Determining the Visible Sites Inside a Simple Polygon *

Cao An Wang
Department of Computer Science
Memorial University of Newfoundland
St.John's, Newfoundland, Canada A1C 5S7

Abstract

Let a set of n points (called sites) lie inside a simple m-sided polygon. In this paper, the problem of reporting the sites visible from a query point is considered. A geometric object is used for answering this query in $O(log(mn^2)+n)$ time. This geometric object may contain worst-case $\Omega(mn^2)$ distinct regions w.r.t. the visibility of these sites, and these regions can be found in $\Theta(mn^2)$ time and space when $cm \leq n$ for any constant c. Moreover, a geometric object (called constrained Voronoi diagram) is used for determining the closest and visible site of a query point in $O(log(mn^2+n^3))$ time. The diagram can be found in $O(mn^2+n^3)$ time and space.

1 Introduction

The notion of visibility underlies many applications, such as image processing, computer graphics, robotic control etc. The visibility polygon problem is also important in computational geometry. The visibility polygon from a single illuminator (a point or an edge) inside or outside a simple polygon has been extensively investigated [1,2,5,6,10,11]. As long as the visibility polygon is given, whether or not the illuminator is visible from a query point can be answered quickly. A natural extension of the above query is to consider multi-illuminators. That is, given a set S of sites, regarded as illuminators in a simple polygon, to find the subset of S visible from a query point in the polygon. Note that each site of S determines a *visible polygon*. The boundaries of these visible polygons divide the simple polygon into a collection of distinct *visible regions* (which will be defined). As long as these regions are identified and are represented by a data structure for the point location problem, the preceding query can be answered efficiently. Furthermore, if reporting the closest and visible site of a query point is considered, then the *constrained Voronoi diagram* of S in polygon [8,9,12] (that is, the Voronoi diagram of S restricted by the boundary of the polygon) can be used to answer this query quickly.

Our algorithms are simple and easy to be implemented. Potential applications exist in the fields previously mentioned. In Section 2, we prove the lower bound for the number of visible regions for a set of sites in a simple polygon and present an optimal algorithm to construct these visible regions. In Section 3, we define 'Constrained Voronoi diagram' and proposes an algorithm for finding the diagram. This diagram can be used for finding the closest visible site of a query point efficiently, and it can also be used to find 'the nearest visible neighboring sites' of the sites and to find 'all nearest neighbors' of the sites efficiently. In Section 4, we conclude our work. For simplicity in the rest of the paper, S stands for a set of n sites and P for the boundary (point set) of a simple m-sided polygon. P also denotes the edge set.

*This work is supported by NSERC grant OPG0041629

2 Finding the visible regions of a simple polygon

Definition: Let $\overline{xy}$ be the line segment spanning points x and y. Points x and y are *visible* from each other in the presence of P iff the points of $\overline{xy}$ do not lie on both sides of P. $\overline{xy}$ is said to *cross* P if x and y are not visible from each other.

Definition: Let P be divided into regions (or called *cells*) by some straight lines. Then, regions $r_1, ...,$ and r_k are called the *neighboring* regions of r inside P if each of these region shares some edges on the boundary of r. A region r is called a *visible region* if the interior of r is visible from some subset S' of S and the interior of each of the neighboring regions $r_1, ...,$ and r_k of r is visible from a subset of S different from S'.

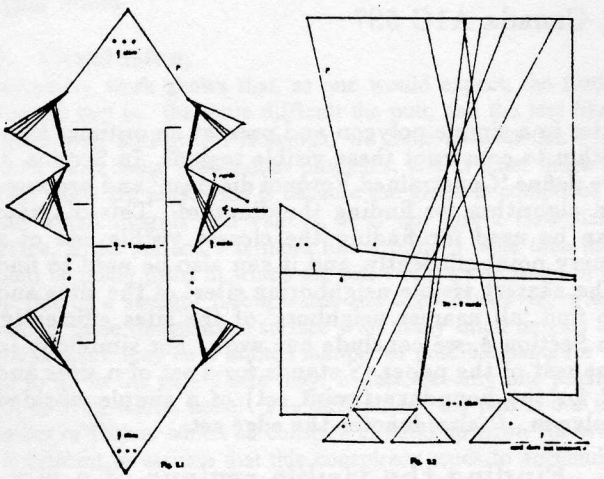
Definition: Given S and P, let $\vec{sp}$ be a ray emitting at a site $s \epsilon S$ and passing through a vertex p of P such that p is visible from s. Let t be the first intersection point of $(\vec{sp} - \overline{sp})$ and P. Then, the line segment $c(=\overline{pt})$, not containing s, is called a *cord* (*associated with* s). A visible region r is *associated with* a cord c if the boundary of r contains an edge of c and the associated site of c is visible to the interior of r.

