BIT 27 (1987), 458–473

CORRECTIONS TO LEE'S VISIBILITY POLYGON ALGORITHM

B. JOE and R. B. SIMPSON

Dept. of Computing Science,
University of Alberta,
Edmonton, Alberta,
Canada T6G 2H1

Dept. of Computer Science,
University of Waterloo,
Waterloo, Ontario,
Canada N2L 3G1

Abstract.

We present a modification and extension of the (linear time) visibility polygon algorithm of Lee. The algorithm computes the visibility polygon of a simple polygon from a viewpoint that is either interior to the polygon, or in its blocked exterior (the cases of viewpoints on the boundary or in the free exterior being simple extensions of the interior case). We show by example that the original algorithm by Lee, and a more complex algorithm by El Gindy and Avis, can fail for polygons that wind sufficiently. We present a second version of the algorithm, which does not extend to the blocked exterior case.

CR Categories: F.2.2.

Keywords: computational geometry, visibility.

1. Introduction.

We consider the following problem: Given a simple polygon P and a viewpoint z in the plane, find all points on the boundary of P that are 'visible' from z. The position of the viewpoint z determines three cases: z in the interior or exterior of P, or on the boundary of P. The exterior case can be further categorized as free exterior or blocked exterior depending on whether there exists a ray from z which does not intersect P or there is no such ray. The boundary and free exterior cases can be handled as simple extensions of the interior case. So from an algorithmic viewpoint, there are basically two cases, the interior case and the blocked exterior case.

Two linear time algorithms for computing the visibility polygon of P from an interior viewpoint have been published, Lee [7] and El Gindy and Avis [1], and Lee also presented a modification for the blocked exterior case in his paper. Lee's algorithm is simpler in structure; in particular it requires only one stack which eventually yields the visibility polygon, as opposed to three in the El

This work was partially supported by grants from the Central Research Fund of the University of Alberta and the Natural Sciences and Engineering Research Council of Canada.

Received February 1987. Revised July 1987.

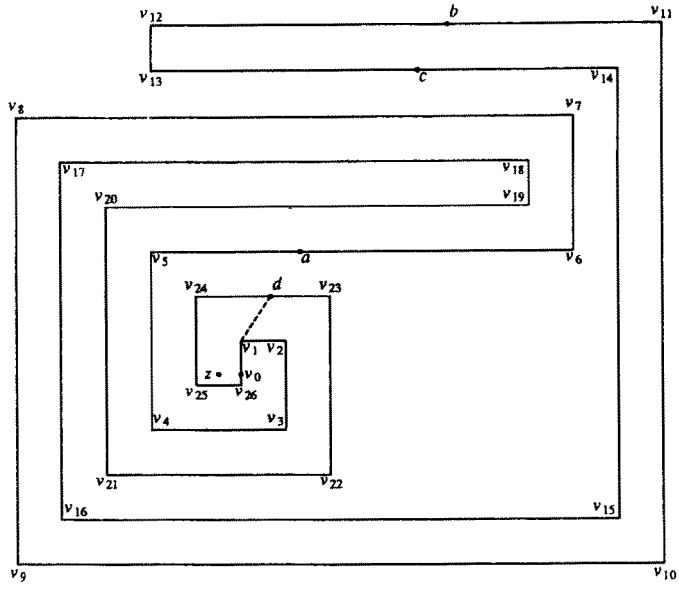

Fig. 1. Example for which the algorithms of Lee and El Gindy-Avis fail for an interior viewpoint z. The labels of vertices are in the exterior of the polygon. $Bd(V(P, z)) = v_0v_1dv_{24}v_{25}v_{26}v_0$; v_1 and v_7 are CCW-oriented points, v_2 and v_6 are CW-oriented points.

Gindy and Avis algorithm. However, all these algorithms can fail for polygons which wind sufficiently and examples of polygons for which they fail are given in Figures 1 and 2. The failure of Lee's algorithm is a technicality that can be readily corrected, once its origins are understood; the failure of the El Gindy and Avis algorithm seems far more fundamental and we do not believe it can be corrected.

In this paper, we present a modification of Lee's algorithm which handles both interior and blocked exterior cases, and a second modification that handles only the interior case (the boundary and free exterior cases being immediately accessible from the interior case). The correctness of these algorithms is proved in Joe and Simpson [6]. For a viewpoint z in the interior or exterior of P, this problem was originally called the two-dimensional hidden line elimination problem in computer graphics [2]. Our interest in this problem arose in the application of finite element triangulation [5], where z is a reflex vertex on the boundary of P and an extended version of the method of Schachter [8] is used to decompose a simple polygon into convex polygons.

In Section 2, we give our notation and definitions. In Section 3, we present pseudocode for the two modified versions of Lee's algorithm and mention where the algorithms of Lee and El Gindy and Avis are incorrect. The viewpoint is assumed to be interior or blocked exterior in Section 3. In Section 4, we describe how the algorithms can also be used for boundary and free exterior

