Computing Geodesic Furthest Neighbors in Simple Polygons*

SUBHASH SURI

Bell Communications Research,
435 South Street, Morristown, New Jersey 07960

Received August 1, 1987; revised March 1, 1988

An algorithm is presented for computing geodesic furthest neighbors for all the vertices of a simple polygon, where geodesic denotes the fact that distance between two points of the polygon is defined as the length of an Euclidean shortest path connecting them within the polygon. The algorithm runs in $O(n \log n)$ time and uses $O(n)$ space; $n$ being the number of vertices of the polygon. As a corollary, the geodesic diameter of the polygon also can be computed within the same time and space bounds. © 1989 Academic Press, Inc.

1. INTRODUCTION

The furthest neighbors problem has been studied for quite some time in computational geometry. In the simplest version of this problem, we are given a planar set $P$ of $n$ points and are asked to find, for each $p_i \in P$, a point $p_k \in P$ such that

$$\text{distance}(p_i, p_k) = \max \{ \text{distance}(p_i, p_j) | 1 \leq j \leq n \},$$

where distance(·) is a metric defined over the point set $P$. Several $O(n \log n)$ time algorithms are known to solve this problem for the $L_2$ metric (e.g., see Preparata and Shamos [9]). Furthermore, given an arbitrary set of points in the plane, $\Omega(n \log n)$ also is a lower bound in the worst case. This lower bound, however, does not apply if additional structure is imposed on the points. For instance, if the points are the vertices of a convex polygon (given in order along the boundary), then furthest neighbors of all the points can be computed in linear time (see Aggarwal et al. [1]). This paper generalizes the all-furthest neighbors problem in a natural way to simple polygons: The given points are the vertices of a simple polygon and the distance between two points is measured by the length of a shortest path connecting the two points inside the polygon.

Let $P$ be a simple polygon having $n$ vertices. Throughout, $P$ will also denote the closed connected region of the plane that is bounded by the polygon $P$. It is well known that, for two points $x$ and $y$ in $P$, there is an unique Euclidean shortest path

* This work was done while the author was at Johns Hopkins University.
connecting $x$ and $y$ within $P$. This path, denoted by $\pi(x, y)$, is polygonal and it has all its corners at the vertices of $P$ (see Lee and Preparata [8]). The geodesic distance of $x$ and $y$ is denoted by $d(x, y)$, and it is the length of $\pi(x, y)$; $P$ is understood throughout. If $x$ is a point in $P$, then another point $y \in P$ is called a geodesic furthest neighbor of $x$ if $y$ is at the maximum geodesic distance from $x$, i.e.,

$$d(x, y) = \max \{d(x, z) | z \in P\}.$$

Let $\phi(x)$ denote the set of all geodesic furthest neighbors of $x$; in general, $x$ may have multiple geodesic furthest neighbors. It is known that a geodesic furthest neighbor is always a (convex) vertex of the polygon (see Asano and Toussaint [2]).

The geodesic diameter of $P$ is denoted by $D(P)$, and it is the maximum geodesic distance between any two points in $P$:

$$D(P) = \max \{d(x, y) | x, y \in P\}.$$

Clearly, the geodesic diameter is achieved between two vertices of $P$ and, therefore, we can compute it in linear time once the geodesic furthest neighbors for all the vertices are known (together with their associated distances). In the following discussion, we concentrate only on finding the geodesic furthest neighbors and omit the straightforward details needed to calculate the associated distances.

Our main result is an $O(n \log n)$ time and $O(n)$ space algorithm for computing geodesic furthest neighbors for all the vertices of $P$. A corollary of our result improves the previously known best bounds for computing the geodesic diameter of an $n$-gon: An $O(n^2)$ time algorithm was presented in Chazelle [3] and an $O(c^2n)$ time algorithm was given in Toussaint [11], where $c$ is the number of convex vertices of the polygon. Recently, Guibas and Hershberger [4] have discovered a different algorithm achieving the same bounds as ours.

The paper is organized in five sections. Section 2 develops some geometric preliminaries that are used later. Section 3 presents an $O(n \log n)$ time and $O(n)$ space algorithm for solving a restricted version of the geodesic furthest neighbors problem. In Section 4, it is shown that the geodesic furthest neighbors problem for $P$ can be solved using at most three instances of the restricted problem. Finally, we conclude in Section 5.

2. Preliminaries

If $\pi$ is a shortest path and $x$ a point, then $x \in \pi$ denotes the fact that $x$ lies on $\pi$. If $x$, $y$, and $z$ are three points in $P$, then the following triangle inequality always holds:

$$d(x, y) \leq d(x, z) + d(z, y).$$
Equality holds if and only if \( z \in \pi(x, y) \). Let \( x', y', z' \) be the three points such that
\[
\begin{align*}
\pi(x, x') &= \pi(x, y) \cap \pi(x, z) \\
\pi(y, y') &= \pi(y, z) \cap \pi(y, x) \\
\pi(z, z') &= \pi(z, x) \cap \pi(z, y).
\end{align*}
\]

Let the geodesic triangle of \( x, y, \) and \( z \), denoted by \( R(x, y, z) \), be the finite region of \( P \) that is bounded by \( \pi(x', y'), \pi(y', z'), \) and \( \pi(z', x') \). Observe that all vertices of \( R(x, y, z) \), with the exception of \( x', y', \) and \( z' \), are reflex and the sum of the interior angles at the three convex vertices is at most 180°. Furthermore, \( R(x, y, z) \) is empty if and only if at least one of \( x, y, \) or \( z \) lies on the shortest path joining the remaining two points.

