A Linear Algorithm for Computing the Visibility Polygon from a Point*

H. EL GINDY AND D. AVIS

School of Computer Science, McGill University, Montreal, Quebec H3A 2K6, Canada

Received October 18, 1980; revised February 11, 1981

The hidden-line problem in two dimensions is a recurrent problem in computer graphics. In this paper a linear, and thus optimal, algorithm is exhibited for solving this problem.

1. INTRODUCTION

A central problem in the area of computer graphics is the hidden-line problem. A reasonable amount of research work has been devoted to developing computer programs to solve this problem in the two- and three-dimensional cases [5].

Shamos [6] proposed an $O(n)$ algorithm for the two-dimensional problem, but a counterexample [3] can be constructed. Two other algorithms with nonlinear time [3] have been proposed [2, 4].

In this paper, we present and prove the correctness of a new linear time algorithm for the two-dimensional problem.

2. DEFINITIONS AND PRELIMINARY RESULTS

A *simple polygon* P is a simply connected subset of the plane whose boundary is a closed chain of line segments, with no two nonadjacent intersecting edges. Such polygons will be represented by a sequence of vertices $\sigma_1,\sigma_2,\ldots,\sigma_n$, where σ_i has real-valued x- and y-coordinates (x_i,y_i). The *interior* of the polygon lies to the right as the edges are traversed. For ease of reference, we describe the edge $\sigma_i\sigma_{i+1}$, indices taken modulo n, by e_i.

Let y and z be two points inside P. y is said to be *visible* from z, if the line segment yz lies entirely inside P. In general, let y and z lie in the same plane with a simple chain Q $q_1,\ldots,q_m$. y and z are visible with respect to chain Q, if the line segment yz does not intersect any line segment of the chain Q.

Let x be a point inside P. The *visibility polygon* from x, denoted by $V(P,x)$, is the set of points in P visible from x, i.e.,

$$V(P,x) = \{z \mid z \in P \text{ and } xz \cap P = xz\}.$$

Given three vertices $\sigma_j = (x_j,y_j), \sigma_i = (x_i,y_i), \sigma_k = (x_k,y_k)$, let

$$T = (x_k - x_i)(y_j - y_i) - (y_k - y_i)(x_j - x_i).$$

We say that $\sigma_j\sigma_i\sigma_k$ is a left (right, no) turn, if T is positive (negative, zero).

Let z be a point in the plane. Form a polar coordinate system with point z as the origin and the negative x-axis as the reference. The polar angle of a point y around z will be denoted by THETA(y,z), and the polar angle around z increases in clockwise direction.

To simplify both the description and the analysis of the algorithm, we assume that no three vertices are collinear.

LEMMA 1. *If $x\sigma_i\sigma_{i+1}$ is a left turn, the entire edge, e_i, is not visible from x.*

Proof. Construct the triangle TR $= (x,\sigma_i,\sigma_{i+1})$. The exterior of the polygon lies to the left as the edge e_i is traversed, and therefore, inside the triangle TR. So, the entire edge is not visible.

LEMMA 2. *If σ_i and σ_{i+1} are visible from x, and $x\sigma_i\sigma_{i+1}$ is a right turn, the entire edge, e_i, is visible from x.*

Proof. Construct the triangle TR $= (x,\sigma_i,\sigma_{i+1})$. Since σ_i and σ_{i+1} are visible from x, the line segments $x\sigma_i$ and $x\sigma_{i+1}$ lie completely inside P. The interior of the polygon lies to the right as the edge e_i is traversed, and therefore, inside the triangle TR. So, the entire triangle lies in P and consequently the edge e_i is visible.

Obviously, if two points on an edge satisfy conditions of Lemma 2, the portion of the edge between them is visible from x.

