TRIANGULATING A STAR-SHAPED POLYGON

A. A. Schoone and J. van Leeuwen

RUU-CS-80-3

April 1980

Rijksuniversiteit Utrecht

Vakgroep informatica

Princetonplein 5
Postbus 80.002
3508 TA Utrecht
Telefoon 030–531454
The Netherlands

TRIANGULATING A STAR-SHAPED POLYGON

A. A. Schoone and J. van Leeuwen

Technical Report RUU-CS-80-3

April 1980

Department of Computer Science

University of Utrecht

P.O. Box 80.002

3508 TA Utrecht, the Netherlands

all correspondence to:

Dr. Jan van Leeuwen
Department of Computer Science
University of Utrecht
P.O. Box 80.002
3508 TA Utrecht
the Netherlands

TRIANGULATING A STAR-SHAPED POLYGON

Anneke A. Schoone & Jan van Leeuwen

Department of Computer Science, University of Utrecht
P.O. Box 80.002, 3508 TA Utrecht, the Netherlands

Abstract. We present two algorithms for triangulating star-shaped n-gons in the plane in O(n) steps. We characterize a more general class of simple n-gons which can be triangulated in linear time as well.

1. Introduction

A triangulation of a simple n-gon P is a way of drawing nonintersecting line-segments between vertices of P through the interior, such that the interior of P is subdivided entirely into triangles. It is an instructive exercise to prove that all simple n-gons can be triangulated. While a simple n-gon may very well admit more than one triangulation, each triangulation will consist of n-3 line-segments ("diagonals") and give rise to n-2 triangles.

Garey et.al. [1] have shown that every simple n-gon can be triangulated in O(n log n) steps. In an important step of the proof they shown that "monotone" simple n-gons can be triangulated in O(n) steps. We shall not define the class of monotone simple polygons formally, but note that it properly includes e.g. the class of convex polygons. In this paper we consider the question what other classes of simple polygons admit a linear time triangulation and prove that the familiar star-shaped polygons form such a class. (We leave it to the reader to verify that star-shaped polygons are not necessarily monotone in the sense of [1].)

A simple polygon P is called star-shaped when there exists a point t in the enterior from which all vertices of P are visible. Given an enumeration of the vertices of a simple n-gon in the order in which they appear on the boundary, it takes only O(n) steps to determine whether the polygon is star-shaped and, if so, to find a point t as described (cf. Lee and Preparata [3]). Hence the star-shaped polygons form an "easily recognized" subclass of the simple polygons.

In section 2 we shall introduce the concept of a "reducible segment" and use it to obtain some general algorithmic tools for later constructions. In section 3 we present two different methods for triangulating star-shaped polygons in linear time. One of the techniques used can be generalized and allows us to characterize the larger class of "reducible polygons" for which a linear time triangulation algorithm exists as well.

2. Preliminaries: reducible segments

A sector of the plane consists of a point t (the tip), two rays emanating from t and the area "between" the two rays (the interior). We shall always assume that the angle α between the two rays is $\leq 180^{\circ}$, thus restricting ourselves to convex sectors. A simple n-chain is a sequence of n vertices $p_1, \ldots, p_n$ with line-segments connecting p_i to p_{i+1} $(1 \leq i < n)$ and infinite rays emanating from p_1 and p_n without any intersections occuring. An n-segment (see figure 1) consists of (i) a sector with some tip t and (ii) a simple n-chain $p_1, \ldots, p_n$ in the interior, with p_1 and p_n on separate edges

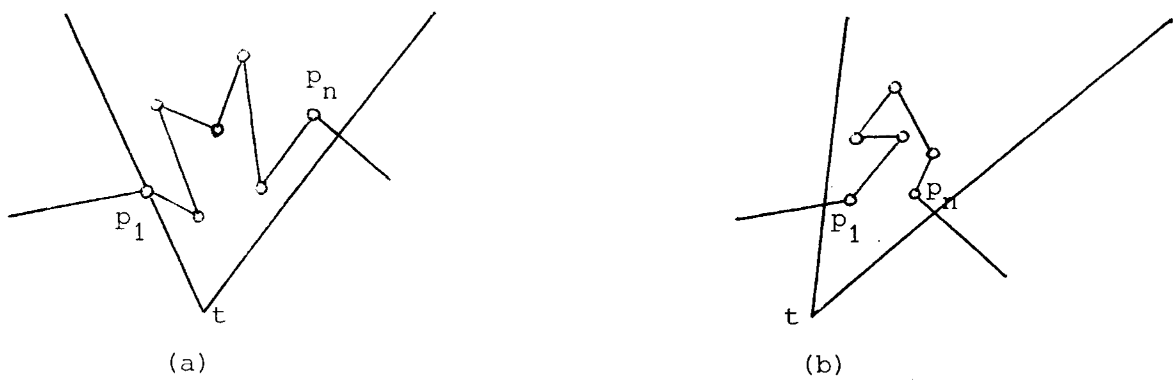


figure 1: examples of segments

of the sector or their "rays" intersecting separate edges. We allow that p_1 and/or p_n coincide with t as degenerate cases. We shall always assume that the sector containing a segment is implicitly understood, unless stated otherwise.

<u>Definition</u>. An n-segment $p_1, \ldots, p_n$ is said to be reducible when for each triple p_{i-1}, p_i, p_{i+1} for which the interior angle between the connecting line-segments is $< 180^{\circ}$ $(1 < i < n)$ the diagonal $\overline{p_{i-1}\, p_{i+1}}$ can be drawn without creating intersections, while the resulting (n-1)-segment $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$ remains reducible within the same sector.