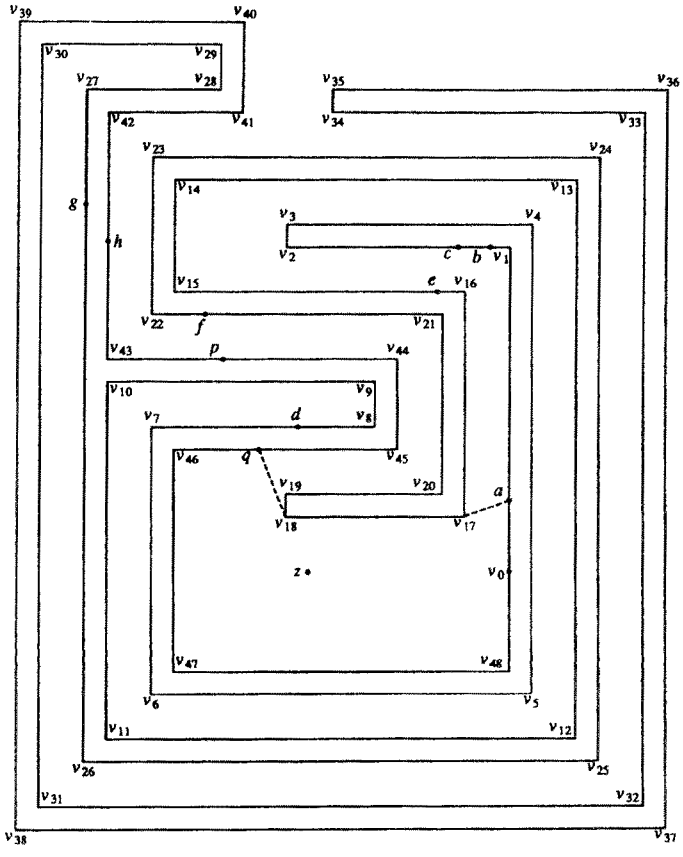

Fig. 2. Example for which Lee's modified algorithm fails for a blocked exterior viewpoint z. The labels of vertices are in the exterior of the polygon. $Bd(V(P, z)) = v_0 a v_{17} v_{18} q v_{46} v_{47} v_{48} v_0$; v_1 and v_{18} are CCW-oriented points, v_4 and v_8 are CW-oriented points.

viewpoints. In Appendix A, we illustrate the algorithms for the examples in Figures 1 and 2.

2. Notation and definitions.

We first present our notation for line segments, rays, and chains. Let u and v be distinct points. The line segment joining u and v is denoted by uv, (uv), $(uv]$, or $[uv)$ depending on whether both endpoints are included, both endpoints are excluded, only u is excluded, or only v is excluded, respectively. The ray originating at u and going through v is denoted by $\overrightarrow{uv}$.

A chain of connected line segments $u_0u_1, \cdots, u_{m-1}u_m$ is denoted by $u_0u_1 \cdots u_m$. The chain is open if $u_0 \neq u_m$ and closed if $u_0 = u_m$. The chain is simple if it does not intersect itself, with the exception that $u_0 = u_m$ is allowed. If chain $C = u_0u_1 \cdots u_m$ is closed and simple, we define $Left(C)$ and $Right(C)$ to be the

regions to the left and right, respectively, of the chain as it is traversed from u_0 to u_m, and we define $Int(C)$ and $Ext(C)$ to be the bounded interior and unbounded exterior regions, respectively, determined by the chain.

Let u, v and w be distinct points. The chain uvw is said to be a *left turn* or *right turn* if w is to the left of or right of, respectively, the directed line from u to v; and uvw is said to be a *backward move* or *forward move* if u, v, w are collinear, and $[uv)$ and $(vw]$ overlap or do not overlap, respectively.

The boundary of the simple polygon P is a simple closed chain. We denote the boundary, interior, and exterior of P by $Bd(P)$, $Int(P)$, and $Ext(P)$, respectively, so that $P = Bd(P) \cup Int(P)$. If the viewpoint $z \in Ext(P)$, then z is said to be *blocked exterior* to P if all rays emanating form z intersect $Bd(P)$, and z is said to be *free exterior* otherwise. Without loss of generality, we assume that the coordinate system is translated so that z is at the origin. We denote the polar angle of a point v by $\theta(v)$ where $0 \leq \theta(v) < 2\pi$.

Let $C = u_0 u_1 \cdots u_m$ be a simple chain which does not intersect z. We define the *angular displacement*, $\alpha(u_i)$, of u_i with respect to z as follows: $\alpha(u_0) = \theta(u_0)$; for $1 \leq i \leq m$, $\alpha(u_i) = \theta(u_i) + 2\pi k$ where k is the integer determined so that $|\alpha(u_i) - \alpha(u_{i-1})| < \pi$. The definition of angular displacement can be extended to any point on the chain C. Note that the angular displacement measures the 'winding' of the chain in addition to the polar angle. We define the *angular displacement* of the chain C to be $\delta(C) = \alpha(u_m) - \alpha(u_0)$. If the chain is closed (i.e. $u_0 = u_m$), then from classical complex integration results ([3], Section 4.6),

$$\delta(C) = \begin{cases} 0 & \text{if } z \in Ext(C) \\ 2\pi & \text{if } z \in Int(C) = Left(C) \\ -2\pi & \text{if } z \in Int(C) = Right(C). \end{cases}$$

Freeman and Loutrel [2], who presented a nonlinear-time algorithm for finding the visibility polygon, used the term *total angle* for angular displacement.

If $z \in P$, then the point v is said to be *visible* from z with respect to P if (zv) is entirely in $Int(P)$. If $z \in Ext(P)$, then the point v is said to be *visible* from z with respect to P if (zv) is entirely in $Ext(P)$. The problem we are considering is to find a subset $\bar{V}(P, z)$ of points on $Bd(P)$ which are visible from z. An equivalent problem is to find the star-shaped simple polygon, $V(P, z)$, the visibility polygon from z, which is the closure of the set $\{u | u \in zv \text{ and } v \in \bar{V}(P, z)\}$. Note that some points on $Bd(V(P, z))$ are not visible from z, according to our definition, but they are included to enable $Bd(V(P, z))$ to be a simple closed curve (see Figures 1 and 2).

In the rest of this paper, except for Section 4, we assume that $z \in Int(P)$ or z is blocked exterior to P. Note that in both of these cases, all rays emanating from z intersect $Bd(P)$, so there exists exactly one visible boundary point at each polar angle. In Section 4, we will describe how the algorithms given in Section 3 can also be used for boundary and free exterior viewpoints.

If $z \in Int(P)$, we orient the vertices of P in counterclockwise order ($Int(P)$ is to the left as $Bd(P)$ is traversed), and label them $v_0, v_1, \cdots, v_{n-1}$, and $v_n = v_0$, where v_0 is the point on $Bd(P)$ which is on the positive x-axis and has the smallest x-coordinate. If z is blocked exterior to P, we orient the vertices of P in clockwise order ($Int(P)$ is to the right as $Bd(P)$ is traversed), and label them $v_0, v_1, \cdots, v_{n-1}$, and $v_n = v_0$, where v_0 is the point on $Bd(P)$ which is on the positive x-axis and has the smallest x-coordinate. Note that v_0 is visible from z in both cases. We consider v_0 and v_n to be 'distinct' points with v_0 on edge v_0v_1 and v_n on edge $v_{n-1}v_n$.

