Pattern Recognition Letters 2 (1984) 155–158
North-Holland

March 1984

A new linear algorithm for triangulating monotone polygons*

Godfried T. TOUSSAINT
School of Computer Science, McGill University, Montreal, Quebec H3A 2K6, Canada

Received 23 August 1983

Abstract: Let $P=(p_1, p_2, \ldots, p_n)$ be a monotone polygon whose vertices are specified in terms of cartesian coordinates in order. A new simple two-step procedure is presented for triangulating P, without the addition of new vertices, in $O(n)$ time. Unlike the previous algorithm no specialized code is needed since the new approach uses well-known existing algorithms for first decomposing P into edge-visible polygons and subsequently triangulating these.

Key words: Algorithms, complexity, computational geometry, triangulation, simple polygons, monotone polygons, edge-visible polygons, visibility, computer graphics, polygon decomposition, image processing, pattern recognition.

1. Introduction

Let $P=(p_1, p_2, \ldots, p_n)$ be a *simple* polygon whose vertices are specified in terms of cartesian coordinates in clockwise order. We assume that the polygon is in *standard* form, i.e., the vertices are distinct and no three consecutive vertices are collinear [1]. A pair of vertices p_ip_{i+1} defines an edge of the polygon where $i=1,2,\ldots,n$ and $p_{n+1}=p_1$. A *simple* polygon is *monotone* if there exist two extreme vertices in a preferred direction (such as $p_{y\max}, p_{y\min}$ if the y direction is preferred) such that they are connected by two polygonal chains monotonic in this direction. In this note we assume we are given a monotone polygon along with a pair of extreme vertices p_i, p_j in a direction of monotonicity. In any case one can determine if a simple polygon is monotone and compute *all* directions of monotonicity in $O(n)$ time with the algorithm of Preparata and Supowit [2]. Without loss of generality we further assume that P is monotonic in the y direction and that we thus know $p_{y\max}$ and $p_{y\min}$. Monotone polygons appear in a variety of applications areas [3]. Garey et al. [4] obtained an $O(n \log n)$ algorithm for triangulating an arbitrary simple polygon Q by first obtaining a planar subdivision of Q and the convex hull of Q into monotone polygons and subsequently triangulating each monotone polygon in linear time. This was the first instance where a non-trivial resticted class of simple polygons admitted triangulation in linear time. Another restricted class of polygons with wide applications is considered in [3] and [5]. These are *edge-visible* polygons. Two points in P are said to be *visible* if the line segment joining them lies inside P. P is said to be *weakly visible* from an edge uv if, for each $z \in P$, there exists a $w \in uv$ (depending on z) such that z and w are visible. A polygon P is *edge-visible* if there exists at least one edge (edge of visibility) of P from which P is *weakly visible*. Note that neither *monotone* nor *edge-visible* polygons are a sub-class of each other. In [5] an $O(n^2)$ algorithm is given for determining whether a polygon is edge-visible. On the other hand computing the region of P visible from an edge can be done on $O(n \log n)$ time [6]. However, in [3] it is shown that if P is *edge-visible* and we are *also* given an *edge-of-visibility* then P can be tri-

*Research supported by N.S.E.R.C. Grant No. A9293.

0167-8655/84/$3.00 © 1984, Elsevier Science Publishers B.V. (North-Holland)

angulated in $O(n)$ time with a very simple well-known convex hull algorithm due to Sklansky (also referred to as the Graham scan [1]). This algorithm, like that of Garey et al. [4], involves backtracking but is simpler than the algorithm in [4]. In this note we show that it is a trivial matter to decompose a *monotone* polygon into *edge-visible* polygons, along with their edges-of-visibility, in $O(n)$ time and thus we can subsequently triangulate the edge-visible polygons in $O(n)$ time with the algorithm of [3]. While the new approach may not be faster in practice than that of [4], it is conceptually simpler, builds on existing algorithms rather than requiring new specialized code, and affords an easy proof of correctness. It is also shown that fundamentally the two algorithms are structurally equivalent although they usually result in different triangulations. The new algorithm is in fact a new way of looking at the algorithm of Garey et al. [4] and hence the proof of correctness given here is also an alternate proof for the algorithm of [4].

