AN OPTIMAL ALGORITHM FOR FINDING THE KERNEL OF A POLYGON

D. T. Lee and F. P. Preparata

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Joint Services Electronics Program

Approved for public release; distribution unlimited

Computational Complexity
Computational Geometry
Design of Algorithms
Optimal Algorithms

The kernel K(P) of a simple polygon P with n vertices is the locus of the points internal to P from which all vertices of P are visible. Equivalently, K(P) is the intersection of appropriate half-planes determined by the polygon's edges. Although the intersection of n generic half-planes is known to require time O(n log n), we show that one can exploit the ordering of the half-planes corresponding to the sequence of the polygon's edges to obtain a kernel finding algorithm which runs in time O(n) and is therefore optimal.
AN OPTIMAL ALGORITHM FOR FINDING
THE KERNAL OF A POLYGON

by

D. T. Lee and F. P. Preparata

This work was supported in part by the National Science Foundation
under Grant NSF MCS-76-17321 and in part by the Joint Services Electronics
Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract DAAB-07-72-
C-0592.

Reproduction in whole or in part is permitted for any purpose of
the United States Government.

Approved for public release. Distribution unlimited.
AN OPTIMAL ALGORITHM FOR FINDING THE KERNEL OF A POLYGON

D. T. Lee and F. P. Preparata
University of Illinois at Urbana-Champaign

Abstract

The kernel $K(P)$ of a simple polygon $P$ with $n$ vertices is the locus of the points internal to $P$ from which all vertices of $P$ are visible. Equivalently, $K(P)$ is the intersection of appropriate half-planes determined by the polygon's edges. Although the intersection of $n$ generic half-planes is known to require time $O(n \log n)$, we show that one can exploit the ordering of the half-planes corresponding to the sequence of the polygon's edges to obtain a kernel finding algorithm which runs in time $O(n)$ and is therefore optimal.

This work was supported by the National Science Foundation under Grant NSF MCS 76-17321 and by the Joint Services Electronics Program under Contract DAAB-07-72-C-0259.
1. The kernel $K(P)$ of a simple polygon $P$ is the locus of the points internal to $P$ which can be joined to every vertex of $P$ by a segment totally contained in $P$. Equivalently, if one considers the boundary of $P$ as a counterclockwise directed cycle, the kernel of $P$ is the intersection of all the half-planes lying to the left of the polygon's edges.

Shamos and Hoey [1] have presented an algorithm for finding the kernel of an $n$-edge polygon in time $O(n \log n)$. Their algorithm is based on the fact that the intersection of $n$ generic half-planes can be found in time $O(n \log n)$; they also show that $O(n \log n)$ is a lower-bound to the time for finding the intersection of $n$ half-planes. However, this lower-bound does not apply to the problem of finding the kernel, since in the latter case the half-planes are ordered according to the sequence of the edges of $P$, nor does their algorithm take advantage of this order. In this note we shall show that, indeed, this ordering can be exploited to yield an algorithm which runs in time linear in the number of the edges. Obviously, since each edge must be examined, the time of our algorithm is optimal within a multiplicative constant.

2. It is obvious that the kernel of the polygon $P$, being the intersection of half-planes, is a convex polygon $K(P)$. We shall denote $P$ by a doubly-linked list of vertices and intervening edges as $v_0 e_1 v_1 e_2 \cdots v_{n-1} e_{n-1} v_n$. 

* This work was supported by the National Science Foundation under Grant NSF MCS 76-17321 and by the Joint Services Electronics Program under Contract DAAB-07-72-C-0259.
We also impose a direction upon each edge such that the interior of the polygon lies to the left of the edge, or, equivalently, the boundary of P is directed counterclockwise. A vertex \( v_i \) is called **reflex** if the angle formed by its two adjacent edges \( e_{i-1} \) and \( e_i \) meeting at \( v_i \) is greater than \( \pi \), and it is called **convex** otherwise.