LEMMA 3. *Let $Q = (q_1,q_2,\ldots,q_m)$ be a simple chain such that q_1 and q_m are visible with respect to the chain Q. If a point y is exterior to the region enclosed by (Q,q_1) and collinear with q_1 and q_m, the region enclosed by (Q,q_1) is not visible from y.*

Proof. Let z be any point interior to the region enclosed by (Q,q_1). Then the line segment zy intersects (Q,q_1) at least once. As yz cannot intersect q_1q_m, it intersects a line segment of the chain Q. So, y and z are not visible with respect to the chain Q and similarly all points in that region. The line segment q_1q_m is said to be the *window* of y on the region enclosed by (Q,q_1).

*Research supported in part by NSERC Grant A3013.

0196-6774/81/020186-12$02.00/0

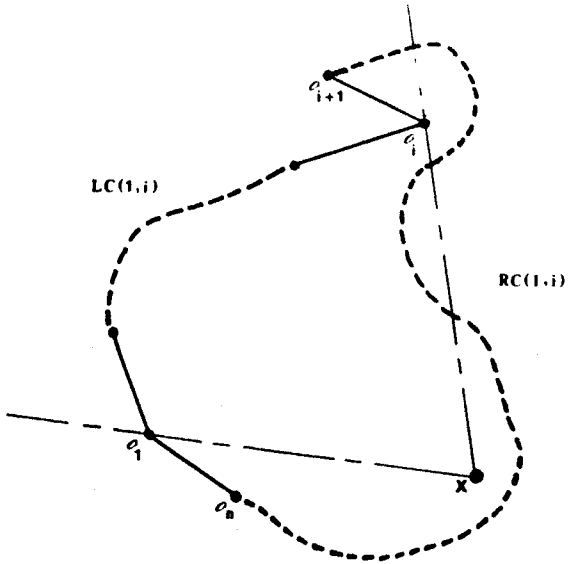


FIGURE 1a

Before presenting the next two lemmas we introduce some further notation. Between any two vertices σ_u and σ_i of P there exist two chains: the left chain LC(u,i) and the right chain RC(u,i). In LC(u,i) the interior of P lies to the right as the edges are traversed from σ_u to σ_i, whereas in RC(u,i) the interior of P lies to the left. Let the subsequence $S = (\sigma_{i_1},\sigma_{i_2},\ldots,\sigma_{i_j},\sigma_i)$ be the visible portion of LC(u,i) from x with respect to LC(u,i).

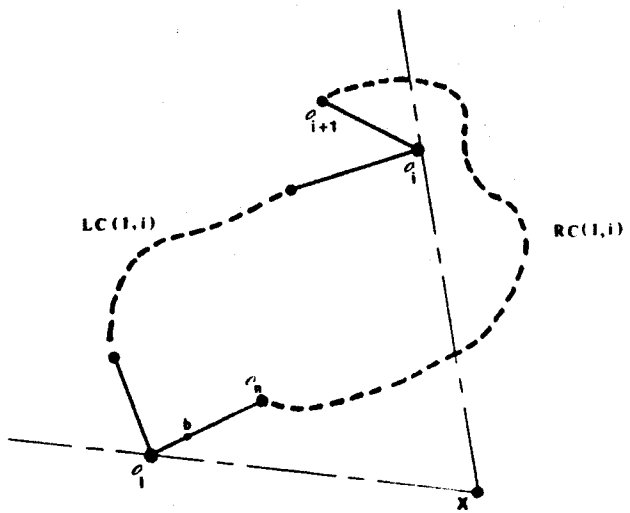


FIGURE 1b

LEMMA 4. *If σ_i is visible from x and both $\sigma_i\sigma_{i+1}$ + $x\sigma_i\sigma_{i+1}$ are left turns, RC$(1,i)$ intersects the half line $x\sigma_i$-extended an odd number of times (intersection at σ_i not included).*

Proof. A half line starting from x intersects the edges of P an odd number of times, as x lies in P. If RC$(1,i)$ intersects the line segment $x\sigma_i$ an even number of times, the lemma is true.