2.1 Lower bound of the number of visible regions of sites in a polygon

Lemma 2.1: The number of visible regions of S inside P is $\Omega(mn^2)$ in the worst case.

Proof: We consider a construction for S and P as shown in Fig. 2.1, where n sites are divided into two groups with roughly equal size, and group 1 is put on the bottom of P and group 2, on the top of P. Then, each cord associated with a site in group

1 intersects $\frac{n}{2}$ cords associated with sites in group 2. There are $\frac{m}{2}-2$ such cords associated with each site in group 1 and there are $\frac{n}{2}$ sites in group 1, hence all the cords associated with the sites in group 1 and those in group 2 will generate $O(mn^2)$ intersections. These intersections determine $O(mn^2)$ regions such that the interior of a region r is visible from a subset S' of S and that of each of its neighboring regions is visible from a subset different from S'. □



2.2 The Algorithm

Lemma 2.2 Assume that no two cords lie on the same straight line. Any cord $c \epsilon C$ intersects at most $2n-2$ other cords in C, where C is the set of cords determined by S and P.

Proof: Suppose that c is associated with a site $s \epsilon S$. Then, c does not intersect the cords associated with s (denoted by C_s) implied by the definition of a cord. Moreover, c intersects at most two of the cords associated with any other site $s' \epsilon (S-\{s\})$ (denoted by $C_{s'}$). This is because if s' lies inside the visible polygon w.r.t. s (denoted by V_s), then all the cords $C_{s'}$ lies outside V_s and c cannot cross any one of $C_{s'}$. If s' lies outside V_s. Note that each cord of $C_{s'}$ and a piece of the boundary of P form a closed polygon. Then, c must start at a vertex outside $V_{s'}$ (that is, inside a closed polygon w.r.t. a cord of $C_{s'}$). Then, c either terminates at the boundary of P or crosses the cord and enters $V_{s'}$. Then, c either terminates at the boundary of P or crosses another cord of $C_{s'}$ and goes out $V_{s'}$ (that is, c enters the corresponding polygon of the cord). Then, c must terminate at the boundary of P. The lemma follows. □

Lemma 2.3 Let $A(C)$ be the arrangement of kn cords for $1 \leq k \leq \frac{m}{2}$ and the m edges of P. Let c be any element of C. A cell of $A(C)$ intersecting c is called *active* cell and the edges of the active cell, *active* edges. Then, the maximum number of active edges w.r.t. c, denoted by N_1, is bounded by $O(m+n)$.

Proof: (Refer to Fig. 2.2.) We shall prove the lemma using the *Zone* theorem presented in [3]. Let the cords be arranged as shown in the figure where cord c crosses exactly two cords per site for every site of $S-\{s\}$, where s is the site associated with c. Then, c crosses at most $2n-2$ cords by Lemma 2.2. These cords cross pairwise above c, except those pairs of cords, each of which is associated with the same site. If we extend the cords of each of those pairs and let them cross at the site, then the arrangement of these extended cords inside P is equivalent to an arrangement of $2n-2$ straight lines in the plane in terms of their intersections. By the *Zone* theorem [Theorem 2.7, p.346, 3], we find that the maximum number of active edges of these $2n-1$ cords above c is less than $5(2n-2)-1$ $(=10n-11)$. Similarly, the number of those active edges below c is also less than $10n-11$. Moreover, the number of active edges of those cords which do not intersect c is bounded by the number of edges of P. This is because each such cord contains exactly one active edge and the cord 'blocks' at least one edge of P from the active cells. Thus, the number of such active edges plus the number of the edges of P are at most m. Finally, c or each of these $2n-1$ cords may end at the interior of an active edge of P and divides the edge into two parts, hence these cords may create at most $4n-2$ active subedges of P. We obtain a bound (not tight) $N_1 < m+24n-24 = O(m+n)$. (A tight bound is $m+17n-18$. We omit the proof here because it is tedious and the non-tight bound serves for our algorithm analysis.) □

Lemma 2.4 Let r and r' be two neighboring visible regions sharing a cord c. Then, the difference between the visible sites of r and those of r' is the site associated with c.

