Higher Dimensional Delaunay Diagrams for Convex Distance Functions (Extended Abstract)

Barry F. Schaudt * R.L. (Scot) Drysdale †

June 2, 1992

Abstract

We describe an algorithm to compute the Delaunay diagram of a non-degenerate set S of n sites in R^d using any convex distance function. The size of a d-dimensional Delaunay diagram is $O(n^{\lfloor (d+1)/2 \rfloor})$. The algorithm that we present runs in time $O(n^{\lceil (d+1)/2 \rceil} + n \lg n)$ using space $O(n^{\lfloor (d+1)/2 \rfloor})$

We also show that the Delaunay diagram for non-Euclidean distance functions need not be a triangulation of the sites in S. For $d > 2$ the interiors of the "triangles" can intersect.

1 Introduction

Voronoi diagrams and Delaunay diagrams and their extensions have been extensively studied by computational geometers in the last few years. A resent survey by Aurenhammer [Aur91] summarizes many of the results. In this paper we will present an algorithm for computing the Delaunay diagram of a set of non-degenerate sites using any convex distance function. (We will define what we mean by a set of non-degenerate sites shortly.) The only primitive that this algorithm uses is an in-sphere test. We will also show that the Delaunay diagram computed using a convex distance function may not be a triangulation. For $d > 2$ the interiors of the "triangles" can intersect.

In two dimensions, the size of the Euclidean Delaunay diagram is $O(n)$. Guibas and Stolfi [GS85] present an optimal $O(n \lg n)$ algorithm for computing the diagram. This algorithm is a divide and conquer algorithm that is quite similar to the three-dimensional divide and conquer convex hull algorithm of Preparata and Hong [PH77].

One variant of the standard Delaunay and Voronoi diagrams measures distances by a non-Euclidean distance function. Examples of such diagrams include the L_p metrics [Lee80]. If we restrict ourselves to the L_1 or L_∞ metrics, then there are several other algorithms known to compute the diagrams [LW80, SDT91].

Chew and Drysdale [CD85] defined a Voronoi diagram where distances are measured by using any convex distance function. A convex distance function is defined by choosing any convex shape as the "unit circle" and any interior point as its "center". To measure the distance from a to b the unit circle is centered at a, and is expanded or contracted until b lies on its boundary. The scaling factor needed to accomplish this is the distance from a to b. The equivalent definition in d dimensions uses a d-dimensional convex shape as the unit d-sphere.

Drysdale [Dry90] gives a practical algorithm for computing the two-dimensional Delaunay diagram, from which the Voronoi diagram can be computed. Drysdale's algorithm is a generalization of Guibas and Stolfi's two-dimensional Delaunay triangulation algorithm.

In higher dimensions the size of the Euclidean Delaunay diagram is $O(n^{\lfloor (d+1)/2 \rfloor})$. For dimensions higher than three, the d-dimensional Delaunay diagram is computed by first transforming the d-dimensional Delaunay diagram problem to a $(d+1)$-dimensional convex hull problem. The sites in d-dimensions are projected onto a paraboloid in $(d+1)$-dimensions and the convex hull of the projected sites is computed. If the sites are non-degenerate, then the downward-facing faces on the convex hull are the faces in the Delaunay diagram. Edelsbrunner [Ede87] discusses this projection.

There are several methods to compute the convex hull in higher dimensions. Seidel's algorithm as presented in Edelsbrunner's book [Ede87] runs in time $O(n^{\lfloor (d+1)/2 \rfloor} + n \lg n)$. Seidel's algorithm is optimal for even dimensions higher than 2. If we compute the d-dimensional Delaunay diagram by computing a $(d+1)$-dimensional convex hull

*Lewis and Clark College, email: schaudt@lclark.edu
†Dartmouth College, email: scot.drysdale@dartmouth.edu

of Seidel, then we can compute the d-dimensional convex hull in time $O(n^{\lceil (d+1)/2 \rceil} + n \lg n)$, which is optimal in odd dimensions higher than 5. Just as Drysdale's two-dimensional Delaunay diagram algorithm is similar to Preparata and Hong's three-dimensional convex hull algorithm, the algorithm that we present in this paper to compute a d-dimensional Delaunay diagram is similar to Seidel's algorithm to compute a $(d+1)$ dimensional convex hull.

