BUILDING VORONOI DIAGRAMS FOR CONVEX POLYGONS IN LINEAR EXPECTED TIME

L. Paul Chew

Technical Report PCS-TR90-147

Building Voronoi Diagrams for Convex Polygons in Linear Expected Time

L. Paul Chew
Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

Keywords: computational geometry, Voronoi diagrams, convex polygons

Abstract

Let P be a list of points in the plane such that the points of P taken in order form the vertices of a convex polygon. We introduce a simple, linear expected-time algorithm for finding the Voronoi diagram of the points in P. Unlike previous results on expected-time algorithms for Voronoi diagrams, this method does not require any assumptions about the distribution of points. With minor modifications, this method can be used to design fast algorithms for certain problems involving unrestricted sets of points. For example, fast expected-time algorithms can be designed to delete a point from a Voronoi diagram, to build an order k Voronoi diagram for an arbitrary set of points, and to determine the smallest enclosing circle for points at the vertices of a convex hull.

Introduction

The Voronoi diagram is one of the most useful data structures in computational geometry. Given n data points in the plane, the Voronoi diagram partitions the plane into n regions, one associated with each data point. The region associated with data point p consists of all point in the plane that lie closer to p than to any of the other n-1 data points. An optimal O(n log n) time algorithm for building a Voronoi diagram was first presented in [8].

We present a simple, linear expected-time algorithm for building the Voronoi diagram for the set of points at the vertices of a convex polygon. We include a number of applications of this technique: deleting a point from a Voronoi diagram, building an order k Voronoi diagram for an arbitrary set of points, building the farthest point Voronoi diagram and determining the smallest enclosing circle for points at the vertices of a convex hull, and building a line segment Voronoi diagram (a skeleton, in the terminology of [5]) for a convex polygon.

Bentley, Weide, and Yao [2] have presented results on expected-time algorithms for Voronoi diagrams. They showed that the Voronoi diagram for a set S of points can be built in linear expected time provided the points of S are uniformly distributed in the unit square. Results presented in this paper are less general in the sense that for us the data points must form a convex polygon; however, no extra assumptions are needed regarding the distribution of the points.

Recently, Aggarwal and Shor [1] have developed a worst-case linear time algorithm for this same problem. Although their result is of great theoretical interest, the simple, linear expected-time algorithm presented here is far easier to implement and is likely to run faster in practice.

Fast Voronoi Diagrams for Convex Polygons

Let P be a list of points in the plane such that the points of P taken in order form the vertices of a convex polygon. Assume P lists the vertices of the polygon in clockwise order. It is relatively easy to construct an example to show that if the Voronoi diagram is built taking the points in order, one point at a time, then it may require time proportional to n^2 to build the diagram in the worst-case (for example, consider points along one branch of a parabola). We show that by finding neighboring points off-line and by inserting the points into the diagram in random order, the Voronoi diagram for P can be determined in linear expected time.

Notation. V(S) represents the Voronoi diagram on a set S of points. $V(p_1,\ldots,p_i)$ represents the Voronoi diagram on the points $p_1,\ldots,p_i$.

Algorithm - building a Voronoi diagram for vertices of a convex polygon.

1. Choose a random order for the points of P. Let $r_1,\ldots,r_n$ represent the points of P in their random order.
2. For each point r_i $(i>1)$, determine a point a_i in $\{r_1,\ldots,r_{i-1}\}$ such that a_i is a neighbor of r_i; that is, a_i shares a Voronoi boundary with r_i in $V(r_1,\ldots,r_i)$.
3. Create $V(r_1)$. For each point r_i $(i>1)$ insert r_i into $V(r_1,\ldots,r_{i-1})$ by following the boundary of its Voronoi region starting with the border between a_i and r_i.

Analysis.

Step 1 obviously runs in linear time.

In step 2 we exploit the simple structure of a convex-polygon Voronoi diagram to determine neighboring points off-line. Here, it is useful to turn the problem backward, pretending that we are eliminating points one at a time from V(P) the completed Voronoi diagram. The convex-polygon Voronoi diagram has such a simple structure that neighbors can be easily determined. Note that if, for instance, p_3 is eliminated then p_2 and p_4 are neighbors in $V(P-\{p_3\})$; if p_4 were then eliminated, p_2 and p_5 would be neighbors, etc.

A simple doubly linked list can be used to keep track of the neighbors of each point still in the diagram. As a point r_i is eliminated, its neighbors are found, one of these neighbors is chosen to be a_i, and the links of both neighbors are adjusted so the eliminated point no longer appears in the list. This clearly runs in linear time.