Proof: Implied by the definition of cord and that of visible region. □

Algorithm Vis-Region(S,P)
Input: S and P.
Output: a collection of visible regions.
Method:

Step 1. For each site $s \epsilon S$, find these vertices in P visible from s, denoted by P_s. (* This can be done by any linear-time algorithm for finding visible polygon from a point inside P [5, p.313,10]. *)

Step 2. (* Find arrangement $A(C)$, where $A(C)$ is represented by a DCEL [10] *)
(a) $A(C) \leftarrow P$.
(b) While $S \neq \emptyset$ Do
  Pick an $s \epsilon S$
  While $P_s \neq \emptyset$ Do
    (i) Pick a $p \epsilon P_s$. Draw ray $\vec{sp}$.
    (ii) Starting at vertex p, identify the cell r of $A(C)$ such that $\vec{sp}$ crosses r and shares p with r; *walk* along the boundary of r to find the next intersection point with $\vec{sp}$, say x. Update $A(C)$ by inserting $\overline{px}$ and splitting r into two new cells r' and r'' along $\overline{px}$.
    (iii) $p \leftarrow x$. Repeat (ii) until $x \epsilon P$.
  EndDo
EndDo

Lemma 2.5 Algorithm Vis-Region(S,P) finds the visible regions in $\Theta(mn^2)$ time and space where $cm \leq n$ for any constant c.

Proof: The interiors of two neighboring cells of $A(C)$ are visible from different subsets of S implied by Lemma 2.4. The interior of a cell of $A(C)$ is visible from the same subset of S because only elements of C and P can be the boundary of the lights of the sites. Then, each cell of $A(C)$ is a visible region by definition. Moreover, $A(C)$ covers the entire region inside P, thus $A(C)$ must determine all the visible regions w.r.t. S and P. Thus, Step 2 of Vis-Region(S,P) produces all the visible regions w.r.t. S and P because it produces $A(C)$. Let us consider the time and space complexities of Vis-Region(S,P). Step 1 takes $O(mn)$ time and space by [5]. Step 2 takes $O((m+n)mn)$ time and space implied by Lemma 2.3 and that is $O(mn^2)$ when $cm \leq n$. It is $\Theta(mn^2)$ by Lemma 2.1. □

Lemma 2.6 Given S and P, the sites visible from a query point can be reported in $O(log(mn^2)+n)$ time, and the data structure for the query can be found in $O(mn^2 log(mn^2))$ preprocessing time and $O(mn^2)$ space.

Proof: $A(C)$ is presented by a data structure for point location by one of the methods of [4,7]. Then, a query point q can be located in a visible region, say r. Let $c = \overline{pt}$ be a cord associated with r (that is, c contains an edge e of r and the associated site s of c is visible to the interior of r). Now, traverse c on $A(C)$ starting at vertex p with its visible site list S_p, where S_p can be obtained in an extra linear-time during the execution of Step 1. Suppose that c crosses a cord associated with site s' and let e' and e'' be the two edges separated by the cord. Then, delete s' from S_p if e' is visible from s' or add s' to S_p otherwise. The new S_p is associated with e''. Continue this traversal until e is reached. The resulting S_p associated with e is the set of the sites visible from q. The correctness of this answer is obvious. For the time complexity of such a query, note that c can be identified in constant time, since we can associate c with r during the execution of Step 2 of Vis-Region in constant time. c crosses at most $2n-2$ cords by Lemma 2.2 and each such crossing causes an update of exactly one site in S_p. Thus, the traversal contains at most $2n-2$ updates. Each update takes constant time. Then, the query can be answered in $O(log(mn^2)+n)$ time, where the first term is due to locating q. Clearly, the data structure takes $O((mn^2)log(mn^2))$ preprocessing time and $O((mn^2)$ space [4,7]. □

Remark: If we store the corresponding list of visible sites for each cell in Step 2 of Vis-Region(S,P), then the query can be answered in $O(log(mn^2)+k)$ time, where $k \leq n$ is the number of sites to be reported, and the space increases by a factor of n.

Remark: The condition $cm \leq n$ can be relaxed to $cm \leq 2^n$ as follows. (1) use ray-shooting method [6] to find all the cords determined by S and P. (2) sort these cords according to their vertices of P along P, for the cords sharing the same vertex, sort them according to their second endpoints along P, and denote the sequence by C_p. (3) insert C_p into $A(C)$ by Step 2(b)(ii)-(iii) of Vis-Region by walking only the active edges of inserted cords. To avoid walking on the active edges of P, for each vertex of P we maintain a pair of endpoints of the currently inserted cords which are the two nearest to this vertex along P. Clearly, (1) takes $O(mnlog(m))$ time [6]. (2) takes $O(mnlog(n))$ time due to the sortings. (3) takes $O(mn^2)$ by Lemma 2.3. But, this method is expensive in practice.

