The Power Crust

Nina Amenta Sunghee Choi Ravi Krishna Kolluri *
University of Texas at Austin

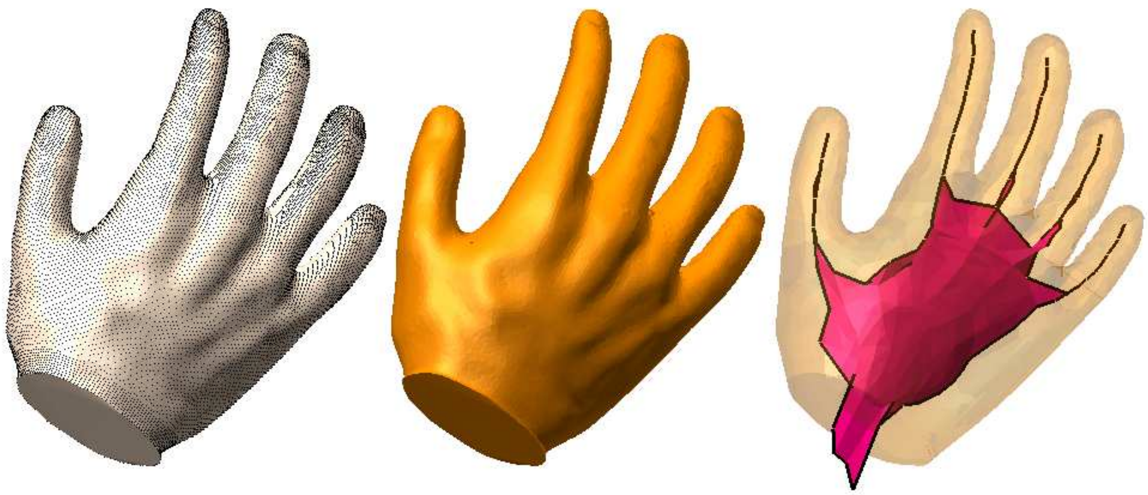

Figure 1. Laser range data, the reconstructed watertight polygonal model, and its simplified medial axis.

Abstract

The *power crust* is a construction which takes a sample of points from the surface of a three-dimensional object and produces a surface mesh and an approximate medial axis. The approach is to first approximate the medial axis transform (MAT) of the object. We then use an inverse transform to produce the surface representation from the MAT.

This idea leads to a simple algorithm with theoretical guarantees comparable to those of other surface reconstruction and medial axis approximation algorithms. It also comes with a guarantee that does not depend in any way on the quality of the input point sample. *Any* input gives an output surface which is the 'watertight' boundary of a three-dimensional polyhedral solid: the solid described by the approximate MAT. This unconditional guarantee makes the algorithm quite robust and eliminates the polygonalization, hole-filling or manifold extraction post-processing steps required in previous surface reconstruction algorithms.

In this paper, we use the theory to develop a power crust implementation which is indeed robust for realistic and even difficult samples. We describe the careful design of a key subroutine which labels parts of the MAT as inside or outside of the object, easy in theory but non-trivial in practice. We find that we can handle areas in which the input sampling is scanty or noisy by simply discarding the unreliable parts of the MAT approximation. We demonstrate good empirical results on inputs including models with sharp corners, sparse and unevenly distributed point samples, holes, and noise, both natural and synthetic.

We also demonstrate some simple extensions: intentionally leaving holes where there is no data, producing approximate offset surfaces, and simplifying the approximate MAT in a principled way to preserve stable features.

1 Introduction

Surface reconstruction is increasingly important in geometric modeling for generating surfaces from data points captured from real objects, often by laser range scanners but also by hand-held digitizers, computer vision techniques, edge detection from medical images, or other technologies. Industrial applications include reverse engineering, product design and the construction of personalized medical appliances.

The medial axis transform, or MAT, is a skeletal shape representation which has been proposed as a tool for various applications in shape recognition and manipulation. It represents a solid by the set of maximal balls completely contained in the interior, rather than by the set of points on the boundary (see Figure 2a). Computing the exact MAT, given a surface representation, is difficult, but a simplified approximate MAT is, in any case, often more useful.

