Visibility of a Simple Polygon from a Point

B. Joe
Dept. of Math and Computer Science
Drexel University

R.B. Simpson
Department of Computer Science
University of Waterloo

Technical Report CS-85-38
August 1985

Visibility of a Simple Polygon from a Point

B. Joe

Dept. of Math. and Computer Science
Drexel University

R. B. Simpson

Dept. of Computer Science
University of Waterloo

ABSTRACT

In 1983, Lee published a linear time algorithm for computing the visibility polygon of a simple polygon from a prescribed point. We present an alternative organization of Lee's algorithm which allows us to present it in relatively simple pseudo-code. This organization provides it in a directly programmable form, and corrects a case in Lee's description which allows it to fail. An example of a polygon is given which shows that the original version by Lee, as well as the 1981 algorithm of El Gindy and Avis can fail.

1. Introduction

Two linear time algorithms for computing the visibility set of a simple polygon, P, from a vision point, z, in the interior of P have been published, El Gindy and Avis (1981) and Lee (1983). Lee (1983) also extends his algorithm to the case where z is in the exterior of P. The cases where z is in the interior or exterior of P arise in the two-dimensional hidden line elimination problem in computer graphics. The case where z is on the boundary of P also arises in the method of Schachter (1978) for decomposing a simple polygon into convex polygons, where z is a reflex vertex of the simple polygon. In this paper, we consider the cases where z is in the interior or on the boundary of P.

Lee's algorithm is simpler in structure; in particular it requires only one stack, which eventually yields the visibility set, as opposed to three in the El Gindy and Avis case. Lee's description of his algorithm assumed, for simplicity, that no three boundary vertices are collinear. This paper had its origin in our attempt to program his algorithm, removing this technicality as a constraint, and extending the class of polygons slightly to simply connected polygons, i.e. polygonal curves which define simply connected domains, which have oriented polygonal boundaries in which some boundary segments (cut or branch lines) may be traversed twice, in opposite directions. Our efforts resulted in rather complicated code, with deeply nested if-then-else constructs, which proved difficult to read and we had little confidence in our ability to debug it, or modify it correctly. This led us to devise a modified organization of the algorithm which we feel is more suitable for coding, which we present in pseudo-code in Section 3. In the process of defining this modified organization, we discovered that Lee's algorithm is not correct for polygons that wind sufficiently as we show in Appendix A. The El Gindy and Avis algorithm also fails for this example.

The algorithm is based on a scan of the boundary of P. Central to our organization is the use of the angular displacement, $\alpha(v)$, of the ray from the vision point z to a point v on the boundary of P as a control variable. In addition to simplifying some test conditions, the use of $\alpha(v)$ clarifies how some polygons with convoluted winding lobes, e.g. Figure A.1, should be

treated. In Section 2 we prove some properties of this angular displacement that are required to justify its use in the algorithm. Both earlier papers provide a statement of a stack condition which is the justification for the correctness of their algorithms. We restate this condition in terms of angular displacement so that it is valid for our organization of the algorithm, and use it to verify the correctness of the algorithm in Section 4.

2. Notation and preliminary results

The boundary of a simple polygon P consists of a sequence of straight-line edges such that they form a cycle and no two nonconsecutive edges intersect. We represent P by a list of vertices in counterclockwise order so that the bounded interior region is to the left as the boundary of P is traversed. We denote the boundary, interior, and exterior of P by $Bd(P)$, $Int(P)$, and $Ext(P)$, respectively, so that $P = Bd(P) \cup Int(P)$. The vision point z is in $Int(P)$ or on $Bd(P)$. If z is in $Int(P)$, we denote the vertices of P as $v_0, v_1, \cdots, v_{n-1}$, and $v_n = v_0$ (the edges of P are v_0v_1, $v_1v_2, \cdots, v_{n-1}v_n$). If z is on $Bd(P)$, we denote the vertices of P as $z, v_0, v_1, \cdots, v_n$ (v_n is the predecessor of z) since this labelling allows us to treat the two cases similarly in our algorithm.

For simplification, we assume that the coordinate system is translated so that z is at the origin. If $z \in Bd(P)$, then we also assume that the coordinate system is rotated so that v_0 is on the positive x-axis. If $z \in Int(P)$, then we also assume that the vertices are relabelled so that $v_0 = v_n$ 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). We denote the polar coordinates of a point v by $(r(v), \theta(v))$ where $0 \leq \theta(v) < 2\pi$.

A subset of $Bd(P)$ composed of the chain $sv_iv_{i+1} \cdots v_{k-1}v_kt$, where s is on edge $v_{i-1}v_i$ and t is on edge v_kv_{k+1}, is denoted as $Ch[s,t]$ or $Ch(s,t)$ depending on whether the two points s, t are included or excluded, respectively. The chain uvw, where u, v, w are distinct points, is said to be a *left turn, right turn,* or *no turn* if w is to the left of, right of, or on, respectively, the directed line from u to v.

The point v is said to be *visible* from z with respect to P if the interior of the line segment joining z and v is in $Int(P)$. The problem we are considering is to find a subset $\bar{V}(P,z)$ of points on $Bd(P)$ which are visible from z, i.e. $\bar{V}(P,z) = \{v \mid v \in Bd(P)$ and v is 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 $\{v \in P \mid v$ is visible from $z\}$. 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.

We define the *angular displacement,* $\alpha(v)$, of a point v on $Bd(P)$ with respect to z as follows : $\alpha(v_0) = \theta(v_0)$; for $1 \leq i \leq n$,

$$\alpha(v_i) = \begin{cases} \alpha(v_{i-1}) + \text{angle}(v_{i-1}zv_i) & \text{if } zv_{i-1}v_i \text{ is a left turn} \\ \alpha(v_{i-1}) - \text{angle}(v_{i-1}zv_i) & \text{if } zv_{i-1}v_i \text{ is a right turn} \\ \alpha(v_{i-1}) & \text{if } z, v_{i-1}, v_i \text{ are collinear} \end{cases} \tag{2.1}$$

i.e. $\alpha(v_i) = \theta(v_i) + 2\pi k$ where k is the integer determined so that $|\alpha(v_i) - \alpha(v_{i-1})| < \pi$. For any $v \in v_{i-1}v_i$ where $1 \leq i \leq n$, $\alpha(v)$ can be defined similarly as $\theta(v) + 2\pi k$ for some integer k so that $\alpha(v)$ is a continuous function as $Bd(P)$ is traversed from $v = v_0$ to $v = v_n$. To complete the definition in the case $z \in Bd(P)$, we define $\alpha(v) = \alpha(v_0)$ if $v \in zv_0$ and $\alpha(v) = \alpha(v_n)$ if $v \in v_0v_n$. Note that the angular displacement measures the 'winding' of $Bd(P)$ in addition to the polar angle. Freeman and Loutrel (1967), who presented a nonlinear-time algorithm for finding $\bar{V}(P,z)$, used the term *total angle* for angular displacement.

From classical complex integration results (Carrier, Krook, and Pearson (1966, Section 2.3), Henrici (1974, Section 4.6)),

$$\alpha(v_n)-\alpha(v_0) = \begin{cases} \beta & \text{if } z \in Bd(P) \\ 2\pi & \text{if } z \in Int(P) \\ 0 & \text{if } z \in Ext(P) \end{cases} \tag{2.2}$$

where $\beta = \text{angle}(v_n z v_0)$ is the interior angle of z in P in the first case, and $v_0 = v_n$ is an arbitrary vertex of P in the last case (this case is referred to in the proof of Lemma 2 below). From our assumption about the location of z, $\alpha(v_0)=\theta(v_0)=0$ if $z \in P$, $\alpha(v_n)=2\pi$ if $z \in Int(P)$, and $\alpha(v_n)=\beta$ if $z \in Bd(P)$. In the case $z \in Int(P)$, since $\alpha(v_n) \neq \alpha(v_0)$ we consider v_0 and v_n to be 'distinct' points where v_0 is on edge v_0v_1 and v_n is on edge $v_{n-1}v_n$.

The following lemma follows from the Jordan curve theorem (see Figure 2.1 for an illustration in the case $z \in Int(P)$).

Lemma 1 : If the intersection of the ray emanating from z at polar angle ϕ, $0 \leq \phi < 2\pi$, and $Int(P)$ is nonempty, then the intersection consists of the line segments u_1u_2, u_3u_4, $\cdots$, $u_{2k-1}u_{2k}$, $k \geq 1$, where

(a) $u_1 = z$ or u_1 is on $Bd(P)$, and u_i, $i > 1$, is on $Bd(P)$,

(b) the interior of $u_{2i-1}u_{2i}$ is in $Int(P)$,

(c) $0 \leq r(u_1) < r(u_2) \leq r(u_3) < r(u_4) \leq \cdots \leq r(u_{2k-1}) < r(u_{2k})$,

(d) u_{2i-1} is on an edge which is oriented in the clockwise direction with respect to z, if $u_{2i-1} \neq z$,

(e) u_{2i} is on an edge which is oriented in the counterclockwise direction with respect to z.