3 Finding constrained Voronoi diagram of sites in a polygon

Definition: The *visible* distance between $s \epsilon S$ and an arbitrary point x is determined by

$$d_p(x,s) = \begin{cases} d(x,s) & \text{if } x \text{ is visible to } s \\ \infty & \text{otherwise} \end{cases}$$

Definition: The *constrained* Voronoi diagram, denoted by $CVor(S,P)$, is a set of *Voronoi cells* (V-cells) $\{V(s_i) \mid s_i \epsilon S\}$ such that $V(s_i) = \{x \epsilon R^2 \mid d_p(x,s_i) < d_p(x,s_j)$ and $d_p(x,s_i) \neq \infty$ for all $s_j \epsilon S$, $s_i \neq s_j\}$. The *boundary* of a V-cell $V(s_i)$ is the closure of $V(s_i)$. A *Voronoi edge* (V-edge) is a maximal straight line segment on the boundary of a V-cell. The *Voronoi vertices* are the endpoints of V-edges.

Constrained Voronoi diagrams have some special properties which do not exist in the other types of Voronoi diagrams. These properties follow directly from the definition of the diagram.

Property 1: A V-edge must be one of the following three types: (1) a section of an edge of P, (2) a segment of the perpendicular bisector of two sites, and (3) a segment of a cord.

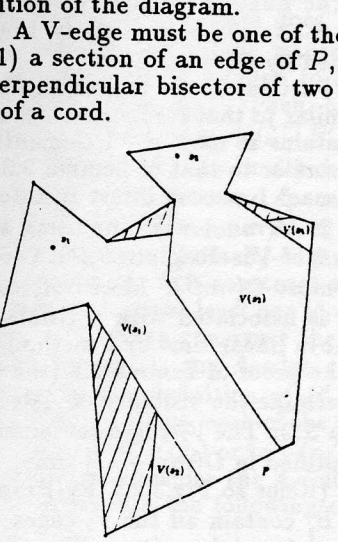


Fig. 3.1

Property 2: For simplicity, we say that an edge *is* a V-edge if it is a V-edge or it is a segment of a V-edge. Let the 'closeness' of sites to an edge be measured by the shortest distances between the sites to a fixed interior point of the edge. An edge e is a type-1 V-edge (denoted by V_1-edge) of V-cell $V(s_i)$ iff e lies on P and s_i is the closest and visible site of e among the sites in S; e is a type-2 V-edge (V_2-edge) iff the two sites, whose perpendicular bisector contains e, are the closest and visible sites of e among the sites in S; e is a type-3 V-edge (V_3-edge) iff the site, which together with a vertex determine the cord containing e, is the closest and visible site of e among the sites in S.

Property 3: A V-cell may consist of several disjoint Voronoi sub-cells, and each of those sub-cells is bounded by some V-edges of the above three types. In Fig. 3.1, each of V(s_1) and V(s_2) consists of two disjoint Voronoi sub-cells.

Property 4: A constrained Voronoi diagram may not cover the entire polygon. That is, some areas of the polygon may not belong to any V-cell due to the blockage of the boundary of the polygon. In Fig. 3.1, each of the shaded areas does not belong to any one of $V(s_1)$ and $V(s_2)$.

3.1 The algorithm

Let $b'_{i,j}$ be the segment of the perpendicular bisector $b_{i,j}$ determined by sites s_i and s_j such that $b'_{i,j}$ is visible from s_i and s_j. Hence, $b'_{i,j}$ is a connecting segment of $b_{i,j}$. $b'_{i,j}$ can be determined in O(m) time since the visible polygon of s_i and that of s_j can be determined in O(m) time [5]. Let B'_i denote the set of $b'_{i,k}$'s for all $s_k \epsilon (S - \{s_i\})$. Let $A(C \cup B'_i)$ be the arrangement of C, the edges and the cords of P, and B'_i. Call the edges in $A(C \cup B'_i)$, $A - edges$ which consist of three types (denoted by A_1-edge, A_2-edge, and A_3-edge) corresponding to those of V-edges.

Lemma 3.1: (a) Any element of B'_i intersects at most $2n$ A_3-edges and at most $n-2$ A_2-edges in $A(C \cup B'_i)$. (b) The number of edges on the *active* cells of $A(C \cup B'_i)$ w.r.t. any $b \epsilon B'_i$ (that is, the cells intersect b) is bounded by O($m+n$).