For step 3, consider a Voronoi diagram V(S) and a new point q to be inserted into V(S). Let Q represent the region of q in $V(S \cup \{q\})$ and let m be the number of edges of Q. If we know one edge of Q then the remaining edges can be found in O(m) time by scanning along the edges of V(S) that lie in the interior of Q. The edges in the interior of Q form a tree (one vertex of Q is the root, the remaining m-1 vertices are the leaves) containing at most 2m-3 edges; thus, the scan for region Q can be completed in O(m) time. In other words, a new point can be inserted into a Voronoi diagram in time proportional to the number of edges in the new region, provided that we already know one of the edges of the new region.

To see that all of step 3 can be done in linear expected time, imagine working step 3 backward, going from $V(r_1,\ldots,r_n)$ to $V(r_1)$. Since $r_1,\ldots,r_n$ is a randomly chosen list, each substep of (backward) step 3 starts by randomly choosing a region of a planar graph. For any planar graph, an application of Euler's formulas shows that the expected number of edges for a region of the graph is < 6. Thus, a region chosen in step 3 (forward or backward) is expected to have < 6 edges. Since, from step 2, we already know one edge of each new region, the time for each (forward) substep is proportional to the number of edges, and the total expected time for step 3 is $O(n)$.

Theorem. Let P be a list of points in the plane such that the points taken in order form the vertices of a convex polygon. The Voronoi diagram for the points of P can be found in linear expected time.

Applications

1. Deleting a point from a Voronoi diagram.

The technique introduced here for convex polygons can be used to delete a point from a Voronoi diagram in $O(m)$ expected time where m is the number of neighbors of the deleted point. To delete a point we need to fill in its former region with edges from the Voronoi diagram of its neighbors. Previously, the best method was to apply the general Voronoi-diagram algorithm requiring $O(m \log m)$ time.

Note that the neighboring points do not necessarily form a convex polygon, so our technique requires some minor modification. To keep the time linear it is sometimes necessary to check both neighbors during the off-line neighbor calculation (step 2).

To see why the modification is required, consider a point q where q is to be eliminated from a Voronoi diagram and let P be the polygon formed by the neighbors of q. Again, it is useful to consider the backward problem of eliminating the points of P, one at a time, this time from the Voronoi diagram of PU{q}. At each stage, adjacent points in P are neighbors in the Voronoi diagram, except in the case where the region of q is infinite. In this case there is one pair of points, such that, although the points are adjacent in P, the points are not neighbors in the Voronoi diagram; these are the points that are separated by the infinite region of q. Sometimes such points become adjacent when q is eliminated, but this does not necessarily occur. The goal is to do the neighbor calculation (step 2 of the algorithm) in such a way that these nonneighbor, adjacent points are not proclaimed as neighbors.

A simple way to find the best neighbor point a_i for the point r_i in step 2 of the algorithm, is to determine both points adjacent in $P-\{r_n,\ldots,r_{i+1}\}$ then choose one without this infinite region problem. A prospective neighbor point a_i is OK if the point q is on the interior side (interior of $P-\{r_n,\ldots,r_{i+1}\}$) of the segment from r_i to a_i. To see that this simple test is adequate, let P_i be the set $P-\{r_n,\ldots,r_{i+1}\}$, and note that a nonneighbor problem can occur only when q is on the convex hull of P_iU{q} (this is the only way that the region of q can be infinite). It is easy to show that the points of P_i are given by order of angle around q. Thus, if q is not on the convex hull of P_iU{q} then q is on the interior side of all line segments of P_i; if q is on the convex hull of P_iU{q} then q is on the exterior side of exactly one line segment of P_i. This one line segement with q on its exterior side connects the two adjacent vertices that should not be proclaimed as neighbors. Thus, simply testing for the location of q relative to a line segment allows the correct neighbor to be chosen, giving linear expected time for the entire algorithm.

An alternate method depends upon probability. Leave step 2 of the algorithm unaltered; if we happen to get a bad choice for the neighbor of point r_{k+1} then it can take up to O(k) time to insert that point into the diagram. Note, though, that since the order of the points is random, there is only a 1/k chance that r_{k+1} is separated from one of its neighbors by the infinite region of q (that is, if q even has an infinite region). Thus, the expected time for inserting a point into the diagram is still constant.

