AN OPTIMAL ALGORITHM FOR DETECTING WEAK VISIBILITY OF A POLYGON
(Preliminary Version)

by Jorg-R. Sack
and Subhash Suri

SCS-TR-114

December 1986

School of Computer Science
Carleton University
Ottawa, Ontario
CANADA K1S 5B6

This research was supported in part by NSERC.

An Optimal Algorithm for Detecting Weak Visibility of a Polygon
(Preliminary Version)

Jörg-R. Sack
School of Computer Science
Carleton University, Ottawa K1S 5B6, Canada

Subhash Suri+
Bell Communications Research
435 South Street
Morristown, N.J. 07960, USA

Abstract

In 1981, Avis and Toussaint gave a linear-time algorithm for the following problem: Given a simple n-vertex polygon P and an edge of P, determine whether each point in P can be seen by some (not necessarily the same) point on the edge. They posed the more general problem of finding a sub-quadratic algorithm for determining whether such an edge exists. In this paper, we present a linear-time algorithm for determining all (if any) such edges of a given simple polygon.

I Introduction

The notion of visibility underlies many applications, such as image processing, robotic control, computer graphic etc. In this paper, we investigate a question pertaining to visibility in simple polygons. Two points in a polygon are considered (*mutually*) *visible* if the line segment joining them does not intersect the exterior of the polygon. If a point x is visible from all other points of the polygon P, then x is called a *star* of P and the set of all stars of P is called the *kernel* of P; the kernel may be empty. A linear-time algorithm for computing the kernel has long been known [LP79].

In this paper, we are concerned with a more general notion of visibility, introduced by Avis and Toussaint [AT81]: A point of a polygon P is *weakly visible* from an edge e if there is a point y on e that can see x. An edge e is called a (*weak-*) *visibility edge* if e can weakly see the entire polygon. We solve the problem of determining all visibility edges of a given polygon in linear time. Previously, only an $O(n^2)$ algorithm was known which followed from Avis and Toussaint's linear time test to check whether a given polygon is weakly visible from a given edge.

There are many problems in computational geometry for which linear-time algorithms exist provided that the input polygon is triangulated (e.g. shortest path, visibility [GHLST86], minimum link distance [Su87], polygon separability [ST85]). At present, the best triangulation algorithm for a simple n-vertex polygon runs in

---

+ This work was done while the second author was with Johns Hopkins University and visited Carleton University.

O(n loglog$n$) time [TV86]), while a linear-time algorithm is known for triangulating weakly visible polygons [TA82]. The triangulation algorithm for weakly visible polygons requires the knowledge of a least one visibility edge. Our linear-time algorithm detects whether a given simple polygon is weakly visible and reports all visibility edges. Thus the algorithms mentioned above can be speeded up if the polygon turns out to be weakly edge-visible.

The paper is organized as follows, in Section II, we introduce some notation and derive a theorem which, using a recent result of Chazelle and Guibas [CG85], enables us to design an O(n log$n$) algorithm for reporting all visibility edges of a given n-vertex polygon. The remainder of the paper is devoted to improving this bound to O(n). In Section III, we solve this problem for polygons with at least one given visibility edge. We first assume that this edge has two convex endpoints. Subsequently, we show how to drop the latter restriction. In Section IV, we deal with the general case of detecting weak edge-visibility of an arbitrary simple polygon and reporting all visibility edges. The solution is partly a reduction to the previously discussed special cases.

II. Preliminaries

Let P be a closed simply connected planar region, whose boundary is given by a polygon of n sides. The boundary of P will be denoted by δP. Let $\{p_1, p_2, \ldots, p_n\}$ and $\{e_1, e_2, \ldots, e_n\}$, respectively, be the vertices and the edges of P listed in a counterclockwise order, where e_i joins p_i and p_{i+1} ; e_i will also denote the vector directed from p_i to p_{i+1} and e_i^{-1} the vector directed from p_i to p_{i-1} . Let $\Lambda(x,y)$ denote the counterclockwise path along δP from the point x to y , where $x,y \in \delta P$. A (point, direction)-pair, (x,u), consists of a point x and a direction u such that the ray originating from x in the direction u is locally directed towards the interior of P. Let x be a point on an edge e_i of P. We define $\sigma(x,e_i)$ (resp. $\sigma(x,e_i^{-1})$) to be p_{i+1} (resp. p_{i-1}) if p_{i+1} (resp. p_{i-1}) is a convex vertex; otherwise, $\sigma(x,e_i)$ (resp. $\sigma(x,e_i^{-1})$) is defined as the first point on δP hit by the ray from x in the direction e_i (resp. e_i^{-1}). The latter definition is also used to define $\sigma(x,u)$, for directions u not co-linear with e_i. An edge $e \in \Lambda(x,\sigma(x,u))$ is called a proper edge of $\Lambda(x,\sigma(x,u))$ if e does not share a point with the ray originating from x in the direction u. Let $\chi(x,u)$ and $\chi^{-1}(x,u)$, respectively, be the set of proper edges in $\Lambda(x,\sigma(x,u))$ and $\Lambda(\sigma(x,u),x)$ (see Figure 1). The following fact can be easily established.

Lemma 1: *The vertex p_i is invisible from each edge in $\chi(p_i, e_i) \cup \chi^{-1}(p_i, e_i^{-1})$.*

For a point $z \in P$, the *visibility polygon* of z, $V(z)$ is the closed set of points in P that are visible from z; if the sides of P are opaque, $V(z)$ is the region of P illuminated by a light source placed at z. An edge on the boundary of $V(z)$ is called a *window* if it does not belong to δP: Windows of $V(z)$ delimit the illuminated regions from the dark ones in P. Note that if $w = ab$ is a window of $V(z)$ such that a lies in the relative interior of the segment (z,b) (for short denoted by zb) then a is a *reflex* vertex of P. Any window w partitions P into two polygons one of which contains the point z; we denote the other such sub-polygon by *ear(w)* of P.

