COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 35, 99–110 (1986)

A Linear Time Algorithm for Minimum Link Paths inside a Simple Polygon

SUBHASH SURI

Department of Electrical Engineering and Computer Science, The Johns Hopkins University, Baltimore, Maryland 21218

Received August 12, 1985; revised February 25, 1986

1. INTRODUCTION

This paper addresses the following problem. Suppose a communication channel is to be set up between two points inside a simple polygon whose sides are opaque to the transmission. The transmission originates at a point s, called *source*, and is to be received at a point d, called *destination*. Clearly, a direct communication is impossible unless the points s and d are in sight of each other inside the polygon. Therefore, some repeaters (mirrors, for instance, if the transmission is optical in nature) are to be installed inside the polygon to make the communication feasible. What is the minimum number of repeaters required? With this motivation, a formal statement of the problem can be given as follows. Let P be a simple polygon. Let s and d be two designated points in P. Find a polygonal path between s and d that is internal to P and has minimum number of vertices possible. If the path is given by the set of vertices $\{s = x_0, x_1, \ldots, x_k = d\}$, then each segment $x_i x_{i+1}$, $0 \leq i \leq k - 1$, is called a *link*, and the entire path is said to consist of k links. The related problem of finding the Euclidean minimum length path between two points internal to a polygon was studied by Lee and Preparata [7]. The problem of minimum link path was originally posed by Toussaint. In this paper, we present an $O(n)$[1] time algorithm for computing a path with minimum number of links possible, where n is the number of vertices of the polygon P. Our algorithm makes use of the recently discovered linear time triangulation algorithm of Tarjan and Van Wyk [12], and linear time edge-visibility algorithm due to Guibas *et al.* [5].

In Section 2, we introduce our key lemmas that lead to an efficient algorithm for a minimum link path. An algorithm for finding a minimum link path is presented in Section 3. This algorithm has worst-case time complexity of $O(n^2)$, which is improved to $O(n)$ in Section 4. Finally, Section 5 concludes with few remarks and directions for further work on this problem.

2. PRELIMINARIES

We assume that the polygon P is given as a counterclockwise sequence of its vertices. The symbol P is used to denote the boundary of the polygon as well as the region of the plane enclosed by it. We begin by describing the notion of *visibility* polygons. A visibility polygon can be viewed as the region in P that is lit from a light source placed somewhere in P. The light source can be either a point or a line segment (usually referred to as an *edge*). We use the symbol V to denote the boundary of the visibility polygon, from either a point source or an edge source. For

[1] El Gindy [4] has independently obtained an algorithm for this problem that runs in $O(n \log n)$ time.

0734-189X/86 $3.00

Copyright © 1986 by Academic Press, Inc.
All rights of reproduction in any form reserved.

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

$$V(x) = \{ z \in P | xz \cap P = xz \},$$

where xz denotes the line segment connecting x and z. For a line segment e inside P, the visibility polygon from e, denoted by $V(e)$, is the set of points in P visible from e, i.e.,

$$V(e) = \{ z \in P | \exists y \in e \text{ s.t. } zy \cap P = zy \}.$$

A *chord* is a line segment that lies entirely interior to P and has its endpoints on the boundary. A chord ab divides P into two subpolygons, P_1 and P_2, such that P_1 and P_2 have their common intersection along ab only. More precisely, let a polygon be represented as a counterclockwise sequence of its vertices. Let $P = (v_1, v_2, \dots, v_n)$. Let a and b lie on the edges $v_i v_{i+1}$ and $v_j v_{j+1}$, $i < j$, respectively. The two subpolygons resulting from cutting P along ab are given as $P_1 = (b, v_{j+1}, v_{j+2}, \dots, v_i, a)$ and $P_2 = (a, v_{i+1}, v_{i+2}, \dots, v_j, b)$. Two line segments l_1 and l_2 are said to overlap if and only if their common intersection is a non-degenerate line segment, i.e., is not a point. The following lemma characterizes the edges that make up $V(e)$. The proof is rather elementary and is omitted here.

LEMMA 1. *Let e be a line segment in P such that vertices of P together with the endpoints of e are in general position. Each edge of $V(e)$ either*

(a) *overlaps with an edge of P, or*

(b) *is a chord with a reflex vertex of P as one endpoint.*

Let T be a *triangulation* of P, where a triangulation of a simple polygon is defined as follows, in the manner of Mehlhorn [8]:

> A triangulation of a vertex set $\{v_1, v_2, \dots, v_n\}$ is a maximal set of nonintersecting straight line segments between points in this set. A triangulation of polygon P is a triangulation of its vertex set such that all the triangulating edges lie in P.