In other words, the ray emanating from z at polar angle ϕ is broken up into line segments which alternate between being in $Int(P)$ and in $Bd(P) \cup Ext(P)$. The line segments in $Int(P)$ have positive length and do not include their endpoints (except $u_1 = z \in Int(P)$). The line segments in $Bd(P) \cup Ext(P)$ may have zero length (consist of a single point). The first line segment may be in $Int(P)$ or $Bd(P) \cup Ext(P)$ depending on the location of z and the angle ϕ. The last line segment is in $Bd(P) \cup Ext(P)$ and is of infinite length. From Lemma 1, it follows that there is at most one point v on $Bd(P)$ visible from z at each polar angle ϕ, and v is on an edge which is oriented in the counterclockwise direction with respect to z if $z \in P$. □

The following lemma uses the same notation as Lemma 1 (see Figure 2.1).

Lemma 2 : For $1 \leq i \leq k$, $\alpha(u_{2i-1}) = \alpha(u_{2i})$ if $u_{2i-1} \neq z$.

Proof : If $u_{2i-1} \neq z$, the line segment $u_{2i-1}u_{2i}$ subdivides P into two simple subpolygons. If u_{2i-1} occurs before u_{2i} in a traversal of $Bd(P)$ from v_0 to v_n, then let Q be the subpolygon formed by $Ch[u_{2i-1},u_{2i}]$ and line segment $u_{2i}u_{2i-1}$, else let Q be the subpolygon formed by $Ch[u_{2i},u_{2i-1}]$ and line segment $u_{2i-1}u_{2i}$, so that $z \in Ext(Q)$. (If u_{2i-1} is the point v_0 or v_n in the case $z \in Int(P)$, then u_{2i-1} can be named appropriately as v_0 or v_n so that $z \in Ext(Q)$ - the other choice would give $z \in Int(Q)$.) If the definition of angular displacement is applied to Q, then, by (2.2), u_{2i-1} and u_{2i} would have the same angular displacement on $Bd(Q)$. This implies that $\alpha(u_{2i-1}) = \alpha(u_{2i})$ on $Bd(P)$ also. □

As elaborated further in the next section, we denote the visibility polygon by the chain $s_0s_1 \cdots s_t$ where $s_0 = v_0$, $s_t = v_n$, $s_j \in Bd(P)$, $0 \leq j \leq t$, s_js_{j+1} is on $Bd(P)$ if $\theta(s_j) \neq \theta(s_{j+1})$, and

$$0 = \theta(s_0) \leq \theta(s_1) \leq \cdots \leq \theta(s_m) = \beta, \; m = t, \text{ if } z \in Bd(P)$$

$$0 = \theta(s_0) \leq \theta(s_1) \leq \cdots \leq \theta(s_m) < 2\pi,$$

$$\theta(s_{m+1}) = \cdots = \theta(s_t) = 0, \; m < t, \text{ if } z \in Int(P).$$

See Figure 2.2 for examples.

Lemma 3 : If $v \in Bd(P)$ is visible from z and $v \neq v_n$, then $\alpha(v) = \theta(v)$.

Proof : If $v \in Bd(P)$ is visible from z and $\theta(v) = \phi$, then v is the point with polar angle ϕ on the chain $s_0s_1 \cdots s_t$ which is closest to z.

We first show by induction that

$$\alpha(s_j) = \theta(s_j)\ ,\ 0 \le j \le m\ . \tag{2.3}$$

$s_0 = v_0$ and $\alpha(v_0) = \theta(v_0) = 0$ so (2.3) is true for $j=0$. Suppose (2.3) is true for $j<m$. If $\theta(s_{j+1})>\theta(s_j)$ then s_js_{j+1} is on $Bd(P)$ and by the definition of angular displacement $\alpha(s_{j+1})=\alpha(s_j)+\text{angle}(s_{j+1}zs_j)$. If $\theta(s_{j+1})=\theta(s_j)$ and s_js_{j+1} is on $Bd(P)$ then $\alpha(s_{j+1})=\alpha(s_j)$ by definition. If $\theta(s_{j+1})=\theta(s_j)$ and s_js_{j+1} is not on $Bd(P)$ then an argument similar to that used in the proof of Lemma 2 shows that $\alpha(s_{j+1})=\alpha(s_j)$. From the inductive hypothesis $\alpha(s_j)=\theta(s_j)$, it follows that $\alpha(s_{j+1})=\theta(s_{j+1})$ in all three cases. Therefore (2.3) is true for $j+1$. Similarly $\alpha(s_j)=2\pi$ for $m+1\le j\le t$.

Now let $v\in Bd(P)$ be visible from z and $v\neq v_n$. If $v=s_j$ for some j then $\alpha(v)=\theta(v)$ since $j\le m$. Otherwise v is on the interior of an edge s_js_{j+1} where $\alpha(s_j)<\alpha(s_{j+1})$, $\alpha(s_j)=\theta(s_j)$, and either $\alpha(s_{j+1})=\theta(s_{j+1})$ or $\alpha(s_{j+1})=2\pi$, so $\alpha(v)=\theta(v)$ by the definition of angular displacement. □

Corollary : If $v\in Bd(P)$ and either $\alpha(v)<0$ or $\alpha(v)>2\pi$, then v is not visible from z.

3. The algorithm

The algorithm carries out a monotone scan of $Bd(P)$ starting from edge v_0v_1 and ending at edge $v_{n-1}v_n$, while manipulating a stack of vertices $s_0, s_1, \cdots, s_t$ such that ultimately the chain $s_0s_1\cdots s_t$ or $zs_0s_1\cdots s_tz$ becomes $Bd(V(P,z))$ depending on whether $z\in Int(P)$ or $z\in Bd(P)$, respectively. The computation of $\alpha(v_i)$ in this scan requires $\alpha(v_{i-1})$ and $\text{angle}(v_{i-1}zv_i)$ as indicated in (2.1). The angular displacement values can be computed for all v_i in a preprocessing step and saved, or computed once when needed during the scan of $Bd(P)$. When s_t is put on the stack, $\alpha(s_t)$, which is either 2π or $\alpha(v_k)$ for some k, can be saved for future reference. Hence $\alpha(v)$ is used for notational convenience in our algorithm and is not a function reference.

Our organization of the algorithm involves a decomposition into three procedures:
ADVANCE - adds vertices to the stack,
RETARD - removes vertices from the stack,
SCAN - scans invisible edges of $Bd(P)$ without modifying the stack.

Each of these procedures is entered with a current edge of $Bd(P)$, v_iv_{i+1}, for which at least a subedge including the point v_{i+1} is to be treated according to the procedure's name. Each procedure thus carries out the operation indicated by its name on successive boundary segments until it determines one that requires treatment by another process.

When either ADVANCE or RETARD determines that the next segment should be processed by SCAN, the process can also determine an interval, or 'window', across which a segment of $Bd(P)$ must pass in order to leave the SCAN process, and the manner of crossing (clockwise or counterclockwise). One end of the window is always the top of the stack, s_t, and the other end is passed to SCAN in argument w, which is a point on the polar ray zs_t (including possibly the point at ∞). The direction of passage out this window is passed to SCAN in logical variable *ccw* (which is true if counterclockwise).

Suppose $Ch[v_0,v_i]$ has been scanned so far by the algorithm, and the stack contains the points $s_0, s_1, \cdots, s_t$. Let $S_i = \text{chain}\, s_0s_1\cdots s_t$. Then the following stack properties are satisfied:

(S1) $0=\alpha(s_0)\le\alpha(s_1)\le\cdots\le\alpha(s_t)\le 2\pi$, $s_0=v_0$, and $0\le t\le i$.

(S2) $s_j\in Bd(P)$ for $0\le j\le t$, and $s_js_{j+1}\in Bd(P)$ if $\alpha(s_j)<\alpha(s_{j+1})$.

(S3) If $v\in Ch[v_0,v_i]$ but v is not on S_i then v is not visible from z.

We note that the following intuitively attractive stack property "S_i is composed of the visible points $v\in Ch(v_0,v_i)$ for which $0<\alpha(v)<\alpha(s_t)$" is not true in general.