The definition of reducibility is fairly straightforward despite its "recursive" form and will be appreciated once it is read as saying that one can "complete" any interior triangle formed by three consecutive vertices along the boundary after the triangle has been cut off. Note that the n-segment of figure 1.a. is reducible and the n-segment of figure 1.b. is not.

Given a n-segment $p_1, \ldots, p_n$ the lower convex hull is defined as the shortest non-intersecting arc C from p_1 to p_n such that, when extended with

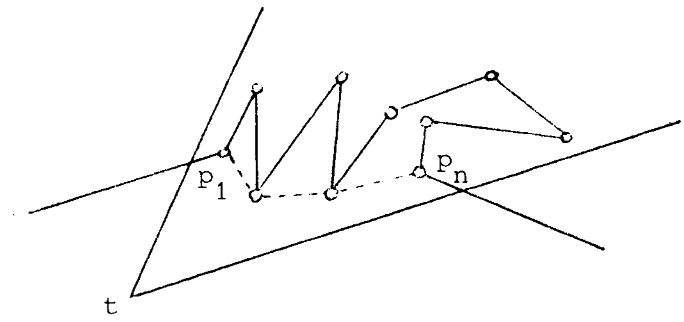


figure 2: a lower convex hull

the rays from p_1 and p_n across the sector boundaries, the entire set of points is on or on the other side of C when viewed from t. A closely related notion appears in Shamos [4] in connection with a geometric formulation of isotonic regression. Viewed from t, the lower convex hull is a concave arc connecting p_1 to p_n through selected vertices of the set.

We shall assume that n-segments are always given with the points $p_1, \ldots, p_n$ stored in a doubly-linked list. Hence we can navigate only with the operations NEXT and PREV (of obvious meaning) and have no random access in the list.

<u>Lemma</u> 1. Given a reducible n-segment $p_1, \ldots, p_n$ its lower convex hull C can be constructed in O(n) steps, while the area between the original segment and C gets triangulated at the same time.

<u>Proof</u>

The algorithm resembles one used by Graham [2]. Given the reducible segment, have a cursor visit $p_3, p_4, \ldots$ in this order and maintain a stack containing the lower convex hull of the points visited. Let the stack contain $c_1 = p_1, c_2, \ldots, c_j = p_{i-1}$ as the cursor advances to p_i. We proceed by

```
while i ≤ n do
begin
   if j = 1 or angle (c_{j-1} c_j p_i) ≥ 180° then
         begin c_{j+1} := p_i; advance cursor end
   else
         begin output triangle c_{j-1} c_j p_i; pop c_j end
end
```

When angle $(c_{j-1}c_jp_i) < 180^\circ$, the condition of reducibility guarantees that the triangle can be drawn in the interior without intersecting previous triangles or edges. The cursor is not advanced until the stack is popped down to c_1 or to some c_k such that angle $(c_{k-1}c_kp_i) \geq 180^\circ$. As figure 3 shows, another triangle is drawn in each time the stack is popped. When the

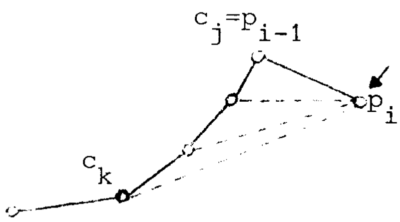


figure 3

cursor finally advances, the stack has been updated correctly to contain the lower convex hull of p_1 to p_i.

Note that we can charge the costs of a step either to the cursor (when it advances) or to the point popped off the stack. It follows that the algorithm takes time proportional to n + #(points not on C), which is O(n).

□

Two neighboring segments $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ with the same tip t are said to be adjacent when either $p_n = q_1$ or the rays emanating from p_n and q_1 coincide, effectively creating an edge joining p_n and q_1 without intersecting any other edges (figure 4) (or when, of course, $p_1 = q_m$ or the rays emanating from p_1 and q_m coincide to form an non-intersecting edge).

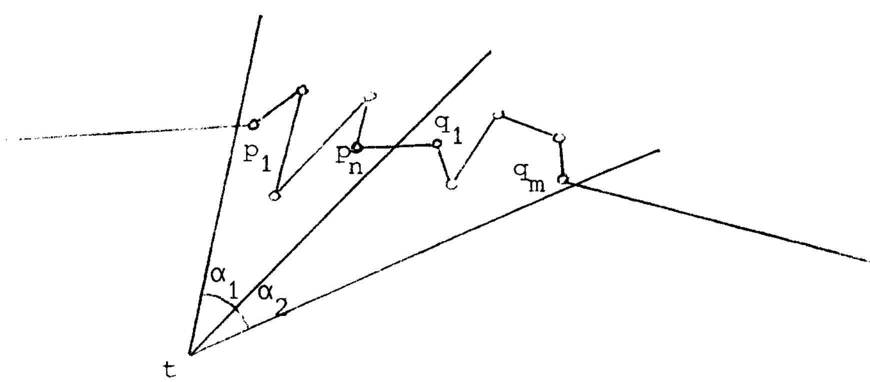


figure 4

Let the sector angles be α_1 and α_2 respectively.

Lemma 2. Let $\alpha_1 + \alpha_2 \leq 180^\circ$. Assuming the lower convex hulls C_1 and C_2 of the adjacent segments $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ have been formed, the lower convex hull C of the full segment $p_1, \ldots, p_n, q_1, \ldots, q_m$ can be constructed in only O(# points of C_1 and C_2 not on C) additional steps, while the area between C_1, C_2 and C gets triangulated at the same time.