Let s and t be points on $Bd(P)$. Then s is said to *occur* before t if s appears before t in a traversal of $Bd(P)$ from v_0 to v_n. If s occurs before t then we denote the chain of $Bd(P)$ from s to t by $Ch[s,t]$, $Ch(s,t)$, $Ch(s,t]$, or $Ch[s,t)$ depending on whether both s and t are included, both s and t are excluded, only s is excluded, or only t is excluded, respectively. Note that $Ch[v_0, v_n] = Bd(P)$.

Let v be a point on $Bd(P)$. If $z \in Int(P)$ [alternatively $Ext(P)$], we define v to be

(a) *CCW-oriented* if v lies on an edge which is oriented counterclockwise (forms a left turn) with respect to z and zv contains a nonempty subsegment $[uv)$ which is entirely in $Int(P)$ [$Ext(P)$],

(b) *CW-oriented* if v lies on an edge which is oriented clockwise (forms a right turn) with respect to z and zv contains a nonempty subsegment $[uv)$ which is entirely in $Ext(P)$ [$Int(P)$],

(c) *CL-oriented* if v lies on an edge which is collinear with z and zv contains a nonempty subsegment $[uv)$ which is entirely on $Bd(P)$.

See Figures 1 and 2 for examples of CCW-oriented and CW-oriented points.

From the definition of 'visible' and the orientation of $Bd(P)$, we have the following two facts.

Fact 1: If (zv) intersects $Bd(P)$, then v is not visible from z.

Fact 2: If v is CW-oriented or CL-oriented, then v is not visible from z.

3. The algorithms.

In this section, we present the two modified versions of Lee's algorithm. We refer to them as Algorithm 1 and Algorithm 2. Algorithm 1 is the version which works correctly for both interior and blocked exterior viewpoints. Algorithm 2 is the version which works correctly for interior viewpoints only. To reduce the number of cases in the algorithms and the correctness proof, we make the simplifying assumption that no two vertices v_i and $v_j, 0 \le i < j < n$, have the same polar angle with respect to z (Lee [7] and El Gindy and Avis [1] also make this assumption). It is straightforward to add the extra cases to the algorithms when this assumption is removed. This is done in [4] for Algorithm 2.

Algorithms 1 and 2 carry out a sequential scan of $Bd(P)$ starting from edge

v_0v_1 and ending at edge $v_{n-1}v_n$, while manipulating a stack of boundary points $s_0, s_1, \cdots, s_t$ such that ultimately the chain $s_0s_1 \cdots s_t$ becomes $Bd(V(P,z))$. When processing the current edge v_iv_{i+1}, the operations that may be performed are:

(a) add boundary points to the stack,
(b) delete boundary points from the stack,
(c) scan edges $v_{i+1}v_{i+2}, v_{i+2}v_{i+3}, \cdots$ for the first edge v_kv_{k+1} to satisfy a certain condition.

Operation (c) is performed if v_iv_{i+1} enters a 'hidden' region where boundary points are not visible from z. There are four types of hidden regions, and the condition for exiting the scan in each of these regions is slightly different. When v_iv_{i+1} is not in a hidden region or upon exiting a hidden region with the new current edge v_iv_{i+1}, operation (a) or (b) is performed if zv_iv_{i+1} is a left turn or right turn, respectively. Note that if zv_iv_{i+1} is a left turn then the points on (v_iv_{i+1}) may be visible from z since they are CCW-oriented, and if zv_iv_{i+1} is a right turn then the points on (v_iv_{i+1}) are not visible from z since they are CW-oriented. Algorithms 1 and 2, as well as Lee's two algorithms for interior and blocked exterior viewpoints, differ only in the conditions for exiting the scan in the four types of hidden regions.

Algorithm 1 is given in the pseudocode below. It is decomposed into the six procedures LEFT, RIGHT, SCANA, SCANB, SCANC, and SCAND, along with a driver procedure VISPOL which repeatedly calls them. Procedure LEFT is called when the previous edge $v_{i-1}v_i$ is not in a hidden region, $zv_{i-1}v_i$ is a left turn, and one or two boundary points have just been added to the stack. Procedure RIGHT is called when the previous edge $v_{i-1}v_i$ is not in a hidden region, $zv_{i-1}v_i$ is a right turn, and $v_{i-1}v_i$ is in front of the tail part of the chain $s_0s_1 \cdots s_t$ so that points must be deleted from the stack. Procedure SCANA, SCANB, SCANC, or SCAND is called when the previous edge $v_{i-1}v_i$ enters one of the four hidden regions. In [6] it is proved that the edges scanned in a hidden region are not visible from z, and that there exists an edge which satisfies the condition for exit from the scan.

The viewpoint z, the number of vertices n, and the vertices $v_0, v_1, \cdots, v_{n-1}, v_n = v_0$ of P are global variables in the six procedures. They all have input/output parameters *oper*, *i*, *t*, *S*, and *w*. Here, *oper* is a string containing the name of the current or next procedure (operation) or FINISH, i is the index of the current edge v_iv_{i+1} ($Ch[v_0, v_i]$ has been processed so far), t is the index of the top stack point s_t. S is the chain of stack points $s_0s_1 \cdots s_t$ and w is a point on $v_{i-1}v_i$ which is required only in SCANC and SCAND; it is used extensively in the correctness proof in [6]. On entering LEFT, SCANA, and SCANB, $w = v_i$; on entering RIGHT, SCANC, and SCAND, w is the point on $v_{i-1}v_i$ such that $\theta(w) = \theta(s_t)$.

The pseudocode is designed so that the following properties are satisfied by the stack points $s_0, s_1, \cdots, s_t$ on entrance to each of the six procedures. In these properties, i and t are the parameters described above. In this section, $\alpha(s_j)$ denotes the angular displacement of s_j on the varying chain $s_0s_1 \cdots s_t$.

(S1) $0 \le t \le i$, $s_j \in Ch[v_0, v_i]$ for $0 \le j \le t$, $s_0 = v_0$, and s_j occurs before s_{j+1}.

(S2) $0 = \alpha(s_0) \le \alpha(s_1) \le \cdots \le \alpha(s_t) \le 2\pi$; at most two consecutive s_j have the same angular displacement.

(S3) If $\alpha(s_j) < \alpha(s_{j+1})$ then $s_j s_{j+1} \subseteq Bd(P)$. If $\alpha(s_j) = \alpha(s_{j+1})$ then $(s_j s_{j+1})$ is not on $Bd(P)$.