Proof: Part (a) of the lemma can be proved by a method similar to that for Lemma 2.2 and by the fact that B'_i contains at most $n-1$ elements. Part (b), by a method similar to that of Lemma 2.3. □

By Lemma 3.1, we can insert B'_i into $A(C)$ to form $A(C \cup B'_i)$ in O($mn+n^2$) time using a method similar to Step 2 of **Vis-Region(S,P)**. Clearly, the space used is at most O(mn^2). Moreover, each A_2-edge in $A(C \cup B'_i)$ is associated with a visible-site list. This can be done in linear-time by a method similar to that stated in the proof of Lemma 2.6 (the elements of B'_i do not determine the visibility of A_2-edges).

Lemma 3.2: The V_2-edges contained by $A(C \cup B'_i)$ can be identified in O($mn+n^2$) time.

Proof: (Refer to Fig.3.2.) By Property 1, the elements of B'_i contain all the V_2-edges with s_i as one of the two determining sites. We only consider one element of B'_i, say $b'_{i,j}$ and show how to identify the V-edges contained by $b'_{i,j}$. Let e be an edge of $b'_{i,j}$. We divide the visible-site list of e, denoted by VS_e, into two sublists: VS^l_e and VS^r_e, where VS^l_e contains s_i and those sites s_k's such that e and s_k lie on the same closed halfplane determined by $b_{i,k}$. VS^r_e contains the remaining sites in VS_e. Property 2 states that if s_i and s_j are the two closest and visible sites of $e \epsilon b_{i,j}$ among S, then e is a V_2-edge. Then, the fact that VS^l_e of e contains only s_i implies that e is a V_2-edge of $V(s_i)$. To determine VS^l_e and VS^r_e of every edge e of $b'_{i,j}$, we first determine $VS^l_{e_1}$ and $VS^r_{e_1}$ of an extreme edge e_1 of $b'_{i,j}$ by a brute-force method. That is, test e_1 against all the halfplanes determined by s_i and the sites in VS_{e_1}. Then, we traverse $b'_{i,j}$ starting at e_1 to determine the two sublists for each of the remaining edges. Let e and e' be two consecutive edges of $b'_{i,j}$ and assume that VS^l_e and VS^r_e are known. Note that e and e' can be separated by either a bisector $b'_{i,k}$ of B'_i or a cord c_k. In the former case, s_k is moved from VS^l_e (resp. from VS^r_e) to VS^r_e (resp. to VS^l_e) if s_k is in VS^l_e (resp. in VS^r_e). In the latter case, s_k is added to (resp. deleted from) VS^r_e or VS^l_e accordingly if s_k is (resp. is not) visible from e'. Then, the two resulting sublists are obviously these of e'. Note that $VS^l_{e_1}$ and $VS^r_{e_1}$ can be found in O(n) time, updating the two sublists for next edge takes constant time, and there are at most O($m+n$) A-edges in $b'_{i,j}$ by Lemma 3.1. Thus, the two sublists for all the A-edges of $b'_{i,j}$ can be determined in O($m+n$) time. The lemma follows since B'_i contains $n-1$ elements. □

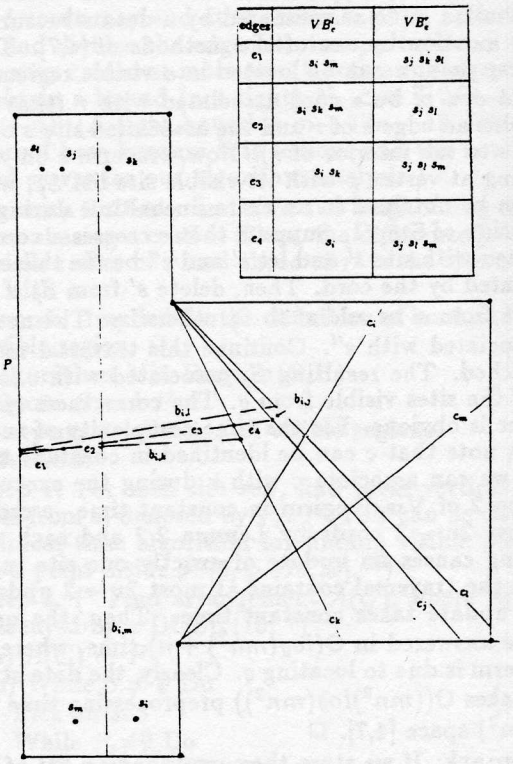
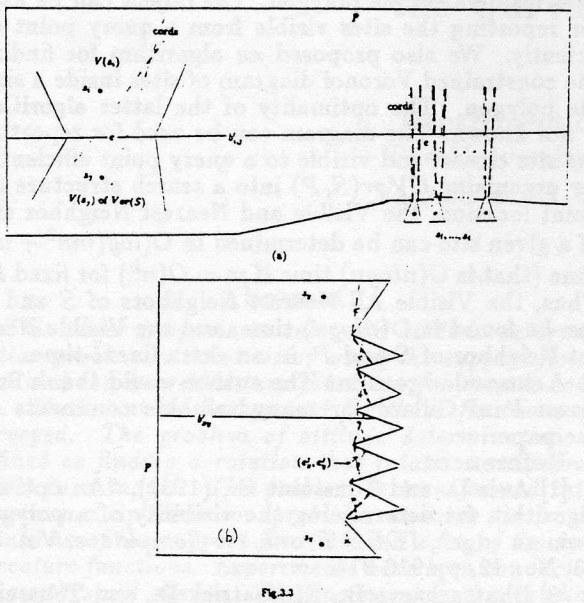


