AN OPTIMAL ALGORITHM FOR DETERMINING EDGE VISIBILITY IN A POINT-VISIBLE POLYGON

S. Y. Shin
T. C. Woo

Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2117

Technical Report 87-8

March 1987

An Optimal Algorithm for Determining Edge Visibility in a Point-Visible Polygon

S. Y. Shin

T. C. Woo*

Department of Industrial and Operations Engineering

The University of Michigan

Ann Arbor, Michigan 48109-2117

March, 1987

*Correspondence should be sent to T.C. Woo at the above address.

Abstract

By testing the floor plan of a building as a polygon, we study the problem of locating a camera for surveillance. If the polygon is point-visible then only one camera is needed. A visible edge corresponds to the wall on which the camera is to be mounted.

While a given edge can be tested for visibility in time linear in the number of edges n [1], there can be O(n) such edges. In this paper, we give an O(n) time algorithm for finding all visible edges in a point-visible polygon.

1. Introduction

Two points u and w in a polygon P are said to be visible if the line segment L(u,w) joining u and w is completely contained in P. Suppose that P represents the floor plan of a building. The celebrated "Art Gallery Problem" [3,4] seeks a bound on the number of guards such that the entire interior of P is visible at all times. The kernel of P, denoted by K(P), is the set of all points in P such that every point u in P is visible from any point w in K(P). Lee and Preparata [6] showed an algorithm for finding K(P) in time linear in the number of edges n of a given polygon P. A polygon with a non-empty kernel is a point-visible polygon [5]. Figure 1 shows the relation of K(P) to a point-visible polygon P.

<Insert Figure 1>

For surveillance applications, a television camera can be located in K(P). To be less obstructive, a camera may be better placed on a wall. However, K(P) may not be on the boundary of P. This leads to the notion of edge-visibility. According to Avis and Toussaint [1], there are three types of edge-visibility. Let an edge $E(v_i,v_{i+1})$ be a directed line segment $L(v_i,v_{i+1})$ joining two adjacent vertices v_i and v_{i+1} on the boundary B(P) of P. Then, (i) P is completely visible from an edge $E(v_i,v_{i+1})$ if, for every point u in P and every point w in $E(v_i,v_{i+1})$, u and w are visible. (ii) P is strongly visible from $E(v_i,v_{i+1})$ if there exists a point w such that every point u in P is visible from w. (iii) P is weakly visible from $E(v_i,v_{i+1})$ if, for every point u in P, there exists some point w in $E(v_i\text{-}v_{i+1})$ (depending on u) such that u and w are visible.

We now relate edge visibility to point visibility. Let a visibility polygon V(u,P) be the set of all points in P, which are visible from a point u [2]. If P is completely visible from an edge $E(v_i,v_{i+1})$, then $E(v_i,v_{i+1})$ is completely contained in K(P). In other words, the visibility polygon V(w,P) from w is the entire polygon P, where w is any

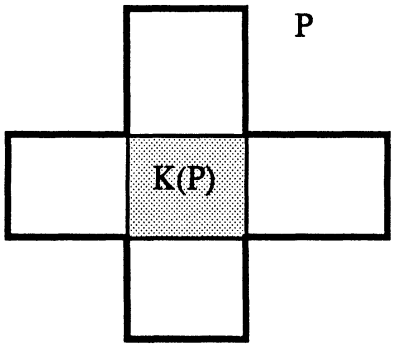


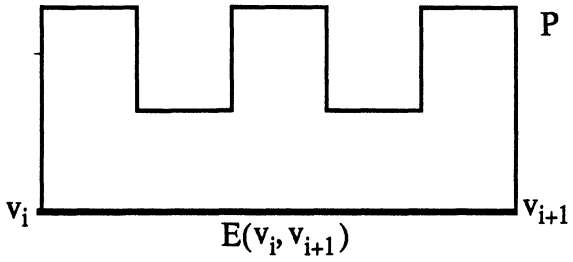
Figure 1 : A point-visible polygon P has a non-empty kernel K(P).

point in the edge $E(v_i,v_{i+1})$. Figure 2(c) illustrates such a polygon. If P is strongly visible from $E(v_i,v_{i+1})$, then $E(v_i,v_{i+1}) \cap K(P) \neq \emptyset$. In other words, there exists some point w in $E(v_i,v_{i+1})$ such that $P = V(w,P)$. Figure 2 (b) illustrates such an edge. On the other hand, if P is weakly visible from $E(v_i,v_{i+1})$, $E(v_i,v_{i+1})$ may not share a point with K(P) as illustrated in Figure 2 (a). However, the union of the visibility polygons V(w,P), where $w \in E(v_i,v_{i+1})$, covers P.