We now give the pseudo-code for our algorithm, followed by some remarks about the algorithm.

```
procedure VISPOL(z,v,n,s,t); [1]
# Input : vision point z in P and vertices v = v_0,v_1, ··· ,v_n of P
#         with v_0 satisfying assumption in Section 2
# Output : visibility polygon vertices s = s_0,s_1, ··· ,s_t where s_0=v_0, s_t=v_n
  s_0 := v_0;
  i := 0; t := 0;  # stack properties are satisfied
  if α(v_1) ≥ α(v_0) then
    vpcase := 'advance';
  else
    vpcase := 'scan'; ccw := true;  # see remark 2
    w := point with polar coordinates (∞,θ(v_0));
  endif;
  while vpcase ≠ 'finish' do
    case vpcase of
        'advance' : ADVANCE(z,v,n,s,t,i,vpcase,ccw,w);
        'retard'  : RETARD(z,v,n,s,t,i,vpcase,ccw,w);
        'scan'    : SCAN(z,v,n,s,t,i,vpcase,ccw,w);
  endwhile;
```

[1] In this pseudo-code, we follow the FORTRAN 77 convention for if statements, i.e. at the termination of any clause of an

```
if ... then ...
else if ... then ...
else if ... then ...
:
:
else ...
endif
```

construct, control passes to the next endif at this level. Also, a '#' symbol indicates that the rest of the line is a comment.

```
procedure ADVANCE(z,v,n,s,t,i,vpcase,ccw,w);
# Input : z, v, n
# Input and output : s, t, i, vpcase
# Output : ccw, w
  # α(v_{i+1}) ≥ max(α(s_t),α(v_i)) and s_t ≠ v_{i+1} is on edge v_i v_{i+1}
  while vpcase = 'advance' do
    if α(v_{i+1}) ≤ 2π then
        i := i+1; t := t+1; s_t := v_i;   # stack properties are satisfied
        if i = n then
          vpcase := 'finish';
        else if α(v_{i+1}) < α(v_i) and v_{i-1}v_i v_{i+1} is a right turn then
          vpcase := 'scan'; ccw := true;   # see remark 2
          w := point with polar coordinates (∞,θ(v_i));
        else if α(v_{i+1}) < α(v_i) and v_{i-1}v_i v_{i+1} is a left turn then
          vpcase := 'retard';   # see remarks 3, 4, and 5
        endif;
        # vpcase remains at 'advance' if α(v_{i+1}) ≥ α(v_i)
    else
        if α(s_t) < 2π then
          t := t+1; s_t := intersection of v_i v_{i+1} and line z→v_0;
        endif;
        vpcase := 'scan'; ccw := false; w := v_0;   # see remark 6
    endif;
  endwhile;
```

```
procedure RETARD(z,v,n,s,t,i,vpcase,ccw,w);
# Input : z, v, n
# Input and output : s, t, i, vpcase
# Output : ccw, w
  # α(v_{i+1}) < α(v_i) and α(v_{i+1}) ≤ α(s_t) ≤ α(v_i)
  while vpcase = 'retard' do
    # see remark 3
    scan backwards s_{t-1}, s_{t-2}, ··· , s_0 for first stack vertex s_j such that either
        (a) α(s_j) < α(v_{i+1}) ≤ α(s_{j+1}), or
        (b) α(v_{i+1}) ≤ α(s_j) = α(s_{j+1}) and v_i v_{i+1} intersects s_j s_{j+1};
    if α(s_j) < α(v_{i+1}) then    # case (a)
        i := i+1; t := j+1;
        s_t := intersection of s_j s_{j+1} and line (ra zv_i;
        t := t+1; s_t := v_i;  # stack properties are satisfied
        if i = n then
          vpcase := 'finish';
        else if α(v_{i+1}) ≥ α(v_i) and v_{i-1} v_i v_{i+1} is a right turn then
          vpcase := 'advance';
        else if α(v_{i+1}) > α(v_i) and v_{i-1} v_i v_{i+1} is a left turn then
          vpcase := 'scan'; ccw := false; w := v_i; t := t-1;
          # see remark 4
        else
          t := t-1;
        endif;
        # vpcase remains at 'retard' if α(v_{i+1}) < α(v_i)
        #    or α(v_{i+1}) = α(v_i) and r(v_{i+1}) > r(v_i)
    else  # case (b) - see remark 5
        if α(v_{i+1}) = α(s_j) and α(v_{i+2}) > α(v_{i+1}) and v_i v_{i+1} v_{i+2} is a right turn then
          vpcase := 'advance'; i := i+1; t := j+1; s_t := v_i;
          # stack properties are satisfied
        else
          vpcase := 'scan'; t := j; ccw := true;
          w := intersection of v_i v_{i+1} and s_j s_{j+1};
        endif;
    endif;
  endwhile;
```

```
procedure SCAN(z,v,n,s,t,i,vpcase,ccw,w);
# Input : z, v, n, ccw, w
# Input and output : s, t, i, vpcase
  # θ(s_t) = θ(w), window is s_t w
  # α(v_i) ≤ α(s_t) if ccw, α(v_i) > α(s_t) if not ccw
  while vpcase = 'scan' do
    i := i+1;  # stack properties are satisfied
    if ccw and α(v_{i+1}) > α(s_t) ≥ α(v_i) then
        # see remarks 2 and 5
        if v_i v_{i+1} intersects s_t w then
          s_{t+1} := intersection of v_i v_{i+1} and s_t w;
          t := t+1; vpcase := 'advance';
        endif;
    else if not ccw and α(v_{i+1}) ≤α(s_t) < α(v_i) then
        # see remarks 4 and 6
        if v_i v_{i+1} intersects s_t w then
          vpcase := 'retard';
        endif;
    endif;
  endwhile;
```

Remark 1 : We show that stack properties (S1) and (S2) are satisfied for all i. Initially $i=0$, $s_0=v_0$, and $\alpha(s_0)=0$. In ADVANCE, each new point, s_t, put on top of the stack satisfies $\alpha(s_{t-1})\leq\alpha(s_t)\leq 2\pi$ and $s_{t-1}s_t$ is on $Bd(P)$. In RETARD, vertices $s_{j+1},\cdots,s_t$ are popped from the stack but $s_0=v_0$ is never popped from the stack as mentioned in remark 3, so $t\geq 0$ is always satisfied; each new point, s_t, put on top of the stack is either on edge s_js_{j+1} with $s_{t-1}=s_j$, $\alpha(s_t)>\alpha(s_{t-1})$, and $s_{t-1}s_t$ on $Bd(P)$ (since s_js_{j+1} is on $Bd(P)$) or is v_i with $\alpha(s_t)=\alpha(s_{t-1})$. In SCAN, each new point, s_t, put on top of the stack is on edge v_iv_{i+1} with $\alpha(s_t)=\alpha(s_{t-1})$. Finally, $t\leq i$ since t is increased by at most 1 each time i is increased by 1.

Remark 2 : If $\alpha(v_{i+1})<\alpha(v_i)$ with $v_i=s_t$ and either $i=0$ or $v_{i-1}v_iv_{i+1}$ is a right turn with s_{t-1} on edge $v_{i-1}v_i$ (see Figure 3.1), then a scan is made starting from edge $v_{i+1}v_{i+2}$ for the first edge v_kv_{k+1} such that $\alpha(v_k)\leq\alpha(s_t)<\alpha(v_{k+1})$ and v_kv_{k+1} intersects s_tw where w is the 'point' with polar coordinates $(\infty,\theta(s_t))$. The existence of edge v_kv_{k+1} follows from Lemmas 1 and 2. Let u be the intersection of v_kv_{k+1} and s_tw. Let v be a point on $Ch(v_i,u)$. Since $Bd(P)$ is simple and $s_0=v_0$ is visible from z or zs_0 is on $Bd(P)$, either $\alpha(v)<0$ or $0\leq\alpha(v)\leq\alpha(s_t)$ and there exists a point on $Ch[v_0,s_t]$ with angular displacement $\alpha(v)$ which is closer to z than v. In the former case, v is not visible from z by Lemma 4. In the latter case, v is not visible from z since part of line segment zv lies in $Ext(P)$.

Remark 3 : If RETARD is entered after exiting ADVANCE, then $s_t=v_i$ and $\alpha(v_{i+1})<\alpha(v_i)$. If RETARD is entered after exiting SCAN, then $\alpha(v_i)>\alpha(s_t)\geq\alpha(v_{i+1})$. Either vertex v_{i+1} is on line segment zs_t with $r(v_{i+1})<r(s_t)$ or part of edge v_iv_{i+1} is in 'front' of chain $S_i=s_0s_1\cdots s_t$ with respect to z. Since $Bd(P)$ is simple, and $s_0=v_0$ is visible from z or zs_0 is on $Bd(P)$, and the $\alpha(s_j)$'s are in nondecreasing order, v_{i+1} must satisfy for some $j\geq 0$ (see Figure 3.2)

(a) $\alpha(s_j)<\alpha(v_{i+1})\leq\alpha(s_{j+1})$, or

(b) $\alpha(v_{i+1})\leq\alpha(s_j)=\alpha(s_{j+1})$ and v_iv_{i+1} intersects s_js_{j+1} where s_js_{j+1} is not on $Bd(P)$.

In case (a), let u be the intersection of s_js_{j+1} and line $\overrightarrow{zv_{i+1}}$. A point v on edge s_ks_{k+1}, where $j+1\leq k<t$, or on us_{j+1} in case (a), is not visible from z since there is a point on v_iv_{i+1} with polar angle $\theta(v)$ which is closer to z than v. A point $v\neq v_{i+1}$ on edge v_iv_{i+1} is not visible from z since v_iv_{i+1} is a clockwise oriented edge with respect to z implies that part of line segment zv lies in $Ext(P)$.

