International Journal of Computer and Information Sciences, Vol. 9, No. 3, 1980

Two Algorithms for Constructing a Delaunay Triangulation[1]

D. T. Lee[2] and B. J. Schachter[3]

Received July 1978; revised February 1980

This paper provides a unified discussion of the Delaunay triangulation. Its geometric properties are reviewed and several applications are discussed. Two algorithms are presented for constructing the triangulation over a planar set of N points. The first algorithm uses a divide-and-conquer approach. It runs in $O(N \log N)$ time, which is asymptotically optimal. The second algorithm is iterative and requires $O(N^2)$ time in the worst case. However, its average case performance is comparable to that of the first algorithm.

KEY WORDS: Delaunay triangulation; triangulation; divide-and-conquer; Voronoi tessellation; computational geometry; analysis of algorithms.

1. INTRODUCTION

In this paper we consider the problem of triangulating a set of points in the plane. Let V be a set of $N \geqslant 3$ distinct points in the Euclidean plane. We assume that these points are not all colinear. Let E be the set of $\binom{N}{2}$ straight-line segments (edges) between vertices in V. Two edges $e_1, e_2 \in E$, $e_1 \neq e_2$, will be said to *properly intersect* if they intersect at a point other than their endpoints. A triangulation of V is a planar straight-line graph $G(V, E')$ for which E' is a maximal subset of E such that no two edges of E' properly intersect.(16)

---

[1] This work was supported in part by the National Science Foundation under grant MCS-76-17321 and the Joint Services Electronics Program under contract DAAB-07-72-0259.

[2] Northwestern University Dept. of Electrical Engineering and Computer Science, Evanston, Illinois 60201.

[3] General Electric Co., P.O. Box 2500, Daytona Beach, Florida 32015.

There is no conceptual difficulty involved in constructing a triangulation. Any set of points can be triangulated if edges are added with the proviso that no new edge intersects an existing edge. We will investigate a particular triangulation called the *Delaunay triangulation.*[3] It has the property that the circumcircle of any triangle in the triangulation contains no point of V in its interior.

This paper is the result of a recent study whose objective was to develop an efficient algorithm for fitting triangular faceted surfaces to digital terrain data. A piecewise planar surface is used as a terrain model by all visual flight simulators. It was concluded that the Delaunay triangulation is an excellant choice for this application, based on the initial objectives of minimizing computation time and producing a good visual display.

In Sec. 2, we will formally define the Delaunay triangulation and review its properties. Then in Sec. 3, we will provide two algorithms for its construction. Section 4 will cover some applications of the triangulation.

2. DEFINITION AND PROPERTIES OF THE DELAUNAY TRIANGULATION

Suppose that we are given a set $V = \{v_1, ..., v_N\}$, $N \geqslant 3$, of points in the Euclidean plane. Assume that these points are not all colinear, and that no four points are cocircular. Let $d(v_i, v_j)$ denote the Euclidean distance between points v_i and v_j. The region $V(i) = \{x \in E^2 \mid d(x, v_i) \leqslant d(x, v_j), j = 1, ..., N\}$ which is the locus of points closer to vertex v_i than to any other vertex is called the Voronoi[37] (or Dirichlet, Wigner-Seithz, Thiessen,[36] or "S"[24]) *polygon* associated with the vertex v_i.

Voronoi polygons may be thought of as the cells of a growth process. Suppose that we let each vertex in V be the nucleus of a growing cell. Cells will propagate outward from their nuclei, simultaneously and at a uniform rate. The border of a growing cell will freeze in place at its points of contact with the border of another growing cell.

Eventually, only the cells whose nuclei are on the convex hull of V are still expanding. The remaining cells have completely partitioned (or tessellated) a region of the plane into a set of nonoverlapping closed convex polygons, one polygon about each nucleus. These closed polygons, together with the open polygons on the convex hull, define a *Voronoi tessellation* of the entire plane.

Let us take a closer look at this process. Since all cells expand at the same rate, the first point of contact between two cells must occur at the midpoint between their nuclei. Likewise, every point of continuing contact must be equidistant from the two nuclei. These points are on the common edge (called a *Voronoi edge*) of two developing Voronoi polygons. This

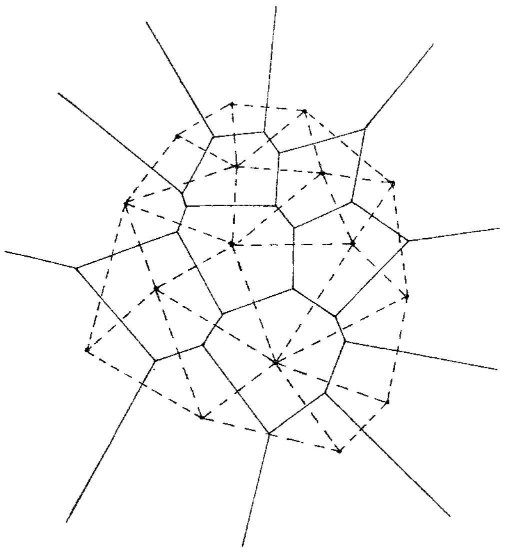


Fig. 1. Voronoi diagram for a set of 16 points (solid lines); Delaunay triangulation (dashed lines).

edge continues elongating until it encounters the border of a third expanding cell. The point of contact (called a *Voronoi point*) of this edge and the border of the third cell must be equidistant from the growth centers of all three cells. It is therefore the circumcenter of the triangle defined by the three nuclei.

Voronoi cells which share a common edge are called *Voronoi neighbors*. The aggregate of triangles formed by connecting the growth centers of all Voronoi neighbors tessellates the area within the convex hull of the point set. This tessellation is called the *Delaunay triangulation* $DT(V)$ of V. An example of a Voronoi tessellation and its dual is shown in Fig. 1.

Each Voronoi point corresponds to a triangle and each Voronoi edge to a Delaunay edge. Since the number of Voronoi points and edges are both $O(N)$,[4] the number of Delaunay triangles and edges are $O(N)$. To be more precise we have the following.

Lemma 1. Given a set V of N points, any triangulation $T(V)$ has the same number of triangles, $N_t = 2(N - 1) - N_h$, and the same number of edges, $N_e = 3(N - 1) - N_h$, where N_h is the number of points on the convex hull of V.