The various cases in the pseudocode below are labeled for reference in the figures. The statement $(oper, i, t, S, w) :=$ (arg1, arg2, arg3, arg4, arg5) used in the procedures is short for the simultaneous execution of the statements $oper :=$ 'arg1'; $i :=$ arg2; $t :=$ arg3; $S :=$ arg4; $w :=$ arg5. A '#' symbol indicates that the rest of the line is a comment.

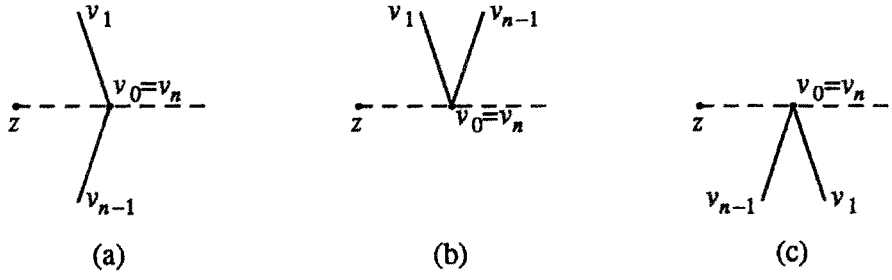

Fig. 3. Illustration of possible configurations of edges v_0v_1 and $v_{n-1}v_n$. (a) and (b) are case (V1), (c) is case (V2).

```
procedure VISPOL(z, n, P, t, V(P, z))        # Pseudocode for Algorithm 1
# Input:  Viewpoint z in Int(P) or blocked exterior to P, and vertices v0, v1, ..., vn
#         of P with labeling and orientation of vertices as described in Section 2.
# Output: Vertices s0, s1, ..., st of visibility polygon V(P, z).

# Assumption (V0): No two vertices vi, vj have the same polar angle with respect to z.

    # See Figure 3 for possible configurations of v0v1 and vn-1vn.
(V1) if zv0v1 is a left turn then
        (oper, i, t, S, w) := (LEFT, 1, 1, v0v1, v1)
(V2) else # zv0v1 is a right turn
        (oper, i, t, S, w) := (SCANA, 1, 0, v0, v1)
     endif

     repeat
        # Properties (S1), (S2), (S3) are satisfied.
        case oper of
           'LEFT'  : LEFT(oper, i, t, S, w)
           'RIGHT' : RIGHT(oper, i, t, S, w)
           'SCANA' : SCANA(oper, i, t, S, w)
           'SCANB' : SCANB(oper, i, t, S, w)
           'SCANC' : SCANC(oper, i, t, S, w)
           'SCAND' : SCAND(oper, i, t, S, w)
        endcase
```

```
(V3)    if oper = 'LEFT' and (s_{t-1}s_t) intersects zv_n^→ then
          # α(s_{t-1}) < 2π < α(s_t), zv_{i-1}v_i is a left turn, and s_t = w = v_i.
            # Replace s_t (shorten s_{t-1}s_t) so that α(s_t) = 2π.
            s_t := intersection of s_{t-1}v_i and zv_n^→
            oper := 'SCANB'
          endif
      until oper = 'FINISH'
      # Properties (S1), (S2), (S3) are satisfied, and s_t = v_n and α(s_t) = 2π.
end # VISPOL

procedure LEFT(oper, i, t, S, w)  # Previous case can be (V1), (L2), (R2), (A3), or (D1).
# (L0) zv_{i-1}v_i is a left turn, s_t = w = v_i, s_{t-1} ∈ [v_{i-1}v_i) and either α(s_t) < 2π or s_t = v_n.

      # See Figure 4 for possible locations of v_iv_{i+1}.
(L1)  if i = n then
            (oper, i, t, S, w) := (FINISH, n, t, s_0 ··· s_t, v_n)
(L2)  else if zv_iv_{i+1} is a left turn then
            (oper, i, t, S, w) := (LEFT, i+1, t+1, s_0 ··· s_t v_{i+1}, v_{i+1})
(L3)  else if zv_iv_{i+1} is a right turn and s_{t-1}v_iv_{i+1} is a right turn then
            (oper, i, t, S, w) := (SCANA, i+1, t, s_0 ··· s_t, v_{i+1})
(L4)  else # zv_iv_{i+1} is a right turn and s_{t-1}v_iv_{i+1} is a left turn
            (oper, i, t, S, w) := (RIGHT, i+1, t, s_0 ··· s_t, v_i)
      endif
end # LEFT
```

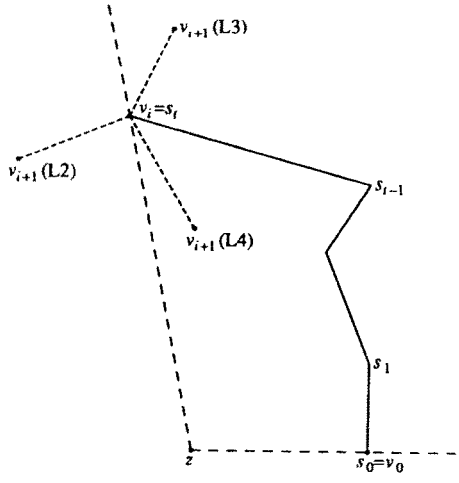

Fig. 4. Illustration of chain $S = s_0 s_1 \cdots s_t$ and edge $v_i v_{i+1}$ in cases (L2), (L3), and (L4) of LEFT.

```
procedure RIGHT(oper, i, t, S, w)  # Previous case can be (L4), (R1), (A1), (B2), or (C1).
# (RO) zv_{i-1}v_i is a right turn, zs_tv_i is a right turn, α(s_{t-1}) < α(s_t), and
#      either (i) s_t = w = v_{i-1} and s_{t-1}s_tv_i is a left turn,
#      or (ii) s_t is not on v_{i-1}v_i, w ∈ [v_{i-1}v_i), and zs_tw is a backward move.

      # See Figure 5 for possible locations of v_{i-1}v_i and v_iv_{i+1}.
      Scan s_ts_{t-1}, ···, s_1s_0 for the first edge s_js_{j-1} such that
          (RA) zs_jv_i is a right turn and zs_{j-1}v_i is a left turn, or
          (RB) zs_{j-1}s_j is a forward move and v_{i-1}v_i intersects (s_{j-1}s_j)
      Delete s_t, s_{t-1}, ···, s_{j+1} from stack
```

