# A New Voronoi-Based Surface Reconstruction Algorithm

Nina Amenta[*] — UT - Austin
Marshall Bern — Xerox PARC
Manolis Kamvysselis[†] — M.I.T.

## Abstract

We describe our experience with a new algorithm for the reconstruction of surfaces from unorganized sample points in $\mathbb{R}^3$. The algorithm is the first for this problem with provable guarantees. Given a "good sample" from a smooth surface, the output is guaranteed to be topologically correct and convergent to the original surface as the sampling density increases. The definition of a good sample is itself interesting: the required sampling density varies locally, rigorously capturing the intuitive notion that featureless areas can be reconstructed from fewer samples. The output mesh interpolates, rather than approximates, the input points.

Our algorithm is based on the three-dimensional Voronoi diagram. Given a good program for this fundamental subroutine, the algorithm is quite easy to implement.

**Keywords:** Medial axis, Sampling, Delaunay triangulation, Computational Geometry

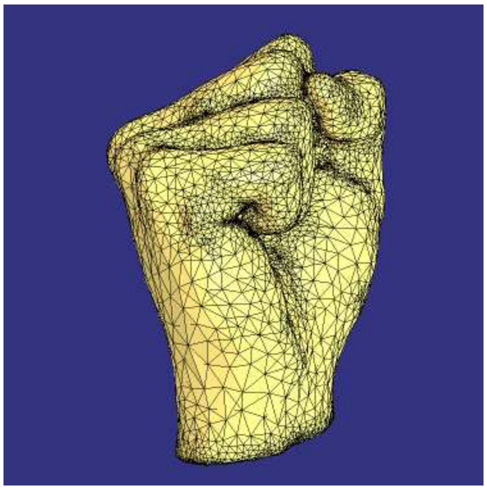

Figure 1. The fist mesh was reconstructed from the vertices alone. Notice that the sampling density varies. Our algorithm requires dense sampling only near small features; given such an input, the output mesh is provably correct.

## 1 Introduction

The process of turning a set of sample points in $\mathbb{R}^3$ into a computer graphics model generally involves several steps: the reconstruction of an initial piecewise-linear model, cleanup, simplification, and perhaps fitting with curved surface patches.

We focus on the first step, and in particular on an abstract problem defined by Hoppe, DeRose, Duchamp, McDonald, and Stuetzle [14]. In this formulation, the input is a set of points in $\mathbb{R}^3$, without any additional structure or organization, and the desired output is a polygonal mesh, possibly with boundary. In practice, sample sets for surface reconstruction come from a variety of sources: medical imagery, laser range scanners, contact probe digitizers, radar and seismic surveys, and mathematical models such as implicit surfaces. While the most effective reconstruction scheme for any one of these applications should take advantage of the special properties of the data, an understanding of the abstract problem should contribute to all of them.

The problem formulation above is incomplete, since presumably we should require some relationship between the input and the output. In this and a companion paper [2], we describe a simple, combinatorial algorithm for which we can prove such a relationship. A nontrivial part of this work is the fitting of precise definitions to the intuitive notions of a "good sample" and a "correct reconstruction". Although the actual definition of a good sample is rather technical, involving the medial axis of the original surface, Figure 1 gives the general idea: dense in detailed areas and (possibly) sparse in featureless ones.

The algorithm is based on the three-dimensional Voronoi diagram and Delaunay triangulation; it produces a set of triangles that we call the *crust* of the sample points. All vertices of crust triangles are sample points; in fact, all crust triangles appear in the Delaunay triangulation of the sample points.

The companion paper [2] presents our theoretical results. In that paper, we prove that given a good sample from a smooth surface, the output of our reconstruction algorithm is topologically equivalent to the surface, and that as the sampling density increases, the output converges to the surface, both pointwise and in surface normal.

Theoretical guarantees, however, do not imply that an algorithm is useful in practice. Surfaces are not everywhere smooth, samples do not everywhere meet the sampling density conditions, and sample points contain noise. Even on good inputs, an algorithm may fail to be robust, and the constants on the running time might be prohibitively large. In this paper, we report on our implementation of the algorithm, its efficiency and the quality of the output.

Overall, we were pleased. The program gave intuitively reasonable outputs on inputs for which the theoretical results do not apply. The implementation, using a freely available exact-arithmetic Voronoi diagram code, was quite easy, and reasonably efficient: it can handle 10,000 points in a matter of minutes. The main difficulty, both in theory and in practice, is the reconstruction of sharp edges.

[*] Much of this work was done while the author was employed by Xerox PARC, partially supported by NSF grant CCR-9404113.

[†] Much of this work was done while the author was an intern at Xerox PARC.

## 2 Related work

The idea of using Voronoi diagrams and Delaunay triangulations in surface reconstruction is not new. The well–known $\alpha$*-shape* of Edelsbrunner et al. [9, 10] is a parameterized construction that associates a polyhedral shape with an unorganized set of points. A simplex (edge, triangle, or tetrahedron) is included in the $\alpha$-shape if it has some circumsphere with interior empty of sample points, of radius at most $\alpha$ (a circumsphere of a simplex has the vertices of the simplex on its boundary). The *spectrum* of $\alpha$-shapes, that is, the $\alpha$-shapes for all possible values of $\alpha$, gives an idea of the overall shape and natural dimensionality of the point set. Edelsbrunner and Mücke experimented with using $\alpha$-shapes for surface reconstruction [10], and Bajaj, Bernardini, and Xu [4] have recently used $\alpha$-shapes as a first step in the entire reconstruction pipeline.