Lemma 2: *Let $z \in \delta P$ be a point and let e be an edge of P invisible from z. Then, there exists a vertex $p_\alpha \in P$, $1 \leq \alpha \leq n$, such that $e \in \chi(p_\alpha, e_\alpha) \cup \chi^{-1}(p_\alpha, e_\alpha^{-1})$.*
Proof Since e is invisible from z, we must have $e \subseteq P - V(z)$. Let $ear(ab)$ be the unique *ear* containing e, where $a = p_j$ is a reflex vertex of P and lies in the relative interior of the segment zb. In the counterclockwise traversal of $V(z)$ starting from z, if a precedes b then $e \in \chi(p_{j-1}, e_{j-1})$ and if b precedes a then $e \in \chi^{-1}(p_{j+1}, e_{j+1}^{-1})$ (see Figure 2). Clearly, we can choose p_α to be p_{j-1} in the former and p_{j+1} in the latter case. This completes the proof. ◊

The following theorem completely characterizes the edges from which P is weakly visible and also leads to a fast method of identifying them. The proof of this theorem follows rather easily from Lemma 2. First, some more notation is introduced. Let E denote the set of edges of P. Let $X = \cup_{1 \leq i \leq n} \chi(p_i, e_i)$ and $X^{-1} = \cup_{1 \leq i \leq n} \chi^{-1}(p_i, e_i^{-1})$.

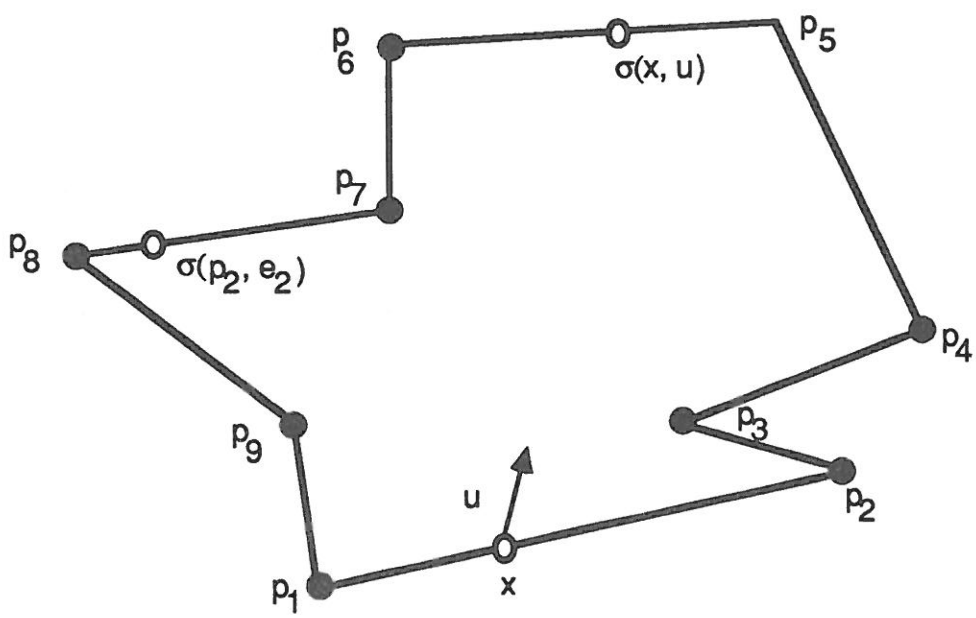

Figure 1: Giving an example for $\chi(x, u) = \{e_2, e_3, e_4\}$, $\chi^{-1}(x, u) = \{e_6, e_7, e_8, e_9\}$, $\chi(p_2, e_2) = \{e_4, e_5, e_6\}$, and $\chi^{-1}(p_2, e_2^{-1}) = \{\}$.

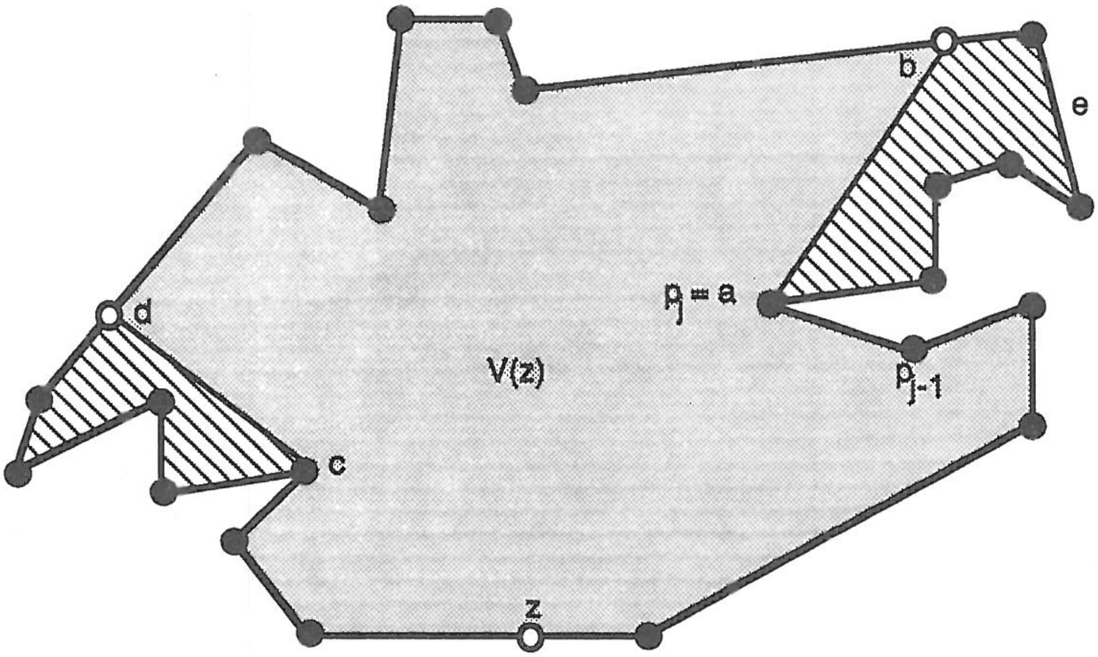

Figure 2: Illustrating the visibility polygon $V(z)$ from a point z and its pockets (shaded).