[4] Note: We say that $g(n) = O(f(n))$ if $|g(n)| \leqslant cf(n)$ for some constant c and all sufficiently large n. We say that $g(n) = \Omega(f(n))$ if $|g(n)| \geqslant cf(n)$ for some constant $c > 0$ and all sufficiently large n. We say that $g(n) = \theta(f(n))$ if both $g(n) = O(f(n))$ and $g(n) = \Omega(f(n))$.

Proof. By induction, see Ref. 11 for example.

Now we will state without proof some properties of the Delaunay triangulation $DT(V)$.

Lemma 2. Given a set $V = \{v_1, ..., v_N\}$ of points, any edge (v_i, v_j) is a Delaunay edge of $DT(V)$ if and only if there exists a point x such that the circle centered at x and passing through v_i and v_j does not contain in its interior any other point of V.

Corollary 1. Given a set $V = \{v_1, ..., v_N\}$ of points, the edge (v_i, v_j) on the boundary of the convex hull of V is a Delaunay edge.

Lemma 3. Given a set $V = \{v_1, ..., v_N\}$ of points, $\triangle v_i v_j v_k$ is a Delaunay triangle of $DT(V)$ if and only if its circumcircle does not contain any other point of V in its interior.

The proofs of these lemmas can be found in Refs 12 and 32. The latter property is called the *circle criterion.* It is often used as a rule for constructing a triangulation. Triangulations may also be constructed according to the *MAX-MIN angle criterion,* i.e., the minimum measure of angles of all the triangles in the triangulation is maximized. We shall investigate the relationship between these two criteria.

The following analysis follows that given by Lawson[9,10] (see also the work of Sibson[33]). Consider a very simple triangulation, that over the vertices of a *strictly convex* quadrilateral. A quadrilateral is called strictly convex if its four interior angles are each less than 180°. A quadrilateral can be partitioned into two triangles in two possible ways. Each of the criteria described above can be thought of as a rule for choosing a preferred triangulation.

By examining the case of four cocircular points (Fig. 2), one can show that the two criteria are equivalent. Suppose that for this example line segment (v_2, v_3) is shorter than (v_3, v_4), (v_4, v_1), and (v_1, v_2). Let the angular measure of the arc $\widehat{v_2, v_3}$ be 2θ. Then the angles $\angle v_3, v_1, v_2$ and $\angle v_3, v_4, v_2$ are each θ. Thus the two possible triangulations over the four points have the same minimum angle. The choice of a preferred triangulation is then arbitrary according to the MAX-MIN angle criterion. The Voronoi tessellation of the quadrilateral also exhibits a tie case. All four Voronoi polygons meet at a single point.

Further analysis shows that moving one point, say point v_4 in Fig. 2, inside the circle causes $\angle v_3, v_4, v_2$ to increase and points v_2 and v_4 to be the growth centers of Voronoi neighbors. Consequently, the two criteria and the Voronoi tessellation of the polygon all now prescribe the connection of vertices v_2 and v_4.

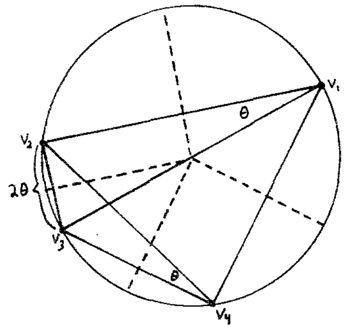


Fig. 2. Lawson's example showing a triangulation over four cocircular points. The Voronoi tessellation is shown as dashed lines.

A fourth criterion has been studied, that of choosing the minimum length diagonal. Shamos and Hoey[32] claim that a Delaunay triangulation is a minimum edge length triangulation. Lawson[9] and Lloyd[16] prove by counterexample that this is not the case (Fig. 3).

Lawson[10] gives the following *local optimization procedure* (LOP) for constructing a triangulation. Let e be an internal edge (in contrast to an edge on the convex hull) of a triangulation and Q be the quadrilateral formed by the two triangles having e as their common edge. Consider the circumcircle of one of the triangles. This circle passes through three vertices of Q. If the fourth vertex of the quadrilateral is within the circle, replace e by the other diagonal of Q, otherwise no action is taken. An edge of the triangulation is said to be *locally optimal* if an application of the LOP would not swap it. Since for any set of vertices, the number of triangles in any triangulation is a constant, a linear ordering over the set of all triangles can be defined as follows. To each triangle in the triangulation we assign a value, which is

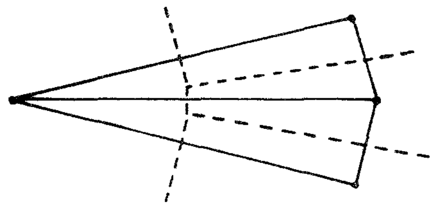


Fig. 3. Lloyd's counterexample to Shamos and Hoey's claim that a Delaunay triangulation is a minimum edge length triangulation. The Voronoi tessellation (shown as dashed lines) indicates the use of the longer diagonal for a Delaunay triangulation.

the measure of its minimum angle. Let N_t denote the number of triangles. For each triangulation we have an indicator vector of N_t components, each component corresponding to the minimum angle of its corresponding triangle. Triangles are sorted in nondecreasing order. Given any two triangulations T and T', define $T < T'$ if and only if the associated indicator vector of T is lexicographically less than that of T'.

Theorem 1.(10,12) Given a triangulation T, if an application of the LOP to an edge e results in a swapping of the edge with any other edge e', thus producing a new triangulation T', then $T < T'$; i.e., T' strictly follows T in the linear ordering defined above.

Proof. Let I be an indicator vector for T. The measures of the smallest angles in the two triangles of T sharing the edge e occur as two of the components of I, say I_j and I_k with $j < k$ and thus $I_j \leqslant I_k$. Since a swap was made when applying the LOP, the smallest angle in both of the two new triangles of T' sharing the edge e' must be strictly greater than I_j. It follows that the indicator vector I for T is lexicographically less than the indicator I' for T' and thus $T < T'$.

Theorem 2.(10,12) All internal edges of a triangulation T of a finite set V are locally optimal if and only if no point of V is interior to any circumcircle of a triangle of T.

