Lecture Notes by Diane Souvaine

TRIANGULATION ALGORITHM

Suppose we have a finite planar subdivision S on n vertices which consists of a large triangle Q and N polygons interior to Q. All of the finite reqions may be triangulated in $O(n \log n)$ time. At first, they may have little uniformity. The polygons themselves may have any number of reflex angles (angles whose measures exceed 180 °). In addition, one region consists of a triangle with N polygonal holes. By adding edges, however, we can break up as many of the $N+1$ regions as necessary and create a subdivision S' whose faces are all polygons monotone with respect to the x-axis. First, we describe a technique for triangulating monotone polygons. Secondly, we will discuss a method for decomposing the faces of a planar subdivision into polygons monotone with respect to the x-axis.

A polygon P with consecutive vertices $v_1, v_2, \ldots, v_m$, leftmost vertex v_1 and rightmost vertex v_j is monotone with respect to the x-axis iff both $v_1, v_2, \ldots, v_{j-1}, v_j$ and $v_1, v_m, \ldots, v_{j+1}, v_j$ are in increasing order of x-coordinate. We can merge these two sequences in time $O(m)$ to form a sequence $q_1, q_2, \ldots, q_m$ containing all vertices of P in increasing order of x-coordinate. To triangulate P, we will move through these vertices in order, adding edges where possible and retaining on a stack those vertices already considered but which still lie on the boundary of a polygon yet to be triangulated.

Before describing the algorithm in general, let us consider the example in Figure ~~27~~ 1. We begin by pushing q_1, q_2 on the stack. Next, we consider q_3. It is already adjacent to q_2 so an edge from q_3 to q_2 would be redundant. As the interior angle at q_2 exceeds 180 °, an edge from q_3 to q_1 could not lie inside the polygon. Push q_3 on the stack. The next vertex, q_4, presents the same problems, so it too is pushed onto the stack. Vertex q_5, however, is adjacent to q_4 and the interior angle at q_4 measures less than 180 °. We pop q_4, add edge e_1, and then consider the angle $\angle q_5 q_3 q_2$. As it also measures less than 180 °, we pop q_3, add edge e_2, and then consider angle $\angle q_5 q_2 q_1$. As it is not smaller than a straight angle, an edge from q_5 to q_1 would not lie in the interior of P, so we push q_5 and move to q_6. It is adjacent to q_1, the first vertex on the stack. We add the edges e_3 and e_4, empty the stack, and then push q_5, q_6. q_7 and q_6 are adjacent and the current interior angle at q_6 measures less than 180 ° so we can pop q_5, add e_5, and then push q_7. As the angle at q_7 exceeds 180 °, no edge can be added at q_8, so we push that vertex on the stack. The last vertex, q_9 must be adjacent to both the bottom and top elements on the stack, q_5 and q_8. We add e_6, the edge from q_9 to the only middle element of the stack. Then we empty the stack, and stop.

The general algorithm proceeds as follows:

Begin

1) $S[1] \leftarrow q_1$

2) $S[2] \leftarrow q_2$

3) $top \leftarrow 2$

4) For $i = 3$ to m begin

5) If q_i is adjacent to $S[top]$, but not to $S[1]$, begin

6) While $top > 1$ and $\angle q_i S[top] S[top{-}1] < 180^\circ$ begin

7) add an edge from q_i to $S[top]$

8) $top \leftarrow top - 1$

end

9) $top \leftarrow top + 1$

10) $S[top] \leftarrow q_i$

end

11) If q_i is adjacent to $S[1]$, but not to $S[top]$, begin

12) For $j = 2$ to top, add an edge from q_i to $S[j]$

13) $S[1] \leftarrow S[top]$

14) $S[2] \leftarrow q_i$

15) $top \leftarrow 2$

end

16) If q_i is adjacent to both $S[1]$ and $S[top]$, begin

17) For $j = 2$ to $top - 1$ add an edge from q_i to $S[j]$

18) $top \leftarrow 0$

end

end

End

Only when $i = m$ and we are processing the last vertex is the condition of step 13 satisfied. Thus, only after processing the last vertex is the stack emptied (step 15). Each vertex is processed once prior to putting it on the stack. At most one edge is added as each vertex is popped from the stack. Thus the entire algorithm runs in linear time.