An early Delaunay-based algorithm, similar in spirit to our own, is the "Delaunay sculpting" heuristic of Boissonnat [6], which progressively eliminates tetrahedra from the Delaunay triangulation based on their circumspheres. In two dimensions, there are a number of recent theoretical results on various Delaunay-based approaches to reconstructing smooth curves. Attali [3], Bernardini and Bajaj [5], Figueiredo and Miranda Gomes [11] and ourselves [1] have all given guarantees for different algorithms.

A fundamentally different approach to reconstruction is to use the input points to define a signed distance function on $\mathbb{R}^3$, and then polygonalize its zero-set to create the output mesh. Such *zero-set* algorithms produce approximating, rather than interpolating, meshes. This approach was taken by Hoppe et al. [14, 13] and more recently by Curless and Levoy [8]. Hoppe et al. determine an approximate tangent plane at each sample point using least squares on $k$ nearest neighbors, and then take the signed distance to the nearest point's tangent plane as the distance function on $\mathbb{R}^3$. The distance function is then interpolated and polygonalized by the marching cubes algorithm. The algorithm of Curless and Levoy is tuned for laser range data, from which they derive error and tangent plane information. They combine the samples into a continuous volumetric function, computed and stored on a voxel grid. A subsequent hole-filling step also uses problem-specific information. Their implementation is especially fast and robust, capable of handling very large data sets.

Functionally our crust algorithm differs from both the $\alpha$-shape and the zero-set algorithms. It overcomes the main drawback of $\alpha$-shapes as applied to surface reconstruction, which is that the parameter $\alpha$ must be chosen experimentally, and in many cases there is no ideal value of $\alpha$ due to variations in the sampling density. The crust algorithm requires no such parameter; it in effect automatically computes the parameter locally. Allowing the sampling density to vary locally enables detailed reconstructions from much smaller input sets.

Like the $\alpha$-shape, the crust can be considered an intrinsic construction on the point set. But unlike the $\alpha$-shape, the crust is naturally two-dimensional. This property makes the crust more suitable for surface reconstruction, although less suitable for determining the natural dimensionality of a point set.

The crust algorithm is simpler and more direct than the zero-set approach. Zero-set algorithms, which produce approximating rather than interpolating surfaces, inherently do some low-pass filtering of the data. This is desirable in the presence of noise, but causes some loss of information. We believe that some of our ideas, particularly the sampling criterion and the normal estimation method, can be applied to zero-set algorithms as well, and might be useful in proving some zero-set algorithm correct.

With its explicit sampling criterion, our algorithm should be most useful in applications in which the sampling density is easy to control. Two examples are digitizing an object with a hand-held contact probe, where the operator can "eyeball" the required density, and polygonalizing an implicit surface using sample points [12], where the distribution can be controlled analytically.

## 3 Sampling Criterion

Our theoretical results assume a *smooth surface*, by which we mean a twice-differentiable manifold embedded in $\mathbb{R}^d$. Notice that this allows all orientable manifolds, including those with multiple connected components.

### 3.1 Geometry

We start by reviewing some standard geometric constructions. Given a discrete set $S$ of sample points in $\mathbb{R}^d$, the *Voronoi cell* of a sample point is that part of $\mathbb{R}^d$ closer to it than to any other sample. The *Voronoi diagram* is the decomposition of $\mathbb{R}^d$ induced by the Voronoi cells. Each Voronoi cell is a convex polytope, and its vertices are the *Voronoi vertices*; when $S$ is nondegenerate, each Voronoi vertex is equidistant from exactly $d+1$ points of $S$. These $d+1$ points are the vertices of the *Delaunay simplex*, dual to the Voronoi vertex. A Delaunay simplex, and hence each of its faces, has a circumsphere empty of other points of $S$. The set of Delaunay simplices form the *Delaunay triangulation* of $S$. Computing the Delaunay triangulation essentially computes the Voronoi diagram as well. See Figure 5 for two-dimensional examples.

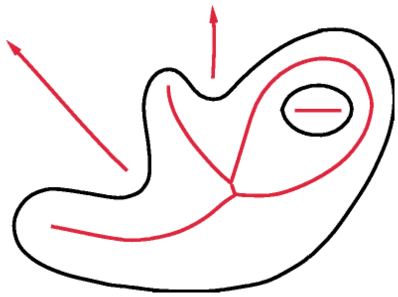

Figure 2. The red curves are the medial axis of the black curves. Notice that components of the medial axis lie on either side of the black curves.