Proof. If no point of V is interior to the circumcircle of any triangle of T, then the application of LOP to any edge will not swap it. Thus all edges are locally optimal. If all edges are locally optimal, then we show that no point of V is interior to the circumcircle of any triangle. Suppose that the circumcircle K of a triangle $\triangle abc$ contains a point v of V. Let δ be the distance of v to its nearest edge, say (a, c), as shown in Fig. 4. Assume that among all triangles of T whose circumcircles contain v as an interior point, none has an edge which is at a distance less than δ from v. Since v is on the opposite side of (a, c) from b, the edge (a, c) is not a boundary edge of T. Thus, there is another triangle $\triangle acq$ sharing an edge with triangle $\triangle abc$. The vertex q cannot be interior to the circle K as this would contradict the hypothesis that edge (a, c) is locally optimal. The vertex q cannot be in the crosshatched region of the diagram, or $\triangle acq$ will contain v in its interior. Suppose that edge (c, q) is the nearest edge of $\triangle acq$ to v. Note that the distance from (c, q) to v is less than δ. Since the circumcircle of $\triangle acq$ also contains v in its interior, we have a contradiction that $\triangle abc$ is the triangle with an edge at the smallest distance from v.

By the above theorem, the edges of the Delaunay triangulation of a finite set V of points are locally optimal. If we assume that no more than

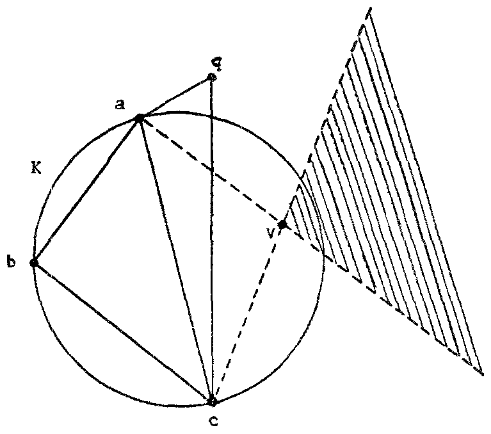


Fig. 4. Illustration of the proof of Theorem 2.

three points are cocircular, the uniqueness of the Delaunay triangulation(33) suggests the following theorem.

Theorem 3.(12) A triangulation $T(V)$ is a Delaunay triangulation if and only if its indicator vector is lexicographically maximum, i.e., no triangulation follows it in the linear ordering.

Proof. If the triangulation T is lexicographically maximum, then all the edges of T must be locally optimal, which implies that no circumcircle of any triangle will contain any other point of V in its interior (Theorem 2). Thus, T is the Delaunay triangulation $DT(V)$. To prove the converse, suppose that the Delaunay triangulation is not maximum in the linear ordering. That is, there exists another triangulation $T(V)$, such that $DT(V) < T(V)$. Repeatedly applying the LOP to $T(V)$ will produce a triangulation $T'(V)$ in which all the edges are locally optimal. Since $DT(V) < T(V) < T'(V)$, $T'(V)$ would also be a Delaunay triangulation by Theorem 2 and Lemma 3. However, since the Delaunay triangulation is unique, $T'(V) = DT(V)$, a contradiction.

Corollary 2. The Delaunay triangulation of a set of points satisfies the MAX-MIN angle criterion. (Note that a triangulation which satisfies the MAX-MIN angle criterion is not necessarily a Delaunay triangulation.)[5]

[5] Consider a triangulation T whose indicator vector is $(i_1, ..., i_t)$ where i_1 is the minimum angle. Suppose that $\triangle abc$ is the triangle with the smallest angle. If applying LOP to the edges of ab, bc, ca will not result in a swap, then T satisfies the MAX-MIN angle criterion. Since we have only checked one triangle in the triangulation, we cannot say that T is a Delaunay triangulation. However, if we apply the LOP to all edges of T, until no more swapping occurs, the resulting triangulation will be Delaunay.

We have shown that the Delaunay triangulation of a set of points is a maximum in the linear ordering over the set of possible triangulations. Now we are ready to describe two algorithms for its construction.

3. TECHNIQUES FOR CONSTRUCTING THE DELAUNAY TRIANGULATION

We will present two algorithms for constructing a Delaunay triangulation. The first is rather involved, but its running time is asymptotically optimal. The second algorithm is simpler to understand, is simpler to program, and requires less overhead. These features make it especially attractive for small and medium-sized data sets ($\simeq$500 points or less).

The first algorithm is comparable to the one given by Lewis and Robinson(15) in that it divides the original data set into disjoint subsets. After obtaining a solution for each of these subsets, it combines solutions to yield the final result. Lewis and Robinson(15) claim, without proof, that their algorithm runs in $O(N \log N)$. But in fact,[6] the running time of their algorithm is $O(N^2)$. The second algorithm that we will present is iterative. It follows the idea proposed by Lawson.(9) Nelson(23) gives a similar method.

3.1. Divide-and-Conquer

Shamos and Hoey(32) present an $\theta(N \log N)$ algorithm for constructing the Voronoi diagram for a set of N points in the plane. Green and Sibson(7) also implement an algorithm for this purpose, but the running time is $O(N^2)$ in the worst case. Lee(13) modifies the procedure given by Shamos and Hoey and extends the method to any L_p metric, $1 \leqslant p \leqslant \infty$.

Once the Voronoi diagram is obtained, its dual graph (i.e., the Delaunay triangulation) can be constructed in an additional $O(N)$ time. To eliminate the need for a two-step procedure, we have developed the following algorithm which constructs a triangulation directly. The running time of the algorithm is shown to be $O(N \log N)$. Shamos and Hoey(32) show that the construction of any triangulation over N points requires $\Omega(N \log N)$ time. Thus, our algorithm is asymptotically optimal.

We will begin by describing the data structures and notation to be used in the sequel. For each point v_i, we keep an ordered adjacency list of points $v_{i1}, \ldots, v_{ik}$, where $(v_i, v_{ij}), j = 1, \ldots, k$, is a Delaunay edge. The list is doubly linked and circular. $PRED(v_i, v_{ij})$ denotes the point v_{ip} which

[6] There are at least four triangulation algorithms in the computer literature which claim to be $O(N \log N)$, but which are in fact $O(N^2)$. A counterexample for several of these is given in Appendix A.

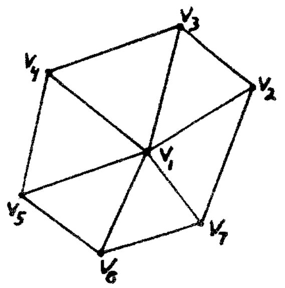


Fig. 5. Example illustrating a doubly linked circular list.

appears clockwise (CW) of and immediately after the point v_{ij}. The counterclockwise function *SUCC* operates in a similar manner. For example in Fig. 5, $v_5 = PRED(v_1, v_6)$ and $v_5 = SUCC(v_1, v_4)$.

