VORONOI DIAGRAMS FROM CONVEX HULLS *

Kevin Q. BROWN
Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.

Received 12 December 1978

Analysis of algorithms, computational complexity, computational geometry

1. Introduction

The problem of construction of planar Voronoi diagrams arises in many areas, one of the most important of which is in nearest neighbor problems. This includes clustering [14], contour maps [6] and (Euclidean) minimum spanning trees [23]. Shamos [22] gives several more applications.

An $\Omega(N \log N)$ time worst case lower bound can be shown for this problem by reducing it to sorting [21]. The challenge is to construct an $O(N \log N)$ time algorithm. Shamos [21] and Shamos and Hoey [23] describe an $O(N \log N)$ time divide-and-conquer algorithm for construction of the planar Euclidean Voronoi diagram. Lee and Wong [16] describe an $O(N \log N)$ time algorithm for the L_1 and L_∞ metrics in the plane, and Drysdale and Lee [8] present an $O(NC^{(\log N)^{1/2}})$ time algorithm for the Voronoi diagram of N line segments (which they have since improved to $O(N(\log N)^2)$ time). Shamos [21], Lee and Preparata [15], and Lipton and Tarjan [17] have produced fast algorithms for searching a Voronoi diagram (or any other straight-line planar graph).

In this paper we describe an $O(N \log N)$ time algorithm for constructing a planar Euclidean Voronoi diagram which extends straightforwardly to higher dimensions. The fundamental result is that a K-dimensional Euclidean Voronoi diagram of N points can be constructed by transforming the points to K + 1-space, constructing the convex hull of the transformed points, and then transforming back to K-space.

Sections 2 through 4 give background material which is essential for understanding of the algorithm which is given in Section 5. Section 2 describes the important features of Voronoi diagrams. Sections 3 and 4 describe the two major tools used in the algorithm – convex hulls and the inversion transform. Following the planar algorithm of Section 5, we then explore fast expected time algorithms and higher dimensions.

2. Definition of planar Voronoi diagrams

Let S be a set of N planar points. A *nearest point* planar Voronoi diagram of S, as pictured in Fig. 1, is a polygonal network of N regions. For each point i of S, region i is the set of all points of the plane which are closer to point i than the other N – 1 points of S. Thus, given an arbitrary point P in the plane, one can determine which of the N points of S is closest to P by determining which of the N regions contains point P. The vertices of these polygonal regions are called *Voronoi points* and the polygonal boundaries of the regions are called *Voronoi polygons.* If a Voronoi polygon is bounded, then it is constructed entirely from edges of the Voronoi diagram. If it is unbounded, then it includes two rays of the diagram.

Each Voronoi point V of the nearest point diagram is equidistant from the three points of S which are nearest V. This yields a property of Voronoi diagrams which is exploited in the algorithm of Section 5.

* This research was partially supported by the Office of Naval Research under contract number N00014-76-C-0370.

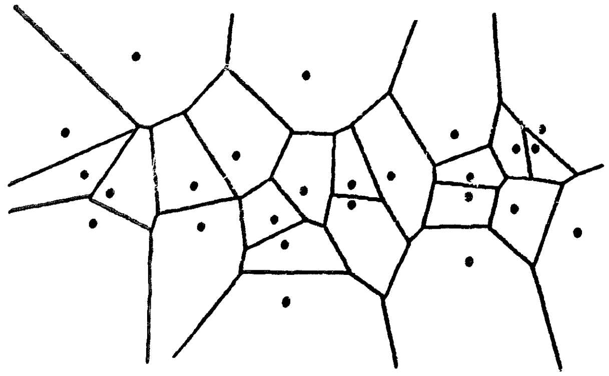
Fig. 1. Planar nearest point Voronoi diagram of N points.

The circle determined by the three nearest neighbors of V is centered at V and furthermore will not contain any of the other N – 3 points of S in its interior. The converse is also true: If the circle determined by three points of S does not contain any of the N – 3 other points of S, then the center of that circle is a Voronoi point. The edges of a Voronoi diagram connect pairs of Voronoi points whose corresponding circles meet at two common points of S. The rays are determined similarly by one circle from a nearest Voronoi point and one circle from a farthest Voronoi point (described below).