Let G be the dual graph of this triangulation. G is obtained by assigning a vertex to each face of T, and joining two vertices t_i and t_j if and only if the triangles t_i and t_j in the triangulated polygon share a common side. Thus, a triangle t_i of T has its corresponding node in the dual graph G labeled t_i. The graph G obtained in this way is a tree, whose nodes have a maximum degree of three. Let t_s and t_d be the triangles in T containing s and d, respectively. Let $X_{s,d}$ be the path in G from node t_s to node t_d. Since G is a tree, this path is unique. Let the nodes in the path $X_{s,d}$ be indexed in increasing order according to their appearance along the path. Then the path $X_{s,d}$ can be given as an ordered list of nodes $\{t_s = t_1, t_2, \dots, t_{k-1}, t_k = t_d\}$. The triangles t_1 through t_k indexed in this manner define a "forward" direction of visibility in P. Our computation of the visibility polygons will progress in this direction. Let $T_{1,k}$ denote the ordered list of triangles $\{t_1, t_2, \dots, t_k\}$. We assume throughout that G is rooted at $t_1 = t_s$.

Our interest in $T_{1,k}$ is motivated due to the following reasons. We show that any minimum link path in P from s to d intersects every triangle of $T_{1,k}$ in the order of increasing indices. For each triangle $t_u \in T_{1,k}$ we compute the minimum number of links in a path from s to a point in t_u. Since $d \in t_k$, these computations will ultimately lead to the minimum number of links required between s and d. By storing certain boundaries of the visibility polygons computed in the procedure, we will also reconstruct the path from s to d with a minimum number of links.

First we show that certain portions of P that are inessential for a minimum link path between s and d can be removed to simplify our discussion, and possibly gain some efficiency in practice. Let x_1, y_1, and z_1 (resp. x_k, y_k, and z_k) be the vertices of t_1 (resp. t_k) such that y_1z_1 (y_kz_k) is shared between t_1 and t_2 (resp. t_{k-1} and t_k). Let $P(x_1y_1)$ and $P(x_1z_1)$ be the polygons not containing t_1 that result from cutting P along x_1y_1 and x_1z_1, respectively. Similarly for polygons $P(x_ky_k)$ and $P(x_kz_k)$. The following lemma is quite straightforward.

LEMMA 2. *The polygon P with $P(x_1y_1)$, $P(x_1z_1)$, $P(x_ky_k)$, and $P(x_kz_k)$ removed, is sufficient for computing a minimum link path between s and d.*

The proof of Lemma 2 is based on the fact that P is a simple polygon, and none of the removed sections contains either s or d. We omit this simple proof.

For the remainder of our discussion, we assume that our polygon P has been trimmed in the manner of Lemma 2. The following corollary follows immediately from Lemma 2.

COROLLARY 1. *In the dual graph G, t_k is a leaf.*

A chord ab is said to intersect a triangle $t_i \in T_{1,k}$ if and only if $(a, b) \cap t_i \neq \varnothing$, where (a, b) is the open line segment from a to b and t_i denotes the boundary and the interior of the triangle t_i. For the remaining discussion, we will focus our attention on only those chords that intersect at least one triangle of $T_{1,k}$. Let e be a chord of P. Define $t_{e\max} \in T_{1,k}$ to be the highest indexed triangle intersected by e. Let P_1 and P_2 be two subpolygons that result from cutting P along e. Assume without loss of generality that P_2 contains all the triangles $t_{e\max+1}, t_{e\max+2}, \ldots, t_k$. The visibility polygon $V(e)$ is computed inside P_2 only.

Let e be a chord of P, and $V(e)$ the visibility polygon of e. Let $t_i \in T_{1,k}$, $1 \leq i \leq k$, be the highest indexed triangle whose interior is (at least partially) visible from e. Let p, q, and r be the three vertices of t_i. Assume that qr is shared between t_i and t_{i+1}. An edge ab of $V(e)$ is defined to be the *window* of e, $w(e)$, if

(a) ab is a chord of P,

(b) ab intersects t_i, and

(c) if ab intersects the boundary of t_i in a_i and b_i, then the interior of the quadrilateral a_ib_irq is not visible from e.

Intuitively the window, $w(e)$, represents the farthest along $T_{1,k}$ that is visible from e in P. Starting from the source s, our algorithm will compute the visibility polygons only for the successive windows. The following lemma establishes that the window is well defined for any chord of P.

LEMMA 3. *Let e be a chord of P. Either there exists an unique $w(e)$, or the destination lies in $V(e)$.*

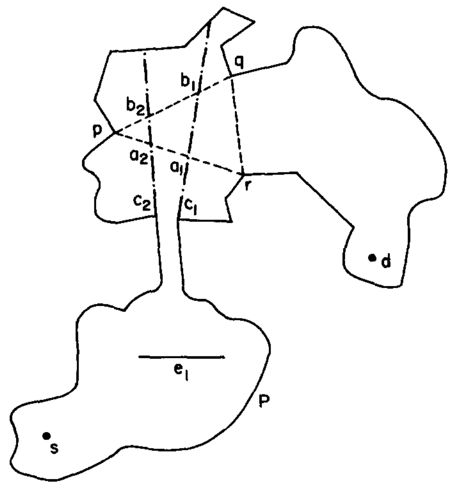

FIG. 1. Proof of Lemma 2.