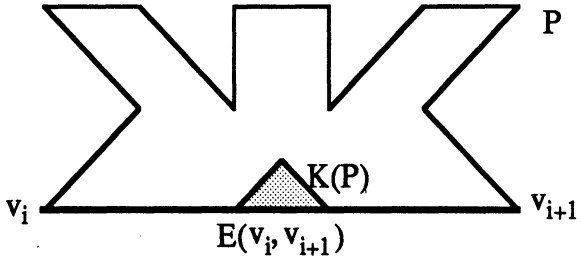
<Insert Figure 2>

In the context of locating surveillance equipment on a wall, a camera can be fixed at any point in a completely visible edge since P is visible from every point in the edge. For a strongly visible polygon, a camera should be mounted at a point u in an edge $E(v_i,v_{i+1})$ such that $u \in K(P)$. These two cases, complete and strong edge-visibility, entail a stationary camera. If the field of view of the camera is sufficiently wide, no panning is required. Otherwise, the camera, while fixed at a point, must rotate about that point for complete coverage. The case of weak edge-visibility entails a moving camera. To ensure that union of visibility polygons at all points in a weakly visible edge $E(v_i,v_{i+1})$, the camera must be translated from v_i and v_{i+1} and back.

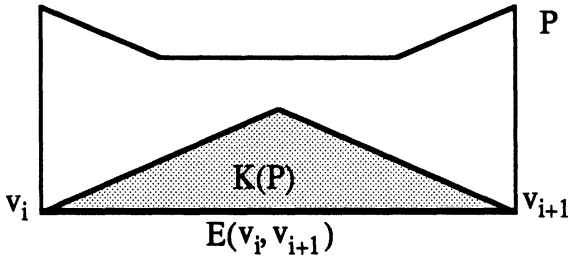
The characterization of edge visibility suggest three problems: that of finding completely, strongly, and weakly visible edges in a point-visible polygon. The first two problems can trivially be solved in O(n) time by checking intersection of K(P) and the boundary B(P) of P. The last problem of fiding a weakly visible edge in a point-visible polygon P is the focus of this paper. Avis and Toussaint [1] gave an O(n) algorithm for determing whether or not a polygon is visible from a given edge. Since there are n edges, finding all the weakly visible edge would take $O(n^2)$ time, if we use their algorithm. Here, we give an O(n) algorithm for soliving this problem.



(a) P is weakly visible from $E(v_i, v_{i+1})$.



(b) P is strongly visible from $E(v_i, v_{i+1})$.



(c) P is completely visible from $E(v_i, v_{i+1})$.

Figure 2 : P is visible from $E(v_i, v_{i+1})$.

2. Characterization of Edge Visibility

In this section we develop properties that enable us to find all weakly visible edges. For brevity, we refer to a weakly visible edge as a visible edge. First, some definitions and terms are in order.

Let P be a simple polygon with n vertices. Each vertex v_i, i = 0,1,2... , (n-1), is represented by its Catesian coordinates, (x_i,y_i). Starting from a vertex v_o and traversing the boundary B(P) of P in the counter-clockwise order, we label the j^{th} vertex from v_o as v_i, where i is taken j modulo n. These vertices in sequence are maintained as a circular doubly linked list.

A vertex v_i is said to be concave if the internal angle at v_i is less than 180 degrees. v_i is said to be reflexive, otherwise. (L(u,w) denotes a directed line segment joining two points u and w with a direction from u to w. An open edge $\tilde{E}(v_i,v_{i+1})$ is a directed line segment $L(v_i,v_{i+1})$ without its two endpoints, v_i and v_{i+1}. Let u and w be two distinct points on B(P). A chain $C_h(u,w)$ is said to be the portion of B(P) from u to w in the counter-clockwise order. An open chain $\tilde{C}_h(u,w)$ is a chain $C_h(u,w)$ excluding its two endpoints u and w. A ray RAY (u,w) is a half line starting at u with the direction from u to w.

Consider an edge $E(v_i,v_{i+1})$ of P. Suppose that $E(v_i,v_{i+1})$ lies completely to the right of $E(v_r,v_{r+1})$ for some $0 \leq r < n$. Since the interior of P lies to the left of $E(v_i,v_{i+1})$, every point on an open edge $\tilde{E}(v_r,v_{r+1})$ is not visible from any point on $E(v_i,v_{i+1})$. This gives a characterization of a visible edge.

Lemma 2.1: Suppose that a simple polygon P is visible from an edge $E(v_i,v_{i+1})$ of P. Then, for every edge $E(v_r,v_{r+1})$, there exists at least a point on $E(v_i,v_{i+1})$, which is on or to the left of $E(v_r,v_{r+1})$.

Lemma 2.1 provides a necessary condition for a visible edge. Avis and Toussaint supplied a necessary and sufficient condition [1].