A *farthest point* planar Voronoi diagram is also a network of polygonal regions. But region i is the set of all points in the plane which are *farther* from point i than any other point of S. As for the nearest point diagram, there is a set of circles which define the Voronoi points for a farthest point diagram. Each farthest point circle passes through three of the N points of S but the interior of a farthest point circle contains *all* of the other N – 3 points rather than none of them. It is important to note that only points which are vertices of the convex hull of the N points of S have nonempty farthest point regions. This is because each Voronoi point V of the farthest point diagram must be equidistant from the three points of S which are *farthest* from V. It is not possible to construct such a Voronoi point from points of S which are not on the convex hull.

Since a nearest (farthest) point Voronoi diagram is a planar graph, the number of Voronoi points is at most 2N – 4 and the number of edges is at most 3N – 6 for N > 2 [13]. Shamos [21,22] and Shamos and Hoey [23] give more information on Voronoi diagrams.

3. Convex hulls

The convex hull of a set of N points is defined (nonconstructively) as the smallest convex set which contains all of the points. In the plane this is a convex polygon of at most N sides. Graham [11], Preparata and Hong [19], and Shamos [22] give O(N log N) worst case time algorithms for constructing the convex hull of N points in the plane. Preparata and Hong [19] have also described an O(N log N) time algorithm for finding the convex hull of N points in 3-space. However, in four dimensions there is an $\Omega(N^2)$ worst case lower bound because the convex hull can have $\theta(N^2)$ edges [12, p.193]. Chand and Kapur [5] describe a convex hull algorithm for an arbitrary number of dimensions whose complexity has yet to be analyzed.

4. The inversion transform

The algorithm makes use of a geometric transform called inversion. We will first describe the two-dimensional case and then generalize to three (and higher) dimensions. For more information on inversion the reader is referred to [7].

The inversion transform is determined by two parameters:

(1) the center of inversion, and

(2) the radius of inversion.

For simplicity of exposition it shall be assumed for now that the center of inversion is the origin and that the radius of inversion is one. (Furthermore, this section describes inversion in terms of polar coordinates rather than cartesian coordinates to more simply illustrate its properties.) If a point P has polar coordinates (R, θ), then the inversion transform of P is

$$(R, \theta) \to (1/R, \theta) .$$

Inversion maps a vector in the direction θ to another vector in the same direction but with its magnitude 'inverted'. Note that inversion is involutory – application of inversion twice yields the original point. An important property of inversion is that a circle which passes through the center of inversion transforms to a line which *does not* pass through the center of inversion, and vice versa. Furthermore, the interior of the circle transforms to one of the half-planes determined by that line and the exterior of the circle transforms to the other half-plane. The properties of inversion in three dimensions are analogous. The transform can be expressed in spherical coordinates as

$$(R, \theta, \phi) \rightarrow (1/R, \theta, \phi) .$$

This transform is involutory, as in two dimensions, and it also transforms any *sphere* which passes through the center of inversion to a *plane* which does not pass through the center of inversion. The interior of the sphere transforms to a half-space bounded by that plane and the exterior of the sphere transforms to the other half-space.

5. Planar Voronoi diagram algorithm

The tools described in the previous sections are here combined to produce an O(N log N) time algorithm for constructing a Voronoi diagram of a set S of N planar points. The algorithm takes advantage of the fact that the Voronoi points of the nearest point diagram can be represented by a set of circles which each

(i) pass through three of the N points of S, and

(ii) do not contain any of these N points in the interior (Section 2).

The object is to generate the circles and determine which pairs of circles correspond to edges of the diagram. All of this information is readily obtained from the convex hull of a set of points S' which is generated by applying inversion to the N points of S. Furthermore, the circles which define the farthest point diagram can be obtained from the same convex hull.

Algorithm for construction of a planar Voronoi diagram

1. Let S be a set of N planar points located in the xy plane of 3-space. Pick a point P in 3-space which is not in the xy plane.
2. Choose any radius of inversion R > 0 and then invert the N points of S with respect to point P and radius R. Call this new set of N points S'.
3. Construct the convex hull of the points in S' in O(N log N) time (by the algorithm of Preparata and Hong [19]). All N of the points of S' will be on the convex hull because inversion about P maps all points of the xy plane to a sphere with P at the apex (Fig. 2).
4. **Comment:** The convex hull has only O(N) faces F_i and edges E_{ij} because a planar graph of $N > 2$ vertices has at most $2N - 4$ regions (faces) and at most $3N - 6$ edges [13]. Each of the O(N) faces corresponds to a Voronoi point of either the nearest or farthest point Voronoi diagram. Each of the O(N) edges corresponds to an edge (or ray) of the nearest or farthest point Voronoi diagram.
5. Each of the O(N) faces F_i of the convex hull determines a plane in 3-space. Invert these planes (with respect to center of inversion P and radius R) obtaining O(N) spheres which intersect the xy plane in O(N) circles. The centers of these circles are the Voronoi points V_i. To distinguish nearest and farthest Voronoi points perform the following simple test:

   For each face F_i (corresponding to Voronoi point V_i) associate the half-space H_i which contains the convex hull and whose boundary plane contains face F_i. If half-space H_i contains point P, then V_i is a Voronoi point of the nearest point Voronoi diagram. Otherwise, V_i is a Voronoi point of the farthest point diagram.
6. To construct the diagram one needs to know how to connect the Voronoi points with line segments and how to construct the rays bounding the infinite Voronoi regions. This is simple because each edge E_{ij} of the convex hull corresponds to an edge (or ray) of the nearest or farthest point diagram, and vice versa. The only problem is to determine for each edge E_{ij} whether it corresponds to a segment of the nearest point diagram, a segment of the farthest point diagram, or a ray (for both diagrams). Let F_i and F_j be the faces bounding edge E_{ij} of the convex hull and let V_i and V_j be the corresponding Voronoi points.

- If V_i and V_j are both nearest Voronoi points, then there is a line segment connecting V_i and V_j in the nearest point diagram.
- If V_i and V_j are both farthest Voronoi points, then there is a line segment connecting V_i and V_j in the farthest point diagram.

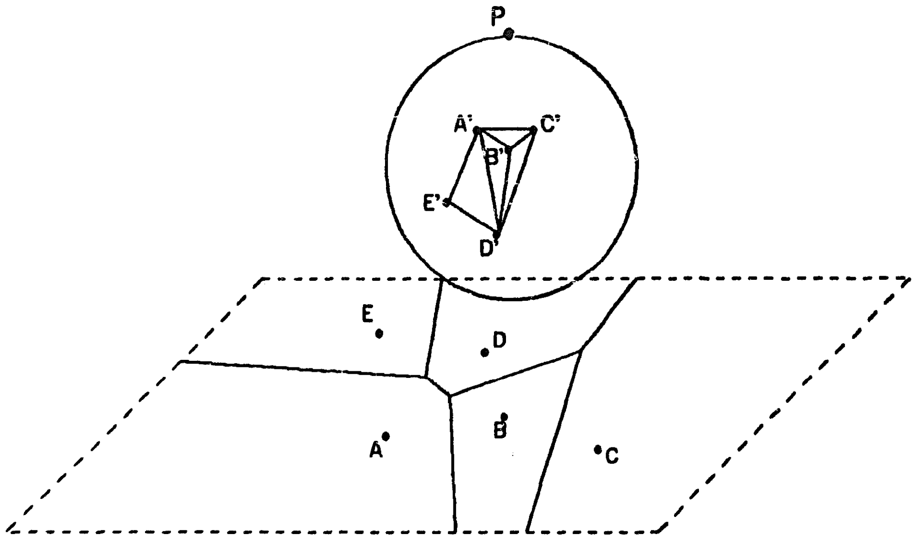

Fig. 2. Planar Voronoi diagram and corresponding convex hull.

- If V_i is a closest Voronoi point and V_j is a farthest Voronoi point, then V_i and V_j determine a ray in both the nearest and farthest point Voronoi diagrams. The points V_i and V_j determine a line, and the desired ray for the nearest point Voronoi diagram is the part of that line which starts at point V_i and does not include point V_j. The ray starting at point V_j which does not include point V_i is for the farthest point Voronoi diagram.

Although it is clear that the above algorithm requires only O(N log N) time and O(N) storage, it is not immediately obvious that it actually constructs the nearest (or farthest) point Voronoi diagram. It remains to be explained

(1) why the centers of the circles (generated in Step 5 above) are the Voronoi points and

(2) why the connection rules (Step 6) for Voronoi points work.

We will now outline the proof for the case of the nearest point diagram. The argument for the farthest point diagram is similar.