Therefore we assume RC$(1,i)$ intersects the line segment $x\sigma_i$ an odd number of times. The conditions of the lemma imply that σ_{i+1} lies outside of the region $R = (x, \text{LC}(1,i), x)$. Now, edge e_n must intersect the interior of R. Otherwise, since RC$(1,i)$ cannot intersect $x\sigma_1$ or LC$(1,i)$, it would have to intersect the segment $x\sigma_i$ an even number of times (refer to Fig. 1a), contradicting our assumption. Construct the triangle TR $= (x,b,\sigma_1)$, Fig. 1b, with b arbitrarily close to σ_1. As in Lemma 1, the triangle TR contains the exterior of P, and therefore, the line segment $x\sigma_1$ lies outside P, which contradicts the visibility of σ_1. Thus the lemma follows.

So, RC$(1,i)$ intersects the half line $x\sigma_i$-extended an odd number of times, and intersects the line segment $x\sigma_i$ an even number of times, if any.

LEMMA 5. *If σ_1 is visible from x and both $x\sigma_i\sigma_{i+1}$ and $\sigma_i\sigma_i\sigma_{i+1}$ are left turns, the portion of RC$(1,i)$ to the left of the half line $x\sigma_i$-extended is not visible from x.*

Proof. By Lemma 4, RC$(1,i)$ intersects the half line $x\sigma_i$-extended an odd number of times. Let k be the intersection point closest to σ_i. As the chain $(\sigma_i,\sigma_{i+1},\ldots,k)$ and point x satisfy the conditions of Lemma 3, the region enclosed by the chain $(\sigma_i,\sigma_{i+1},\ldots,k,\sigma_i)$ is not visible from x.

3. THE ALGORITHM

Input. A circular linked list P

$$\sigma_1, \text{succ}(\sigma_1), \text{succ}(\text{succ}(\sigma_1)), \ldots, \sigma_n$$

containing the x- and y-coordinates of the vertices in clockwise order, and a point x inside P.

Method. The algorithm examines the vertices of P in clockwise order, updating three stacks. Let σ_1 be the starting point. After scanning σ_i, the stacks are as follows.

Stack VP contains the portion of LC$(1,i)$ visible from x with respect to LC$(1,i)$. The points in *VP* are in clockwise order about x. Upon termination of the algorithm, *VP* contains the visibility polygon, $V(P,x)$.

Stack L contains windows of x on hidden regions generated by LC(1,i). The windows are in clockwise order about x. These windows are visible from x with respect to LC(1,i) and are also included in VP.

Stack R contains windows of x on hidden regions generated by LC(1,i). The windows are in anticlockwise order about x. These windows are not visible from x.

We now introduce the procedures VIS-POL and REGIONAL to compute the visibility polygon, $V(P,x)$.

Procedure VIS-POL

Preprocessing Step

Compute the intersection points of the negative x-axis, starting at x, with the edges of P.

$STR \leftarrow$ the closest intersection to x
if STR is not a vertex **then** insert STR into the sequence of vertices representing P.

Initialization Step

(*Comments*: Point INF has the coordinates $(-\infty, y_x)$; ***STR-INF*** is the window of x on the exterior of P behind the edge $\sigma_{STR}\sigma_{SUCC(STR)}$, and its polar angle about x is 2π.)

PUSH(VP) $\leftarrow$ STR; PUSH(VP) $\leftarrow$ SUCC(STR)
PUSH(R) $\leftarrow$ STR-INF
$t \leftarrow STR$

General Step

$u \leftarrow$ SUCC(t); $\sigma \leftarrow$ SUCC(u)

Case 0. $\sigma = STR$.
STOP

Case 1. $xu\sigma$ is a right turn.

(*Comment*: Determine if σ lies in a hidden region associated with a window in R.)

