Visibility testing and counting

Sharareh Alipour

Sharif University of Technology

e-mail: shalipour@ce.sharif.edu

Mohammad Ghodsi

Sharif University of Technology

Institute for Research in Fundamental Sciences (IPM)

e-mail: Ghodsi@sharif.edu

Alireza Zarei

Sharif University of Technology

e-mail: zarei@sharif.ir

Maryam Pourreza

Sharif University of Technology

e-mail: pourreza@ce.sharif.ir

Abstract

For a set of n disjoint line segments S in $\mathbb{R}^2$, the visibility testing problem (VTP) is to test whether the query point p sees a query segment $s \in S$. For this configuration, the visibility counting problem (VCP) is to preprocess S such that the number of visible segments in S from any query point p can be computed quickly. In this paper, we solve VTP in expected logarithmic query time using quadratic preprocessing time and space. Moreover, we propose a $(1+\delta)$-approximation algorithm for VCP using at most quadratic preprocessing time and space. The query time of this method is $O_\epsilon(\frac{1}{\delta^2}\sqrt{n})$ where $O_\epsilon(f(n)) = O(f(n)n^\epsilon)$ and $\epsilon > 0$ is an arbitrary constant number.

Keywords: Computational geometry, Visibility, Randomized algorithm, Approximation algorithm

1. Introduction

Statement of the problem. Suppose that we have a bounding box which covers a set S containing n disjoint segments. Two points $p, q \in \mathbb{R}^2$ are visible from each other with respect to (w.r.t.) S, if there exists no segment $s \in S$ intersecting line segment $\overline{pq}$. We say that a segment $\overline{st} \in S$ is visible (w.r.t. S) from a point p, if a point $q \in \overline{st}$ can be found from which p is visible. Two problems are considered here: *the visibility testing problem (VTP)* in which we decide whether or not a given

[☆] Fully documented templates are available in the elsarticle package on CTAN.

query segment $s \in S$ is visible from a query point p, and *the visibility counting problem (VCP)* for which we count the number of segments $s \in S$ which are visible to a given query point p. Scrutinizing the visibility polygon of the query point can be helpful for answering the mentioned problems. The visibility polygon of a given point $p \in \mathbb{R}^2$ is defined as

$$VP_S(p) = \{q \in \mathbb{R}^2 : p \text{ and } q \text{ are visible (w.r.t. } S)\}.$$

and the visibility polygon of a given segment $\overline{st}$ is defined as

$$VP_S(\overline{st}) = \bigcup_{q \in \overline{st}} VP_S(q) = \{p \in \mathbb{R}^2 : \overline{st} \text{ and } p \text{ are visible (w.r.t. } S)\}.$$

Consider the $2n$ end-points of the segments of S as vertices of a geometric graph. Add an edge between each pair of visible vertices such that the edge is a straight line. The result is *the visibility graph of* S or $VG(S)$. We can extend each edge of $VG(S)$ in both directions until it intersects the segments in S (or the bounding box). Each edge in $VG(S)$ creates at most two new vertices and some new edges. Adding all these vertices and edges to $VG(S)$ results in a new graph called *the extended visibility graph* or $EVG(S)$.

Related Works. The optimal running time to compute $VP_S(p)$ is $O(n \log n)$ that uses $O(n)$ space [1, 2]. [3] has also presented an output sensitive algorithm in which preprocessing steps are in $O(n^2)$ time and $O(n^2)$ space and as a result, $VP_S(p)$ is computed in $O(|VP_S(p)| \log(\frac{n}{|VP_S(p)|}))$ time where $|VP_S(p)|$ is the number of vertices in $VP_S(p)$.

Gudmundsson and Morin considered a version of VTP where a segment $s \in S$ is chosen randomly in the preprocessing time and the goal is to test whether any given query point p can see s. Suppose that m_s is the number of edges of $EVG(S)$ incident on s and $m_s \leq k \leq m_s^2$. They gave an algorithm with preprocessing time and space of $O(k)$ that answers each query in $O_\epsilon(\frac{m_s}{\sqrt{k}})$ [4]. Also, the presented algorithm in [5, 6] answers each query in $O(\log n)$ time by using $O(m_s^2/l)$, where $l \geq 1$ is a space/time tradeoff parameter of the data structure.