Lemma 2.2: A simple polygon P is weakly visible from $E(v_i,v_{i+1})$ if and only if the boundary B(P) of P is weakly visible from $E(v_i,v_{i+1})$.

A trivial lower bound in time complexity for checking this condition for a given edge is $\Omega(n)$ [1]. Since there are n edges in P, it is hard to accomplish a linear time complexity with Lemma 2.2.

We will develop another characterization of a visible edge by exploiting point-visibility of P. A point x in RAY $(v_i,v_j) \cap B(P)$ is said to be a visible intersection point if x is visible from v_i. Let

$VX_{i,i+1}$ = v_{i+1}, if v_{i+1} is concave.

Otherwise, it is the first visible intersection point in RAY $(v_i,v_{i+1}) \cap \tilde{C}_h(v_{i+1},v_i)$ as $C_h(v_{i+1},v_i)$ is scanned counter-clockwise from v_{i+1}.

$VX_{i+1,i}$ = v_i, if v_i is concave.

Otherwise, it is the first visible interesection point in RAY $(v_{i+1},v_i) \cap \tilde{C}_h(v_{i+1}, v_i)$ as $C_h(v_{i+1}, v_i)$ is scanned clockwise from v_i.

Suppose that v_{i+1} is not concave. Then, there does not exist any point in $\tilde{E}(v_i,v_{i+1})$, which is visible from $\tilde{C}_h(v_{i+1}, VX_{i,i+1})$. As illustrated in Figure 3, the same is true for $\tilde{E}(v_i,v_{i+1})$ and $\tilde{C}_h(VX_{i+1,i}, v_i)$. This motivates the notion of a <u>feasible chain</u> which is defined as follows:

$$FC(VX_{i,i+1},VX_{i+1,i}) = \{x | x \in E(v_i,v_{i+1}) \text{ or } x \in C_h(VX_{i,i+1},VX_{i+1,i})\}.$$

This definition gives rise to a new characterization of a visible edge.

<Insert Figure 3>

<u>Lemma 2.3</u>: A simple polygon P is visible from $E(v_i,v_{i+1})$ if and only if $E(v_i,v_{i+1}) \cap FC(v_k,v_{k+1}) \neq \emptyset$ for all $0 \leq k < n$.

[Proof] Let P be visible from $E(v_i,v_{i+1})$. Suppose that there exists a feasible chain $FC(v_r,v_{r+1})$ such that $FC(v_r,v_{r+1}) \cap E(v_i,v_{i+1}) = \emptyset$ for some $r \neq i$. Then, there does not exist any point in $\tilde{E}(v_r,v_{r+1})$, which is visible from $E(v_i,v_{i+1})$. This is a contradiction. Hence, $E(v_i,v_{i+1}) \cap FC(v_k,v_{k+1}) \neq \emptyset$ for all k.

Now, assume that $E(v_i,v_{i+1}) \cap FC(v_k,v_{k+1}) \neq \emptyset$ for all k. Suppose that $E(v_i,v_{i+1})$ is not a visible edge. By Lemma 2.1, there exist a point x in B(P), which is not visible from $E(v_i,v_{i+1})$. If the visibility polygon V(x,P) from x is taken out from P, then the remaining portion of P is partitioned into one or more disjoint regions as shown in Figure 3. Clearly, only one of these regions contains $E(v_i,v_{i+1})$ on its boundary. Let R denote this region. R is bounded by a chain $C_h(u,w)$ and a line segment L(w,u) on the boundary of V(x,P). Without loss of generality, let u be closer to x than w is. Then, u must be a vertex of P, say v_j. As illustrated in Figure 4, the internal angle (v_{j-1},v_j,w) is greater than or equal to 180 degrees. Therefore, either $R \cap FC(v_{j-1},v_j) = u$ or $R \cap FC(v_{j-1},v_j) = \{u,w\}$. Since $L(w,u) \subset V(x,P)$, $E(v_i,v_{i+1}) \cap FC(v_{j-1},v_j) = \emptyset$, which is a contradiction. Hence, $E(v_i,v_{i+1})$ must be a visible edge.∎

