

Notes on Voronoi and Delaunay Diagrams

Let $S = \{p_1, \dots, p_n\}$ be a set of n points (*sites*) in the plane.

Definition of Voronoi diagram. For any subset $Q \subseteq S$ of sites we define the *Voronoi region*, $V(Q)$, for Q to be the set of all points $p \in \mathbb{R}^2$ such that Q is the set of closest sites to p .

There are 2^n different subsets of S , but only a very few of them have a nonempty Voronoi region, $V(Q)$. In fact, if S has no four cocircular points, then $V(Q)$ is empty for any Q having $|Q| \geq 4$. For each singleton set, $Q = \{p_i\}$, $V(Q) = V(\{p_i\})$ is nonempty and is often called the *Voronoi cell associated with site p_i* . We usually abuse notation slightly and let $V(p_i) = V(\{p_i\})$.

By our definition, $V(p_i)$ is always an *open* set (it does not include its boundary); this is in contrast with the notation in O'Rourke, which defines $V(p_i)$ to be a closed set that includes its boundary. If Q consists of two points, $Q = \{p_i, p_j\}$, and $V(Q)$ is nonempty, then $V(Q) = V(\{p_i, p_j\})$ is called a *Voronoi edge*; it will be a subset of the *bisector*, $b(p_i, p_j)$, between p_i and p_j . Voronoi edges are *open* line segments (they do not include their endpoints). If Q consists of 3 or more points, then $V(Q)$ will be a single point (or will be empty); we call such points *Voronoi vertices*. The *Voronoi diagram* for S , denoted $\mathcal{V}(S)$, is defined to be the partition of \mathbb{R}^2 into the regions $V(Q)$, for all $Q \subseteq S$. The Voronoi diagram is a polygonal subdivision, consisting of vertices, edges, and (convex) polygonal faces (cells).

Definition of Delaunay diagram. We define the *Delaunay diagram*, $\mathcal{D}(S)$, to be the “straight line dual” of the Voronoi diagram, $\mathcal{V}(S)$: Specifically, we place a Delaunay *node* (vertex) at each site p_i and we join two sites p_i and p_j with a straight line segment if and only if the Voronoi cells $V(p_i)$ and $V(p_j)$ share a common boundary segment (i.e., if and only if the Voronoi edge $V(\{p_i, p_j\})$ exists (is nonempty)). While this definition gives a specific recipe for drawing the diagram, which is defined as a particular drawing of a planar dual graph of the Voronoi diagram, the definition alone does not suffice to be able to claim that the diagram is “nice” in the sense that no two edges cross. Since it is a planar dual of a planar graph (the vertices and edges of the Voronoi diagram), the graph itself must be planar; however, it requires a proof (first given by Delaunay) that the particular drawing we give (with nodes at the sites p_i and edges drawn with *straight* segments) is a *plane graph*, having no crossing edges.

Properties of Voronoi/Delaunay Diagrams.

1. Voronoi cells are the interiors of convex polygons or of “unbounded convex polygons”, whose boundary edges are Voronoi edges and whose vertices are Voronoi vertices.
2. Each Voronoi cell $V(p_i)$ contains the point p_i and corresponds to the Delaunay node p_i . Each Voronoi edge $V(\{p_i, p_j\})$ corresponds to its dual Delaunay edge, (p_i, p_j) . (**NOTE:** The line segment $V(\{p_i, p_j\})$ need not cross the line segment $p_i p_j$. Draw an example!) Each Voronoi vertex $v = V(\{p_{i_1}, p_{i_2}, \dots, p_{i_m}\})$ ($m \geq 3$) corresponds to its dual Delaunay face; the point v is the center of a circle, C , that passes through the sites p_{i_1}, \dots, p_{i_m} and no other site lies inside or on the circle C . (**NOTE:** The point v need not lie inside the corresponding Delaunay face. Draw an example!)