$rs \leftarrow$ POP(R)
While $R \neq \emptyset$ and THETA(σ,x) > THETA(r,x) and $u\sigma$ does not intersect rs
do $rs \leftarrow$ POP(R)

Case 1a. $u\sigma$ intersects rs at point k.

SCAN the vertices of P from σ until an edge, e_i, intersects rk at a point l.
SUCC(u) $\leftarrow k$; SUCC(k) $\leftarrow l$; SUCC(l) $\leftarrow \sigma_{i+1}$;
PUSH(VP) $\leftarrow k$; PUSH(VP) $\leftarrow l$;
PUSH(R) $\leftarrow rl$;
$t \leftarrow k$

Case 1b. THETA(σ,x) $\leq$ THETA(r,x).

(*Comment*: Vertex σ is not in a hidden region.)

PUSH(VP) $\leftarrow \sigma$
PUSH(R) $\leftarrow rs$
$t \leftarrow$ SUCC(t)

Case 2. $xu\sigma$ is a left turn and $tu\sigma$ is a left turn.

SCAN the vertices of P from σ until an edge, e_i, intersects xu-extended in k.
SUCC(u) $\leftarrow k$; SUCC(k) $\leftarrow \sigma_{i+1}$
PUSH(VP) $\leftarrow k$
PUSH(L) $\leftarrow uk$
$t \leftarrow u$

Case 3. $xu\sigma$ is a left turn and $tu\sigma$ is a right turn.

(*Comment*: Determine if vertex σ is in a hidden region associated with the window in L.)

if $L \neq \emptyset$ **then** $rs \leftarrow$ POP(L)
else $rs \leftarrow \emptyset$
While $L \neq \emptyset$ and THETA(σ,x) < THETA(r,x) and $u\sigma$ does not intersect rs
do $rs \leftarrow$ POP(L)
if $L = \emptyset$ and THETA(σ,x) < THETA(r,x) and $u\sigma$ does not intersect rs
then $rs \leftarrow \emptyset$

Case 3a. $u\sigma$ intersects rs at point k.

SCAN the vertices of P from σ until an edge, e_i, intersects rk at a point l.
$w \leftarrow$ POP(VP)
While $w \neq s$ **do** $w \leftarrow$ POP(VP);
SUCC(r) $\leftarrow l$, SUCC(l) $\leftarrow \sigma_{i+1}$
PUSH(VP) $\leftarrow l$
PUSH(L) $\leftarrow rl$
$t \leftarrow r$

Case 3b. $rs = \emptyset$ or $\mathrm{incl}(\sigma,x) \geq \mathrm{incl}(r,x)$.

(*Comment*: Vertex σ is not in a hidden region.)

If $rs \neq \emptyset$ then PUSH(L) $\leftarrow rs$
$r \leftarrow$ POP(VP); $s \leftarrow$ POP(VP);
While rs does not intersect $x\sigma$-extended
do $r \leftarrow s$; $s \leftarrow$ POP(VP)
let k be the found intersection point
SUCC(s) $\leftarrow k$
PUSH(VP) $\leftarrow s$; PUSH(VP) $\leftarrow k$.
$t \leftarrow k$
call REGIONAL

end

Procedure REGIONAL

(*Comment*: This procedure is called when the previous checks are not sufficient to find whether the next vertex lies in a hidden region or not.)

$y \leftarrow$ SUCC(σ)

Case a. $x\sigma y$ is a left turn.

SUCC(k) $\leftarrow \sigma$
PUSH(VP) $\leftarrow \sigma$

Case b. $x\sigma y$ is a right turn and $u\sigma y$ is a left turn.

SUCC(k) $\leftarrow \sigma$
PUSH(VP) $\leftarrow \sigma$
PUSH(L) $\leftarrow \sigma k$

Case c. $x\sigma y$ is a right turn and $u\sigma y$ is a right turn.