If the point v_i is on the convex hull $CH(V)$, then the first entry on its adjacency list is the point denoted $FIRST(v_i)$, which appears after v_i if we traverse the boundary of $CH(V)$ in a CCW direction. Let $l(v_i, v_j)$ denote the line segment directed from v_i to v_j.

Now we are ready to construct the Delaunay triangulation. First, we sort the given set V of N points in lexicographically ascending order and rename the indices so that $v_1 < v_2 < \cdots < v_N$, such that $(x_i, y_i) = v_i < v_j$ if and only if either $x_i < x_j$ or $x_i = x_j$ and $y_i < y_j$. Next we divide V into two subsets V_L and V_R, such that $V_L = \{v_1, ..., v_{\lfloor N/2 \rfloor}\}$ and $V_R = \{v_{\lfloor N/2 \rfloor + 1}, ..., v_N\}$. Now we recursively construct the Delaunay triangulations $DT(V_L)$ and $DT(V_R)$. To merge $DT(V_L)$ and $DT(V_R)$ to form $DT(V_L U V_R)$, we make use of the convex hull $CH(V_L U V_R)$. The convex hull can be obtained in $O(N)$ time[26] from the union of the convex hulls $CH(V_L)$ and $CH(V_R)$. The convex hulls can also be computed recursively. Forming the union of $CH(V_L)$ and $CH(V_R)$ will result in two new hull edges which are the lower and upper common tangents of $CH(V_L)$ and $CH(V_R)$. These two common tangents are known to be in the final Delaunay triangulation.

The following subroutine finds the lower common tangent of $CH(V_L)$ and $CH(V_R)$. For each convex hull $CH(S)$, we maintain two points $LM(S)$ and $RM(S)$, which are the leftmost and rightmost points of S, respectively.

Subroutine HULL

(**Comment:** HULL is input two convex hulls. It finds their lower common tangent (Fig. 6). The upper common tangent can be found in an analogous manner.)

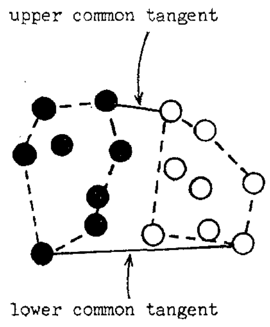


Fig. 6. Illustration of the upper and lower common tangents of two convex hulls.

(**Comment:** $l(X, Y)$ denotes the line *directed* from X to Y.)

```
    $X \leftarrow RM(V_L)$; $Y \leftarrow LM(V_R)$
    $Z \leftarrow FIRST(Y)$; $Z' \leftarrow FIRST(X)$; $Z'' \leftarrow PRED(X, Z')$
A:          IF ($Z$ is-right-of $l(X, Y)$)
            $Z \leftarrow SUCC(Z, Y)$
            $Y \leftarrow Z$
            ELSE
              IF ($Z''$ is-right-of $l(X, Y)$
              $Z'' \leftarrow PRED(Z'', X)$
              $X \leftarrow Z''$
              ELSE
              RETURN $(X, Y)$
              ENDIF
            ENDIF
            GO TO A
    END HULL
```

The lower common tangent will be used as an input to the following subroutine which merges the two triangulations $DT(V_L)$ and $DT(V_R)$.

Subroutine MERGE

(**Comment:** MERGE is input two triangulations and the upper and lower common tangents of their convex hulls. It merges the two triangulations, starting with the lower common tangent, zigzagging upward until the upper common tangent is reached.)

(**Comment:** Initially, the points adjacent to the endpoints of the lower common tangent are examined. Using the circle criterion, we either connect:

1. the left endpoint (in V_L) of the lower common tangent to a point adjacent to the right endpoint (in V_R) of the lower common tangent, or
2. the right endpoint (in V_R) of the lower common tangent to a point adjacent to the left endpoint (in V_L) of the lower common tangent.

The above process is repeated with the newly found edge taking the place of the lower common tangent, and each succeeding new edge taking the place of that. The subroutine continues in this manner until the upper common tangent is reached.)

(**Comment:** The adjacency list of the right (left) endpoint in V_R (V_L) of the current edge being considered is examined in a CW (CCW) direction starting with the left (right) endpoint of the edge.)

(**Comment:** INSERT(A, B) inserts point A into the adjacency list of B and point B into the adjacency list of A at proper positions. DELETE(A, B) deletes A from the adjacency list of B and B from the adjacency list of A.)

(**Comment:** QTEST(H, I, J, K) tests the quadrilateral having CCW ordered vertices H, I, J, K. It returns TRUE if the circumcircle of $\triangle HIJ$ does not contain K in its interior, and returns FALSE otherwise.)

```
step 1:  INITIALIZATION
         $BT \leftarrow$ lower common tangent
         $UT \leftarrow$ upper common tangent
          $L \leftarrow$ left endpoint of $BT$
          $R \leftarrow$ right endpoint of $BT$
step 2:        DO UNTIL ($BT$ equals $UT$)
step 3:        $A \leftarrow B \leftarrow$ FALSE
step 4:        INSERT($L$, $R$)
step 5:        $R_1 \leftarrow PRED(R, L)$
step 6:          IF ($R_1$ is-left-of $l(L, R)$)
step 7:          $R_2 \leftarrow PRED(R, R_1)$
step 8:            DO UNTIL (QTEST($R_1$, $L$, $R$, $R_2$))
                   DELETE($R$, $R_1$)
                   $R_1 \leftarrow R_2$
                   $R_2 \leftarrow PRED(R, R_1)$
                   END DO UNTIL
```

```
step 9:              ELSE
                     $A \leftarrow$ TRUE
                     ENDIF
step 10:           $L_1 \leftarrow SUCC(L, R)$
step 11:             IF ($L_1$ is-right-of $l(R, L)$)
step 12:             $L_2 \leftarrow SUCC(L, L_1)$
step 13:               DO UNTIL (QTEST($L, R, L_1, L_2$))
                       DELETE ($L, L_1$)
                       $L_1 \leftarrow L_2$
                       $L_2 \leftarrow SUCC(L, L_1)$
                       END DO UNTIL
step 14:             ELSE
                     $B \leftarrow$ TRUE
                     ENDIF
step 15:             IF ($A$)
                     $L \leftarrow L_1$
step 16:             ELSE
step 17:               IF ($B$)
                       $R \leftarrow R_1$
step 18:               ELSE
step 19:                 IF (QTEST($L, R, R_1, L_1$))
                         $R \leftarrow R_1$
step 20:                 ELSE
                         $L \leftarrow L_1$
                         ENDIF
step 21:               ENDIF
step 22:             ENDIF
step 23:           $BT \leftarrow l(L, R)$
                   END DO UNTIL
step 24:   END MERGE
```