The *power crust* algorithm constructs piecewise-linear approximations to both the object surface and the MAT given an input point sample S from the object surface. The algorithm is to first use S to approximate the MAT, and then apply an *inverse transform* to the MAT to produce a piecewise-linear surface approximation. The main tools we use are the *Voronoi diagram* and a convenient kind of weighted Voronoi diagram called the *power diagram*; these are defined in Section 2.

A two-dimensional version of the algorithm is shown in Figure 2.

*Computer Sciences Department, Austin, TX 78712, USA. Supported by NSF/CCR-9731977. amenta@cs.utexas.edu

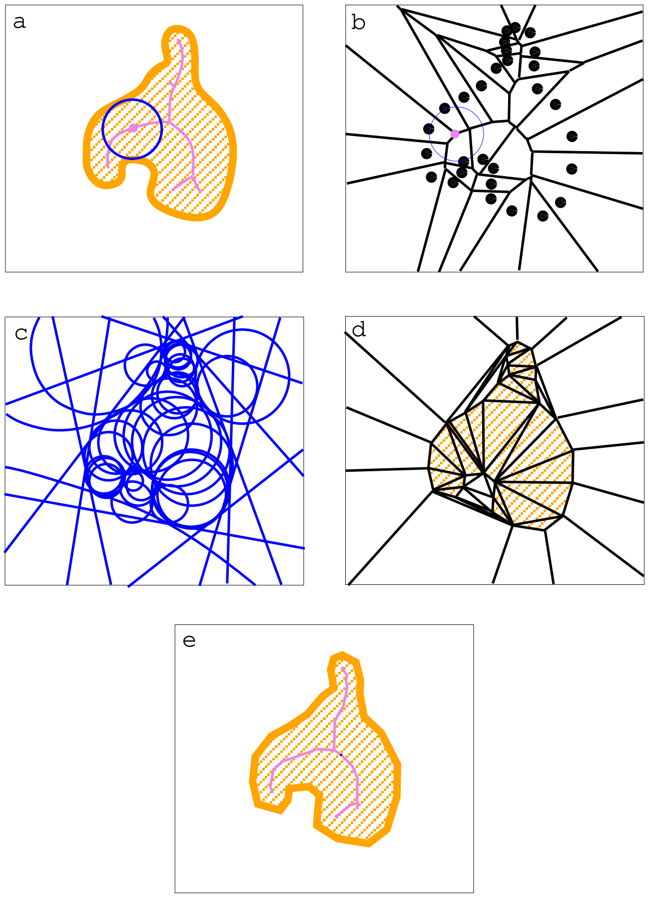

Figure 2. Two-dimensional example of power crust construction. a) An object with its medial axis; one maximal interior ball is shown. b) The Voronoi diagram of S, with the Voronoi ball surrounding one pole shown. In 2D, we can select all Voronoi vertices as poles, but not in 3D. c) The inner and outer polar balls. Outer polar balls with centers at infinity degenerate to halfspaces on the convex hull. d) The power diagram cells of the poles, labeled inner and outer. e) The power crust and the power shape of its interior solid.

The MAT is approximated by a subset V of the Voronoi vertices of S, called the *poles*, which (where S is a good sample) lie near the medial axis. The balls surrounding the poles and touching the nearest samples are the *polar balls*. The polar balls approximate maximal balls contained in the interior or exterior of F; the radii of the polar balls define weights on the poles. We approximate the inverse transform using the *power diagram* of the set of weighted poles which, like the usual Voronoi diagram, divides space into polyhedral cells. We label a subset of the power diagram cells as representing the interior of the object. The subset of the two-dimensional polygonal faces of the power diagram which separate these inner cells from the outer cells forms our output surface, the *power crust*. We connect the inner poles according to the connectivity of their power diagram cells, forming a simplicial complex approximating the interior medial axis, which we call the *power shape*.