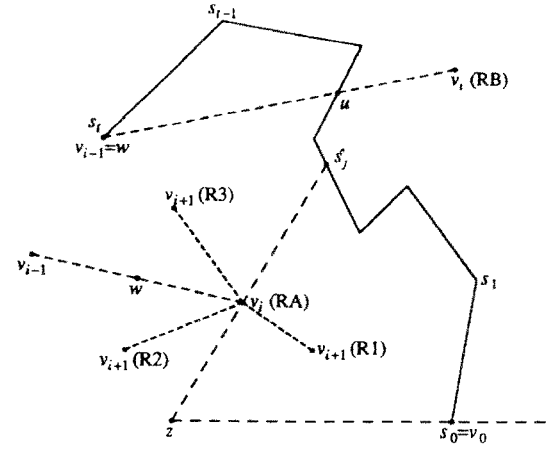

Fig. 5. Illustration of chain S, edge $v_{i-1}v_i$ in cases (RA) and (RB), and edge v_iv_{i+1} in cases (R1), (R2), and (R3) of RIGHT.

```
        if case (RA) then # zs_{j-1}s_j is a left turn
            # Replace s_j (shorten s_{j-1}s_j).
            s_j := intersection of (s_{j-1}s_j) and z→v_i
            # zs_jv_i is a backward move and (s_jv_i) is not on Bd(P).
(R1)        if zv_iv_{i+1} is a right turn then
                (oper, i, t, S, w) := (RIGHT, i+1, j, s_0 ··· s_j, v_i)
(R2)        else if zv_iv_{i+1} is a left turn and v_{i-1}v_iv_{i+1} is a right turn then
                (oper, i, t, S, w) := (LEFT, i+1, j+2, s_0 ··· s_jv_iv_{i+1}, v_{i+1})
(R3)        else # zv_iv_{i+1} is a left turn and v_{i-1}v_iv_{i+1} is a left turn
                (oper, i, t, S, w) := (SCANC, i+1, j, s_0 ··· s_jv_i)
            endif
(R4)    else # case (RB)
            u := intersection of v_{i-1}v_i and (s_{j-1}s_j)      # u ∈ (v_{i-1}v_i)
            # Delete s_j from stack.
            (oper, i, t, S, w) := (SCAND, i, j-1, s_0 ··· s_{j-1}, u)
        endif
end # RIGHT

procedure SCANA(oper, i, t, S, w)      # Previous case can be (V2) or (L3).
# (A0) zv_{i-1}v_i is a right turn, s_t = v_{i-1}, α(s_t) < 2π, and w = v_i.
#      If i > 1 then α(s_{t-1}) < α(s_t) and s_{t-1}s_tv_i is a right turn.

    # See Figure 6 for possible exit cases.
(AS) Scan v_iv_{i+1}, ···, v_{n-1}v_n for the first edge v_kv_{k+1} to intersect zs_t→
    u := intersection of v_kv_{k+1} and zs_t→      # u ∈ (v_kv_{k+1})
(A1) if zv_kv_{k+1} is a right turn and zs_tu is a backward move then
        # δ(s_tv_i ··· v_ku) = -2π
        (oper, i, t, S, w) := (RIGHT, k+1, t, s_0 ··· s_t, u)
(A2) else if zv_kv_{k+1} is a right turn and zs_tu is a forward move then
        # δ(s_tv_i ··· v_ku) = -2π
        (oper, i, t, S, w) := (SCAND, k+1, t, s_0 ··· s_t, u)
(A3) else if zv_kv_{k+1} is a left turn and zs_tu is a forward move then
        # δ(s_tv_i ··· v_ku) = 0
        (oper, i, t, S, w) := (LEFT, k+1, t+2, s_0 ··· s_tuv_{k+1}, v_{k+1})
```

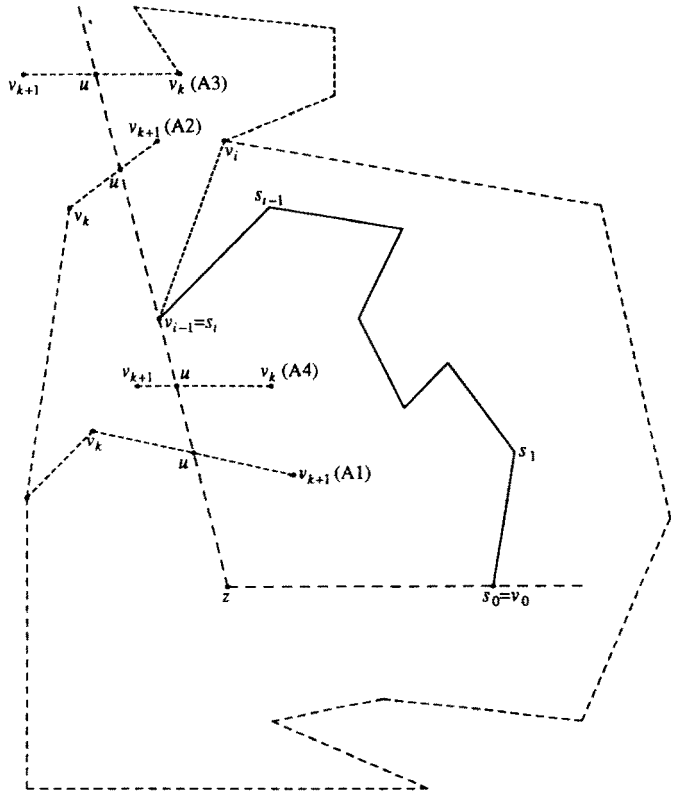

Fig. 6. Illustration of chain S and scan for first edge v_kv_{k+1} to intersect $z\vec{s_t}$ in cases (A1), (A2), and (A3) of SCANA; case (A4) is not possible. $\delta(s_tv_i\cdots v_ku)$ is -2π in cases (A1), (A2) and 0 in case (A3).

(A4) else # zv_kv_{k+1} is a left turn and zs_tu is a backward move
# This case is not possible.
endif
end # SCANA

procedure SCANB(*oper*, i, t, S, w) # Previous case is (V3).
(B0) $zv_{i-1}v_i$ is a left turn, $\alpha(s_{t-1}) < \alpha(s_t) = 2\pi$, $s_t \in (v_{i-1}v_i)$, and $w = v_i$.