Proof. If $d \in V(e)$, the claim follows immediately. Otherwise, we show the unique existence of a chord that meets the requirements of $w(e)$. The proof of existence shows the existence of a chord that meets the conditions (a) and (b). Condition (c) is taken care of by the uniqueness proof.

Existence of w(e). Let $t_{e\max} \in T_{1,k}$ be the highest indexed triangle intersected by e. Clearly, e can see the entire triangle $t_{e\max}$. Since $d \notin V(e)$, e cannot see the entire triangle t_k. Let $t_i \in T_{1,k}$, $e\max < i \leq k$, be the highest indexed triangle of $T_{1,k}$ whose interior is at least partially visible from e. Since e can see the interior of t_i but not of t_{i+1}, $V(e)$ must intersect t_i. This intersection can take place only along a chord. Moreover, this chord meets conditions (a) and (b) of $w(e)$. If $i = k$, then it follows from Corollary 1 that t_k is a leaf of G. Clearly two sides of t_k are edges of P and the intersection of $V(e)$ with t_k must be along a chord, which meets the conditions (a) and (b) of $W(e)$. The proof of existence of $w(e)$ is complete.

Uniqueness of w(e) (see Fig. 1). Let the three vertices of t_i be p, q, and r. We will present the argument for the case $i < k$ only. The other case is very similar and will not be discussed here. Assume without loss of generality that qr is the common boundary of t_i and t_{i+1}. Since the interior of t_{i+1} is not visible from e, no chord of $V(e)$ intersects qr. Let c_1 be a chord of $V(e)$ that intersects pr and pq in a_1 and b_1, respectively. Similarly, let a_2 and b_2 correspond to chord c_2. Since $V(e)$ is a simple polygon, c_1 and c_2 do not intersect. Note that there can be at most two such chords. If a_1b_1 lies in the quadrilateral a_2b_2rq, let c_1 be the window $w(e)$, otherwise let c_2 be the window. Since qr is not visible from e, interior of quadrilateral a_1b_1rq in the former and quadrilateral a_2b_2rq in the latter case is not visible from e. Otherwise, there must be another chord of $V(e)$ that intersects the interior of the said quadrilateral, but that is impossible. The proof of uniqueness is complete. □

Now we want to establish that there are no gaps in the visibility polygon $V(e)$ with respect to the set $T_{1,k}$, i.e., $V(e)$ intersects all the triangles of $T_{1,k}$ that lie between e and $w(e)$.

LEMMA 4. *Let e be a chord and $w(e)$ its window. Let $t_{e\max}$ be the highest indexed triangle of $T_{1,k}$ intersected by e. Let $t_{w\max}$ be the highest indexed triangle of $T_{1,k}$ intersected by $w(e)$. Let $y \in t_{w\max} \cap w(e)$. Let $x \in e$ be a point that is visible from y:*

(a) *The line segment xy intersects all triangles $t_i \in T_{1,k}$, $e\max \le i \le w\max$.*

(b) *Any other triangle of T that is intersected by xy is also intersected by e.*

Proof. (a) Let p, q, and r be the vertices of $t_{e\max}$. Assume that qr is shared between $t_{e\max}$ and $t_{e\max+1}$. Clearly, $qr \cap e = \varnothing$. Let P_1 be the subpolygon that results from cutting P along qr and contains e. Let P_2 be the other subpolygon. If $y \in t_{e\max}$, the claim follows vacuously. Otherwise, since $y \in P_2$ and $x \in P_1$, the segment xy must intersect qr and $t_{e\max}$. Let xy intersect qr in x'. Since $x' \in t_{e\max}$ and $y \in t_{w\max}$, and $e\max < w\max$, xy must cross a sequence of triangles, namely, $t_{e\max} = t_{i_1}, t_{i_2}, \ldots, t_{i_j} = t_{w\max}$ such that two adjacent triangles share a common side. This sequence, therefore, represents a path from $t_{e\max}$ to $t_{w\max}$ in G. But, since G is a tree, this path is unique. The sequence t_{i_1} through t_{i_j} must, therefore, coincide with the sequence of triangles in $T_{1,k}$ indexed between $e\max$ and $w\max$. It follows that xy intersects all the triangles of $T_{1,k}$ indexed between $e\max$ and $w\max$. The claim is thus proved.