The Voronoi diagram is computationally expensive. For the billion point sets collected by long-term projects like the Digital Michelangelo Project [35] or the IBM digitization of Michelangelo's Pietá [8], computing the Voronoi diagram with current codes is not feasible. Codes exist, however, that can handle the hundred thousand or few million points of many industrial applications.

The advantage of using the Voronoi diagram is that it produces global information about the shape of the object and the distance function induced by its surface on $\mathbb{R}^3$. The approximate MAT represents this information. Besides enabling later applications, it also simplifies the surface reconstruction process: on *any* input, the power crust is the 'watertight' boundary of the three-dimensional solid described by the approximate MAT, with no clean-up post-processing steps required.

Other theoretical guarantees on the quality of the output depend on assumptions about the object surface and the density of the input sample. In a companion paper [3], we prove theoretical guarantees for the power crust which are similar to those of previous Voronoi-diagram based surface reconstruction algorithms, under similar assumptions. We also give topological and geometric guarantees on the quality of the reconstructed medial axis, similar to those of [23].

Such theoretical results, of course, do not necessarily imply that an algorithm is practical. Real input data sets rarely meet any precise sampling requirement everywhere. Some of the ideas that are important in the development of the theory do prove to be useful, however, in developing a robust implementation.

One important area in which the implementation needs to be well designed is in the crucial subroutine which labels each cell in the power diagram as inside or outside of the object. We perform this labeling by traversing the cell structure. While in theory any traversal suffices, in practice it may not be clear how labels should be propagated in places where the sampling assumptions fail. We use lemmas about the local geometry to define a weighting heuristic for propagating the labels in which we have the most confidence first.

Another use of the theoretical results is to detect where the MAT approximation is unreliable. We find that simply discarding unreliable poles and using the remaining part of the MAT approximation gives reasonable outputs. This idea is useful in two contexts: when there is noise in the data, and for models with sharp corners. Sharp corners are a problem for all surface reconstruction algorithms, but reconstructing the sharp corners of mechanical parts is an important problem with industrial applications. Noise arises in laser range scanner input, mostly when several scans are combined, and small errors in alignment produce point samples that lie near, but not on, the surface.

Finally, we demonstrate three simple but useful extensions of the algorithm. First, we consider intentionally leaving holes in the output surface models where there are large holes in the input data, for instance at the bottom of the hand in Figure 1. We characterize large holes by examining the polar balls corresponding to the cells on either side of the power diagram face.

Second, we construct approximate offset surfaces. The exact offset surface, like the exact MAT, is difficult to compute. But an approximate offset surface sometimes suffices, for instance when constructing the inside surface of a mold based on a model of the outside surface. We get an approximate offset surface just by increasing the radius of every outer polar ball, and decreasing the radius of every inner polar ball, by a constant.

Third, we demonstrate a well-founded simplification heuristic for the approximate medial axis. The medial axis of a complicated object is not really useful for applications like shape decomposition or feature recognition, since the medial axis is 'unstable' - small perturbations of the surface can produce large changes in the true medial axis. We produce simplified approximations that are demonstrably impervious to noise.

Related work

There is a large body of related work, concerning surface reconstruction, the three-dimensional MAT, MAT simplification and our main tool, the Voronoi diagram.

Surface reconstruction: Earlier in computer graphics, Uselton [48] and others investigated the problem of surface reconstruction. Hoppe et al. [32] gave a clean abstraction of the reconstruction problem. Their solution approximated the signed distance function induced by the surface F, and constructed the output surface as a polygonal approximation of the zero-set of this function. Curless and Levoy [18] gave a really effective algorithm which represents the signed distance function on a voxel grid. Their algorithm is designed for very large data sets, and is used in the Digital Michelangelo Project [35]. The introduced a post-processing step for hole-filling.

Amenta, Bern and Kamvysselis [2] first gave an algorithm with theoretical guarantees, and also defined the poles. Our theoretical results [3] are comparable to theirs, and to those in a later and simpler version due to Amenta,Choi,Dey and Leekha [4]. Tamal Dey [19] reports that he has extended this algorithm to handle objects with sharp corners. All these algorithms have the drawback that they produce a collection of triangles, from which a triangulated manifold must be extracted in post-processing.