Proof

Let C_1 and C_2 be available as stacks $c_1, \ldots, c_k$ and $d_1, \ldots, d_l$. Beginning with j=k and i=1, let there be a cursor ↑1 on c_j and a cursor ↑2 on d_i. Unless angle $(c_{j-1}c_jd_i) \geq 180^\circ$ (or j=1) ánd angle $(c_jd_id_{i+1}) \geq 180^\circ$ (or i=1), it is always possible to draw either triangle $c_{j-1}c_jd_i$ or triangle $c_jd_id_{i+1}$ without creating intersections, as an analysis of all conceivable local situations

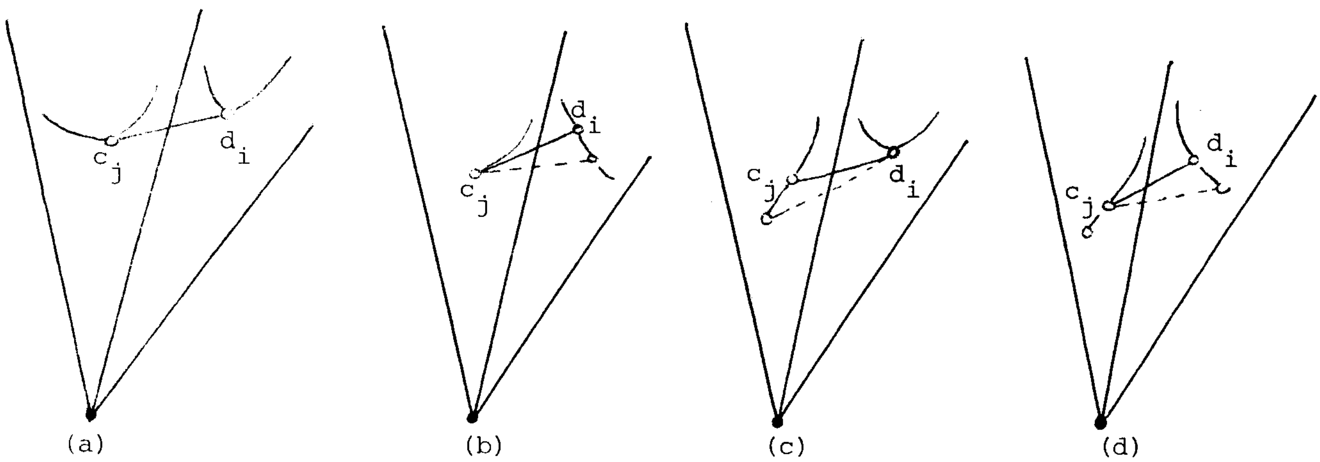


figure 5

(figure 5 a-d) shows. After deciding which one to take, either draw triangle $c_{j-1}c_jd_i$ and move ↑1 backward or draw triangle $c_jd_id_{i+1}$ and move ↑2 forward. Slowly the edge $\overline{c_jd_i}$ will converge to the tangent of C_1 and C_2. When the algorithm stops, we only need to concatenate $c_1, \ldots, c_j$ and $d_i, \ldots, d_l$ as lists to obtain a consistent representation of C.

The run-time of the algorithm clearly is proportional to the number of times a cursor moved, i.e., to the number of points removed from C_1 and C_2 to obtain C.

□

Note that the lemma is valid even when $p_1=t=q_m$. The notion of a lower convex hull is voided in this case, but the algorithm yet triangulates the simple region enclosed by the arc C_1 from t to p_n, the edge $\overline{p_nq_1}$ and the arc C_2 from q_1 back to t correctly in linear time.

In the situation of lemma 2 the points of C_1 and C_2, when viewed in the sector with tip t and angle $\alpha_1 + \alpha_2$, need not form a reducible segment. Using the lower convexity of C_1 and C_2 enabled us to save time in computing the combined lower convex hull. It carries through when we add another adjacent segment to it and so on (figure 6).

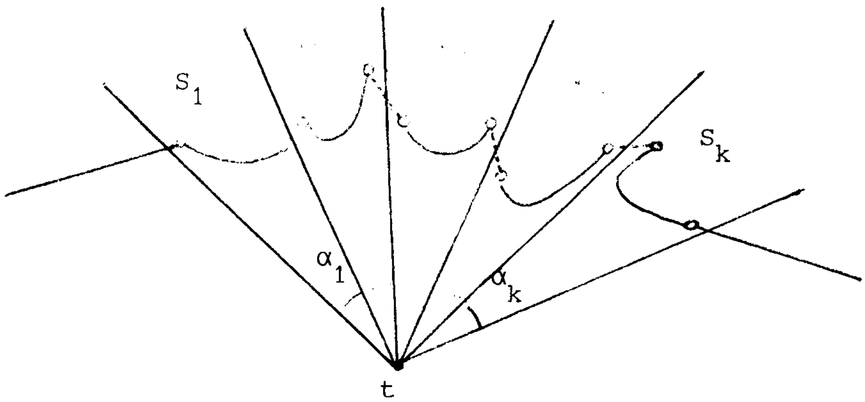


figure 6

<u>Theorem</u> A. Let S_1 to S_k be reducible segments in neighboring sectors with the same tip t and angles α_1 to α_k, such that S_i is adjacent to S_{i+1} $(1 \le i < k)$ and $\alpha_1 + \ldots + \alpha_k \le 180^o$. The lower convex hull C of $S_1 \cup .. \cup S_k$ can be constructed in linear time, while the area between C and S_1 to S_k (with the joining edges) gets triangulated at the same time.

Proof

First compute the lower convex hulls (as lists of points) of each of the segments separately. Next combine the segments one after another, using the technique of lemma 2. After the combined lower convex hull of sectors S_1 to S_i has been computed, the addition of S_{i+1} to it takes a number of steps proportional to the number of points that now get eliminated from the contour. It follows that the total run-time remains linear. The desired triangulation is obtained at no extra charge, as the algorithm proceeds.

□

Again, theorem A remains valid even when the "first" point of S_1 and/or the "last" point of S_k coincide with t.