See Figure 7 for possible exit cases.
(BS) Scan $v_iv_{i+1}, \cdots, v_{n-1}v_n$ for the first edge v_kv_{k+1} to intersect $(s_tv_n]$
zv_kv_{k+1} must be a right turn. $\delta(s_tv_i\cdots v_ku)$ may be 0 or -2π.
u := intersection of v_kv_{k+1} and $(s_tv_n]$
(B1) if $u = v_{k+1} = v_n$ then
(*oper*, i, t, S, w) := (FINISH, n, $t+1$, $s_0\cdots s_tv_n$, v_n)
(B2) else # $u \in (v_kv_{k+1})$
(*oper*, i, t, S, w) := (RIGHT, $k+1$, t, $s_0\cdots s_t$, u)
endif
end # SCANB

procedure SCANC(*oper*, i, t, S, w) # Previous case is (R3).
(C0) $zv_{i-1}v_i$ is a left turn, $zv_{i-2}v_{i-1}$ is a right turn, $v_{i-2}v_{i-1}v_i$ is a left turn, $w = v_{i-1}$,
s_t is not on $v_{i-1}v_i$, zs_tw is a backward move, and $\alpha(s_{t-1}) < \alpha(s_t) < 2\pi$.

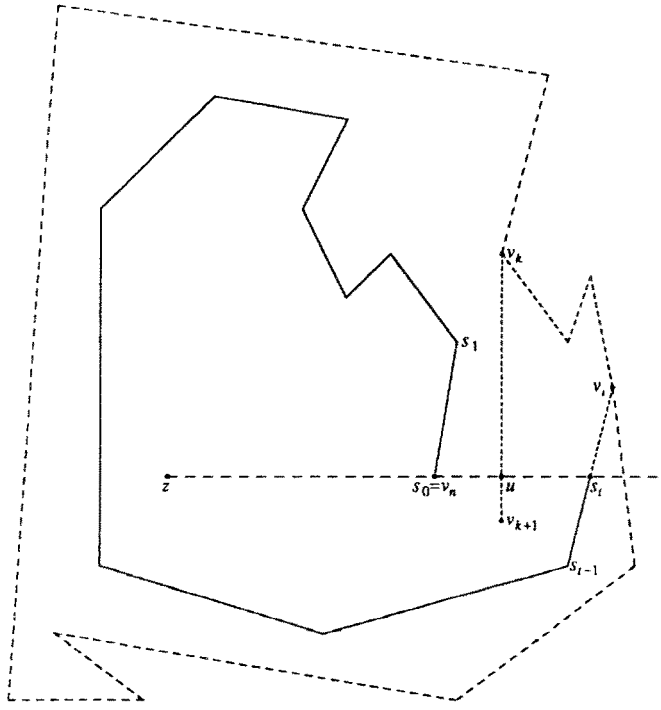

Fig. 7. Illustration of chain S and scan for first edge v_kv_{k+1} to intersect $(s_tv_n]$ in SCANB. $\delta(s_tv_i\cdots v_ku)$ may be 0 or -2π.

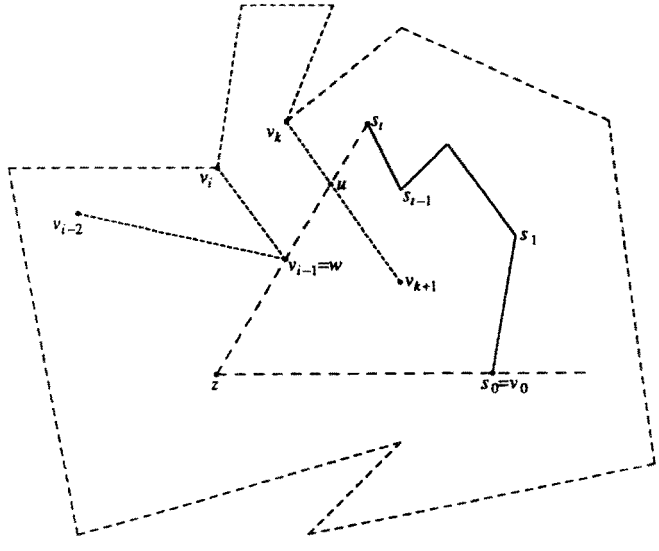

Fig. 8. Illustration of S and scan for first edge v_kv_{k+1} to intersect (s_tw) in SCANC; $\delta(wv_i\cdots v_ku)$ may be 0 or 2π.

```
        # See Figure 8 for possible exit cases.
(CS)    Scan v_i v_{i+1}, ..., v_{n-1} v_n for first edge v_k v_{k+1} to intersect (s_t w)
        # z v_k v_{k+1} must be a right turn. δ(w v_i ... v_k u) may be 0 or 2π.
(C1)    u := intersection of v_k v_{k+1} and (s_t w)      # u ∈ (v_k v_{k+1})
        (oper, i, t, S, w) := (RIGHT, k+1, t, s_0 ... s_t, u)
end # SCANC

procedure SCAND(oper, i, t, S, w)      # Previous case can be (R4) or (A2).
# (D0) z v_{i-1} v_i is a right turn, s_t is not on v_{i-1} v_i, α(s_t) < 2π, w ∈ (v_{i-1} v_i),
#      z s_t w is a forward move, and if t ≥ 1 then α(s_{t-1}) < α(s_t).

        # See Figure 9 for possible exit cases.
(DS)    Scan v_i v_{i+1}, ..., v_{n-1} v_n for the first edge v_k v_{k+1} to intersect (s_t w)
        # z v_k v_{k+1} must be a left turn. δ(w v_i ... v_k u) may be 0 or 2π.
(D1)    u := intersection of v_k v_{k+1} and (s_t w)      # u ∈ (v_k v_{k+1})
        (oper, i, t, S, w) := (LEFT, k+1, t+2, s_0 ... s_t u v_{k+1}, v_{k+1})
end # SCAND
```