If \( \pi_1 \) and \( \pi_2 \) are two shortest paths in \( P \), then we call them disjoint if they do not share a point, i.e., \( \pi_1 \cap \pi_2 = \emptyset \). Otherwise, we say that \( \pi_1 \) and \( \pi_2 \) intersect. Note that if two points \( x_1 \) and \( x_2 \) lie in the common intersection \( \pi_1 \cap \pi_2 \), then so does the shortest path connecting \( x_1 \) and \( x_2 \), i.e., \( \pi(x_1, x_2) \subseteq \pi_1 \cap \pi_2 \).

In the following, a traversal of \( P \) always proceeds in a counterclockwise order. Let \( p_1, p_2, p_3, p_4 \) be four points in this order on the boundary of \( P \). Uniqueness of geodesics in \( P \) easily implies that \( \pi(p_1, p_3) \) has the structure
\[
\pi(p_1, p_3) = \pi(p_1, a_1) \cup \pi(a_1, a_3) \cup \pi(a_3, p_3),
\]
where \( a_1 \in \pi(p_1, p_4), a_3 \in \pi(p_2, p_3), \) and \( \pi(a_1, a_3) \) is disjoint from both \( \pi(p_1, p_4) \) and \( \pi(p_2, p_3) \), except at the endpoints. See Fig. 1. Similarly, \( \pi(p_2, p_4) \) has the structure
\[
\pi(p_2, p_4) = \pi(p_2, a_2) \cup \pi(a_2, a_4) \cup \pi(a_4, p_4),
\]
where \( a_2 \in \pi(p_2, p_3), a_4 \in \pi(p_1, p_4), \) and \( \pi(a_2, a_4) \) is disjoint from both \( \pi(p_1, p_4) \) and \( \pi(p_2, p_3) \), except at the endpoints. If \( \pi(p_1, p_4) \) and \( \pi(p_2, p_3) \) intersect, then we let \( a_1 = a_2 = a_3 = a_4 = a \), for an arbitrary point \( a \in \pi(p_1, p_4) \cap \pi(p_2, p_3) \). Otherwise,

![Fig. 1. Junction points.](image-url)
\(\pi(p_1, p_4), \pi(p_2, p_3)\) are disjoint and \(a_1, a_2, a_3, a_4\) are uniquely defined. We call the points \(a_1, a_2, a_3, a_4\) the junction points for the shortest-path pair \(\pi(p_1, p_4), \pi(p_2, p_3)\). Notice that junction points need not be all distinct even if \(\pi(p_1, p_4)\) and \(\pi(p_2, p_3)\) are disjoint; however, at least one of \(a_1 \neq a_4\) or \(a_2 \neq a_3\) must hold. The following lemma determines the relative position of the junction points; we omit its straightforward proof.

**Lemma 1.** Let \(p_1, p_2, p_3, p_4\) be four points in this order on the boundary of \(P\). Let \(a_1, a_2, a_3, a_4\) be the junction points for the shortest-path pair \(\pi(p_1, p_4), \pi(p_2, p_3)\). Then, the following holds:

(i) \(d(p_1, a_1) \leq d(p_1, a_4)\)

(ii) \(d(p_2, a_2) \leq d(p_2, a_3)\).

Next we establish a key property of geodesic furthest neighbors, called the "crossing property." The property roughly states this: As we scan the vertices of \(P\) counterclockwise around the boundary, their geodesic furthest neighbors also move counterclockwise. However, since the geodesic furthest neighbors in general are not unique, we need to describe their counterclockwise motion more precisely, which we do in the following lemma.

**Lemma 2 (crossing property).** Let \(p_1, p_2, p_3, p_4\) be four points in this order on the boundary of \(P\). Suppose that \(p_3 \in \phi(p_2)\) and \(p_4 \in \phi(p_1)\). Then we also have \(p_3 \in \phi(p_1)\) and \(p_4 \in \phi(p_2)\).

**Proof.** The proof is by contradiction. Suppose that \(p_3 \notin \phi(p_1)\). Let \(a_1, a_2, a_3, a_4\) be the junction points for the pair \(\pi(p_1, p_4), \pi(p_2, p_3)\). We note the following inequalities; (1) holds because \(p_3 \in \phi(p_2)\), and (2) holds because \(p_4 \in \phi(p_1)\) but \(p_3 \notin \phi(p_1)\). Refer to Fig. 1.

\[
\begin{align*}
d(a_2, a_3) + d(a_3, p_3) & \geq d(a_2, a_4) + d(a_4, p_4) \quad (1) \\
d(a_1, a_4) + d(a_4, p_4) & > d(a_1, a_3) + d(a_3, p_3) \quad (2)
\end{align*}
\]

Inequality (3) now is obtained by adding together the corresponding sides of (1) and (2).

\[
d(a_2, a_3) + d(a_1, a_4) > d(a_2, a_4) + d(a_1, a_3). \quad (3)
\]

To finish the proof, we show that (3) is false. Consider a point \(e \in \pi(a_1, a_3) \cap \pi(a_2, a_4)\); such a point exists due to the relative positions imposed on the junction points by Lemma 1. Using the triangle inequality, we have

\[
\begin{align*}
d(a_2, a_3) + d(a_1, a_4) & \leq \{d(a_2, e) + d(e, a_3)\} + \{d(a_1, e) + d(e, a_4)\} \\
& \leq \{d(a_2, e) + d(e, a_4)\} + \{d(a_1, e) + d(e, a_3)\} \\
& \leq d(a_2, a_4) + d(a_1, a_3),
\end{align*}
\]
which contradicts (3), thus, disproving our (unfounded) hypothesis that $p_3 \notin \phi(p_1)$. The proof for $p_4 \in \phi(p_2)$ is analogous and we omit the details. The lemma follows. 