3. The Delaunay diagram is a plane graph (has no crossing edges). (This is Delaunay’s theorem.) If there are no four cocircular points, then it is a triangulation of the convex hull of S – the *Delaunay triangulation*. If there are subsets of four or more sites that are cocircular, the faces of $\mathcal{D}(S)$ may not all be triangles; some may be convex polygons. No site can lie interior to a face of $\mathcal{D}(S)$.
4. The boundary of the face at infinity of $\mathcal{D}(S)$ is the boundary of the convex hull of the sites. Those pairs of sites that appear consecutively on the boundary of the convex hull have corresponding Voronoi edges that are unbounded (they are rays).
5. $p_i p_j$ defines a Delaunay edge if and only if there is a “site-free” circle passing through p_i and p_j , meaning that no sites other than p_i and p_j lie inside or on the circle.
6. $V(\{p_i, p_j\})$ defines a (nonempty) Voronoi edge if and only if there is a “site-free” circle passing through p_i and p_j , meaning that no sites other than p_i and p_j lie inside or on the circle.
7. For any partitioning of the sites S into “red” and “blue” sites (say, R and B), the shortest edge (line segment) linking a red site to a blue site must be a Delaunay edge. This fact has corollaries:

- (a) $\mathcal{D}(S)$ is a connected graph.
- (b) $NNG \subseteq \mathcal{D}(S)$ (Lemma 5.5.1), where NNG is the nearest neighbor graph. Thus, NNG can be computed in time $O(n \log n)$.
- (c) $MST \subseteq \mathcal{D}(S)$ (Theorem 5.5.5). A consequence of this is that the MST can be computed in time $O(n \log n)$ for n points in the plane (since Kruskal’s algorithm can be implemented in time $O(E \log E)$ for an E -edge graph, if one uses the appropriate data structures). The MST can be used to obtain a “2-approximation” to a Traveling Salesperson tour, as explained in Section 5.5.5. In fact, using a slightly smarter method (Christofides heuristic), a 1.5-approximation can be computed. Using much more sophisticated methods, a $(1 + \epsilon)$ -approximation can be computed efficiently, for any fixed $\epsilon > 0$.

8. For $n \geq 3$, $\mathcal{V}(S)$ has at most $2n - 5$ Voronoi vertices and at most $3n - 6$ Voronoi edges. (Thus, for $n \geq 3$, $\mathcal{D}(S)$ has at most $2n - 5$ faces and at most $3n - 6$ edges.)

Thus, the “size” of the Voronoi diagram (in terms of the number of bytes needed to store its edges, vertices and faces in, say, a winged-edge data structures) of n sites is $O(n)$. Likewise, the size of a Delaunay diagram is linear ($O(n)$).

9. Assume that no four points of S are cocircular. Then $\mathcal{D}(S)$ is a triangulation of the convex hull, $CH(S)$, and this triangulation is “optimal” (or “nicest”) in the following precise sense: $\mathcal{D}(S)$ maximizes the minimum angle of the triangles. More specifically, if we take the $3t$ angles defined by the t triangles of $\mathcal{D}(S)$, and organize them in a sequence (the “angle sequence”), $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{3t})$, sorted from smallest to largest, then the Delaunay triangulation lexicographically maximizes (over all possible triangulations) the vector α . (It maximizes α_1 , the smallest angle; subject to that being minimum, it maximizes α_2 , etc.) See Section 5.5.2 and Theorem 5.5.2.

10. Given the Voronoi diagram $\mathcal{V}(S)$, one can preprocess it in time $O(n)$ in order to support “nearest neighbor queries” (find the closest site to a query point $q \in \mathbb{R}^2$) that can be answered in time $O(\log n)$ using “point location” data structures. See Section 7.11.
11. The largest empty circle having its center within $CH(S)$ must be centered either at a Voronoi vertex or at a point on the boundary of $CH(S)$ where a Voronoi edge crosses the boundary of $CH(S)$.
12. If we “lift” the points S onto the paraboloid of revolution, $z = x^2 + y^2$ in 3D, we get a set S' of points in 3D. (These points are in convex position – each point of S' is a vertex of the convex hull $CH(S')$.) The Delaunay diagram $\mathcal{D}(S)$ is exactly what one would “see” if we look “up” at the convex hull $CH(S')$; i.e., the edges of $CH(S')$ that bound facets that are the “lower” hull are exactly corresponding to the Delaunay edges. See Theorem 5.7.2. The upper hull correspond to the “furthest-point Delaunay”, which is the dual to the “furthest-point Voronoi” diagram (a decomposition of the plane into regions according to which site is *furthest* away, rather than closest).
13. $\mathcal{D}(S)$ and $\mathcal{V}(S)$ can be computed in time $O(n \log n)$, using $O(n)$ space. Algorithms that achieve this worst-case time bound can be based on either divide-and-conquer or plane sweep (using a clever method of Fortune). If the diagrams are built incrementally, the worst-case time bound is $O(n^2)$; however, if we add sites in random order, the *expected* running time is only $O(n \log n)$.
14. There is a lower bound of $\Omega(n \log n)$ on the time needed to construct the Voronoi or Delaunay diagram of a set of points, if the input is simply the point set. This follows from the fact that there is an $\Omega(n \log n)$ lower bound for computing convex hulls, since the Delaunay diagram (or Voronoi) can be used to compute the convex hull in linear time. (i.e., given either one of $\mathcal{D}(S)$ or $\mathcal{V}(S)$, one can easily traverse the diagrams to extract the convex hull in linear ($O(n)$) time)
15. If either one of $\mathcal{D}(S)$ or $\mathcal{V}(S)$ is given, then the other can be computed in time $O(n)$ from it. (See problem 2 of Section 5.7.5.)

