Delaunay Triangular Meshes in Convex Polygons

Barry Joe

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

CS-84-15

June 1984

Delaunay Triangular Meshes in Convex Polygons

Barry Joe

Department of Computer Science
University of Waterloo

Abstract

An algorithm for producing a triangular mesh in a convex polygon is presented. It is used in a method for the finite element triangulation of a complex polygonal region of the plane in which the region is decomposed into convex polygons. The interior vertices of the mesh are chosen to be on a quasi-uniform grid, different mesh spacings are specified for the edges of the polygon, and the mesh is a Delaunay triangulation. The correctness of the algorithm is proved and the time complexity is discussed.

Key words - mesh generation, finite element method, computational geometry, Delaunay triangulation

1. Introduction

A common approach for generating triangular meshes in general regions of the plane for the finite element method is to first decompose the region into simpler subregions and then to triangulate each subregion (Cavendish (1974), Shaw and Pitchen (1978), Bykat (1976), Bank (1982)). We have developed a method for the finite element triangulation of a complex polygonal region of the plane in which the region is decomposed into convex polygons such that small interior angles in the polygons are avoided and then a triangular mesh is generated in each convex polygon such that triangles with small angles are avoided (Joe and Simpson (1983), Joe (1984)). Figure 1.1 illustrates a triangulation of a region produced by this method.

In this paper, we describe the algorithm used in this method for generating a triangular mesh in a convex polygon P, and prove its correctness. The triangles of the mesh satisfy an optimal angle criterion, i.e. the mesh is a Delaunay triangulation (Lawson (1977)). Let $e_1, e_2, \ldots, e_m$ be the m edges of P in counterclockwise order. The input of the algorithm consists of the vertices of P in counterclockwise order, triangle size parameter h for the interior of P, and triangle size parameters $h_1, h_2, \ldots, h_m$ for the edges $e_1, e_2, \ldots, e_m$, respectively. Vertices are generated in the interior of P using a quasi-uniform grid of spacing h (cf. Shaw and Pitchen (1978)). Vertices are generated on edge e_i at an equal spacing of approximately h_i. In the Delaunay triangulation of these vertices, the triangles in the interior of P have constant area $h^2/2$. Different h_i are specified on the e_i so that triangles which are graded in size are produced near ∂P (the boundary of P).

In our finite element triangulation method, the triangle size parameters for the interior of the convex polygons in the decomposition of a region are restricted to differ by a factor of at most two in adjacent polygons so that a gradual change of triangle sizes occurs between adjacent polygons. The triangle size parameter for an edge of the decomposition is set to the geometric mean of the triangle size parameters for the interior of the one or two convex polygons with this edge. Hence the parameters h_i for the edges of a convex polygon P in the decomposition satisfy

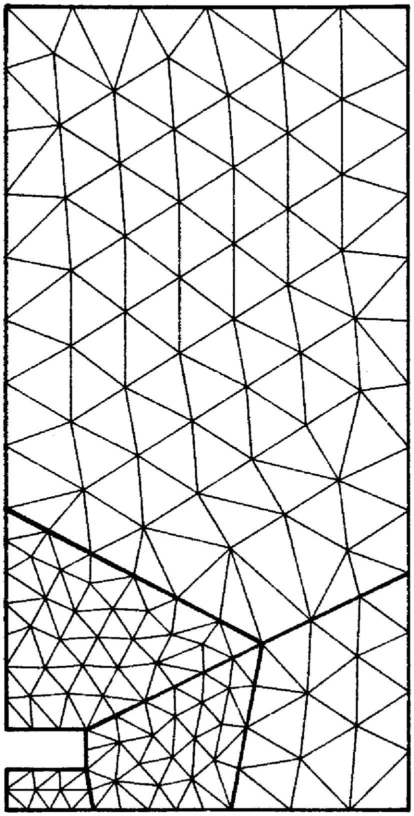

Figure 1.1 Illustration of triangulation of a region;
thicker lines indicate decomposition into convex polygons

$$h/\sqrt{2} \le h_i \le \sqrt{2}h. \tag{1.1}$$

With this restriction, the triangles near ∂P have approximately constant area $h^2/2$ and the number of triangles in P is approximately $2a/h^2$ where a is the area of P, so the triangle size parameter h can be determined from a and the desired number of triangles in the mesh.

In the standard triangulation problem of computational geometry, the vertices are given as input and a (Delaunay) triangulation is constructed to cover the convex hull of the vertices. The construction of a Delaunay triangulation (or any triangulation) of the vertices requires $O(n_t \log n_t)$ time where n_t is the number of triangles in the triangulation (Shamos and Hoey (1975), Lee and Schachter (1979)). However, in finite element triangulation, the vertices are generated as well as the triangles. By using information about the location of the vertices relative to each other, our algorithm constructs a Delaunay triangulation of the vertices in an expected time of $O(n_t)$ when the h_i satisfy (1.1) and P contains no 'small' interior angles.

In Section 2, we define a valid triangulation and a Delaunay triangulation. In Section 3, we describe an algorithm for shrinking a convex polygon which we use to determine a convex polygon, $int(P)$, in the interior of P. In our triangulation algorithm, we construct a preliminary triangulation, $VT(P)$, by generating triangles in $int(P)$ and in the strip near ∂P, as described in Sections 4 and 5, respectively. The validity of $VT(P)$ is discussed in Section 6. In Section 7, we describe how $VT(P)$ is converted into a Delaunay triangulation, $DT(P)$. The time complexity of the triangulation algorithm is discussed in Section 8.

2. Valid and Delaunay triangulations

In this section we state conditions for a valid triangulation in a region and state some properties of a Delaunay triangulation of a set of vertices. We say that a collection of triangles is a valid triangulation of a polygonal region of the plane if the triangles form a 'tiling' of the region without overlaps or gaps. Let w_1, w_2, w_3 be distinct vertices. We define triangle $\Delta w_1w_2w_3$ to be a counterclockwise (CCW) triangle if the interior of the triangle is to the left of the three directed edges w_1w_2, w_2w_3, w_3w_1; otherwise $\Delta w_1w_2w_3$ is a clockwise (CW) triangle. If w_1, w_2, w_3 are collinear, we consider $\Delta w_1w_2w_3$ to be a CW triangle. Note that the ordering of the vertices of a triangle determine whether it is a CCW or CW triangle. Four conditions that ensure that a collection of triangles, $\Delta_1, \Delta_2, \ldots, \Delta_N$, is a valid triangulation of a region have been established by Simpson (1981). They can be described in geometric terms as follows:

(a) Δ_i is a CCW triangle for all i.

(b) Each edge e of Δ_i is either the only edge joining its vertices or there is one other triangle, Δ_j, having an edge joining these vertices. In the latter case, the direction of e as an edge of Δ_i is opposite to its direction as an edge of Δ_j. In the former case, e is a boundary edge and Δ_i is its boundary triangle. (2.1)

(c) No interval of a boundary edge intersects a triangle other than its boundary triangle.

(d) A vertex can have at most one boundary edge directed away from it.

Let V be a set of vertices in the plane such that they are not all collinear. A Delaunay triangulation of V is a (valid) triangulation in the convex hull of V which satisfies the max-min angle criterion: for any two triangles in the triangulation that share a common edge, if the quadrilateral formed from the two triangles with the common edge as its diagonal is strictly convex, the replacement of the diagonal by the alternative one does not increase the minimum of the six angles in the two triangles making up the quadrilateral (Sibson (1978)). In other words, the Delaunay triangulation is the triangulation which maximizes the minimum angle in the triangles globally as well as locally in any two adjacent triangles which form a strictly convex quadrilateral.

A Delaunay triangulation also satisfies the circle criterion: the circumcircle of any triangle in the triangulation contains no vertex of V in its interior. The Delaunay triangulation is unique if no four vertices are co-circular. An edge uv, where u and v are vertices of V, is a Delaunay edge if it is an edge in a Delaunay triangulation of V.

Lemma 2.1 (Lee and Schachter (1979)) : An edge uv is a Delaunay edge if and only if there exists a point c such that the circle centred at c and passing through u and v does not contain any other vertex of V in its interior.

The following local optimization procedure (LOP) of Lawson (1977) can be used to convert a triangulation of V into a Delaunay triangulation. Let e be an internal edge (i.e. an edge not on the boundary of the convex hull) of a triangulation of V and Q be the quadrilateral formed by the two triangles having e as a common edge. If the circumcircle of one of the triangles contains the fourth vertex of Q in its interior, then e is replaced by the other diagonal of Q (so that the minimum of the angles in the two triangles is increased). Edge e is said to be *locally optimal* if an application of LOP would not swap it. Note that e is locally optimal if Q is a nonconvex quadrilateral, and that the validity conditions (2.1) are unaffected by the LOP.

Theorem 2.1 (Lawson (1977)) : All internal edges of a triangulation T of V are locally optimal if and only if T is a Delaunay triangulation of V.

In our algorithm, we generate a preliminary valid triangulation, $VT(P)$, in P which satisfies the four conditions of (2.1). Then $VT(P)$ is converted into a Delaunay triangulation, $DT(P)$, by applying LOP to the internal edges of $VT(P)$. We have constructed $VT(P)$ so that LOP only has to be applied to a subset of the internal edges.

3. Shrinking a convex polygon

Let $p_0, p_1, \ldots, p_{m-1}, p_m$ be the vertices of convex polygon P in counterclockwise order, where $p_m = p_0$ and all interior angles are less than 180°. Let Q be obtained by shrinking P by a distance of $r > 0$, i.e. if $v \in Q$ then the distance from v to $\partial P \geq r$. In this section, we present an algorithm for constructing Q which uses the same approach as the algorithm of Lee and Preparata (1979) for finding the kernel of a simple polygon. They exploit the ordering of the half-planes corresponding to the polygon edges to obtain a linear time algorithm.

First we give some notation for representing Q and ∂Q. Let l_i be the directed line from p_i to p_{i+1}, L_i be the directed line parallel to l_i and at distance r to the left of l_i, and H_i be the half-plane to the left of and including L_i. Let qL_iq' denote the directed line segment from q to q' where q and q' are two points on L_i, and let $qL_i\infty$ and ∞L_iq denote the directed half-lines starting and ending at q, respectively, on L_i. Q is the intersection of the half-planes $H_0, H_1, \ldots, H_{m-1}$. If r is sufficiently small then Q is a convex polygon otherwise Q is degenerate, i.e. Q is either empty or a single point or a line segment. In the degenerate case, we treat Q as empty. If Q is a convex polygon then ∂Q consists of line segments from $n \geq 3$ of the lines L_i (if $\bigcap_{i \neq j} H_i = Q$ then no line segment of L_j is part of ∂Q). Using the above notation

$$\partial Q = q_0 L_{k_0} q_1 L_{k_1} q_2 \cdots q_{n-1} L_{k_{n-1}} q_n, \quad 0 \leq k_0 < k_1 < \cdots < k_{n-1} \leq m-1,$$

where $q_n = q_0$ is the intersection of L_{k_0} and $L_{k_{n-1}}$ and q_i is the intersection of $L_{k_{i-1}}$ and L_{k_i}, $1 \leq i \leq n-1$ (see Figure 3.1).

Our algorithm for constructing Q scans in order the edges $p_i p_{i+1}$ of P and constructs a sequence of convex polygons $Q_1, Q_2, \ldots, Q_{m-1}$ such that Q_i is the intersection of half-planes $H_0, H_1, \ldots, H_i$. Since P is a convex polygon, the polar angles, θ_i, of the vectors $p_{i+1} - p_i$ (or lines l_i) are ordered. Without loss of generality assume $\theta_0 = 0°$, i.e. l_0 is a horizontal line directed from left to right, so that $0° = \theta_0 < \theta_1 < \cdots < \theta_{m-1} < 360°$. Let s be the smallest index such that $\theta_s > 180°$ (note that $2 \leq s \leq m-1$). If $\theta_{s-1} = 180°$ then let $s' = s-2$ otherwise let $s' = s-1$. The following facts can be seen from elementary geometry:

(a) For $1 \leq i \leq s'$, Q_i is an unbounded convex polygon.

(b) If $\theta_{s-1} = 180°$, Q_{s-1} is either an unbounded convex polygon or degenerate. (3.1)

(c) For $s \leq i \leq m-1$, Q_i is either a bounded convex polygon or degenerate.