Our algorithm is most similar to an old algorithm of Boissonnat [11], which labels a subset of the Delaunay tetrahedra of S as the interior of the solid; we eliminate some ambiguities in that labeling algorithm by working with the power diagram of the poles. Boissonnat and Cazals [12] have recently given an algorithm which reconstructs a smooth surface related to the Sibson interpolant of the sample points, also with theoretical guarantees.

Another algorithm based on the Delaunay triangulation is the α-shape algorithm of Edelsbrunner and Mücke [20]. We use many of the ideas developed in the context of α-shapes, although in a different way. In particular, our inverse transform algorithm and the definition of the power shape is based on the theory of *weighted* α-shapes [21]. A drawback of α-shapes for surface reconstruction is that when the sampling is non-uniform it is difficult and sometimes impossible to choose α so as to balance hole-filling against loss of detail.

Bernardini et al. [8] have developed a system based, conceptually, on α-shapes, while *avoiding* the computation of the Voronoi diagram. This allows them to reconstruct larger models. It would be very interesting to combine their ideas with the new geometric definitions we use here, which avoid the drawbacks of α-shapes. Another tracing algorithm [27] runs quickly but is unnecessarily sensitive to the distribution of the samples.

Medial axis construction and approximation:

The three-dimensional MAT has been proposed as a tool for various applications [30, 45] including object decomposition for mesh generation [41, 42], bounding objects for collision detection [34], CAD [10], shape morphing and animation [40, 46], and motion planning [15, 28, 51].

Computing the exact three-dimensional MAT directly is difficult because of numerical issues. Hoffman [31] gave an early algorithm for the MAT of CSG objects. Finding the exact MAT of polyhedra given by their boundaries has only recently been demonstrated by Culver, Keyser and Manocha [17] using a tracing algorithm; see also [43, 36]. But computing the exact MAT of a complicated object model captured from real-world data is probably overkill; a simplified approximate MAT suffices for applications such as computing bounding volumes, decomposing solids, morphing, or motion planning.

The idea of approximating the three-dimensional MAT using a subset of the Voronoi vertices of a sample of points on the object surface is well-known [5, 13, 34, 41, 46, 47], and the connection to Voronoi-based surface reconstruction methods has been notes as well [2, 6, 26]. When the object boundary is given as well as the point sample, it can be used to discard spurious parts of the Voronoi diagram and to refine the approximation [47]. Our use of the power diagram, and the resulting definitions of the power crust and the power shape are new.

Another approach is to approximate the MAT using an octree [24, 50]. Etzion proves that for polyhedra this method converges in the limit to the true MAT, and uses an approximate, but combinatorially correct, MAT to construct an exact MAT for simple objects [23].

MAT simplification: The MAT is 'unstable' - small perturbations in the object boundary can induce large features in the medial axis. As a result, bumpy objects like the shell in Figure 18 have medial axes which are too complicated to be very useful.

This common observation is somewhat misleading, however. *Parts* of the medial axis are unstable, and other parts, corresponding to significant object features impervious to noise, are quite stable. Several researchers have noted, some by analyzing the action of smoothing functions on the boundary [7], some by considering the effect of noise on the medial axis [5, 6], some by studying the effects of sampling on MAT approximation [3, 12], and still others by experience [45, 38, 37], that the maximum solid angle γ formed by the vectors connecting a medial axis point m to its closest surface points is related to the stability of m, as is the radius of the maximal ball centered at m. These criteria have been used for simplification of three-dimensional medial axes [5, 45]. We use a very simple formulation of this idea to eliminate parts of the medial axis induced by features below a given noise threshold.