We use Lemma 2 to design a divide-and-conquer algorithm for computing the geodesic furthest neighbors of $P$. We first consider a restricted version of the problem, which is described below. Let $U$ and $V$ be two non-overlapping portions of the boundary of $P$. For each vertex $u \in U$, we want to find another vertex $v \in V$ whose geodesic distance from $u$ is maximum in $V$. We call this the **restricted furthest neighbors** problem. It turns out that the restricted furthest neighbors problem lends itself to a "cleaner" solution, and yet it has all the complexity of the general problem. In Section 4, we show that geodesic furthest neighbors for all the vertices of $P$ can be computed by solving at most three instances of the restricted furthest neighbors problem.

### 3. THE RESTRICTED FURTHEST NEIGHBORS PROBLEM

If $p_1$, $p_2$ are two vertices of $P$, then the boundary of $P$ between $p_1$ and $p_2$ in a clockwise (resp. counterclockwise) traversal is called the **clockwise chain** (resp. **counterclockwise chain**) from $p_1$ to $p_2$. Let $U = (u_1, u_2, ..., u_r)$ be a counterclockwise chain and let $V = (v_1, v_2, ..., v_t)$ be a clockwise chain of $P$. Furthermore, assume that $U$ and $V$ are disjoint, except perhaps at the endpoints. If $u \in U$ is a vertex, then we call a vertex $v \in V$ a **restricted furthest neighbor** of $u$ (with respect to $V$) if, among all vertices of $V$, $v$ has the maximum geodesic distance from $u$, i.e.,

$$d(u, v) = \max \{d(u, v_k) | 1 \leq k \leq t\}.$$  

Throughout, $V$ will be understood for all restricted furthest neighbors. Let $\psi(u)$ be the set of all restricted furthest neighbors of $u$. We show in this section how to

![Fig. 2. Polygon $P[a, b; c, d]$ shown shaded.](image-url)
compute restricted furthest neighbors for all the vertices of $U$ in $O(n \log n)$ time and linear space.

If $u_a$ and $u_b$, $a \leq b$, are two vertices of $U$ and $v_c$ and $v_d$, $c \leq d$, are two vertices of $V$, then our algorithm recursively solves problems of the following type: Find restricted furthest neighbors of the vertices $u_i$, $i = a, \ldots, b$, in the clockwise chain $(v_c, \ldots, v_d)$. To this end, we introduce the following notation. Let $P[a, b; c, d]$ denote the closed (finite) region of $P$ that is obtained by joining the counterclockwise chain $(u_a, \ldots, u_b)$ and the clockwise chain $(v_c, \ldots, v_d)$ by the shortest paths $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$. See Fig. 2. We say that $P[a, b; c, d]$ is degenerate if $\pi(u_a, v_c) \cap \pi(u_b, v_d) \neq \emptyset$; that is, $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ have one or more vertices of $P$ in common.

We say that a region $R \subseteq P$ is "geodesically convex" (relative to $P$) if, for any two points $x$ and $y$ in $R$, the shortest path $\pi(x, y)$ lies entirely in $R$. The following fact is easy to prove (see Pollack and Sharir [10] for similar ideas):

**Lemma 3.** $P[a, b; c, d]$ is geodesically convex.

Therefore, computation of geodesic furthest neighbors of $u_i$, $i = a, \ldots, b$, in $(v_c, \ldots, v_d)$ can be done entirely within $P[a, b; c, d]$. For ease of reference, we call this the "restricted furthest neighbors problem for $P[a, b; c, d]$." Our goal is to solve the restricted furthest neighbors problem for $P[1, s; 1, t]$. In the following, the size of a polygon denotes the total number of edges on its boundary.

First, consider the problem of computing a restricted furthest neighbor of a single vertex $u \in (u_a, \ldots, u_b)$ in the clockwise chain $(v_c, \ldots, v_d)$. If $P[a, b; c, d]$ is triangulated, then we can do this in time linear in the size of $P[a, b; c, d]$, as follows. We compute the shortest path tree (which is the union of the shortest paths to all the vertices) of $P[a, b; c, d]$ from $u$. This takes linear time, using a result of Guibas et al. [5]. Since a restricted furthest neighbor of $u$ always is a vertex, we can find it by simply comparing the geodesic distances from $u$ to all the vertices of $(v_c, \ldots, v_d)$, in linear time. We use this subroutine to recursively divide the problem of computing restricted furthest neighbors. To solve our problem, we invoke the following procedure on the polygon $P[1, s; 1, t]$. Refer to Fig. 3 for the following.

![Fig. 3. Algorithm RFN.](image_url)
Algorithm RFN\((P[a, b; c, d])\).

1. if \(P[a, b; c, d]\) is degenerate then
   2. Find a \(v_j \in (v_c, ..., v_d)\) such that \(d(u_i, v_j) = \max \{d(u_i, v) | c \leq j \leq d\}\);
   3. Output \(v_j\) as a restricted furthest neighbor of all \(u_i, i = a, ..., b\);
4. elseif \((b - a) \leq 2\) then
   5. for \(i = a\) to \(b\) do
      6. Find a \(v_j \in (v_c, ..., v_d)\) such that \(d(u_i, v_j) = \max \{d(u_i, v) | c \leq j \leq d\}\);
      7. Output \(v_j\) as a restricted furthest neighbor of \(u_i\);
   8. end
9. else
10. Let \(k = \lceil(a + b)/2\rceil\);
11. Find a \(v_j \in (v_c, ..., v_d)\) such that \(d(u_k, v_j) = \max \{d(u_i, v) | c \leq j \leq d\}\);
12. Construct and triangulate \(P[a, k; c, d]\) and \(P[k, b; c, d]\);
13. Recursively call RFN\((P[a, k; c, d])\) and RFN\((P[k, b; c, d])\).
end

3.1. Correctness of Algorithm RFN

Let \(P[a, b; c, d]\) be a subproblem considered at some stage of the recursion. Inductively assume that \((v_c, ..., v_d)\) contains a restricted furthest neighbor for all \(u_i, i = a, ..., b\). The induction hypothesis clearly holds for the basis case, namely, \(P[1, s; 1, t]\).