The version of VTP where s is fixed and chosen in the preprocessing time and VCP can be solved using $EVG(S)$. There is an optimal $O(n \log n + m)$ time algorithm to compute $VG(S)$ [7], where $m = O(n^2)$ is the number of the edges of $VG(S)$. This algorithm can be used to compute $EVG(S)$ in $O(n \log n + m)$ time as well [7]. Considering the planar arrangement of the edges of $EVG(S)$ as a planar graph, all points in any face of this arrangement have the same set of visible segments. The number of visible segments can be computed for each face in the preprocessing step. 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. As can be seen, the space and the time used in the preprocessing step is high. However, without any preprocessing, the query can be answered by computing the visibility polygon of query point in $O(n \log n)$ time which is also high. There are also some other results with

a trade-off between the query time and the preprocessing cost [8, 9, 10, 11, 12] (a complete survey is presented in the visibility book of [13]).

The primary concern in VCP is to propose an approximation algorithm with acceptable approximation factor which reduces the preprocessing cost. The first approximation algorithm for VCP was proposed by [2]. They represented the visiblity polygon of each segment by using the union of a set of triangular convex regions. The approximation factor for the resulting algorithm was 3. With an improved covering scheme, Gudmundsson and Morin presented a 2-approximation algorithm [4]. Let $0 < \alpha \leq 1$, using a data structure of size $O_\epsilon(m^{1+\alpha})$ and a preprocessing time of $O_\epsilon(m^{1+\alpha})$, this algorithm can answer each query in $O_\epsilon(m^{(1-\alpha)/2})$ time. If we show the number of visible segments from p by m_p, m'_p is also returned by this algorithm such that $m_p \leq m'_p \leq 2m_p$. [14] and [15] also provide the same result.

Throughout this paper, $\epsilon > 0$ is an arbitrary small constant number and $O_\epsilon(f(n)) = O(f(n)n^\epsilon)$.

Two other approximation algorithms have also been introduced for VCP by Fischer *et.al.* [5, 6]. In the first algorithm, a (r/m)-cutting, $1 \leq r \leq n$, for $EVG(S)$ (see [4]) is built using a data structure of size $O((m/r)^2)$. With this cutting, the queries are answered in $O(\log n)$ time and an absolute error of r compared to the exact answer. In the second algorithm, a random sampling method is used to build a data structure of size $O((m^2 \log^{O(1)} n)/l)$, $1 \leq l \leq n$, and any query is answered in $O(l \log^{O(1)} n)$ time. Note that, for any constant $\delta > 0$, we can have an approximation up to an absolute value of δn for the VCP. Moreover, δ can have effects on the constant factor of both the data structure size and the query time.

Our Results. In this paper, we first propose an algorithm for VTP which answers each query in an expected $O(\log n)$ time. This algorithm uses $O(n^2.\alpha(n))$ preprocessing time where $\alpha(n)$ is a pseudo-inverse of Ackermann's function and $O(n^2)$ space. We also present a $(1+\delta)$-approximation algorithm for VCP using a randomized approach. This algorithm uses $O(n^2.\alpha(n))$ preprocessing time and $O(n^2)$ space and answers each query in $O_\epsilon(\frac{1}{\delta^2}\sqrt{n})$ time. The experimental results demonstrate the efficiency of the algorithm.

In the 2nd section of this paper, we introduce our algorithm for VTP. In the 3rd section, the algorithm for VCP is proposed. In Section 4, we present our experimental results and finally the last section contains a summary of the results and the conclusion.

2. Visibility Testing Problem

In this section, we propose an algorithm to solve VTP. The preliminary version of this result appeared in [14]. Let $r_{\vec{d}}(p)$ denote a ray emanating from p in the direction of $\vec{d}$. Also, assuming that $s', s \in S$, by $pr_p(a') = a$ we mean that $r_{\overrightarrow{pa}}(p)$ intersects $a \in s$ right after it intersects $a' \in s'$ (see

Fig 1)