Voronoi diagram computation: The applicability of the power crust depends on the feasibility of computing the three-dimensional Voronoi diagram, or more specifically its dual the Delaunay triangulation, so we briefly review the state of the art. Ten years ago there were no robust three dimensional Delaunay codes, and simple randomized theoretical algorithms [16, 29] were just being developed. The first program based on these methods, `qhull`, uses floating point arithmetic with user-specified tolerances for robustness. Somewhat later programs such as `hull`, the free open-source code which we use, employ adaptive precision exact arithmetic, evaluating geometric predicates only to the precision required to produce a combinatorially correct output. In our experience, `hull` is about four times slower than `qhull`, but it has the advantage of being completely robust.

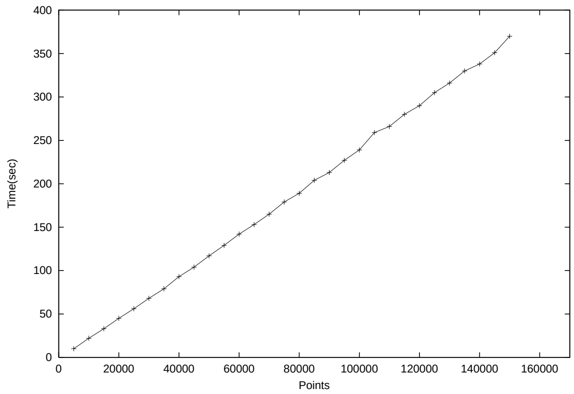

Figure 3. The time required for `hull` to compute the three-dimensional Delaunay triangulation of a set of random points distributed uniformly on the surface of an ellipsoid scales linearly. Wall-clock time was measured on a 400MHz Sun with 1G memory.

Efficient adaptive precision exact arithmetic is an active research area, and the next generation of 3D Delaunay codes, now in the late stages of development, are reported to be about an order of magnitude faster than `hull`. Jonathan Shewchuk [44], at Berkeley, has computed the Delaunay triangulation of 20 million points, and can produce the Delaunay triangulation of 10,000 samples from an object surface at 1.8 seconds on a Pentium II. At INRIA, Boissonnat and Cazals [12] report that Devillers' 3D Delaunay triangulation code handles a comparable 500,000 randomly distributed points per minute (performance is slightly worse for points distributed on a surface).

Scalability is also an important factor. The *worst-case* complexity of a three-dimensional Delaunay triangulation is $O(n^2)$, but in general only contrived examples achieve this bound. Experience suggests that many real instances require linear time. To illustrate this bit of 'folklore', we computed the Delaunay triangulation of different numbers of points distributed on an ellipsoid. The running times are shown in Figure 3.

Alternatively, one could use an approximate Voronoi diagram computation based on space subdivision (eg. [23],[50]) instead of the exact Voronoi diagram. This approach might be capable of handling larger inputs. We decided to begin by developing an implementation of the exact algorithm, however, since it is actually easier using current Delaunay codes, and it demonstrates the strengths and weaknesses inherent in the power crust construction rather than possible artifacts of the approximation method.

2 Geometric definitions

Let F be the boundary of a three-dimensional object. To avoid having to deal with points at infinity, we assume that F is contained in a bounded open region Q. F divides Q into *interior* and *exterior* solids.

2.1 Medial axis transform

We represent a ball $B = B_{c,\rho} \subset Q$ by its center c and radius ρ. We say B is *empty* (with respect to F) if the interior of B contains no point of F. A *medial ball* is a maximal empty ball; that is, it is completely contained in no other empty ball. The center of a medial ball is either a point with more than one closest point on F, or a center of curvature of F.

Definition: The *medial axis transform* of surface F is the set of medial balls. The centers of the medial balls form the *medial axis* of F; the MAT includes the radii as well.

We could equivalently define the medial axis as the closure of the set of all points with more than one closest point on F. Notice that either way the medial axis includes both a part inside of F and a part outside of F.

The medial axis of a three-dimensional solid is generally a (non-regular) two-dimensional surface, but it accurately reflects the topology of the solid in that it has the same connected components, loops, and so on. Formally, the medial axes of a solid is *homotopy equivalent* to the solid [14].