First, suppose that the algorithm executes steps 1–3. If \(P[a, b; c, d]\) is degenerate, then let \(x \in \pi(u_a, v_c) \cap \pi(u_b, v_d)\) be a point. It is easy to see that \(x\) lies on every shortest path \(\pi(u, v)\), where \(u \in (u_a, ..., u_b)\) and \(v \in (v_c, ..., v_d)\). It follows that, if \(v_j\) is a restricted furthest neighbor of some vertex \(u \in (u_a, ..., u_b)\), then \(v_j\) also is a restricted furthest neighbor of every other vertex \(u \in (u_a, ..., u_b)\). Therefore, by the induction hypothesis, the algorithm correctly computes the restricted furthest neighbors for all the vertices of \((u_a, ..., u_b)\) in step 3.

If the algorithm executes steps 4–8, then again the induction hypothesis ensures that the restricted furthest neighbors reported in step 7 are all correct.

Finally, consider the steps 9–13. Clearly, \(v_j \in \psi(u_k)\). The algorithm then recursively solves the two subproblems: Find restricted furthest neighbors of \(u_i, i = a, ..., k\), in the clockwise chain \((v_t, ..., v_d)\), and find restricted furthest neighbors of \(u_j, j = k, ..., b\), in the clockwise chain \((v_c, ..., v_t)\). Since \(v_j \in \psi(u_k)\), the crossing property (i.e., Lemma 2) guarantees that \((v_t, ..., v_d)\) contains a restricted furthest neighbor for each \(u_i, i = a, ..., k\) and that \((v_c, ..., v_t)\) contains a restricted furthest neighbor for each \(u_j, j = k, ..., b\). The induction hypothesis, therefore, holds for each of the two subproblems. This proves the correctness of the algorithm RFN.

3.2. Time Complexity of Algorithm RFN

Since the size of \(U\)-chain is halved at each step (see steps 10 and 12), the algorithm stops after \(\lceil \log s \rceil\) (which is \(O(\log n)\)) levels of recursion. We will establish the \(O(n \log n)\) bound on the running time of RFN\((P[1, s; 1, t])\) by proving that each level of recursion requires \(O(n)\) time. The proof consists of two part. First, we show that each invocation of the procedure RFN (excluding the recursive calls)
requires time linear in the size of the input polygon. Then, we show that the total size of all the polygons at any level of recursion is $O(n)$. We begin by considering the time spent by the algorithm on the input $P[a, b; c, d]$.

Assume inductively that the input polygon is triangulated. (An initial triangulation of $P[1, s; 1, t]$ may be obtained either by employing any of the well-known $O(n \log n)$-time algorithms (e.g., [3, 6]), or by employing the recently discovered $O(n \log \log n)$-time algorithm of Tarjan and Van Wyk [12].) Now, step 1 (i.e., checking whether $n(u, u)$ and $n(u, v)$ have a vertex in common) clearly takes time linear in size of $P[a, b; c, d]$. Steps 2–11 each can be performed in linear time, using the shortest-path tree algorithm of Guibas et al. [5]. Finally, to show that the two subpolygons created in step 12 can be triangulated in linear time, we exploit the fact that $P[a, b; c, d]$ is triangulated. We construct, in linear time, a shortest path tree of $P[a, b; c, d]$ from $u$, and refine the resulting map into a triangulation of each of the two (children) polygons. Details of this simple procedure are given in Appendix A. This completes the first part of our complexity analysis.

Next, for the second part of our proof, we want to show that the total size of all the polygons at each level of recursion is $O(n)$. We do this in two stages. First, we prove that the total number of distinct edges among all the polygons constructed during $RFN(P[1, s; 1, t])$ is $O(n)$, and then show that an edge may belong to only a constant number of polygons at any level of the recursion.

To count the number of distinct edges in all the polygons, we first introduce some notation. The edges of the polygon $P[a, b; c, d]$ either belong to the chains $(u, ..., u)$ and $(v, ..., v)$ or they belong to the shortest paths $\pi(u, v)$ and $\pi(u, v)$. We call the edges of the former kind, the primary edges and of the latter kind, the connector edges. The shortest paths, $\pi(u, v)$ and $\pi(u, v)$, are called the connectors of $P[a, b; c, d]$. Clearly, there are only $n$ distinct primary edges in all the polygons. In the following, we show that there only are a linear number of distinct connector edges among all the polygons constructed during $RFN(P[1, s; 1, t])$. This claim follows from a more general result proved below.

**Lemma 4.** Let $B$ be a simple polygon. Let $a, c$ be two arbitrary but fixed vertices of $B$ and let $b, d$ be two other vertices of $B$ such that $a, b, c, d$ appear in this order in a counterclockwise traversal of $B$. Then, all edges of $\pi(b, d)$, except perhaps one, belong to $E(a) \cup E(c)$, where $E(x)$ denotes the set of edges in the shortest path tree of $B$ from the point $x$.