2. The new algorithm

We assume the reader is familiar with [3] and the algorithm for triangulating edge-visible polygons when an edge-of-visibility is given. Thus we concentrate on the decomposition step. Since P is monotone in the y direction, $p_{y\max}$ and $p_{y\min}$ split

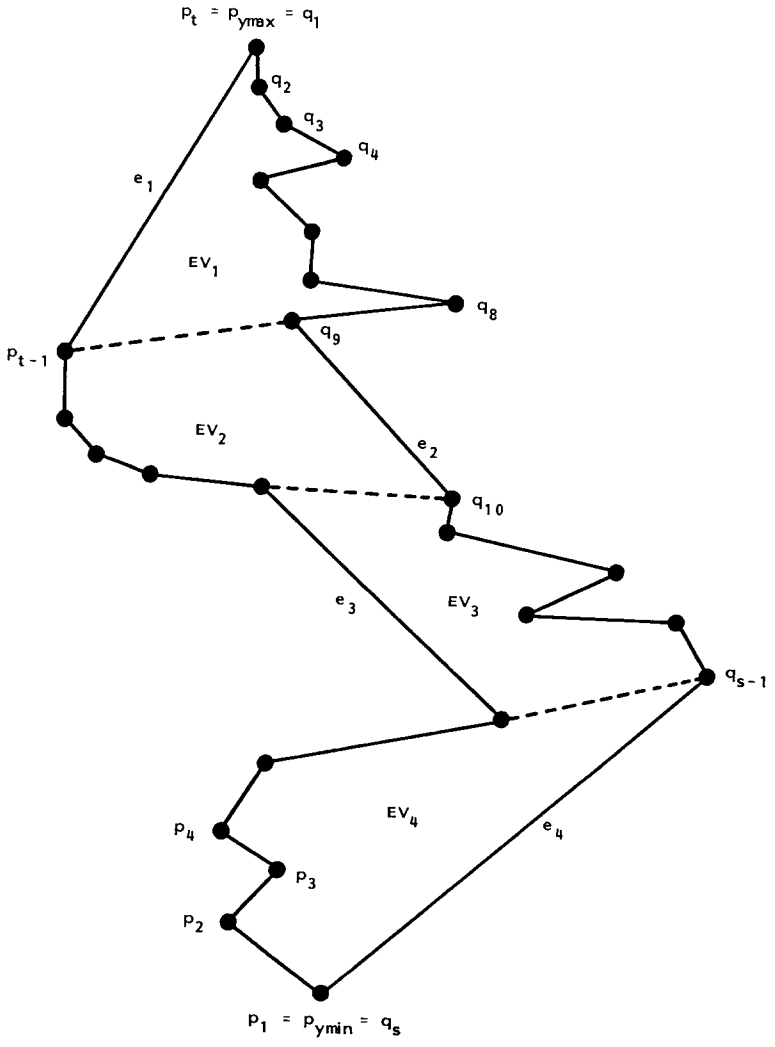

Fig. 1.

the boundary of P into two monotone chains

$$C_1 = (p_{y\min}, \ldots, p_{y\max})$$

and

$$C_2 = (p_{y\max}, \ldots, p_{y\min}).$$

Let C_1 be denoted by $C_1 = (p_1, p_2, \ldots, p_t)$ and let $C_2 = (q_1, q_2, \ldots, q_s)$. Thus

$$p_1 = q_s = p_{y\min} \quad \text{and} \quad p_t = q_1 = p_{y\max}$$

(see Figure 1). To simplify the description of the algorithm we further assume that no two vertices have the same y coordinates.

Procedure DECOMPOSE

Step 1: Merge the two sorted lists $(q_1, q_2, \ldots, q_s)$ and $(p_t, p_{t-1}, \ldots, p_1)$ to obtain a new sorted list C_m.

Step 2: Scan C_m and whenever a p (or q) vertex follows a q (or p) vertex add a diagonal between them.

