithm returns a value C'_p such that with probability at least $1 - \frac{1}{\log m}$, $(1-\delta)C(G(p)) \leq C'_p \leq (1+\delta)C(G(p))$ when $C(G(p)) > \frac{1}{\delta^2}m^{\beta/2}\log m$.*

5 Conclusion

In this paper, a randomized algorithm is proposed to compute an approximation answer to VCP. The main ideas of the algorithm that reduce the complexity of previous methods are random sampling and breaking the query into two steps. The time and space complexity of our algorithm depend on the size of $EVG(S)$. A planar graph is associated to each query point p. It is proven that the answer is equal to $ve_p - C(G(p))$, where ve_p is the number of visible end-points and $C(G(p))$ is the number of connected components in the planar graph. To improve the running time of our algorithm instead of finding the exact values of ve_p and $C(G(p))$, we approximate these values. Although an exact calculation of ve_p using a tradeoff between the query time and the space is possible.

References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, Advances in Discrete and Computational Geometry, volume 223 of Contemporary Mathematics, pages 156. American Mathematical Society Press, 1999.

[2] Alipour, S., Zarei, A.: Visibility Testing and Counting. FAW-AAIM 2011, Jinhua, China, LNCS (Volume 6681) by Springer-Verlag, 343-351 (2011)

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

[4] Asano, T.: An efficient algorithm for finding the visibility polygon for a polygonal region with holes. IEICE Transactions. 557–589(1985)

[5] Bose, P., Lubiw, A. and Munro, J. I.: Eficient visibility queries in simple polygons. Computational Geometry Theory and Applications. 23(7), 313–335(2002)

[6] Bondy, J. A., Murty, U. S. R.: Graph theory with applications (Vol. 290). London: Macmillan (1976)

[7] Fischer, M., Hilbig, M., Jahn, C., Meyer auf der Heide F. and Ziegler M.: Planar visibility counting. CoRR, abs/0810.0052. (2008)

[8] Fischer, M., Hilbig, M., Jahn, C., Meyer auf der Heide F. and Ziegler M.: Planar visibility counting. In Proceedings of the 25th European Workshop on Computational Geometry(EuroCG 2009).203–206(2009)

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

[10] Ghosh, S. K.: Visibility algorithms in the plane. Cambridge university press. (2007)

[11] Gudmundsson, J., Morin, P.: Planar visibility: testing and counting. Annual Symposium on Computational Geometry. 77–86 (2010)

[12] Nouri, M. and Ghodsi, M.: Space/query-time tradeoff for computing the visibility polygon. Computational Geometry. 46(3), 371–381 (2013)

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

[14] Suri, S. and O'Rourke, J.: 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)

[15] Vegter, G.: The visibility diagram: A data structure for visibility problems and motion planning. In: Gilbert, J.R., Karlsson, R. (eds.) SWAT(1990). LNCS, 447, pp. 97–110. Springer, Heidelberg (1990)

[16] Zarei, A. and Ghodsi, M.: Efficient computation of query point visibility in polygons with holes. In Proceedings of the 21st Annual ACM Symposium on Computational Geometry. (SCG 2005). (2005).

Appendix

Proof of Theorem 2.2

Assume that we have the data structures of [15] and the algorithms of Lemma 3.7 and Lemma 4.3. For a given query point p, we first run the algorithm of [15] for $\frac{2}{\delta^3}m^{\beta/2}\log m$ steps. If $VP_S(p)$ is calculated, then the exact value of m_p is computed in $O(\frac{1}{\delta^3}m^{\beta/2}\log m)$. Otherwise, we calculate ve'_p and C'_p in $O(\frac{1}{\delta^2}m^{\beta/2}\log m)$ time by Lemma 3.7 and Lemma 4.3. In the second case, we have $m_p > \frac{2}{\delta^3}m^{\beta/2}\log m$. As $m_p \leq 2ve_p$, so $ve_p > \frac{1}{\delta^3}m^{\beta/2}$. Lemma 3.7 also implies $|ve'_p - ve_p| \leq \delta ve_p$.

In the following it is shown that if $C'_p < (1+\delta)\frac{1}{\delta^2}m^{\beta/2}\log m$, then with probability at least $1 - \frac{1}{\log m}$, we have $C(G(p)) < \frac{1+\delta}{1-\delta}\frac{1}{\delta^2}m^{\beta/2}\log m$.

