Triangulating Polygons with Holes

(Extended Abstract)

R. Bar-Yehuda *and* R. Grinwald

Computer Science Department

Technion - IIT, Haifa 32000, Israel

ABSTRACT

In this paper we study the time complexity of triangulating polygons, not only as a function of the number of vertices n, but also as a function of h, the number of holes, and H, the number of the holes' vertices. Let $T(n,h,H)$ be that time complexity. We present an algorithm that triangulates a polygon in $O(T(n,0,0) + h \log n + H \log H)$ time. If the holes are convex and in other practical cases, variations of the main algorithm take $O(T(n,0,0) + h \log n)$.

1. Introduction

The problem of triangulating a simple polygon is one of the major research topics in computational geometry. The first to present an $O(n \log n)$ time algorithm were Garey, Johnson, Preparata and Tarjan [Ga78]. Since then a lot of effort has been invested in finding an $o(n \log n)$ algorithm, thus showing that triangulation is easier than sorting. Tarjan and Van Wyk [TV88] have presented an $O(n \log\log n)$ time algorithm. Clarkson, Tarjan and Van Wyk also presented a randomized algorithm that has $O(n \log^* n)$ expected time. The recent result for a deterministic algorithm that triangulates a simple polygon is Chazelle's $O(n \log^* n)$ time algorithm [C90]. The only known lower bound for triangulating a simple polygon (without holes) is $\Omega(n)$.

An algorithm that handles polygons with holes is that of Fournier and Montuno [FM84], and its time is $T(n,h,H) = O(n \log n)$. We do not know of a better result nor of a previous work that has taken h and/or H as parameters in the time complexity. The only known lower bound for triangulating a polygon is $\Omega(n + h \log h)$ (even for the case that all the holes are triangles).

We present an algorithm that triangulates a polygon with holes in time $T(n,h,H) = O(T(n,0,0) + h \log n + H \log H)$. In many applications (VLSI, robotics) where $H<<n$, this is better than the known algorithms. A similar method will work in time $O(T(n,0,0) + h \log n)$ if the holes are convex, and in other special cases.

Our algorithm combines several ideas to achieve its result:

(1) Horizontal trapezoid decomposition (trapezoidation) of a polygon P (with holes) - $TR(P)$, is linear time equivalent to triangulation [CI84][FM84].

(2) Using the point location method of Kirkpatrick [K83].

(3) Creating a simple polygon without holes from a simple polygon with holes by connecting each hole horizontally to its closest right-side edge.

2. The main algorithm

The input of our algorithm is a list of polygons $P_0, P_1, \ldots, P_h$, where P_0 is the surrounding polygon, and the rest are its holes (in arbitrary order). Each polygon contains an ordered list of vertices, where the

surrounding polygon vertices are given in clockwise order, and the holes vertices are given in counter-clockwise order. The output is the polygon decomposed into trapezoids. Our data structure is similar to that in [FM84] (see also [CI84]). We will denote by $|P_i|$ the number of vertices of the i-th polygon.

The following are the general steps of our algorithm:

1. Compute the trapezoidation $TR(P_0)$ of the surrounding polygon (without the holes) in the best known algorithm ($O(T(|P_0|,0,0))$ time).
2. Process $TR(P_0)$ for point location. Because trapezoids are convex, this step takes $O(|P_0|)$ time [K83].
3. Build a rectangle that encloses the holes, and compute the trapezoidation $TR(\bigcup_{i=1}^{h}\{P_i\})$ of it including the holes. This step is done in a straight forward algorithm [FM84], and takes $O(H \log H)$ time.
4. For each hole P_i , $1 \le i \le h$ (see figure 1.):
   - 4.1 Locate its topmost vertex v_i.
   - 4.2 Find the trapezoid T_i in $TR(P_0)$ in which v_i resides, using the point location structure built in step 2 ($O(\log n)$).
   - 4.3 Compute the horizontal distance $dist(v_i,T_i)$ between v_i and the right edge of T_i.
   - 4.4 Find the closest edge e_i in $TR(\bigcup_{i=1}^{h}\{P_i\})$ that v_i sees on its right, and compute the horizontal distance $dist(v_i,e_i)$ to this edge ($O(1)$ time).
   - 4.5 If $dist(v_i,e_i) < dist(v_i,T_i)$ connect v_i to e_i ; otherwise connect v_i to the right edge of T_i ($O(1)$ time). By *connecting* we mean adding two connecting edges in opposite directions and three vertices, so that the hole is joined with the polygon on its right. Note that the edges added do not change the trapezoid decomposition of the polygon with its holes.
5. Now we have a single simple polygon without holes. Compute its trapezoidation in the best known algorithm ($O(T(n,0,0))$ time).