Remark 4 : If case (a) in remark 3 occurs, then the stack and i are updated with s_t, $t=j+1$, set to the intersection of s_js_{j+1} and $\overrightarrow{zv_i}$, and then v_i is pushed on top of the stack. If

$\alpha(v_{i+1}) \geq \alpha(v_i)$ and $v_{i-1}v_iv_{i+1}$ is a right turn then ADVANCE is entered next; else if $\alpha(v_{i+1}) \leq \alpha(v_i)$ then the next edge v_iv_{i+1} is processed by RETARD as in remark 3. The remaining case occurs when $\alpha(v_{i+1}) > \alpha(v_i)$ and $v_{i-1}v_iv_{i+1}$ is a left turn (see Figure 3.3). v_i is popped from the stack. Then a scan is made starting from edge $v_{i+1}v_{i+2}$ for the first edge v_kv_{k+1} such that $\alpha(v_k) > \alpha(s_t) \geq \alpha(v_{k+1})$ and v_kv_{k+1} intersects s_tv_i. The existence of edge v_kv_{k+1} follows from Lemmas 1 and 2. Let v be a point on $Ch(v_i,v_k)$. Since $Bd(P)$ is simple, v lies in the 'region' bounded by $Ch[s_t,v_i]$ and line segment v_is_t, and there exists a point on $Ch[s_t,v_i]$ with angular displacement $\alpha(v)$ which is closer to z than v, so v is not visible from z. A point $v \neq v_{k+1}$ on edge v_kv_{k+1} or $v = v_i$ is not visible from z since part of line segment zv lies in $Ext(P)$.

Remark 5 : If case (b) in remark 3 occurs, then the stack is updated with $s_t = s_j$ as illustrated in Figure 3.4. If $\alpha(v_{i+1}) = \alpha(s_t)$, and $\alpha(v_{i+2}) > \alpha(v_{i+1})$, and $v_iv_{i+1}v_{i+2}$ is a right turn (v_{i+2} exists since $v_{i+1} \neq v_n$) then $v_{i+1}v_{i+2}$ is in 'front' of v_iv_{i+1} with respect to z so ADVANCE is entered next. Otherwise, a scan is made starting from edge $v_{i+1}v_{i+2}$ for the first edge v_kv_{k+1} such that $\alpha(v_k) \leq \alpha(s_t) < \alpha(v_{k+1})$ and v_kv_{k+1} intersects s_tw where w is the intersection of v_iv_{i+1} and line $\overrightarrow{zs_t}$. The existence of edge v_kv_{k+1} follows from Lemmas 1 and 2. Let u be the intersection of v_kv_{k+1} and s_tw. Let v be a point on $Ch(w,u)$. Since $Bd(P)$ is simple, v lies in the 'region' bounded by $Ch[s_t,w]$ and line segment ws_t, and there exists a point on $Ch[s_t,w]$ with angular displacement $\alpha(v)$ which is closer to z than v, so v is not visible from z.

Remark 6 : If $\alpha(v_{i+1}) > 2\pi \geq \alpha(v_i)$ (see Figure 3.5) then s_t is the point on v_iv_{i+1} with $\alpha(s_t) = 2\pi$ and a scan is made starting from edge $v_{i+1}v_{i+2}$ for the first edge v_kv_{k+1} such that $\alpha(v_k) > \alpha(s_t) \geq \alpha(v_{k+1})$ and v_kv_{k+1} intersects v_0s_t. The existence of edge v_kv_{k+1} follows from Lemmas 1 and 2. Let v be a point on $Ch(s_t,v_k)$. If $\alpha(v) > 2\pi$ or $\alpha(v) < 0$, then v is not visible from z by Lemma 4. If $0 \leq \alpha(v) \leq 2\pi$ then, since $Bd(P)$ is simple, v lies on line segment s_ts_0 or in the exterior of the region bounded by chain $S_{i+1} = s_0s_1 \cdots s_t$ and line segment s_ts_0, so v is not visible from z. A point $v \neq v_{k+1}$ on edge v_kv_{k+1} is not visible from z since part of line segment zv lies in $Ext(P)$.

4. Correctness and running time of the algorithm

In this section, we verify the correctness of the algorithm and show that the algorithm runs in linear time.

Theorem 1 : For all i, $0 \leq i \leq n$, the stack properties (S1), (S2), (S3) are satisfied after $Ch[v_0,v_i]$ has been scanned (at the places indicated in the pseudo-code), and the algorithm terminates with $s_t = v_n$.

Proof : From remark 1, (S1) and (S2) are satisfied for each i. From remarks 2 to 6, (S3) is satisfied for each i. In SCAN, i is increased by at least 1 and from remarks 2, 4, 5, 6 there exists an edge v_iv_{i+1} which causes ADVANCE or RETARD to be entered next. In ADVANCE and RETARD, i is increased by 1 each time through the while loop, except possibly when SCAN is entered next. Therefore, eventually the algorithm must exit ADVANCE or RETARD with $i = n$ and $s_t = v_n$ and terminate. □

Theorem 2 : $Bd(V(P,z)) = S_n \cup \{v_nz, zv_0 \text{ if } z \in Bd(P)\}$ where S_n is the chain $s_0s_1 \cdots s_t$ at the end of the algorithm (i.e. the algorithm correctly computes $V(P,z)$).

Proof : If $v \in Ch[v_0,v_n]$ but v is not on S_n then v is not visible from z by Theorem 1 and (S3). Therefore if $v \in Ch[v_0,v_n]$ is visible from z, then v is on S_n. If v is on S_n but v is not the closest point to z on S_n with polar angle $\theta(v)$, then v is not visible from z since the intersection of the interior of line segment zv and $Bd(P)$ is nonempty. If v is on S_n and v is the closest (or only) point to z on S_n with polar angle $\theta(v)$ (where $0 \leq \theta(v) < 2\pi$ if $z \in Int(P)$ and $0 < \theta(v) < \theta(v_n)$ if $z \in Bd(P)$), then v is on $Bd(P)$ by Theorem 1 and (S2), and v must be visible from z since there exists a point on $Bd(P)$ with polar angle $\theta(v)$ which is visible from z and all other points on $Bd(P)$ with polar angle $\theta(v)$ are not visible from z. Since the $\alpha(s_j)$'s are in nondecreasing order by Theorem 1 and (S1), any point on S_n which is not visible from z is on an edge s_js_{j+1} in which $\alpha(s_j) = \alpha(s_{j+1})$; these edges are added to connect the visible points on $Bd(P)$. Also, note that if three or more consecutive stack vertices have the same angular displacement, then their distances

from z are in strictly increasing or decreasing order since $Bd(P)$ is simple. Therefore $Bd(V(P,z))=S_n \cup \{v_n z, z v_0$ if $z \in Bd(P)\}$. □

Theorem 3 : The algorithm given in procedures VISPOL, ADVANCE, RETARD, and SCAN runs in $O(n)$ time.

Proof : Each edge $v_i v_{i+1}$, $0 \leq i \leq n-1$, on $Bd(P)$ is scanned once in the algorithm. For each edge $v_i v_{i+1}$ scanned, at most 2 points are pushed onto the stack. Therefore at most $2n$ points are pushed onto the stack in the algorithm, and at most $2n$ points are popped from the stack in RETARD. The computation of the angular displacement of a polygon vertex v_i or stack vertex s_t can be done in constant time (as mentioned in Section 3). Also, computing the intersection of two edges or lines and determining whether $v_{i-1} v_i v_{i+1}$ is a left or right turn can each be done in constant time. Therefore the algorithm runs in $O(n)$ time. □

Appendix A

In this appendix, we illustrate the algorithm with the polygon P and the vision point $z \in Int(P)$ in Figure A.1, and show that the original version of Lee's algorithm and El Gindy and Avis's algorithm fail for this example.

Our algorithm : $s_0 = v_0$ is initially pushed on the stack in VISPOL. ADVANCE is entered first and $s_1 = v_1$ is pushed on the stack. Next, SCAN is entered with $ccw =$ true and $w = (\infty, \theta(v_1))$; SCAN is exited when edge $v_{11} v_{12}$ is scanned and $s_2 = a$ is pushed on the stack where a is on $v_{11} v_{12}$ (note that $v_7 v_8$ crosses $s_1 w$ at point c in the counterclockwise direction but $\alpha(c) = \alpha(s_1) - 2\pi$. Then, ADVANCE is entered and $s_3 = v_{12}$ and $s_4 = v_{13}$ are pushed on the stack. Then, RETARD is entered with case (b) occurring : $v_{13} v_{14}$ intersects $s_1 s_2$ so s_4, s_3, and s_2 are popped from the stack. Then, SCAN is entered with $ccw =$ true and $w = b$ where b is on $v_{13} v_{14}$; SCAN is exited when edge $v_{23} v_{24}$ is scanned and $s_2 = g$ is pushed on the stack where g is on $v_{23} v_{24}$ (note that $v_{19} v_{20}$ crosses $s_1 b$ at point e in the counterclockwise direction but $\alpha(e) = \alpha(s_1) - 2\pi$. Finally, ADVANCE is entered and $s_3 = v_{24}$, $s_4 = v_{25}$, $s_5 = v_{26}$, and $s_6 = v_{27}$ are pushed on the stack. The final stack chain is $S_{27} = s_0 s_1 s_2 s_3 s_4 s_5 s_6 = v_0 v_1 g v_{24} v_{25} v_{26} v_{27} = Bd(V(P,z))$.