Now we describe our algorithm for constructing Q. Initially Q_1 is the intersection of H_0 and H_1 and $\partial Q_1 = \infty L_0 q_1 L_1 \infty$ where q_1 is the intersection of L_0 and L_1. For $2 \leq i \leq s'$, Q_i is the intersection of Q_{i-1} and H_i and ∂Q_i is obtained by modifying ∂Q_{i-1}. Suppose

$$\partial Q_{i-1} = q_0 L_{k_0} q_1 \cdots q_n L_{k_n} q_{n+1}, \; q_0 = q_{n+1} = \infty, \; 0 = k_0 < k_1 < \cdots < k_n = i-1. \tag{3.2}$$

Since Q_{i-1} is convex and $\theta_j < \theta_i < 180°$ for $j < i$, ∂Q_{i-1} and L_i intersect at exactly one point, t (see Figure 3.2). Let $j \geq 0$ be the index such that t is on the line segment $q_j L_{k_j} q_{j+1}$ and $t \neq q_j$. j can be found by scanning backwards from n for the first q_j to the left of L_i where $q_0 = \infty$ is

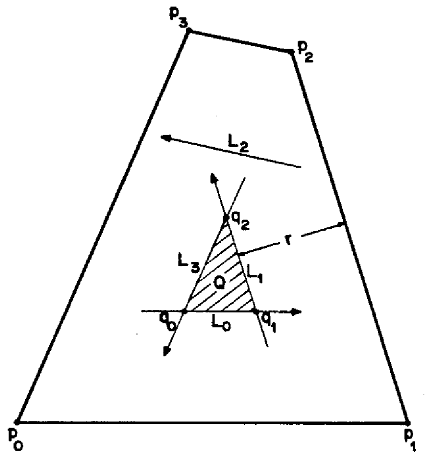

Figure 3.1 Illustration of convex polygon Q

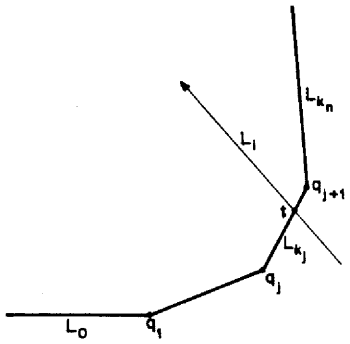

Figure 3.2 ∂Q_{i-1} and L_i intersect at point t

considered to be to the left of L_i. The polygonal curve $tL_{k_j}q_{j+1} \cdots q_{n+1}$ is to the right of L_i so it is not part of ∂Q_i. It is replaced by $tL_i\infty$ to obtain $\partial Q_i = q_0 \cdots q_j L_{k_j} t L_i \infty$ which can be expressed in the same form as (3.2). If $\theta_{s-1} = 180°$ then if all points of ∂Q_{s-2} (in particular $q_1, q_2, \ldots, q_n$) are to the right of or on L_{s-1} then Q_{s-1} is degenerate and the algorithm terminates otherwise ∂Q_{s-1} is obtained by modifying ∂Q_{s-2} as above.

Suppose Q_{s-1} is not degenerate. Q_s is either a bounded convex polygon or degenerate by (3.1c) and ∂Q_s is obtained by modifying ∂Q_{s-1}. If all points of ∂Q_{s-1} are to the right of or on L_s then Q_s is degenerate and the algorithm terminates. Otherwise L_s intersects ∂Q_{s-1} at exactly two points, t_1 and t_2, since Q_{s-1} is convex and $\theta_s > 180°$. Suppose ∂Q_{s-1} is in the form of (3.2) with $i = s$ and t_1 occurs before t_2 in the polygonal curve (see Figure 3.3). Suppose j and l are the indices ($j < l$) such that t_1 is on line segment $q_j L_{k_j} q_{j+1}$ and $t_1 \neq q_{j+1}$ and t_2 is on line segment $q_l L_{k_l} q_{l+1}$ and $t_2 \neq q_l$. l can be found by scanning backwards from n for the first q_l to the left of L_s and j can be found by scanning forward from 0 for the first q_{j+1} to the left of L_s. The polygonal curves $t_2 L_{k_l} q_{l+1} \cdots q_{n+1}$ and $q_0 \cdots q_j L_{k_j} t_1$ are to the right of L_s so they are not part of ∂Q_s. They are replaced by $t_2 L_s t_1$ to obtain $\partial Q_s = t_1 L_{k_j} q_{j+1} \cdots q_l L_{k_l} t_2 L_s t_1$ which can be expressed in the same form as (3.3) below.

Suppose $s+1 \leq i \leq m-1$ and Q_{i-1} is not degenerate. ∂Q_i is obtained by modifying

$$\partial Q_{i-1} = q_0 L_{k_0} q_1 \cdots q_n L_{k_n} q_{n+1},\ q_0 = q_{n+1},\ 0 \leq k_0 < k_1 < \cdots < k_n \leq i-1. \tag{3.3}$$

Consider the position of L_i with respect to Q_{i-1}. There are three cases (see Figure 3.4):

(a) Q_{i-1} is to the right of or on L_i,

(b) Q_{i-1} is to the left of or on L_i,

(c) ∂Q_{i-1} and L_i intersect at exactly two points, t_1 and t_2.

In case (a), $q_0, q_1, \ldots, q_n$ are to the right of or on L_i, Q_i is degenerate, and the algorithm terminates. In case (b), $Q_i = Q_{i-1}$ and the condition that q_0 is to the left of or on L_i is sufficient to determine this case since $\theta_{k_0} \geq 0°$ and $\theta_{k_n} < \theta_i$, i.e. the slope of L_i lies between the slopes of L_{k_n}

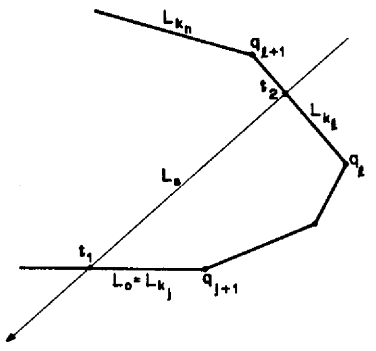

Figure 3.3 ∂Q_{s-1} and L_s intersect at points t_1 and t_2

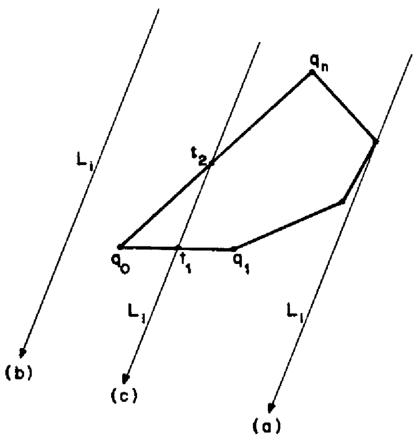

Figure 3.4 Three cases of position of L_i with respect to Q_{i-1}

and L_{k_0}. In case (c), ∂Q_i is determined in the same way as ∂Q_s above. If Q_i is not degenerate for $s'+1 \le i \le m-1$, then the algorithm terminates with $\partial Q = \partial Q_{m-1}$.

The pseudo-code for our algorithm is given in procedure SHRINK.

```
Procedure SHRINK($P,m,r,Q,n$);
# Input: convex polygon $P = p_0 p_1 \cdots p_{m-1} p_m$ and shrinking distance $r$
# Output: $Q = \phi$ and $n = 0$ or convex polygon $Q = q_0 q_1 \cdots q_{n-1} q_n$ and $3 \le n \le m$
    $\alpha$ := polar angle of $p_1 - p_0$;
    $q_1$ := intersection of $L_0$ and $L_1$;
    $k_0$ := 0; $k_1$ := 1;
    $i$ := 2; $n$ := 1;
    $\theta$ := (polar angle of $p_3 - p_2$) $-$ $\alpha$;
    Translate angle $\theta$ to the interval $[0°, 360°)$;
    while $\theta \le 180°$ do
        # polygonal boundary curve is $\infty L_{k_0} q_1 \cdots q_n L_{k_n} \infty$ and next line is $L_i$
        while $n \ge 1$ and $q_n$ is to the right of or on $L_i$ do $n := n - 1$;
        if $\theta = 180°$ and $n = 0$ then return;
        $t$ := intersection of $L_{k_n}$ and $L_i$;
        $n := n + 1$; $q_n := t$; $k_n := i$;
        $i := i + 1$; $\theta$ := (polar angle of $p_{i+1} - p_i$) $-$ $\alpha$;
        Translate angle $\theta$ to the interval $[0°, 360°)$;
    $j$ := 0; $q_j := \infty$; $q_{n+1} := \infty$;
    while $i \le m - 1$ do
        # polygonal boundary curve is $q_j L_{k_j} q_{j+1} \cdots q_n L_{k_n} q_{n+1}$ and next line is $L_i$
        if $q_j = \infty$ or $q_j$ is to the right of $L_i$ then
            while $n \ge j + 1$ and $q_n$ is to the right of or on $L_i$ do $n := n - 1$;
            if $n = j$ then $n$ := 0; return;
            $t_2$ := intersection of $L_{k_n}$ and $L_i$;
            $n := n + 1$; $q_n := t_2$; $k_n := i$;
            while $q_{j+1}$ is to the right of or on $L_i$ do $j := j + 1$;
            $t_1$ := intersection of $L_{k_j}$ and $L_i$;
            $q_j := t_1$; $q_{n+1} := t_1$;
        $i := i + 1$;
    for $i$ := 0 to $n + 1 - j$ do $q_i := q_{i+j}$;
    $n := n + 1 - j$;
    return;
```

We now derive the time complexity for the algorithm. For each $i \ge 2$, ∂Q_i is obtained from ∂Q_{i-1} by removing zero or more line segments from the beginning and end of ∂Q_{i-1}. Each line segment is parallel to an edge of ∂P and when it is removed from ∂Q_{i-1} it is not involved in any further computation. Therefore at most m line segments are removed from the ∂Q_i, $1 \le i \le m-2$. All other computations clearly require $O(m)$ time. Therefore we have shown

Theorem 3.1 : Procedure SHRINK determines Q in $O(m)$ time.

4. Triangulation of the interior of P

Let $int(P)$ be obtained by shrinking P by a distance of $h/\sqrt{2}$, i.e.

$$int(P) = \{v \mid v \in P \text{ and distance from } v \text{ to } \partial P \geq h/\sqrt{2}\}. \qquad (4.1)$$

If $h/\sqrt{2}$ is sufficiently small then $int(P)$ is a convex polygon, otherwise $int(P)$ is degenerate, i.e. it is empty, a point, or a line segment. In the degenerate case, no interior triangulation is done, so we assume that $int(P)$ is a convex polygon in this section.

Let v_b and v_t be the endpoints of a diameter of $int(P)$. The algorithm of Shamos (1975) is used to compute the diameter in linear time. We rotate the coordinate system so that line segment $v_b v_t$ is parallel to the y-axis with $y(v_t) > y(v_b)$, where $y(v)$ denotes the y-coordinate of vertex v. We introduce $n+1$ horizontal lines, $y = y_i$, through $int(P)$, evenly spaced h apart with $n = \lfloor (y(v_t) - y(v_b))/h \rfloor$; $dy = (y(v_t) - y(v_b) - n\,h)/2$; and $y_i = y(v_t) - dy - i\,h$, $0 \leq i \leq n$. Let a_i and b_i be the x-coordinates of the left and right endpoints of $y = y_i$ in $int(P)$ and let $m(i) = \lfloor (b_i - a_i)/h \rfloor$; $dx_i = (b_i - a_i - m(i)h)/2$; and $x_{i,j} = a_i + dx_i + j\,h$, $0 \leq j \leq m(i)$. On each line, we introduce a sequence of mesh vertices $(x_{i,j}, y_i)$ for $0 \leq j \leq m(i)$, spaced h apart. Note that at least one vertex is generated on each line and that the vertex which is nearest to ∂P is between $h/\sqrt{2}$ and $h/2 + h/\sqrt{2}$ distance from ∂P (see Figure 4.1). These vertices do not lie on a uniform square grid of spacing h because those on $y = y_{i+1}$ may be shifted horizontally with respect to those on $y = y_i$ or $y = y_{i+2}$. Consequently we have referred to them as being on a *quasi-uniform* grid.