<Insert Figure 4>

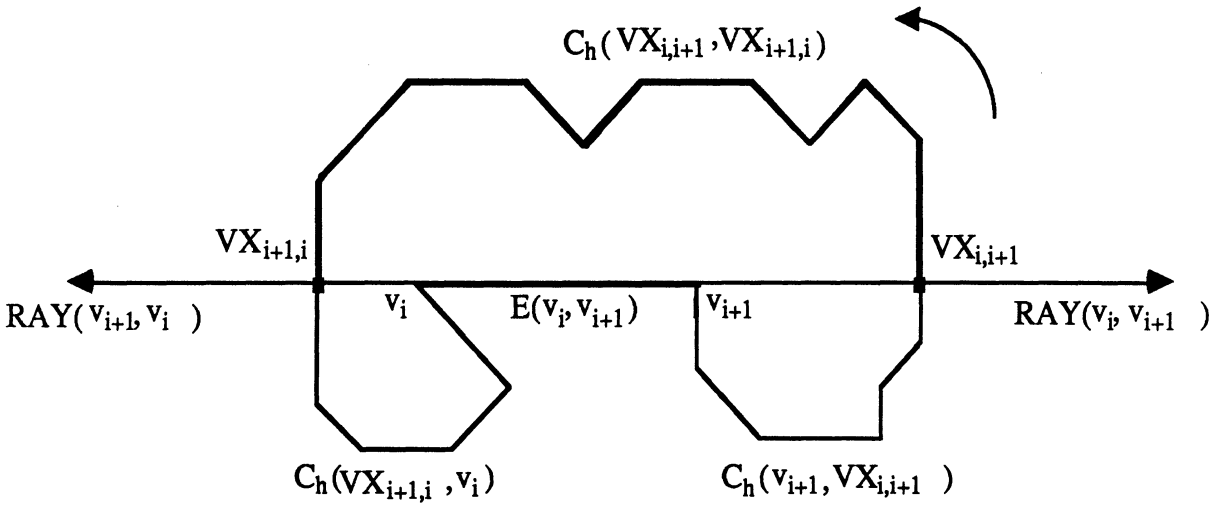


Figure 3 : $E(v_i, v_{i+1})$is visible from neither $C_h(v_{i+1}, VX_{i,i+1})$ nor $C_h(VX_{i+1,i}, v_i)$.

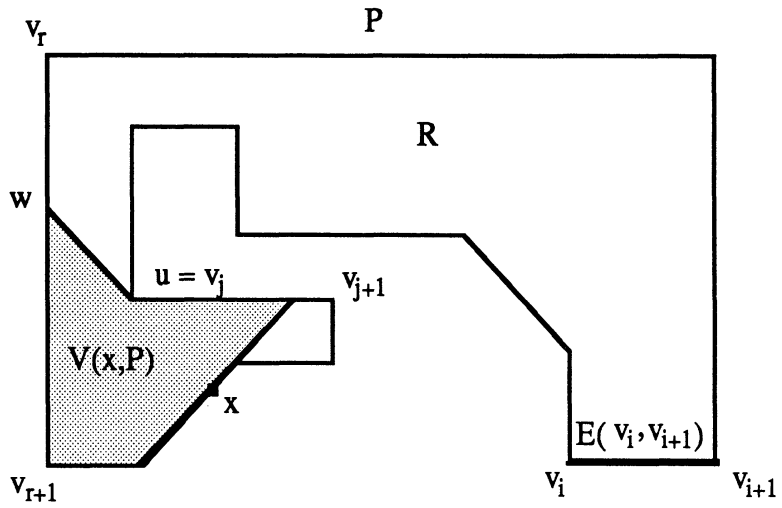


Figure 4 : $E(v_i, v_{i+1}) \sqcap FC(v_j, v_{j+1}) = \emptyset$.

3. Determining All Visible Edges

The basic idea for finding all visible edges is to discard, from B(P), every edge which does not share any point with $FC(v_i,v_{i+1})$, $0 \leq i < n$. In other words, an edge which lies completely in $\tilde{C}_h(v_{i+1}, VX_{i,i+1})$ or $\tilde{C}_h(VX_{i+1,i},v_i)$ cannot be a visible edge. Hence, if all the edges lying completely in $\tilde{C}_h(v_{i+1}, VX_{i,i+1})$ or $\tilde{C}_h(VX_{i+1,i}, v_i)$ for any i are discarded from B(P), what remains must be all visible edges by Lemma 2.3. In order to find $C_h(v_{i+1}, VX_{i,i+1})$ and $C_h(VX_{i+1,i},v_i)$, it is required to compute $VX_{i,i+1}$, and $VX_{i+1,i}$, respectively, By definition, $VX_{i,i+1}$ is v_{i+1} if v_{i+1} is concave. However, it is not obvious how to compute $VX_{i,i+1}$ when v_{i+1} is reflexive. The following lemma suggests a way for computing $VX_{i,i+1}$.

Lemma 3.1: Let P be point-visible. Suppose that v_{i+1} is reflexive. Then, $VX_{i,i+1}$ is the first interesection point of $\tilde{C}_h(v_{i+1},v_i)$ and $RAY(v_i,v_{i+1})$ as $C_h(v_{i+1},v_i)$ is scanned counter-clockwise from v_{i+1}.

