4.   Applications

Meisters' [Me] Two-Ears Theorem was motivated by the problem of triangulating a simple polygon. In fact Meisters suggests a greedy, but concise algorithm to achieve this goal, i.e., "find an ear and cut it off." Continued application of this operation to the smaller remaining polygon yields eventually a triangulation. A naive implementation of this procedure yields an algorithm with running time $O(n^3)$. The results presented here show that the greedy ear-cutting approach to triangulating a simple polygon can be implemented in $O(n^2)$ time.

In [To] a new algorithm for triangulating a simple polygon is proposed that runs in time $O(n(1+t_0))$, $t_0 < n$, where t_0 denotes the number of triangles in the triangulation that share no edges with the polygon. The algorithm first finds a diagonal as a starting place after which the main procedure is applied twice to the two remaining sub-polygons. The results of this note suggest a more elegant and concise implementation of this algorithm, i.e., find an ear as an initialization step and then apply the main procedure only once.

5.   References

[Me]   Meisters, G. H., "Polygons have ears," *American Mathematical Monthly*, June/July 1975, pp. 648-651.

[Le]   Levy, L. S., *Geometry: Modern Mathematics via the Euclidean Plane*, Prindle, Weber & Schmidt, Inc., Boston, Mass., 1970.

[Ho]   Ho, W. C., "Decomposition of a polygon into triangles," *The Mathematical Gazette,* vol. 59, 1975, pp. 132-134.

[To]   Toussaint, G. T., "An Output-Complexity-Sensitive Polygon Triangulation Algorithm," *Snapshots of Computational & Discrete Geometry,* G. T. Toussaint, *Editor*, Tech. Rept. SOCS-88.11, McGill University, June 1988, pp. 55-68.

[TV]   Tarjan, R. E. and Van Wyk, C. J., "An O(n log log n)-time algorithm for triangulating simple polygons," *SIAM Journal on Computing*, 1988.

(p_i, p_{i+1}, y) are empty then so is (p_{i-1}, p_i, p_{i+1}) which is a contradiction. Suppose y lies inside of triangle (p_{i-1}, p_i, p_{i+1}). Since triangle (p_{i-1}, p_i, y) is empty, p_{k+1} does not lie in triangle (p_{i-1}, p_i, y). Similarly, p_k does not lie in triangle (p_i, p_{i+1}, y). Therefore there is no place for segment $p_k p_{k+1}$. *Case 2:* Since $z = p_{i-1}$, p_k does not lie between rays $p_i y$ and $p_i p_{i-1}$. Similarly, p_{k+1} does not lie between rays $p_i y$ and $p_i p_{i+1}$. Since p_{k+1} lies in the same half-plane defined by *ray*(p_i) as p_{i-1} and p_k and p_{i+1} lie in the other there is no place for segment $p_k p_{k+1}$. Q.E.D.

Lemma 4: Finding a vertex p_j such that (p_i, p_j) is a diagonal can be done in linear time.

Proof: It is clear that the construction described in the proof of Lemma 3 can be implemented to run in linear time. Q.E.D.

3.   The Algorithm

We now describe the algorithm. The recursive function FindAnEar takes as input a good sub-polygon and a vertex. Initially we call FindAnEar with the simple polygon P and any vertex of P.

ALGORITHM *FindAnEar* (GSP,p_i)

Given a good sub-polygon GSP of a polygon P and a vertex p_i of GSP this algorithm reports a proper ear.

1. If p_i is an ear report it and exit.
2. Find a vertex p_j such that (p_i, p_j) is a diagonal of GSP. Let GSP′ be the good sub-polygon of GSP formed by (p_i, p_j). Relabel the vertices of GSP′ so that $p_i = p_0$ and $p_j = p_{k-1}$ (or $p_j = p_0$ and $p_i = p_{k-1}$ as appropriate) where k is the number of vertices of GSP′.
3. FindAnEar(GSP′, $\lfloor k/2 \rfloor$)).

END *FindAnEar*

The correctness of the algorithm follows from Lemmas 1, 2 and 3.

Theorem: Algorithm FindAnEar runs in $O(n)$ time.