It is clear from the above description that the total time of our algorithm is $O(T(n,0,0) + h \log n + H \log H)$.

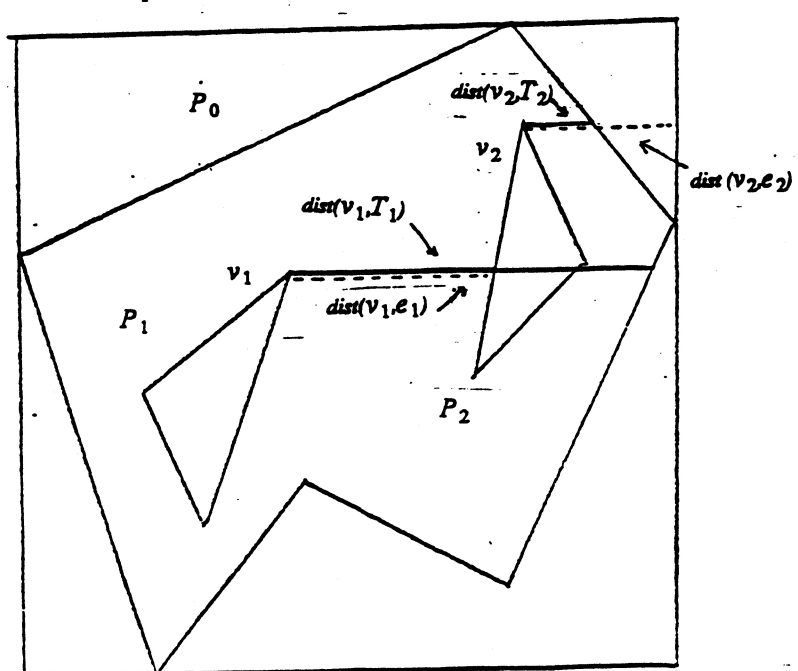

Figure 1.: the surrounding polygon P_0 with two holes P_1 and P_2, and the enclosing rectangle. The distances computed in steps 4.3 and 4.4 are shown.

3. Special Cases

In the following restricted cases we achieve a time bound of $T(n,h,H) = O(T(n,0,0) + h \log n)$.

1. Constant size holes -
In many practical cases (e.g. VLSI) the holes are of constant size (e.g. rectangles). This property may be checked in $O(n)$ time, and if it holds, $H = O(h)$, and thus the total time becomes $O(T(n,0,0) + h \log n)$. We will use this result in the following.

2. Convex (or y-monotone) holes -
The property of convex holes is also quite common in robotics, computer vision and VLSI, and may be checked in $O(n)$ time. In this case we add some extra processing to the main algorithm to achieve a better time bound.

The modified algorithm:

2.1 Preprocess the holes: degenerate each (convex) hole to a simple 2-vertex hole (segment): its upper vertex is the original hole topmost vertex, and the its lower vertex is the original hole bottommost vertex ($O(H)$ time).

2.2 Run steps 1 through 4 (of the main algorithm) on the polygon with the 'degenerated' holes ($O(T(n,0,0) + h \log n)$ time).

2.3 Now replace the 'degenerated' holes with the original ones; in so doing we trapezoidize each hole (in linear time). Going from one trapezoid to the next we fix (shorten) the connecting edges so that they connect an edge of the original hole. The total time of this step, over all the holes, is $O(H)$.

2.4 We have again one simple polygon (without holes) with $O(n)$ vertices. We trapezoidize it using the best known algorithm ($O(T(n,0,0))$ time).

We conclude that the total time for triangulating a polygon with h convex holes is $O(T(n,0,0) + h \log n)$. The above method is applicable also for y-monotone holes with the same time bound. If the surrounding polygon is also convex (or y-monotone), we replace step 4 of the main algorithm and step 2.4 with merging $TR(P_0)$ with $TR(\bigcup_{i=1}^{h}\{P_i\})$ in linear time, and the total time becomes $O(n + h \log h)$ (which is obviously optimal).