END DECOMPOSE

This procedure is illustrated in Figure 1 where P is decomposed into the four edge-visible polygons EV_i, $i = 1, \ldots, 4$. Note that the highest and lowest vertices of each EV polygon define an edge e_i, $i = 1, \ldots, 4$. These are precisely the edges-of-visibility of each EV polygon as will be shown below. We now prove the main result of this paper. Let $L(x, y)$ denote the directed line through x and y in direction xy. Also let $\mathrm{RH}(x, y)$ denote the closed half-plane to the right of $L(x, y)$.

Theorem. *Procedure* DECOMPOSE *correctly decomposes a monotone polygon P into edge-visible polygons in* $O(n)$ *time.*

Proof. Consider first the complexity. Step 1 can be done in $O(n)$ time with any standard linear algorithm for merging two sorted lists as in Knuth [7]. In step 2, at each step of the scan, we can test whether a diagonal must be added, and do so if necessary, in constant time. Therefore, DECOMPOSE takes $O(n)$ time to execute. Thus we turn to the question of correctness. First we prove that the algorithm produces a valid planar decomposition

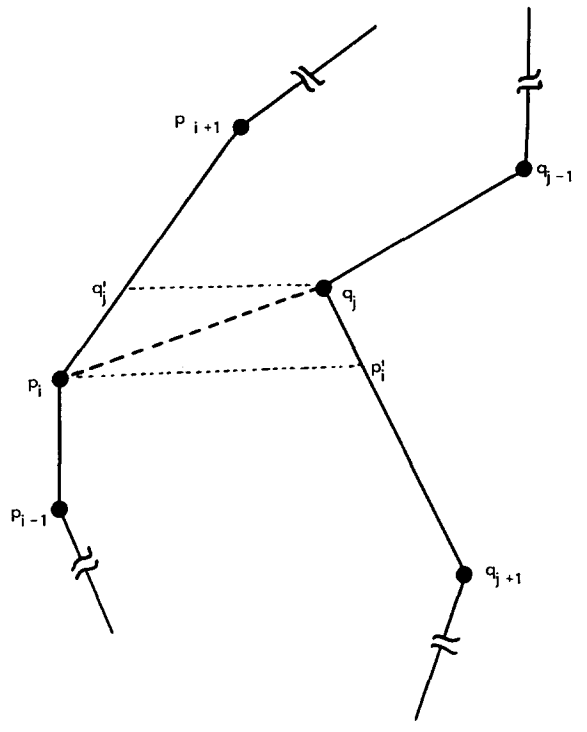

Fig. 2.

into sub-polygons, i.e., we must show that each diagonal added lies completely in P and does not intersect any other diagonal. Assume that diagonal $q_j p_i$ has been added and refer to Figure 2. Since q_j and p_i are adjacent vertices in the sorted list C_m it follows that no other vertices may lie in the strip defined by the horizontal lines passing through p_i and q_j. Therefore the horizontal lines through p_i and q_j must intersect edges $p_i p_{i+1}$ and $q_j q_{j+1}$, say, at q_j' and p_i' respectively. Furthermore, due to the *simplicity* (Jordan's Curve Theorem) of P it follows that $q_j p_i p_{i+1}$ and $p_i q_j q_{j+1}$ are both right turns. Therefore the quadrilateral $Q(p_i, p_i', q_j, q_j')$ is convex and lies in P. Since Q is convex, the diagonal $p_i q_j$ must lie in Q and since $Q \in P$ so must $p_i q_j \in P$. Consider now the question of planarity and assume $p_i q_j$ intersects another diagonal $p_k q_m$. From the above observations it follows that either

$$p_k \in \mathrm{RH}(q_j, q_j') \quad \text{and} \quad q_m \in \mathrm{RH}(p_i, p_i')$$

or

$$q_m \in \mathrm{RH}(q_j, q_j') \quad \text{and} \quad p_k \in \mathrm{RH}(p_i, p_i').$$