Theorem 1: *P is weakly visible from an edge e_i if and only if $e_i \in E - X \cup X^{-1}$.*

Proof $\Rightarrow$ Suppose P is weakly visible from an edge e_i, for $1 \le i \le n$. Consider $p_j \in P$. Since $\chi(p_j, e_j) \cup \chi^{-1}(p_j, e_j^{-1})$ consists of only those edges that are invisible from p_j, and e_i is visible from p_j by the hypothesis, we must have $e_i \notin \chi(p_j, e_j) \cup \chi^{-1}(p_j, e_j^{-1})$. Since p_j was chosen arbitrarily, it follows that $e \notin X \cup X^{-1}$ and the proof is complete in one direction.

$\Leftarrow$ Let $e_i \in E - X \cup X^{-1}$ be an edge. We show that P is weakly visible from e_i. It is known that a simple polygon is weakly visible from an edge if and only if the boundary of the polygon is weakly visible from that edge. Hence, we only have to show that δP is weakly visible from e_i. Our proof is by *co-linear.* Suppose $z \in \delta P$ is a point invisible from e_i. Lemma 2 guarantees the existence of a vertex $p_\alpha \in P$ such that $e_i \in \chi(p_\alpha, e_\alpha) \cup \chi^{-1}(p_\alpha, e_\alpha^{-1})$ which contradicts our choice of e_i. ◊

Using Theorem 1, we can easily design an $O(n \log n)$ algorithm for computing X and X^{-1} as follows: Using a result of Chazelle and Guibas [CG85], we can preprocess P in $O(n \log n)$ to obtain a data structure that computes $\sigma(p_i, e_i)$ in $O(\log n)$ time. The indices of the edges of $\chi(p_i, e_i)$, for $i = 1, 2, \ldots, n$, define a connected interval between 1 and n, where the endpoints of the interval are computable from $\sigma(p_i, e_i)$ and p_i in O(1) time. Hence, X can be computed in O(n logn) time by simply finding the union of all such intervals. Similarly, for X^{-1}. And thus the set of visibility edges can be determined in O(n log$n$). In the next section, we present a more complicated O(n) algorithm for computing the weakly visible edge.

III Solving the Problem for Edge-Visible Polygons

In this section we show how to compute, in linear time, all visibility edges of a polygon P provided that at least one such edge is known. The discussion restricts itself to computing X since X^{-1} can be determined similarly. For ease of notation we assume edge e_1 to be the edge from which P is weakly visible. Let us first restrict ourselves to the case where both vertices of edge e_1 are convex; we refer to such polygons as edge-visible polygons with convex base e_1. Later we show how to overcome this restriction.

To compute X, we linearly scan δP, processing the pairs (p_i, e_i) in order of increasing i and update X whenever necessary. In the following, all traversals of δP are assumed to be counterclockwise, unless otherwise specified. We call a pair (p_i, e_i) *non-trivial* if $\sigma(p_i, e_i) \neq p_{i+1}$, otherwise (p_i, e_i) is *trivial*. Clearly, whether a pair is trivial can be checked in constant time. During the algorithm, we move a segment z_1z_2 around δP with $z_1 = p_{i-1}$ and $z_2 = \sigma(p_{i-1}, e_{i-1})$ or a point co-linear with e_{i-1} (to be specified later), where (p_{i-1}, e_{i-1}) is the last non-trivial pair processed by the algorithm. At any time during the execution of the algorithm, we maintain that all proper edges of $\Lambda(z_1, z_2)$ are currently in X. Initially, we set $z_1 = p_1$, $z_2 = p_2$ and $X = \emptyset$. In general, let (p_i, e_i) be the next (unprocessed) pair considered by the algorithm. If either (p_i, e_i) is trivial, or $\sigma(p_i, e_i)$ lies in $\Lambda(z_1, z_2)$, we simply advance to the next pair. Note that, in this case, all proper edges of $\chi(p_i, e_i)$ must be already in the current X. Otherwise, z_1 and z_2 are advanced on δP and we update X to include all proper edges encountered on the scan from the current position of z_2 to its new position. Our algorithm is described below in pseudo-code.

Algorithm A: Compute X for edge-visible polygons with convex base

input: A simple polygon P known to be weakly visible from the convex base e_1
output: All visibility edges of P

Initialization

$z_1 := p_1$; $z_2 := p_2$;

General Step
for i := 2 to n **do**
if (both endpoints of e_i are on the closed left half-plane defined by z_1 and z_2)
or (half-ray(p_i, e_i) intersects the segment(z_1, z_2) at a point other than p_i)
then
scan the boundary of P starting at z_2 until a point z on the half-ray(p_i, e_i) (but not on e_i) is encountered;
place all those proper edges of $\Lambda(p_{i+1}, z)$ into X that have been scanned;
$z_1 := p_i$; $z_2 := z$

Lemma 3: *Let P be an edge-visible polygon with convex base e_1. Then*
(1) $\sigma(p_i, e_i)$ equals the first point co-linear with e_i in a ***counterclockwise*** *traversal of δP starting at p_i.*
(2) $\sigma(p_i, e_i^{-1})$ equals the first point co-linear with e_i in a ***clockwise*** *traversal of δP starting at p_i.*

Proof (1) Suppose that $\sigma(p_i, e_i)$ is not the first point co-linear with e_i in a counterclockwise traversal of δP starting at p_i. Then the open path $\Lambda(p_i, \sigma(p_i, e_i))$ intersects the ray from p_i in the direction of e_i at least twice, thereby creating a region of points which are invisible from e_1.
(2) follows analogously. ◊

Time Complexity of Algorithm-A

Each execution of the general step considers a previously unprocessed pair (p_i, e_i) and may also advance z_1 and z_2 on δP. There are only n pairs (p_i, e_i). The weak-visibility of P from edge e_1 guarantees that e_1 is not in $\chi(p_i, e_i)$, for i=1,2,...,n, which means that z_1 and z_2 go around δP at most once. Hence Algorithm-A runs in linear time.