A subset (possibly empty) of these vertices are then triangulated in a scan down the strips $y_{i+1} \leq y \leq y_i$ (see Figure 4.2). For each pair of lines, let $a = \max(x_{i,0}, x_{i+1,0})$ and $b = \min(x_{i,m(i)}, x_{i+1,m(i+1)})$. (Note that it is possible that $b < a$. In this case $a - b \leq h$ since on lines $y = y_i$ and $y = y_{i+1}$ there exists a vertex at distance $\leq h/2$ from diameter $v_b v_t$.) The vertices on the two lines for which the x-coordinate is in the interval $[a-h,\, b+h]$ are then connected up to form a sequence of similar triangles in which the area is $h^2/2$ and the angles are between 45° and 90° inclusive. This process is carried out for each pair of lines in which $a \leq b$

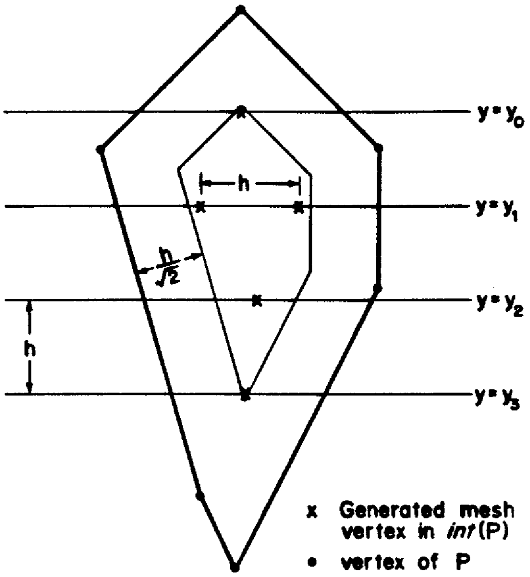

Figure 4.1 Generation of mesh vertices in $int(P)$

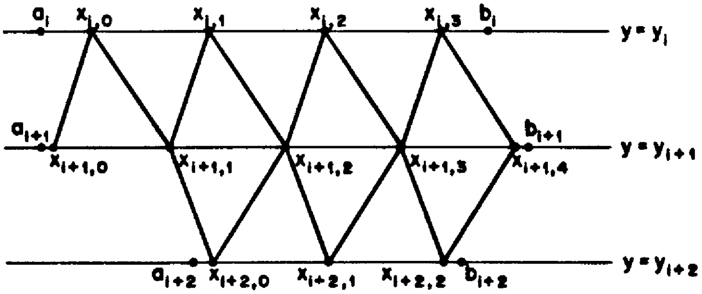

Figure 4.2 Triangulation of mesh vertices in $int(P)$

and $m(i) + m(i+1) > 0$. If $b < a$ or $m(i) = m(i+1) = 0$, then no triangles are formed between $y = y_i$ and $y = y_{i+1}$. In this case, the shortest edge joining a vertex on each line is introduced.

We need to identify a boundary of this set of vertices, so that we can triangulate the strip between it and ∂P. Let V_I be the set of vertices, and E_I be the following set of edges:

(a) edges of triangles formed in the scans described above,

(b) additional edges joining consecutive vertices at the ends of lines $y = y_i$, (4.2)

(c) the shortest edge joining a vertex on line $y = y_i$ to a vertex on line $y = y_{i+1}$, in the case that no triangles are formed between these lines.

Then $G_I = (V_I, E_I)$ is a connected planar graph, and if it contains at least two vertices, then a counterclockwise closed walk, C, can be formed to identify the boundary of G_I by including an edge e of E_I k times in C if e occurs in $2-k$ triangles, $0 \le k \le 2$. If an edge e occurs twice in C, the direction is opposite in its two occurrences (see Figure 5.1). If V_I contains only one vertex then let C be this single vertex.

C consists of a walk, C_L, down the left side of G_I followed by the sequence of edges on the line $y = y_n$ from left to right, and a walk, C_R, up the right side of G_I followed by the sequence of edges on the line $y = y_0$ from right to left. C_L consists of paths from $(x_{i,0}, y_i)$ to $(x_{i+1,0}, y_{i+1})$, $i = 0,1, \ldots, n-1$, which are constructed as follows. The leftmost edge between lines $y = y_i$ and $y = y_{i+1}$ is either part of the triangulation or the edge formed in (4.2c). This edge joins vertices $(x_{i,j}, y_i)$ and $(x_{i+1,k}, y_{i+1})$ with either $j = 0$ or $k = 0$ (or both). If $j = 0$ then the path formed by the vertices $(x_{i,0}, y_i)$, $(x_{i+1,k}, y_{i+1})$, $(x_{i+1,k-1}, y_{i+1})$, $\ldots$, $(x_{i+1,0}, y_{i+1})$ is a subwalk of C_L, otherwise the path formed by the vertices $(x_{i,0}, y_i)$, $(x_{i,1}, y_i)$, $\ldots$, $(x_{i,j}, y_i)$, $(x_{i+1,0}, y_{i+1})$ is a subwalk of C_L. For example, $(x_{i,0}, y_i)$, $(x_{i+1,0}, y_{i+1})$, $(x_{i+1,1}, y_{i+1})$, $(x_{i+2,0}, y_{i+2})$ is a subwalk of C_L in Figure 4.2. Similarly C_R consists of paths from $(x_{i+1,m(i+1)}, y_{i+1})$ to $(x_{i,m(i)}, y_i)$, $i = n-1, n-2, \ldots, 0$. C_L and the reverse of C_R are constructed during the scan of lines to form the triangles of $int(P)$.

The following lemma implies that the edges of E_I (see (4.2)) are locally optimal in any triangulation of the mesh vertices (including those on ∂P) which contains these edges.

Lemma 4.1 : If $e \in E_I$, then e is a Delaunay edge.

Proof : E_I can be partitioned into two disjoint sets of edges E_1 and E_2 where E_1 contains the horizontal edges and E_2 contains the edges which join vertices on two consecutive lines $y = y_i$ and $y = y_{i+1}$. First suppose $e \in E_1$. Let the vertices of edge e be $v_1 = (x_{i,j}, y_i)$ and $v_2 = (x_{i,j} + h, y_i)$. Let S be the circle of radius $h/2$ with centre at $(x_{i,j} + h/2, y_i)$ (see Figure 4.3a). Clearly no vertices of V_I can be in the interior of S. Mesh vertices on ∂P lie a distance $\geq h/\sqrt{2}$ from the centre of S by (4.1), so they are not in the interior of S. By Lemma 2.1, e is a Delaunay edge.

Now suppose $e \in E_2$. Let the vertices of edge e be $v_1 = (x_{i,j}, y_i)$ and $v_2 = (x_{i+1,k}, y_{i+1})$, and let $d = |x_{i,j} - x_{i+1,k}|$. Let S be the circle whose diameter is e. It is apparent from Figure 4.3b that no vertices of V_I can be in the interior of S if $d \leq h$. To establish this, we note that e may have been formed by either (4.2a) or (4.2c). In the first case, it is clear that $d \leq h$ as a consequence of the triangulation process in $int(P)$ (see Figure 4.2). In the second case, $d = a - b \leq h$ as discussed in the paragraph above (4.2). The radius of S is $\sqrt{h^2 + d^2}/2 \leq h/\sqrt{2}$. As above, mesh vertices on ∂P are not in the interior of S. By Lemma 2.1, e is a Delaunay edge. □

The pseudo-code for the generation of mesh vertices, triangles, and closed walk C in $int(P)$ is given in procedure INTTRIANG. In this procedure, a triangle is represented by a list of three vertex coordinates in counterclockwise order, the function $append(list1, list2)$ appends the elements of the second list at the end of the first list, and the function $reverse(list)$ reverses the elements of the list.

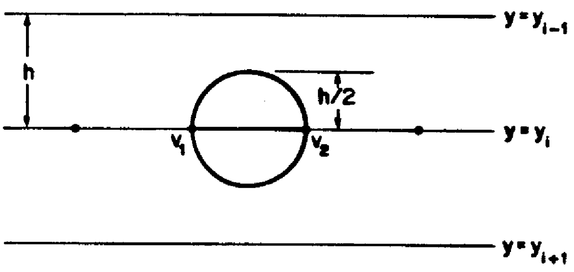

Figure 4.3a Edge v_1v_2 is in E_1

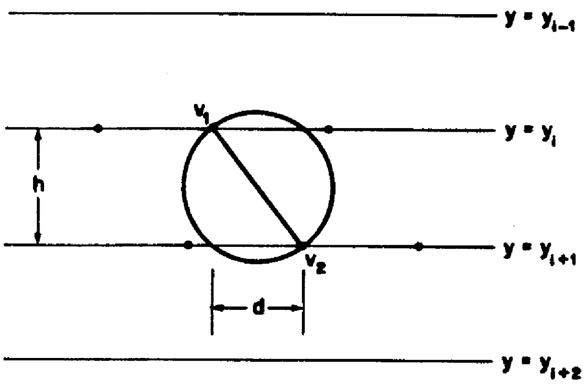

Figure 4.3b Edge v_1v_2 is in E_2

```
Procedure INTTRIANG($int(P), h, TI, nt, C, nc$);
# Input: convex polygon $int(P)$ with diameter parallel to y-axis and triangle size parameter $h$
# Output: list of triangles $TI$, number of triangles $nt$, closed walk $C$, number of edges $nc$ in $C$
    # Generate mesh vertices in $int(P)$
    $ymax$ := maximum y-coordinate of vertices of $int(P)$;
    $ymin$ := minimum y-coordinate of vertices of $int(P)$;
    $n := \mathrm{trunc}((ymax - ymin)/h)$;
    $dy := (ymax - ymin - n\,h)/2$;
    for $i := 0$ to $n$ do $y_i := ymax - dy - i\,h$;
    Scan down the left and right sides of $\partial int(P)$ to determine the points $(a_i, y_i)$ and $(b_i, y_i)$;
    for $i := 0$ to $n$ do
          $m(i) := \mathrm{trunc}((b_i - a_i)/h)$;
          $dx_i := (b_i - a_i - m(i)h)/2$;
          for $j := 0$ to $m(i)$ do $x_{i,j} := a_i + dx_i + j\,h$;
    $nt := 0$; $TI := [\,]$;
    if $n = 0$ then $nc := 0$; $C := [(x_{0,0}, y_0)]$; return;
    $C_L := [(x_{0,0}, y_0)]$; $C_R := [(x_{0,0}, y_0), (x_{0,1}, y_0), \ldots, (x_{0,m(0)}, y_0)]$;
    for $i := 0$ to $n-1$ do
          $a := \max(x_{i,0}, x_{i+1,0})$;
          $b := \min(x_{i,m(i)}, x_{i+1,m(i+1)})$;
          $l0$ := smallest integer such that $x_{i,l0} \geq a - h$;
          $l1$ := smallest integer such that $x_{i+1,l1} \geq a - h$;
          $r0$ := largest integer such that $x_{i,r0} \leq b + h$;
          $r1$ := largest integer such that $x_{i+1,r1} \leq b + h$;
          # Generate the triangles between $y = y_i$ and $y = y_{i+1}$
          if $l0 < r0$ or $l1 < r1$ then
             $j := l0$; $k := l1$;
             while $j < r0$ and $k < r1$ do
                   if $x_{i+1,k} \leq x_{i,j}$ then
                      $nt := nt + 1$; $TI(nt) := \Delta[(x_{i+1,k}, y_{i+1}), (x_{i+1,k+1}, y_{i+1}), (x_{i,j}, y_i)]$;
                      $k := k + 1$;
                   else
                      $nt := nt + 1$; $TI(nt) := \Delta[(x_{i,j+1}, y_i), (x_{i,j}, y_i), (x_{i+1,k}, y_{i+1})]$;
                      $j := j + 1$;
             if $j = r0$ then while $k < r1$ do
                      $nt := nt + 1$; $TI(nt) := \Delta[(x_{i+1,k}, y_{i+1}), (x_{i+1,k+1}, y_{i+1}), (x_{i,j}, y_i)]$;
                      $k := k + 1$;
             else while $j < r0$ do
                      $nt := nt + 1$; $TI(nt) := \Delta[(x_{i,j+1}, y_i), (x_{i,j}, y_i), (x_{i+1,k}, y_{i+1})]$;
                      $j := j + 1$;
          # Generate the subwalks of $C_L$ and $C_R$ between $y = y_i$ and $y = y_{i+1}$
          if $x_{i,l0} \leq x_{i+1,l1}$ then
             $C_L := \mathrm{append}(C_L, [(x_{i,1}, y_i), (x_{i,2}, y_i), \ldots, (x_{i,l0}, y_i), (x_{i+1,0}, y_{i+1})])$
          else $C_L := \mathrm{append}(C_L, [(x_{i+1,l1}, y_{i+1}), (x_{i+1,l1-1}, y_{i+1}), \ldots, (x_{i+1,0}, y_{i+1})])$;
          if $x_{i,r0} \leq x_{i+1,r1}$ then
             $C_R := \mathrm{append}(C_R, [(x_{i+1,r1}, y_{i+1}), (x_{i+1,r1+1}, y_{i+1}), \ldots, (x_{i+1,m(i+1)}, y_{i+1})])$
          else $C_R := \mathrm{append}(C_R, [(x_{i,m(i)-1}, y_i), (x_{i,m(i)-2}, y_i), \ldots, (x_{i,r0}, y_i), (x_{i+1,m(i+1)}, y_{i+1})])$;
    $C_L := \mathrm{append}(C_L, [(x_{n,1}, y_n), (x_{n,2}, y_n), \ldots, (x_{n,m(n)-1}, y_n)])$;
    $C := \mathrm{append}(C_L, \mathrm{reverse}(C_R))$;
    $nc$ := number of edges in $C$;
    return;
```