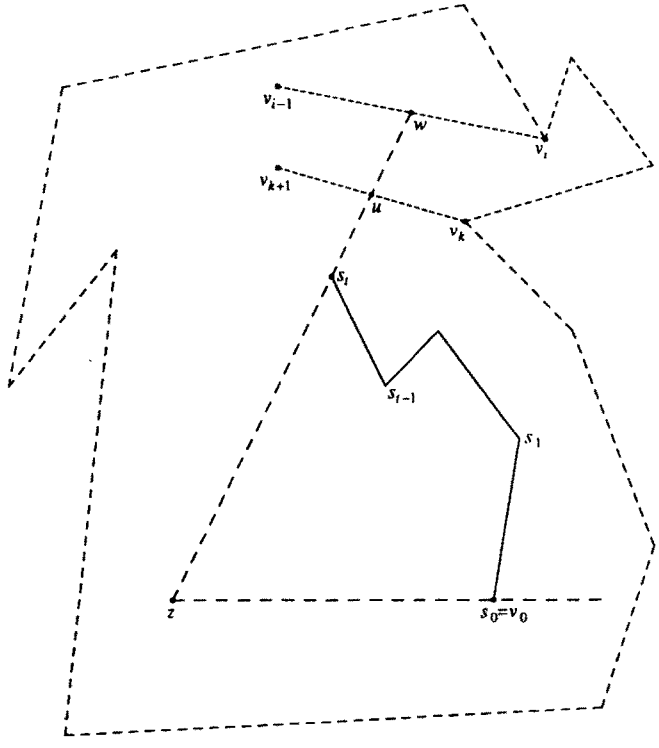

Fig. 9. Illustration of chain S and scan for first edge $v_k v_{k+1}$ to intersect $(s_t w)$ in SCAND; $\delta(w v_i \cdots v_k u)$ may be 0 or 2π.

Algorithm 2 only differs from Algorithm 1 in when the exits from the four scan statements (AS), (BS), (CS), (DS) occur. In Algorithm 2, the exits in these four statements occur at the first edge $v_k v_{k+1}$ which intersects the line segment or ray and also contains a point u such that $\delta(x v_i \cdots v_k u) = 0$, where $x = s_t$

for (AS), (BS) and $x = w$ for (CS), (DS). So it is possible that the exit does not occur at the first edge to intersect the line segment or ray, since the first intersecting point $\bar{u}$ may satisfy $\delta(xv_i \cdots v_k\bar{u}) = \pm 2\pi$ as seen in the above comments for Algorithm 1. For polygons that wind a lot, such as Figures 1 and 2, exits will occur at different points in the two algorithms. With the change to (AS), only case (A3) can occur in SCANA for Algorithm 2. Algorithm 2 does not work correctly for blocked exterior viewpoints, since the exit condition in SCANA is not guaranteed to be satisfied. In Appendix A, Algorithm 1 is illustrated for the examples in Figure 1 (interior viewpoint) and Figure 2 (blocked exterior viewpoint), and Algorithm 2 is illustrated for the example in Figure 1.

Lee's algorithm for interior viewpoints has the same exit condition in SCANA as Algorithm 2 and the same exit conditions in SCANB, SCANC, SCAND as Algorithm 1 (Lee's algorithm does not take into account that $\delta(xv_i \cdots v_k u) = \pm 2\pi$ may occur in SCANB, SCANC, SCAND). This inconsistency in the exit conditions causes his algorithm to fail for polygons that wind sufficiently; either the exit condition of a scan statement is not satisfied or the algorithm terminates with an incorrect visibility polygon. Lee's modified algorithm for blocked exterior viewpoints is based on a modification of an incorrect algorithm, so it is also incorrect for polygons that wind sufficiently. In Lee's modified algorithm, the exit condition in SCANA is the same as Algorithm 1 until the first occurrence of case (A1) or (A2), then the exit condition changes to that of his original algorithm or Algorithm 2, i.e. (A3) only; the exit conditions in SCANB, SCANC, SCAND are the same as Algorithm 1 (again, Lee's algorithm does not take into account that $\delta(xv_i \cdots v_k u) = \pm 2\pi$ may occur).

The El Gindy and Avis algorithm also has a step which is similar to the scan statement in SCANA. But this algorithm neglects the fact that the first intersection of $z\vec{s_t}$ may occur after traversing a clockwise circle of angular displacement. Therefore it also fails for polygons that wind sufficiently. We do not believe that this algorithm can be corrected since El Gindy and Avis try to maintain the property 'chain $S = s_0 \cdots s_t$ contains the boundary points visible with respect to $Ch[v_0, v_i]$', which is different from the less restrictive property 'if $v \in Ch[v_0, v_i)$ but v is not on $S = s_0 \cdots s_t$ then v is not visible from z' satisfied by Algorithms 1 and 2.

4. Extension.

In this section, we indicate how Algorithms 1 and 2 can be used for boundary and free exterior viewpoints. For a boundary viewpoint z, we orient the vertices of P in counterclockwise order and label them $z, v_0, v_1, \cdots, v_{n-1}, v_n$, and z, where v_0 is the successor vertex of z and v_n is the predecessor vertex of z. We also assume that the coordinate system is translated and rotated so that z is at the origin and v_0 is on the positive x-axis. This implies that $\theta(v_n) < 2\pi$ is the interior

angle at z. No modifications are required in the pseudocode, but case (V3) and procedure SCANB imply that $\alpha(s_t) \leq \theta(v_n)$ in property (S2). The visibility polygon $V(P, z)$ is $zs_0s_1 \cdots s_tz$ where $s_0s_1 \cdots s_t$ is the chain of stack points at the end of the algorithm. The reason why the algorithms are correct is that there are no boundary points with polar angle in the interval $(\theta(v_n), 2\pi)$ which are visible from z.

If $z \in Ext(P)$, but it is not known whether z is blocked or free exterior, then the angular displacement of the boundary vertices can be used to classify z as in [2]. Suppose the vertices $v_0, v_1 \cdots, v_{n-1}$, and $v_n = v_0$ of P are oriented in clockwise order with v_0 being an arbitrary vertex. The angular displacement of the vertices, $\alpha(v_0), \alpha(v_1), \cdots, \alpha(v_n)$, as well as their maximum and minimum, can be computed in linear time. Let $\alpha_{\max} = \max\{\alpha(v_i)\}$ and $\alpha_{\min} = \min\{\alpha(v_i)\}$. If $\alpha_{\max} - \alpha_{\min} \geq 2\pi$ then z is blocked exterior; otherwise z is free exterior.