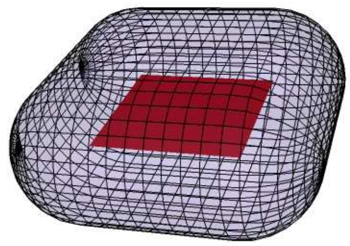

Figure 3. In three dimensions, the medial axis of a surface is generally a two-dimensional surface. Here, the square is the medial axis of the rounded transparent surface. A nonconvex surface would have components of the medial axis on the outside as well, as in the 2D example of Figure 2.

The *medial axis* of a $(d-1)$-dimensional surface in $\mathbb{R}^d$ is (the closure of) the set of points with more than one closest point on the surface. An example in $\mathbb{R}^2$ is shown in Figure 2, and in $\mathbb{R}^3$ in Figure 3. This definition of the medial axis includes components on the exterior of a closed surface. The medial axis is the extension to continuous surfaces of the Voronoi diagram, in the sense that the

Voronoi diagram of $S$ can be defined as the set of points with more than one closest point in $S$.

In two dimensions, the Voronoi vertices of a dense set of sample points on a curve approximate the medial axis of the curve. Somewhat surprisingly—a number of authors have been misled—this nice property does not extend to three dimensions.

### 3.2 Definition

We can now describe our sampling criterion. A good sample is one in which the sampling density is (at least) inversely proportional to the distance to the medial axis. Specifically, a sample $S$ is an $r$-*sample* from a surface $F$ when the Euclidean distance from any point $p \in F$ to the nearest sample point is at most $r$ times the distance from $p$ to the nearest point of the medial axis of $F$.

The constant of proportionality $r$ is generally less than one. In the companion paper [2], we prove our theorems for small values of $r$ such as $r \leq .06$, but the bounds are not tight. Hence the theoretical results apply only when the sampling is very dense.

We observe that in practice $r = .5$ generally suffices. Figure 4 shows a reconstruction from a dense sample, and from a sample thinned to roughly $r = .5$. We did not compute the medial axis, which can be quite a chore. Instead, we used the distance to the nearest "pole" (see Section 4.2) as a reasonable, and easily computed, estimate of the distance to the medial axis.

Figure 4. The sampling spacing required to correctly reconstruct a surface is proportional to the distance to the medial axis. On the left is a surface reconstructed from a dense sample. The color represents estimated distance to medial axis—red means close. On the right, we use the estimated distance to thin the data to a .5-sample (meaning that the distance to the nearest sample for any point on the surface is at most half the distance to the medial axis), and then reconstruct. There were about 12K samples on the left and about 3K on the right.

Notice that our sampling criterion places no constraints on the distribution of points, so long as they are sufficiently dense. It inherently takes into account both the curvature of the surface—the medial axis is close to the surface where the curvature is high—and also the proximity of other parts of the surface. For instance, although the middle of a thin plate has low curvature, it must be sampled densely to resolve the two sides as separate surfaces. In this situation an $r$-sample differs from the distribution of vertices typically produced by mesh simplification algorithms, which only need to consider curvature.

At sharp edges and corners, the medial axis actually touches the surface. Accordingly, our criterion requires infinitely dense sampling to guarantee reconstruction. Sharp edges are indeed a problem in practice as well, although the reconstruction errors are not noticeable when the sampling is very dense. We discuss a heuristic approach to resolving sharp edges in Section 6, and propose a stronger theoretical approach in Section 7.

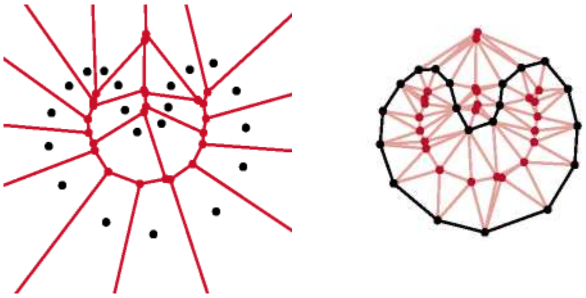

Figure 5. The two-dimensional algorithm. On the left, the Voronoi diagram of a point set $S$ sampled from a curve. Just as $S$ approximates the curve, the Voronoi vertices $V$ approximate the medial axis of the curve. On the right, the the Delaunay triangulation of $S \cup V$, with the crust edges in black. Theorem 1 states that when $S$ is an $r$-sample, for sufficiently small $r$, the crust edges connect only adjacent vertices.

## 4 The crust algorithm

### 4.1 Two Dimensions

We begin with a two-dimensional version of the algorithm [1]. In this case, the crust will be a graph on the set of sample points $S$. We define the crust as follows: an edge $e$ belongs to the crust if $e$ has a circumcircle empty not only of all other sample points but also of all Voronoi vertices of $S$. The crust obeys the following theorem [1].

**Theorem 1.** *The crust of an $r$-sample from a smooth curve $F$, for $r \leq .25$, connects only adjacent sample points on $F$.*

The medial axis provides the intuition behind this theorem. An important lemma is that for any sample $S$, an edge between two nonadjacent sample points cannot be circumscribed by a circle that misses both the medial axis and all other samples. When $S$ is an $r$-sample for sufficiently small $r$, the Voronoi vertices approximate the medial axis, and any circumcircle of an edge between nonadjacent samples contains either another sample or a Voronoi vertex. An edge between two adjacent samples, on the other hand, is circumscribed by a small circle, far away from the medial axis and hence from all Voronoi vertices.