[Proof] Let x be the first intersection point of $\tilde{C}_h(v_{i+1},v_i)$ and $RAY(v_i,v_{i+1})$ as $\tilde{C}_h(v_{i+1},v_i)$ is scanned counter-clockwise from v_{i+1}. If x is internally visible from v_i, then $x = VX_{i,i+1}$ by definition. Suppose that x is not internally visible from v_i. Then, $x \notin V(v_i,P)$. Let w be the first intersection point of $\tilde{C}_h(v_{i+1},v_i)$ and $RAY(v_i,v_{i+1})$, which is visible from v_i, as $C_h(v_{i+1},v_i)$ is scanned counter-clockwise from v_{i+1}. Clearly, the region R bounded by $C_h(v_{i+1},w)$ and $L(w,v_{i+1})$ is a simple polygon. Since w is colinear with $E(v_i,v_{i+1})$, $R \cap V(v_i,P) = L(v_{i+1},w)$. Suppose that w is visible from x. Then, $L(w,x) \subset P$. Since $L(v_i,w) \subset P$ and since x is colinear with $E(v_i,v_{i+1})$, $L(v_i,x) \subset P$, which is a contradiction. Therefore $w \notin V(x,P)$, and thus $L(v_{i+1},w) \cap V(x,P) = \varnothing$ as shown in Figure 5.

Now consider V(x,P). By the way in which w is chosen, $x \in R$. Since $L(v_{i+1},w) \cap V(x,P) = \varnothing$, $V(x,P) \subset (R \setminus L(v_{i+1},w))$. Therefore, $V(x,P) \cap V(v_i,P) = \varnothing$, i.e., there does

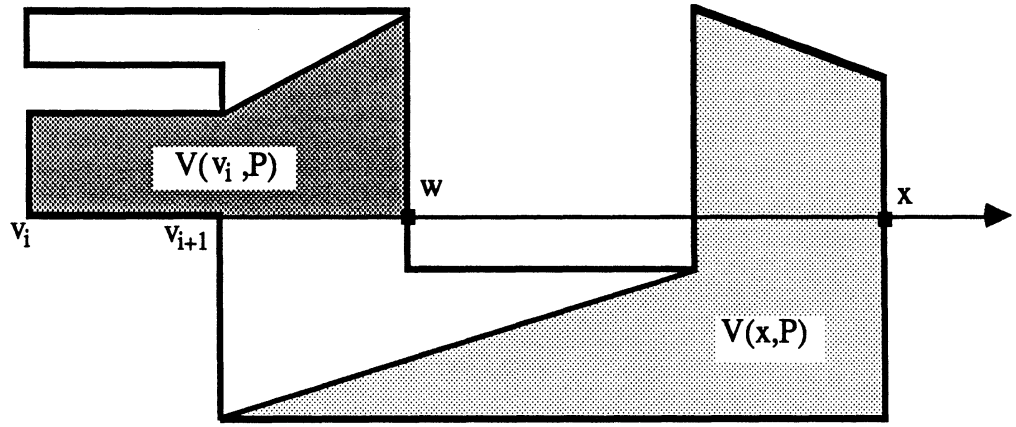


Figure 5 : $L(v_{i+1}, w) \sqcap V(x,P) = \emptyset$.

not exist any point y in P such that v_i and x can be internally visible from y. This is contradiction since P is point-visible. Thus, x is visible from v_i. Hence, $x = VX_{i,i+1}$. ∎

<Insert Figure 5>

The companion lemma of Lemma 3.1 can be stated for $VX_{i+1,i}$ from the symmetry of $VX_{i,i+1}$ and $VX_{i+1,i}$.

Lemma 3.2: Let P be point-visible. Suppose that v_i is reflexive. Then, $VX_{i+1,i}$ is the first intersection point of $\tilde{C}_h(v_{i+1},v_i)$ and $RAY(v_{i+1},v_i)$ as $C_h(v_{i+1},v_i)$ is scanned clockwise from v_i.

In order to refer to the open chains $\tilde{C}_h(v_{i+1},VX_{i,i+1})$ and $\tilde{C}_h(VX_{i+1,i},v_i)$ conveniently, $FC(v_i,v_{i+1})$ is further decomposed as shown in Figure 5. A chain $C_h(VX_{i,i+1}, v_{i+1})$ is said to be a right-feasible chain $RFC(v_i,v_{i+1})$ with respect to $E(v_i,v_{i+1})$. $C_h(v_i,VX_{i+1,i})$ is said to be a left-reasible chain $LFC(v_i,v_{i+1})$ with respect to $E(v_i,v_{i+1})$. It is clear that $\tilde{C}_h(v_{i+1},VX_{i,i+1})$ and $\tilde{C}_h(VX_{i+1,i},v_i)$ are the complements of $RFC(v_i,v_{i+1})$ and $LFC(v_i,v_{i+1})$, respectively. Furthermore, $FC(v_i,v_{i+1}) = RFC(v_i,v_{i+1}) \cap LFC(v_i,v_{i+1})$. Therefore, the following lemma is immediate from Lemma 2.3.