It is worth noting that Chazelle [Cha91] has an optimal convex hull algorithm in any fixed dimension.

2 Definitions and Basic Properties

If $S = \{x_1, x_2, \ldots, x_{k+1}\}$, then the convex hull of S is a polytope. If the dimension of the convex hull of S is k, then the convex hull of S is a k-simplex. In 2-dimensions, a simplex is a triangle and in 3-dimensions, a simplex is a tetrahedron. In d-dimensions, the $(d-1)$-faces of a simplex are called facets. We will refer to a d-simplex as a cell.

If f is a k-simplex, then simplex$(f, v_1, v_2, \ldots, v_i)$ will denote the simplex of the $k+1$ vertices of f and the sites $v_1, v_2, \ldots, v_i$.

We will now define non-degeneracy conditions which will insure that each k-face in the Delaunay diagram is a simplex composed of $k+1$ vertices. The standard definition of non-degeneracy in the Euclidean Delaunay diagram states that no $d+2$ sites are cospherical. This implies that the set of points equidistant from $k+1$ sites is a $d-k$ flat. A k-flat can be defined as the intersection of $d-k$ hyperplanes whose normal vectors are linearly independent [Ede87]. We need a similar definition for convex distance functions.

Definition 2.1 *A set of sites in R^d is nondegenerate if and only if*

(1) For any two sites p_1 and p_2, $\{x | d(p_1, x) = d(p_2, x)\}$ divides R^d into two sets, where each set is connected, unbounded and has dimension d. Furthermore, the set of points equidistant from p_1 and p_2 has dimension $d-1$.

(2) For any $k+1$ sites $p_1, \ldots, p_{k+1}$, $\{x | d(p_1, x) = d(p_2, x) = \ldots = d(p_{k+1}, x)\}$ divides $\{x | d(p_1, x) = \ldots = d(p_k, x)\}$ into two sets, where each set is connected, unbounded and has dimension $d-k$. Furthermore, the set of points equidistant from $p_1, p_2, \ldots, p_{k+1}$ has dimension $d-k+1$.

The definition of non-degeneracy implies that no $d+2$ sites are cospherical (the sphere defined by our choice of convex distance function). In cases where the unit sphere has a flat side, this also implies that no two points lie on a line parallel to that flat side. (In the L_∞ metric, this means that no two points can have the same value for the j^{th} coordinate.) We conjecture that the usual definition of non-degeneracy – no $d+2$ sites are cospherical and no two sites lie on a line parallel to a flat side of the unit sphere – implies our definition of non-degeneracy. This conjecture is true for the L_∞ metric and is also true for any convex polytope distance function. We also note that if a set of sites fails to satisfy these conditions that a small perturbation of the sites will be non-degenerate.

Among other things needed for the proof of correctness of our algorithm, this definition of non-degeneracy implies that the set of all points equidistant from d sites is a one-dimensional curve. The d sites define a family of spheres, where each sphere touches all of the sites. Each sphere in this family has its center on the curve. There are two infinite spheres associated with this family. In the Euclidean metric, the infinite spheres are halfspaces and the interiors of the infinite spheres do not intersect. For convex distance functions, the infinite spheres may not be halfspaces and their interiors may intersect. See Figure 1. Our in-sphere primitive also determines which infinite sphere a site is in.

With the Euclidean metric, the Delaunay diagram of a set of sites where no $d+2$ of these sites are cospherical is a triangulation of the interior and boundary of the convex hull of these sites. One of the properties that any triangulation must satisfy is that the interior of two simplicies must not intersect. For a formal definition of a triangulation, see Rothschild and Strauss [RS85].

Unfortunately, once we leave the Euclidean metric, the Delaunay diagram may not be a triangulation. It is possible to construct examples where the interiors of the simplicies intersect. An example of intersecting Delaunay simplicies is shown in Figure 2 for a 3-dimensional Delaunay diagram using the L_∞ metric. The Euclidean Delaunay diagram is a triangulation because the intersection of two hyperspheres lies in a hyperplane.

Even though the interiors of the simplicies may intersect each facet of the Delaunay diagram is incident on exactly two cells and we can still compute the Delaunay diagram with a simple algorithm. The fact that we do not have a triangulation complicates only the proof of correctness.