(b) It is sufficient to show that any triangle of T that is intersected by xx' is also intersected by e. If x lies in any triangle of $T_{1,k}$, i.e., $x \in t_i$, $t_i \in T_{1,k}$ and $e\min \le i \le e\max$, the argument presented in (a) can be repeated to show that xx' intersects only the triangles of $T_{1,k}$ indexed from i through $e\max$. Otherwise proceed as follows. Let p, q, and r be the vertices of $t_{e\max}$, where qr is shared between $t_{e\max}$ and $t_{e\max+1}$. Let e intersect pq and pr at a_1 and b_1, respectively (see Fig. 2). Let xx' intersect pr at b_2. Since the triangle $\Delta a_1 b_1 b_2$ is internal to $V(e)$, it cannot have holes. Hence, there are no vertices of P in the interior of triangle $\Delta a_1 b_1 b_2$. xx', therefore, intersects exactly those triangles that are intersected by xa_1. Since xa_1 is a segment of e, the claim follows. □

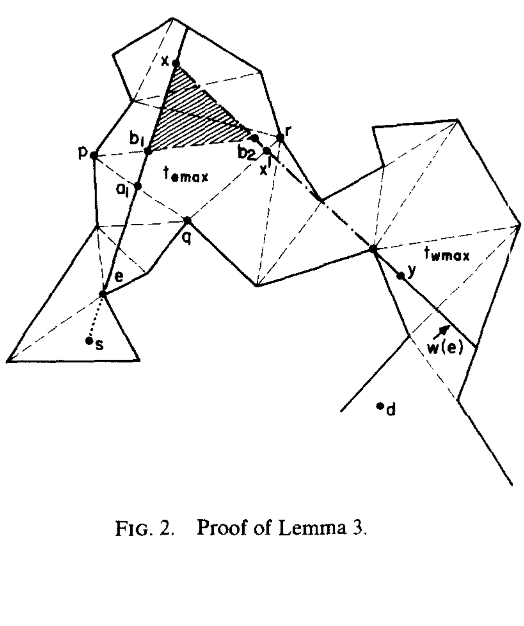

FIG. 2. Proof of Lemma 3.

COROLLARY 2. *$V(e)$ intersects every triangle $t_i \in T_{1,k}$, for* e min $\leq i \leq w$ max.

The following lemma establishes a property of $w(e)$ that is crucial for an efficient computation of a minimum link path.

LEMMA 5. *Let $e_1, e_2, \ldots, e_i$ be a sequence of chords such that*

(a) *e_1 intersects $t_1 = t_s$,*

(b) *$e_j = w(e_{j-1})$, $2 \leq j \leq i$, and*

(c) *$d \notin \bigcup_{j=1}^{i-1} V(e_j)$.*

Let x be a point in $\bigcup_{j=1}^{i-1} V(e_j)$. Then any path in P from x to d intersects e_i.

Proof. Consider the two subpolygons P_1 and P_2 that result from cutting P along the chord e_i. Assume without loss of generality that $x \in P_1$. We first establish that $d \in P_2$. All triangles $t_u \in T_{1,k}$, $1 \leq u \leq e_{i\,\min} - 1$, are completely contained in P_1, where $e_{i\,\min}$ is the lowest indexed triangle of $T_{1,k}$ intersected by e_i. In addition, P_1 also partially contains all the triangles $t_u \in T_{1,k}$, $e_{i\,\min} \leq u \leq e_{i\,\max}$, where $e_{i\,\max}$ is the highest indexed triangle of $T_{1,k}$ intersected by e_i. P_1 intersects no other triangle of $T_{1,k}$ besides these. Hence, if $d \in P_1$ then $k \leq e_{i\,\max}$.

Since $e_j = w(e_{j-1})$, for $2 \leq j \leq i$, Lemma 4 implies that $\bigcup_{j=1}^{i-1} V(e_j)$ intersects all triangles $t_u \in T_{1,k}$, $1 \leq u \leq e_{i\,\max}$. Let e_u, $1 \leq u \leq i-1$, be the first chord such that $V(e_u)$ intersects t_k. Since $d \notin V(e_u)$, d must lie in that part of t_k that is hidden from e_u by a chord. Since k is the highest indexed triangle of $T_{1,k}$, this chord must be $w(e_u)$. If $u < (i-1)$, it is easily seen that $d \in V(e_{u+1})$, thus contradicting the hypothesis on d. If $u = (i-1)$, d clearly must lie on that side of $w(e_u) = e_i$ that goes into P_2 after splitting P along e_i. Therefore, $d \in P_2$.

Clearly, the common intersection of P_1 and P_2 lies only on e_i. Since $x \in P_1$ and $d \in P_2$, any path from x to d must intersect e_i. This establishes the claim of the lemma. □

Note that both s and d can be regarded as degenerate cases of a chord.