Fig. 3.2

By Lemma 3.2, we can find all those V_2-edges of $CVor(S,P)$ one site at a time. Note that the V-edges in an A-cell and the boundary of this A-cell are connected because a V-edge must end at either an A-edge or another V-edge. The connections of these V-edges in an A-cell are determined when they are identified. Thus, inserting all these identified V_2-edges to $A(C)$ takes time proportional to the size of $A(C)$ and the number of these V-edges. Let B' denote *all the V-edges* identified. B' contains all the V_2-edges in $CVor(S,P)$. Call a cell of $A(C \cup B')$ or $A(C)$, $A-cell$.

Lemma 3.3: (a) The maximum number of V-

edges in $CVor(S,P)$ is $O(mn^2+n^3)$. (b) The maximum number of the Voronoi (sub) cells which are not bounded by any V_2-edge is $O(mn)$.



Proof: (a) We shall consider the three types of V-edges separately. By Property 1 and the definition of B', $A(C \cup B')$ contains all V-edges of the three types in $CVor(S,P)$. Thus, the bounds for the three types of A-edges in $A(C \cup B')$ are these for the corresponding types of V-edges. (1) The number of A_1-edges is bounded by $O(mn+n^2)$ because there are at most $O(n^2)$ bisectors and at most $O(mn)$ cords, and each of which can intersect boundary P at most two points delimiting some A_1-edges. (2) The number of V_2-edges is the size of B'. To find the size of B', let us first consider the maximum number of V-edges on an element $b'_{i,j}$ of B'_i. Property 2 states that an A-edge e of $b'_{i,j}$ is a V-edge iff s_i and s_j are the two closest and visible sites of e. There are two places that s_i and s_j could be the closest and visible sites of e (refer to Fig.3.3a): e is on the sharing boundary of $V(s_i)$ and $V(s_j)$ of $Vor(S)$ and e is outside $V(s_i)$ and $V(s_j)$. In the former case, if e is visible from s_i and s_j, then e belongs to B' because no other site in S can be closer to e than s_i (s_j) to e, and clearly all such edges are connected to form one V-edge. In the latter case, e belongs to B' if e is visible from s_i and s_j but e is not visible from $s_l, ...,$ and s_t, where $s_l, ...,$ and s_t are the sites closer to e than s_i and s_j to e. By Lemma 3.1, $b'_{i,j}$ can cross at most two cords per site of $s_l, ...,$ and s_t so that it contains a sequence of $2(t-l)+1$ edges. By arranging the sites, these edges are alternatively visible to only s_i and s_j, and visible to s_i and s_j as well as one of $s_l, ...,$ and s_t. Then, a half of these edges are V-edges by definition. Then, $b_{i,j}$ contains at most $(t-l)+1 \leq O(n)$ disjoint V-edges. The number of $b'_{i,j}$'s is at most $\frac{n(n-1)}{2}$. Hence, the size of B' is bounded up by $O(n^3)$. (3) The number of these A_3-edges in $A(C \cup B')$ not sharing points with any V_2-edges is at most $O(mn^2)$, because there are at most $O(mn^2)$ A_3-edges in $A(C)$ implied by Lemma 2.2 and there are at most $O(mn+n^2)$ A_1-edges by Part (a)(1) of this lemma. The number of those A_3-edges in $A(C \cup B')$ sharing points with V_2-edges is at most $O(n^3)$ since there are at most $O(n^3)$ V_2-edges by part (a)(2) of this lemma. Thus, the number of A_3-edges in $A(C \cup B')$ (thus, V-edges) is at most $O(mn^2+n^3)$.