The definition of the two-dimensional crust leads to the following simple algorithm, illustrated in Figure 5. First compute the Voronoi diagram of $S$, and let $V$ be the set of Voronoi vertices. Then compute the Delaunay triangulation of $S \cup V$. The crust consists of the Delaunay edges between points of $S$, since those are the edges with circumcircles empty of points in $S \cup V$. Notice that the crust is also a subset of the Delaunay triangulation of the input points; adding the Voronoi vertices filters out the unwanted edges from the Delaunay triangulation. We call this technique *Voronoi filtering*.

### 4.2 Three Dimensions

This simple Voronoi filtering algorithm runs into a snag in three dimensions. The nice property that all the Voronoi vertices of a sufficiently dense sample lie near the medial axis is no longer true. Figure 6 shows an example. No matter how densely we sample, Voronoi vertices can appear arbitrarily close to the surface.

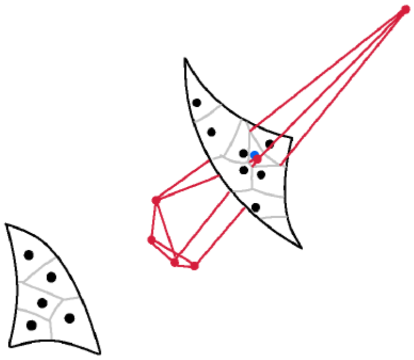

Figure 6. In three dimensions, we can use only a subset of the Voronoi vertices, since not all Voronoi vertices contribute to the approximation of the medial axis. Here, one sample on a curved surface is colored blue, and the edges of its three-dimensional Voronoi cell are drawn in red. One red Voronoi vertex lies near the surface, equidistant from the four samples near the center. The others lie near the medial axis, near the center of curvature on one side and halfway to an opposite patch of the surface on the other.

On the other hand, many of the three-dimensional Voronoi vertices *do* lie near the medial axis. Consider the Voronoi cell $V_s$ of a sample $s$, as in Figure 6. The sample $s$ is surrounded on $F$ by other samples, and $V_s$ is bounded by bisecting planes separating $s$ from its neighbors, each plane nearly perpendicular to $F$. So the Voronoi cell $V_s$ is long, thin and roughly perpendicular to $F$ at $s$. $V_s$ extends perpendicularly out to the medial axis. Near the medial axis, other samples on $F$ become closer than $s$, and $V_s$ is cut off. This guarantees that some vertices of $V_s$ lie near the medial axis. We give a precise and quantitative version of this rough argument in [2].

This leads to the following algorithm. Instead of using all of the Voronoi vertices in the Voronoi filtering step, for each sample $s$ we use only the two vertices of $V_s$ farthest from $s$, one on either side of the surface $F$. We call these the *poles* of $s$, and denote them $p^+$ and $p^-$. It is easy to find one pole, say $p^+$: the farthest vertex of $V_s$ from $s$. The observation that $V_s$ is long and thin implies that the other pole $p^-$ must lie roughly in the opposite direction. Thus in the basic algorithm below, we simply choose $p^-$ to be farthest vertex from s such that $sp^-$ and $sp^+$ have negative dot-product. Here is the basic algorithm:

1. *Compute the Voronoi diagram of the sample points $S$*

2. *For each sample point $s$ do:*

   (a) *If $s$ does not lie on the convex hull, let $p^+$ be the farthest Voronoi vertex of $V_s$ from $s$. Let $n^+$ be the vector $sp+$.*

   (b) *If $s$ lies on the convex hull, let $n^+$ be the average of the outer normals of the adjacent triangles.*

   (c) *Let $p^-$ be the Voronoi vertex of $V_s$ with negative projection on $n^+$ that is farthest from $s$.*

3. *Let $P$ be the set of all poles $p^+$ and $p^-$. Compute the Delaunay triangulation of $S \cup P$.*

4. *Keep only those triangles for which all three vertices are sample points in $S$.*

Notice that one does not need an estimate of $r$ to use the crust algorithm; the basic algorithm requires no tunable parameters at all. The output of this algorithm, the *three-dimensional crust*, is a set of triangles that resembles the input surface geometrically. More precisely, we prove the following theorem [2].

**Theorem 2.** *Let $S$ be an $r$-sample from a smooth surface $F$, for $r \leq .06$. Then 1) the crust of $S$ contains a set of triangles forming a mesh topologically equivalent to $F$, and 2) every point on the crust lies within distance $5r \cdot d(p)$ of some point $p$ on $F$, where $d(p)$ is the distance from $p$ to the medial axis.*

The crust, however, is not necessarily a manifold; for example, it often contains all four triangles of a very flat "sliver" tetrahedron. It is, however, a visually acceptable model.