3. ALGORITHM FOR MINIMUM LINK PATH

Our algorithm computes the visibility polygons for successive windows, starting from s. Let $s = e_1$ be the first chord. Compute $V(e_1)$. If $d \in V(e_1)$, we have a one-link path, i.e., s and d can be joined by a straight line inside P, and we can stop. Otherwise, by Lemma 3, $w(e_1)$ exists. Also, Lemma 5 says that a path from s to d intersects $w(e_1)$. Let $e_2 = w(e_1)$, and compute $V(e_2)$, and so on until we find a window $e_i = w(e_{i-1})$ such that $d \in V(e_i)$. The minimum link path from s to d can then be found by constructing a sequence of point $d_{i+1}, d_i, d_{i-1}, \ldots, d_1$ where $d_{i+1} = d$ and d_j, $1 \leq j \leq i$ is a point on e_j that can see d_{j+1}. Since, $d_1 = s$, the path has i links. Thus, the algorithm for finding a minimum link path can be given as follows:

ALGORITHM A. An algorithm for finding a minimum link path.

1. $w(e_0) \leftarrow s$; {*Initialization*}
2. $i \leftarrow 0$;
3. **repeat**
4. $\quad i \leftarrow i + 1$;

5. $e_i \leftarrow w(e_{i-1})$;
6. Compute $V(e_i)$;
7. **until** $d \in V(e_i)$;
8. $d_{i+1} \leftarrow d$;
9. **for** $j = i$ **down to** 1 **do**
10. $d_j \leftarrow$ a point on e_j that can see d_{j+1};
11. **output** *minimum link path* $= \{s = d_1, d_2, \dots, d_i, d_{i+1} = d\}$;

end {Algorithm A}

Correctness and Time Complexity of Algorithm A. The correctness of Algorithm A can be easily established as follows. Let $e_1, e_2, \dots, e_k$ be the set of chords output by Algorithm A such that

(1) $e_1 = s$,

(2) $e_j = w(e_{j-1}), 1 < j \leq k$,

(3) $d \in V(e_k)$ but $d \notin V(e_j)$ for $j < k$.

Then we claim that a minimum link path (MLP) between s and d requires k links. Assume for a contradiction that MLP: $(s = v_1, v_2, \dots, v_l = d)$ is a minimum link path such that $l < k$. Clearly, $v_2 \in V(e_1)$, since $v_1 = s = e_1$. Lemma 5 dictates that MLP intersects e_2, and it follows that $v_3 \in V(e_2)$. The vertices of MLP and windows of Algorithm A can thus be matched one to one, which contradicts the assumption that $l < k$. It is therefore established that Algorithm A does indeed output a minimum link path from s to d.

The time complexity analysis can be done in the following manner. Since a minimum link path from s to d can have $O(n)$ links, the **repeat** loop can be executed $O(n)$ times. Computation of $V(e_i)$ is in fact computing visibility polygon from an edge, and can be done in $O(n)$ time using very recent results due to Guibas *et al.* [5]. The property $d \in V(e_i)$ can be checked in time proportional to the size of $V(e_i)$, which is $O(n)$. That the actual construction of the path (lines 8–11) takes only $O(n)$ time can be established as follows. We preprocess each visibility polygon $V(e_j)$ using the methods of Guibas *et al.* [5] so that given a point x on the boundary of $V(e_j)$ a point $y \in e_j$ such that xy lies in $V(e_j)$ can be located in $O(\log n_j)$, where n_j is the size of $V(e_j)$ (see [5] for details). Hence, if the path has k links, then the total cost of path reconstruction is $O(\sum_{i=1}^{k}(\log n_i))$, where $\sum_{i=1}^{k} n_i = n$. Clearly, $O(\sum_{i=1}^{k}(\log n_i)) = O(n)$. It follows, therefore, that the time complexity of Algorithm A is $O(n^2)$. It is noteworthy that if we spend $O(n)$ time for computation of $V(e_i)$ each time, by considering the entire polygon P, Algorithm A can actually take $\Omega(n^2)$ time: Fig. 3 shows a polygon that forces the minimum link path to have $n/2$ links.

In the following section we show how the time complexity of Algorithm A can be improved to $O(n)$. This improvement in the running time is achieved by computing $V(e_i)$'s more efficiently. Instead of spending $O(n)$ time for computing each $V(e_i)$ we spend only $O(n_i)$ time, where n_i is the total number of triangles of T that are intersected by $V(e_i)$. Since $\sum_i n_i = O(n)$, even though the **repeat** loop of Algorithm A can be executed $\Omega(n)$ times, the computation of all $V(e_i)$'s together will require only $O(\sum_i n_i) = O(n)$ total time. The technique for computing $V(e_i)$'s in $O(n_i)$ time is detailed in the following section.

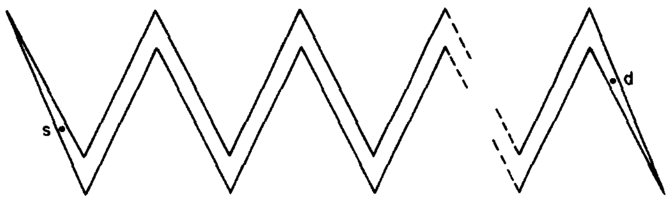

FIG. 3. A polygon that forces a minimum path of $n/2$ links between s and d.