5. Triangulation near the boundary of P

Mesh vertices are generated on the edges of ∂P as follows. Let e_i be an edge of ∂P with triangle size parameter h_i (see Section 1). Mesh vertices are generated on e_i at an equal spacing of $\bar{h}_i$, the nearest length to h_i which is an integral submultiple of $|e_i|$ (the length of e_i). $\bar{h}_i$ is computed as follows:

```
$k := \mathrm{trunc}(|e_i|/h_i)$;
$r := |e_i|/h_i - k$;                                        (5.1)
if $r > k/(2k+1)$ then $k := k+1$;
$\bar{h}_i := |e_i|/k$;
```

If h_i satisfies (1.1), then the restriction of $\bar{h}_i$ compared to h is

$$\min(|e_i|, \sqrt{2}h/3) \leq \bar{h}_i \leq 4\sqrt{2}h/3 \tag{5.2}$$

from (5.1).

In this section we describe a procedure for triangulating the strip, A, between ∂P and C in the case of at least one vertex in the interior of P. (The case in which $int(P)$ is degenerate and there are no vertices in the interior of P will be discussed later in this section.) We believe that the spacing of mesh vertices on ∂P restricted by (5.2) is small enough for the following procedure to generate a valid triangulation in A. However, we have been unable to prove this. On the other hand, as Figure 5.4 below shows, if the spacing of vertices on ∂P is too large relative to h then an invalid triangulation can be formed. We show how the procedure can be modified to produce a valid triangulation in this case. In Section 7 we describe how to modify the triangulation of A so that the total triangulation of P is a Delaunay triangulation.

If there are at least two vertices in $int(P)$, then let the closed walk C determined in the preceding section be represented by the list of vertices $C = [v_0, v_1, \ldots, v_{nc}]$ where $v_0 = v_{nc} = (x_{0,0}, y_0)$ and $nc \geq 2$ is the number of edges $v_j v_{j+1}$ in C; otherwise let $C = [v_0]$ and $nc = 0$. Let the counterclockwise cycle of edges on ∂P be represented by the list of mesh vertices $B = [u_0, u_1, \ldots, u_{nb}]$ where $u_0 = u_{nb}$, $nb \geq 3$ is the number of edges $u_i u_{i+1}$ in B, and the numbering of the u_i is done as follows (see Figure 5.1). If there are at least two vertices in $int(P)$ then we number the u_i so that u_0 is closest to v_0 among the vertices on ∂P with y-

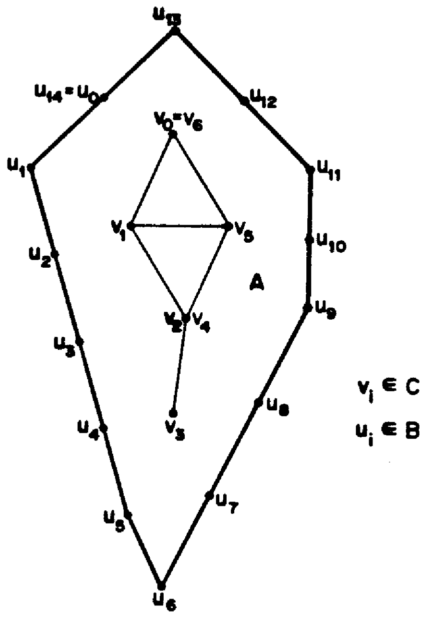

Figure 5.1 Illustration of closed walk C, cycle B, and strip A

coordinate greater than y_0 (note that edge u_0v_0 is entirely in A). Otherwise we number the u_i so that u_0 is the vertex on ∂P which is closest to v_0. In the former case the selection of u_0 is a heuristic for finding a Delaunay edge u_0v_0. In the latter case,

Lemma 5.1 : If there is only one vertex v_0 in $int(P)$ and u_0 is the vertex on ∂P closest to v_0 then u_0v_0 is a Delaunay edge.

Proof : Let S be the circle of radius $|u_0v_0|$ centred at v_0. S contains no vertices in its interior since u_0 is the closest vertex to v_0. Let S' be the circle of radius $|u_0v_0|/2$ whose diameter is u_0v_0. S' is contained inside S therefore S' contains no vertices in its interior. By Lemma 2.1, u_0v_0 is a Delaunay edge. □

We now describe how to generate edges of the type $u_i v_j$ and triangles of the types $\Delta u_i u_{i+1} v_j$ and $\Delta v_{j+1} v_j u_i$ in the strip A by 'merging' B and C and 'zigzagging' counterclockwise around A starting and ending at edge $u_0 v_0$. Note that A is to the left of B and to the right of C so that for a valid triangulation of A (see Section 2), CCW triangles $\Delta u_i u_{i+1} v_j$ and $\Delta v_{j+1} v_j u_i$ must be generated. Suppose $u_i v_j$ is the last edge generated and $i < nb$ or $j < nc$ (initially $i = 0$ and $j = 0$). The direction of edge $u_i v_j$ is from u_i to v_j in the last triangle and its direction in the next triangle will be from v_j to u_i. If $i = nb$ then the next edge and triangle generated are $u_{nb} v_{j+1}$ and $\Delta v_{j+1} v_j u_{nb}$. If $j = nc$ then the next edge and triangle generated are $u_{i+1} v_{nc}$ and $\Delta u_i u_{i+1} v_{nc}$. Otherwise either edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ or edge $u_{i+1} v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are generated next based on the following test.

Consider the quadrilateral $Q = u_i u_{i+1} v_{j+1} v_j$. v_j and v_{j+1} are to the left of the directed line from u_i to u_{i+1}, but the positions of u_i and u_{i+1} relative to the directed line, $l(v_j v_{j+1})$, from v_j to v_{j+1} may be one of the following four cases (see Figure 5.2):

(a) u_i and u_{i+1} are both to the right of $l(v_j v_{j+1})$,

(b) u_i (u_{i+1}) is to the left of or on (right of) $l(v_j v_{j+1})$,

(c) u_i (u_{i+1}) is to the right of (left of or on) $l(v_j v_{j+1})$, (5.3)

(d) u_i and u_{i+1} are both to the left of or on $l(v_j v_{j+1})$.

In case (a), Q is a strictly convex quadrilateral and if the circle through the vertices u_i, u_{i+1}, v_j does not contain vertex v_{j+1} in its interior then edge $u_{i+1} v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen else edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ are chosen. In case (b), Q is a nonconvex quadrilateral and edge $u_{i+1} v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen. In case (c), Q is a nonconvex quadrilateral and edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ are chosen. In case (d), Q is a nonsimple quadrilateral and edge $u_{i+1} v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen; the other triangle $\Delta v_{j+1} v_j u_i$ is oriented in the wrong (clockwise) direction (see (2.1a)). In cases (a), (b), and (c), the chosen edge is locally optimal in Q (see Section 2).

Let $T(P)$ be the collection of triangles formed in A and $int(P)$ as described above and in

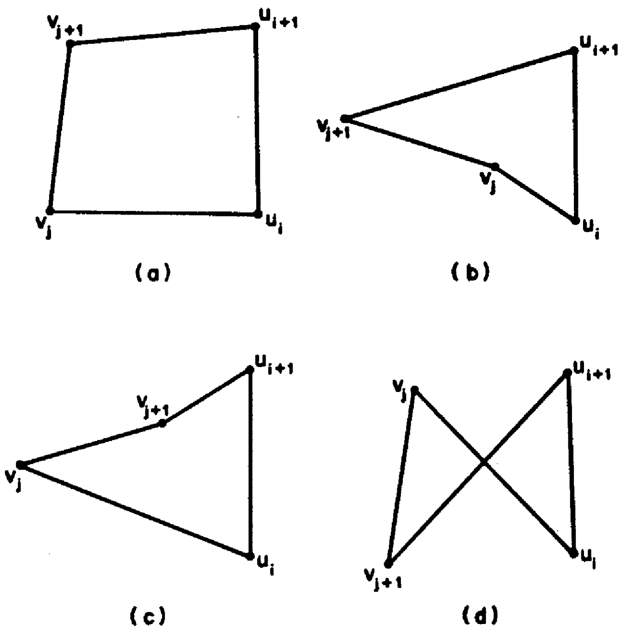

Figure 5.2 Four cases of quadrilateral Q

Section 4. $T(P)$ is illustrated in Figure 5.3. It is not clear that the triangles in $T(P)$ do not overlap, i.e. that they produce a valid triangulation. If $|u_i u_{i+1}|$ is too large relative to h for some i, then $T(P)$ can be an invalid triangulation (e.g. see Figure 5.4). After discussing the case of no interior vertices, we indicate how this merge procedure can be modified to generate a valid triangulation, $VT(P)$, even if the $|u_i u_{i+1}|$ are large relative to h.

Now we consider the case of no vertices in the interior of P. Let v_b and v_t be the endpoints of a diameter of P and suppose the coordinate system is rotated so that line segment $v_b v_t$ is parallel to the y-axis with $y(v_t) > y(v_b)$ (as was done for $int(P)$ in Section 4). Let $B = [u_0, u_1, \ldots, u_{mb}, \ldots, u_{nb-1}, u_{nb}]$ be the counterclockwise cycle of vertices on ∂P where

$u_0 = u_{nb} = v_t$ and $u_{mb} = v_b$. B can be split into two chains $B_L = [u_0, u_1, \ldots, u_{mb}]$ and $B_R = [u_{nb}, u_{nb-1}, \ldots, u_{mb}] = [v_0, v_1, \ldots, v_{mc}]$ on the left and right sides of ∂P, respectively, such that the y-coordinates of the vertices strictly decrease from $y(v_t)$ to $y(v_b)$. Note that mb and mc are both at least one. B_L and B_R can be merged to produce a valid triangulation, $VT(P)$, in P in a way similar to the procedure described above (see Figure 5.5), and $VT(P)$ can be converted into a Delaunay triangulation in a way similar to that described in Section 7. The validity of $VT(P)$ is discussed in Section 6.