We need some more tools, to cover other ways of combining (reducible) segments. Two neighboring segments $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ with the same tip t and $\alpha_1 = \alpha_2 = 180^o$ are said to be adjacent (figure 7) when either $p_n = q_1$ or the rays emanating from p_n and q_1 coincide and, in addition, either $q_m = p_1$ or the rays emanating from them coincide (effectively creating

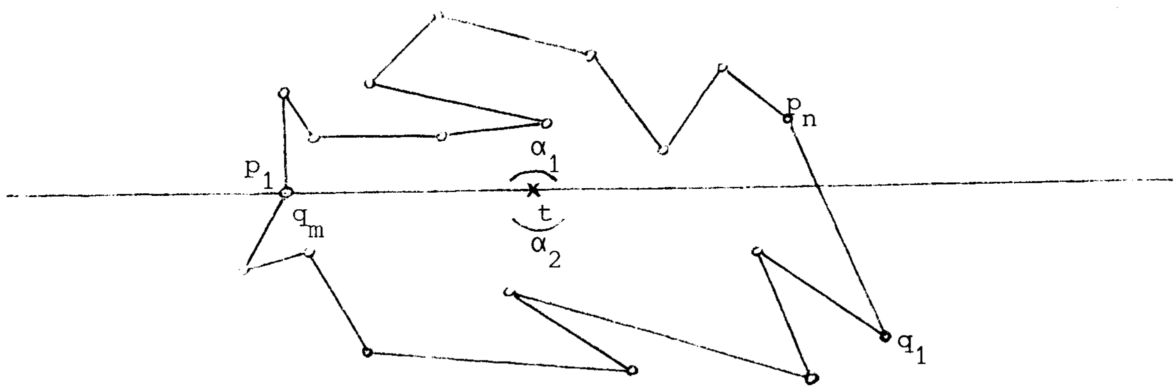


figure 7

an edge joining q_m and p_1 without intersections).

Lemma 3. Let $\alpha_1 = \alpha_2 = 180^o$. Assuming the lower convex hulls C_1 and C_2 of the adjacent segments $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ have been formed, the area enclosed by C_1, C_2 and the edges $\overline{p_n q_1}$ and $\overline{q_m p_1}$ can be triangulated in linear time.

Proof

Let C_1 and C_2 be available as stacks $c_1, \ldots, c_k$ and $d_1, \ldots, d_l$ respectively. Essentially the same algorithm as in lemma 2 applies, using cursors $\uparrow 1$ (starting at $c_k = p_n$) and $\uparrow 2$ (starting at $d_1 = q_1$). If $\uparrow 1$ is at c_i and $\uparrow 2$ is at d_j, then the crucial observation is again that by the concavity of the two contours one can always draw the triangle $c_{i-1}c_i d_j$ and move $\uparrow 1$ backward or draw the triangle $c_i d_j d_{j+1}$ and move $\uparrow 2$ forward (and it takes only O(1) time to decide which one creates no intersections). The algorithm keeps triangulating from right to left until the cursors eventually hit the end of their lists ($c_1 = p_1$ for $\uparrow 1$ and $d_l = q_m$ for $\uparrow 2$). The run-time clearly is proportional to the combined size of C_1 and C_2.

□

Next consider three segments S_1, S_2 and S_3 in neighboring segments with the same tip t and angles α_1, α_2 and α_3 with $\alpha_1 + \alpha_2 + \alpha_3 = 360^o$. Assume the segments are adjacent in this order, in the sense that S_3 is again adjacent

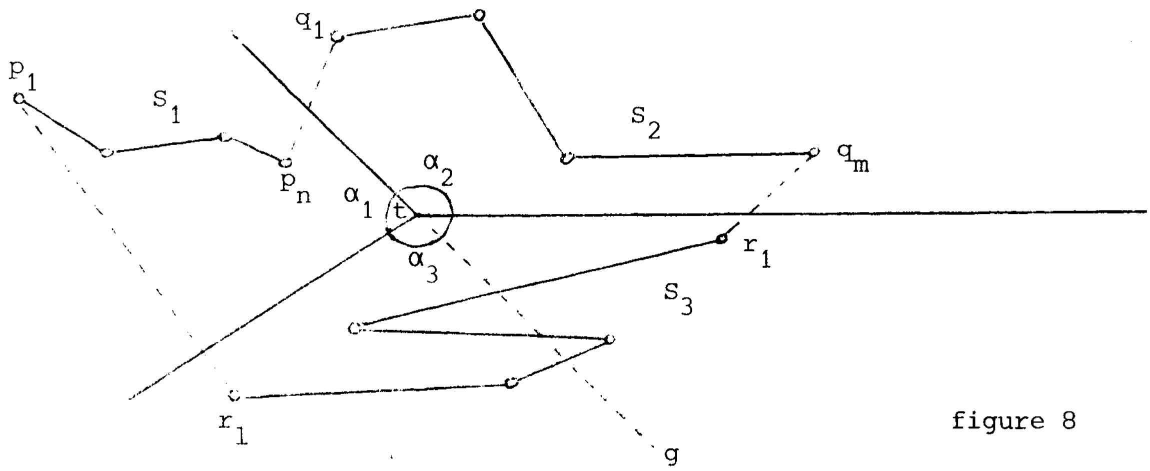


figure 8

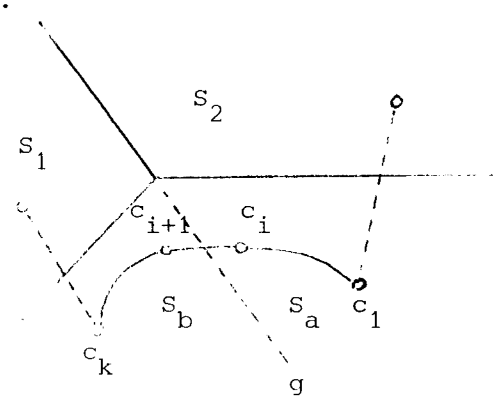
to S_1 (figure 8).