Lee's algorithm : This algorithm proceeds in the same way as our algorithm until edge $v_{13} v_{14}$ intersects $s_1 s_2$. Then a scan is made for the first edge to cross $s_1 b$ in the counterclockwise direction; in this case the edge is $v_{19} v_{20}$ and the intersection point is e. Then $s_2 = e$, $s_3 = v_{20}$, $s_4 = v_{21}$, $s_5 = v_{22}$, and $s_6 = v_{23}$ are pushed on the stack as the scan of $Bd(P)$ proceeds in the counterclockwise direction. Then the precautionary step is executed since $s_6 = v_{23}$ subtends an angle greater than 2π with respect to z; a scan is made for the first edge which crosses $v_0 h$ in the counterclockwise direction where h is on $v_{22} v_{23}$; but in this case no such edge exists when the scan of $Bd(P)$ is completed at edge $v_{26} v_{27}$. Therefore Lee's algorithm fails for this example and other polygons in which the scan in a hidden region does not account for the winding of the boundary.

El Gindy and Avis's algorithm : This algorithm is for polygons which are oriented in the clockwise direction. It can be easily modified for polygons oriented in the counterclockwise direction by interchanging all occurrences of 'left turn' and 'right turn'. This algorithm does not handle parts of $Bd(P)$ with negative angular displacement correctly so v_0, v_1, f, c, v_8, v_9, v_{10}, p, q are pushed on the stack as the scan proceeds to edge $v_{14} v_{15}$. Then q, p, v_{10}, v_9, v_8, c are popped from the stack as the scan proceeds to edge $v_{19} v_{20}$. Then e, v_{20}, v_{21}, v_{22}, h, v_{27} are pushed on the stack as the scan proceeds to the last edge $v_{26} v_{27}$, and the algorithm terminates with v_0, v_1, f, e, v_{20}, v_{21}, v_{22}, h, v_{27} as the visibility polygon vertices which is clearly wrong.

Acknowledgment

This work has been partially supported by a grant from the Natural Sciences and Engineering Research Council of Canada and a Drexel Faculty Development Research Grant.

References

G. F. Carrier, M. Krook, and C. E. Pearson (1966), *Functions of a Complex Variable,* McGraw-Hill.

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

H. Freeman and P. P. Loutrel (1967), An algorithm for the solution of the two-dimensional hidden-line problem, *IEEE Trans. on Electronic Computers,* EC-16, pp. 784-790.

P. Henrici (1974), *Applied and Computational Complex Analysis, Vol. 1,* John Wiley & Sons.

D. T. Lee (1983), Visibility of a simple polygon, *Computer Vision, Graphics, and Image Processing,* 22, pp. 207-221.

B. Schachter (1978), Decomposition of polygons into convex sets, *IEEE Trans. on Comp.,* C-27, pp. 1078-1082.

B. Joe (1984), Finite Element Triangulation of Complex Regions Using Computational Geometry, Ph.D. thesis, Technical Report CS-84-19, University of Waterloo.

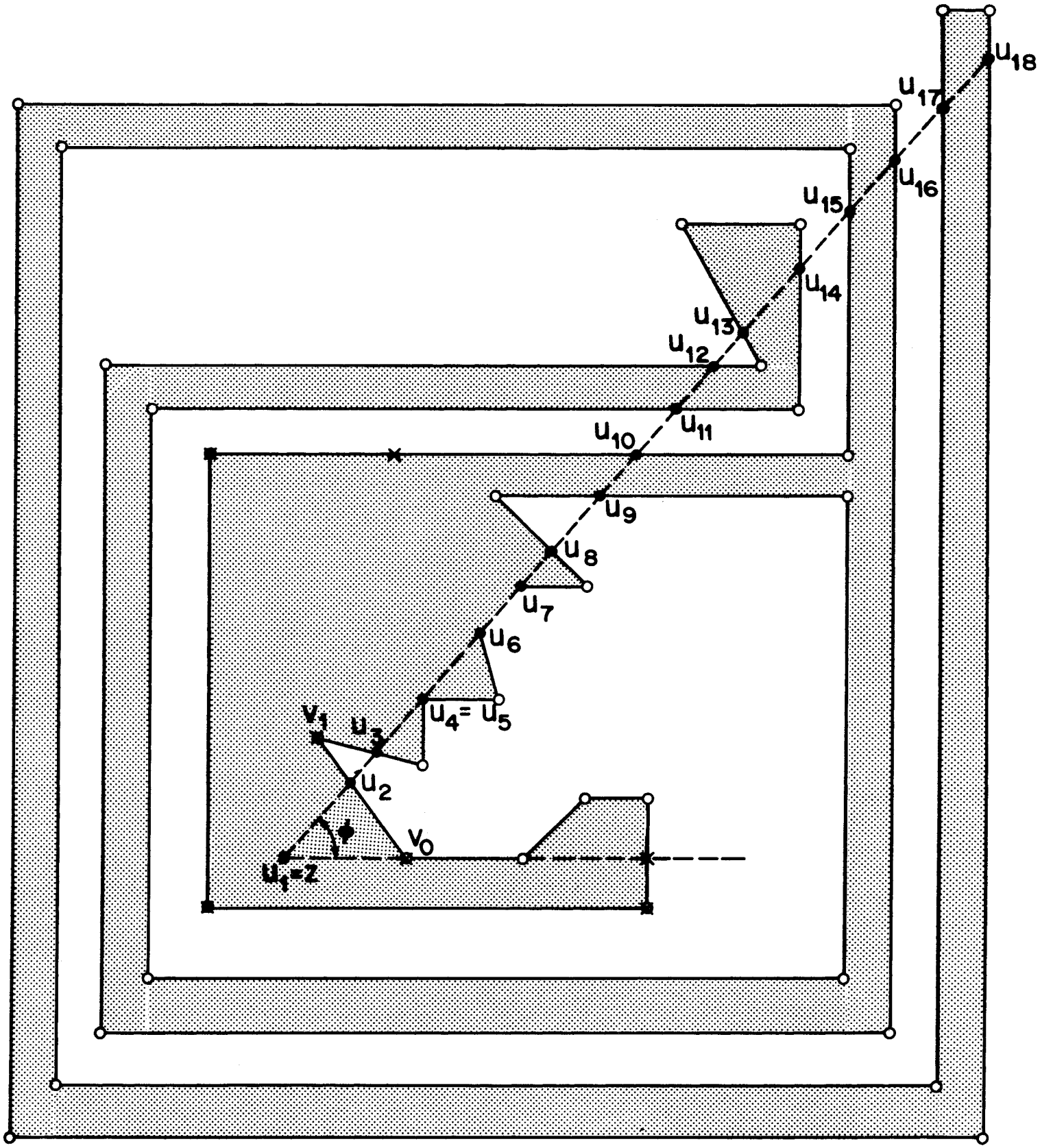


Figure 2.1 Intersection of P and the ray from z at polar angle ϕ.
o - polygon vertices, • - intersection point,
x - vertices of visibility polygon from z