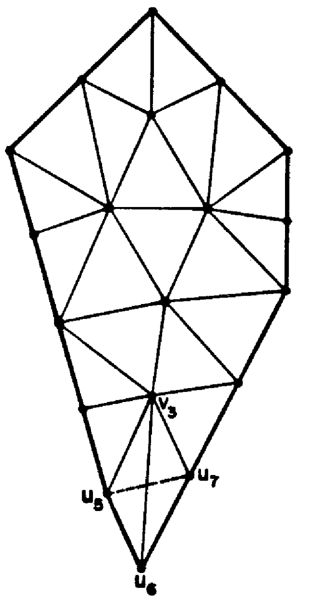

Figure 5.3 Illustration of $T(P)$ and $DT(P)$; u_6v_3 is in $T(P)$; u_5u_7 is in $DT(P)$

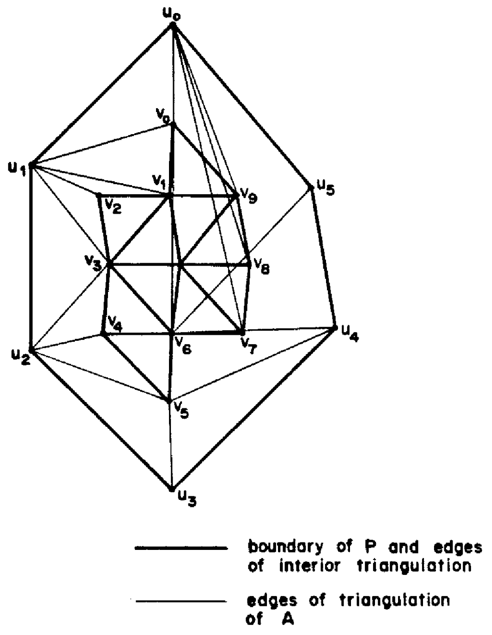

Figure 5.4 Illustration of invalid triangulation $T(P)$; note overlap by triangles $\Delta v_6 v_5 u_4$ and $\Delta v_9 v_8 u_0$

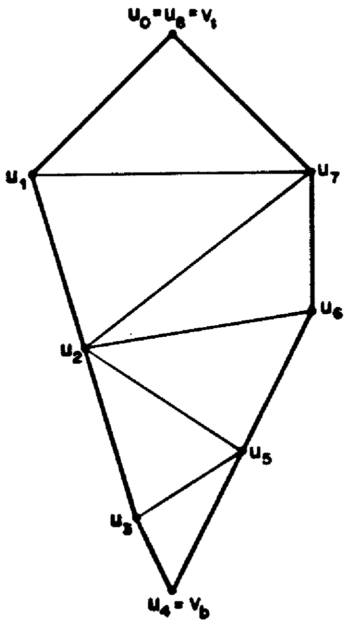

Figure 5.5 Illustration of chains B_L, B_R and $VT(P)$ in the case of no interior vertices

The pseudo-code for the merge of B and C or B_L and B_R to generate triangles in the strip A or polygon P, respectively, is given in procedure MERGE. The list of triangles, TM, produced by this procedure is modified by the procedure of Section 7 and then, in the case of at least one vertex in the interior of P, appended to the list of triangles, TI, produced by procedure INTTRIANG.

```
Procedure MERGE($inter, L, nl, M, nm, TM, EM$);
# Input: if $inter$ = true then $L = B$, $nl = nb$, $M = C$, $nm = nc$
#             else $L = B_L$, $nl = mb$, $M = B_R$, $nm = mc$
# Output: list of triangles $TM$ and list of edges $EM$;
#            if $inter$ = true then $TM$ and $EM$ each contain $nb + nc$ elements
#               else $TM$ and $EM$ each contain $mb + mc - 2$ elements
  $nll := nl$; $nmm := nm$;
  $nt := 0$;
  if not $inter$ then
      $nt := nt + 1$;
      $TM(nt) := \Delta u_0 u_1 v_1$; $EM(nt) := u_1 v_1$;
      if $nl + nm = 3$ then return;
      $nl := nl - 1$; $nm := nm - 1$;
      $i := 1$; $j := 1$;
  else $i := 0$; $j := 0$;
  while $i < nl$ and $j < nm$ do
      if ($u_i$ and $u_{i+1}$ are both to the left of or on $l(v_j v_{j+1})$) or
         ($v_{j+1}$ is not in the circle through $u_i$, $u_{i+1}$, $v_j$) then
         $nt := nt + 1$;
         $TM(nt) := \Delta u_i u_{i+1} v_j$; $EM(nt) := u_{i+1} v_j$;
         $i := i + 1$;
      else
         $nt := nt + 1$;
         $TM(nt) := \Delta v_{j+1} v_j u_i$; $EM(nt) := u_i v_{j+1}$;
         $j := j + 1$;
  if $i < nl$ then
      if not $inter$ and $j = nm$ then $nl := nl + 1$;
      while $i < nl$ do
         $nt := nt + 1$;
         $TM(nt) := \Delta u_i u_{i+1} v_j$; $EM(nt) := u_{i+1} v_j$;
         $i := i + 1$;
  else  # $j < nm$
      if not $inter$ and $i = nl$ then $nm := nm + 1$;
      while $j < nm$ do
         $nt := nt + 1$;
         $TM(nt) := \Delta v_{j+1} v_j u_i$; $EM(nt) := u_i v_{j+1}$;
         $j := j + 1$;
  $nl := nll$; $nm := nmm$;
  return;
```

As mentioned above, procedure MERGE can fail to produce a valid triangulation of P in the case of at least one interior vertex if the spacing of the u_i on ∂P is too large relative to h. In Section 6, we examine procedure MERGE to see how it can be modified to always produce a valid triangulation. We will show that an invalid triangulation $T(P)$ is due to an overlapping triangle $\Delta v_j v_{j-1} u_i$ in which v_{j+1} is to the right of $l(v_{j-1}v_j)$ and u_i is to the left of or on $l(v_j v_{j+1})$. Intuitively, we can think of this as resulting from there being too few vertices in B for the number of vertices in C. The modification consists of generating CCW triangle $\Delta v_{j+1} v_j v_{j-1}$ instead of $\Delta v_j v_{j-1} u_i$, replacing subwalk $v_{j-1}v_j v_{j+1}$ of C with edge $v_{j-1}v_{j+1}$, and restarting the merge of B and the new shortened C from edge $u_r v_{j-1}$ where $\Delta v_{j-1} v_{j-2} u_r$ is formed by the merge procedure. The pseudo-code for the modification is given in procedure MMERGE. Let $VT(P)$ be the triangulation produced by procedures INTTRIANG and MMERGE. Procedure MMERGE and the validity of $VT(P)$ will be discussed in Section 6. In procedure DELTRIANG of Section 8, procedure MMERGE is used to produce a valid triangulation of A in the case of at least one interior vertex, and procedure MERGE is used to produce a valid triangulation of P in the case of no interior vertices.

```
Procedure MMERGE($B, nb, C, nc, TM, EM, ne$);
# Input: boundary cycle $B$, closed walk $C$, number of edges $nb$ and $nc$ in $B$ and $C$
# Output: list of triangles $TM$ and edges $EM$ in $A$, each containing $nb+nc$ elements,
#       and the number of edges $ne$ of type $v_j v_{j+m}$, $m \geq 2$, generated in $A$.
#       The edges of type $v_j v_{j+m}$ are stored at the end of $EM$ in the reverse order
#       that they are generated. The edges of type $u_i v_j$ are stored at the front of $EM$
#       in the order that they are generated. The triangles are stored in a similar way.
# The working arrays $s$, $p$, $r$, and $n$ are used as follows:
#       $v_{s(j)}$ and $v_{p(j)}$ are the successor and predecessor vertices of $v_j$ in $C$.
#       $s$ and $p$ are updated when a vertex is removed from $C$.
#       $u_{r(j)}$ is the vertex of $B$ in the triangle $\Delta v_{s(j)} v_j u_{r(j)}$ and
#       $n(j)$ is the index of this triangle in list $TM$.
#       $r$ and $n$ are used to determine how far to backtrack the merge of
#       $B$ and $C$ when a vertex is removed from $C$.
  for $j := 0$ to $nc-1$ do
    $s(j) := j+1$; $p(j+1) := j$; $r(j) := 0$; $n(j) := 0$;
  $nt := 0$; $ne := 0$;
  $i := 0$; $j := 0$;
  while $i \leq nb$ and $j \leq nc$ and $i+j < nb+nc$ do
    if $(j = nc)$ or ($u_i$ and $u_{i+1}$ are both to the left of or on $l(v_j v_{s(j)})$)
      or ($v_{s(j)}$ is not in the circle through $u_i$, $u_{i+1}$, $v_j$) then
      $nt := nt+1$;
      $TM(nt) := \Delta u_i u_{i+1} v_j$; $EM(nt) := u_{i+1} v_j$;
      $i := i+1$;
    else
      if $(s(j) = nc)$ or ($v_{s(s(j))}$ is to the left of or on $l(v_j v_{s(j)})$)
        or ($u_i$ is to the right of or on $l(v_{s(j)} v_{s(s(j))})$) then
        $nt := nt+1$;
        $TM(nt) := \Delta v_{s(j)} v_j u_i$; $EM(nt) := u_i v_{s(j)}$;
        $r(j) := i$; $n(j) := nt$;
        $j := s(j)$;
      else
        $flag :=$ false;
        repeat   # remove $v_{s(j)}$ from $C$
          $TM(nb+nc-ne) := \Delta v_{s(s(j))} v_{s(j)} v_j$;
          $EM(nb+nc-ne) := v_j v_{s(s(j))}$;
          $ne := ne+1$;
          $s(j) := s(s(j))$; $p(s(j)) := j$;
          if $j = 0$ then $i := 0$; $nt := 0$;
            else $i := r(p(j))$; $nt := n(p(j))$;
          if $(j > 0)$ and ($v_{s(j)}$ is to the right of $l(v_{p(j)} v_j)$)
            and ($u_i$ is to the left of or on $l(v_j v_{s(j)})$) then
            $j := p(j)$
          else $flag :=$ true;
        until $flag$;
  return;
```

6. Validity of triangulation VT(P)

$VT(P)$ is a valid triangulation of P if the triangles form a 'tiling' of P without overlaps or gaps. We will show that $VT(P)$ is valid for the three cases of (i) no interior vertices, (ii) one interior vertex, and (iii) two or more interior vertices. In case (ii), $VT(P)$ (or $T(P)$) is valid since the triangles $\Delta u_i u_{i+1} v_0$, $0 \leq i \leq nb-1$, are formed and they clearly tile P without overlaps or gaps.

In case (i), each edge added to $VT(P)$ in procedure MERGE subdivides an untriangulated convex subregion of P into two convex subregions - a triangle and a smaller untriangulated subregion. For example, in Figure 5.5, edge $u_1 u_7$ subdivides P into triangle $\Delta u_0 u_1 u_7$ and convex subpolygon $u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_1$. If all the untriangulated convex subregions are nondegenerate (i.e. not a line segment) then the triangles of $VT(P)$ clearly tile P without overlaps or gaps so $VT(P)$ is valid. An untriangulated convex subregion can be degenerate if one side of ∂P contains only two vertices and the other side contains three or more collinear vertices at the bottom. In this case degenerate triangles (i.e. with three collinear vertices) are formed. For example, in Figure 6.1a, triangles $\Delta u_0 u_1 u_3$ and $\Delta u_2 u_3 u_1$ are formed by procedure MERGE and the latter triangle is degenerate. If the collinear vertices at the bottom are perturbed slightly so that no three vertices are collinear and the polygon remains convex then all the untriangulated subregions are nondegenerate and the triangulation is valid (e.g. see Figure 6.1b). Therefore we still consider the triangulations with the degenerate triangles to be 'valid'. The procedure described in the next section will convert these triangulations into Delaunay triangulations.

In the rest of this section we will consider the case of two or more interior vertices. We will show that $VT(P)$ is valid by showing how $T(P)$ (the triangulation produced by procedures INTTRIANG and MERGE) can sometimes be invalid and how a modification can be made to procedure MERGE to always produce a valid triangulation. In Section 2, we mentioned that a triangulation of a region is valid if it satisfies the four conditions of (2.1).

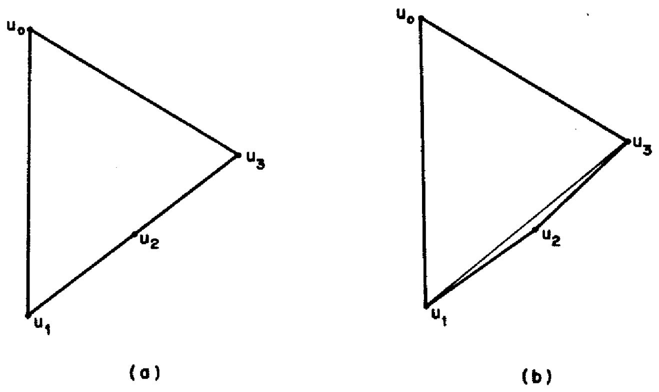

Figure 6.1 (a) Example where degenerate triangle $\Delta u_2 u_3 u_1$ is formed by procedure MERGE
(b) Perturbation of chain $u_1 u_2 u_3$ to obtain valid triangulation

Lemma 6.1 :

(a) $T(P)$ satisfies conditions (b), (c), and (d) of (2.1).

(b) If $\Delta v_{j+1} v_j u_i$ is produced by procedure MERGE and $i < nb$, then $\Delta v_{j+1} v_j u_i$ is CCW.

(c) If all triangles of type $\Delta v_{j+1} v_j u_i$ produced by procedure MERGE are CCW, then $T(P)$ is a valid triangulation of P.