Correctness of Algorithm-A

Let p_i be the first non-trivial vertex examined by Algorithm-A. By Lemma 3, the scan-step starts at z_2 (initially equal to p_2), and advances counter- clockwise along the boundary of P so as to find correctly $\sigma(p_i, e_i)$. All proper edges encountered during that scan are placed correctly onto X (Lemma 1). z_1 are z_2 are assigned to p_i, $\sigma(p_i, e_i)$, respectively. Thereby P is partitioned into two edge-visible polygons $P_z = (z_1, p_{i+1}, \ldots, z_2)$ and closure($P - P_z$) each having a convex base (the convex base of polygon P_z is $z_1 z_2$). Consider now a non-trivial vertex p_j in P_z. Since P is weakly visible from e_1, for such a vertex p_j, $\sigma(p_j, e_j)$

lies either in P_Z or on the chain $\Lambda(z_2, p_1)$. To ensure linearity of the algorithm repeated boundary scans must be avoided to distinguish between these cases. The algorithm performs an O(1) test checking whether the ray(p_j,e_j) intersects $z_1 z_2$. If the test is false then clearly $\sigma(p_j,e_j)$ is inside P_Z. If $\sigma(p_j,e_j)$ is on the chain $\Lambda(z_2, p_1)$ then the subsequent scan of the boundary starting at z_2 correctly identifies $\sigma(p_j,e_j)$, i.e. $z = \sigma(p_j,e_j)$ (by Lemma 3). Otherwise, $\sigma(p_j,e_j)$ is inside P_Z and all edges of $\chi(p_j,e_j)$ are already in X. Algorithm-A, however, continues to scan the boundary of P starting at z_2 until $z(p_j,e_j)$ $(\neq\sigma(p_j,e_j))$ is encountered. Since z_2 is initially to the right of the ray(p_j,e_j) all proper edges encountered during this scan are on the right half-plane of e_j and thus cannot see p_j. By Lemma 2, there exist a vertex p_α such that these edges are in $\chi(p_\alpha, e_\alpha)$ which guarantees the correctness of this step. See Figure 3 for an illustration. In general, the segment $z_1 z_2$ may create more than one pocket of P, but since the analysis is similar to the case discussed here it is omitted.

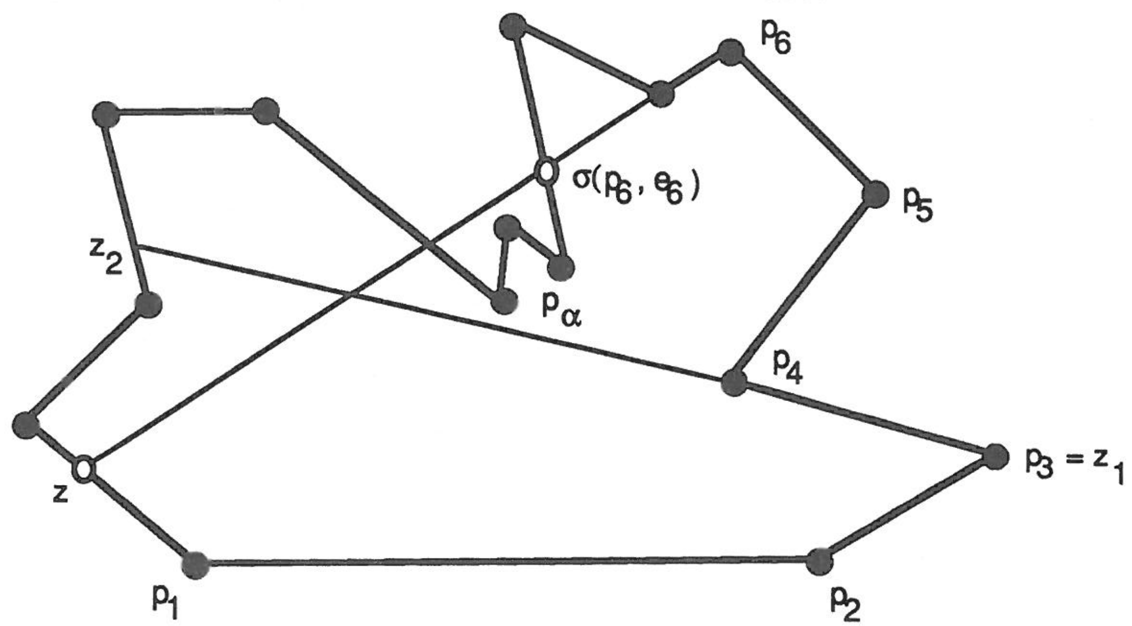

Figure 3: Algorithm-A advances from z_2 to $z \neq \sigma(p_6,e_6)$.

Theorem 2: Let P be a simple edge-visible polygon with a convex base. Then its set of visibility edges can be determined in time, linear in the number of vertices of P.

Proof The visibility edges of P are the edges of E-$(X \cup X^{-1})$. By the above, Algorithm-A correctly determines X, in linear time; X^{-1} can be computed in a similar way. The set operations can be performed in linear time. ◊

Next, we show that Algorithm-A can be modified to report all weak visibility edges for polygons P that are weakly visible from a known edge, say e_1, whose (one or both) endpoints may be reflex vertices. Without loss of generality, assume both endpoints of e_1 to be reflex. We can partition P into three edge-visible polygons P_o, P_1, P_2 each of which having a convex base. The main part is then to discuss the interaction of vertices from one polygon and their invisible edges in another polygon. For this, let $q_s = \sigma(p_2, e_2^{-1})$ and $q_r = \sigma(p_1, e_1)$, where q_s is on e_s and q_r is on e_r. Let P_o, P_1, and P_2 be defined as follows:

$$P_o = (p_1, p_2, q_r, p_{r+1}, \ldots, p_s, q_s),$$
$$P_1 = (p_1, q_s, p_{s+1}, \ldots, p_n), \text{ and}$$
$$P_2 = (p_2, p_3, \ldots, p_r, q_r).$$

Clearly P_1 and P_2 are star-shaped from p_1 and p_2, respectively. Note that hereby a weakly edge visible polygon P is partitioned into (at most) three edge-visible sub-polygons, P_o, P_1, and P_2 each of which having a convex base.

Proposition 1: *All proper edges of P_1, P_2 are in $X \cup X^{-1}$.*
Proof The proper edges of P_2, i.e. all those edges of P_2 which have none of its endpoints in P_o, are in $\chi(p_1, e_1)$. The proper edges of P_1, i.e. all those edges of P_1 which have none of its endpoints in P_o, are in $\chi^{-1}(p_2, e_1^{-1})$. Thus the result follows.