Now we wish to add edges between existing vertices in the planar subdivision S to achieve a planar subdivision S' whose faces are all polygons monotone with respect to the x-axis. First we must identify those vertices which violate the condition of monotonicity. Consider the example of Figure 2. First we sort the vertices relative to their x-coordinate and label them $v_1, \ldots, v_{13}$. At each vertex, v_i, we add two temporary vertical edges: one extends from v_i to the edge above, forming a temporary vertex there; the other extends from v_i to the edge below, also forming a temporary vertex. When this process is complete, the faces of S' are all trapezoids whose bases are parallel to the y-axis (some are degenerate and form triangles with a single side parallel to the y-axis). [See Figure 3 ~~B~~]. We focus on the 5 trapezoids which each contain a permanenent vertex in the interior of a side. In each case, we add an edge from that vertex to another permanent vertex lying on the same trapezoid.

TRAPEZOID	EDGE
T_1	(v_3, v_1)
T_2	(v_5, v_3)
T_3	(v_7, v_8)
T_4	(v_9, v_{10})
T_5	(v_{12}, v_{13})

Following this procedure, delete all temporary edges. Six polygons monotonic in x result. [See Figure 4 ~~C~~].

To add the temporary edges, we will sweep a vertical line across S' keeping an accurate list of active edges and their relationship to each other. Each edge will have a field which points to the edge above and a field which points to the edge below. The sweep line begins at v_1 and v_2, the leftmost vertices. The edge e_1 enters and leaves the data structure almost at the same instant. As the vertical line moves toward v_3, the data structure contains two edges. At v_3, two more edges are inserted into the data structure and the pointers are updated:

EDGE	ABOVE	BELOW
e_2	e_4	ϕ
e_3	ϕ	e_5
e_4	e_5	e_2
e_5	e_3	e_4

Since v_3 joins e_4 and e_5, we extend a vertical edge, t_1, from v_3 to e_3 and a second edge, t_2, from v_3 to e_2. At v_4, e_6 replaces e_4 in the data base and vertical edges are added from v_4 to e_5 and e_2. At v_5, two new edges are inserted, and at v_6 another exchanges takes place. At v_7, two edges are deleted.

If we keep all of these edges in the leaves of a balanced tree and if each interior node contains the value of the uppermost edge of its subtree, then the processing time at each vertex is only $O(\log\ n)$ where n is the number of vertices. Consequently, all of the temporary edges may be added to S' in time $O(n \log n)$. As S' is a planar graph, it contains fewer that $2n$ trapezoids.

We can consider each one and add any necessary permanent edges in time $O(n)$. Thus the overall time of this algorithm is $O(n \log n)$.

BIBLIOGRAPHIC NOTES

The algorithm for triangulating monotone polygons is due to Garey, Johnson, Preparata and Tarjan [GJPT78]. The method of decomposition of a the faces of a planar subdivision into monotone polygons is derived from the work of Chazelle and Incerpi [Ch84b] and [CI83].