2.2 Voronoi diagram

The Voronoi diagram of a set $S \subset Q$ of sample points is the subdivision of Q into cells, each cell consisting of the points $x \in Q$ closest to a particular input point $s \in S$ (see Figure 2b). Each Voronoi cell is a convex polyhedron, and the vertices of these polyhedra are the *Voronoi vertices*. A Voronoi vertex v in $\mathbb{R}^3$ is shared by the cells of at least four samples. The *Voronoi ball* centered at v has these samples on its boundary, and no samples in its interior. Notice the analogy to the maximal empty balls in the definition of the medial axis: a Voronoi ball is a maximal empty ball with respect to S.

The well-known idea of approximating the medial axis with the Voronoi vertices is motivated by this analogy. In fact the vertices of the two-dimensional Voronoi diagram approach the medial axis as the sampling density goes to infinity, as expected. In three dimensions, however, the Voronoi diagram can, and usually does, contain vertices very close to the surface and far from the medial axis, even when F is smooth and the sampling is noise-free and arbitrarily dense. This is because four sample points close together on F can determine a Voronoi ball whose center lies very near, or right on, F itself [1, 2]. Such a Voronoi vertex is dual to a very small flat Delaunay tetrahedron. In two dimensions, in contrast, a Voronoi circle determined by three points close together on a curve has to lie far from the curve and near the medial axis.

2.3 Power diagram

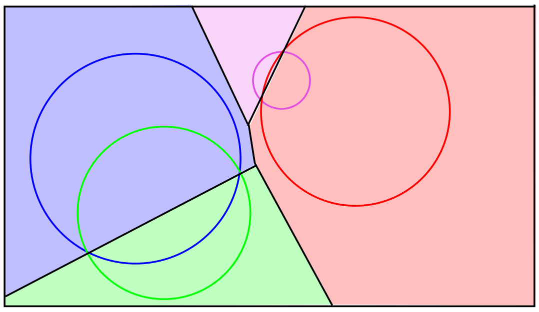

Figure 4. The *power diagram* of four weighted points. A point c with weight ρ^2 is represented by a ball centered at c with radius ρ.

Our plan is to use the Voronoi diagram to approximate the medial axis with a finite set of balls. Edelsbrunner [21] showed that finite sets of balls are intimately related to *power diagrams*, a kind of weighted Voronoi diagram. We can think of a ball $B_{c,\rho}$ as a point c with weight ρ^2. The *power distance* between an ordinary unweighted point x in $\mathbb{R}^3$ and $B_{c,\rho}$ is

$$d_{pow}(x, B_{c,\rho}) = d^2(c,x) - \rho^2$$

where function d represents the usual Euclidean distance. When x is inside the ball $B_{c,\rho}$, d_{pow} is negative, and when x is outside, d_{pow} is positive.

We can use d_{pow} to define a weighted Voronoi diagram, the *power diagram*. The power diagram is the subdivision of Q into cells, each cell consisting of the points $x \in Q$ closest, in power distance, to a particular input point $v \in V$. Figure 4 shows a two-dimensional example.

The advantage of using d_{pow} rather than some more natural weighted distance function is that the power diagram has polyhedral cells and can be computed by essentially the same algorithm as the usual Voronoi diagram. As in Figure 4, the face of the power diagram separating the cells of two intersecting balls is a subset of the plane in which the boundaries of the two balls intersect.

3 Constructions

Our constructions of the poles, the power crust and the power shape are based on special properties of the Voronoi diagram of a set of points densely distributed on an object surface. To prove theoretical guarantees about the quality of our algorithm requires some assumption about the quality of the input sample S with respect to the surface F; in practice, we want to implement the algorithm so as to give good results even when this assumption is not met.

3.1 Sampling assumption

We assume, for the analysis, that surface F is smooth and without boundary. Our assumption about the density of sample S is taken from [1, 2]:

Definition: The *Local Feature Size* function, $LFS(x)$, is the minimum Euclidean distance from point x to any point of the medial axis. S is an *r-sample* from F when the distance from any point $x \in F$ to the nearest sample is at most $rLFS(x)$.