Proposition 2: For all p_i located on proper edges of P_1, $\sigma(p_i, e_i)$ is in P_1 and for p_n, $\sigma(p_n, e_n)$ is in P_o.
Proof P_1 is star-shaped from p_1 thus p_1 lies on the left half-plane defined by (p_i, $\sigma(p_i, e_i)$), for each vertex p_i, i=s+1,...,n. ◊

By Proposition 1, we can restrict ourselves to examining those vertices of P_1 and P_2 for which $\sigma(p_i, e_i)$ is in P_o. By Proposition 2, this implies that only one of the vertices of P_1, i.e. the vertex p_n in P_1 needs to be examined to compute X for P_1. Analogously for the computation of X^{-1} for vertices in P_2. Both computations can be done in linear time; it remains to compute:

(a) X, X^{-1} for P_o,
(b) X for all vertices p_i in P_2 with respect to $P_2 \cup P_o$, and
(c) X^{-1} for all vertices p_i in P_1 with respect to $P_1 \cup P_o$.

P_0 is an edge-visible polygon with a convex base thus, by Theorem 2, X and X^{-1} can be computed in linear time (case(a)). The analysis for cases (b) and (c) is quite similar to that of Algorithm-A; it is thus omitted here.

Theorem 3: Let P be an n-vertex polygon weakly visible from some known edge. Then all visibility edges of P can be computed in O(n) time.
Proof We note that an analogue to Lemma 3 holds for arbitrary edge-visible polygons, i.e. for all p_i in P_2, $\sigma(p_i, e_i)$ equals the first point co-linear with e_i in a counterclockwise traversal of δP starting from p_i. The remainder then follows from Propositions 1 and 2, and the above discussion. ◊

IV Computing all visibility edges for an arbitrary simple polygon

To solve the general case of computing all edges from which an arbitrary simple polygon is weakly visible, we partition P into several regions, each of which is weakly visible from some edge. Let $x = p_x$ be a vertex of P and $V(x)$ the visibility polygon from x with respect to P, constructed in linear time using e.g. [Le83]. Let $(r_1,\ldots,r_p,l_1,\ldots,l_s)$ be the windows of $V(x)$, where

(a) For $i=1,\ldots,p$, $r_i = a_ib_i$ such that a_i lies in the relative interior of the segment(x,b_i) and cuts off a right pocket, $R_i = (a_i,\ldots,b_i)$, whose edges lie to the right of the ray(x,b_i) originating at x through b_i. For $j=1,\ldots,s$, $l_j = c_jd_j$ such that c_j lies in the relative interior of the segment(x,d_j) and cuts off a left pocket, $L_j = (d_j,\ldots,c_j)$, whose edges lie to the left of the ray(x,d_j).
(b) $k_1<k_2$ implies that l_{k1} precedes l_{k2} (respectively, r_{k1} precedes r_{k2}) in the counterclockwise traversal of $V(x)$.

A counterclockwise traversal of $V(x)$ may encounter the windows in order as:
(0) $r_1,\ldots,r_p,l_1,\ldots,l_s$ (0-switch case)
(1) $r_1,\ldots,r_{p-1},l_1,r_p,l_2,l_3,\ldots,l_s$ (1-switch case)
The 1-switch case is easier, since, to see both pockets R_p and L_1, an edge must extend from the left half-plane defined by segment(x,d_1) to the right half-plane defined by segment(x,b_p). Additionally, it has to see $\Lambda(c_1,a_p)$ and the edges containing x. Thus at most three edges are candidates: the edges containing x and the edge containing c_1 and a_p (if any). In linear time, using the algorithm by Avis and Toussaint, P can be tested for weak visibility from each of these three edges. If the left- and right-window arrangement alternates more than once the algorithm may terminate returning that P is not weakly edge-visible. The edges incident to x are candidates for visibility edges only if at least one of L or R is empty, i.e. left edge at x is a candidate only if L is empty and right edge is a candidate only if R is empty.

We restrict ourselves for the remainder of this paper to the more interesting 0-switch case. We partition P into three polygons $P_{\text{int}} = (x,b_p,\ldots,d_1)$, $L = (d_1,\ldots,x)$, and $R = (x,\ldots,b_p)$. See Figure 4. (Strictly speaking, both L and R are themselves composed of two polygons since the boundaries touch at c_1 and a_p, respectively. For ease of notation this is ignored here.)

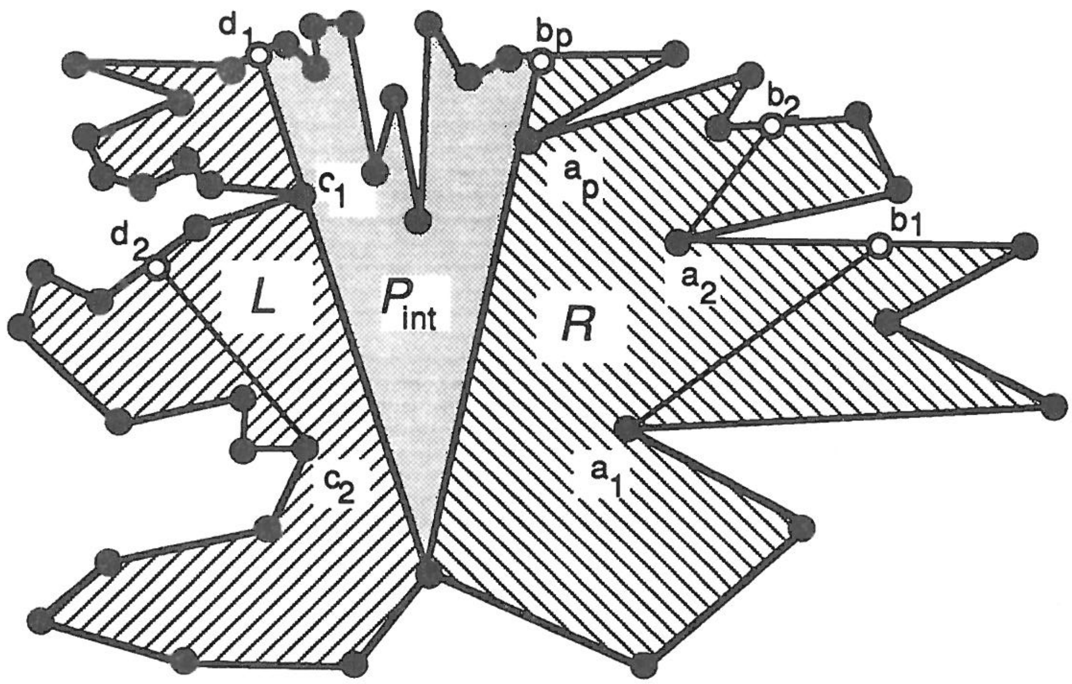

Figure 4: A polygon P and its partitioning into the three polygons P_{int}, L, R.