The algorithm we shall outline scans in order the vertices of P and constructs a sequence of polygonal chains \( K_0, K_1, \ldots, K_{n-1} \), called **kernel chains**. Each of these chains is a sequence of portions of straight lines, whose first and last members are half-lines and all others are line segments. As we shall show, the polygonal chain \( K_i \) bounds the intersection of the appropriate half-planes determined by \( e_0, e_1, \ldots, e_i \). Due to convexity, the angle between two consecutive edges of a kernel chain is always \( < \pi \). Notationally, if points \( w_i \) and \( w_{i+1} \) belong to the line containing the edge \( e_{s_i} \) of P, then \( w_i s_i w_{i+1} \) denotes the segment between \( w_i \) and \( w_{i+1} \) and directed like \( e_{s_i} \); moreover, \( \Lambda \) denotes a point at infinity and, for example, \( \Lambda w e \) denotes a half line terminating at vertex \( w \) and directed like edge \( e \).

If P has no reflex vertex, then P is convex and \( K(P) = P \). Thus let \( v_0 \) be a reflex vertex of P. Referring to figure 1, we set \( K_0 = \Lambda v_0 e_{n-1} \) equal to the intersection of the half-planes lying to the left of edges \( e_{n-1} \) and \( e_0 \). Notationally, \( K_0 \) will be represented by the string of symbols \( \Lambda e_0 v_0 e_{n-1} \). For each \( K_i \) it will be convenient to distinguish two vertices, \( F_i \) and \( L_i \), which delimit the sequence of vertices of \( K_i \) which are visible from \( v_i \); these two vertices play, as we shall see, a very important role in the construction of \( K_{i+1} \) from \( K_i \). Obviously, in \( K_0 \) we have \( F_0 = L_0 = v_0 \).
We now develop the advancing mechanism of the algorithm, i.e., the process of constructing \((K_{i+1}, F_{i+1}, L_{i+1})\) from \((K_i, F_i, L_i)\). For later ease of reference, it is convenient to distinguish a hierarchy of different cases.

(1) \(v_i\) is reflex (see figures 2a and b). In this case \(L_i\) lies on or to the left of the half line \(v_i e_{i-1} A\) and, obviously, \(L_{i+1} < L_i\). Candidates for \(F_{i+1}\) are only points belonging to the subchain delimited by \(F_i\) and \(L_i\). We now examine where the segment \(v_{i+1} F_i\) lies with respect to \(v_i v_{i+1}\).

(1.1) \(v_{i+1} F_i\) lies to the right of \(v_i v_{i+1}\) (figure 2a). We scan the
(2.1.1) \( v_{i+1} \in w'e_1 \Lambda \) (figure 3a). Clearly \( L_{i+1} \leftarrow v_{i+1} \) and \( F_{i+1} \leftarrow w' \); also, we obtain \( K_{i+1} = \alpha w'e_1 v_{i+1} e_1 \Lambda \).

Figure 3. Advancing mechanism when \( v_i \) is convex.
(2.1.2) $v_{i+1} \in v_i e_i w'$ (figure 3b). Let $\gamma_j$ denote the counterclockwise angle from the directed segment $w_j v_{i+1}$ to $e_i$. If $w_s \ldots w_{s+r}$ is the sequence of kernel vertices from $F_i$ to $L_i$, then we successively examine the angles $\gamma_s$, $\gamma_{s+1}$, $\ldots$, until we find a minimal $\gamma_{s+p}$. We then set $L_{i+1} \leftarrow w'$, $F_{i+1} \leftarrow w_{s+p}$, and $K_{i+1} \leftarrow \omega w' e_i A$.

(2.2) $L_i$ lies to the left of $v_i e_i A$ (figures 3c,d). Let $K_i = \omega L_i e_i A$. We determine the intersection $w'$ of $L_i A$ and $v_i e_i A$.

(2.2.1) $v_{i+1} \in w' e_i A$ (figure 3c). In this case, we set

$L_{i+1} \leftarrow v_{i+1}$, $F_{i+1} \leftarrow w'$ and $K_{i+1} \leftarrow \omega L_i e_i w' e_i v_{i+1} e_i A$.