**Proof.** If either $a$ or $c$ lies on $\pi(b, d)$, then the lemma follows trivially. Therefore, assume otherwise. Let $(x_1, ..., x_i, x_{i+1}, ..., x_j, x_{j+1}, ..., x_m)$ be the sequence of vertices of $\pi(b, d)$, where $x_i = b$, $x_m = d$, and where $i$ is the smallest index such that $x_i, x_{i+1} \notin E(a) \cup E(c)$ and $j$ is the largest index such that $x_j, x_{j+1} \notin E(a) \cup E(c)$. See Fig. 4. Let $B_a$ (resp. $B_c$) be the region of $B$ bounded by $\pi(b, d)$ and the counterclockwise (resp. clockwise) chain of $B$ between $d$ and $b$. We claim that, in $B_a$, $\pi(x_i, x_{i+1})$ is edge-disjoint from both $\pi(x_i, a)$ and $\pi(a, x_{i+1})$. The proof is by contradiction. Suppose that $\pi(x_i, x_{i+1})$ and $\pi(x_i, a)$ overlap on some edge $x_k x_{k+1}$,
where $i \leq k \leq j$. But, then $\pi(x_i, x_{k+1}) \subseteq \pi(x_i, a)$, which in turn implies that $x_i x_{i+1} \in E(a)$. A contradiction! A similar argument shows that $\pi(a, x_{j+1}) \cap \pi(x_i, x_{j+1}) = \{x_{j+1}\}$.

Because $x_i \neq x_{j+1}$, the geodesic triangle $R(x_i, x_{j+1}, a)$ is nonempty and its convex vertices are $x_i$, $x_{j+1}$, and $a_{ij}$, where $a_{ij}$ is a vertex satisfying $\pi(a, a_{ij}) = \pi(a, x_i) \cap \pi(a, x_{j+1})$. All the interior vertices of $\pi(x_i, x_{j+1})$, therefore, are reflex in $B_c$. Analogous reasoning will show that all the interior vertices of $\pi(x_i, x_{j+1})$ are reflex in $B_c$ also. This, however, is possible only if $\pi(x_i, x_{j+1})$ consists of at most one edge. In conclusion, at most one edge of $\pi(b, d)$ is not in $E(a) \cup E(c)$ and the lemma follows.  

To see how Lemma 4 implies that the total number of distinct connector edges among all the polygons constructed during $RFN(P[l, s; 1, t])$ is $O(n)$, we reason as follows. Every connector is a shortest path $\pi(u, v)$, for some $u \in U$ and $v \in V$. By slightly abusing the notation, let $E(u_1)$ (resp. $E(u_s)$) denote the set of edges in the shortest path tree of $P[l, s; 1, t]$ from the vertex $u_1$ (resp. $u_s$). If $\pi$ is a connector of some polygon constructed by $RFN$ at some stage of the recursion, then all edges of $\pi$, except perhaps one, belong to $E(u_1) \cup E(u_s)$, by Lemma 4. During its entire run, $RFN(P[l, s; 1, t])$ constructs at most $s$ polygons, and each polygon has only two connectors. The total number of connectors, therefore, is at most $2s \leq 2n$ and, with the possible exception of at most $2s \leq 2n$ edges, all edges of these connectors belong to $E(u_1) \cup E(u_s)$. Since $E(u_1)$ and $E(u_s)$ each has at most $n$ edges, the total number of distinct connector edges is no more than $4n$. In summary, the following fact has been established now.

**Lemma 5.** There are $O(n)$ distinct edges among all the polygons constructed by $RFN(P[l, s; 1, t])$.

We now are almost ready to prove that the total size of all the polygons at any
level of recursion is $O(n)$. If the input polygon is degenerate, then the procedure RFN solves the problem directly (i.e., with no further recursive calls). Therefore, we need only consider the non-degenerate polygons at any level. (This follows because if $P[a, b; c, d]$ is a degenerate polygon at level $i$ of the recursion, then the “parent” polygon of $P[a, b; c, d]$ is necessarily a non-degenerate polygon at level $i - 1$ of the recursion. That is, the total size of all the polygons is linearly related to the total size of all the non-degenerate polygons constructed during the course of the algorithm.) We show below that an edge may belong to only a constant number of non-degenerate polygons at any level of recursion. This would imply a linear upper bound on the total size of all non-degenerate polygons of the same level, because there are only a linear number of distinct edges.

The primary edges of any two polygons at the same level of recursion clearly are disjoint; it follows from the division step of our algorithm (see Step 12–13). We claim that, at any level of the recursion, an edge can be in the connectors of at most four non-degenerate polygons. Observe that this claim immediately implies the desired linear upper bound on the total size of all the non-degenerate polygons at any level of the recursion. (An edge is either a primary edge or a connector edge and, in either capacity, it lies in only a constant number of non-degenerate polygons of the same level.) We need some more preliminaries.

In the following discussion, we assume that a shortest path $\pi(x, y)$ is directed from $x$ to $y$: an edge $e$ of $\pi(x, y)$ is directed from its endpoint $a$ to the endpoint $b$ if $a$ is (geodesically) closer to $x$ than $b$. In this way, the two copies of $e$, one directed from $a$ to $b$ and the other directed from $b$ to $a$, are distinguished from each other.

**Lemma 6.** Let $a_1, a_2, a_3, b_1, b_2, b_3$ be six point in this order in a counterclockwise traversal of $P$. Suppose that the directed shortest paths $\pi(a_1, b_1)$ and $\pi(a_3, b_3)$ have a directed edge $e$ in common. Then, the same directed edge $e$ also is included in the shortest path $\pi(a_2, b_2)$.

**Proof.** Let $z$ be an interior point of $e$. We show that $z$ lies on $\pi(a_2, b_2)$, which proves the lemma. Without loss of generality, suppose that $e$ is vertical, is directed upward from its lower endpoint $v_a$ to the upper endpoint $v_b$. Let $xy$ be the maximal chord in $P$ through $z$ that is perpendicular to $e$. Clearly, $a_i$ lie below $xy$ and $b_i$ lie above $xy$, for $i = 1, 2, 3$. Therefore, $\pi(a_i, b_i)$ intersects $xy$. Let us assume, for a contradiction, that the point of intersection is $z' \neq z$.