4. EFFICIENT COMPUTATION OF $V(e_i)$'s

In this section, we substantiate our claim that the visibility polygon $V(e_i)$ can be computed in $O(n_i)$, where n_i is the number of triangles of T that are intersected by $V(e_i)$. The basic idea is as follows. Let e_i be a chord of P for which we want to compute $V(e_i)$. Let $W(e_i)$ be the window of e_i. Let $t_{e_{i\,\min}}$ be the lowest indexed triangle intersected by e_i. Let $t_{w_{i\,\max}}$ be the highest indexed triangle intersected by $w(e_i)$. Lemma 4 implies that $V(e_i)$ intersects all triangles $t_u \in T_{1,k}$, $e_{i\,\min} \le w_{i\,\max}$. To be more precise, let $y \in w(e_i) \cap t_{w_{i\,\max}}$ be a point. Let $x \in e_i$ be a point that is visible from y. Lemma 4 proves that the segment xy intersects only those triangles of T that are either intersected by e_i or are triangles of $T_{1,k}$ indexed between $e_{i\,\max}$ and $w_{i\,\max}$. Therefore, we can perform a binary search exclusively on the triangles of $T_{1,k}$ in the "forward" direction to locate $t_{w_{i\,\max}}$. The binary search starts by computing the visibility region from e_i inside a small set of triangles and doubles the number of triangles under consideration at each step. These notions are made more precise in the following.

Consider the dual graph of the triangulation, G. Let x and y be two nodes in G. There is a unique path in G from x to y. The union of triangles x and y along with all the triangles that lie along the path from x to y describe a polygonal region in P. We use this duality between length of a path in G and size of the corresponding polygonal region in P to successively double the size of the polygonal region considered for computing the visibility polygon from e_i. By size of a polygonal region P_i we mean the number of vertices of P_i.

Let e_i be a chord of P for which we need to compute the visibility polygon $V(e_i)$. Let $e_{i\,\max} - e_{i\,\min} + 1 = m_1$. Let $m = m_1 + m_2$ be the total number of triangles of T that are intersected by e_i: First m_1 triangles are in $T_{1,k}$ indexed between $e_{i\,\min}$ and $e_{i\,\max}$, and remaining m_2, namely, $\{t_1', t_2', \dots, t_{m_2}'\}$ are in $T - T_{1,k}$. Let P_2 be the subpolygon that results from cutting P along e_i and contains all the triangles $t_{e_{i\,\max}+1}, \dots, t_k$.

We start by computing the visibility region from e_i inside $P_2 \cap \{t_{e_{i\,\min}} \cup t_{e_{i\,\min}+1} \cup \cdots \cup t_{e_{i\,\max}} \cup t_1' \cup \cdots \cup t_{m_2}'\}$. This region is trivially entirely visible from e_i. At next step we double the size of the polygonal region under consideration by including m more triangles of $T_{1,k}$ starting from $t_{e_{i\,\max}+1}$. The visibility computation is done all over again at each new step. In general, let P^j be the polygonal region under consideration at the jth step. Let $|P^j|$ denote the number of triangles in P^j. Let $t_{j_{\max}}$ be the highest indexed triangle of $T_{1,k}$ that is included in P^j. The polygonal region considered for the $(j+1)$th step is $P^j \cup \{t_{j_{\max}+1}, \dots, t_{j_{\max}+|P^j|}\}$. We will call this procedure of successively increasing the size of a polygonal region by a factor of 2, "doubling." The doubling is discontinued if *no* point of the common diagonal of the $j_{\max}$-th and $(j_{\max}+1)$-th triangles is visible from e_i, in which case $w_{i\max}$ must lie between $e_{i\min}$ and $j_{\max}$. In addition, $w_{i\max} > l_{\max}$, for any $l < j$; otherwise the doubling must have been stopped at the lth step. Hence we conclude that if doubling stops at jth step, then $e_{i\min} \le w_{i\max} \le j_{\max}$, and $V(e)$ intersects at least half the triangles of P^j.

Now we are ready to compute the actual visibility polygon $V(e_i)$. The reason for this final computation is the following. The visibility region computed at the jth step of doubling may be a proper subset of $V(e_i)$. This is due to the fact that all the triangles of the set $T - T_{1,k}$ have been completely ignored. Therefore, the part of $V(e_i)$ that lies in the triangles of $T - T_{1,k}$ has not yet been computed. However, $t_{w_{i\max}}$ is correctly determined. In order to compute $V(e_i)$, therefore, the final computation considers the following triangles

(a) all triangles of P^j, and

(b) all triangles t_u such that in G the least common ancestor of t_u with $t_{j_{\max}}$ is indexed between $e_{i\min}$ and $j_{\max}$.