To show that the above algorithm merges two triangulations correctly, it is sufficient to show that each edge we insert into the triangulation is a Delaunay edge (step 4). Initially, the first edge (L, R) is a lower common tangent and is known to be a Delaunay edge. Steps 5–8 delete the edges in $DT(V_R)$ which are not Delaunay edges in $DT(V)$ by determining if L is within the circumcircle of $\triangle R, R_1, R_2$. If so, the edge (R, R_1) is not a Delaunay edge and must be deleted. Steps 10–13 perform the same operation on the edges in $DT(V_L)$. An example of this process is shown in Fig. 7(a). Now the circumcircle K_R of $\triangle L, R, R_1$ does not contain any point of V_R in its interior and the circumcircle K_L of $\triangle R, L, L_1$ does not contain any point of V_L in its interior. Now as shown in Fig. 7(b), we have a choice of

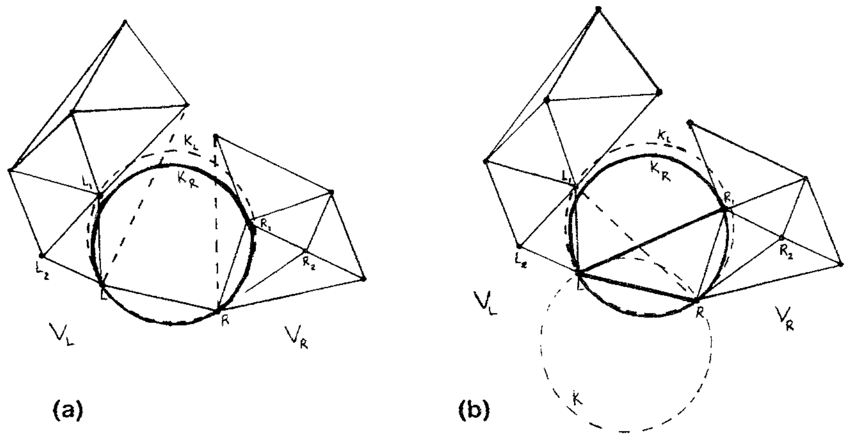


Fig. 7. Illustration for the merge process. Bold lines are new Delaunay edges introduced by the MERGE algorithm.

either connecting L_1 to R or R_1 to L. Step 19 chooses the correct edge by applying the circle criterion. In Fig. 7, K_L contains R_1 in its interior, so we choose the edge (L, R_1). All that we have left to do is to show that the edge (L, R_1) is a Delaunay edge, or equivalently that the circle K_R does not contain any point of V_L in its interior. Since the edge (L, R) is known to be a Delaunay edge, by Lemma 2 there exists a circle K passing through L and R which contains no point of V in its interior. It is also known that the circle K_L contains no point of V_L in its interior. Since the circle K_R lies inside the union of K and K_L , it follows that K_R contains no point of V_L in its interior. We combine this result with the fact that K_R does not contain any point of V_R in its interior to conclude that K_R contains no point of V in its interior. Thus, the edge (L, R_1) is a Delaunay edge. The edge (L, R_1) can now be inserted to replace the edge (L, R). In showing that the next edge to be added after (L, R_1) is a Delaunay edge, the circle K_R plays the same role as the circle K just did.

Now let us analyze the algorithm MERGE. We first note that during the merge process, once an edge is deleted, it will never be reexamined. Since the total number of edges deleted is $O(N)$ and the total number of edges added is also $O(N)$, the time needed for the merge is $O(N)$. Let $t(N)$ denote the time required to construct the Delaunay triangulation for a set of N points. We have the following recurrence relation;

$$t(N) = 2t(N/2) + M(N/2, N/2)$$
$$t(1) = 0$$

where M represents the time required for the merge process. Since $M(N/2, N/2) = O(N)$, $t(N) = O(N \log N)$.

3.2. Iteration

The algorithm presented in this section iteratively triangulates a set of points within a rectangular region. If the point set does not include all four vertices of the rectangle, the missing vertices are implicitly added. The algorithm uses the swapping approach developed by Lawson.[9,10] Since the convex hull of our point set is a rectangle and is known in advance, we need not compute it, as is done in Lawson's algorithm. The initial ordering of the point set also differs from that used by Lawson.

This algorithm was developed with the terrain fitting problem in mind. Terrtain regions are processed in rectangular blocks. Adjacent terrain regions must fit together without any gaps. Once a triangular faceted surface is fit to a terrain region, the accuracy of the fit is usually computed. If the approximation surface does not meet the given accuracy constraints, additional vertices are added and the triangulation is updated. An iterative triangulation algorithm is ideal for updating.

Algorithm TRIBUILD

INITIALIZATION

step 1: Given a set V of N points within a rectangle, remove any points which fall on the vertices of the rectangle.

step 2: Partition the rectangle into approximately $N^{1/2}$ bins (smaller rectangular regions).

step 3: Reorder the points by bins, starting at some bin and proceeding to neighboring bins (see Fig. 8).

step 4: Place the first point into the rectangle. Connect the point to the four corners of the rectangle to produce an initial triangulation.

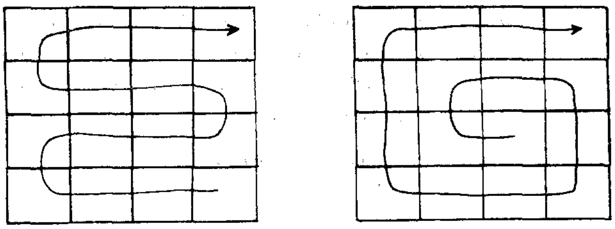


Fig. 8. Two possible bin ordering schemes.