Proof: Clearly Step 1 can be done in time linear in the number of vertices in GSP. By Lemma 4, Step 2 can also be done in time linear in the number of vertices in GSP. On the first two calls to FindAnEar, GSP has $O(n)$ vertices. Consider any subsequent call. Let k be the number of vertices in GSP. We have $i = \lfloor k/2 \rfloor$ and (p_0, p_{k-1}) the cutting edge of GSP. Consider Step 2. If $0 \le j \le i-2$ then GSP′ $= (p_j, p_{j+1}, ..., p_i)$. Otherwise, $i+2 \le j \le k-1$ and GSP′ $= (p_i, p_{i+1}, ..., p_j)$. In either case, GSP′ contains no more than $\lfloor k/2 \rfloor + 1$ vertices. Q.E.D.
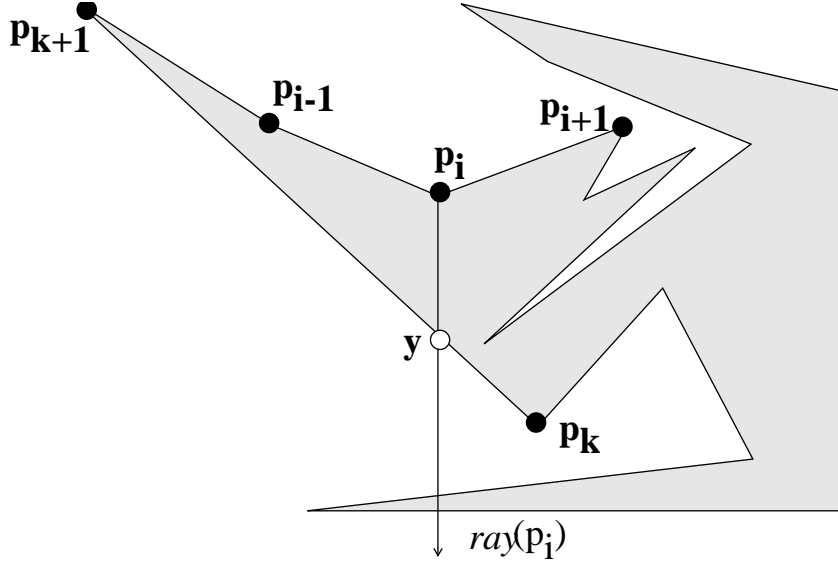
Figure 1. Constructing a diagonal.

tersection point of $ray(p_i)$ with the boundary of P. Note that is done simply by testing each edge of the polygon for intersection with $ray(p_i)$. Let y be the intersection point on edge (p_k, p_{k+1}). It is clear that y must exist and that $y \neq p_{i-1}$ or p_{i+1}. Note that the line segment (p_i, y) lies entirely inside P. Thus if y is a vertex then (p_i, y) is a diagonal. Suppose y is not a vertex. Then p_{k+1} and p_{i-1} lie in one of the half-planes defined by $ray(p_i)$ and p_k and p_{i+1} lie in the other. We will show that if triangle (p_i, y, p_{k+1}) does not contain a vertex p_j such that (p_i, p_j) is a diagonal of P then triangle (p_i, y, p_k) does.

Let $R = \{p_r \in P$ such that p_r lies in triangle (p_i, y, p_{k+1}), $k+1 < r < i\}$. If $R = \varnothing$ then by the Jordan Curve Theorem and the fact that line segment (p_i, y) lies entirely inside P, the interior of triangle (p_i, y, p_{k+1}) is empty. If $p_{k+1} \neq p_{i-1}$ then (p_i, p_{k+1}) is a diagonal. Otherwise let $z = p_{i-1}$. If $R \neq \varnothing$ then for all $p_r \in R$ compute $\angle yp_ip_r$ and let z be the vertex that minimizes this angle. By choice of z, the interior of triangle (p_i, y, z) is empty. Thus if $z \neq p_{i-1}$ then (p_i, z) is a diagonal. Hence if no diagonal has been found thus far then $z = p_{i-1}$. Similarly define $S = \{p_s \in P$ such that p_s lies in triangle (p_i, y, p_k), $\iota < s < k\}$. If $S = \varnothing$ the interior of triangle (p_i, y, p_k) is empty and if $p_k \neq p_{i+1}$ then (p_i, p_k) is a diagonal. Define w analogously to z, i.e., either $w = p_{i+1}$ or w is the vertex that minimizes $\angle yp_ip_s$. By choice of w the interior of triangle (p_i, y, w) is empty. We will show that $w \neq p_{i+1}$ and then it follows that (p_i, w) is a diagonal. Recall that $z = p_{i-1}$ so that triangle (p_{i-1}, p_i, y) is empty. Assume that $w = p_{i+1}$ so that triangle (p_i, y, p_{i+1}) is empty.

Case 1: p_i is a convex vertex. Since p_i is not an ear at least one vertex of P lies in triangle (p_{i-1}, p_i, p_{i+1}). Suppose y lies outside of triangle (p_{i-1}, p_i, p_{i+1}). Since triangle (p_{i-1}, p_i, y) and triangle

the right as the edges are traversed.