Without loss of generality, assume that $z'$ lies to the right of $z$ on $xy$. See Fig. 5. Let $b'_2$ be a point above $xy$ that is in the intersection of $\pi(a_1, b_1)$, and $\pi(a_2, b_2)$, and let $a'_2$ be a point below $xy$ that is in the intersection of $\pi(a_3, b_3)$ and $\pi(a_2, b_2)$; the assumed position of $z'$ ensures that these points exist. Since $\pi(a'_2, b'_2)$ goes through $z'$ but not through $z$, the geodesic triangle $\triangle(a'_2, z, b'_2)$ is nonempty, implying that the interior angle at each of its three convex vertices is nonzero.

Next, we note that $\pi(z, b'_2)$ goes through $v_b$ and $\pi(a'_2, z)$ goes through $v_a$; $v_a$ and $v_b$ being the endpoints of $e$. (This follows because $z$, $v_b$ and $b'_2$ lie on $\pi(a_1, b_1)$ in that order order, and $\pi(a'_2, z)$ lies on $\pi(a_3, b_3)$ in that order.) Therefore, $z$ is
convex vertex of $R(a'_2, z, b'_2)$. Since the interior angle at $z$ is $180^\circ$, the interior angles at the remaining two vertices must be zero. But, this contradicts our previous assertion that these angles are all nonzero. The point $z$, therefore, must lie on $\pi(a_2, b_2)$ and the lemma follows. 

We now employ Lemma 6 to prove that a directed edge belongs to the connectors of at most two non-degenerate polygons at any level of recursion.

**Lemma 7.** Let $P[a_1, b_1; c_1, d_1]$, $P[a_2, b_2; c_2, d_2]$, and $P[a_3, b_3; c_3, d_3]$ be three non-degenerate polygons that occur at the same level of recursion in $RFN(P[1, s; 1, t])$ such that $a_1 < b_1 < a_2 < b_2 < a_3 < b_3$ and $c_1 < d_1 < c_2 < d_2 < c_3 < d_3$. Then, the directed connectors of $P[a_1, b_1; c_1, d_1]$ and $P[a_3, b_3; c_3, d_3]$ are edge-disjoint.

**Proof.** The proof is by contradiction. Suppose that the two directed connectors $\pi(x_1, y_1)$ and $\pi(x_3, y_3)$ share a (directed) edge, say, $e$, where $x_1 \in \{u_{a_1}, u_{b_1}\}$ and $y_i \in \{v_{c_i}, v_{d_i}\}$, for $i \in \{1, 3\}$. Then, by Lemma 6, $e$ is included in any shortest path $\pi(x, y)$, where $x$ (resp. $y$) lies in the (closed) counterclockwise chain between $x_1$ and $x_3$ (resp. $y_1$ and $y_3$). Therefore, $\pi(u_{a_1}, v_{c_1})$ and $\pi(u_{b_3}, v_{d_3})$ each must also include $e$, which contradicts the hypothesis that $P[a_2, b_2; c_2, d_2]$ is non-degenerate. The lemma follows.

Since there are $O(n)$ distinct edges, each edge can be directed in only two ways, and each directed edge lies in at most a constant number of non-degenerate polygons at each level, the total size of all the polygons considered at any level of the recursion is $O(n)$. Further, since there are $O(\log n)$ such levels and, at each level, the time spent is linear in the total size of all the polygons, the running time
of the algorithm $RFN(P[1, s; 1, t])$ is $O(n \log n)$. The space requirement clearly is linear.

**Theorem 1.** Let $P$ be a simple polygon of $n$ vertices. Let $U = (u_1, u_2, ..., u_n)$ and $V = (v_1, v_2, ..., v_t)$ be two nonoverlapping chains of $P$. Then, there is an algorithm that computes, for all vertices of $U$, their restricted furthest neighbors in $V$, in time $O(n \log n)$ and space $O(n)$.

In the next section, we show that geodesic furthest neighbors for all the vertices of $P$ can be computed by solving at most three instances of the restricted furthest neighbors problem.

### 4. Computing Geodesic Furthest Neighbors for Vertices of $P$

In the following, we always traverse $P$ in a counterclockwise order. Let $(u_1, u_2, ..., u_n)$ be the sequence of vertices of $P$. The notation $(u_a, ..., u_b)$ denotes the counterclockwise chain of $P$ between the vertices $u_a$ and $u_b$; the indices increment cyclically, with the convention that $u_{n+1} = u_1$. Our problem is to compute geodesic furthest neighbors for all the vertices $u_i$, $i = 1, 2, ..., n$. (Recall that $y$ is a geodesic furthest neighbor of $x$, if $d(x, y) = \max \{d(x, z) | z \in P\}$.)

Pick an arbitrary vertex $u_i$ of $P$. Let $u_j \in \phi(u_i)$ be a geodesic furthest neighbor of $u_i$ and let $u_k \in \phi(u_j)$ be a geodesic furthest neighbor of $u_j$; $u_i = u_k$ is possible but not necessary. Assume, without loss of generality, that $u_i, u_j,$ and $u_k$ are in this order in a counterclockwise traversal of $P$. Our key lemma is stated below. Refer to Fig. 6.

**Lemma 8.** Under the conditions stated above,

(A) for any vertex $u_i \in (u_i, ..., u_j)$, there exists another vertex $u_m \in (u_j, ..., u_i)$ satisfying $u_m \in \phi(u_i)$.