(*Comment*: Vertex y lies in a hidden region)

SCAN vertices of P from y until an edge, e_i, intersects σk at l.
SUCC(k) $\leftarrow l$; SUCC(l) $\leftarrow \sigma_{i+1}$.
PUSH(VP) $\leftarrow l$
PUSH(R) $\leftarrow \sigma l$

end

4. Analysis of the Algorithm

In this section, the performance of the algorithm is analyzed. First, the correctness of the algorithm is shown by Lemma 6; then the linearity is proved.

Lemma 6. *Prior to the execution of the general step, the following conditions are satisfied.*

(a) VP contains the portion of the visible edges [illegible] *respect to the edges scanned so far.*

(b) L *and* R *contain the windows of* x *on hidden region created by the edges scanned so far.*

(c) *Elements in* L, *if any, are in clockwise order about* x, *and elements in* R *are in anticlockwise order about* x.

Proof. By induction. In the preprocessing step, a visible vertex STR is found or generated. After the initialization step,

$$VP = (STR, \mathrm{SUCC}(STR)), \qquad L = \emptyset, \qquad R = (STR\text{-}INF)$$

and conditions (a)–(c) are obviously satisfied. Prior to execution of the general step, LC(STR,u) is the chain examined so far. Assume conditions (a)–(c) are satisfied for

$$VP = (z_1, z_2, \ldots, t, u),$$

$$L = (l_1, l_2, \ldots),$$

$$R = (r_1, r_2, \ldots),$$

and σ is the next vertex (succ(u)), to be examined. Consider each case separately.

Case 1. Elements of R satisfying the While statement conditions are entirely hidden by the edge $u\sigma$, so they are redundant and can be deleted. As elements are deleted from the top of the stack, the order of the remaining elements is not changed.

Case 1a. Let the hidden region associated with rs be enclosed by the chain (Q,r,s).

The portion $k\sigma$ of the edge $u\sigma$ is hidden. The algorithm scans the edges of P until an intersection with rk is encountered at l. The chain $(k,\sigma,\ldots,\sigma_i,l)$ is hidden. By Lemma 3, the region enclosed by the chain $(Q,r,l,\sigma_i,\ldots,\sigma,k,s)$ is hidden, and its window rl is added to R.

Case 1b. Vertex σ is not hidden. By Lemma 2, the edge $u\sigma$ is visible with respect to (LC(STR,u),σ) and is added to VP.

Case 2. The algorithm SCANS the edges of P until an intersection with xu-extended is encountered at k. By Lemma 5, the chain $(u,\sigma,\ldots,\sigma_i,k)$ is not visible. By Lemma 3, the region bounded by the chain $(u,\sigma,\ldots,\sigma_i,k,u)$ is hidden. As u is the top element in VP, which is kept in clockwise order, elements in stack L remain in clockwise order after the window uk is added.

Case 3. Elements of L satisfying the While statement conditions are entirely blocked by the edge $u\sigma$, so they can be deleted as in case 1.

chain (Q,r,s).

The portion $k\sigma$ of the edge $u\sigma$ is hidden. The algorithm scans the edges of P until an intersection with rk is encountered at l. By Lemma 3, the region bounded by the chain $(Q,r,l,\sigma_i,\ldots,s)$ is hidden, and its window rl is added to L. The hidden vertices

$$s,\mathrm{succ}(s),\ldots,\sigma_i$$

are deleted from VP.

Case 3b. By Lemma 3, the region bounded by the chain $(\sigma,k,r,\ldots,u,\sigma)$ is hidden. The hidden vertices

$$r,\mathrm{succ}(r),\ldots,u$$

are deleted from VP.

At this stage, the previously used checks are not sufficient to decide whether the next vertex y (succ(σ)) exists in the generated hidden region or not. We use the line through $x\sigma$ and the line segment $u\sigma$ to define three regions, as shown in Fig. 2.