Theorem B. Let S_1 to S_3 be segments in neighboring sectors with the same tip t and angles α_1 to α_3, such that S_i is adjacent to $S_{i \pmod 3)+1}$ $(1 \le i \le 3)$ and $\alpha_1 + \alpha_2 + \alpha_3 = 360^\circ$. After the lower convex hulls C_1, C_2 and C_3 of the segments have been formed, it takes only a linear number of additional steps to triangulate the area enclosed by C_1, C_2 and C_3 and the joining edges ($\overline{p_n q_1}$, $\overline{q_m r_1}$ and $\overline{r_1 p_1}$ in figure 8)

Proof

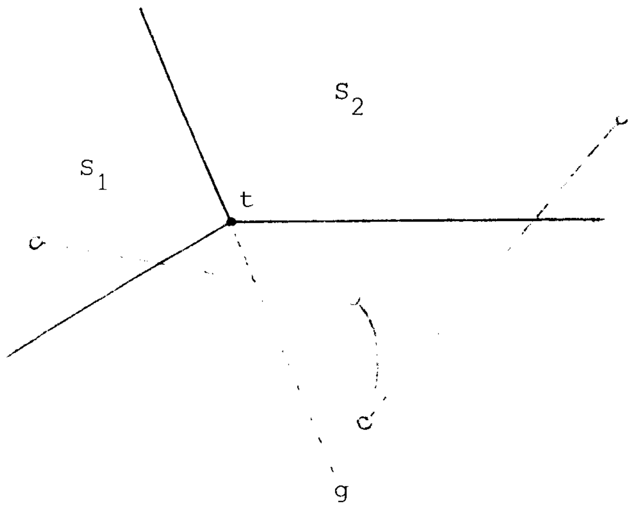
Let g (see figure 8) be the extension of the half-line separating S_1 and S_2. Without loss of generality we may assume that $\alpha_1 < 180^\circ$ and $\alpha_2 < 180^\circ$, so g cuts through S_3 in a nontrivial manner. Let the lower convex convex hull C_3 of S_3 be available as a list $c_1, \ldots, c_k$. The algorithm is simple, but depends on the way g intersects C_3. The following cases can occur (we leave it to the reader to verify that the intersection of g and C_3 can be computed in O(k) steps and that the cases are likewise easily distinguished):

Case I.



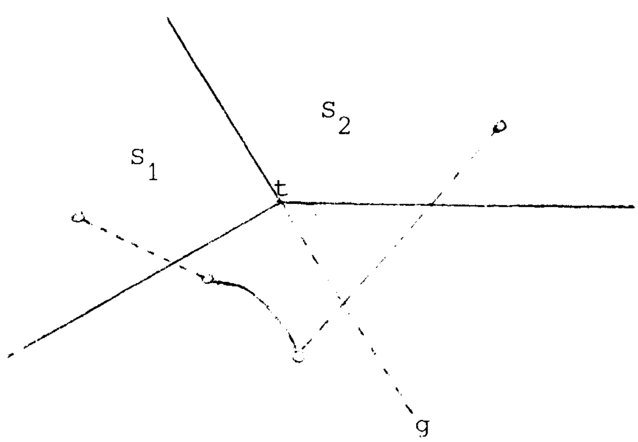
Let S_a be the segment c_1 to c_i and S_b be the segment c_{i+1} to c_k. By lemma 2 the lower convex hull of the adjacent segments C_1 and S_b can be computed in linear time, as can the lower convex hull of the adjacent segments S_a and C_2. Lemma 3 enables us to finish the triangulation of the enclosed region in another linear number of steps.

Case II.



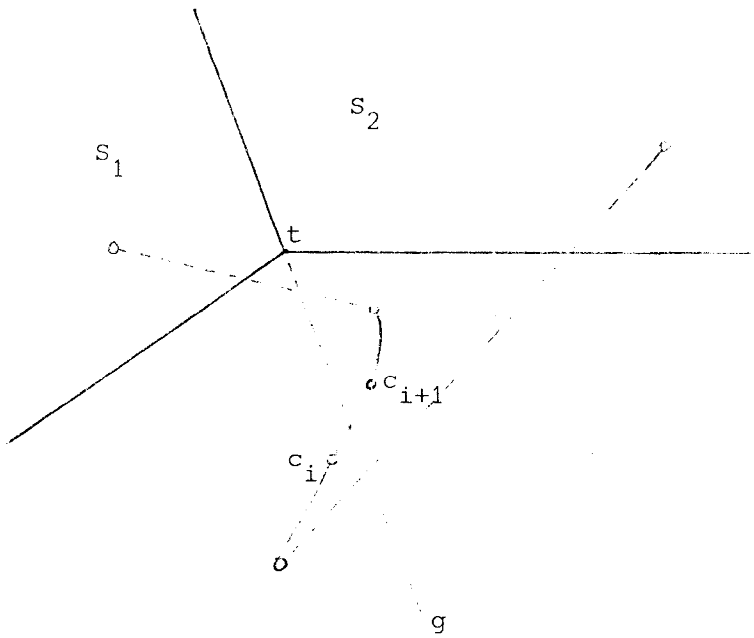
C_3 is entirely to the "S_2-side" of g, hence lemma 2 can be used to compute the lower convex hull of the combined segments C_2 and C_3. By lemma 3 another linear number of steps suffices to triangulate the area between the resulting contour and C_1.

Case III.