To prove that the circles generated in Step 5 are centered at the Voronoi points it is sufficient to show that

(a) these circles each pass through three of the N points of S and

(b) do not contain any of the other $N - 3$ points in the interior (Section 2).

Part (a) follows from the fact that inversion is involutory (Section 4). Part (b) is shown by contradiction: If the circle passing through points A, B and C of S contains another point $Q \in S$ in its interior, then the convex hull of the transformed points S' will not contain a (nearest Voronoi point) face A'B'C' because of the presence of point Q'. To prove that Step 5 generates all of the Voronoi points we simply use the reverse argument. If V_i is a Voronoi point, then the sphere defining the circle for V_i transforms (by inversion) to a plane containing a face of the convex hull of S'.

Step 6 of the algorithm obtains the edges of the Voronoi diagram directly from the edges of the (three-dimensional) convex hull. An edge E_{ij} of the convex hull which separates (nearest point) faces F_i and F_j maps to a line segment between Voronoi points V_i and V_j. But the circles corresponding to Voronoi points V_i and V_j meet at two of the N points of S because the corresponding faces F_i and F_j share an edge E_{ij}. This is exactly the characterization given for edges of the Voronoi diagram in Section 2. Similarly, the rays are determined by edges E_{ij} where V_i is a nearest Voronoi point and V_j is a farthest Voronoi point (or vice versa). In this case, too, the circles corresponding to V_i and V_j meet at two of the N points of S. Thus, since the Voronoi points and edges (and rays) connecting the Voronoi points are correctly generated by the above algorithm, the Voronoi diagram is constructed in O(N log N) time.

6. Fast expected-time algorithms

The most expensive part of the algorithm for construction of a Voronoi diagram is the construction of the convex hull. Thus, if the convex hull can be constructed in fast *expected* time, then the Voronoi diagram can be constructed in fast expected time. However, the O(N) expected time algorithms of Bentley and Shamos [1], Eddy [9] or Floyd [10] do not apply because their result depends on a sublinear expected number of points on the convex hull, and for the Voronoi diagram algorithm there are always N vertices on the convex hull.

Bentley, Weide and Yao [2], on the other hand, describe how a planar Voronoi diagram can be constructed in linear expected time. The only condition is that the probability density of the underlying distribution must be bounded above and below by (nonzero) constants. The algorithm does not make use of inversion. Instead, it applies an extension of Weide's [24] technique for an O(N) expected time sort to the planar Voronoi diagram problem.

7. Higher dimensions

The K-dimensional Voronoi diagram algorithm is an extension of the planar algorithm. One first embeds the N K-dimensional points of S in K + 1-space and then inverts them to N K + 1-dimensional points S'. The convex hull of S' is constructed and then the Voronoi diagram is obtained by transforming the parts of the convex hull back to K-space. The transform back to K-space first inverts each hyperface of the convex hull to obtain a set of K + 1-spheres whose intersection with K-space is a set of K-spheres. These K-spheres each pass through K + 1 points of S and are centered at the Voronoi points. The other components of the K-dimensional Voronoi diagram are obtained by connection rules similar to those in Step 6 of the algorithm in Section 5. For example, if the K-sphere for Voronoi point V_i passes through K of the K + 1 points determining the K-sphere for Voronoi point V_j, then a 1-dimensional edge is drawn between V_i and V_j. If the spheres for a set of three or more Voronoi points share K – 1 points of S, then a 2-dimensional edge is drawn between the Voronoi points of that set. (A 2-dimensional edge between L points is a convex polygon with L vertices.) The rules for three and higher dimensional edges are similar. The time complexity of the K-dimensional Voronoi diagram algorithm is dominated by the time to construct a K + 1-dimensional convex hull of N points. In particular, a Voronoi diagram in 3-space corresponds to a convex hull in 4-space, which may have $\Omega(N^2)$ edges (Section 3). Preparata's $\Omega(N^2)$ worst case lower bound for the 3-dimensional Voronoi diagram [18] is thus not unexpected.