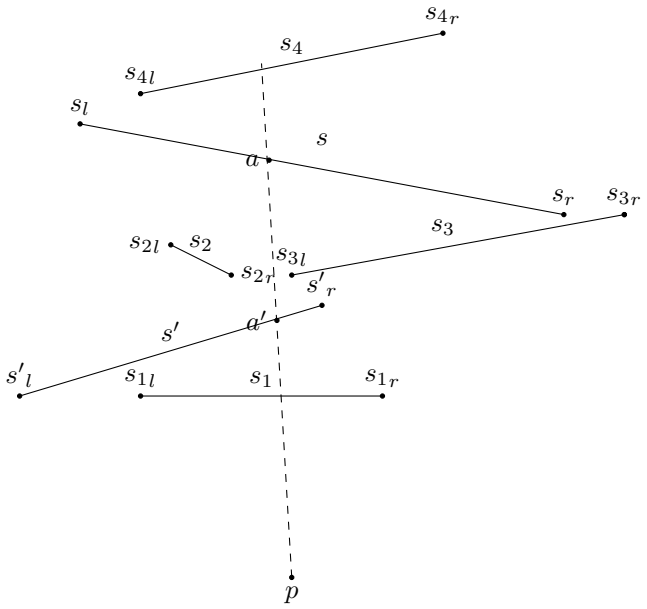

Figure 1: $pr_p(a) = a'$ and for any point $q \in s$ there is a point $q' \in s_i$, such that $s_i \in \{s', s_2, s_3\}$ and $pr_p(q') = q$.

Assume that $s \in S$ and a point p are the query inputs. If s is not visible from p, then there must exist a subset of $S - s$ such that for any point $a \in s$, there is an $a' \in s'$ such that $pr_p(a') = a$ for $s' \in S - s$, see Fig 1. So, we start from the left end-point of s (denoted as s_l) and move towards the right end-point, or s_r. (The left and right end-points of any segment are defined according to their order of radial sweeping around p), see Fig 1. If s_l is visible from p, then s and p are visible. Otherwise, assume that for a point $q' \in s' \in S - s$, we have $pr_p(q') = s_l$. If s'_r appears after s_r in the radial sweep around p, then s is not visible from p (see Fig 2(a)). Otherwise, let t be the intersection point of s and $\overrightarrow{ps'_r}(p)$. We conclude that the subsegment $\overline{s_l t}$ of s is not visible from p. So, we should only check the part of s that lies to the right of t. So, we continue on $\overline{ts_r}$. This is done by checking whether s'_r and p are visible. If yes other and $pr_p(s'_r) = t \in s$, we conclude that p is visible from a point $t' \in \overline{ts_r}$ with an arbitrarily small distance to t. If so, p and s are visible (see Fig 2(b)). Otherwise, assume that $s'' \neq s'$ is the last segment intersected by $\overrightarrow{ps'_r}(p)$ before intersecting s. So, we continue on s'' in the same way as done for s' (see Fig 2(c) and (d)). This algorithm is formally shown in Algorithm 1. The algorithm returns true if and only if there is a point on s that is visible to p. This is the answer of the VTP and implies its correctness.

2.1. Analysis of the running time and space

For a segment s, the running time of this algorithm depends on the number of steps in which the algorithm processes a subset S'_s of S. Let $c_s(p)$ be a subset of S such that for each $s' \in c_s(p)$ there is