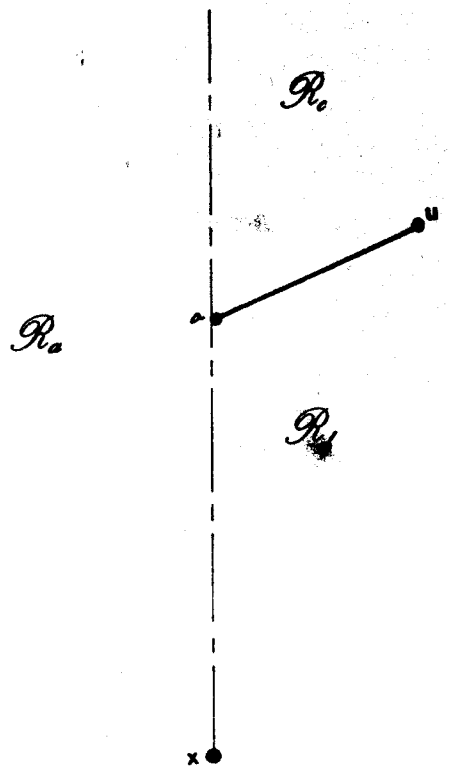


FIGURE 2

Region a. The vertex y does not lie in the generated hidden region. Vertex σ is add to VP, and return to procedure VIS-POL.

Region b. Vertex y lies outside the generated hidden region, vertex σ is added to VP. As the top element of L lies to the left of the line through $x\sigma$, the addition of the window σk will maintain the clockwise order about x.

Region c. Let the generated hidden region be enclosed by the chain (Q,σ,k).

The vertex y is hidden. The algorithm scans the edges of P until an intersection with σk is encountered at l. By Lemma 3, the region bounded by the chain $(Q,\sigma,y,\ldots,\sigma_i,l,k)$ is hidden, and also the region enclosed by $(\sigma,y,\ldots,\sigma_i,l,\sigma)$ is hidden.

THEOREM . *Procedures* VIS-POL *and* REGIONAL *compute the visibility polygon in linear time.*

Proof. When the algorithm examines the visibility of a vertex, it pushes at most two new vertices into VP, and one new element into L or R. The general step is executed at most n times. So, VP cannot have more than $2n$ elements, and both L and R can have at most n elements.

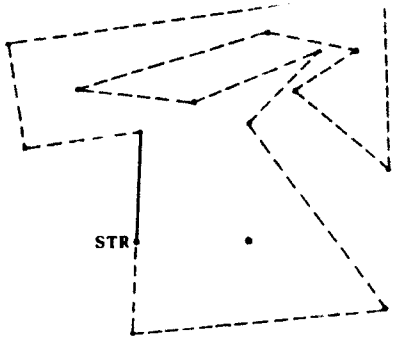
If a point is found to be hidden, it is never reexamined. So, over the whole execution of the algorithm, the While loops in cases 3a, 3b will be executed less than $2n$ times, and each While loop in cases 1,3 will be executed at most n times. Obviously the SCAN operation in the two procedures will be executed at most n times.

So, the algorithm runs in time $O(n)$.

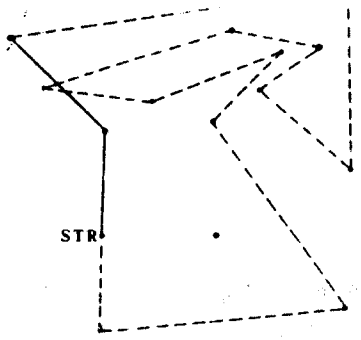
The algorithm has been programmed in FORTRAN and is available from the authors.