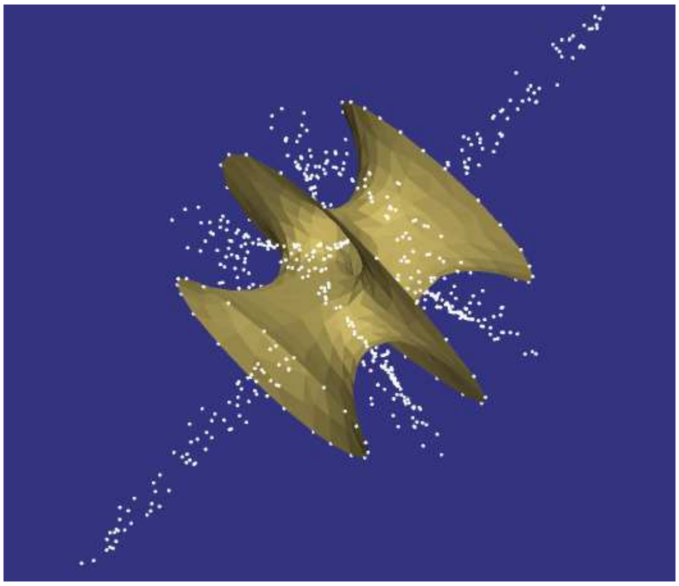

Figure 7. The crust of a set of sample points and the poles (white points) used in its reconstruction. Each sample selects the two vertices of its Voronoi cell that are farthest away, one on either side of the surface, as poles. The poles lie near the medial axis of the surface, sketching planes separating opposite sheets of surface that degenerate to one-dimensional curves where the cross-section of the surface is circular.

## 4.3 Normal Estimation and Filtering

Additional filtering is required to produce a guaranteed piecewise-linear manifold homeomorphic to $F$, and to ensure that the output converges in surface normal as the sampling density increases.

In fact, whatever the sampling density, the algorithm above may output some very thin crust triangles nearly perpendicular to the surface. We have an important lemma [2], however, which states that the vectors $n^+ = sp^+$ and $n^- = sp^-$ from a sample point to its poles are guaranteed to be nearly orthogonal to the surface at $s$. The angular error is linear in $r$. The intuition (put nicely by Ken Clarkson) is that the surface normal is easy to estimate from a point far away, such as a pole $p$, since the surface must be nearly normal to the largest empty ball centered at $p$.

We can use these vectors in an additional *normal filtering* step, throwing out any triangles whose normals differ too much from $n^+$ or $n^-$. When normal filtering is used, the normals of the output triangles approach the surface normals as the sampling density increases. We prove in [2] that the remaining set of triangles still contains a subset forming a piecewise-linear surface homeomorphic to $F$.

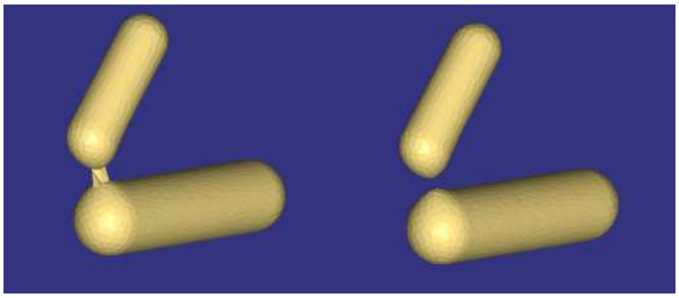

Figure 8. The crust of points distributed on an implicit surface (left). The additional normal filtering step is needed to separate the two connected components (right), which are undersampled at their closest point. Triangles are deleted if their normals differ too much from the direction vectors from the triangle vertices to their poles. These vectors are provably close to the surface normals.

Normal filtering can be useful in practice as well, as shown in Figure 8. In the usual case in which $r$ is unknown the allowable difference in angle must be selected experimentally. Normal filtering can be dangerous, however, at boundaries and sharp edges. The directions of $n^+$ and $n^-$ are not nearly normal to all nearby tangent planes, and desirable triangles might be deleted.

We note that $n^+$ and $n^-$, our Voronoi-based estimates of normal direction, could be useful in the zero-set reconstruction methods, which depend on accurate estimation of the tangent planes. For the algorithm of Hoppe et al. [14], a Voronoi-based estimate could replace the estimate based on the $k$-nearest neighbors. The Voronoi-based estimate has the advantage that it is not sensitive to the distribution; whereas, for instance, on medical image data, all $k$ nearest neighbors might lie in the same slice, and so would the estimated tangent plane. In the algorithm of Curless and Levoy [8], the Voronoi-based estimate could be checked against the bounds on normal direction derived from the laser-range scanner.

### 4.4 Manifold Extraction

After the normal filtering step, all the remaining triangles are roughly parallel to the surface. We can define a sharp edge as one which is adjacent to triangles only on one side of a plane through the edge and roughly perpendicular to the surface. Notice that an edge of degree one counts as a sharp edge. If the surface $F$ is indeed a smooth manifold without boundary, we are guaranteed that the normal-filtered crust contains a piecewise-linear manifold homeomorphic to $F$. Any triangle adjacent to a sharp edge cannot belong to this piecewise-linear manifold, and can be safely deleted. We continue recursively until no such triangle remains. A piecewise-linear manifold can then be obtained by a *manifold extraction* step which takes the outside surface of the remaining triangles on each connected component. This simple approach, however, cannot be applied when $F$ is not a smooth manifold without boundary. In that case we do not know how to prove that we can extract a manifold homeomorphic to $F$.