(d) $T(P)$ is not a valid triangulation of P if and only if there is a CW triangle of type $\Delta v_{j+1} v_j u_{nb}$ produced by procedure MERGE.

Proof : The boundary edges of P are the edges $u_i u_{i+1}$ of B. Conditions (c) and (d) of (2.1) are clearly, satisfied by $T(P)$ since the triangles formed in A by procedure MERGE are of the types $\Delta u_i u_{i+1} v_j$ and $\Delta v_{j+1} v_j u_i$. We now show that condition (2.1b) is satisfied by all edges of $T(P)$. The edges of E_I (see (4.2)) which are not edges of C clearly satisfy condition (2.1b) (see

Figure 4.2). An edge $e = v_j v_{j+1}$ of C occurs either once or twice in C. In the latter case, $v_j = v_{k+1}$ and $v_{j+1} = v_k$ for some $k \neq j$ and e occurs in opposite directions in the triangles $\Delta v_{j+1} v_j u_i$ and $\Delta v_{k+1} v_k u_l$ formed in A. In the former case, e occurs in opposite directions in the triangles $\Delta v_{j+1} v_j u_i$ in A and $\Delta v_j v_{j+1} w$ in $int(P)$. A boundary edge $u_i u_{i+1}$ of B occurs only once in the triangle $\Delta u_i u_{i+1} v_j$ formed in A. The other edges of $T(P)$ are of the type $u_i v_j$, $u_i \in B$ and $v_j \in C$, formed in A. One of the triangles with edge $u_i v_j$ is either $\Delta u_{i-1} u_i v_j$ or $\Delta v_j v_{j-1} u_i$; the other triangle with edge $u_i v_j$ is either $\Delta u_i u_{i+1} v_j$ or $\Delta v_{j+1} v_j u_i$, where the indices of u_i and v_j are taken modulo nb and nc respectively. In all four possibilities, edge $u_i v_j$ occurs in opposite directions in the two triangles. Therefore condition (2.1b) is satisfied by all edges of $T(P)$.

Now we determine when condition (2.1a) is satisfied or not satisfied. The triangles generated by procedure INTTRIANG are all CCW. The triangles generated by procedure MERGE are of the types $\Delta u_i u_{i+1} v_j$ or $\Delta v_{j+1} v_j u_i$. $\Delta u_i u_{i+1} v_j$ is a CCW triangle since v_j is to the left of directed edge $u_i u_{i+1}$. $\Delta v_{j+1} v_j u_i$ is a CCW triangle if it is formed from case (a) or (c) of (5.3) (see Figure 5.2). It cannot be formed from case (b) or (d) of (5.3). So $\Delta v_{j+1} v_j u_i$ may be a CW triangle only when there are no more quadrilaterals $u_i u_{i+1} v_{j+1} v_j$ and $i = nb$. By condition (2.1a), $T(P)$ is a valid triangulation if the triangles of type $\Delta v_{j+1} v_j u_{nb}$ are all CCW.

If $T(P)$ is not a valid triangulation then condition (2.1a) is not satisfied and there is a CW triangle of the type $\Delta v_{j+1} v_j u_{nb}$. To show that the converse is also true, suppose $\Delta v_{j+1} v_j u_{nb}$ is a CW triangle produced by procedure MERGE. Edge $v_j v_{j+1}$ occurs either once or twice in C. In the former case, CCW triangle $\Delta v_j v_{j+1} w$ is formed by procedure INTTRIANG, and $\Delta v_{j+1} v_j u_{nb}$ and $\Delta v_j v_{j+1} w$ overlap. In the latter case, $v_j = v_{k+1}$ and $v_{j+1} = v_k$ for some $k \neq j$ and $\Delta v_{k+1} v_k u_l$ is formed by procedure MERGE; either $\Delta v_{k+1} v_k u_l$ is CCW if $l < nb$ or $\Delta v_{k+1} v_k u_l \equiv \Delta v_{j+1} v_j u_{nb}$ if $l = nb$; in both subcases $\Delta v_{j+1} v_j u_{nb}$ and $\Delta v_{k+1} v_k u_l$ overlap. Therefore $T(P)$ is not a valid triangulation. □

We now examine the triangles produced in A by procedure MERGE to see how CW triangles $\Delta v_{j+1}v_ju_{nb}$ can be formed and how the procedure can be modified to produce a valid triangulation in this case. Since the triangles of type $\Delta u_iu_{i+1}v_j$ are always CCW, we concentrate on the triangles of type $\Delta v_{j+1}v_ju_i$. Let $\Delta v_{j+1}v_ju_{r(j)}$, $0 \le j \le nc-1$, be the triangles of type $\Delta v_{j+1}v_ju_i$ produced by procedure MERGE, where $0 \le r(0) \le r(1) \le \cdots \le r(nc-1) \le nb$.

We first make some definitions which will be used in the following lemmas. We define v_j to be a reflex vertex of C if v_{j+1} is to the right of $l(v_{j-1}v_j)$ and define v_j to be a convex vertex of C otherwise. Note that v_0 is a convex vertex of C since it is the leftmost interior mesh vertex on the top line $y = y_0$. Let v_{mid} be the rightmost interior mesh vertex on the bottom line $y = y_n$. v_{mid} is also a convex vertex of C. Let C_L be the subwalk of C from v_0 to v_{mid} and C_R be the subwalk of C from v_{mid} to v_{nc}. Let $[w,z]$ denote the 'interval' of ∂P going counterclockwise from point w to point z inclusive. A parenthesis will be used in place of the bracket if the endpoint w or z is not included, e.g. (w,z), $(w,z]$, $[w,z)$. Let w_j and z_j be the intersections of line $l(v_jv_{j+1})$ with ∂P where w_j is the intersection closer to v_j and z_j is the intersection closer to v_{j+1}. (w_j, z_j) is the interval of ∂P to the right of $l(v_jv_{j+1})$. Define $I_j = (w_j, z_j)$ if u_0 is not in (w_j, z_j), $I_j = [u_0, z_j)$ if u_0 is in (w_j, z_j) and $v_jv_{j+1} \in C_L$, and $I_j = (w_j, u_{nb}]$ if $u_0 = u_{nb}$ is in (w_j, z_j) and $v_jv_{j+1} \in C_R$. Recall that $u_{r(j)}$ is a vertex of the triangle $\Delta v_{j+1}v_ju_{r(j)}$ produced by procedure MERGE.

Lemma 6.2 : If $u_{r(j)}$ is not in I_j for some j, then $T(P)$ is not a valid triangulation.

Proof : If $u_{r(j)}$ is not in (w_j, z_j) then $\Delta v_{j+1}v_ju_{r(j)}$ is CW so $r(j) = nb$ and $T(P)$ is not valid by Lemma 6.1. Suppose u_0 is in (w_j, z_j) so that $I_j \ne (w_j, z_j)$ and suppose $u_{r(j)}$ is in (w_j, z_j) but not in I_j. First suppose $v_jv_{j+1} \in C_L$. Then $u_{r(j)}$ is in (w_j, u_{nb}) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ must intersect the walk $u_0v_0v_1 \cdots v_j$ (see Figure 6.2). Now suppose $v_jv_{j+1} \in C_R$. Then $u_{r(j)}$ is in (u_0, z_j) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ must intersect the walk $v_{j+1}v_{j+2} \cdots v_{nc}u_{nb}$. In both cases, $\Delta v_{j+1}v_ju_{r(j)}$ overlaps other triangles of $T(P)$ so $T(P)$ is not valid. □

Lemma 6.3 : If v_j is a reflex vertex of C and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ to the left of or on $l(v_jv_{j+1})$, then $\Delta v_jv_{j-1}u_{r(j-1)}$ is an overlapping triangle and $T(P)$ is not a valid tri-

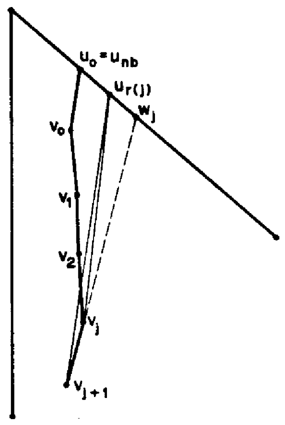

Figure 6.2 $\Delta v_{j+1} v_j u_{r(j)}$ intersects walk $u_0 v_0 v_1 \cdots v_j$

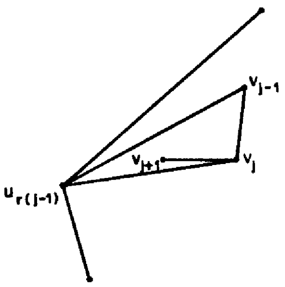

Figure 6.3 $\Delta v_j v_{j-1} u_{r(j-1)}$ intersects edge $v_j v_{j+1}$

angulation.

Proof : If $u_{r(j-1)}$ is to the left of or on $l(v_jv_{j+1})$ then part of edge v_jv_{j+1} lies in the interior or on the boundary of $\Delta v_jv_{j-1}u_{r(j-1)}$ (see Figure 6.3). Therefore $\Delta v_jv_{j-1}u_{r(j-1)}$ and $\Delta v_{j+1}v_ju_{r(j)}$ overlap and $T(P)$ is not a valid triangulation. □

Lemma 6.4 :

(a) If v_j is a reflex vertex of C and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1} and to the right of $l(v_jv_{j+1})$ then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.

(b) If v_j is a convex vertex of C, $0<j<nc$, and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1} then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.

(c) If $j=0$ then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.

Proof : Define $u_{t(j)}$ to be the vertex of largest index in I_j.

(a) Let u_l, $r(j-1)\le l\le nb$, be the vertex of largest index to the right of $l(v_jv_{j+1})$ such that u_i, $i=r(j-1),\ldots,l$, are to the right of $l(v_jv_{j+1})$. If $l>r(j-1)$ then quadrilaterals $u_iu_{i+1}v_{j+1}v_j$, $i=r(j-1),\ldots,l-1$, are case (a) of (5.3) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ may be formed from one of these quadrilaterals. If not, then CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$, $r(j)=l$, is formed because either $l=nb$ or quadrilateral $u_lu_{l+1}v_{j+1}v_j$ is case (c) of (5.3). v_j is a reflex vertex and $u_{r(j-1)}$ is in I_{j-1} and to the right of $l(v_jv_{j+1})$ imply that $u_{r(j-1)}$ is in I_j and $r(j-1)\le r(j)\le t(j)=l$. Therefore $u_{r(j)}$ is in I_j.

(b) v_j is a convex vertex and u_0 is chosen so that $y(u_0)>y(v_0)$ imply that $t(j)\ge t(j-1)$. $u_{r(j-1)}$ is in I_{j-1} implies that $r(j-1)\le t(j-1)$. Let u_k, $r(j-1)\le k\le t(j)$, be the vertex of smallest index to the right of $l(v_jv_{j+1})$ such that u_i, $i=k,\ldots,t(j)$, are to the right of $l(v_jv_{j+1})$. If $k>r(j-1)$ then $\Delta u_iu_{i+1}v_j$, $i=r(j-1),\ldots,k-1$ are formed from case (b) or (d) of (5.3). If $k<t(j)$ then quadrilaterals $u_iu_{i+1}v_{j+1}v_j$, $i=k,\ldots,t(j)-1$, are case (a) of (5.3) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ may be formed from one of these quadrilaterals. If not, then CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$, $r(j)=t(j)$, is formed because either $t(j)=nb$ or quadrilateral

$u_{t(j)}u_{t(j)+1}v_{j+1}v_j$ is case (c) of (5.3). $u_{r(j-1)}$ is in I_{j-1} and $k \le r(j) \le t(j)$ imply that $u_{r(j)}$ is in I_j.

(c) Let u_k, $0 \le k \le t(j)$, be the vertex of smallest index to the right of $l(v_jv_{j+1})$ such that u_i, $i = k, \ldots, t(j)$, are to the right of $l(v_jv_{j+1})$. The rest of the proof is the same as (b) with $r(j-1)$ replaced by 0 (also omit '$u_{r(j-1)}$ is in I_{j-1}' from the last sentence). □

Lemma 6.5 : If all vertices v_j of C are convex then $T(P)$ is a valid triangulation.

Proof : By parts (c) and (b) of Lemma 6.4 and induction, $\Delta v_{j+1}v_ju_{r(j)}$, $j = 0, 1, \ldots, nc-1$, are CCW triangles. By Lemma 6.1, $T(P)$ is a valid triangulation. □

Lemma 6.6 : If for all reflex vertices v_j of C, $u_{r(j-1)}$ is to the right of $l(v_jv_{j+1})$, then $T(P)$ is a valid triangulation.