By Lemma 3.5 we have,

$$p(|C(G(p) - C'_p| > \delta C(G(p)) \leq \frac{m^{\beta/2}}{\delta^2 C(G(p)}$$
$$p(C(G(p)) > \frac{C'_p}{1-\delta}) \leq \frac{m^{\beta/2}}{\delta^2 C(G(p))}.$$

Which means, if $C(G(p)) > \frac{1}{\delta^2}m^{\beta/2}\log m$, then with probability at most $\frac{1}{\log m}$ we have, $C(G(p)) > \frac{C'_p}{1-\delta}$. So, with a probability at least $1 - \frac{1}{\log m}$ we have

$$C(G(p)) \leq \frac{C'_p}{1-\delta} \leq \frac{1+\delta}{1-\delta}\frac{1}{\delta^2}m^{\beta/2}\log m.$$

Since, $C(G(p)) \leq m_p$ and $m_p > \frac{1}{\delta^3}m^{\beta/2}\log m$, we have, $C(G(p)) \leq \frac{1+\delta}{1-\delta}\delta m_p$. We know that $ve_p = m_p + C(G(p))$, and with probability at least $1 - \frac{1}{\log m}$ we have, $(1-\delta)ve_p \leq ve'_p \leq (1+\delta)ve_p$, thus

$$\frac{ve'_p}{1+\delta} \leq m_p + C(G(p)) \leq m_p + \frac{1+\delta}{1-\delta}\delta m_p.$$

Which implies

$$m_p \leq \frac{ve'_p}{1-\delta} \leq \frac{1+\delta^2(1+\delta)}{(1-\delta)^2}m_p.$$

Let $1 + \delta^* = \frac{1+\delta^2(1+\delta)}{(1-\delta)^2}$, then

$$m_p \leq \frac{ve'_p}{1-\delta} \leq (1+\delta^*)m_p.$$

So, $\frac{ve'_p}{1-\delta}$ is reported as the approximated value of m_p.

If $C'_p > (1+\delta)\frac{1}{\delta^2}m^{\beta/2}\log m$, then according to Lemma 3.5, we have,

$$P(|C'_p - C(G(p)| \geq \frac{1}{\delta}m^{\beta/2}\log m) \leq \frac{m^{\beta/2}C(G(p))}{\frac{1}{\delta^2}m^{\beta}\log^2 m}.$$

So, $P(C(G(p)) < C'_p - \frac{1}{\delta}m^{\beta/2}\log m) \leq \frac{m^{\beta/2}C(G(p))}{\frac{1}{\delta^2}m^{\beta}\log^2 m}$. We know that $C'_p > (1+\delta)\frac{1}{\delta^2}m^{\beta/2}\log m$. So, if $C(G(p)) \leq \frac{1}{\delta^2}m^{\beta/2}\log m$, then $P(C(G(p)) < \frac{1}{\delta^2}m^{\beta/2}\log m) \leq \frac{1}{\log m}$. We conclude that with probability at least $1 - \frac{1}{\log m}$, $C(G(p)) > \frac{1}{\delta^2}m^{\beta/2}\log m$, and so we can use Lemma 4.3.

$$\begin{aligned} m_p = ve_p - C(G(p)) &\leq \frac{ve'_p}{1-\delta} - \frac{C'_p}{1+\delta} \\ &\leq \frac{(1+\delta)ve_p}{1-\delta} - \frac{(1-\delta)C(G(p))}{1+\delta} \\ &\leq ve_p - C(G(p)) + \frac{2(\delta)ve_p}{1-\delta} + \frac{2\delta C(G(p))}{1+\delta}. \end{aligned}$$

We know that $C(G(p)) \leq m_p$. Moreover, we have, $m_p \leq 2ve_p$, so

$$\begin{aligned} &\leq m_p + \tfrac{4\delta m_p}{1-\delta} + \tfrac{2\delta m_p}{1+\delta} \\ &\leq (1 + \tfrac{4\delta}{1-\delta} + \tfrac{2\delta}{1+\delta})m_p. \end{aligned}$$

Let $\delta^* = \frac{4\delta}{1-\delta} + \frac{2\delta}{1+\delta}$, then

$$m_p \leq \tfrac{ve'_p}{1-\delta} - \tfrac{C'_p}{1+\delta} \leq (1 + \delta^*)m_p.$$

Therefore, $\frac{ve'_p}{1-\delta} - \frac{C'_p}{1+\delta}$ is reported as the approximated value of m_p. Note that $\delta^* < 1$ and it can be arbitrary small by choosing δ small enough.

The query time of our algorithm is $O(\frac{1}{\delta^3} m^{\beta/2} \log m)$, where the dependence of the variable δ^* to δ is as follows. When δ is less than a fixed constant C, δ^* is at most a linear fixed multiple of δ and hence, the query time of the algorithm can be expressed as $O(\frac{1}{\delta^3} m^{\beta/2} \log m)$. Note that for $\delta > C$ since $\delta^{-3} < C^{-3}$ it will be absorbed in the constant hidden in $O(m^{\beta/2} \log m)$.