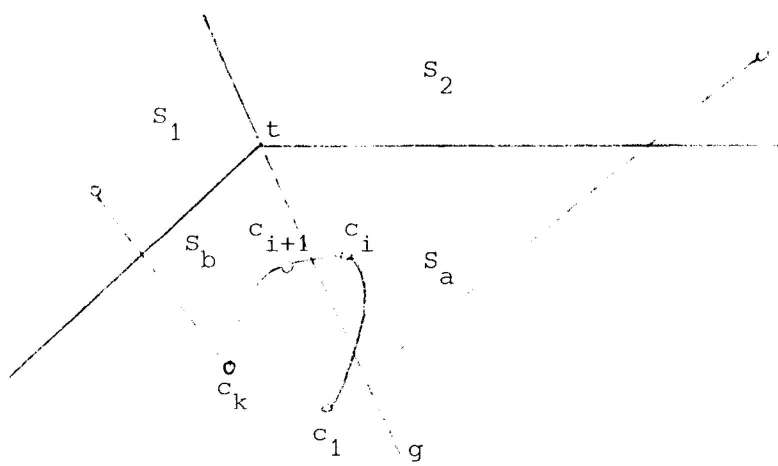
This is analogous to case II and dealt with similarly, by first combining C_1 and C_3.

Case IV.



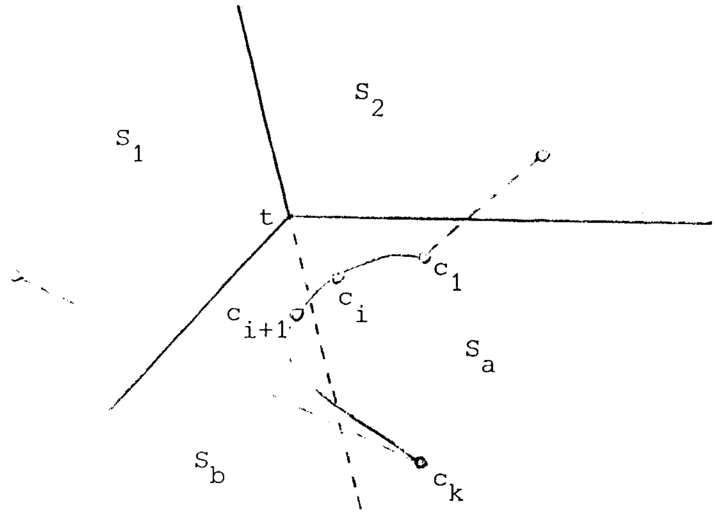
The reader may verify that one can still run the algorithm of lemma 2 on C_2 and C_3 to obtain the combined lower convex hull C' in linear time and triangulate at the same time. Because C' must lie entirely on the "S_2-side" of the separating line, lemma 3 can be used to triangulate the area between C' and C_1 again.

Case V.



Let S_a be the segment from c_1 to c_i and S_b from c_{i+1} to c_k. The algorithm to follow is completely similar to the one for case I.

Case VI.



This is symmetric to case V. If we let S_a be the segment from c_1 to c_i and S_b the usual segment from c_{i+1} to c_k, then the same algorithm will do.

□

3. Triangulating star-shaped and other polygons

Given a simple polygon, its kernel is defined as the set of all points t in the interior from which all vertices of the polygon are visible. A (simple) polygon is called star-shaped whenever it has a non-empty kernel.

Consider a star-shaped polygon with some point t in its kernel. Whenever the angle between three consecutive vertices p' p" and p"' along its contour is $<180^{\circ}$ and t is not contained in the triangle p'p"p"' (see figure 9), one

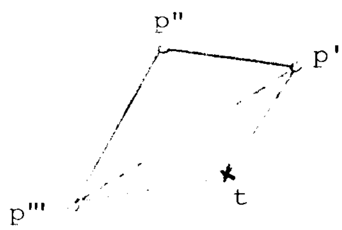


figure 9

can draw the triangle p'p"p"' without intersecting edges of the polygon and "eliminate" p" from the contour, thus reducing the task of triangulating the given polygon to the same task on a polygon with one vertex less. The observation fundamental to various triangulation algorithms is that the resulting smaller polygon still is star-shaped and has the same point t in its interior.

Consider a star-shaped polygon P. Let its vertices be $p_1, \ldots, p_n$ and assume we know the location of some point t in its kernel. (If no such point was given, then an algorithm of Lee and Preparata [3] will find one for us in O(n) time.) We shall present a fairly direct triangulation algorithm for P first.

Theorem C. Star-shaped polygons can be triangulated in linear time.

Proof

We shall present an algorithm that (for reasons of explanation only) consists of four stages.

Stage 1

Maintain a list (a stack) $c_0, c_1, \ldots$ (initially containing p_1 and p_2) and have a cursor visit $p_3, p_4, \ldots$ in this order. The computation we do is very similar to that of lemma 1, but is carried through a little further. Suppose the stack is $c_0, c_1, \ldots, c_j$ as the cursor gets to p_i. Then repeat

if angle $(c_{j-1}c_jp_i) \geq 180^o$ **then**
    begin $c_{j+1} := p_i$; move cursor forward **end**
else if t not within triangle $c_{j-1}c_jp_i$ **then**
    begin output triangle $c_{j-1}c_jp_i$; pop c_j **end**
  else
    stop.