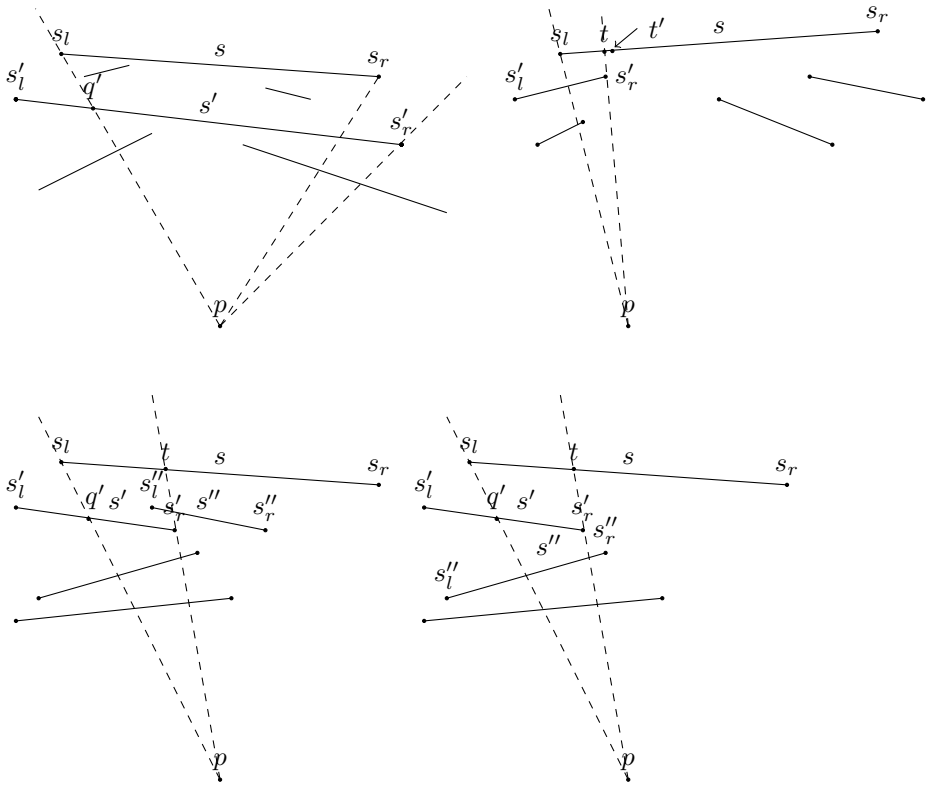

Figure 2: The algorithm for VTP begins from the left end-point to the right end-point. (a) s'_r swept after s_r, which means it is the end of algorithm. (b) p sees some part of s right after $pr_p(s'_r)$. (c) and (d) s'' is the last segment intersected by $\underset{ps_r}{r\longrightarrow}(p)$ before intersecting s.

Algorithm 1 The Visibility Testing Algorithm

1: *Input*:
2: S: a set of segments and the data structure used for ray shooting
3: $s \in S$: a given segment
4: p: a query point
5: *Output*:
6: *true*: if s is visible from p
7: *false*: otherwise
8: **if** p and s_l are visible from each other **then**
9: **return** true
10: **else**
11: let s' be the segment such that $pr_p(q') = s_l$ for a point $q' \in s'$.
12: **while** *true* **do**
13: **if** s_r is swept before s'_r while a radial sweep line around p is moving from s_l towards s_r **then**
14: **return** false
15: **else**
16: **if** s' is the only segment intersected by $\overrightarrow{ps'_r}$ before intersecting s **then**
17: **return** true
18: **else**
19: let s'' be the last segment intersected by $\overrightarrow{ps'_r}$ before intersecting s (except s' itself). Let $s' = s''$
20: **end if**
21: **end if**
22: **end while**
23: **end if**

at least one point $a' \in s'$ and $pr_p(a') = a \in s$. So, $S'_s \subset c_s(p)$ (see Fig 3). Other than S'_s, the running time of this algorithm also depends on the way we find the segment s' in lines 11 and 19 of Algorithm 1. This can be answered by using the data structures for solving the ray-shooting problem. Assume that for each end-point s'_r, we can answer this query in $f(n)$ time. Then, the total time complexity of this algorithm is $O(|c_s(p)|f(n))$ where $|.|$ is the size of corresponding set. Theorem 2.1 implies that by preprocessing the segments in $O_\epsilon(n^2)$ time and space, we have $f(n) = O(\log n)$.

Theorem 2.1. *[? / Given] a set of n disjoint line segments S, we can preprocess the segments using $O(n^2\alpha(n))$ time, where $\alpha(n)$ is a pseudo-inverse of Ackermann's function, and $O(n^2)$ space such that for a given query point p and a direction $\vec{d}$, the first segment in S intersected by $r_{\vec{d}}(p)$ is determined in $O(\log n)$ time.*

Note that to find the last segment before s which is crossed by $r_{\overrightarrow{ps'_r}}(p)$, we first find the intersection point of s and r and then shoot a ray from that point towards p. The first segment intersected by this ray is the last segment before s intersected by $r_{\overrightarrow{ps'_r}}(p)$.