Since G is rooted at t_1, the least common ancestor is well defined. The refined algorithm for computing $V(e_i)$ can now be given as

ALGORITHM B. An algorithm for computing $V(e_i)$ efficiently.

1. $P^1 \leftarrow$ union of all the triangles that are intersected by e_i;
2. $V^1 \leftarrow$ region of P^1 visible from e_i;
3. $j \leftarrow 1$;
4. **repeat**
5. $\quad j \leftarrow j+1$;
6. $\quad P^j \leftarrow P^{j-1} \cup \{|P^{j-1}|$new triangles of $T_{1,k}\}$
7. $\quad V^j \leftarrow$ region of P^j visible from e_i;
8. $\quad t_{j_{\max}} \leftarrow$ highest indexed triangle in P^j;
9. **until** $V^j \cap \{$common diagonal of $t_{j_{\max}}$ and $t_{j_{\max}+1}\} = \varnothing$;
   {Final computation of $V(e_i)$}
10. $P^{\text{final}} \leftarrow P^j \cup \{$all triangles $t_u \in T - T_{1,k}$ whose least common ancestor with $t_{j_{\max}}$ in G is indexed between $e_{i\min}$ and $j_{\max}\}$
11. $V(e_i) \leftarrow$ region of P^{final} visible from e_i;

end {Algorithm B}

Proof of Correctness of Algorithm B. We show that Algorithm B correctly computes $V(e_i)$. First we show that $t_{w_{i\max}}$ is correctly determined. Let $t_{w_{i\max}}$ be the highest indexed triangle intersected by $w(e_i)$. Lemma 4 proves that for any point $y \in t_{w_{i\max}} \cap w(e_i)$ and a point $x \in e_i$ that is visible from y, the line segment xy intersects only those triangles of T that are either intersected by e_i, or are triangles of $T_{1,k}$ indexed between $e_{i\max}$ and $w_{i\max}$. Since P^j considers all triangles of $T_{1,k}$ indexed between $e_{i\min}$ and $j_{\max}$, where $j_{\max} > w_{i\max}$, along with t'_1 through t'_{m_2}, which are the triangles intersected by e_i of the set $T - T_{1,k}$, $w_{i\max}$ is correctly determined. Next, if $V(e_i)$ intersects any triangle of $t_u \in T - T_{1,k}$, then the least common ancestor of t_u and $t_{j_{\max}}$ in G must be indexed between $e_{i\min}$ and $j_{\max}$ (see Fig. 4). This can be established in the following manner. We can assume without loss of generality that $j_{\max} \le k$. Let $t_u \in T - T_{1,k}$ be a triangle that is intersected by $V(e_i)$. Let the lowest common ancestor of t_u and t_k in G be t_v, $1 \le v \le k$. We

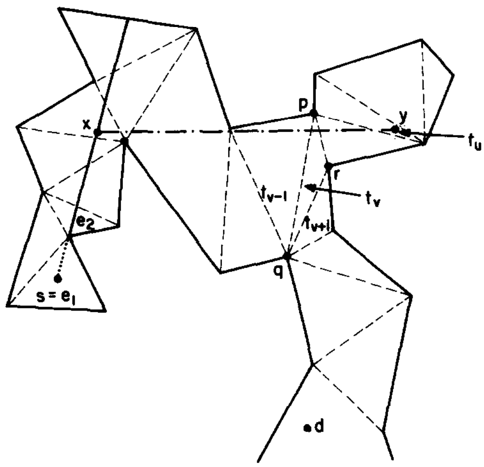

FIG. 4. Proof of correctness of Algorithm-B. The point $y \in t_u$, where $t_u \in T - T_{1,k}$. The segment xy intersects the lowest common ancestor of t_u and $t_{j_{\max}}$, which is t_v.

will present the argument assuming that $v < k$; the other case is very similar and will not be discussed. Let p, q, and r be three vertices of t_v, where qr is shared between t_v and t_{v+1} and pq is shared between t_v and t_{v-1}. Let P_1 be the subpolygon that results from cutting P along pr and contains t_u. Let P_2 be the other subpolygon. It should be clear that e_i lies in P_2. Let $y \in t_u$ be a point, and $x \in e_i$ be a point that is visible from y. The line segment xy must intersect pr and, therefore, triangle t_v. Hence, $v \leq w_{i\,\max}$. Since $w_{i\,\max} < j_{\max}$, it follows that $v < j_{\max}$. The claim is therefore established. Since all such triangles are included in the final computation, $V(e_i)$ is correctly computed.