Lemma 4: Let e (v) denote the edge(s) containing the point (vertex) v. *Let* e_k *be a visibility edge of P with at most one endpoint in* P_{int}. *Then* $e_k \in Q = \{e(c_1), e(d_1), e(x), e(a_p), e(b_p)\}$.
Proof To weakly see both sub-polygons $(d_1,\ldots,c_1)$ and $(c_1,\ldots,x)$ of polygon L, an edge e_k must lie (at least partially) in the closed right half-plane defined by the ray(x,d_1). Thus among all edges with at least one endpoint in L only edges $\{e(c_1), e(d_1), e(x)\}$ are candidates. Analogously for the edges $e(a_p)$, $e(b_p)$ ◊

Lemma 5: The polygons P_{int}, L, R (defined above) form a partition of P. If e_k is a visibility edge of P with both endpoints in P_{int}, then P_{int}, L, R are weakly visible from edges $e(x)$, xd_1, and xb_p, respectively.
Proof Omitted.

To compute the set of visibility edges, $W(P) = X \cup X^{-1}$, for P, we use Lemma 4 and 5, as follows:

<u>Step (1)</u> Choose a vertex x of P and compute $V(x)$
<u>Step (2)</u> If $V(x) = P$ then $W(P)$ can be computed in linear time by Theorem 3

Step (3) Otherwise, compute the set Q. For each edge $e \in Q$ test whether P is weakly visible from e using the linear-time algorithm of Avis and Toussaint.

Step (4) Let $e \in Q$ be any edge from which P is weakly visible (as obtained from (3)). Using Theorem 3, in linear time, $W(P)$ can be computed and the algorithm terminates.

Otherwise, no edge in Q is a visibility edge for P. Then $W(P)$ consists of only those edges (if any) whose both endpoints are in P_{int} (cf Lemma 4) and the necessary condition for weak visibility of P, as obtained from Lemma 5 needs be tested.

Step (5) Test in linear-time, whether L and R are weakly visible from the segments xd_1, and xb_p, respectively. If one of the tests should fail the algorithm terminates and reports that P is not weakly visible.

Step (6) Otherwise, compute the set of visibility edges $W(L \cup P_{int})$ and $W(R \cup P_{int})$ for polygons $L \cup P_{int}$ and $R \cup P_{int}$, respectively. Any edge in both weak-visibility sets is then reported to be in $W(P)$.

As will be shown in the remainder of this section also the final step (6) can be performed in linear time. Since the computations of $W(L \cup P_{int})$ and $W(R \cup P_{int})$ are symmetric, we discuss w.l.o.g. the problem of determining $W(L \cup P_{int})$. $W(P_{int})$ can be determined in linear time (Lemma 5 and Theorem 3). If $W(P_{int})$ is empty the algorithm may terminate reporting that P is not weakly visible. Otherwise, consider polygon $L \cup P_{int}$. The set $W(L \cup P_{int})$ consists of all those edges in $W(P_{int})$ which are not in $\chi(p_k, e_k) \cup \chi^{-1}(p_k, e_k^{-1})$ for any vertices p_k in L. This is to be computed next.

Let p_k be a non-trivial vertex in L. Since P_{int} as well as L are weakly visible from the segment(x,d_1) we can apply Lemma 3 with respect to polygon $P_{int} \cup L$. We get:

(a) if $\sigma(p_k,e_k) \in P_{int}$ then $\sigma(p_k,e_k)$ is the first point co-linear with e_k on a **clockwise** traversal of $P_{int} \cup L$ starting at p_k (or, if wanted starting at d_1).

(b) if $\sigma(p_k,e_k) \in L$ then $\sigma(p_k,e_k)$ is the first point co-linear with e_k on a **counter-clockwise** traversal of $P_{int} \cup L$ starting at p_k.

(c) if $\sigma(p_k,e_k^{-1}) \in P_{int}$ then $\sigma(p_k,e_k^{-1})$ is the first point co-linear with e_k on a **clockwise** traversal of $P_{int} \cup L$ starting at p_k.

(d) if $\sigma(p_k,e_k^{-1}) \in L$ then $\sigma(p_k,e_k^{-1})$ is the first point co-linear with e_k on a **clockwise** traversal of $P_{int} \cup L$ starting at p_k.

Thus for the computation of $\cup \chi^{-1}(p_k, e_k^{-1})$ for all vertices p_k in L (cases (c), (d)), the same analysis as of Algorithm-A holds (running Algorithm-A clockwise), i.e. if

$z = \sigma(p_k, e_k)$ then the edges reported invisible are correct, otherwise, the false edges reported to be in $\chi(p_k, e_k)$, indeed are invisible from another vertex p_j (Lemma 3).

The more complicated case of computing $\cup\, \chi(p_k, e_k)$, for all vertices p_k in L, is discussed now. The difficulty arises from the fact that to find $\sigma(p_k, e_k)$ the orientation of the traversal depends on whether $\sigma(p_k, e_k)$ is in L or not. Testing these for each vertex p_k may require quadratic time. We can apply Algorithm-A moving counterclockwise around the boundary of $P_{int} \cup L$ starting at d_1, with $z_1 = d_1$ and $z_2 = x$, provided we take an additional step described next.