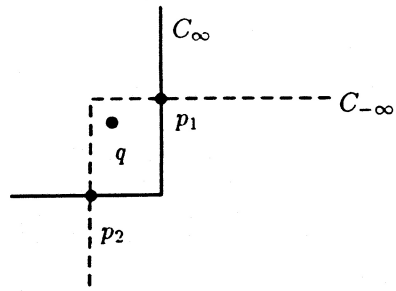

Figure 1: In the L_∞ metric, the infinite circles with sites p_1 and p_2 on the boundary are shown. The intersection of the interiors of the two infinite circles is non-empty. If q is a site in the intersection of the interiors of the two infinite circles, then there is no circumcircle of q, p_1, and p_2. Also, q is in all circles with p_1 and p_2 on the boundary. That is, all circumcircles of p_1 and p_2 contain q.

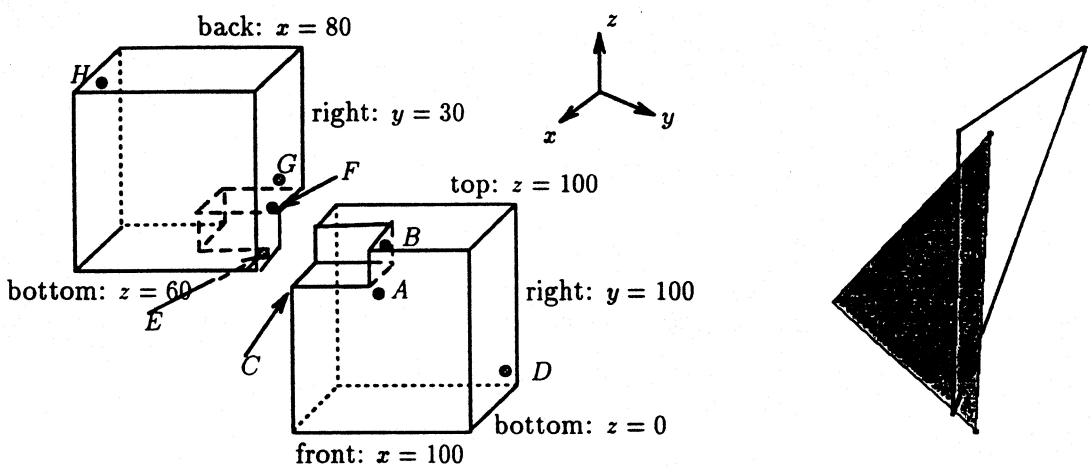

Figure 2: The interiors of two Delaunay simplicies may intersect. An example of this is shown in 3-dimensions using the L_∞ metric. Here, the interiors of tetrahedrons $ABCD$ and $EFGH$ intersect. The sites that determine the right cube are $A(100, 32, 59), B(99, 31, 100), C(98, 0, 58)$ and $D(0, 90, 10)$. The sites that determine the left cube are $E(101, 29, 60), F(102, 30, 101), G(80, 28, 102)$ and $H(170, -60, 160)$. In this figure, the cubes are pulled apart for illustrative purposes. On the right, we see a view of just the shaded triangle ABC and the unshaded triangle EFG. Sites A and B are the lower and upper points on the shaded triangle respectively.

3 The Algorithm

Our d-dimensional Delaunay diagram algorithm is a generalization of Guibas and Stolfi's incremental Delaunay triangulation algorithms. Each point to be inserted will be located in the cell or cells that contain it. Then we will determine which old cells are eliminated due to this point, and which faces are connected to the new point to form new cells. The algorithm is quite similar to Seidel's convex hull algorithm as presented by Edelsbrunner [Ede87].

We will assume that our sites lie inside some convex polytope whose vertices V are part of the set S and that we already have computed the Delaunay diagram of the sites on this convex polytope. We will insert new points in lexicographical order, so that any cell containing a new point must have a vertex in V.

Before we present the incremental algorithm, we need a theorem that tells what faces we must change when a new site is inserted. The notation simplex(e,p), where e is a k simplex and p is a point, denotes the simplex constructed from p and the vertices of e. We call this process "connecting p to the k-simplex e."