Time Complexity of Algorithm B. Now we must establish that Algorithm B computes $V(e_i)$ in time $O(n_i)$, where n_i is size of $V(e_i)$. First, it should be clear that if number of triangles of T intersected by $V(e_i)$ equal n_i, then at no step more than $2n_i$ triangles are considered. In additional, at each stage of the computation the number of triangles considered by the algorithm doubles from the previous value. The total cost of computing $V(e_i)$, for some constant c, is

$$2cn_i + cn_i + \frac{cn_i}{2} + \cdots + c$$

which is $O(n_i)$.

Again, a triangle of T can lie in at most two visibility polygons, namely, $V(e_i)$ and $V(w(e_i))$. Hence, the total cost of computing all $V(e_i)$'s in Algorithm A is $O(\Sigma_i n_i) = O(n)$. Using very recent results of Tarjan and Van Wyk [12], a triangulation of P can be computed in $O(n)$ time. The dual graph G can be obtained in $O(n)$ additional time. Since we proved in Section 3 that given all $V(e_i)$'s the minimum link path can be reconstructed in $O(n)$ time, the following theorem is established.

THEOREM 1. *Given two designated points s and d inside a simple polygon P, a path of minimum links between s and d can be computed in $O(n)$ time, where n is the number of vertices of P.*

5. CONCLUSIONS

We have considered the problem of finding a minimum link path between two designated points inside a simple polygon. We first describe an algorithm that runs in $O(n^2)$ time. Later, we improve the running time of this algorithm to $O(n)$. Using a similar approach we can also preprocess the polygon P in $O(n)$ time and space to obtain a data structure for minimum link path queries. This data structure can

(1) output the minimum number of links needed between a fixed source s and an arbitrary destination d in $O(\log n)$ time,

(2) construct the actual path from s to d in time $O(\log n + k)$, where k is the number of links in the path.

The details of this preprocessing will be presented in [10].

The general problem of finding minimum link paths between two points in the presence of polygonal obstacles seems more difficult. Recently Suri and O'Rourke [9] showed that the boundary of an edge-visibility polygon in the presence of polygonal holes may have $\Omega(n^4)$ vertices in the worst case. Similar worst-case lower bounds may apply to the minimum link path problem when the holes are present.

Recently Toussaint [T] suggested a new measure of the shape complexity of a simple polygon. Let $d(x, y)$, the distance between x and y, be defined as the minimum number of links in a polygonal path between x and y. This metric, called the *minimum link metric*, suggests a new measure for the *internal* diameter of a polygon. Define the *link-diameter* of a polygon P to be the largest distance between two points of P in the minimum link metric. Our algorithm for minimum link path between two points can be used to given an $O(n^2)$ algorithm for this problem. However, Suri [10] has recently discovered a more efficient algorithm that has running time of $O(n \log^2 n)$. He also presents a simpler approximation algorithm with running time $O(n \log n)$, whose output is guaranteed to be within two of the link-diameter.

ACKNOWLEDGMENTS

I wish to thank Professor Joseph O'Rourke with whom I had many fruitful discussions on this problem and whose guidance provided constant encouragement. This work was supported by NSF Grant DCR83-51468.

REFERENCES

1. B. Chazelle and L. J. Guibas, Visibility and intersection problems in plane geometry, in *Symposium on Computational Geometry*, 1985, pp. 135–146.
2. H. El Gindy and D. Avis, A linear algorithm for computing the visibility polygon from a point, *J. Algorithms*, **2**, 1981, 186–197.
3. H. El Gindy, An Efficient Algorithm for Computing the Weak Visibility Polygon from an Edge in Simple Polygons, Technical Report, School of Computer Science, McGill University, Jan. 1984.
4. H. El Gindy, Hierarchical Decomposition of Polygons with Applications, Ph.D. Thesis, School of Computer Science, McGill University, May 1985.
5. L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan, Linear time algorithms for visibility and shortest path problems inside a simple polygon, *Second Symposium on Computational Geometry*, 1986.
6. D. T. Lee and A. Lin, Computing Visibility Polygon from an Edge, technical report, Department of Electrical Engineering and Computer Science, Northwestern University, Jan. 1984.
7. D. T. Lee and F. Preparata, Euclidean shortest paths in the presence of rectilinear barriers, *Networks* **14**, 1984, 393–410.
8. K. Mehlhorn, "*Multi-dimensional Searching and Computational Geometry*," EATCS Monographs on

Theoretical Computer Science Vol. 3, Springer-Verlag, New York/Berlin, 1984.
9. S. Suri and J. O'Rourke, Worst-Case optimal algorithms for constructing visibility polygons with holes, *2nd Symposium on Computational Geometry*, IBM, Yorktown Heights, 1986.
10. S. Suri, Computing the Link Diameter of a Simple Polygon, technical report, The Johns Hopkins University, April, 1986.
11. G. Toussaint, private communication, January 1986.
12. R. E. Tarjan & C. Van Wyk. A linear time algorithm for triangulating simple polygons, manuscript, Oct. 1985.