In either case p_k and q_m are not adjacent in C_m. However, if $p_k q_m$ is an added diagonal, then by

step 2 of the algorithm they must be adjacent in C_m. This is a contradiction. Therefore the decomposition is planar. It remains to show that each subpolygon is edge-visible and points to its edge of visibility. Consider one of the resulting subpolygons, say EV_i. From the scan of C_m in step 2 it follows that the 'birth' of EV_i occurs during a jump from a p (or q) to a q (or p) vertex and the 'completion' of EV_i happens when we return to a p (or q) vertex. Therefore the vertices of EV_i with maximum and minimum y coordinates, say ev_{ymax} and ev_{ymin}, define an edge of EV_i. Since, by construction, EV_i is monotone in the y direction it follows that for every point $u \in EV_i$ there exists a point $w \in ev_{ymax}ev_{ymin}$ (with the same y coordinate as u) such that $uw \in EV_i$. Therefore EV_i is edge visible from $ev_{ymax}ev_{ymin}$ and its edge of visibility is known. □

3. Concluding remarks

It was mentioned in the introduction that the algorithm presented here is structurally equivalent to the algorithm of Garey et al. [4]. We now expand on this remark. Referring to Figure 1, the algorithm in [4] starting at p_{ymax} applies the Graham backtracking scan to compute the 'inner covex hull' of C_2 until it reaches q_i where q_{i+1} lies below p_{t-1}. It then joins p_{t-1} to all the 'convex hull' vertices since it is shown in [4] that this region is star-shaped with respect to p_{t-1}. The scan then jumps to C_1 and alternates back and forth as it proceeds downwards. Although the two triangulations resulting from the two algorithms need not be the same, the switching rule used in [4] to jump from C_1 to C_2 and vice versa causes diagonals to be added precisely where DECOMPOSE does also. Thus the algorithm in [4] *implicitly* decomposes P into edge-visible polygons as it goes along. The two triangulations may differ in the interior of the edge-visible polygons. Thus in [4] the Graham scan is used implicitly to triangulate edge-visible polygons and the results presented here provide an alternate proof of correctness of the algorithm of Garey et al. [4].

Decompositions of polygons into edge-visible polygons are of interest in their own right for shape analysis [8] although here we would like minimal or near-minimal decompositions in the sense of minimizing the number of sub-polygons obtained. In this regard the algorithm presented here can do poorly. If C_1 and C_2 interleave we obtain $n-2$ edge-visible polygons (i.e. triangles). In Figure 1 we have four but this is double that of the minimal decomposition since only e_1 and e_3 are necessary as e_3 can see $EV_2 \cup EV_3 \cup EV_4$. An interesting open problem is to decompose simple polygons into the minimum number of edge-visible polygons. For *convex, star-shaped,* and *spiral* components this problem was recently solved by Keil [9]. For additional results along these lines see also [10].

References

[1] Shamos, M.I. (1978). Computational geometry. Ph.D. thesis, Yale University.

[2] Preparata, F.P. and K.J. Supowit (1981). Testing a simple polygon for monotonicity. *Information Processing Letters* 12, 161–164.

[3] Toussaint, G.T. and D. Avis (1982). On a convex hull algorithm for polygons and its application to triangulation problems. *Pattern Recognition* 15, 23–29.

[4] Garey, M.R., D.S. Johnson, F.P. Preparata, and R.E. Tarjan (1978). Triangulating a simple polygon. *Information Processing Letters* 7, 175–179.

[5] Avis, D. and G.T. Toussaint (1981). An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Transactions on Computers* 30, 910–914.

[6] El-Gindy, H. (1983). Personal communication.

[7] Knuth, D.E. (1973). *Sorting and Searching.* Addison Wesley, New York.

[8] Feng, H. and T. Pavlidis (1975). Decomposition of polygons into simpler components: feature generation for syntactical pattern recognition. *IEEE Transactions on Computers* 24, 636–650.

[9] Keil, M. (1983). Decomposition of polygons into simpler components. Ph.D. thesis, University of Toronto.

[10] Kouta, M.M. and J. O'Rourke (1982). Fast algorithms for polygon decomposition. Tech. Report JHU 82-10, Johns Hopkins University.