Theorem 3.1 *Suppose D is a Delaunay diagram of $\bar{S}$, where $\bar{S}$ is a subset of $S - V$, and p is a site in $S - \bar{S} - V$. If D' is the Delaunay diagram of the sites $\bar{S} \cup \{p\} \cup V$, then each face of D' is one of the following types.*

(i) a face f of D is also a face of D' if and only if there exists a cell F of D such that $f \subset F$ and p is not in the circumsphere of F.

(ii) if f is a face of D, then simplex(f,p) is a face of D' if and only if among the cells of D containing f there is at least one cell with p is in its circumsphere and at least one cell with p not in its circumsphere.

Proof: Omitted ∎

The data structure that we will use is the incidence graph. The incidence graph consists of $d+3$ doubly linked lists $I_{-1}, I_0, I_1, \ldots, I_{d+1}$. The list I_k contains the k-faces . The list I_{-1} always consists of a single node, $\emptyset$, and this node is considered a subface of each entry in I_0, that is, each vertex. Likewise, I_{d+1} consists of a single node and is a superface of each entry in I_d. An entry e of I_j contains in addition to the pointers needed to maintain the doubly linked list a pointer to each superface that it is incident to and a pointer to each subface that it is incident to. We will also keep a list of cells that contain a vertex from V.

In addition to the pointers we will also need a mark field for each entry in I. A mark is one of the following: "unmarked","to be deleted", "to be connected to p", or "not to be deleted". The only faces that are marked "not to be deleted" are cells.

We will also need another data structure, L, that will contain nodes that we have marked. This data structure will consist of $d+2$ doubly linked lists $L_{-1}, L_0, L_1, \ldots, L_d$. Each list L_k is a list of the k-faces in I_k that we have looked at.

To add a new site p to a Delaunay diagram, the algorithm proceeds in two phases: a mark phase and an update phase. The update phase will just perform all the updates indicated by the mark phase.

The mark phase has two parts. We can informally think of the first part in terms of spheres being pushed through facets of the Delaunay cells. We start with a cell that contains p and make one copy of the circumsphere of this cell for each facet of the cell. We push each copy of the circumsphere through a different facet. Each of these spheres will change shape while remaining in contact with all d points of its facet. We continue pushing the sphere until it contacts another point and becomes a circumsphere of some adjacent cell. If this circumsphere contains p and if it has not already been considered, then we mark the cell of the circumsphere for deletion and repeat this process for each facet of the cell. The cells are marked by doing a depth first search.

The next part of the mark phase will then mark all the k-faces, $k < d-1$, of the cells that the first part considered. They will be marked "to be deleted", or "to be connected to p" as specified by Theorem 3.1. The mark phase of the algorithm is outlined in Figure 3.

In order to find the cells that contain the new site, our algorithm considers the sites in lexicographical order. This means that the new site is outside the convex hull of the previous sites and the new site is in a circumsphere of a cell that has at least one vertex in V.

The second phase of the algorithm will perform all the necessary updates to the data structure I. The update phase is outlined in Figure 4. The update phase looks at each face f in L, which contains the marked faces, starting with the faces in L_{-1}. Only when all the faces in L_k are updated, will the faces in L_{k+1} be updated.

Note that this algorithm is a higher dimensional flip algorithm. All the facets that will be flipped are marked in the first phase and flipped in the second phase.

Part 1.
Find all the cells that contain contain p by searching through all cells that have a site in V as a vertex.
for each cell that contains p do DFS(C).

Part 2.
At this point all cells that must be deleted and cells that are adjacent to a cell that must be deleted are marked. We must now mark all faces incident to cells that we have looked at.
for i := d-2 downto -1 do
    for each face f in L_{i+1} do
        for each subface e of f do
            if e is ummarked then Add e to L_i; Mark(e) = mark(f)
            if f is marked "to be connected to p" then Mark(e) = mark(f)
DFS(C)
Add cell C to list L_d.
if p is in the circumsphere of C then Mark C to be deleted
else Mark C not to be deleted.
if p is in the circumsphere of C then
    for each $(d-1)$-face, e, of cell C do
        Let C_a be the other cell incident upon e.
        if we have not already looked at C_a then DFS(C_a)
        if e is unmarked then
            Add e to list L_{d-1}
            if e is adjacent to two cells whose spheres contain p then Mark e for deletion.
            if e is adjacent to one cell whose sphere contains p and one cell whose circumsphere does not contain p, then Mark e to be connected to p

Figure 3: The mark phase of the algorithm.