ITERATION

(**Comment:** The remaining points in V will be iteratively added to the triangulation. After each point is added, it will be connected to the vertices of its enclosing triangle. The triangulation will then be restructured so that the MAX-MIN angle criterion is globally satisfied. (See Fig. 9.)

step 1: Input the next point to the existing triangulation. Connect this point to the vertices of its enclosing triangle.

step 2: Step 1 will produce up to four strictly convex quadrilaterals. (Four quadrilaterals only occur when a newly introduced point falls on the edge of the triangulation.) Each of these quadrilaterals has an alternate diagonal. Swap a diagonal with its alternate, if doing so is required to satisfy the

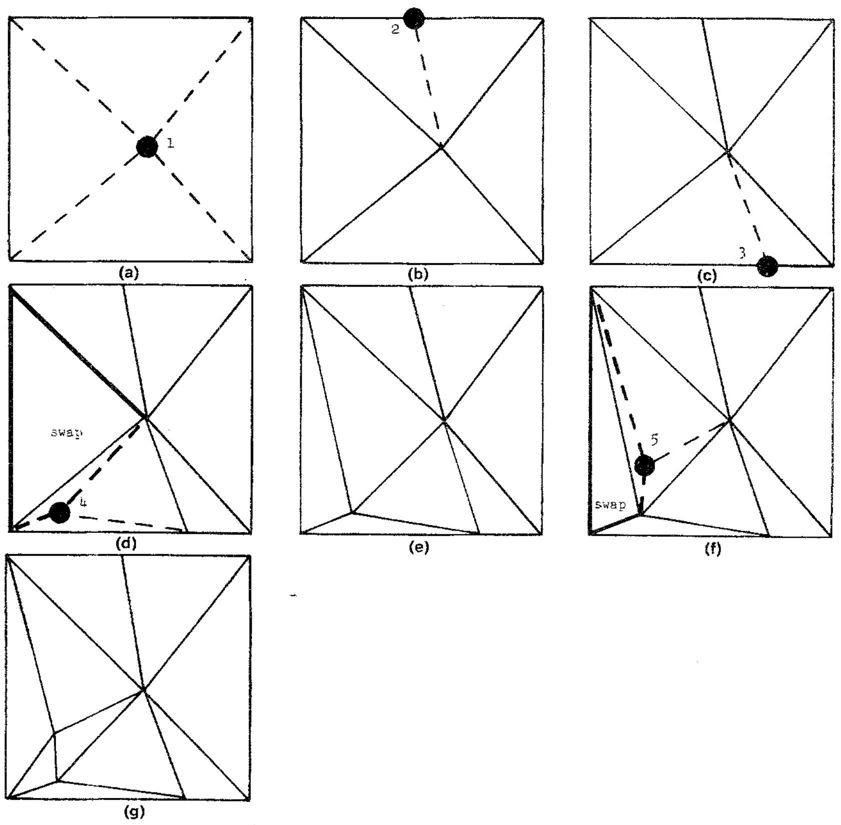


Fig. 9. Example showing a Delaunay triangulation produced by the iterative algorithm. Newly introduced points are shown as filled circles.

MAX-MIN angle criterion within the quadrilateral (i.e., use the LOP within the quadrilateral).

step 3: Each swap performed in step 2 may result in two new quadrilaterals that need to be tested. If one of these quadrilaterals doesn't satisfy the MAX-MIN angle criterion, swap its diagonal with its alternate.

step 4: This swapping procedure may propagate further outward. Lawson(10) has shown that this process will always terminate.

step 5: If all points in V have been used then stop, otherwise go to step 1.
END TRIBUILD

Step 1 requires the identification of the enclosing triangle of a point. This can be accomplished by the following very simple subroutine TRIFIND.(10) The subroutine assumes that the triangulation is stored using a variation of Lawson's data structure (given in Appendix B).

Subroutine TRIFIND

(**Comment:** The triangulation is stored in the form given in Appendix B. This subroutine locates the triangle τ which encloses the point (x_0, y_0).)

(**Comment:** $X(\tau, i)$ denotes the x value of the ith vertex of triangle τ.)

```
        $\tau \leftarrow$ last triangle created
LOOP: DO FOR $I \leftarrow 1$ to 3
          $I$PLUS1 $\leftarrow I(\text{mod } 3) + 1$
          IF $[(y_0 - Y(\tau, I)) * (X(\tau, I\text{PLUS1}) - x_0).\text{GT}.(x_0 - X(\tau, I)) *$
             $(Y(\tau, I\text{PLUS1}) - y_0)]$
          (Comment: If $(x_0, y_0)$ is not in $\tau$, jump to the neighbor of $\tau$
             which is in the direction of the point.
          $\tau \leftarrow N(\tau, I\text{PLUS1})$
          GO TO LOOP
          ENDIF
      END DO FOR
      RETURN ($\tau$)
      END TRIFIND
```

Each iteration of algorithm TRIBUILD requires an $O(N)$ search performed by TRIFIND, followed by an $O(N)$ swapping procedure. Thus, the algorithm is $O(N^2)$.

An empirical examination of algorithm TRIBUILD yields somewhat better results. If the initial point set is uniformly distributed within its enclosing rectangle, then the number of data points in each bin will be approximately $O(N^{1/2})$. Thus the search procedure will be $O(N^{1/2})$. The swapping procedure which updates the triangulation can be propagated many times. We have found that two levels of swaps are nearly always sufficient to globally satisfy the MAX-MIN angle criterion. Thus the algorithm is roughly $O(N^{3/2})$, empirically.

4. EXAMPLES AND APPLICATIONS

4.1. Random Delaunay Triangulation

A two-dimensional Poisson process of intensity λ can be used to describe a random distribution of points in the plane. This process is characterized by the property that the expected number of points within a region of area A is λA, irrespective of the shape or orientation of the region.

Suppose we let the points chosen by a Poisson process seed a Delaunay triangulation. The resulting network of triangles inherits the properties of homogeneity and isotropy from the driving point process.

A random Delaunay triangulation is probably the only nonregular triangulation which is mathematically tractable. Many of its statistical properties have been derived by Miles.[21] Its principal first-order statistics are given below; the paper by Miles also provides the associated second-order statistics.

$$E[\text{cell area}] = \pi/\lambda$$
$$E[\text{cell circumference}] = 32/[3\pi(\lambda/2\pi)^{1/2}]$$
$$E[\text{cell in-radius}] = (8\lambda/\pi)^{-1/2}$$

Miles has also derived the probability density function $f(\alpha)$ for an arbitrary angle α in the triangulation.

$$f(\alpha) = 4\{(\pi - \alpha)\cos\alpha + \sin\alpha\}(\sin\alpha/\alpha)$$

For certain applications, we would like a triangulation with as few small angles as possible. The distribution $f(\alpha)$ provides a characterization of the "goodness" of a triangulation.

4.2. Interpolating Functions of Two Variables

A major application of triangulations is the interpolation of functions of two variables, where the function is initially defined only at an irregular set of locations. These locations are used as the vertices of a triangulation.

