Pattern Recognition Letters 14 (1993) 719–722
North-Holland

September 1993

PATREC 1149

Slicing an ear using prune-and-search

Hossam ElGindy

Department of Computer Science, University of Newcastle, Australia

Hazel Everett

Department of Mathematics and Computer Science, University of Quebec, Montreal, Canada

Godfried Toussaint

School of Computer Science, McGill University, Montreal, Canada

Received 16 March 1993

Abstract

ElGindy, H., H. Everett and G. Toussaint, Slicing an ear using prune-and-search, Pattern Recognition Letters 14 (1993) 719–722.

It is well known that a *diagonal* of a simple polygon P can be found in linear time with a simple and practically efficient algorithm. An *ear* of P is a triangle such that one of its edges is a diagonal of P and the remaining two edges are edges of P. An ear of P can easily be found by first triangulating P and subsequently searching the triangulation. However, although a polygon can be triangulated in linear time, such a procedure is conceptually difficult and not practically efficient. In this note we show that an *ear* of P can be found in linear time with a simple, practically efficient algorithm that does not require pre-triangulating P.

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-CAM, robotics and solid modelling. Until recently the fastest algorithm available (in theory) was due to Tarjan and Van Wyk [6]. Their algorithm runs in $O(n \log\log n)$ time, where n is the number of vertices of P but makes use of complicated data structures. Although a linear time algorithm eluded many researchers for more than a decade Bernard Chazelle finally established that such an algorithm exists [1]. In contrast with the triangulation problem, it is well known that several, practically efficient and conceptually simple, linear-time algorithms exist for finding a single *diagonal* in P. In fact most polygon triangulation algorithms incorporate one of these as a subroutine [5]. Although finding a diagonal is straightforward, its conceptual simplicity is deceiving, and, as shown in [4], several published algorithms are in fact incorrect. In this note we consider the problem of finding an ear. An *ear* of P is a triangle such that one of its edges is a diagonal of P and the remaining two edges are edges of P. An ear of P can easily be

Correspondence to: G.T. Toussaint, Computational Geometry Lab., McGill University, 3480 University Street, Montreal, Quebec, Canada H3A 2A7.

found by first triangulating P and subsequently searching the triangulation. However, although a polygon can be triangulated in linear time, such a procedure is conceptually difficult and not practically efficient. In this note we show that an *ear* of P can be found in linear time with a simple, practically efficient algorithm that does not require pre-triangulating P.

2. Preliminaries

A polygon P is a closed path of n ($n>3$) straight line segments. A polygon is represented by a sequence of vertices $P=(p_0, p_1, \ldots, p_{n-1})$ 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}), $0 \leqslant i \leqslant n-1$, (subscript arithmetic taken modulo n) are the edges of P. A polygon is simple if no two non-consecutive edges intersect. We assume that the vertices are given in clockwise order so that the interior of the polygon lies to the right as the edges are traversed. 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.

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 [2] 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 not an endpoint of the cutting edge 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. □

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.

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. □

The proof of the following lemma is a generalization of Levy's proof of the existence of diagonals [3].

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 find a vertex p_j such that (p_i, p_j) is a diagonal. Refer to Figure 1. Construct a ray, $ray(p_i)$, at p_i that bisects the interior of $\angle p_{i-1}p_ip_{i+1}$. Find the first intersection point of $ray(p_i)$ with the boundary of P. Note that this 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.

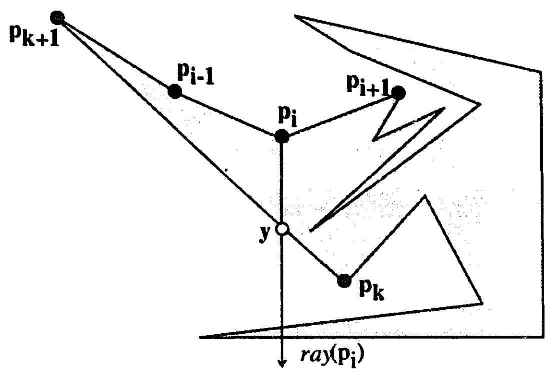

Figure 1. Constructing a diagonal.

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 $R=\emptyset$ 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 $R=\emptyset$ 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 $(p_i, y, p_k), i<s<k\}$. If $S=\emptyset$, 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, y, p_{i-1}) 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, y, p_{i-1}) and triangle (p_i, y, p_{i+1}) 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, y, p_{i-1}) is empty, p_{k+1} does not lie in triangle (p_i, y, p_{i-1}). Similarly, p_k does not lie in triangle (p_i, y, p_{i+1}). But then there is no place for segment p_kp_{k+1}.

Case 2: *otherwise.* Since $z=p_{i-1}$, p_k does not lie between rays p_iy and p_ip_{i-1}. Similarly, p_{k+1} does not lie between rays p_iy and p_ip_{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_kp_{k+1}. □

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. □

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 $(p_i, p_j)$. Re-label 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 $0 \leqslant j \leqslant i-2$ then $GSP' = (p_j, p_{j+1}, \ldots, p_i)$. Otherwise, $i+2 \leqslant j \leqslant k-1$ and $GSP' = (p_i, p_{i+1}, \ldots, p_j)$. In either case, *GSP'* contains no more than $\lfloor k/2 \rfloor + 1$ vertices. □

4. Conclusion

Besides the theoretical motivation of establishing that a simple linear time algorithm exists not only for finding a diagonal of a polygon but also an ear, there exists a practical application of our result. Meisters' [2] *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 if 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. While this greatly improves Meisters' algorithm we do not advocate its use in practice for which the two algorithms proposed in [5] are much preferred.

References

[1] Chazelle, B. (1990). Triangulating a simple polygon in linear time. *Proc. 31st Annual Symposium on Foundations of Computer Science.* IEEE Press, New York, 220–230.

[2] Meisters, G.H. (1975). Polygons have ears. *Amer. Math. Monthly,* June/July 1975, 648–651.

[3] Levy, L.S. (1970). *Geometry: Modern Mathematics via the Euclidean Plane.* Prindle, Weber & Schmidt, Boston, MA.

[4] Ho, W.C. (1975). Decomposition of a polygon into triangles. *Math. Gaz.* 59, 132–134.

[5] Toussaint, G.T. (1991). Efficient triangulation of simple polygons. *Visual Computer* 7, 280–295.

[6] Tarjan, R.E. and Van Wyk, C.J. (1988). An $O(n \log\log n)$-time algorithm for triangulating simple polygons. *SIAM J. Comput.,* 1988.