Lemma 2.1. *If we choose $s \in S$ randomly, then the expected size of $c_s(p)$ is $O(1)$.*

Proof. We partition the segments of $c_s(p)$ into 2 subsets. For each segment $s' \in c_s(p)$, if $pr_p(s'_r) \in s$ or $pr_p(s'_l) \in s$ then we put it in $e(c_s(p))$ and else we put s' in $m(c_s(p))$. Trivially, $e(c_s(p))$ and $m(c_s(p))$

are disjoint and $c_s(p) = e(c_s(p)) \bigcup m(c_s(p))$ (see Fig 3). We prove that $|m(c_s(p))| \leq |e(c_s(p))|+1$. Sort the segments in $c_s(p)$ regarding to the order of their projection on s from s_l to s_r. Assume that s_1 and s_2 are two adjacent segments. Trivially, at least one of these segments is in $e(c_s(p))$ or in other word two segments of $m(c_s(p))$ can not be adjacent. This means that $|m(c_s(p))| \leq |e(c_s(p))| + 1$. For each segment s_i, we compute $c_{s_i}(p)$. Obviously, for any $s_i \in S$, $|m(c_{s_i}(p))| \leq |e(c_{s_i}(p))| + 1$. It is simple to see that for a given point p, each segment t can be a member of $e(c_{s_i}(p))$ for at most two segments of S. Then, $\sum_{s\in S} |c_s(p)| = \sum_{s\in S} |e(c_s(p))| + \sum_{s\in S} |m(c_s(p))| \leq 2\sum_{s\in S} |e(c_s(p))| + 1 = O(n)$. Therefore, the expected size of $c_s(p)$ of a randomly chosen segment $s \in S$ is $\frac{O(n)}{n} = O(1)$. □

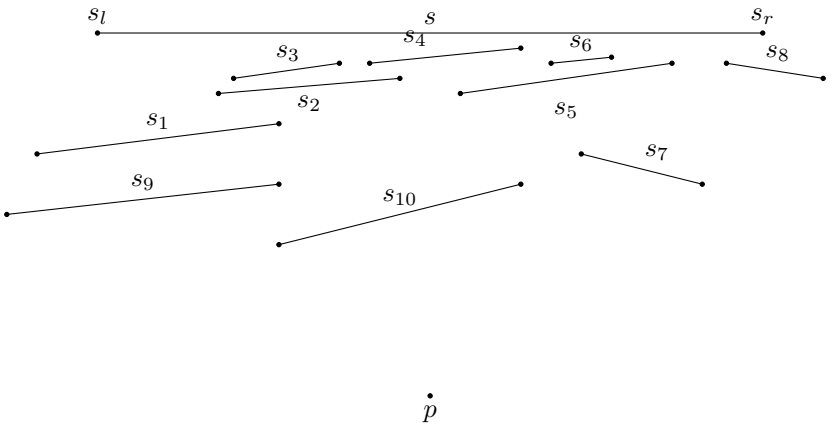

Figure 3: s is not visible from p and for any point $q \in s$ there is a point q' in $c_s(p) = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$ such that $pr_p(q') = q$ for which $m(c_s(p)) = \{s_1, s_7\}$ and $e(c_s(p)) = \{s_2, s_3, s_4, s_5, s_6, s_8\}$. In this example s_2 is not visible to p either and $m(c_{s_2}(p)) = \{s_{10}\}$ and $e(c_{s_2}(p)) = \{s_1\}$. Another segment which is not visible to p is s_1 and $m(c_{s_1}(p)) = \{s_9\}$ and $e(c_{s_1}(p)) = \{s_{10}\}$. For this configuration of segments we have: $e(c_{s_9}(p)) = \{s_{10}\}$. In this example if we run the VTP algorithm on s and p, the order of segments in S'_s is: s_1, s_3, s_2 ,s_5 and s_7. Obviously $S'_s \subset c_s(p)$.

Corollary 2.1. *VTP can be answered in $O(\log n)$ expected query time using $O(n^2.\alpha(n))$ preprocessing time and $O(n^2)$ space.*

3. Visibility Counting Problem

In this section, we propose an algorithm to approximate the answer of VCP. The algorithm is composed of two phases. In the first phase, we run the algorithm of [3] to compute $VP_S(p)$ for $O_\epsilon(\sqrt{n})$ times. In [3], the algorithm for computing the visibility polygon, consists of a rotational sweep of a line about the query point. During the sweep, the subsegments visible from p along the sweep-line are collected. We sweep the line until at most $\sqrt{n}$ of the visible subsegments are counted. So, if before counting $\sqrt{n}$ visible segments, the line completely sweeps around p, then we have $VP_S(p)$ and $|VP_S(p)| \leq \sqrt{n}$. If $VP_S(p)$ is calculated in this phase, then we have the exact value of m_p from $VP_S(p)$.