(b) Note that if all $O(mn)$ cords of C do not cross each other, then they can form at most $O(mn)$ A-cells. Thus, in order to form more than $O(mn)$ A-cell, some cords associated with different sites and emitting at different vertices must intersect (refer to Fig.3.3b). $O(mn)$ cords can form at most $O(mn^2)$ A-cells implied by Lemma 2.2. However, some of these A-cells are only a part of a V-(sub)cell. To see this, let (e^i_x, e^i_y) be the pairs of A_3-edges associated with sites s_x and s_y such that they are visible from both s_x and s_y and they share the crossover point v_i. Clearly, the maximum number of these pairs are $\frac{m}{2}-2$ (that is, $1 \leq i \leq \frac{m}{2}-2$). For a pair (e^j_x, e^j_y) which do not intersect b'_{xy}, one of e^j_x and e^j_y must entirely lie on the halfplane $H(b_{xy})$ not containing its associated site. Then, this edge cannot belong to $CVor(S,P)$. Since one of every two such edges in the $\frac{m}{2}-4$ pairs does not belong to $CVor(S,P)$, each of the A-cells whose boundary contains such a pair must be merged to another A-cell. That is true for all $s_x, s_y \epsilon S$. Then, only $O(mn)$ such V-cells can be formed from $A(C \cup B')$. □

Lemma 3.4: $CVor(S,P)$ can be determined from $A(C \cup B')$ in $O(mn^2+n^3)$ time and space.

Proof: We shall construct one V-cell at a time from $A(C \cup B')$ by deleting those A_3-edges which are not V_3-edges. To do so, we first construct all those V-cells, each of which contains at least one V_2-edge on its boundary (case 1). We then construct the remaining V-cells (case 2). In case 1, any A-edge e belonging to B' is a V-edge by the definition of B'. Let e be determined by s_i and s_j. Thus, each of the two A-cells sharing e belongs either to $V(s_i)$ or to $V(s_j)$. Without loss of generality, let us consider the A-cell belonging to $V(s_i)$. We now execute the following expansion procedure.

We assume that each of the type-2 or type-3 edges has two sides w.r.t. the following traversal. Traverse the boundary of the A-cell starting at e. If an edge e' is encountered. We shall identify e' as follows: If e' is of type-1, then e' is a V_1-edge of $V(s_i)$ by Property 2; If e' is of type-2, then e' is a V_2-edge by the definition of B'; If e' is of type-3 and if e' and s_i lie on the same halfplane determined by $b_{i,k}$, where s_k is the site associated with the cord containing e', then e' is not a V_3-edge and e' is deleted, otherwise e' is a V_3-edge. We mark the identified V-edge on the side facing the cell. For further traversal, we shall consider two cases. Let e' and e'' be two consecutive edges. If both e' and e'' are V-edges, then we proceed to next edge e''' of e'' on the boundary of the current A-cell. If e' is a

V-edge and e'' is not, then the current A-cell should be expanded. We turn e'' at the shared point v of e' and e'' in the direction of leaving e' and find the next edge e''' sharing v. Repeat this process on e'''. This traversal continues until all the edges in the boundary of the current A-cell are marked.

Then, we take a new edge which has at least one side unmarked from B' and repeat the above expansion procedure until B' is empty. Clearly, the remaining A-cells are bounded by only A_1-edges and A_3-edges. In case 2, we arbitrarily choose a remaining A-cell. According to its visible-site list, we use a brute-force method to determine its closest site, say s_i. Then, this A-cell belongs to some V-(sub)cell of $V(s_i)$. We then expand it to the desired V-(sub) cell by the above procedure. Repeat the above expansion procedure for each remaining A-cell until all the A-cells in $A(C \cup B')$ are examined. Let us consider the time complexity of the above process. The expansion procedure takes time proportional to the size of the resulting V-(sub)cell and the number of A-edges deleted in the procedure. Thus, case 1 takes time proportional to the number of V-edges identified and the A-edges deleted, which is bounded by $O(mn^2 + n^3)$. In case 2, the brute-force method to determine the closest and visible site for an A-cell takes time proportional to the size of its visible-site list, which is bounded by n. There are $O(mn)$ A-cells which are bounded by only A_1-edges and A_3-edges by Lemma 3.3, case 2 takes at most $O(mn^2)$ time. The space bound is the same. □

Algorithm Find-CVor(S;P)

Input: S and P.

Output: CVor(S;P), represented by SLPG.