### 4.5 Complexity

The asymptotic complexity of the crust algorithm is $O(n^2)$ where $n = |S|$, since that is the worst-case time required to compute a three-dimensional Delaunay triangulation. Notice that the number of sample points plus poles is at most $3n$. As has been frequently observed, the worst-case complexity for the three-dimensional Delaunay triangulation almost never arises in practice. All other steps are linear time.

## 5 Implementation

### 5.1 Numerical Issues

Robustness has traditionally been a concern when implementing combinatorial algorithms like this one. Our straightforward implementation, however, is very robust. This success is due in large part to the rapidly improving state of the art in Delaunay triangulation programs. We used Clarkson's *Hull* program. *Hull* uses exact integer arithmetic, and hence is thoroughly robust, produces exact output, and requires no arithmetic tolerancing parameters. The performance cost for the exact arithmetic is fairly modest, due to a clever adaptive precision scheme. We chose *Hull* so that we could be sure that numerical problems that arose were our own and did not originate in the triangulation. Finding the exact Delaunay triangulation is not essential to our algorithm.

*Hull* outputs a list of Delaunay tetrahedra, but not the coordinates of their circumcenters (the dual Voronoi vertices) which always contain some roundoff error. Fortunately, the exact positions of the poles are not important, as the numerical error is tiny relative to the distance between the poles and the surface. We computed the location of each Voronoi vertex by solving a $4 \times 4$ linear system with a solver from *LAPACK*. The solver also returns the condition number of the coefficient matrix, which we used to reject unreliable Voronoi vertices. Rejected Voronoi vertices were almost always circumcenters of "slivers" (nearly planar tetrahedra) lying flat on the surface; for a good sample such vertices cannot be poles. It is possible that this method also rejects some valid poles induced by very flat tetrahedra spanning two patches of surface. We have not, however, observed any problems in practice. Presumably there is always another Voronoi vertex nearby that makes an equally good pole.

### 5.2 Efficiency

Running times for the reconstruction of some large data sets are given in the table below; the reconstructions are shown in Figure 9. We used an SGI Onyx with 512M of memory.

| Model | Time (min) | Num. Pts. |
|---|---|---|
| Femur | 2 | 939 |
| Golf club | 12 | 16864 |
| Foot | 15 | 20021 |
| Bunny | 23 | 35947 |

The running time is dominated by the time required to compute the Delaunay triangulations. *Hull* uses an incremental algorithm [7], so the running time is sensitive to the input order of the vertices. The triangulation algorithm builds a search structure concurrently with the triangulation itself; the process is analogous to sorting by incrementally building a binary search tree. When points are added in random order, the search structure is balanced (with extremely high probability) and the expected running time is optimal. In practice, random insertions are slow on large inputs, since both the search structure and the Delaunay triangulation begin paging. We obtained better performance by first inserting a random subset of a few thousand points to provide a balanced initial search structure, and then inserting the remaining points based on a crude spatial subdivision to improve locality.

Most likely much greater improvements in efficiency can be achieved by switching to a three-dimensional Delaunay triangulation program that, first, does not use exact arithmetic, and second, uses an algorithm with more locality of reference.

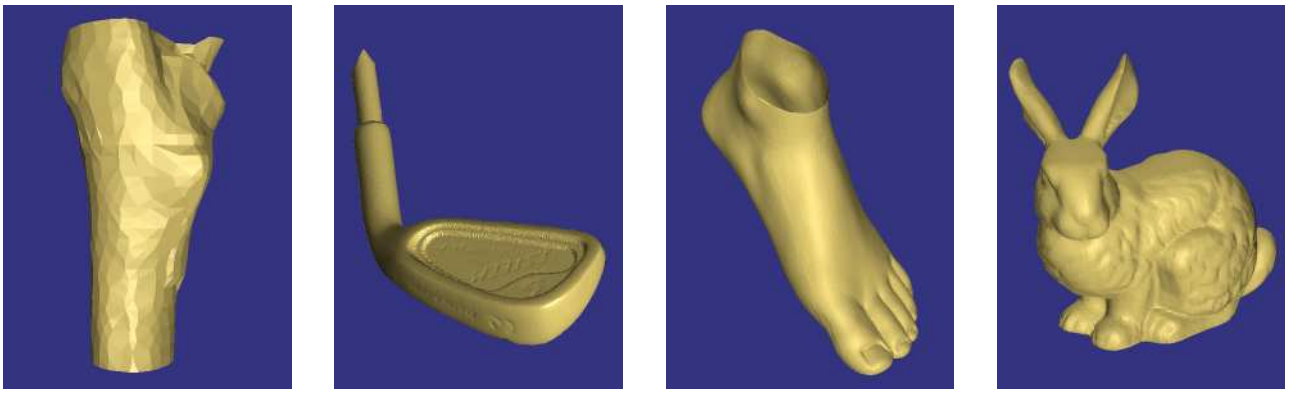

Figure 9. Femur, golf club, foot and bunny reconstructions. Notice the subtle "3" on the bottom of the club (apparently a 3-iron), showing the sensitivity of the algorithm. The foot, like all our reconstructions, is hollow. The bunny was reconstructed from the roughly 36K vertices of the densest of the Stanford bunny models in 23 minutes.