(2.2.2) $v_{i+1} \in v_i e_i w'$ (figure 3d). In this case, $F_{i+1}$ is determined exactly as in the corresponding case described in (2.1.2) (figure 3b) whereas $L_{i+1} \leftarrow w'$ and $K_{i+1} \leftarrow \omega L_i e_i w' e_i A$.

In all of the above cases, it is immediate to realize that $K_{i+1}$ is the intersection of $K_i$ and of the half-plane to the left of $e_i$.

Using the advancing mechanism described above, we ultimately obtain the kernel chain $K_{n-1}$, which, if $K(P)$ is nonempty, is nonsimple (see figure 4), i.e., it has a crossing point $w$. Our remaining task is finding $w$. Let $K_{n-1} = A e' w e_1' \ldots w m e_i A$. We scan the edge sequence of $K_{n-1}$, starting from $e_2'$, and at the $i$-th step, for $i \geq 2$, we check whether $w_i$ lies to the left of the line containing $e_i'$, directed like $e_i'$. Let $s$ be the smallest value of $i$ for which $w_i$ lies to the right of $e_i'$. Next we scan the vertex sequence ($w_1 w_2 \ldots$) until we reach a vertex $w_r$, such that $w_{r-1}$ and $w_r$ lie on opposite side of $e_s'$. At this point we check whether $e_s'$ and $e_{r-1}'$ intersect: if they do, their intersection is the sought $w$; otherwise we replace $w_1$ with $w_r$ and continue the process (repeating the alternate scanning of the edge sequence and vertex
sequence) until the intersection is found.

3. We now analyze the performance of the algorithm outlined above.

In case (1.1) we scan $K_1$ starting from $F_1$, both counterclockwise and clockwise, and let $v_i$ be the total number of edges visited before finding the two intersections $w'$ and $w''$. This process actually removes $v_i - 2$ edges from $K_1$ (those comprised between $w_s$ and $w_{t-1}$ in figure 2a) and since each of the removed edges is colinear with a distinct edge of $P$, the total number of vertices visited by the algorithm in handling case (1.1) is at most $O(n)$.

In case (1.2), we scan $K_1$ counterclockwise starting from $F_1$, and clearly $(p+1)$ is the total number of vertices visited before we find $w_{s+p}$.

But in this process the distinguished point "F" has advanced $p$ positions.

Figure 4. Finding $K(P)$ from $K_{n-1}$. 
(from \( F_i = w_s \) to \( F_{i+1} = w_{s+t} \) ) counterclockwise. Since the number of vertices of any \( K_j \) is at most \( O(n) \), and the point "p" can only advance on kernel chains, we conclude that the total number of vertices visited by the algorithm in handling case (1.2) is at most \( O(n) \).

In case (2.1) the intersection \( w' \) of \( v_i e_i \wedge \) and \( w_t w_{t-1} \) involves scanning \( K_i \) clockwise from \( L_i \). Let \( \mu_i \) be the total number of edges visited before finding \( w' \). This process actually removes \( \mu_i \) edges from \( K_i \) (those comprised between \( w_t \) and \( \Lambda \) ). Here again, since each of the removed edges is colinear with a distinct edge of \( P \), the total number of vertices visited by the algorithm in finding \( w' \) in case (2.1) is at most \( O(n) \).

Case (2.1.1) requires a constant amount of work. Case (2.1.2) requires globally an amount of work at most \( O(n) \), by an argument identical to that developed for case (1.2).

The discussion of cases (2.2), (2.2.1), and (2.2.2) is exactly analogous to that of (2.1), (2.1.1), and (2.1.2), respectively.

Finally, it is straightforward to realize that finding the intersection \( w \) in \( K_{n-1} \) requires at most \( O(n) \) operations.

In summary, we conclude that finding the kernel of a simple polygon runs in time \( O(n) \), which is clearly optimal within a factor.

References