Otherwise, $|VP_S(p)| > \sqrt{n}$ and we choose a random subset $RS_1 \subset S$ such that each segment is chosen with the probability of $\frac{1}{\sqrt{n}}$. Then, for each segment $s_i \in RS_1$, we check whether s_i is visible to p with respect to S, using the proposed algorithm of VTP in the previous section. Let X_1 be the number of visible segments from p in RS_1 with respect to S. We run this for t random subset $RS_i \subset S$. Let X_i be the number of visible segments from p in RS_i with respect to S. We report $m'_p = \frac{\sum_{i=1}^{t} \sqrt{n} X_i}{t}$ as the approximated value of m_p.

Lemma 3.1. *(Chebyshev's Inequality) Let X_1, X_2,...,X_t be any random variable with $E(X_i) = \mu$, and let $\varepsilon > 0$ be any positive real number. Then*

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

Using Lemma 3.1 we have

Lemma 3.2. *With a probability close to 1 we have*

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

Proof. In Lemma 3.1, we choose $\varepsilon = \delta m_p$, where δ is a constant number which can be arbitrarily small and $t = \frac{1}{\delta^2}$. Obviously, $E(\sqrt{n}X_i) = m_p$ and $Var(\sqrt{n}X_i) = n(m_p)(1 - \frac{1}{\sqrt{n}})\frac{1}{\sqrt{n}}$. Therefore, with the probability

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

Since $m_p > O_\epsilon(\sqrt{n})$, then $\mathbb{P} \sim 0$. This means that with probability of at least $1 - \mathbb{P}$, we have

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

□

3.1. Analysis

In the first phase of the algorithm, we run the method of [3] for $O_\epsilon(\sqrt{n})$ times. In the second phase according to Corollary 2.1, in the expected time of $O(\log n)$ we can check whether p and a random selected segment are visible to each other with respect to S. Since we chose $O(\sqrt{n})$ random segments for each RS_i, the query time is $O(t\sqrt{n}\log n)$. Therefore, considering both phases, the query time, preprocessing time and space are $O_\epsilon(t\sqrt{n})$, $O(n^2.\alpha(n))$ and $O(n^2)$, respectively. Consequently, we conclude the result of this section in the following theorem.

Theorem 3.1. *VCP can be approximated in expected $O_\epsilon(\frac{1}{\delta^2}\sqrt{n})$ time using $O(n^2)$ space and $O(n^2.\alpha(n))$ preprocessing time. This algorithm returns a value m'_p such that with probability of arbitrarily close to 1, $(1-\delta)m_p \leq m'_p \leq (1+\delta)m_p$ when $m_p > \sqrt{n}$ and returns the exact value when $m_p \leq \sqrt{n}$.*

According to [4], by space and preprocessing time of $O_\epsilon(m)$, VCP for each query is answered in $O_\epsilon(\sqrt{m})$, where m is the number of edges in $EVG(S)$. However, in our method, with $O_\epsilon(n^2)$ space and preprocessing time, the query time is $O_\epsilon(t\sqrt{n})$. Note that we choose δ a constant number, so the query time of this algorithm is $O_\epsilon(\sqrt{n})$. Furthermore, the approximation factor of [4] is 2, while ours is $1 + \delta$.

4. Experimental results

In this section, we present our experimental results. First of all for VTP, we generate random segments for different values of n and also a random point p. Then we calculate $c_{s_i}(p)$ and $average(n) = \sum_{i=1}^{n} c_{s_i}(p)/n$ for every set of segments for some specified values of n. Fig 4 reveals our results for different values of n. It is worth noting that for each experimented n, we computed $average(n)$ with different random points and we achieved the same result. This result indicates that the expected number of segments that cover a randomly chosen segment is independent of the size of n, the location of p and the structure of other segments. Fig 5 and 6 show calculated $average(n)$ for different locations of p and different values of n.