![Fig. 6. Illustration for the proof of Lemma 8.](image-url)
(B) For any vertex \( u_i \in (u_j, \ldots, u_k) \), there exists another vertex 
\( u_m \in (u_k, \ldots, u_i, \ldots, u_j) \) satisfying \( u_m \in \phi(u_i) \).

(C) For any vertex \( u_i \in (u_k, \ldots, u) \), there exists another vertex 
\( u_m \in (u, \ldots, u_j, \ldots, u_k) \) satisfying \( u_m \in \phi(u_i) \).

Proof. The proofs for (A) and (B) are similar, and we give details only for (A). Let \( u_i \in (u_j, \ldots, u_j) \) be a vertex and let \( u_m \) be a geodesic furthest neighbor of \( u_i \). If 
\( u_m \in (u_j, \ldots, u_k, \ldots, u_i) \), then we are done. Otherwise, we proceed as follows. If, in a 
(counterclockwise) traversal of \( (u_j, \ldots, u_j) \), \( u_l \) precedes \( u_m \), then we have the four 
points \( u_i, u_l, u_m, u_j \) that are in this order and that satisfy \( u_m \in \phi(u_i) \) and \( u_l \in \phi(u_j) \). But then Lemma 2 ensures that we also have \( u_j \in \phi(u_i) \), and we are done. If, instead, 
\( u_m \) precedes \( u_l \) in the traversal of \( (u_j, \ldots, u_j) \), then we have the four points \( u_i, u_l, u_k, u_m \) 
that are in this order and that satisfy \( u_m \in \phi(u_j) \) and \( u_k \in \phi(u_j) \). Again, Lemma 2 
ensures that we also have \( u_k \in \phi(u_i) \). This finishes the proof for (A).

Next, we consider (C). Again, let \( u_i \in (u_k, \ldots, u_i) \) be a vertex and let \( u_m \) be a 
geodesic furthest neighbor of \( u_i \). If \( u_m \in (u_i, \ldots, u_j, \ldots, u_k) \), then we are done 
Otherwise, proceed as follows. If, in the traversal of \( (u_k, \ldots, u_i) \), \( u_m \) precedes \( u_l \), then we 
have the four points \( u_i, u_l, u_m, u_j \) that are in this order and that satisfy \( u_m \in \phi(u_i) \) and 
\( u_j \in \phi(u_l) \). By Lemma 2, we also have \( u_l \in \phi(u_i) \), and we are done. If, on the 
other hand, \( u_l \) precedes \( u_m \) in the traversal of \( (u_k, \ldots, u_i) \), then we give the following 
two-stage proof.

First, suppose that there is a vertex \( u_p \in (u_i, \ldots, u_j, \ldots, u_k) \) that is a geodesic 
furthest neighbor of \( u_k \). Then, we get the four points \( u_k, u_l, u_m, u_p \) (in this order) 
that satisfy \( u_m \in \phi(u_l) \) and \( u_k \in \phi(u_p) \). Lemma 2 guarantees that \( u_p \in \phi(u_i) \), and we 
are done. Hence, to finish the proof, we only need to prove the existence of the 
desired vertex \( u_p \). This is done in the following.

Suppose, for a contradiction, that \( u_p \) does not exist, i.e., no vertex of 
\( (u_i, \ldots, u_j, \ldots, u_k) \) is a geodesic furthest neighbor of \( u_k \). Consider a geodesic furthest 
nearbor of \( u_k \), say, \( u_q \). By hypothesis, \( u_q \) lies in \( (u_{k+1}, \ldots, u_{i-1}) \). We note the 
following inequalities: they use the assumptions that \( u_j \in \phi(u_i) \) and \( u_k \in \phi(u_j) \).

\[
d(u_i, u_j) \geq d(u_i, u_k) \tag{4}
\]
\[
d(u_j, u_k) \geq d(u_j, u_q) \tag{5}
\]

The following inequality holds because \( u_j \) is not a geodesic furthest neighbor of \( u_k \), 
but \( u_q \) is. Thus,

\[
d(u_k, u_q) > d(u_k, u_j). \tag{6}
\]

Summing up the corresponding sides of (4), (5) and (6), we get

\[
d(u_i, u_j) + d(u_k, u_q) > d(u_i, u_k) + d(u_j, u_q),
\]

which violates the triangle inequality. Therefore, our assumption that 
\( (u_i, \ldots, u_j, \ldots, u_k) \) does not contain any geodesic furthest neighbor of \( u_k \) must be false. 
This completes our proof of the lemma.
To compute geodesic furthest neighbors for the vertices of \( P \), we first triangulate \( P \). We then pick an arbitrary vertex \( u_i \). Let \( u_j \) be a geodesic furthest neighbor of \( u_i \), and let \( u_k \) be a geodesic furthest neighbor of \( u_j \). We can compute \( u_j \) and \( u_k \) in time \( O(n) \), using the shortest-path tree algorithm of Guibas et al. [5]. Suppose, without loss of generality, that \( u_i, u_j, \) and \( u_k \) are in this order in a (counterclockwise) traversal of \( P \). We compute restricted furthest neighbors of the vertices of \( U_i \) in the chain \( V_i, i = 1, 2, 3 \), where

Problem 1. \( U_1 = (u_j, \ldots, u_j) \) and \( V_1 = (u_j, \ldots, u_k, \ldots, u_i) \).

Problem 2. \( U_2 = (u_j, \ldots, u_k) \) and \( V_2 = (u_k, \ldots, u_j, \ldots, u_j) \).

Problem 3. \( U_3 = (u_k, \ldots, u_i) \) and \( V_3 = (u_i, \ldots, u_j, \ldots, u_k) \).