If z is free exterior to P, then the problem of computing $V(P, z)$ can be reduced to the following equivalent problem of computing $V(Q, z)$ with z on $Bd(Q)$ as in [7]. Let v_j and v_k be the vertices closest to z with angular displacement $\alpha_{\min}$ and $\alpha_{\max}$, respectively. $Bd(P)$ can be partitioned into the front chain $F = v_jv_{j+1} \cdots v_{k-1}v_k$ and the back chain $B = v_kv_{k+1} \cdots v_{j-1}v_j$ where the indices are taken modulo n. Since the vertices of P are oriented in clockwise direction, chain F is in front of chain B, i.e. for every point $v \neq v_j$ or v_k on B, vz intersects F, so $B - \{v_j, v_k\}$ is not visible from z by Fact 1 and can be ignored. Further v_j and v_k are both visible from z, so $V(P, z) = V(Q, z)$ where Q is the polygon with vertices $z, v_j, v_{j+1}, \cdots, v_{k-1}, v_k$, and z in counterclockwise order.

Appendix A

In this appendix, we illustrate Algorithms 1 and 2 for the polygon P and the interior viewpoint z in Figure 1. We also illustrate Algorithm 1 for the polygon P and the blocked exterior viewpoint z in Figure 2. The sequence of procedures P_m called by VISPOL, along with the parameter values i, t, S, and w on entering each procedure, are given in Tables 1, 2, and 3.

Table 1. *Sequence of procedure calls for polygon P and interior viewpoint z in Figure* 1 (*Algorithm* 1).

m	P_m	i	t	w	S
1	LEFT	1	1	v_1	v_0v_1
2	SCANA	2	1	v_2	unchanged
3	SCAND	6	1	a	unchanged
4	LEFT	24	3	v_{24}	$v_0v_1dv_{24}$
5	LEFT	25	4	v_{25}	$v_0v_1dv_{24}v_{25}$
6	LEFT	26	5	v_{26}	$v_0v_1dv_{24}v_{25}v_{26}$
7	LEFT	27	6	v_{27}	$v_0v_1dv_{24}v_{25}v_{26}v_{27}$
8	FINISH	27	6	v_{27}	unchanged

Table 2. *Sequence of procedure calls for polygon P and interior viewpoint z in Figure 1 (Algorithm 2).*

m	P_m	i	t	w	S
1	LEFT	1	1	v_1	v_0v_1
2	SCANA	2	1	v_2	unchanged
3	LEFT	12	3	v_{12}	$v_0v_1bv_{12}$
4	LEFT	13	4	v_{13}	$v_0v_1bv_{12}v_{13}$
5	RIGHT	14	4	v_{13}	unchanged
6	SCAND	14	1	c	v_0v_1
7	LEFT	24	3	v_{24}	$v_0v_1dv_{24}$
8	LEFT	25	4	v_{25}	$v_0v_1dv_{24}v_{25}$
9	LEFT	26	5	v_{26}	$v_0v_1dv_{24}v_{25}v_{26}$
10	LEFT	27	6	v_{27}	$v_0v_1dv_{24}v_{25}v_{26}v_{27}$
11	FINISH	27	6	v_{27}	unchanged

Table 3. *Sequence of procedure calls for polygon P and blocked exterior viewpoint z in Figure 2 (Algorithm 1).*

m	P_m	i	t	w	S
1	LEFT	1	1	v_1	v_0v_1
2	LEFT	2	2	v_2	$v_0v_1v_2$
3	SCANA	3	2	v_3	unchanged
4	RIGHT	8	2	d	unchanged
5	SCANC	9	2	v_8	v_0v_1c
6	RIGHT	16	2	e	unchanged
7	RIGHT	17	2	v_{16}	v_0v_1b
8	LEFT	18	3	v_{18}	$v_0av_{17}v_{18}$
9	SCANA	19	3	v_{19}	unchanged
10	LEFT	22	5	v_{22}	$v_0av_{17}v_{18}fv_{22}$
11	SCANA	23	5	v_{23}	unchanged
12	SCAND	27	5	g	unchanged
13	LEFT	43	7	v_{43}	$v_0av_{17}v_{18}fv_{22}hv_{43}$
14	RIGHT	44	7	v_{43}	unchanged
15	SCAND	44	3	p	$v_0av_{17}v_{18}$
16	LEFT	46	5	v_{46}	$v_0av_{17}v_{18}qv_{46}$
17	LEFT	47	6	v_{47}	$v_0av_{17}v_{18}qv_{46}v_{47}$
18	LEFT	48	7	v_{48}	$v_0av_{17}v_{18}qv_{46}v_{47}v_{48}$
19	LEFT	49	8	v_{49}	$v_0av_{17}v_{18}qv_{46}v_{47}v_{48}v_{49}$
20	FINISH	49	8	v_{49}	unchanged

REFERENCES

1. H. El Gindy and D. Avis: *A linear algorithm for computing the visibility polygon from a point*, J. Algorithms, 2, (1981), pp. 186–197.
2. H. Freeman and P. P. Loutrel: *An algorithm for the solution of the two-dimensional hidden-line problem*, IEEE Trans. on Electronic Computers, EC-16, (1967), pp. 784–790.
3. P. Henrici: *Applied and Computational Complex Analysis*, Vol. 1, John Wiley & Sons, (1974).

4. B. Joe and R. B. Simpson: *Visibility of a simple polygon from a point*, Technical Report CS-85-38. Dept. of Computer Science, Univ. of Waterloo, (1985).
5. B. Joe and R. B. Simpson: *Triangular meshes for regions of complicated shape*, Int. J. for Num. Meth. in Eng., 23 (1986), pp. 751–778.
6. B. Joe and R. B. Simpson: *Algorithms and correctness proofs for visibility polygon computations*, Technical Report CS-87-03, Dept. of Computer Science, Univ. of Waterloo, (1987).
7. D. T. Lee: *Visibility of a simple polygon*, Computer Vision, Graphics, and Image Processing, 22 (1983), pp. 207–221.
8. B. Schachter: *Decomposition of polygons into convex sets*, IEEE Trans. on Comp., C-27, (1978), pp. 1078–1082.