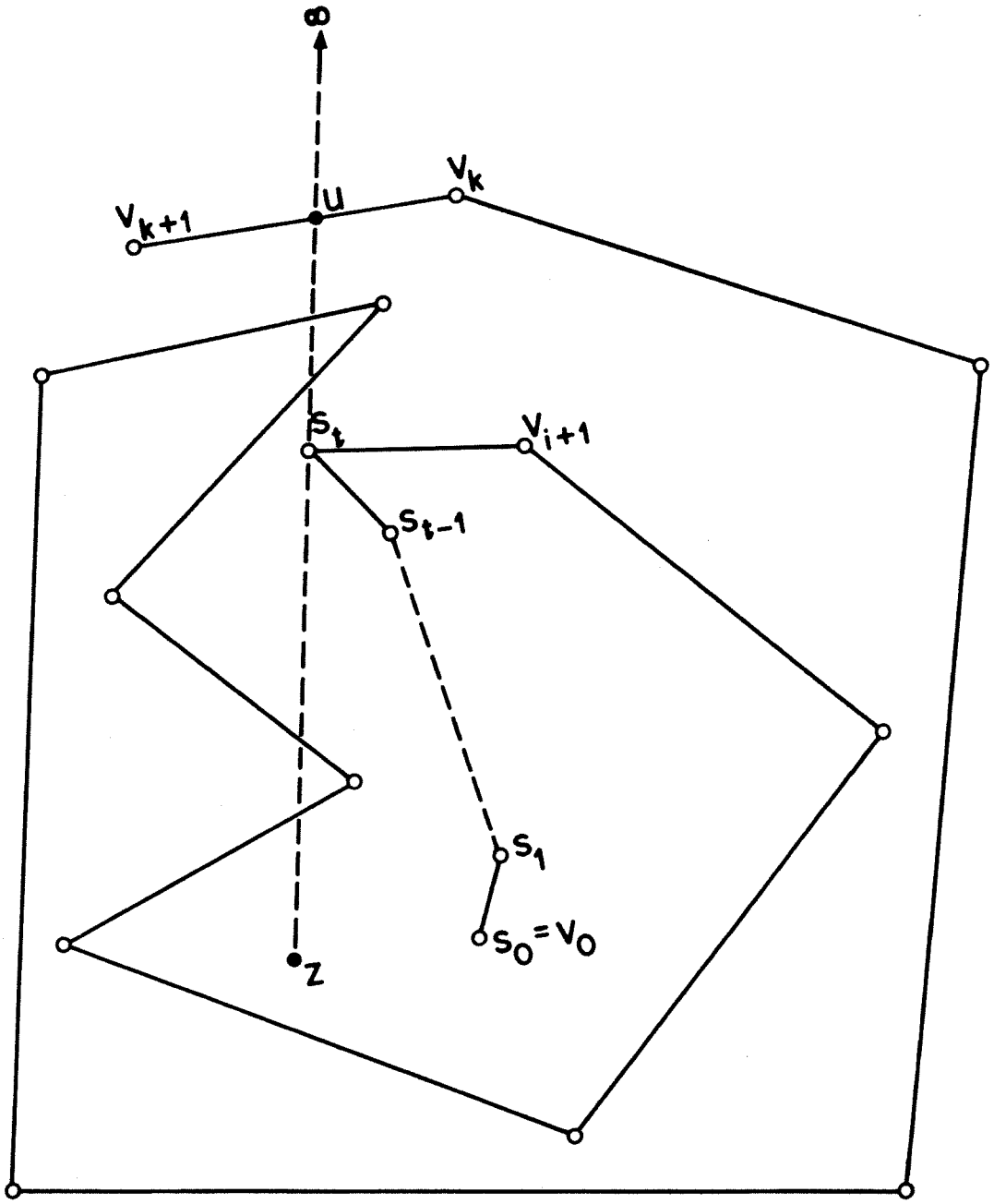
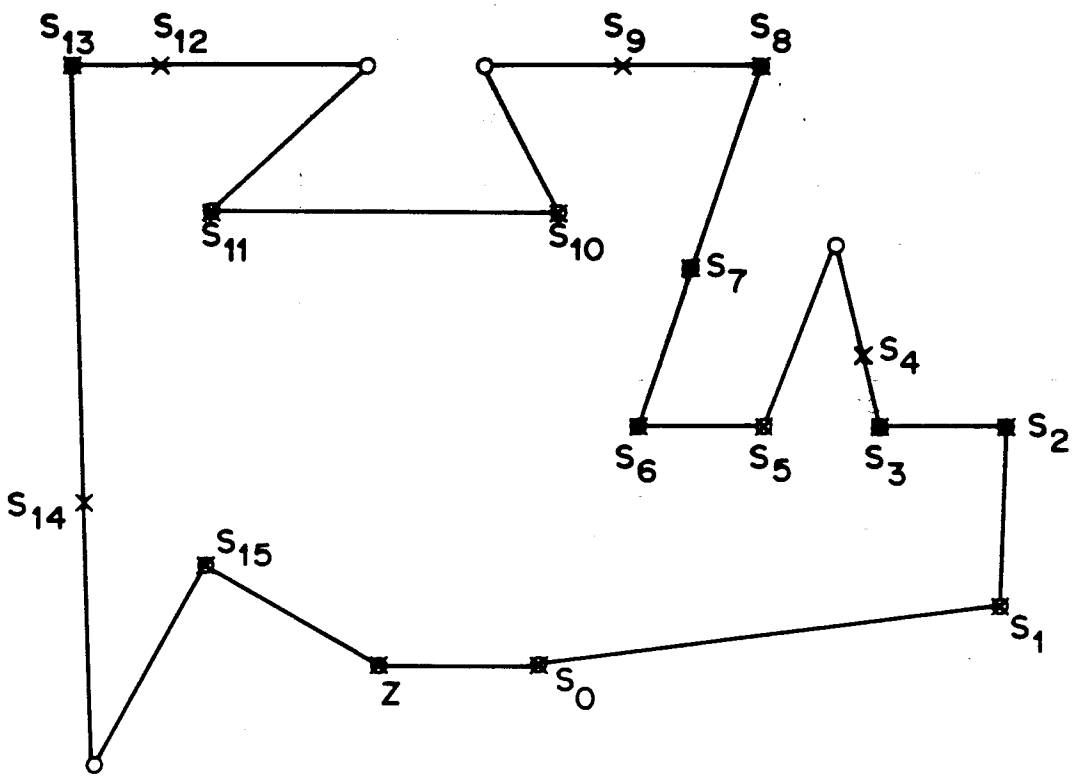
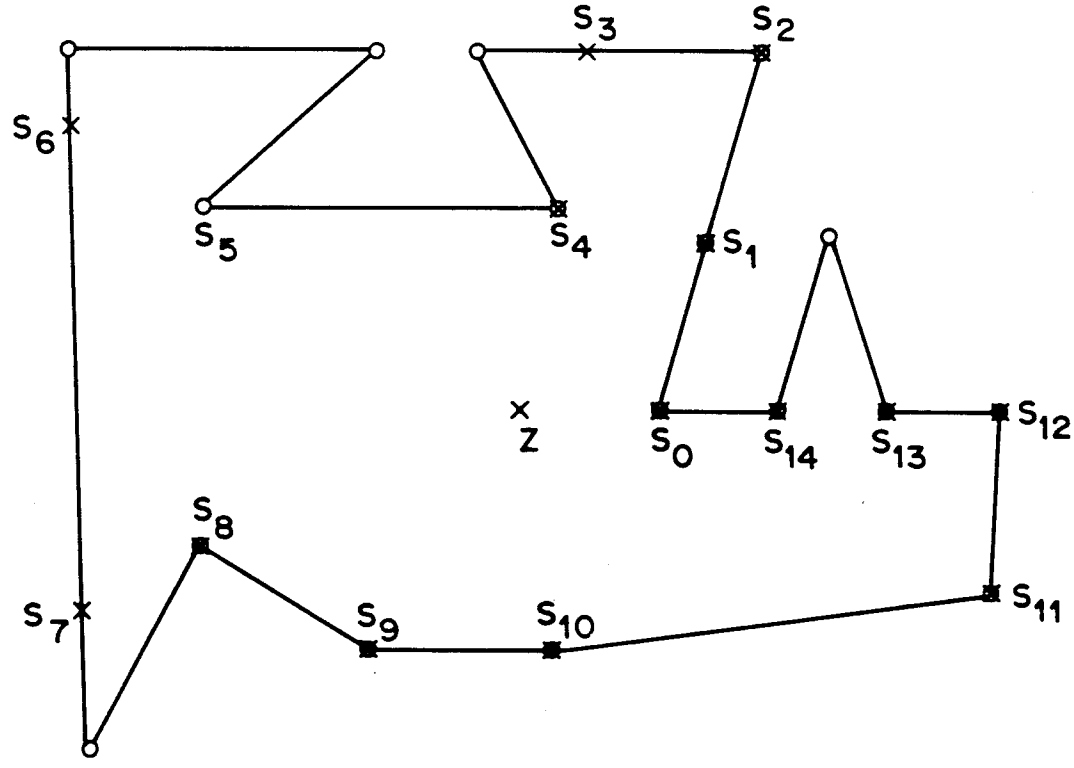


Figure 3.1 Illustration for remark 2 (ADVANCE, SPAN).