3. Rectilinearly disjoint holes -
We refer to a set of polygons as 'rectilinearly disjoint' if the smallest rectangles enclosing each polygon do not intersect or contain one another. This property can be checked in $O(h \log h)$ time. The method described in this section will apply also if the convex hulls of the holes are computed previous to the running of our algorithm, and are disjoint. In the following discussion we assume that the enclosing rectangles (convex hulls) of the holes do not intersect the surrounding polygon either.

The modified algorithm:

3.1 Build the (smallest) enclosing rectangle for each hole ($O(H)$ time).

3.2 Run steps 1 through 4 (of the main algorithm) on the polygon with the rectangular holes ($O(T(n,0,0) + h \log n)$ time).

3.3 (Similar to step 2.3) We fix the connecting edges so they will connect the 'real' hole and not the rectangle (see figure 2.). We do this by trapezoidizing the inside of each rectangle (the outside of the hole) by connecting the topmost vertex of hole P_i to the top edge of its enclosing rectangle, thus creating a simple polygon with $|P_i| + 5$ vertices. We walk from one trapezoid

to another and fix (lengthen) each connecting edge so it that will connect an edge of the 'real' hole (actually merging $TR(P_0)$ with $TR(P_i)$). The total time of this step over all the holes is $O(T(H,0,0))$.

3.4 We have again a simple polygon (without holes) with $O(n)$ vertices. We trapezoidize it using the best known algorithm ($O(T(n,0,0))$ time).

The total time for this algorithm is also $O(T(n,0,0) + h \log n)$.

4. Open problems

It will be interesting if a similar method may handle polygons with non-convex holes more efficiently. The major open problem remains to find an algorithm that triangulates a polygon in time $T(n,h,H) = T(n,0,0) + h \cdot o(\log n)$.

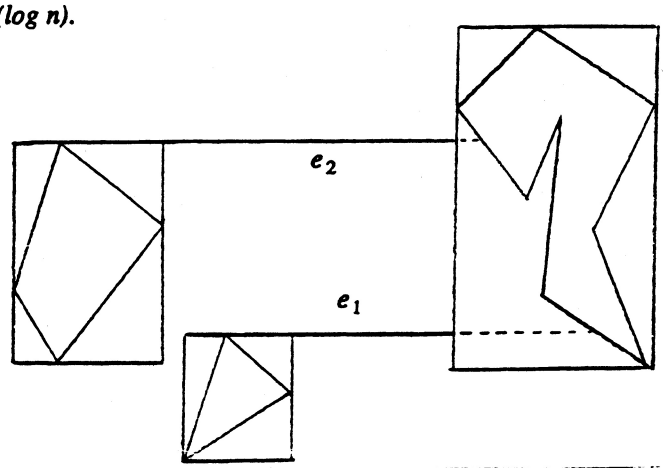

Figure 2.:lengthening the connecting edges e_1 and e_2.

5. References

[C90] Chazelle B., Efficient Polygon Triangulation. *Report CS-TR-249-90, Princeton University, Princeton*, 1990.

[CI84] Chazelle B. and J. Incepri, Triangulating and Shape Complexity. *ACM Transactions on Graphics*, 3 (1984), 135-152.

[CTV88] Clarkson K. L., R. E. Tarjan and C. J. Van Wyk, A Fast Las Vegas Algorithm for Triangulating a Simple Polygon. *Proc. the ACM Symp. on Computational Geometry*, 1988, 18-22.

[FM84] Fournier A. and D. Y. Montuno, Triangulating Simple Polygons and Equivalent Problems. *ACM Transactions on Graphics*, 3 (1984), 153-174.

[Ga78] Garey M. R., D. S. Johnson, F. P. Preparata And R. E. Tarjan, Triangulating a Simple Polygon. *Information Processing Letters*, 7 (1978), 175-180.

[K83] Kirkpatrick D. G., Optimal search in planar subdivisions, *SIAM Journal on computing*, 12 (1983), 28-35.

[TV88] Tarjan R. E. and C. J. Van Wyk, An $O(n \log\log n)$ -time Algorithm for Triangulating a Simple Polygon. *SIAM Journal on Computing*, 17 (1988), 143-178.