If point z in Algorithm-A equals $\sigma(p_k, e_k)$ for all vertices p_k in L then the result will be correct. Otherwise, the segment $z_1 z_2$ is intersected by δP and our algorithm may fail to report some edges that are actually invisible from p_k. Let X^A (X^L) denote the set of edges of P_{int} reported by our algorithm as invisible (actually invisible) from the vertices p_k in L. It is easily seen that $X^L \supseteq X^A$. Using Lemma 2, we can prove that all but at most two edges of $X^L - X^A$ belong to $X(P_{int}) \cup X^{-1}(P_{int})$. Moreover, the missing edges can be found in linear time (e.g. by an additional clockwise scan of the candidate edges of P_{int}). We use the algorithm of Avis and Toussaint [AT81] from each of these (at most) two edges to see whether P is weakly visible from them. Thus we get:

Theorem 4: Given a simple polygon on n vertices, all the edges, possibly none, from which P is weakly visible can be found in optimal $O(n)$ time and space.

Bibliography

[AT81] Avis D. and G.T. Toussaint, "An optimal algorithm for determining the visibility polygon from an edge", *IEEE Transaction on Computers*, Vol. C-30, No. 12, pp. 910-914 (1981).

[CG85] Chazelle B. and L.J. Guibas, "Visibility and intersection problems in plane geometry", *Proc. First ACM Symposium on Computational Geometry*, pp. 135-146 (1985).

[EA81] ElGindy H. and D. Avis, "A linear algorithm for computing the visibility polygon from a point", *Journal of Algorithms*, 2, pp. 186-197 (1981).

[GHLST86] Guibas, L., J.Hershberger, D.Leven, M. Sharir, and R. Tarjan, "Linear-time algorithms for visibility and shortest path problems inside a triangulated simple polygon", Tech. Rept. 218, Computer Science Dept., Courant Institute (1986).

[Le83] Lee D.T., "Visibility if a simple polygon", *Computer Vision, Graphics and Image Processing*, 22, pp. 207-221 (1983).

[LP79] Lee D.T. and F. Preparata, "An optimal algorithm for finding the kernel of a polygon", *Journal of the ACM*, 26, pp. 415-421 (1979).

[OR87] O'Rourke J, *Art-Gallery Theorems* and Algorithms, Oxford University Press (1987).

[Sa84] Sack J.-R. "Rectilinear computational geometry", Tech. Rept. SCS-TR54, School of Computer Science, Carleton University (1984).

[ST85] Sack J.-R. and G.T. Toussaint, "Translating Polygons in the Plane", Proc. STACS 1985, *Lecture Notes in Computer Science* 182, Springer, Berlin Heidelberg New York Tokyo, 1985, pp. 310-321.

[Su87] Suri, S., "A polygon partitioning technique for link distance problems", Ph.D. Thesis, Dept. of Computer Science, Johns Hopkins University (1987).

[TA82] Toussaint G.T. and D. Avis, "On a convex hull algorithm for polygons and its applications to triangulation problems", *Pattern Recognition,* Vol. 15, No, 1, pp. 23-29 (1982).

[TV86] Tarjan R. and C. Van Wyk, "An O(n loglog n) algorithm for triangulating simple polygons," preprint (submitted to *SIAM J. Computing*) (1986).

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-66 **On the Futility of Arbitrarily Increasing Memory Capabilities of Stochastic Learning Automata**
B.J. Oommen, October 1984. Revised May 1985.

SCS-TR-67 **Heaps in Heaps**
T. Strothotte, J.-R. Sack, November 1984. Revised April 1985.

SCS-TR-68 **Partial Orders and Comparison Problems**
out-of-print M.D. Atkinson, November 1984. See Congressus Numerantium 47 ('86), 77-88

SCS-TR-69 **On the Expected Communication Complexity of Distributed Selection**
N. Santoro, J.B. Sidney, S.J. Sidney, February 1985.

SCS-TR-70 **Features of Fifth Generation Languages: A Panoramic View**
Wilf R. LaLonde, John R. Pugh, March 1985.

SCS-TR-71 **Actra: The Design of an Industrial Fifth Generation Smalltalk System**
David A. Thomas, Wilf R. LaLonde, April 1985.

SCS-TR-72 **Minmaxheaps, Orderstatisticstrees and their Application to the Coursemarks Problem**
M.D. Atkinson, J.-R. Sack, N. Santoro, T. Strothotte, March 1985.

SCS-TR-73 **Designing Communities of Data Types**
Wilf R. LaLonde, May 1985.
Replaced by SCS-TR-108

SCS-TR-74 **Absorbing and Ergodic Discretized Two Action Learning Automata**
out-of-print B. John Oommen, May 1985. See IEEE Trans. on Systems, Man and Cybernetics, March/April 1986, pp. 282-293.

SCS-TR-75 **Optimal Parallel Merging Without Memory Conflicts**
Selim Akl and Nicola Santoro, May 1985

SCS-TR-76 **List Organizing Strategies Using Stochastic Move-to-Front and Stochastic Move-to-Rear Operations**
B. John Oommen, May 1985.

SCS-TR-77 **Linearizing the Directory Growth in Order Preserving Extendible Hashing**
E.J. Otoo, July 1985.

SCS-TR-78 **Improving Semijoin Evaluation in Distributed Query Processing**
E.J. Otoo, N. Santoro, D. Rotem, July 1985.

SCS-TR-79 **On the Problem of Translating an Elliptic Object Through a Workspace of Elliptic Obstacles**
B.J. Oommen, I. Reichstein, July 1985.

SCS-TR-80 **Smalltalk - Discovering the System**
W. LaLonde, J. Pugh, D. Thomas, October 1985.

SCS-TR-81 **A Learning Automation Solution to the Stochastic Minimum Spanning Circle Problem**
B.J. Oommen, October 1985.

SCS-TR-82 **Separability of Sets of Polygons**
Frank Dehne, Jörg-R. Sack, October 1985.

SCS-TR-83 out-of-print **Extensions of Partial Orders of Bounded Width**
M.D. Atkinson and H.W. Chang, November 1985. See Congressus Numerantium, Vol. 52 (May 1986), pp. 21-35.

SCS-TR-84 **Deterministic Learning Automata Solutions to the Object Partitioning Problem**
B. John Oommen, D.C.Y. Ma, November 1985

SCS-TR-85 out-of-print **Selecting Subsets of the Correct Density**
M.D. Atkinson, December 1985. To appear in Congressus Numerantium, Proceedings of the 1986 South-Eastern conference on Graph theory, combinatorics and Computing.