Lemma 3.3: A simple polygon P is internally visible from $E(v_i,v_{i+1})$ if and only if $E(v_i,v_{i+1}) \cap RFC(v_k,v_{k+1}) \neq \emptyset$ and $E(v_i,v_{i+1}) \cap LFC(v_k,v_{k+1}) \neq \emptyset$ for all $0 \leq k < n$.

Based on Lemma 3.3, two operations, the right scan and the left scan will be developed. These operations are for discarding, from B(P), all edges which do not share any point with either $RFC(v_i,v_{i+1})$ or $LFC(v_i,v_{i+1})$ for some $0 \leq i < n$. For the right scan, B(P) is traversed from v_0 in the clockwise order to visit all vertices of P. At each vertex v_i, the edges of P are scanned in the counter-clockwise order from $E(v_2,v_3)$ until the edge $E(v_r, v_{r+1})$ containing $VX_{i,i+1}$ is encountered. Since $VX_{i,i+1} \in E(v_r,v_r+1)$, all the scanned edges except $E(v_r,v_{r+1})$ are completely contained in $C_h(v_{i+1}, VX_{i,i+1})$ and do not share any point with $RFC(v_i,v_{i+1})$. These

edges can be discarded by Lemma 3.3. This process is called the right scan at v_i. Performing the right scan at each vertex in turn, it is possible to discard all edges which do not share any point with $RFC(v_k,v_{k+1})$ for some k. The left scan is symmetric to the right scan. The following data structures will be used for scanning operations:

(1) VERTEX: An array containing all vertices of P in the order appearing on B(P)

(2) EDGE: A doubly linked list containing all edges of P in the order appearing on B(P)

Algorithm 3.1 is for performing the right scan. In the algorithm, CW(x) and CCW(x) denote the next clockwise and counter-clockwise edges of edge x, respectively. Mod (i,n) is a function computing i modulo n. SCAN and RIGHT are Boolean variables which control the outer and inner while-loops, respectively.

lAgorithm 3.1: (Performing the right scan)

```
                procedure   RSCAN (VERTEX,EDG
                   begin
Step 0                NEXT ←CCW(E(v1,v2)) in EDGE
                      i←0
                      SCAN←true
Step 1                while (EDGE ≠ ∅ and SCAN) do
                            begin
Step 1a                      RIGHT←true
                             if vi+1 is concave, then RIGHT←false
Step 1b                      while (EDGE ≠ ∅ and RIGHT) do
                                begin
                                   if NEXT is to the right of RAY(vi,vi+1),
                                   then begin
                                      SAVE←NEXT
                                      NEXT←CCW(NEXT) in EDGE
                                      discard SAVE from EDGE
                                   end
                                   else RIGHT←false
                                end
Step 1c                      if CW(NEXT) = E(vi+1,vi+2), then
                                NEXT←CW(NEXT)
                             i←mod(i-1,n)
                             if i = 0, then SCAN←false
                            end
                   end
```

Lemma 3.4: Algorithm 3.1 does not discard any visible edge but discards, in O(n) time, all edges which are completely contained in $\tilde{C}_h(v_{i+1}, VX_{i,i+1})$ for some $0 \leq i < n$.

[Proof] First, we inductively show that the algorithm does not discard any visible edge but discard all edges which are completely contained in $\tilde{C}_h(v_{i+1}, VX_{i,i+1})$ for some $0 \geq i < n$. The induction will be on the number of times Step 1 is reached.

Initially, step 0 sets NEXT = $E(v_2,v_3)$, i = 0, and SCAN = true. Therefore, $v_i = v_o = v_n$. Since neither EDGE = $\varnothing$ nor SCAN = false, the body of the outer while-loop is executed. Clearly, $v_{i+1} = v_1$. If v_1 is concave, the inner while-loop is skipped by setting RIGHT = false. Otherwise, the inner while-loop (Step 1b) scans EDGE from $E(v_2,v_3)$ in the counter-clockwise order until an edge $E(v_t,v_{t+1})$ not completely lying to the right of $RAY(v_o,v_1)$ is encountered. $E(v_t,v_{t+1})$ must contain $VX_{o,1}$. Therefore, all edges which are completely in $\tilde{C}_h(v_1, VX_{o,1})$ are discarded, but no visible edge is discarded from EDGE. Notice that this is what the right scan at v_n (or equivalently v_o) is required to do. Furthermore, NEXT points to $E(v_1,v_2)$ by Step 1c so that the right scan at v_{n-1} can be performed.

Suppose that the following is true until Step 1 is executed k times, $0 < k < n$:

(1) No visible edge is discarded.

(2) All edges, which are contained in $\tilde{C}_h(v_{i+1}, VX_{i,i+1})$, $k < i \geq n$, are discarded.