Thus, each time we advance the cursor to a next p_i we draw as many triangles to previous points as possible, until t gets in the way. It should be obvious that this can go on until the cursor moves to the first p_i across the line $\overline{p_1t}$ (see figure 10). After drawing some more triangles perhaps, the algorithm (or rather, this stage of it) will stop with some list $c_0 = p_1, c_1, \ldots, c_j$. Note that it represents the lower convex hull of p_1 to p_{i-1}, with the part

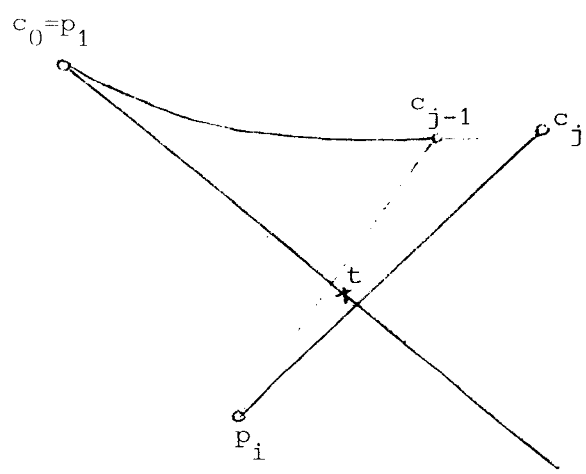


figure 10

beyond the line $\overline{c_j p_i}$ cut off. The area enclosed by the contour from p_1 to p_i and the curve c_0, ..., c_j, p_i will have been triangulated completely at this time.

Stage 2

Continue and carry out the algorithm of stage 1 on the contour c_j, p_i, p_{i+1}, ... The algorithm will compute the lower convex hull (viewed from the line $\overline{c_j t}$) as far as it can while disseminating triangles, until t stands in the way for drawing the next triangle (figure 11). As before, the algorithm will stop just after the cursor moves across the line $\overline{c_j t}$ to p_1.

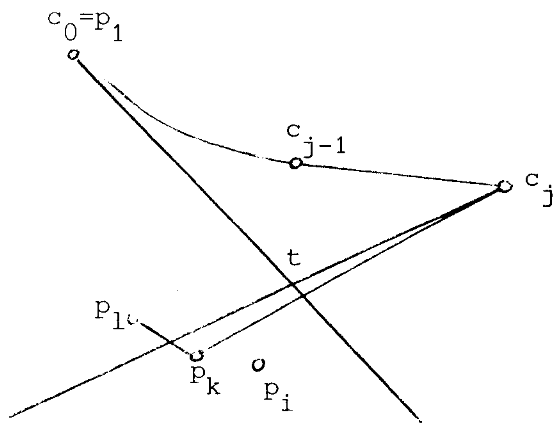


figure 11

After drawing the triangles it could (while popping the stack), the next triangle the algorithm inspects must contain t. Observe that this triangle must have one vertex across the line $\overline{c_0 t}$ from p_1 (see figure 11). There is only one vertex of the contour c_j, p_i, ... this can be, namely c_j. It follows that the stack maintained by the algorithm during stage 2 must have popped all the way to the bottom and contain just c_j and some p_k. The entire contour up to p_1 is "behind" $\overline{c_j p_k p_1}$. (It is possible that $p_1 = p_i (= c_0)$ but the cursor cannot have moved further).

Stage 3

Now carry out the same algorithm again on the contour p_1, ..., p_n, c_0, ..., c_j. Clearly one may omit the test whether a triangle contains t at this stage and the algorithm becomes identical to that of lemma 1. Because p_1, ..., c_j (viewed in the sector with tip t and bounding line $\overline{c_j t}$) is a reducible segment (!), the algorithm even has the same effect and computes the lower convex hull of the contour (see ↘ in figure 12).

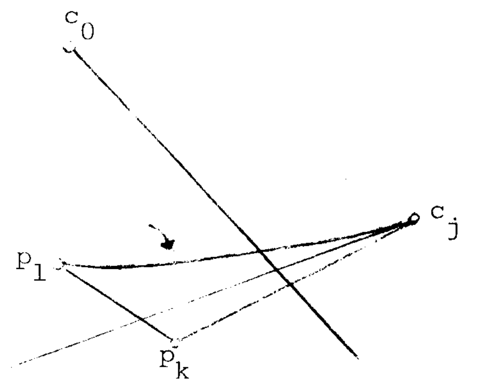


figure 12

Stage 4

Finish the triangulation by drawing edges from p_k to all nodes between p_1 and c_j on the concave contour resulting from the previous stage (see

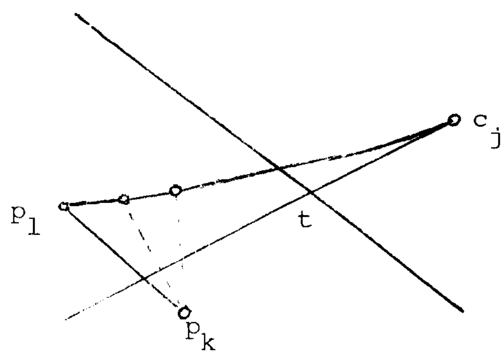


figure 13

figure 13). The edges will not cause intersections, as is easily seen.

It is interesting to observe that the edges drawn in stage 4 will all pass t "to the left". This is so because the line $\overline{p_k c_{j-1}}$ passed t to the left (as $\overline{p_i c_{j-1}}$ did) and all other edges must be further to the left of it. Hence we could have put stages 3 and 4 together and run the algorithm of stage 1 on the contour p_k, p_1, ..., c_j. By the time the algorithm stops, the triangulation must be complete, with t contained in the final triangle.

All steps of the algorithm can be charged to a point, either because the cursor moved by it or because it got eliminated from the contour. As no point will be charged more than a constant, the algorithm's total running time is linear.