SCS-TR-86 **Robot Navigation in Unknown Terrains Using Learned Visiblity Graphs. Part I: The Disjoint Convex Obstacles Case**
B. J. Oommen, S.S. Iyengar, S.V.N. Rao, R.L. Kashyap, February 1986

SCS-TR-87 **Breaking Symmetry in Synchronous Networks**
Greg N. Frederickson, Nicola Santoro, April 1986

SCS-TR-88 **Data Structures and Data Types: An Object-Oriented Approach**
John R. Pugh, Wilf R. LaLonde and David A. Thomas, April 1986

SCS-TR-89 **Ergodic Learning Automata Capable of Incorporating Apriori Information**
B. J. Oommen, May 1986

SCS-TR-90 **Iterative Decomposition of Digital Systems and Its Applications**
Vaclav Dvorak, May 1986.

SCS-TR-91 **Actors in a Smalltalk Multiprocessor: A Case for Limited Parallelism**
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-92 **ACTRA - A Multitasking/Multiprocessing Smalltalk**
David A. Thomas, Wilf R. LaLonde, and John R. Pugh, May 1986

SCS-TR-93 **Why Exemplars are Better Than Classes**
Wilf R. LaLonde, May 1986

SCS-TR-94 **An Exemplar Based Smalltalk**
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-95 **Recognition of Noisy Subsequences Using Constrained Edit Distances**
B. John Oommen, June 1986

SCS-TR-96 **Guessing Games and Distributed Computations in Synchronous Networks**
J. van Leeuwen, N. Santoro, J. Urrutia and S. Zaks, June 1986.

SCS-TR-97 **Bit vs. Time Tradeoffs for Distributed Elections in Synchronous Rings**
M. Overmars and N. Santoro, June 1986.

SCS-TR-98 **Reduction Techniques for Distributed Selection**
N. Santoro and E. Suen, June 1986.

SCS-TR-99	**A Note on Lower Bounds for Min-Max Heaps** A. Hasham and J.-R. Sack, June 1986.
SCS-TR-100	**Sums of Lexicographically Ordered Sets** M.D. Atkinson, A. Negro, and N. Santoro, May 1987.
SCS-TR-102	**Computing on a Systolic Screen: Hulls, Contours, and Applications** F. Dehne, J.-R. Sack and N. Santoro, October 1986.
SCS-TR-103	**Stochastic Automata Solutions to the Object Partitioning Problem** B.J. Oommen and D.C.Y. Ma, November 1986.
SCS-TR-104	**Parallel Computational Geometry and Clustering Methods** F. Dehne, December 1986.
SCS-TR-105	**On Adding *Constraint Accumulation* to Prolog** Wilf R. LaLonde, January 1987.
SCS-TR-107	**On the Problem of Multiple Mobile Robots Cluttering a Workspace** B. J. Oommen and I. Reichstein, January 1987.
SCS-TR-108	**Designing Families of Data Types Using Exemplars** Wilf R. LaLonde, February 1987.
SCS-TR-109	**From Rings to Complete Graphs - Θ(n logn) to Θ(n) Distributed Leader Election** Hagit Attiya, Nicola Santoro and Shmuel Zaks, March 1987.
SCS-TR-110	**A Transputer Based Adaptable Pipeline** Anirban Basu, March 1987.
SCS-TR-111	**Impact of Prediction Accuracy on the Performance of a Pipeline Computer** Anirban Basu, March 1987.
SCS-TR-112	**ε-Optimal Discretized Linear Reward-Penalty Learning Automata** B.J. Oommen and J.P.R. Christensen, May 1987.
SCS-TR-113	**Angle Orders, Regular n-gon Orders and the Crossing Number of a Partial Order** N. Santoro and J. Urrutia, June 1987.
SCS-TR-114	**An Optimal Algorithm for Detecting Weak Visibility of a Polygon** Jorg-R. Sack and Subhash Suri, December 1986.
SCS-TR-115	**Time is Not a Healer: Impossibility of Distributed Agreement in Synchronous Systems with Random Omissions** N. Santoro, June 1987.
SCS-TR-116	**A Practical Algorithm for Boolean Matrix Multiplication** M.D. Atkinson and N. Santoro, June 1987.
SCS-TR-117	**Recognizing Polygons, or How to Spy** James A. Dean, Andrzej Lingas and Jörg-R. Sack, August 1987.

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-118 **Stochastic Rendezvous Network Performance - Fast, First-Order Approximations**
J.E. Neilson, C.M. Woodside, J.W. Miernik, D.C. Petriu, August 1987.

SCS-TR-120 **Searching on Alphanumeric Keys Using Local Balanced Tree Hashing**
E.J. Otoo, August 1987.

SCS-TR-121 **An O(√n) Algorithm for the ECDF Searching Problem for Arbitrary Dimensions on a Mesh-of-Processors**
Frank Dehne and Ivan Stojmenovic, October 1987.

SCS-TR-122 **An Optimal Algorithm for Computing the Voronoi Diagram on a Cone**
Frank Dehne and Rolf Klein, November 1987.

SCS-TR-123 **Solving Visibility and Separability Problems on a Mesh-of-Processors**
Frank Dehne, November 1987.

SCS-TR-124 **Deterministic Optimal and Expedient Move-to-Rear List Organizing Strategies**
B.J. Oommen, E.R. Hansen and J.I. Munro, October 1987.

SCS-TR-125 **Trajectory Planning of Robot Manipulators in Noisy Workspaces Using Stochastic Automata**
B.J. Oommen, S. Sitharam Iyengar and Nicte Andrade, October 1987.

SCS-TR-126 **Adaptive Structuring of Binary Search Trees Using Conditional Rotations**
R.P. Cheetham, B.J. Oommen and D.T.H. Ng, October 1987.

SCS-TR-127 **On the Packet Complexity of Distributed Selection**
A. Negro, N. Santoro and J. Urrutia, November 1987.

SCS-TR-128 **Efficient Support for Object Mutation and Transparent Forwarding**
D.A. Thomas, W.R. LaLonde and J. Duimovich, November 1987.

SCS-TR-129 **Eva: An Event Driven Framework for Building User Interfaces in Smalltalk**
Jeff McAffer and Dave Thomas, November 1987.