The value of the function at a point, other than a vertex, is computed by performing an interpolation within the triangle containing the point. A triangulation composed of nearly equiangular triangles is considered good for this purpose. McLain(19) and Lawson(10) have used the Delaunay triangulation for this purpose. Also see the work of Powell and Sabin.(25)

4.3. Decomposition of Polygons into Convex Sets

An algorithm for decomposing polygons into triangles may be based upon the concept of the Delaunay triangulation. There are applications in pattern recognition and computer graphics for which one wants to combine adjacent triangles into larger convex sets. An $O(mN)$ algorithm for decomposing a nonconvex polygon of N sides and m reflex angles into convex sets is given by Schachter.(30) [An $O(N \log N)$ polygon triangulation algorithm not based upon the Delaunay triangulation is given by Garey *et al.*(5)]

4.4. Terrain Fitting

A rectangular terrain region may be represented by an array of elevation values. A two-dimensional digital filter (e.g., a Wiener filter) can be applied to these data to extract local extrema (i.e., peaks of mountains and "pits" of valleys) and ridge line segments.

We would like the local extrema to be the vertices of a triangulation and the ridge line segments to fall on the edges of the triangulation. This structure can be obtained as follows. Let L denote the set of local extrema and E the set of endpoints of the ridge segments. For each element of $L \cup E$ falling within the smallest circumscribing circle of a ridge segment, construct a normal projection onto the segment. Let P denote the projection point set. Now, let the points in $L \cup E \cup P$ seed a Delaunay triangulation. Each element of $L \cup E \cup P$ has an associated elevation. The triangulation therefore defines a piecewise planar approximation to the terrain surface. For certain applications, an approximated surface must fit the original data to a given error tolerance. If this error bound is not satisfied, additional vertices are added, and the triangulation is updated. An iterative algorithm is well suited for updating.

The above technique assumes that the line segment set is sparse in the plane. A good solution for the more general problem of triangulating any planar set of points and line segments is given by Lee.(12)

A Delaunay triangulation is an excellent choice for the terrain fitting and display problem. Triangles with very small angles produce a poor computer graphics display. Maximizing the minimum angle within the triangulation insures the best possible visualization of the data. Further requirements are met concerning speed of construction.

4.5. Spatial Pattern Models

The Voronoi and Delaunay tessellations have been extensively used to model spatial patterns in a wide range of fields including astronomy[8] biomathematics,[1,18,24,35] computer science[4,11,14,29,31,32] geography,[2,17,27] meteorology,[36] metallurgy,[6] numerical analysis,[10,24,25] and packing and covering.[22,28] We will briefly consider two somewhat representative examples.

Suppose that we are given a collection of sites, where each site has a random variable associated with it. Let these sites seed a Voronoi tessellation. The statistical dependence between (Voronoi) neighboring sites may be specified in terms of the Delaunay edge length between sites and the Voronoi border length between their cells. See the work of Besag[1] for details and references.

Stearns[34] poses the following problem:

> A domain wall in ferromagnetic materials can be considered as a two-dimensional membrane which, when subject to an r.f. field, will oscillate in a manner determined by the boundary conditions. One possible set of boundary conditions would correspond to pinning the wall at impurities whose positions are random in the wall. In describing wall motion, we must know the area distributions of triangles formed from three impurity sites. These triangles will contain no other impurity pinning points in their interior and will be called "good" triangles. What is the probability distribution of the areas of the resulting network of "good" triangles formed by choosing N points distributed uniformly in a given area

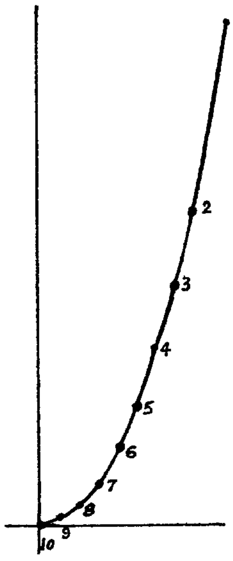
Miles[20] interprets "good" to mean Delaunay, and proposes a solution.

5. DISCUSSION

We have presented two algorithms for constructing the Delaunay triangulation for a set of N points in the Euclidean plane. The first algorithm is based upon a divide-and-conquer approach. It runs in $\theta(N \log N)$ time, which is asymptotically optimal. The second algorithm iteratively adds points to an existing triangulation, updating the triangulation to include each newly introduced point as a vertex. Although it could take $O(N^2)$ time in the worst case, it runs fairly well for the average case.

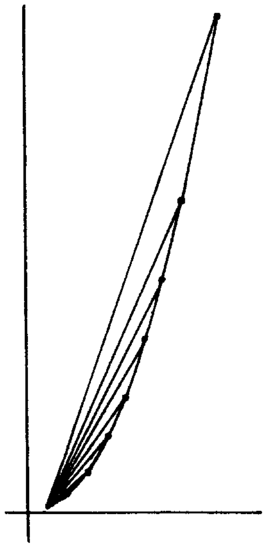
APPENDIX A: AN EXAMPLE FOR WHICH ITERATIVE ALGORITHMS WORK IN $O(N^2)$ WORST CASE TIME

Consider 10 points on the parabola $y = \frac{1}{2}x^2$. The points in the diagram are numbered in the order in which they are added to the existing set.



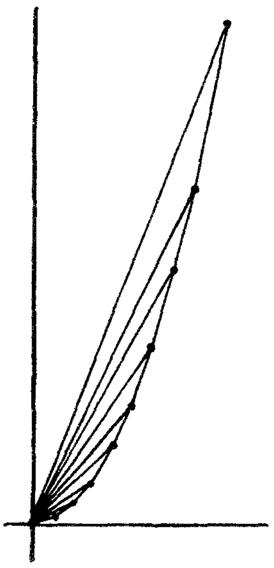
A

Let $S = \{1, 2, ..., 9\}$. $DT(S)$ is given below.



B

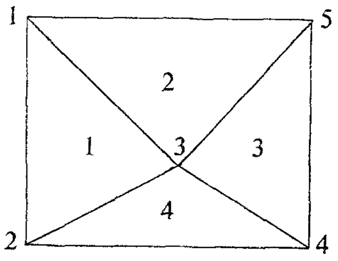
Now when point 10 is added, all the edges incident with point 9 are deleted. The resultant triangulation is given below. Thus, in general the total work involved in updating the triangulation is $\sum_{i=4}^{N}(i-1) = O(N^2)$ in the worst case.



C

APPENDIX B: DATA STRUCTURE FOR A TRIANGULAR NETWORK

The data structure used by the iterative algorithm will be described by an example.