Proof : We will show by induction that for all j, $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j. By Lemma 6.4(c), $\Delta v_1v_0u_{r(0)}$ is CCW with $u_{r(0)}$ in I_0. Suppose $0 < j < nc$ and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1}. If v_j is a convex vertex then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j by Lemma 6.4(b). If v_j is a reflex vertex then it is given that $u_{r(j-1)}$ is to the right of $l(v_jv_{j+1})$ so $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j by Lemma 6.4(a). Therefore $\Delta v_{j+1}v_ju_{r(j)}$, $j = 0, 1, \ldots, nc-1$, are CCW triangles. By Lemma 6.1, $T(P)$ is a valid triangulation. □

Conjecture 6.1 : If $|u_iu_{i+1}|$ is sufficiently small relative to h for all i (e.g. as produced by the spacing from (1.1), (5.1), (5.2)), then $T(P)$ is a valid triangulation.

Rationale : If $|u_iu_{i+1}|$ is sufficiently small relative to h for all i then for all reflex vertices v_j of C there exists a vertex u_k in I_{j-1} such that u_k is to the right of $l(v_jv_{j+1})$; in order for CCW triangle $\Delta v_jv_{j-1}u_k$ to be generated by procedure MERGE, either $u_k = u_{nb}$ or u_{k+1} must not be in the interior of the circle through the vertices v_{j-1}, v_j, u_k. It appears from the way that C is constructed from $int(P)$ that such a vertex u_k exists for all reflex vertices v_j and the hypothesis of Lemma 6.6 is satisfied, so $T(P)$ is a valid triangulation. □

If $|u_iu_{i+1}|$ is too large relative to h for some i, then procedure MERGE can produce an invalid triangulation $T(P)$ (see Figure 5.4). Lemmas 6.3, 6.4, 6.5, and 6.6 suggest how procedure

MERGE can be modified to produce a valid triangulation in this case. By Lemma 6.6, if $T(P)$ is not valid then there exists a reflex vertex v_j such that $u_{r(j-1)}$ is to the left of or on $l(v_j v_{j+1})$. Suppose v_j is the reflex vertex of smallest index such that $u_{r(j-1)}$ is to the left of or on $l(v_j v_{j+1})$ (e.g. in Figure 5.4, $v_j = v_6$). Then $\Delta v_{k+1} v_k u_{r(k)}$, $k = 0, \ldots, j-1$, are CCW with $u_{r(k)}$ in I_k by Lemma 6.4 but $\Delta v_j v_{j-1} u_{r(j-1)}$ is an overlapping triangle by Lemma 6.3. Therefore instead of generating $\Delta v_j v_{j-1} u_{r(j-1)}$, generate CCW triangle $\Delta v_{j+1} v_j v_{j-1}$ which does not overlap any triangles in $C \cup$ (interior of C), replace subwalk $v_{j-1} v_j v_{j+1}$ of C with edge $v_{j-1} v_{j+1}$, and rerun procedure MERGE with B and the new shortened C which still lies in $int(P)$. If this process is repeated enough times when overlapping triangles $\Delta v_j v_{j-1} u_{r(j-1)}$ are detected, then the resulting triangulation is valid because either the hypothesis of Lemma 6.6 is satisfied or $C \cup$ (interior of C) eventually becomes a convex set so the hypothesis of Lemma 6.5 is satisfied. Procedure MMERGE of Section 5 contains this modification to procedure MERGE in which the merge is restarted from edge $u_{r(j-2)} v_{j-1}$ instead of edge $u_0 v_0$ (if $j \geq 2$) when it is determined that a CCW triangle $\Delta v_{j+1} v_j v_{j-1}$ must be added to the triangulation, since the triangles in A between $u_0 v_0$ and $u_{r(j-2)} v_{j-1}$ would remain the same. We have shown in the above discussion that

Theorem 6.1 : The triangulation, $VT(P)$, produced by procedures INTTRIANG and MMERGE is a valid triangulation.

7. Converting VT(P) into a Delaunay triangulation

We describe a procedure for converting $VT(P)$ into a Delaunay triangulation, $DT(P)$. We discuss only the case of at least one interior vertex, since the case of no interior vertices is similar. From Lemma 4.1, the edges of the interior triangulation and C (as defined in Section 4) are locally optimal in any triangulation of the mesh vertices which contains these edges. From Theorem 2.1, $VT(P)$ can be converted into a Delaunay triangulation by applying LOP to the internal edges in the triangulation of A until they are all locally optimal.

Procedure MMERGE produces $nt = nb + nc$ triangles, $TM(1), TM(2), \ldots, TM(nt)$, and nt edges, $EM(1), EM(2), \ldots, EM(nt)$, in A, where $ne \geq 0$ edges are of the type $v_j v_{j+m}$, $m \geq 2$, and the triangles and edges are ordered as follows. Edges $EM(k)$, $k=1,\ldots,nt-ne$, are of the type u_iv_j and are in the order that they are generated. Edges $EM(k)$, $k=nt-ne+1,\ldots,nt$, are of the type v_jv_{j+m}, $m\geq 2$, and are in the reverse order that they are generated (these edges are added to ensure the validity of the triangulation of A). For $k=1,\ldots,nt-ne-1$, $EM(k)$ is the common edge of $TM(k)$ and $TM(k+1)$. $EM(nt-ne)=u_0v_0=u_{nb}v_{nc}$ is the common edge of $TM(nt-ne)$ and $TM(1)$. For $k=nt-ne+1,\ldots,nt$, $EM(k)$ is the common edge of $TM(k)$ and $TM(l)$ for some $l<k$. For $1\leq k\leq nt$, let A_k be the region formed by the adjacent triangles $TM(1),\ldots,TM(k)$. We define an *internal* edge in a triangulation of A_k to be an edge of the type u_iv_j, u_iu_{i+m}, or v_jv_{j+m}, $m\geq 2$, which is a common edge of two triangles in A_k.

The following step is performed for $k=1,2,\ldots,nt-1$ to obtain locally optimal internal edges in the triangulation of A. Suppose the internal edges in the triangulation of A_k are locally optimal (this is trivially true for $k=1$). Then the internal edges in the triangulation of A_{k+1} are made locally optimal as follows (with the exception mentioned in the next paragraph when $k=nt-ne-1$). The triangulation of A_{k+1} consists of the triangles in A_k plus the triangle TM(k+1). Let $e=EM(k)$ if $k\leq nt-ne-1$ and $e=EM(k+1)$ if $k\geq nt-ne$. Then e is the only internal edge in the triangulation of A_{k+1} which may not be locally optimal. Therefore apply LOP to e. If it is swapped for the other diagonal edge of the quadrilateral, Q, formed by the two triangles having e as a common edge, then the edges of Q which are internal edges of A_{k+1} may no longer be locally optimal so they are placed in a stack of edges for which LOP must be applied. If the stack is not empty, then the top edge is popped from the stack and LOP is applied to this edge. This may cause another swap and more internal edges of A_{k+1} to be pushed onto the stack. This process of applying LOP is repeated until the stack is empty which implies that the internal edges in the triangulation of A_{k+1} are locally optimal. Lawson (1977) shows that this process must always terminate. (There are only a finite number of possible triangulations of A_{k+1}; a linear ordering of these triangulations can be defined using the sorted vector of the smallest angle in each triangle; and each swap produced by LOP causes a strict advance through this linear ordering of triangulations.)

The above step must be performed twice when $k = nt - ne - 1$ (i.e. the triangulation of A 'closes') since $EM(k)$ and $EM(k+1) = u_0v_0$ may not be locally optimal in the triangulation of A_{k+1}. First the above step is performed for $e = EM(k)$ and then it is performed for $e = EM(k+1)$. We have chosen u_0 so that u_0v_0 is likely to be a Delaunay edge (see Section 5) so u_0v_0 is not likely to be swapped when LOP is applied to it. After the above step is performed for $k = nt - 1$, the internal edges in the triangulation of $A = A_{nt}$ and P are locally optimal and a Delaunay triangulation, $DT(P)$, is obtained by Lemma 4.1 and Theorem 2.1. Figure 5.3 illustrates a Delaunay triangulation $DT(P)$ obtained from $VT(P)$ by making one diagonal edge swap: u_5u_7 replaces u_6v_3 in the quadrilateral $u_5u_6u_7v_3$.

The pseudo-code for converting the triangles of $VT(P)$ in A into Delaunay triangles is given in procedure CONVERT. This procedure can also be used in the case of no interior vertices.

```
Procedure CONVERT($inter, TM, EM, nt, ne$);
# Input: if $inter$ = true then $TM$, $EM$, and $ne$ are output from procedure MMERGE
#             and $nt = nb + nc$
#        else $TM$ and $EM$ are output from procedure MERGE, $nt = mb + mc - 2$,
#             and $ne = 0$
# Output: updated list of triangles $TM$ such that all triangles are Delaunay
#         and updated list of edges $EM$ such that all edges are Delaunay
   for $k$ := 1 to $nt$ do
      if not $inter$ and $k = nt$ then return;
      if $k \le nt - ne - 1$ then $kk := k + 1$ else $kk := k$;
      $top$ := 1;
      $stack(top) := k$;
      while $top \ge 1$ do
           $l := stack(top)$;
           $top := top - 1$;
           Search $TM(kk), TM(kk-1), \ldots, TM(1)$ sequentially until the two
                triangles $TM(m)$ and $TM(n)$ containing edge $EM(l)$ are found;
           $Q$ := quadrilateral $w_1w_2w_3w_4$ where $TM(m) = \Delta w_1w_2w_3$,
                $TM(n) = \Delta w_1w_3w_4$, and $EM(l) = w_1w_3$;
           if $w_4$ is in the circle through $w_1, w_2, w_3$ then
                $TM(m) := \Delta w_1w_2w_4$;
                $TM(n) := \Delta w_2w_3w_4$;
                $EM(l) := w_2w_4$;
                $w_5 := w_1$;
                for $q$ := 1 to 4 do
                   if $1 \le$ index$(w_qw_{q+1}) \le k$ then
                        $top := top + 1$;
                        $stack(top)$ := index$(w_qw_{q+1})$;
   return;
```

In this procedure, the function $index(ww')$ is defined to be 0 if ww' is an edge of the type

u_iu_{i+1} or v_jv_{j+1} otherwise it is the index of edge ww' in the list EM. The function is used to determine whether an edge is an internal edge in the triangulation of A_{k+1}. For each triangle $\Delta w_1w_2w_3$ in the list TM, we store the values $index(w_1w_2)$, $index(w_2w_3)$, and $index(w_3w_1)$ in addition to the vertices of the triangle, so that it is easy to determine $index(w_qw_{q+1})$ in the innermost for loop. When LOP is applied to an internal edge e, the list TM must be searched for the two triangles containing e. The reason for the sequential search starting backwards from $TM(kk)$ will be discussed below.

We now estimate the number of edge swaps required for converting $VT(P)$ into $DT(P)$ since the efficiency of the procedure depends on the number of swaps. In $DT(P)$, strip A may contain edges of the types u_iv_j, u_iu_{i+m}, and v_jv_{j+m}, $m \geq 2$. In $VT(P)$, A contains mostly edges of type u_iv_j and possibly some edges of type v_jv_{j+m}. If u_iv_j is a Delaunay edge then it is likely to be an edge of $VT(P)$ since u_0v_0 is likely to be a Delaunay edge and in cases (a), (b), and (c) of (5.3) the next edge $u_{i+1}v_j$ or u_iv_{j+1} is chosen to be locally optimal in the quadrilateral $u_iu_{i+1}v_{j+1}v_j$. So edge swaps are needed to obtain Delaunay edges of types u_iu_{i+m} and v_jv_{j+m}. The following theorem indicates when no swaps are required.

Theorem 7.1 : If u_0v_0 is a Delaunay edge and there are no Delaunay edges of types u_iu_{i+2} or v_jv_{j+2}, then $VT(P) = T(P)$ is a Delaunay triangulation.