(3) NEXT points to the first edge to be traversed for the right scan at v_k.

Notice that NEXT does not necessarily point to $E(v_{k+2}, v_{k+3})$ since $E(v_{k+2},v_{k+3})$ may be discarded from EDGE during the previous scans. Therefore, assumption (3) can be restated as follows:

(3) NEXT points to $E(v_s,v_{s+1})$ in EDGE, where

$E(u_s,u_{s+1}) = E(v_{k+2},v_{k+3})$, if $E(v_{k+2}, v_{k+3})$ is in EDGE.

Otherwise, it is the edge in EDGE such that no edge in $C_h(v_{k+2},v_s)$, $k+2 \neq s$, exists in EDGE.

We show that these three assumptions are satisifed after the k^{th} execution of Step 1 (the outer while-loop). There are two possibilities depending on EDGE.

Case 1: EDGE $= \varnothing$

Case 2: EDGE $\neq \varnothing$

If EDGE $= \varnothing$, then the algortihm is terminated. By assumptions (1) and (2) in the induction hypothesis, the result holds trivially. In this case, there does not exist any visible edge. Suppose that EDGE $\neq \varnothing$. There are two subcases:

Case 2a: NEXT points to $E(v_{k+2}, v_{k+3})$

Case 2b: NEXT does not point to $E(v_{k+2},v_{k+3})$.

Case 2a: If v_k is concave, then the inner while-loop is skipped. Therefore, no edge is discarded from EDGE. By the induction hypothesis, assumptions (1) and (2) are satisfied. By Step 1c, assumption (3) is also satisfied. Thus, the result is obvious.

Suppose that v_k is reflexive. Since NEXT points to $E(v_{k+2},v_{k+3})$, Step 1b (the inner while loop) scans EDGE from $E(v_{k+2},v_{k+3})$ in the counter-clockwise order until the edge $E(v_t,v_{t+1})$ not completely lying to the right of $RAY(v_k,v_{k+1})$ is found. If such an edge $E(v_t,v_{t+1})$ does not exist, then EDGE $= \varnothing$. In this case, all the discarded edges in the k^{th} execution of Step 1 are lies to the right of $E(v_k,v_{k+1})$. By Lemma 2.1, none of them can be a visible edge. This together with the induction hypothesis guarantee the result. Now, assume that $E(v_t,v_{t+1})$ exists. By Lemma 3.1,

$$C_h(v_{k+1}, VX_{k,k+1}) \subseteq C_h(v_{k+1},v_{t+1}).$$

Therefore, no edge in $\tilde{C}_h(v_{k+1}, VX_{k,k+1})$ remains in EDGE. However, some edges not contained in $\tilde{C}_h(v_{k+1},VX_{k,k+1})$ may be discarded. These edges, if any, must lie

to the right of $E(v_k,v_{k+1})$, and none of them can be a visible edge. Thus, assumptions (1) and (2) follow. After executing Step 1b, no edge completely contained in $C_h(v_{k+2},v_t)$ is available in EDGE, and NEXT points to $E(v_t,v_{t+1})$. Therefore, if $E(v_{k+1}, v_{k+2})$ is in EDGE, then it must be CW (NEXT), i.e., the clockwise edge of NEXT. Accordingly, by Step 1c, assumption (3) also satisfied.

Case 2b: In this case, NEXT points to the edge $E(v_s, v_{s+1})$ in EDGE such that no edge in $C_h(v_{k+2},v_s)$ exists in EDGE. By a similar argument as in Case 2a, the result follows.

Since the three assumptions are true for all $0<k\leq n$, the correctness of the algorithm is immediate.

Now, let us analyze the time complexity. At each vertex v_i, $0\leq i<n$ (or equivalently $0<i\leq n$), Algorithm 3.1 scans undiscarded edges in EDGE from $E(v_s,v_{s+1})$ in the counter-clockwise order until either EDGE = $\varnothing$ or an edge $E(v_t,v_{t+1})$, which does not completely lie to the right of RAY (v_i,v_{i+1}), is encountered, where

$$E(v_s,v_{s+1}) = \begin{cases} E(v_{i+2},v_{i+3}) \text{ if } E(v_{i+2},v_{i+3}) \text{ is in EDGE} \\ \text{the edge in EDGE such that no edge in } C_h(v_{i+2},v_s), \\ i+2 \neq s \text{ exists in EDGE, otherwise,} \end{cases}$$

Therefore,

Total number of edges scanned by Algorithm 3.1

$$= \sum_{i=0}^{n} \text{the number of edges scanned at } v_i$$

$= \sum_{i=0}^{n-1}$ the number of edges scanned and undiscarded at v_i

$+ \sum_{i=0}^{n-1}$ the number of edges scanned and discarded at v_i

Since there at most one scanned and undiscarded edge at each v_i, i.e., the last scanned edge if EDGE $\neq \emptyset$, the total number of edges scanned and undiscarded is at most n. The total number of edges scanned and discarded is also at most n since a discarded edge is never scanned again. Thus, the total number of edges scanned by Algorithm 3.1 is at most 2n. Since a constant number of operations are needed for each scanned edge, Algorithm 3.1 requires O(n) time in the worst case.∎

From the symmetry of right and left scans, a similar algorithm can be given for the left scan.