5. ILLUSTRATIVE EXAMPLE

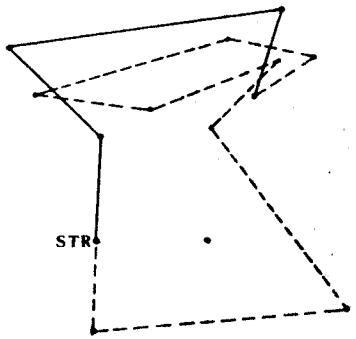
In this section, an example is given to demonstrate the performance of the algorithm in its different cases. The vertices are relabeled such that $\sigma_1 = \sigma_{\mathrm{SUCC}(STR)}$. The portions of edges stored in VP are shown in a continuous line, and the unscanned edges are shown in broken lines.



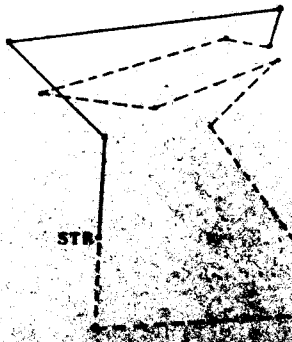
a. Initialization step

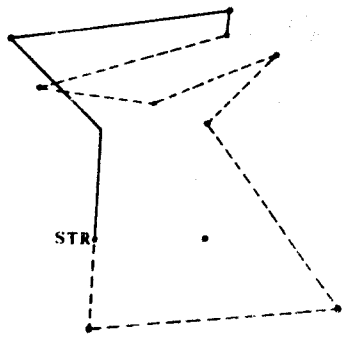


b. Case 2

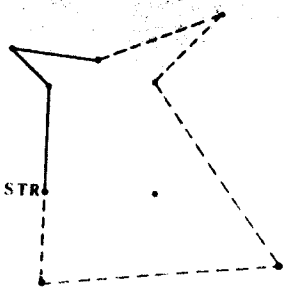


c. Case 3b

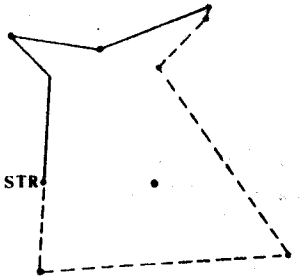




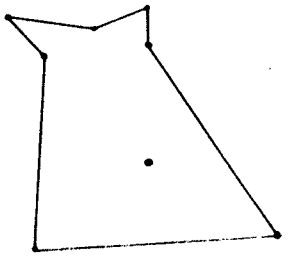
e. Case 3b



f. Case 3a



g. Case 1a



h. Visibility Polygon

6. Conclusion

We have introduced a linear time algorithm for computing the visibility polygon from a given point inside the polygon. Using the same method of storing windows on the generated hidden regions in sorted order around the given point, we can also find the visible, or partially visible, edges in the free exterior and blocked exterior configurations [3]. A related problem is to check the visibility of a simple planar polygon from a given set of points inside it. A group of algorithms have been presented in [3] to answer this question. An interesting question would be to find a minimal set of points from which the complete polygon is visible. Another related problem has been presented in [1]. In this paper the question of visibility from an edge, rather than a point, is addressed.

Acknowledgment

The authors gratefully acknowledge many useful discussions with Godfried Toussaint.

References

1. D. Avis and G. T. Toussaint, An optimal algorithm for determining the visibility of a polygon from an edge, *IEEE Trans. Comput.*, in press.
2. L. S. Davis and M. L. Benedikt, Computational models of space: Isovists and Isovist Fields, *Computer Graphics Image Processing* **11** (1979), 49–72.
3. H. El Gindy, "Visibility in Polygons with Applications," M.Sc. thesis, McGill University, 1980.
4. H. Freeman and P. P. Loutrel, An algorithm for the solution of the two dimensional hidden line problem, *IEEE Trans. Electron. Comput.* **EC-16**, No. 6, 784–790.
5. J. G. Griffiths, Bibliography of hidden-line and hidden-surface algorithms, *Computer Aided Design* **10**, No. 3 (1978), 203–206.
6. M. I. Shamos, "Problems in Computational Geometry," Carnegie-Mellon University, 1975; revised, 1977.