Vertex	X	Y
1	1	1
2	1	31
3	18	17
4	31	31
5	31	1

Triangle	Neighboring triangles (counterclockwise):			Vertices (counterclockwise):		
T	$N(T, 1)$	$N(T, 2)$	$N(T, 3)$	$V(T, 1)$	$V(T, 2)$	$V(T, 3)$
1	2	0	4	1	2	3
2	1	3	0	3	5	1
3	4	0	2	4	5	3
4	1	0	3	2	4	3

The following conventions are used: Triangles $N(T, 1)$, $N(T, 2)$, and T meet at vertex $V(T, 1)$. Zero denotes a null triangle.

ACKNOWLEDGMENTS

We would like to thank S. Dix, C. L. Lawson, D. Milgram, R. E. Stearns, and one of the reviewers for their thoughtful comments on this paper.

REFERENCES

1. J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *J. Royal Stat. Soc. B* 36(2):192–236 (1974).
2. D. J. Bogue, "The Structure of the Metropolitan Community," University of Michigan Contributions of the Institute of Human Adjustment Social Science, University of Michigan, Ann Arbor, Michigan (1949).
3. B. Delaunay, "Sur la sphère vide," *Bull. Acad. Science USSR* VII: *Class. Sci. Mat. Nat.* 793–800 (1934).
4. H. Fuchs and Z. M. Kedem, "The Highly Intelligent Tablet as an Efficient Printing Device for Interactive Graphics," University of Texas (Dallas) (1979). Program in Math. Science (1979).
5. M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan, "Triangulating a simple polygon," *Inform. Proc. Letters* 7:175–179 (June 1978).
6. E. N. Gilbert, "Random subdivision of space into crystals," *Ann. Math. Stat.* **33**: 958–972 (1962).
7. P. J. Green and R. Sibson, "Computing Dirichlet tesselations in the plane," *Computer J.* **21**(2):168–173 (July 1978).
8. T. Kiang, "Random fragmentation in two and three dimensions," *Z. Astrophysik* **64**:433–439 (1966).

9. C. L. Lawson, "Generation of a Triangular Grid with Applications to Contour Plotting," Technical Memo. 299, Jet Propulsion Laboratory, Pasadena, California (February 1972).
10. C. L. Lawson, "Software for C^1 surface interpolation," *in Mathematical Software III*, J. Rice, Ed. (Academic Press, New York, 1977).
11. D. T. Lee, "On Finding k-Nearest Neighbors in the Plane," Technical Report Eng. 76-2216, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois (1976).
12. D. T. Lee, "Proximity and Reachability in the Plane," Ph.D. Thesis, Coordinated Science Laboratory Report ACT-12, University of Illinois, Urbana, Illinois (1978).
13. D. T. Lee, "Two-dimensional Voronoi diagrams in the L_p-metric," *J. ACM* (accepted for publication).
14. D. T. Lee and C. K. Wong, "Voronoi diagrams in L_1 (L_∞) metrics with two-dimensional storage applications," *SIAM J. Computing* (accepted for publication).
15. B. A. Lewis and J. S. Robinson, "Triangulation of planar regions with applications," *Computer J.* **21**(4):324–332 (Nov. 1978).
16. E. L. Lloyd, "On Triangulation of a Set of Points in the Plane," Technical Report MIT/LCS/TM-88, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts (May 1977).
17. P. E. Lloyd and P. Dicken, *Location in Space: A Theoretical Approach to Economic Geography* (Harper and Row, New York, 1972).
18. B. Matern, "Spatial variation," *Medd. Statens Skogsforskninginstit.* **36**(5):1–144 (1960).
19. D. H. McLain, "Two dimensional interpolation from random data," *Computer J.* **19**(2):178–181 (1976); Errata, *Computer J.* **19**(4):384 (1976).
20. R. E. Miles, "Solution to problem 67-15 (Probability distribution of a network of triangles)," *SIAM Rev.* **11**:399–402 (1969).
21. R. E. Miles, "On the homogeneous planar Poisson point-process," *Math. Biosciences* **6**:85–127 (1970).
22. J. Molnar, "Packing of congruent spheres in a strip," *Acta Math. Acad. Sciences Hungaricae Jomus* **31**(1–2):173–183 (1978).
23. J. M. Nelson, "A triangulation algorithm for arbitrary planar domains," *Appl. Math. Modelling* **2**:151–159 (September 1978).
24. E. C. Pielou, *Mathematical Ecology* (Wiley, New York, 1977).
25. M. J. D. Powell and M. A. Sabin, "Piecewise quadratic approximations on triangles," *ACM Trans. Math. Software* **3**(4):316–325 (December 1977).
26. F. P. Preparata and S. J. Hong, "Convex hulls of finite sets of points in two and three dimensions," *Comm. ACM* **20**(2):87–93 (February 1977).
27. D. Rhynsburger, "Analytic delineation of Thiessen polygons," *Geographical Analysis* **5**(2):133–144 (April 1973).
28. C. A. Rogers, *Packing and Covering* (Cambridge University Press, 1964).
29. B. Schachter, A. Rosenfeld, and L. S. Davis, "Random mosaic models for textures," *IEEE Trans. Systems, Man, and Cybernetics* **8**(9):694–702 (September 1978).
30. B. J. Schacter, "Decomposition of polygons into convex sets," *IEEE Trans. Computers* *C*-**72**(11):1078–1082 (November 1978).
31. M. I. Shamos, *Computational Geometry* (Springer-Verlag, New York, 1977).
32. M. I. Shamos and D. Hoey, "Closest-point problems," *Proceedings of the 16th Annual Symposium on the Foundations of Computer Science*, pp. 151–162 (October 1975).
33. R. Sibson, "Locally equiangular triangulations," *Computer J.* **21**(3):243–245 (August 1978).

34. B. Stears, "Probability distribution of a network of triangles (Problem 67-15)," *SIAM Rev.* **11**:399 (1969).
35. P. Switzer, "Reconstructing patterns from sampled data," *Ann. Math. Stat.* **38**:138–154 (1967).
36. A. H. Thiessen, "Precipitation averages for large areas," *Monthly Weather Rev.* **39**:1082–1084 (1911).
37. G. Voronoi, "Nouvelles applications des parametres continus à la théorie des formes quadratiques. Deuxième Mémoire: Recherches sur les parallelloedres primitifs," *J. Reine Angew. Math.* **134**: 198–287 (1908).