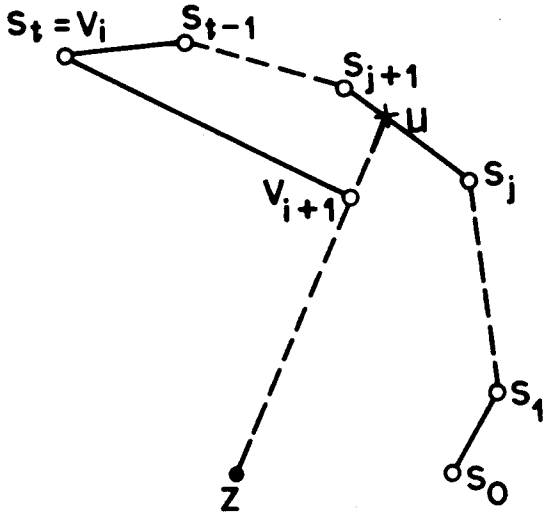


(a) $z \in Bd(P)$ $m = t = 15$

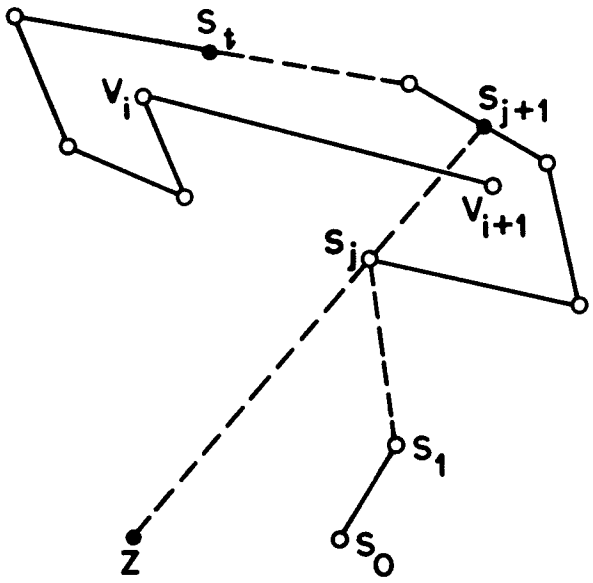


(b) $z \in Int(P)$ $m = 11$, $t = 14$

Figure 2.2 Examples of chains $s_0 s_1 \cdots s_m s_{m+1} \cdots s_t$.
o - polygon vertices, x - visibility polygon vertices



Case (a) of procedure RETARD



Case (b) of procedure RETARD

Figure 3.2 Illustration for remark 3.

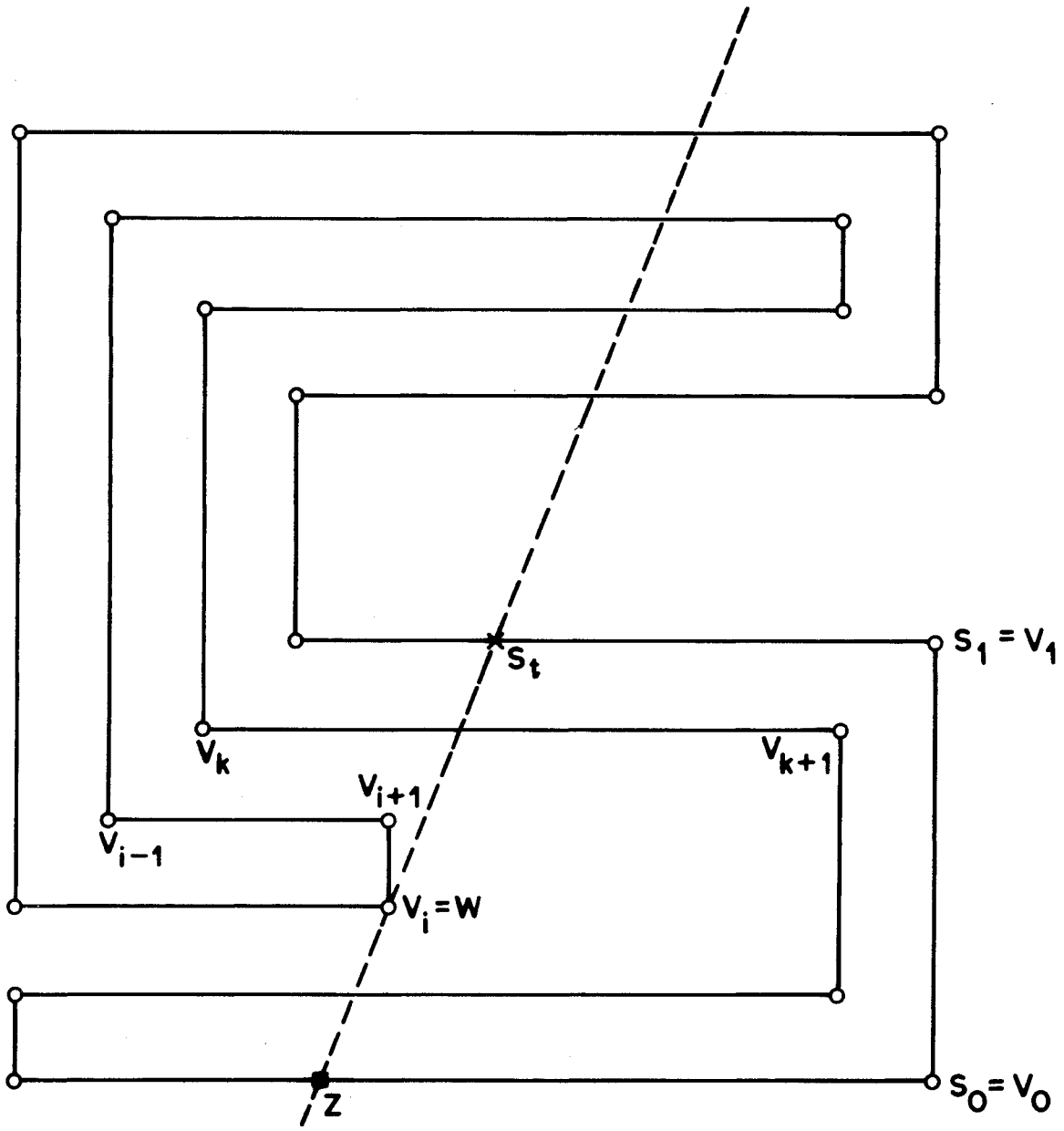


Figure 3.3 Illustration for remark 4 (RETARD, SCAN).
The values of $i=8$, v_i, w, $t=2$, s_t are set as on exit from RETARD.
The values of $k=15$ and v_k are as on exit from SCAN.

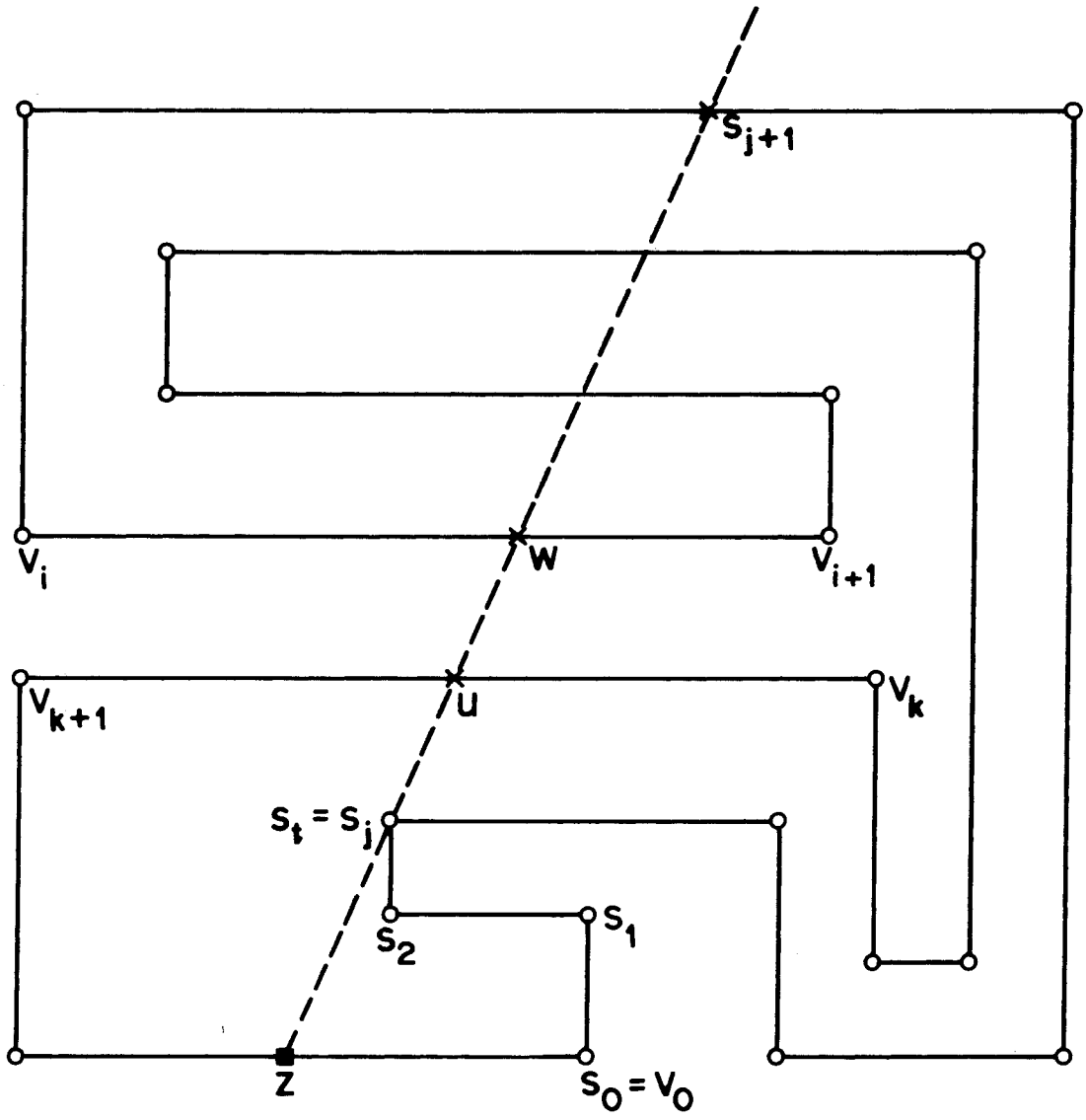


Figure 3.4 Illustration for remark 5 (RETARD, SCAN).
The values of $j=3=t$, s_t, $i=9$, v_i, w are set as on exit from RETARD.
The values of $k=17$, and v_k are as on exit from SCAN.