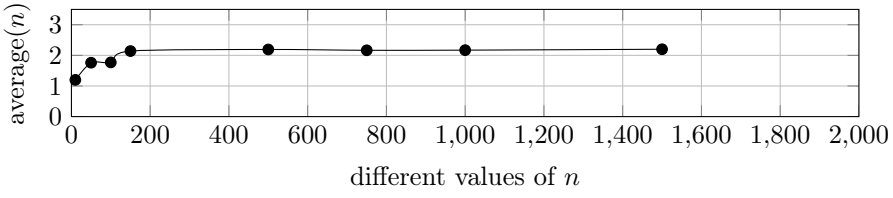

Figure 4: Average(n) for different values of n

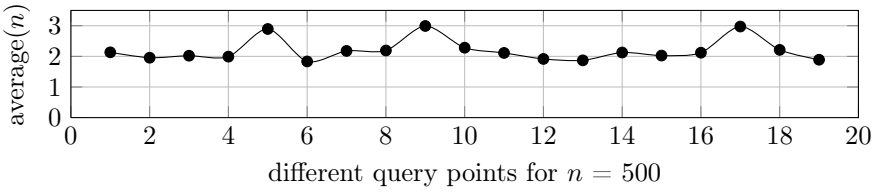

Figure 5: Average(n) for different locations of p and n=500

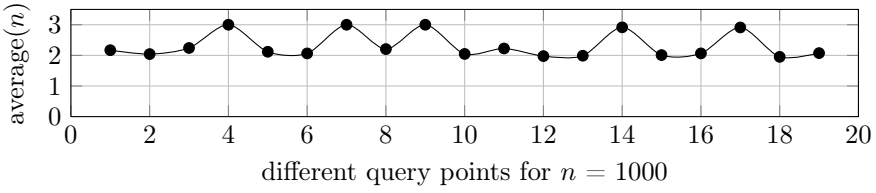

Figure 6: Average(n) for different locations of p and n=1000

For VCP, we generate a new random set of segments and also a random query point for each n. Using different values of δ and t, we calculate m_p and m'_p. Our experimental results demonstrate that when $m_p > \sqrt{n}$, our result is a $1+\delta$ approximation of the main result. Table 1 illustrates this conclusion.

n	δ	$t = \frac{1}{\delta^2}$	m_p	m'_p
10000	1	1	5971	5700
10000	1	1	2985	3200
10000	1	1	7021	6200
10000	1	1	3973	3900
10000	0.5	4	4964	5700
10000	0.5	4	5029	4600
10000	0.5	4	4986	4400
10000	0.5	4	5071	6150
10000	0.25	16	4998	4675
10000	0.25	16	4982	5175
10000	0.25	16	4905	5275
10000	0.25	16	5036	5100
100000	1	1	39641	42690
100000	1	1	80070	70202
100000	1	1	19859	17392
100000	1	1	50021	48382
100000	0.5	4	49882	51070
100000	0.5	4	49940	50754
100000	0.5	4	50001	56604
100000	0.5	4	49990	46169
100000	0.25	16	49977	49647
100000	0.25	16	50202	46326
100000	0.25	16	50256	51702
100000	0.25	16	49816	45773
1000000	1	1	399439	409000
1000000	1	1	600475	572000
1000000	1	1	699566	712000
1000000	1	1	900418	901000
1000000	0.5	4	499632	509000
1000000	0.5	4	499833	491000
1000000	0.5	4	499444	499000
1000000	0.5	4	500249	501500
1000000	0.25	16	500395	512250
1000000	0.25	16	499721	487000
1000000	0.25	16	500044	497250
1000000	0.25	16	500273	499250

Table 1: The values of m_p and m'_p for various values of δ and t.

5. Conclusion

In this paper, we propose an algorithm to answer VTP. We prove that the expected time to answer each query is $O(\log n)$ for any query point p and a randomly chosen segment $s \in S$. This means that the expected time for answering VTP of a point and a segment is equal to the time required for two query points. Using the result of this algorithm, a randomized algorithm is proposed to approximate the answer of VCP. The idea of this algorithm is to choose a random sample of segments and extend

the answer of this sample to the whole input. This algorithm improves the time complexity of the previous methods. Our experimental data and the implementation prove efficiency of our method. Precisely, the experimental results show that the average running time obtained in theory is the actual running time in practice.