2. Order k Voronoi diagrams.

For a set S of points in the plane, the order k Voronoi diagram for S divides the plane into regions such that (1) each region R corresponds to a k-size subset H of S, and (2) every point of region R is closer to some point in H than to any point of S not in H. Order k Voronoi diagrams first appeared in [8]. Lee [6] has shown that, for a set S of size n, the order k Voronoi diagram for S has O(k(n-k)) regions. Further, he has shown that the diagram can be built in $O(k^2 n \log n)$ time. Chazelle and Edelsbrunner [3] have presented algorithms using a transfomation to 3 dimensions and taking time $O(n^2 \log n + k(n-k)\log^2 n)$, space O(k(n-k)), or time $O(n^2+k(n-k)\log^2 n)$, space $O(n^2)$.

Our algorithm for building order k Voronoi diagrams is based on Lee's method. The basic step is essentially the deletion of a point from a Voronoi diagram. Using the O(m) expected time algorithm for deleting a point, the order k Voronoi diagram for a set of n points can be built in $O(k^2n + n \log n)$ expected time. This can be done using O(kn) space (more space is needed if it is necessary to allow fast access to the list of data points associated with a region).

3. Farthest point Voronoi diagrams.

The farthest point Voronoi diagram is equivalent to the order (n-1) Voronoi diagram [8]. Nonempty regions of the farthest point Voronoi diagram of S correspond to points on the convex hull of S. A slight modification of our techniques can be applied to produce the farthest point Voronoi diagram directly from the convex hull in O(m) expected time, where m is the number of points on the convex hull. For many applications, the convex hull can be built in linear time (example: for points sorted by x-coordinate, connect the points in order and use the "Graham-scan" [4] twice, once on the top and once on the bottom); thus, the farthest point Voronoi diagram for a sorted set S can be built in linear expected time.

4. Smallest enclosing circle.

Given a set S of points in the plane, the goal is to find the smallest circle that encloses all the points of S. Megiddo [7] has developed an algorithm for this problem that runs in O(n) worst-case time where n is the number of points in S. Our algorithm is less general. We can find the smallest enclosing circle for a set P of vertices of a convex polygon in linear expected time. This is done by building the farthest point Voronoi diagram for P; then, as in [8], using the diagram to determine the smallest enclosing circle in linear time. This technique is based on the fact that the center of the smallest enclosing circle must lie on an edge or a vertex of the farthest point Voronoi diagram.

For many applications, the set S already happens to be sorted; thus, the convex hull can be constructed in linear time. In this situation, the method suggested here runs quickly (with high probability) and is likely to be easier to implement than Megiddo's more general algorithm.

5. Line Segment Voronoi Diagram for a Convex Polygon.

Voronoi diagrams can be defined for a set S of line segments instead of a set of points; the region associated with line segment s consists of the set of all points in the plane that are closer to s than to any other line segment of S. Kirkpatrick [5] and Yap [9] have given methods for constructing the Voronoi diagram for S in O(n log n) time where n is the number of line segments in S. (Note: these line segments may intersect only at their endpoints.) A modification of the technique presented here can be used to construct the line segment Voronoi diagram in O(n) expected time for an n-sided convex polygon.

References

[1] A. Aggarwal and P. Shor, A linear time algorithm for computing the Voronoi diagram of a convex polygon, Technical Report, Math Sciences Research Institute, Berkeley, California (1986).

[2] J. L. Bentley, B. W. Weide, and A. C. Yao, Optimal expected-time algorithms for closest point problems, ACM Trans. Math. Software, 6 (1980), 563-580.

[3] B. Chazelle and H. Edelsbrunner, An improved algorithm for constructing kth-order Voronoi diagrams, Proceedings of the Symposium on Computational Geometry (1985), 228-234.

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

[5] D. G. Kirpatrick, Efficient computation of continuous skeletons, Proc. 20th IEEE Symposium on Foundations of Computer Science (1979), 18-27.

[6] D. T. Lee, On k-nearest neighbor Voronoi diagrams in the plane, IEEE Transactions on Computers, C-31:6 (1982), 478-487.

[7] N. Megiddo, Linear-time algorithms for linear programming in R^3 and related problems, SIAM J. Comput., 12:4 (1983), 759-776.

[8] M. I. Shamos and D. Hoey, Closest-point problems, Proc. 16th IEEE Symposium on Foundations of Computer Science (1975), 151-162.

[9] C. K. Yap, An O(n log n) algorithm for the Voronoi diagram of a set of simple curve segments, Technical Report, Courant Institute, New York University (Oct. 1984).