Method:

Step 1. Construct arrangement $A(C)$ by Algorithm A(C).

Step 2. For each site $s_i \epsilon S$ Do (* Construct the set B_i' associated with s_i *)

$B_i \leftarrow \emptyset$;

For each site $s_j \epsilon (S - \{s_i\})$ Do

insert $b'_{i,j}$ to $A(C \cup B_i)$

EndDo;

For each edge $b'_{i,j} \epsilon B_i'$

identify all the A-edges in $b'_{i,j}$ which are

(sub) V-edges

EndDo;

EndDo.

Step 3 Form $A(C \cup B')$ and use expansion procedure to find all the V-cells of $CVor(S, P)$.

Lemma 3.5: Algorithm Find-CVor(S,P) produces $CVor(S, P)$ in $O(mn^2 + n^3)$ time and space.

Proof: Step 1 constructs $A(C)$, Step 2 finds $A(C \cup B')$ by Lemma 3.2. Step 3 identify all the (sub) V_1-edges and V_3-edges of $CVor(S, P)$ by Lemma 3.4. Thus, Find-CVor(S,P) produces $CVor(S, P)$. Let us consider the time complexity. Step 1 takes $O(mn^2+n^3)$ by Lemma 2.5. Step 2 takes $O(mn^2+n^3)$ by Lemma 3.2. Steps 3 takes time linear in the size of $A(C \cup B')$ by Lemma 3.3. □

4 Concluding remarks

In this paper, we present a worst-case optimal algorithm for constructing a geometric object w.r.t the sites inside a simple polygon. The object can be used for reporting the sites visible from a query point efficiently. We also proposed an algorithm for finding the constrained Voronoi diagram of sites inside a simple polygon. The optimality of the latter algorithm is not known. The diagram can be used for reporting the site closest and visible to a query point efficiently. By organizing $CVor(S, P)$ into a search structure for point location, the Visible and Nearest Neighbor site of a given site can be determined in $O(log(mn^2 + n^3)$ time (that is $O(nlogn)$ time if $m = O(n^k)$ for fixed k). Thus, the Visible All Nearest Neighbors of S and P can be found in $O(nlogn)$ time, and the Visible Nearest Neighbor of S and P, in an extra linear-time.

Acknowledgement The author would thank Professor Paul Gillard for many valuable comments on the paper.

References

[1] Avis D. and Toussaint G., (1981), "An optimal algorithm for determining the visibility of a polygon from an edge", *IEEE Trans. on Computers*, Vol. C-30, No. 12, pp.910-914.

[2] Bhattacharya B., Kirkpatrick D., and Toussaint G., (1989), "Determining sector visibility of a polygon", *Proceedings of 5th ACM Symposium on Computational Geometry*, Saarbruchen, West Germany, pp.247-254.

[3] Edelsbrunner H.,O'Rourke J., Seidel R., (1986), "Constructing arrangements of lines and hyperplanes with applications", *SIAM J. Computing.* , Vol.15, No.2, pp.341-363.

[4] Edelsbrunner H., Guibas L., and Stolfi J., (1986), "Optimal point location in a Monotone subdivision", *SIAM J. Comput.* 15 pp.317-340.

[5] El Gindy H. and Avis D., (1981), "A linear algorithm for computing the visibility polygon from a point", *J. Algorithms* 2(2), pp. 186-197.

[6] Guibas L., Hershberger J., Leven D., Sharir M., and Tarjan R., (1987), "Linear time algorithms for visibility and shortest path problems inside triangulated simple polygons", *Algorithmica* 2, pp.209-233.

[7] Kirkpatrick D., (1983), "Optimal search in planar subdivisions", *SIAM J. Comput.* 12(1) pp.28-35.

[8] Lee D. T., Lin A., (1986), "Generalized Delaunay triangulations for planar graphs", *Discrete Computational Geometry* , 1, pp.201-217.

[9] Lingas A., (1989), "Voronoi diagrams with barriers and their applications", *Information Processing Letters* , 32(1989), pp. 191-198.

[10] Preparata F., Shamos M.,(1985), " *Computational Geometry* ", springer Verlag.

[11] Sack J. and Suri S., (1990), "An optimal algorithm for detecting weak visibility of a polygon", *IEEE Trans. on Computers*, Vol. 39, No. 10, pp.1213-1219.

[12] Wang C. and Tsin Y., (1990), "Finding constrained and weighted Voronoi diagram in the plane", *Proceedings of 2nd Canadian Conference on Computational Geometry*, pp. 200-203.