# 6 Heuristic Modifications

As we have noted, our algorithm does not do well at sharp edges, either in theory or in practice. The reason is that the Voronoi cell of a sample $s$ on a sharp edge is not long and thin, so that the assumptions under which we choose the poles is not correct. For example, the Voronoi cell of a sample $s$ on a right-angled edge is roughly fan-shaped. The vector $n^+$ directed towards the first pole of $s$ might be perpendicular to one tangent plane at $s$, but parallel to the other. The second pole would then be chosen very near the surface, punching a hole in the output mesh.

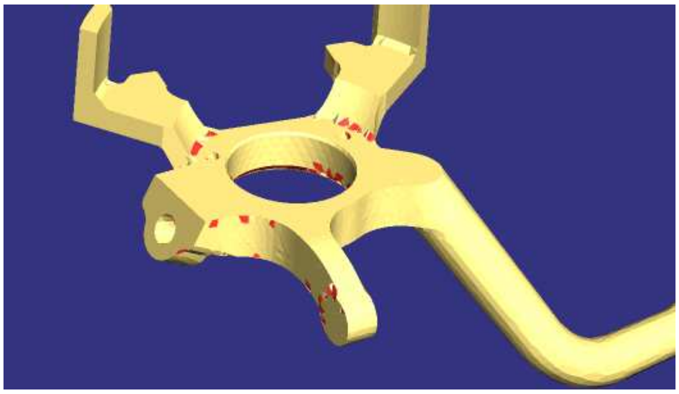

Figure 10. We resolve the sharp edges on this model of a mechanical part by using the two farthest Voronoi vertices as poles, regardless of direction. The basic algorithm forces the poles to lie in opposite directions, but is only guaranteed to work properly on a smooth surface. The red triangles do not appear in the reconstruction when using the basic algorithm.

We experimented with other methods for choosing the second pole. We found that choosing as $p^-$ the Voronoi vertex with the greatest negative projection in the direction $n^+$ gave somewhat better results. This modification should retain the theoretical guarantees of the original algorithm. The best reconstructions, however, were produced by a different heuristic: choosing the farthest and the second farthest Voronoi vertices, regardless of direction, as the two poles (see Figure 10). This heuristic is strongly biased against choosing poles near the surface, avoiding gaps near sharp edges but sometimes allowing excess triangles filling in sharp corners. We believe that pathological cases could be constructed in which this fill causes a topologically incorrect reconstruction irrespective of the sampling density.

Boundaries pose similar problems in theory, but the reconstructions produced by the crust algorithm on surfaces with boundaries are usually acceptable. Figure 7 and the foot in Figure 9 are examples of perfectly reconstructed boundaries. When the boundary forms a hole in an otherwise flat surface, with no other parts of the surface nearby, the crust algorithm fills in the hole.

Undersampling also causes holes in the output mesh. For example, consider a sample in the middle of a a flat plate. Although its second pole lies in the correct direction, if there are two few sample points on the opposite side of the plate, the pole may fall near the surface on the opposite side and cause a hole. We experimented with heuristics to compensate for this undersampling effect, and for similar reconstruction errors in undersampled cylindrical regions. We found that moving all poles closer to their samples by some constant fraction allowed thin plates and cylinders to be reconstructed from fewer samples, while sometimes introducing new holes on other parts of the model. We were sometimes able to get a perfect reconstruction by taking the union of a crust made with this modification and one without.

# 7 Research Directions

We have identified a number of future research directions.

## 7.1 Noise

Small perturbations of the input points do not cause problems for the crust algorithm, nor do a few outliers. But when the noise level is roughly the same as the sampling density, the algorithm fails, both in theory and in practice. We believe, however, that there is a Voronoi-based algorithm, perhaps combining aspects of crusts and $\alpha$-shapes, that reconstructs noisy data into a "thickened surface" containing all the input points, some of them possibly in the interior. See Melkemi [15] for some suggestive experimental work in $\mathbb{R}^2$.

## 7.2 Sharp Edges and Boundaries

We would like to modify the crust algorithm to handle surfaces with sharp edges and to provide theoretical guarantees for the reconstruction of both sharp edges and boundaries. Interpolating reconstruction algorithms like ours have an advantage here, since approximating reconstruction algorithms smooth out sharp edges. One important goal is to develop reliable techniques for identifying samples that lie on sharp edges or boundaries. As noted, the Voronoi cells of such samples are not long and thin. This intuition could be made precise, and perhaps combined with more traditional filtering techniques.

### 7.3 Using Surface Normals

A variation on the problem is the reconstruction of surfaces from unorganized points that are equipped with normal directions. This problem arises in two-dimensional image processing when connecting edge pixels into edges. In three dimensions, laser range data comes with some normal information, and we have exact normals for points distributed on implicit surfaces. It should be possible to show that with this additional information, reconstruction is possible from much sparser samples. In particular, when normals are available, dense sampling should not be needed to resolve the two sides of a thin plate, suggesting that a different sampling criterion than distance to medial axis is required.