A vertex p_i of a simple polygon P is called an ear if the line segment (p_{i-1}, p_{i+1}) lies entirely in P. We say that two ears p_i and p_j are non-overlapping if the interior of triangle (p_{i-1}, p_i, p_{i+1}) does not intersect the interior of triangle (p_{j-1}, p_j, p_{j+1}). Meisters [Me] has proven the following theorem.

Two-Ears Theorem: Except for triangles every simple polygon has at least two non-overlapping ears.

A good sub-polygon of a simple polygon P, denoted by GSP, is a sub-polygon whose boundary differs from that of P in at most one edge. We call this edge, if it exists, the cutting edge. A proper ear of a good sub-polygon GSP is an ear of GSP which is also an ear of P.

Lemma 1: A good sub-polygon has at least one proper ear.

Proof. Let (p_i, p_j) be the cutting edge of GSP. By the Two-Ears Theorem GSP has at least two non-overlapping ears. It cannot be the case that the only ears of GSP are p_i and p_j since these would overlap. Thus some other vertex of GSP is an ear and it is a proper ear. Q.E.D.

The strategy of the algorithm is as follows. Given a polygon P on n vertices, split it in $O(n)$ time into two sub-polygons such that one of these sub-polygons is a good sub-polygon with at most $\lfloor n/2 \rfloor + 1$ vertices. This splitting step is the crucial step in the algorithm. Subsequently, apply the algorithm recursively to this good sub-polygon which, by Lemma 1, is guaranteed to have a proper ear. The worst case running time of the algorithm is given by the recurrence $T(n) = cn + T(\lfloor n/2 \rfloor + 1)$, where c is a constant, which has solution $T(n) \in O(n)$.

We require several lemmas which concern the splitting step. The line segment joining two non-consecutive vertices p_i and p_j of P is called a *diagonal* of P if it lies entirely inside P.

Lemma 2: A diagonal of a good sub-polygon GSP splits GSP into one good sub-polygon and one sub-polygon that is not good.

Proof. GSP contains exactly one edge, the cutting edge, which is not an edge of P. The cutting edge is entirely contained in one of the sub-polygons formed by the diagonal. Then the other sub-polygon is a good sub-polygon since it consists of edges of P and the diagonal which becomes its cutting edge. Q.E.D.

The proof of the following lemma is a generalization of Levy's proof of the existence of diagonals [Le].

Lemma 3: If vertex p_i is not an ear then there exists a vertex p_j such that (p_i, p_j) is a diagonal of P.

Proof: Given a vertex p_i which is not an ear we will show how to construct a diagonal (p_i, p_j). Refer to Figure 1. Construct a ray, *ray*(p_i), at p_i that bisects the interior of $\angle p_{i-1} p_i p_{i+1}$. Find the first in-

Slicing an Ear in Linear Time

Hossam ElGindy
Hazel Everett
Godfried Toussaint

School of Computer Science
McGill University
Montreal

ABSTRACT

It remains as one of the major open problems in computational geometry, whether there exists a linear-time algorithm for triangulating a simple polygon P. Yet it is well known that a diagonal of P can easily be found in linear time. In this note we show that an *ear* of P can be found in linear time. An *ear* is a triangle such that one of its edges is a diagonal of P and the remaining two edges are edges of P. Applications of this result are indicated.

1.   Introduction

The triangulation of simple polygons has received much attention in the computational geometry literature because of its many applications in such areas as pattern recognition, computer graphics, CAD and solid modeling. Nevertheless, it remains as one of the major open problems in computational geometry, whether there exists a linear-time algorithm for triangulating a simple polygon P. The fastest algorithm to date is due to Tarjan & Van Wyk [TV] and runs in $O(n \log \log n)$ time, where n is the number of vertices of P. On the other hand, it is well known that several linear-time algorithms exist for finding a single diagonal in P and most polygon triangulation algorithms incorporate one of these as a subroutine [To]. Although finding a diagonal is straightforward, its simplicity is deceiving, and, as shown in [Ho], several published algorithms are in fact incorrect. In this note we generalize this result to finding an ear. We show that an ear of P can be found in linear time. An ear is a triangle such that one of its edges is a diagonal of P and the remaining two edges are edges of P. Two applications of this result are indicated.

2.   Preliminaries

A polygon P is a closed path of straight line segments. A polygon is represented by a sequence of vertices $P = (p_0, p_1, ..., p_{n-1})$ where p_i has real-valued x,y-coordinates. We assume that no three vertices of P are collinear. The line segments (p_i, p_{i+1}), $0 \le i \le n-1$, (subscript arithmetic taken modulo n) are the edges of P. A polygon is simple if no two nonconsecutive edges intersect. We assume that the vertices are given in clockwise order so that the interior of the polygon lies to