8. Conclusion

The use of a geometric transform has been shown to be very useful for construction of a fast algorithm for generating a planar Voronoi diagram. Geometric transforms enable fast algorithms for other geometric problems also. The intersection of N half-spaces in E^3 can be constructed in O(N log N) time through use of a duality transform between points and planes [3,20, 25]. The Euclidean diameter of a set of N points in 3-space can be determined in O(N log N) time through use of the same duality transform applied to a convex hull [4]. The union of a set of N arbitrary planar circles can be constructed in O(N log N) time by using inversion to transform the circles to a set of N half-3-spaces which are then intersected in O(N log N) time. Geometric transforms are becoming a powerful tool in the construction of geometric algorithms.

Acknowledgment

Mike Shamos introduced Voronoi diagrams to me and Jon Bentley made several helpful comments which improved the presentation of this paper.

References

[1] J.L. Bentley and M.I. Shamos, Divide and conquer for linear expected time, Information Processing Lett. 7 (2) (1978) 87–91.

[2] J.L. Bentley, B.W. Weide and A.C. Yao, Optimal expected-time algorithms for closest-point problems, Allerton Conference (1978).

[3] K.Q. Brown, Fast intersection of half-spaces, Rep. CMU-CS-78-129, Carnegie-Mellon University, Computer Science Department (1978).

[4] K.Q. Brown, Geometric transforms for fast geometric algorithms, Thesis proposal at Carnegie-Mellon University, Computer Science Department (1978).

[5] D.R. Chand and S.S. Kapur, An algorithm for convex polytopes, J. ACM 17 (1) (1970) 78–86.

[6] J.C. Davis and M.J. McCullagh, Display and Analysis of Spatial Data (Wiley, New York, 1975).

[7] C.W. Dodge, Euclidean Geometry and Transformations (Addison-Wesley, Reading, MA, 1972).

[8] R.L. Drysdale III and D.T. Lee, Generalized Voronoi diagram in the plane, Allerton Conference (1978).

[9] W. Eddy, A new convex hull algorithm for planar sets, ACM Trans. Math. Software 3 (4) (1977) 398–403.

[10] R. Floyd, personal communication from Jon Bentley.

[11] R.L. Graham, An efficient algorithm for determining the convex hull of a planar set, Information Processing Lett. 1 (4) (1972) 132–133.

[12] B. Grunbaum, Convex Polytopes (Wiley, New York, 1967).

[13] F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1969).

[14] J.A. Hartigan, Clustering Algorithms (Wiley, New York, 1975).

[15] D.T. Lee and F.P. Preparata, Location of a point in a planar subdivision and its applications, SIAM J. Comput. 6 (3) (1977) 594–606. Also: Proc. Eight Annual ACM Symposium on Automata and Computability Theory.

[16] D.T. Lee and C.K. Wong, Voronoi diagrams in L_1 (L_∞) metrics with 2-dimensional storage applications, Rep. RC 6848 (#29359), IBM T.J. Watson Research Center (1977).

[17] R.J. Lipton and R.E. Tarjan, Applications of a separator theorem for planar graphs, in: Proc. 18th Annual IEEE Symposium on Foundations of Computer Science (1977).

[18] F.P. Preparata, Editor, Steps into Computational Geometry, Rep. R-760 UILU-ENG 77-2207, Coordinated Science Laboratory, Applied Computation Theory Group, University of Illinois, Urbana (1977).

[19] F.P. Preparata and S.J. Hong, Convex hulls of finite sets of points in two and three dimensions, Comm. ACM 20 (2) (1977) 87–93.

[20] F.P. Preparata and D.E. Muller, Finding the intersection of a set of n half-spaces in time O(n log n), Rep. R-803 UILU-ENG 77-2250, Coordinated Science Laboratory, Applied Computation Theory Group, University of Illinois, Urbana (1977).

[21] M.I. Shamos, Geometric complexity, in: Proc. 7th Annual ACM Symposium on Automata and Computability Theory (1975) 224–233.

[22] M.I. Shamos, Computational geometry, Ph.D. Thesis, Yale University, Department of Computer Science (May 1978). To appear as: Computational Geometry (Springer, Berlin).

[23] M.I. Shamos and D. Hoey, Closest-point problems, in: Proc. 16th Annual IEEE Symposium on Foundations of Computing (1975) 151–162.

[24] B.W. Weide, Statistical methods in algorithm design and analysis, Ph.D. Thesis, Carnegie-Mellon University, Department of Computer Science (1978).

[25] J. Zolnowsky, Topics in computational geometry, Ph.D. Thesis, Stanford University, Department of Computer Science (1978).