### 7.4 Compression

One intriguing potential application (pointed out by Frank Bossen) of interpolating, rather than approximating reconstruction, is that it can be used as a lossless mesh compression technique. A model created by interpolating reconstruction can be represented entirely by its vertices, and no connectivity information at all must be stored. A model which differs only slightly from the reconstruction of its vertices can be represented by the vertices and a short list of differences. These differences might be encoded efficiently using some geometrically defined measure of "likelihood" on Delaunay triangles. The vertices themselves could then be ordered so as to optimize properties such as compressibility or progressive reconstruction by an incremental algorithm. With the current best geometry compression method [16], most of the bits are already used to encode the vertex positions, rather than connectivity, but the connectivity is encoded in the ordering of the vertices. Allowing arbitrary vertex orderings could improve compression; we are experimenting with an octree encoding.

Our current crust algorithm is not incremental, and our implementation is too slow for real-time decompression, so this application motivates work in both directions.

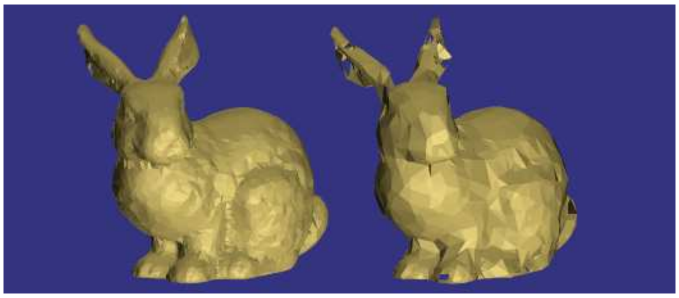

Figure 11. Reconstructions from subsets of the samples resemble the final reconstructions. The crust of the first 5 % of the points in an octree encoding of the bunny samples is still quite recognizable (right); the crust of 20 % of the points is on the left. Rough reconstructions like these could be shown during progressive transmission.

## Acknowledgments

We thank David Eppstein (UC–Irvine) for his collaboration in the early stages of this research, and Frank Bossen (EPF–Lausanne) and Ken Clarkson (Lucent) for interesting suggestions. We thank Ping Fu (Raindrop Geomagic) for the fist and the mechanical part, Hughes Hoppe (Microsoft) for the head, the golf club and the foot, Chandrajit Bajaj (UT–Austin) for the femur, Paul Heckbert (CMU) for the hot dogs, and the Stanford Data Repository for the bunny. We thank Ken Clarkson and Lucent Bell Labs for *Hull*, and The Geometry Center at the University of Minnesota for *Geomview*, which we used for viewing and rendering the models.

## References

[1] Nina Amenta, Marshall Bern and David Eppstein. The Crust and the $\beta$-Skeleton: Combinatorial Curve Reconstruction. To appear in *Graphical Models and Image Processing*.

[2] Nina Amenta and Marshall Bern. Surface reconstruction by Voronoi filtering. To appear in *14th ACM Symposium on Computation Geometry*, June 1998.

[3] D. Attali. $r$-Regular Shape Reconstruction from Unorganized Points. In *13th ACM Symposium on Computational Geometry*, pages 248–253, June 1997.

[4] C. Bajaj, F. Bernardini, and G. Xu. Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. *SIGGRAPH '95 Proceedings*, pages 109–118, July 1995.

[5] F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds using $\alpha$-shapes, In *9th Canadian Conference on Computational Geometry*, pages 193–198, August 1997.

[6] J-D. Boissonnat. Geometric structures for three-dimensional shape reconstruction, *ACM Transactions on Graphics* 3: 266–286, 1984.

[7] K. Clarkson, K. Mehlhorn and R. Seidel. Four results on randomized incremental constructions. *Computational Geometry: Theory and Applications*, pages 185–121, 1993.

[8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96 Proceedings*, pages 303–312, July 1996.

[9] H. Edelsbrunner, D.G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane, *IEEE Transactions on Information Theory* 29:551-559, (1983).

[10] H. Edelsbrunner and E. P. Mücke. Three-dimensional Alpha Shapes. *ACM Transactions on Graphics* 13:43–72, 1994.

[11] L. H. de Figueiredo and J. de Miranda Gomes. Computational morphology of curves. *Visual Computer* 11:105–112, 1995.

[12] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces, In *SIGGRAPH '94 Proceedings*, pages 269–277, July 1994.

[13] H. Hoppe. Surface Reconstruction from Unorganized Points. Ph.D. Thesis, Computer Science and Engineering, University of Washington, 1994.

[14] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface Reconstruction from Unorganized Points. In *SIGGRAPH '92 Proceedings*, pages 71–78, July 1992.

[15] M. Melkemi, $\mathcal{A}$-shapes and their derivatives, In *13th ACM Symposium on Computational Geometry*, pages 367–369, June 1997

[16] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *Research Report RC20340*, IBM, 1996.