(Here both \( U_i \) and \( V_i \) are given as counterclockwise sequences of vertices, \( i = 1, 2, 3 \).) If \( u_k \) and \( u_i \) are the same, then we need to solve only two instances of the problem. By solving Problems 1, 2, 3, we compute a restricted furthest neighbor for each vertex \( u_i \) of \( P \). By Lemma 8, this restricted furthest neighbors also is a geodesic furthest neighbor of \( u_i \), for all \( i \). Since Problems 1, 2, and 3 each can be solved in \( O(n \log n) \) time and \( O(n) \) space, we obtain the following theorem.

**Theorem 2.** Let \( P \) be a simple polygon of \( n \) vertices. Then, there is an algorithm that computes geodesic furthest neighbors for all the vertices of \( P \) in time \( O(n \log n) \) and space \( O(n) \).

The geodesic distance between the vertices of \( P \) and their geodesic furthest neighbors also can be computed within the same time and space bounds. Clearly, the geodesic diameter is the maximum such distance over all vertices of \( P \), i.e.,

\[
D(P) = \max \{d(u, v) | u \in P \text{ and } v \in \phi(u) \},
\]

where \( u \) and \( v \) are the vertices of \( P \). The following result, therefore, is an easy corollary of Theorem 2.

**Theorem 3.** Let \( P \) be a simple polygon of \( n \) vertices. Then, there is an algorithm for computing the geodesic diameter of \( P \) in time \( O(n \log n) \) and space \( O(n) \).

5. DISCUSSION

We have presented an \( O(n \log n) \) time and \( O(n) \) space algorithm for computing geodesic furthest neighbors of all the vertices of an \( n \)-vertex polygon. As a corollary, the geodesic diameter of the polygon also can be computed within the same bounds. Our algorithm uses the linear-time shortest-path-tree algorithm of Guibas et al. [5] as a subroutine. The linearity of their algorithm relies heavily on a sophisticated data structure, called finger tree, which in practice is quite com-
plicated to implement. However, as Guibas et al. point out, the shortest-path tree algorithm can be implemented in $O(n \log n)$ time using a much simpler data structure, called doubly-linked linear list. Using this simpler version of the shortest-path tree algorithm, our algorithm for computing geodesic furthest neighbors can be implemented in time $O(n \log^2 n)$, which, though asymptotically inferior, may be more practical.

The geodesic diameter has applications in pattern recognition and image processing. For instance, Lantuejoul and Maisonneuve [7] have shown that all the classical morphological transformations (dilation, erosion, skeletonization, etc.) as well as the topological transformations can be defined rigorously in the metric space $(X, d)$, where $X$ is some part of an image to be analyzed and $d$ is the geodesic distance function. Lantuejoul and Maisonneuve use geodesic distance to formulate robust definitions of entities such as the “length of a fiber,” the “center of an image,” and other regional extrema. Since images often are modeled as polygons, the length of an image is captured by the geodesic diameter of the associated polygon.

APPENDIX A

Let $u$ be a vertex of the polygon $P$. Consider the straight-line planar map defined by the edges of the shortest path tree of $P$ from $u$. Each face of this planar map is either a triangle or a ‘funnel.’ See Fig. 7 for the illustration of a funnel, and refer to Lee and Preparata [8] or Guibas et al. [5] for more details on funnels. Funnels are such simple structures that they easily can be triangulated in linear time. It follows that, given a shortest path tree (with all the adjacency information), a polygon can be triangulated in linear time.

Now, consider the two polygons, $P[a, k; l, d]$ and $P[k, b; c, l]$, that are constructed out of $P[a, b; c, d]$ in step 12 of the algorithm RFN. We will show how to obtain a triangulation of $P[a, k; l, d]$ in time proportional to its size, given that $P[a, b; c, d]$ is triangulated; the procedure for triangulating the other polygon, $P[k, b; c, l]$, is identical. Construct the shortest path tree of $P[a, b; c, d]$ from the vertex $u_k$, and delete from it all the vertices that do not lie in $P[a, k; l, d]$ (along with their incident edges). Let $M[a, k; l, d]$ be the planar map defined by the

![Fig. 7. Illustration of a funnel (shade area).](image-url)
remaining edges of the shortest path tree. The map \( M[a, k; l, d] \) can be constructed in time proportional to the size of \( P[a, b; c, d] \), using the shortest-path tree algorithm of Guibas et al. [5]. We claim that edges of the map \( M[a, k; l, d] \) constitute the shortest path tree of \( P[a, k; l, d] \) from the vertex \( u_k \). This easily follows from the observation that \( P[a, k; l, d] \) is geodesically convex and, therefore, the shortest path from any vertex \( u \in P[a, k; l, d] \) to \( u_k \) uses only the vertices that are in \( P[a, k; l, d] \). Therefore, we have the following two facts:

1. the faces of \( M[a, k; l, d] \) partition the polygon \( P[a, k; l, d] \), and
2. each face of \( M[a, k; l, d] \) is either a triangle or it is a funnel contained in \( P[a, k; l, d] \).

It now is easy to refine the map \( M[a, k; l, d] \) into a triangulation of \( P[a, k; l, d] \) in linear time.

ACKNOWLEDGMENTS

I am thankful to Boris Aronov who helped simplify the main algorithm of this paper. Many clarifying discussions with Joseph O'Rourke are acknowledged. Perceptive comments by Greg Sullivan on an earlier draft have improved the quality of this paper. This work was partially supported by NSF Grant DCR–83–51486 and grants from General Motors, IBM, and Martin-Marietta.

REFERENCES

11. G. T. Toussaint, Computing geodesic properties of polygons, manuscript, School of Computer Science, McGill University, 1986.