Proof : First we show that under the hypothesis of the theorem there are no Delaunay edges of types u_iu_{i+m} or v_jv_{j+m}, $m \geq 3$. Suppose u_lu_{l+m}, $m \geq 3$, is a Delaunay edge. Then $u_lu_{l+1} \cdots u_{l+m}u_l$ must form a simple subpolygon of P which contains no vertices in its interior. In a Delaunay triangulation of the mesh vertices, this subpolygon contains $m-1$ triangles involving only the vertices $u_l, u_{l+1}, \ldots, u_{l+m}$. Therefore there must be a Delaunay edge u_iu_{i+2} where $l \leq i \leq l+m-2$. This contradicts the hypothesis therefore there are no Delaunay edges of type u_iu_{i+m}, $m \geq 3$. Similarly there are no Delaunay edges of type v_jv_{j+m}, $m \geq 3$. Therefore the Delaunay edges in A are all of the type u_iv_j.

We show by induction that the edges produced by procedure MERGE (which are the same as those produced by procedure MMERGE) are Delaunay edges. Suppose $u_i v_j$ is a Delaunay edge (initially $u_0 v_0$ is a Delaunay edge). Then $\Delta u_i u_{i+1} v_j$ or $\Delta v_{j+1} v_j u_i$ must be a Delaunay triangle and $u_{i+1} v_j$ or $u_i v_{j+1}$ must be a Delaunay edge, respectively. If $i = nb$, then $\Delta v_{j+1} v_j u_i$ is the only possible triangle therefore $u_i v_{j+1}$ is a Delaunay edge. Similarly if $j = nc$, then $u_{i+1} v_j$ is a Delaunay edge. Suppose $i < nb$ and $j < nc$. Consider quadrilateral $Q = u_i u_{i+1} v_{j+1} v_j$ in (5.3). In case (a), the circle test chooses the next Delaunay edge. In cases (b), (c), and (d) the chosen triangle is a Delaunay triangle since the other triangle is not a 'valid' triangle - it is a CW or overlapping triangle (see Figure 5.2). Therefore procedure MERGE generates the next Delaunay edge, either $u_{i+1} v_j$ or $u_i v_{j+1}$. By induction, procedure MERGE produces Delaunay edges in A. The edges generated by procedure INTTRIANG are Delaunay edges by Lemma 4.1. Therefore $VT(P) = T(P)$ is a Delaunay triangulation. □

In general, $DT(P)$ contains edges of types $u_i u_{i+m}$ and $v_j v_{j+m}$, $m \geq 2$. The location of the mesh vertices and Lemma 2.1 can be used to determine where these Delaunay edges are likely to occur. We first suppose that the parameters h_i satisfy (1.1). An edge $v_j v_{j+m}$ may be a Delaunay edge if it lies entirely in A and angle$(v_{j+m} v_{j+l} v_j)$ for some l, $0 < l < m$, is smaller than approximately 120°, e.g. the edge joining $v_j = (x_{i+1,0}, y_{i+1})$ and $v_{j+2} = (x_{i+2,0}, y_{i+2})$ in Figure 4.2. It is unlikely that $m > 2$ and the number of Delaunay edges of type $v_j v_{j+2}$ and the number of swaps to obtain these edges are expected to be small relative to nc.

An edge $u_i u_{i+m}$ may be a Delaunay edge if it is near a vertex of P or $int(P)$ with an interior angle smaller than approximately 90°, e.g. $u_5 u_7$ in Figure 5.3. If P does not contain 'small' interior angles (e.g. $\leq 20°$), then m is likely small relative to nb and the number of Delaunay edges of type $u_i u_{i+m}$, $m \geq 2$, and the number of swaps to obtain these edges are expected to be small relative to nb. If P contains a small interior angle (from elementary geometry it can be seen that at most two interior angles of a convex polygon can be less than 60°), then as this angle decreases, the maximal value m of a Delaunay edge $u_i u_{i+m}$ near this angle increases and the number of swaps required to obtain the Delaunay edges in the subpolygon $u_i u_{i+1} \cdots u_{i+m} u_i$ increases. Therefore if the h_i satisfy (1.1) and P contains no small interior angles, then the number of swaps needed to obtain $DT(P)$ from $VT(P)$ is expected to be small relative to $nb + nc$.

Now we suppose that the h_i do not satisfy (1.1) for all i. As the ratios h_i/h increase, more Delaunay edges of type $v_j v_{j+m}$, $m \geq 2$, are likely to occur in $DT(P)$ and be produced in $VT(P)$ by procedure MMERGE. As the ratios h_i/h decrease, more Delaunay edges of type $u_i u_{i+m}$, $m \geq 2$, are likely to occur in $DT(P)$ and the number of swaps needed to obtain $DT(P)$ from $VT(P)$ can be $O(nb^2)$ if P contains a small interior angle.

When LOP is applied to edge $EM(k)$ in A_{k+1} for $k \leq nt-ne-1$, edge swaps, if any, occur for triangles $TM(n)$ near $TM(k+1)$ with indices $n \leq k+1$. Therefore we store the triangles of A in a linear list and when LOP is applied to an edge e in the triangulation of A_{k+1}, a search is made sequentially backwards in this list starting from $TM(k+1)$ until the two triangles containing edge e are found. The number of triangles searched is expected to be a small constant if P does not contain small interior angles. When LOP is applied to edge $EM(nt-ne) = u_0 v_0$ in A_{nt-ne}, one of the triangles containing edge $u_0 v_0$ is near the end of list TM and the other is near the front of TM. By using this information, few triangles are searched when LOP is applied to $u_0 v_0$. If $VT(P)$ contains an edge of type $v_j v_{j+m}$, $m \geq 2$, then $O(nt)$ triangles are expected to be searched when LOP is applied to this edge since the two triangles containing this edge may not be close together in TM. By Conjecture 6.1, ne is expected to be zero if the h_i satisfy (1.1). Therefore if P contains no small interior angles and the h_i satisfy (1.1), then the number of triangles searched in the applications of LOP is expected to be $O(nb+nc)$.

8. Summary and time complexity

The pseudo-code for our triangulation algorithm is summarized in procedure DELTRIANG.

```
Procedure DELTRIANG($P, m, hlist, T, n_t$);
# Input: convex polygon $P$ with $m$ vertices and list of
#        triangle size parameters $hlist = [h, h_1, h_2, \ldots, h_m]$
# Output: list of triangles $T$ and number of triangles $n_t$
   SHRINK($P, m, h/\sqrt{2}, int(P), ni$);
   if $ni > 0$ then $inter$ := true else $inter$ := false;
   if $inter$ then
      Rotate coordinate system so that diameter of $int(P)$ is parallel to y-axis;
      INTTRIANG($int(P), h, TI, nt, C, nc$);
   else
      Rotate coordinate system so that diameter of $P$ is parallel to y-axis;
      $TI := [\,]$; $C := [\,]$; $nt := 0$; $nc := -2$;
   for $i := 1$ to $m$ do
      Generate mesh vertices on edge $e_i$ at an equal spacing of $\bar{h}_i$
      as computed by (5.1);
   if $inter$ then
      Determine $B = [u_0, u_1, \ldots, u_{nb}]$;
      MMERGE($B, nb, C, nc, TM, EM, ne$);
   else
      Determine $B_L = [u_0, u_1, \ldots, u_{mb}]$, $B_R = [u_{nb}, u_{nb-1}, \ldots, u_{mb}]$;
      MERGE($inter, B_L, mb, B_R, nb - mb, TM, EM$);
      $ne := 0$;
   CONVERT($inter, TM, EM, nb + nc, ne$);
   $T$ := append($TI, TM$);
   $n_t := nt + nb + nc$;
   return;
```

We now discuss the time complexity for procedure DELTRIANG to construct the Delaunay triangulation, $DT(P)$, in P. Let n_t and n_v be the number of triangles and mesh vertices, respectively, generated in P by procedure DELTRIANG. Let nb be the number of mesh vertices on ∂P and m be the number of vertices of P. n_t, n_v, nb, and m are related by the formulas (Lawson (1977))

$$n_t = 2n_v - nb - 2 \text{ and } n_t \geq nb - 2 \geq m - 2. \tag{8.1}$$

By Theorem 3.1, $int(P)$ is determined from P in $O(m)$ time by procedure SHRINK. The diameter of $int(P)$ or P is found in $O(m)$ time (Shamos (1975)). In procedure INTTRIANG, the generation of mesh vertices in $int(P)$ requires $O(n_v)$ time and the generation of triangles and closed walk C requires $O(n_t)$ time. The generation of mesh vertices on ∂P requires $O(nb)$ time. The generation of triangles in A by procedure MMERGE requires $O(n_t)$ time if ne, the number of edges of type $v_j v_{j+k}$, $k \geq 2$, is zero or bounded by a small constant (ne is expected to be zero by Conjecture 6.1 if the h_i satisfy (1.1)). We conjecture that the time complexity of procedure MMERGE is $O(n_t)$ even if ne is large. In the case of no interior vertices, the generation of triangles in P by procedure MERGE requires $O(n_t)$ time. From the discussion at the end of Section 7, the number of edge swaps and the number of triangle searches in the applications of LOP in procedure CONVERT are expected to be $O(n_t)$ if P contains no small interior angles and the h_i satisfy (1.1). Therefore if P contains no small interior angles and the h_i satisfy (1.1), then the time complexity for constructing $DT(P)$ by procedure DELTRIANG is expected to be $O(n_t) = O(n_v)$ using (8.1). Otherwise the time complexity of procedure DELTRIANG may be nonlinear in n_t since the number of swaps in procedure CONVERT may be $O(nb^2)$.

Procedure DELTRIANG has been implemented in PASCAL and used to generate triangular meshes in over a thousand convex polygons created from the decomposition of various regions by our finite element triangulation method (Joe (1984)). There are few interior angles less than 20° in these polygons and the parameters h_i satisfy (1.1). We made the following observations about the performance of procedure DELTRIANG. In the case of at least two interior vertices, the heuristic for selecting u_0 so that $u_0 v_0$ is a Delaunay edge (see Section 5) has not failed, and no edges of type $v_j v_{j+k}$, $k \geq 2$, have been generated by procedure MMERGE, i.e. procedures INTTRIANG and MERGE have not produced an invalid triangulation $T(P)$. Also, the average number of edge swaps per polygon and the average number of triangle searches per application of LOP are small constants. Therefore the empirical time complexity of procedure DELTRIANG is $O(n_t)$ for these polygons and triangle size parameters.

Acknowledgment

The author would like to thank Prof. R. B. Simpson for many helpful discussions and his valuable comments on preliminary drafts of this paper. This work has been partially supported by a scholarship from the Natural Sciences and Engineering Research Council of Canada.

References

R. E. Bank (1982), PLTMG users' guide, Dept. of Mathematics, University of California at San Diego.

A. Bykat (1976), Automatic generation of triangular grid: I - subdivision of a general polygon into convex subregions, II - triangulation of convex polygons, *Int. J. for Num. Meth. in Eng.,* 10, pp. 1329-1342.

J. C. Cavendish (1974), Automatic triangulation of arbitrary planar domains for the finite element method, *Int. J. for Num. Meth. in Eng.,* 8, pp. 679-696.

B. Joe (1984), Finite element triangulation of complex regions using computational geometry, Ph.D. thesis, Dept. of Computer Science, University of Waterloo.

B. Joe and R. B. Simpson (1983), Finite element triangulation of complex regions, submitted to *ACM Trans. on Graphics.*

C. L. Lawson (1977), Software for C^1 surface interpolation, *Mathematical Software III,* ed. J. R. Rice, Academic Press, pp. 161-194.

D. T. Lee and F. P. Preparata (1979), An optimal algorithm for finding the kernel of a polygon, *JACM,* 26, pp. 415-421.

D. T. Lee and B. J. Schachter (1979), Two algorithms for constructing a Delaunay triangulation, Rep. 79ASD007, General Electric Co., Daytona Beach, Florida.

M. I. Shamos (1975), Geometric complexity, *Proc. of the 7th Annual ACM Symp. on Theory of Computing,* pp. 224-233.

M. I. Shamos and D. Hoey (1975), Closest-point problems, *Proc. 16th Annual Symp. on Foundations of Computer Science,* pp. 151-162.

R. D. Shaw and R. G. Pitchen (1978), Modifications to the Suhara-Fukuda method of network generation, *Int. J. for Num. Meth. in Eng.,* 12, pp. 93-99.

R. Sibson (1978), Locally equiangular triangulations, *Computer Journal,* 21, pp. 243-245.

R. B. Simpson (1981), A two-dimensional mesh verification algorithm, *SIAM J. Sci. Stat. Comput.,* 2, pp. 455-473.