Algorithms to Build $\mathcal{D}(S)$ or $\mathcal{V}(S)$ Here is a summary of standard methods to construct the Voronoi or Delaunay diagram:

1. Divide and conquer: Split the point set in half (e.g., according to the median x -coordinate), recursively build the diagram for each half, and then merge. The tricky part is doing the merge efficiently; it can be done in time $O(n)$, as I briefly explained in class. Overall, then, we get a running time of $T(n) \leq 2T(n/2) + O(n)$, which solves to $T(n) = O(n \log n)$. The algorithm uses $O(n)$ space (storing the diagrams, which are planar graphs).
2. Plane sweep: Steve Fortune devised a clever plane sweep algorithm in 1985 to compute the diagrams in $O(n \log n)$ time. An overview is in Section 5.4.4.
3. Incremental construction: By adding points one by one and updating the diagrams, it is possible to achieve worst-case time $O(n^2)$, since the worst-case time for insertion is $O(n)$. In fact, the time it takes to insert a new point p_i into the existing Voronoi diagram is proportional

to the number of sides in the cell of p_i in the revised diagram; the number of sides is linear in the number of sites, but often much smaller in practice.

In fact, if points are added in random order, it is possible to build the diagrams in *expected* time $O(n \log n)$, since, on average, the insertion cost is low ($O(\log n)$, which comes from a Harmonic series computation).

4. Flipping algorithms: One can start with any triangulation of S and “flip” diagonals that are not “locally Delaunay” until every edge is locally Delaunay. It can be shown that this takes time $O(n^2)$ in the worst case. See Problem 5, Section 5.4.5, and see the notes below.
5. Because of the connection to convex hulls in 3D, any algorithm that works for convex hulls in 3D can be used to compute a Delaunay diagram in 2D (and thus also the Voronoi diagram). For instance, QuickHull can be used to compute Voronoi and Delaunay diagrams (even in higher dimensions; see <http://www.geom.umn.edu/locate/qhull> for the recently released QHull 3.0).

Edge-Flipping Algorithm (Lawson) Consider an arbitrary triangulation, T , of S (more precisely, of the convex hull of S). There are many ways that we can get *some* triangulation, but often one is already known to us. (Note, though, that if no triangulation is known, it does take at least time $\Omega(n \log n)$ to compute a triangulation of a set of points given in arbitrary order.)

Each edge of the convex hull, $CH(S)$, is an edge in T and is also an edge in $\mathcal{D}(S)$. Thus, our interest concerns only those *interior* (non-hull) edges of T . We say that an interior edge $e = ab$, bounding the triangles abc and abd , is *locally Delaunay* if point c does not lie inside the circle $C(a, b, d)$ through points a , b , and d (which is equivalent to saying that d does not lie inside the circle through a , b , and c). Basically, $e = ab$ is locally Delaunay if and only if ab is a Delaunay edge in the Delaunay triangulation of just the four points a , b , c , d . If $e = ab$ is *not* locally Delaunay, then the four points a , b , c , d must be in convex position (they define a convex quadrilateral). Further, if we replace $e = ab$ with $e' = cd$ (we “flip” the diagonal of the quadrilateral $acbd$), then the new edge e' is now locally Delaunay (because one can prove that a lies outside the circle through b , c and d). This operation of swapping e' for e is called an *edge flip*.

The edge flipping algorithm starts with T and repeatedly modifies the triangulation by a series of edge flips: Just pick any edge that is not locally Delaunay and flip it. After doing so, check the local neighborhood of triangles to see if the “local Delaunayhood” of any other edge has changed, and update its status. Continue until all edges are locally Delaunay. There is a theorem that says that a triangulation is Delaunay if and only if each edge is locally Delaunay (i.e., the local property holding for *all* edges implies the global property of Delaunayhood).

We claim that at most $O(n^2)$ flips are needed in order to get from any one triangulation T to the Delaunay. This can be shown using the interpretation of Delaunay triangulation as convex hulls in 3D, as I explained in class.