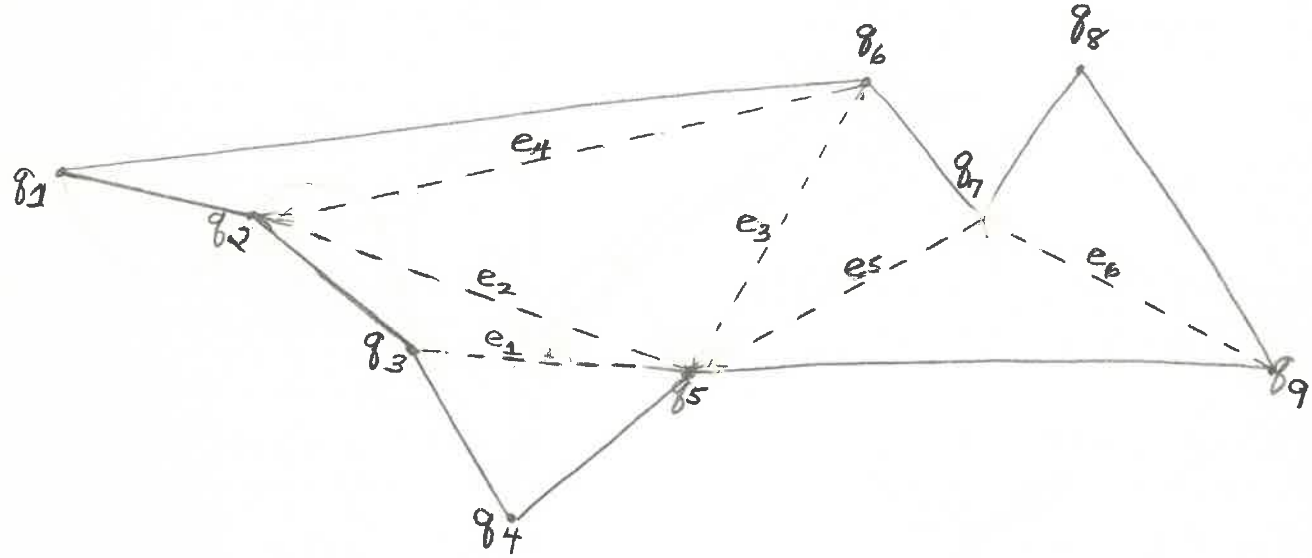

FIGURE 1

TRIANGULATION OF A MONOTONE POLYGON

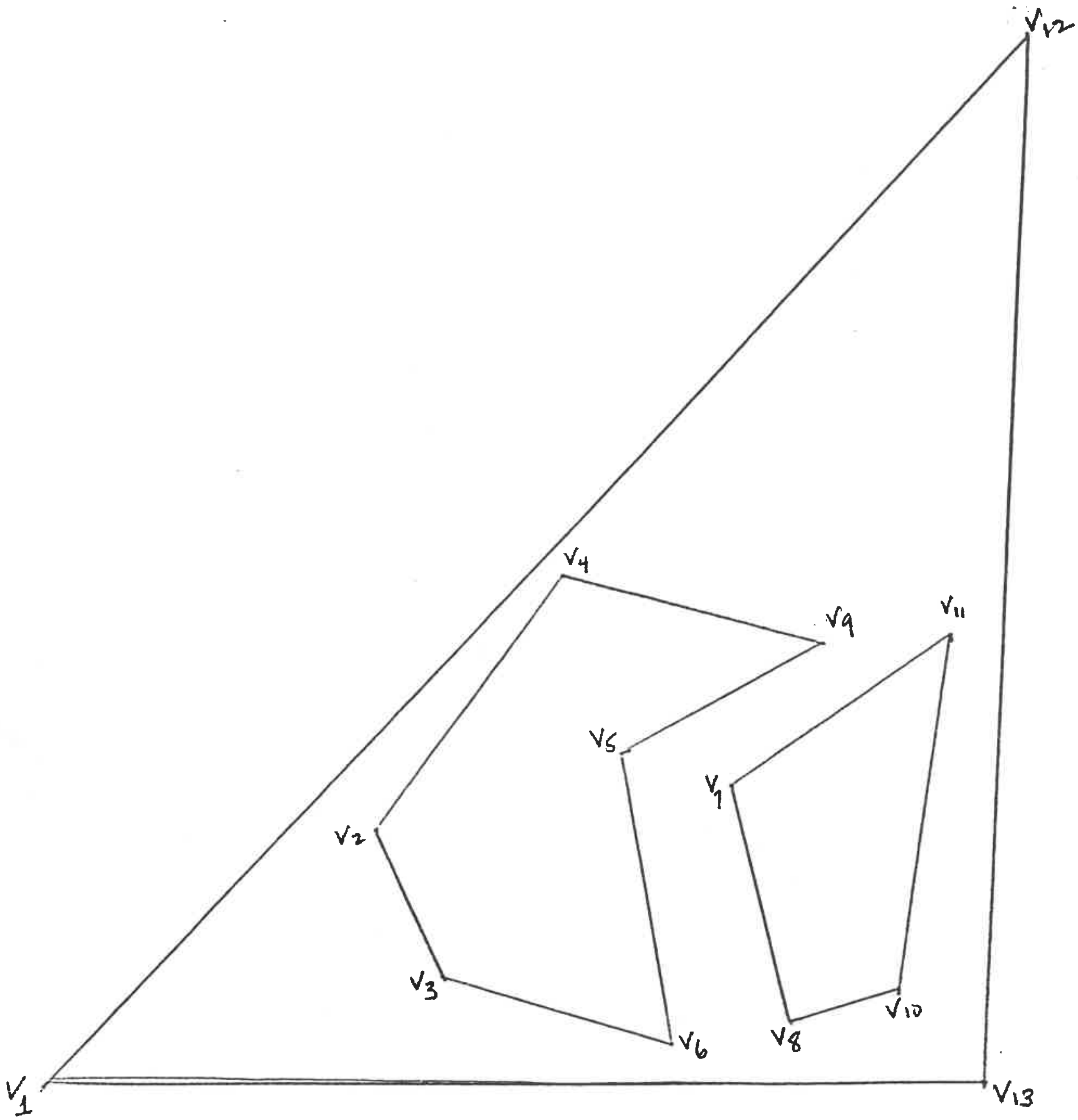

FIGURE 2