□

The (three-stage) "swing-around" algorithm of theorem C can be modified in several ways. For example, instead of fixing a bottom of the stack (such as $c_0=p_1$), one might allow it to grow "backward", as long as triangles with preceding vertices can still be drawn without running into t. Whenever the algorithm gets stuck on such a triangle (or when it cannot draw any other triangle at all), move the cursor forward and continue. Assume the points of the star-shaped polygon are stored in a doubly-linked, circular list L. The algorithm would become

```
p := an arbitrary starting point;
#L := n;
while #L > 3 do
   begin
       a := NEXT(p);
       b := PREV(p);
       if angle (a p b) ≥ 180° then
                   p := a
       else if t not contained in triangle a p b then
                   begin
                     output triangle a p b;
                     PREV(a) := b;
                     NEXT(b) := a;
                     #L := #L-1;
                     p := b
                   end
             else
               p := a
   end;
output triangle NEXT(p), p, PREV(p).
```

There is yet another way to look at the triangulation problem for star-shaped polygons. Draw the line $\overline{p_1 t}$ (see figure 14). As P is star-shaped, the line will intersect P in only one other point, say, on the edge $\overline{p_i p_{i+1}}$.

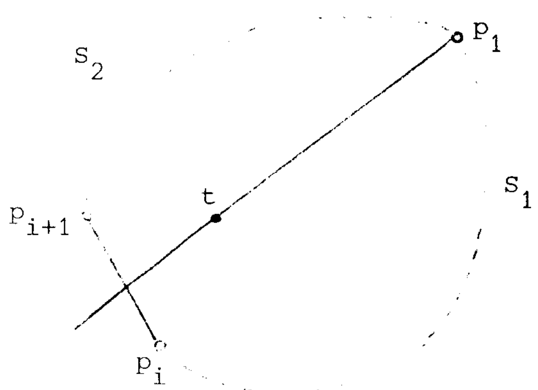


figure 14

(If $\overline{p_1 t}$ intersects the contour exactly in a vertex, then the following discussion carries through as well.) Let S_1 and S_2 be the segments from p_1 to p_i and from p_{i+1} to p_n, respectively. Note that S_1 and S_2 are reducible segments, by the general character of star-shaped polygons! It follows that we could first compute the lower convex hulls of S_1 and S_2 (separately) as in lemma 1 and next apply lemma 3 to complete the triangulation of the enclosed region. Observing that the intersection of $\overline{p_1 t}$ with P can be computed in O(n) steps (just walk around the contour and see when $\overline{p_1 t}$ is crossed), we get yet another linear time algorithm to triangulate P. This can be generalized as follows.

Definition. A (simple) polygon is said to be reducible whenever it is composed of a (finite) number of reducible segments S_1 to S_k in neighboring sectors around the same tip t of angles α_1 to α_k, such that $\alpha_1 + \ldots + \alpha_k = 360^\circ$ and S_i is adjacent to $S_{i (\mathrm{mod}\ k)+1}$ $(1 \le i \le k)$.

Figure 15 shows the idea of reducible polygons. Note that we do not require

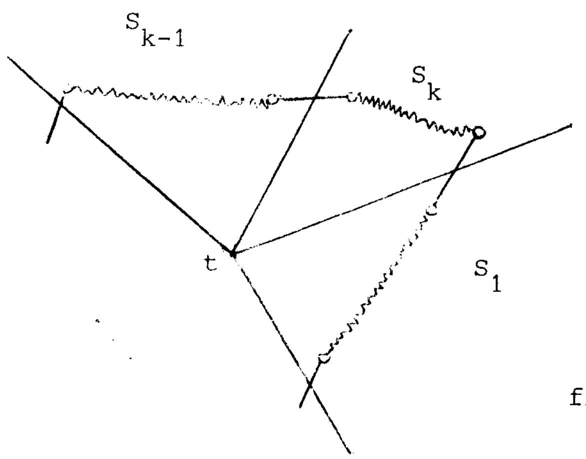


figure 15

k to be bounded by a constant, although it will always be bounded by n. The following result generalizes theorem C.

Theorem D. Reducible polygons can be triangulated in linear time.

Proof

Let a reducible polygon be given in terms of its adjacent, reducible segments S_1 to S_k. If k=2, then $\alpha_1 = \alpha_2 = 180^\circ$ and the theorem follows by applying lemma 1 to S_1 and S_2, and lemma 3 to the result of it. Hence assume that $k \ge 3$. As $\alpha_1 + \ldots + \alpha_k = 360^\circ$ but each individual α is $\le 180^\circ$, there must be two indices $1 \le i < j \le k$ such that

$$\alpha_1 + \ldots + \alpha_i \leq 180^\circ$$
$$\alpha_{i+1} + \ldots + \alpha_j \leq 180^\circ$$
$$\alpha_{j+1} + \ldots + \alpha_k \leq 180^\circ$$

These indices are easily found in O(k) steps, by running up largest possible sums $\leq 180^\circ$ in one sweep over the α's. Theorem A enables us to compute the lower convex hulls C_1, C_2 and C_3 of $S_1 \cup .. \cup S_i$, $S_{i+1} \cup \ldots \cup S_j$ and $S_{j+1} \cup .. \cup S_k$, respectively, in linear time, while triangulating the region beyond the boundaries along with it. By theorem B the region within the boundaries can be triangulated in another linear number of steps.

□

Star-shaped polygons are a simple subclass of the reducible polygons. While they are easily recognized (cf. [3]), no efficient algorithm is known to test whether a given polygon is reducible.

4. References

[1] Garey, M.R., D.S. Johnson, F.P. Preparata and R.E. Tarjan, Triangulating a simple polygon, Inf. Proc. Lett. 7 (1978) 175-179.

[2] Graham, R.L., An efficient algorithm for determining the convex hull of a finite planar set, Inf. Proc. Lett. 1 (1972) 132-133.

[3] Lee, D.T. and F.P. Preparata, An optimal algorithm for finding the kernel of a polygon, J. ACM 26 (1979) 415-421.

[4] Shamos, M.I., Computational geometry, Ph.D. Thesis, Yale University, 1978 (to be published).