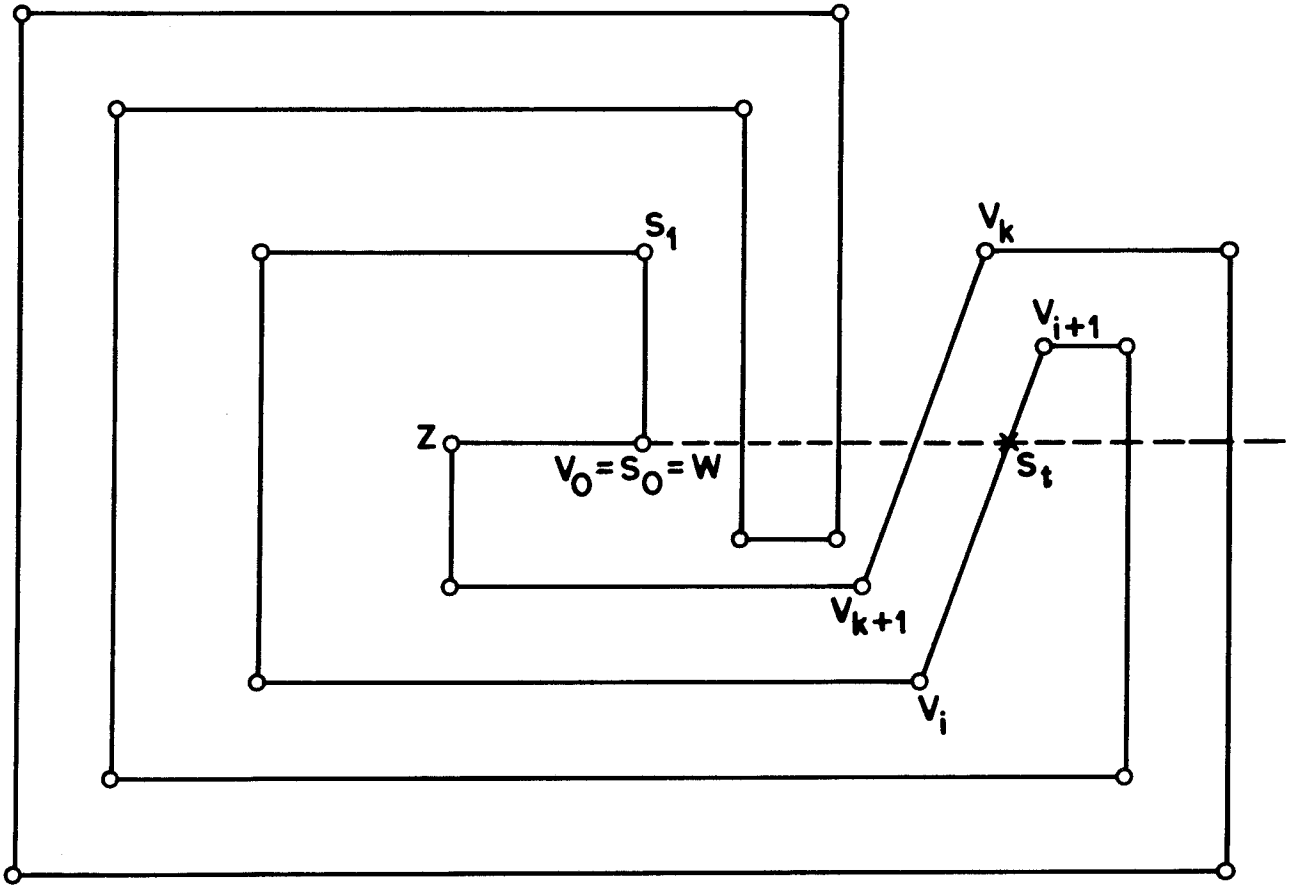


Figure 3.5 Illustration for remark 6 (ADVANCE, SCAN).
The values of $i=4$, v_i, w, $t=5$, s_t are set as on exit from ADVANCE.
The values of $k=18$ and v_k are as on exit from SCAN.

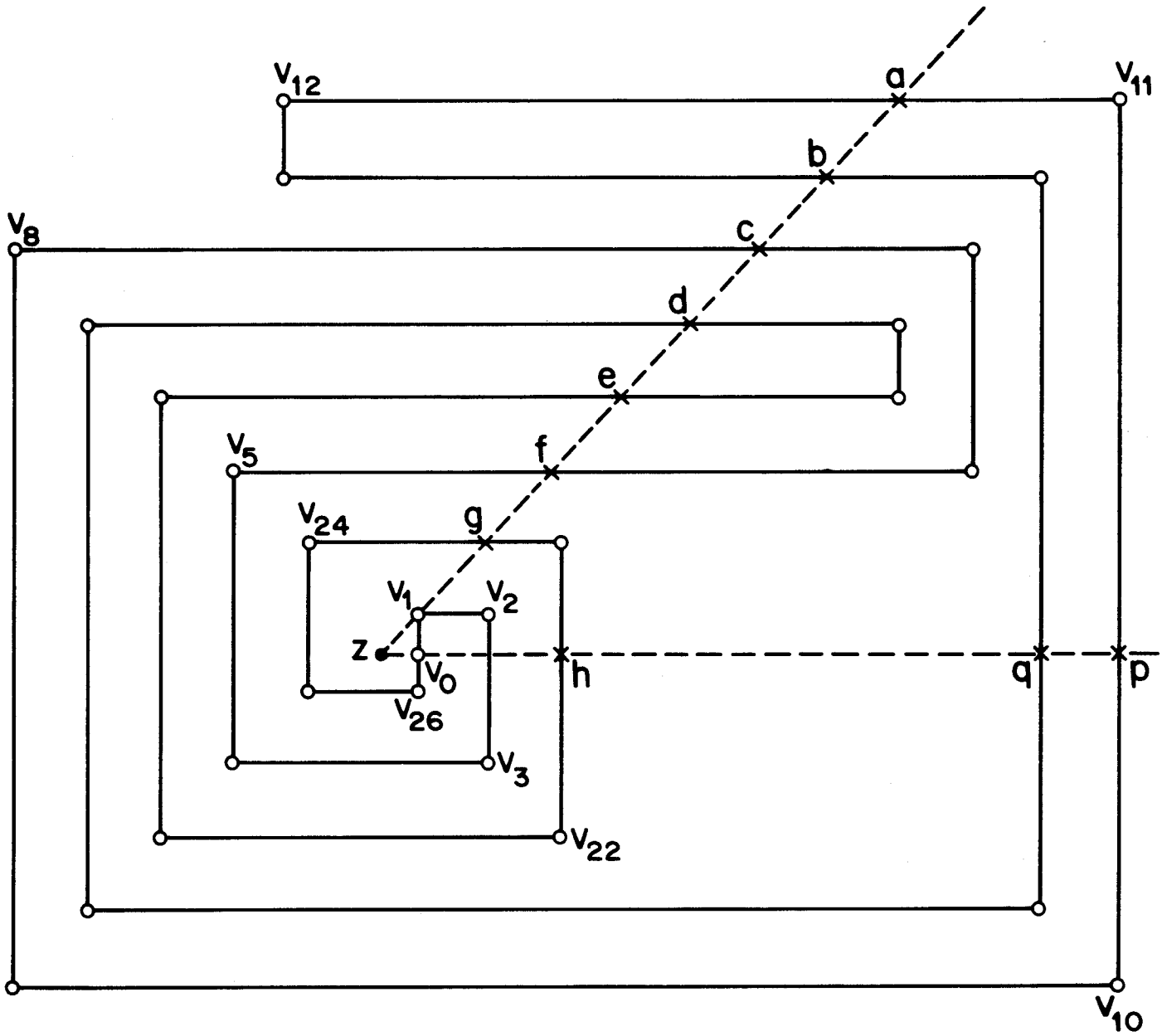


Figure A.1 Example for which the algorithms of Lee and El Gindy and Avis fail.