References

[1] T. Asano, An efficient algorithm for finding the visibility polygon for a polygonal region with holes, IEICE TRANSACTIONS (1976-1990) 68 (9) (1985) 557–589.

[2] S. Suri, J. O'Rourke, Worst-case optimal algorithms for constructing visibility polygons with holes, in: Proceedings of the Second Annual Symposium on Computational Geometry, SCG '86, ACM, New York, NY, USA, 1986, pp. 14–23. doi:10.1145/10515.10517.
URL http://doi.acm.org/10.1145/10515.10517

[3] G. Vegter, The visibility diagram: a data structure for visibility problems and motion planning., in: J. R. Gilbert, R. G. Karlsson (Eds.), SWAT, Vol. 447 of Lecture Notes in Computer Science, Springer, 1990, pp. 97–110.
URL http://dblp.uni-trier.de/db/conf/swat/swat90.html#Vegter90

[4] J. Gudmundsson, P. Morin, Planar visibility: testing and counting., in: J. Snoeyink, M. de Berg, J. S. B. Mitchell, G. Rot, M. Teillaud (Eds.), Symposium on Computational Geometry, ACM, 2010, pp. 77–86.
URL http://dblp.uni-trier.de/db/conf/compgeom/compgeom2010.html#GudmundssonM10

[5] M. Fischer, M. Hilbig, C. Jähn, F. M. auf der Heide, M. Ziegler, Planar visibility counting, CoRR abs/0810.0052.
URL http://dblp.uni-trier.de/db/journals/corr/corr0810.html#abs-0810-0052

[6] M. Fischer, M. Hilbig, C. Jähn, F. Meyer auf der Heide, M. Ziegler, Planar visibility counting, in: Proc. 25th European Workshop on Computational Geometry, 2009, pp. 203–206.

[7] S. K. Ghosh, D. M. Mount, An output-sensitive algorithm for computing visibility, SIAM J. Comput. 20 (5) (1991) 888–910. doi:10.1137/0220055.
URL http://dx.doi.org/10.1137/0220055

[8] B. Aronov, L. J. Guibas, M. Teichmann, L. Z. 0001, Visibility queries and maintenance in simple polygons., Discrete & Computational Geometry 27 (4) (2002) 461–483.
URL http://dblp.uni-trier.de/db/journals/dcg/dcg27.html#AronovGTZ02

[9] P. Bose, A. Lubiw, J. I. Munro, Efficient visibility queries in simple polygons, Comput. Geom. Theory Appl. 23 (3) (2002) 313–335. doi:10.1016/S0925-7721(01)00070-0.
URL http://dx.doi.org/10.1016/S0925-7721(01)00070-0

[10] M. Pocchiola, G. Vegter, The visibility complex., Int. J. Comput. Geometry Appl. 6 (3) (1996) 279–308.
URL http://dblp.uni-trier.de/db/journals/ijcga/ijcga6.html#PocchiolaV96

[11] A. Zarei, M. Ghodsi, Efficient computation of query point visibility in polygons with holes., in: J. S. B. Mitchell, G. Rote (Eds.), Symposium on Computational Geometry, ACM, 2005, pp. 314–320.
URL http://dblp.uni-trier.de/db/conf/compgeom/compgeom2005.html#ZareiG05

[12] D. Chen, H. Wang, Visibility and ray shooting queries in polygonal domains, in: F. Dehne, R. Solis-Oba, J.-R. Sack (Eds.), Algorithms and Data Structures, Vol. 8037 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 244–255. doi:10.1007/978-3-642-40104-6_22.
URL http://dx.doi.org/10.1007/978-3-642-40104-6_22

[13] S. Ghosh, Visibility Algorithms in the Plane, Cambridge University Press, New York, NY, USA, 2007.

[14] S. Alipour, A. Zarei, Visibility testing and counting, in: Proceedings of the 5th Joint International Frontiers in Algorithmics, and 7th International Conference on Algorithmic Aspects in Information and Management, FAW-AAIM'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 343–351.
URL http://dl.acm.org/citation.cfm?id=2021911.2021949

[15] M. N. Baygi, M. Ghodsi, Space/query-time tradeoff for computing the visibility polygon., Comput. Geom. 46 (3) (2013) 371–381.
URL http://dblp.uni-trier.de/db/journals/comgeo/comgeo46.html#BaygiG13