4 Correctness, Complexity and Run Time

The proof of correctness of the algorithm is similar to the proof of correctness for Guibas and Stolfi two-dimensional Delaunay diagram algorithm [GS85]. We define the equivalent of an inCircle test and then show that we only need to apply the test locally.

Definition 4.1 *Let f be a facet incident to cells C_1 and C_2, with $C_1 = simplex(f, X)$ and $C_2 = simplex(f, Y)$ we will say f passes the sphere test if and only if.*

1. *The circumsphere of $simplex(f, X)$ exists and does not contain Y and*
2. *The circumsphere of $simplex(f, Y)$ exists and does not contain X.*

Note that if the circumsphere of simplex(f, X) does not exist, then there are no empty circumspheres of f. Again, see Figure 1.

Lemma 4.2 *A diagram is Delaunay if and only if all its facets pass the sphere test.*

Proof: Omitted. ∎

In order to prove the runtime of the algorithm we need the following.

Theorem 4.3 *The worst case size of the Delaunay diagram is $\theta(n^{\lfloor (d+1)/2 \rfloor})$.*

Proof: Omitted. ∎

Theorem 4.4 *The algorithm runs in time $O(n^{\lfloor (d+2)/2 \rfloor} + n \log n)$ using $O(n^{\lfloor (d+1)/2 \rfloor})$ space.*

Proof: This proof is similar to the proof of the runtime of Seidel's convex hull algorithm as presented by Edelsbrunner [Ede87].

```
for i := −1 to d do
    for each face f in list L_i do
        if f is marked to be deleted then
            Delete face f and all pointers from f to superfaces and to subfaces in I
            if f is a cell and f has a site in V as a vertex then
                remove f from the list of cells that have a site in V as a vertex.
        else if f is marked to be connected to p then
            Create a new face, simplex(f,p) in I_{i+1}
            if simplex(f,p) is a cell and f has a site in V as a vertex then
                Add simplex(f,p) to the list of cells that have a site in V as a vertex.
            Make f a subface of simplex(f,p) in I
            Make simplex(f,p) a superface of f in I
            for each subface e of f do
                Make simplex(e,p) a subface of simplex(f,p) in I
                Make simplex(f,p) a superface of simplex(e,p) in I
Free all nodes in L.
```

Figure 4: The update phase of the algorithm.

5 Open Problems

The first open problem that we would like to see proved is our non-degeneracy conjecture: the usual definition of non-degeneracy is equivalent to our definition.

Second, our Delaunay diagram algorithm is closely resembles Seidel's convex hull algorithm. Perhaps, Chazelle's new optimal convex hull algorithm [Cha91] can be used to design an optimal Delaunay diagram for convex distance functions.

References

[Aur91] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23:345–405, 1991.

[CD85] L.P. Chew and R.L. Drysdale. Voronoi diagrams based on convex distance functions. *Proceedings of the First Annual Symposium on Computational Geometry*, pages 235–244, 1985.

[Cha91] Bernard Chazelle. An optimal convex hull algorithm and new results on cuttings. *32nd Symposium on Foundations of Computer Science*, pages 29–38, 1991.

[Dry90] R.L. Drysdale. A practical algorithm for computing the Delaunay triangulation for convex distance functions. *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 159–168, 1990.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[GS85] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4:74–123, 1985.

[Lee80] D.T. Lee. Two-dimensional Voronoi diagram in the L_p metric. *Journal of the ACM*, 27:604–618, 1980.

[LW80] D.T. Lee and C.K. Wong. Voronoi diagrams in $L_1(L_\infty)$ metrics with two-dimensional storage applications. *SIAM Journal of Computing*, 9:200–211, 1980.

[PH77] F.P. Preparata and J.S. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20:87–93, 1977.

[RS85] B.L. RothSchild and E.G. Straus. On triangulations of the convex hull of n points. *Combinatorica*, 5:167–179, 1985.

[SDT91] Gary M. Shute, Linda L. Deneen, and Clark D. Thomborson. An $O(n \lg n)$ plane-sweep algorithm for L_1 and L_∞ Delaunay triangulations. *Algorithmica*, 6:207–221, 1991.