Algorithm 3.2: (Performing the left scan)

```
                Procedure  LSCAN (VERTEX, EDGE)
                   begin
Step 0                NEXT←CW (E(v_{n-1},v_0)) in EDGE
                      i←0
                      SCAN←true
Step 1                while (EDGE ≠ ∅ and SCAN) do
                            begin
Step 1a                        LEFT←true
                               if v_i is concave, then LEFT←false
Step 1b                        while (EDGE ≠ ∅ and LEFT) do
                                  begin
                                     if NEXT is to the left of RAY (v_{i+1},v_i)
                                     then begin
                                        SAVE←NEXT
                                        NEXT←CW(NEXT) in EDGE
                                        discard SAVE from EDGE
                                     end
                                     else LEFT←false
                                  end
                               if CCW (NEXT) = E(v_{i-1},v_i), then
                                  NEXT←CCW (NEXT)
                               i←mod (i + 1,n)
                               if i = 0, then SCAN←false
                            end
                   end
```

Lemma 3.5: Algorithm 3.2 does not discard any visible edge but discards, in O(n) time, all edges which are contained in $\bar{C}_h$ $(VX_{i+1,i}, v_i)$ for some $0 \leq i < n$.

From Algorithms 3.1 and 3.2, a linear time algorithm for determining all visible edges in a point-visible polygon is immediate.

Algorithm 3.3: (Determining all visible edges in a point-visible polygon)

```
            procedure   VEDGE-P
                  begin
Step 1                REDGE ← EDGE
                      Call RSCAN (VERTEX, REDGE)
Step 2                LEDGE←EDGE
                      Call LSCAN (VERTEX, LEDGE)
Step 3                VEDGE←REDGE∩LEDGE
                  end
```

Theorem 3.1: Algorithm 3.3 finds all visible edge of a point-visible polygon in O(n) time.

[Proof] From Lemma 3.4, Step 1 discards all edges from an edge list REDGE, which are completely in $\bar{C}_h(v_{i+1}, VX_{i,i+1})$ for some $0 \leq i < n$. After executing Step 1, every edge in REDGE shares at least a point with RFC (v_k, v_{k+1}) for all $0 \leq k < n$. Similarily, after executing Step 2, every edge in LEDGE shares at least a point with LFC (v_k, v_{k+1}) for all $0 \leq k < n$ (see Lemma 3.5). By Lemma 3.3, VEDGE which is REDGE ∩LEDGE (see Step 3) contains all visible edges.

By Lemmas 3.4 and 3.5, Steps 1 and 2 can be done in O(n) time. Since both REDGE and LEDGE are sorted, Step 3 can also be done in O(n) time. Hence the time complexity of Algorithm 3.3 is O(n). ∎

A trivial lower bound in time complexity for computing all visible edge in a point-visible polygon with n edges is $\Omega(n)$. Thus, Algorithm 3.3. is optimal within some constant multiplictive factor.

4. Conclusion

It has been an open problem to determine a visible edge, if any, of a simple polygon in O(n) time ever since Avis and Toussaint [1] presented a linear time algorithm for determining if a simple polygon is visible from a given edge.

In this paper, we show that if the kernel of a simple polygon is non-empty, then all visible edges, whether they are completely, strongly, or weakly visible, can be found in O(n) time.

References

1. Avis D. and Toussaint, G., An Optimal Algorithm for Determining the Visibility of a Polygon from an EDGE, IEEE Tran. Comput., C-30, 12, (1981), 910-914.

2. El Gindy, H. and Avis, D., A Linear Algorithm for Computing the Visibility Polygon from a Point, J. Algorithm, 2, (1981), 186-187.

3. Fisk, S., Short Proff of Chvátal's Wachman Theorem, J. of Combinatorial Theory B, Vol. 24, (1978), 374.

4. Kahn, J., Klawe, M., Kleitman, D., Traditional Galleries Require Fewer Watchman, IBM Research Report RJ3021, (Dec. 1980).

5. Lee, D. and Preparata, F., An Optimal Algorithm for Finding the Kernel of a Polygon, J. ACM, 26, 3 (July 1976), 415-421.

6. Woo, T. and Shin, S., A Linear Time Algorithm for Triangulating a Point-Visible Polygon, ACM Trans. Graphics, Vol. 4, 1, (1985), 60-70.