Here $LFS(x)$ serves as a local measure of "level of detail" on F; when the medial axis is close to the surface either the curvature is high or some other patch of the surface is nearby. Note that when F is smooth, the distance from any point to the medial axis is strictly greater than zero. At a sharp corner of F the medial axis meets the surface, and according to our definition the sampling density would have to be infinite.

3.2 Poles

Where the sampling is dense, the Voronoi cell of every sample s is long and skinny and perpendicular to the surface. This happens because in directions tangent to the surface the Voronoi cell is bounded by the proximity of other samples on the same local patch of surface; the reader may wish to refer to Figure 2b for some intuition. We quantify this idea in Lemma 1, below. The Voronoi cell of s extends perpendicularly away from the surface. It cannot extend much farther than the medial axis, because there s ceases to be the closest surface point and samples on some other patch of surface will be closer. Thus, the Voronoi vertices at the two ends of the long, skinny Voronoi cell should lie near the medial axis.

This motivates the selection of the *poles* [1, 2] as an approximation of the medial axis:

Definition: The *poles* of a sample $s \in S$ are the farthest vertex of its Voronoi cell in the interior of F and the farthest vertex of its Voronoi cell on the exterior of F. Let V be the set of poles, for all $s \in S$.

When S is a dense sample, the set V of poles excludes Voronoi vertices close to F and forms a good estimate of the medial axis, albeit as a discrete set of points. Note also that the vectors from s to its poles approximate the surface normals at s.

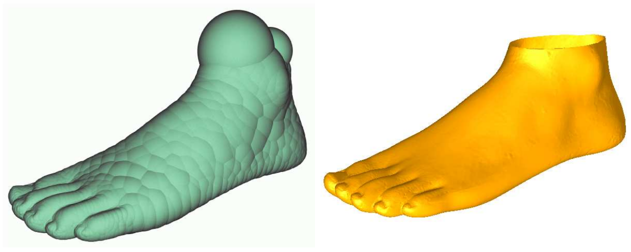

Figure 5. A set of inner polar balls and the resulting three-dimensional power crust. The opening at the top of the foot was detected because large inner polar balls protrude out of the model, and intentionally left as a hole; see Section 7.1.

To avoid dealing with infinity, we add a set Z of eight points, the vertices of a large box surrounding F, to S, so that both poles of each sample in S are bounded. In our implementation, this box is five times larger than the minimum bounding box of S.

Each pole v is the center of a Voronoi ball, which we shall call its *polar ball*. The set of polar balls for all $v \in V$ gives our approximation of the medial axis transform: the MAT is an infinite set of balls, and the approximation is the similar finite set of polar balls.

The polar balls corresponding to poles inside of F are *inner polar balls*; *outer polar balls* are defined analogously. The union of the inner polar balls forms a good approximation of the object bounded by F (we make this quite formal in [3]), and similarly the union of outer polar balls forms a good approximation of the complement of the object; see Figure 5 for an example.

3.3 Power crust

Now we consider the power diagram of the polar balls, which subdivides $\mathbb{R}^3$ into a set of cells.

Definition: The *power crust* is the boundary between the power diagram cells belonging to inner poles and power diagram cells belonging to outer poles.

Since most points of the interior solid bounded by F are inside the union of the inner polar balls, and outside of the union of outer polar balls, they belong to cells of the power diagram corresponding to inner poles. Similarly most points in the exterior solid belong to cells corresponding to outer poles.

A two-dimensional face of the power crust separates cells corresponding to an inner and an outer pole. The two polar balls should intersect shallowly, if at all, since the inner polar ball is mostly inside the object and the outer polar ball is mostly outside (this is formalized in Lemma 3, below). So the power crust face lies near the boundaries of both unions of balls, and hence near the boundary F of the object. A theorem along these lines is given in [3]. The power crust actually interpolates the input samples in S, which lie on the surface of the union of the inner, and of the outer, polar balls.

3.4 Power shape

The definition of the power crust implies a way to connect the poles to form a topologically correct [3] approximation of the medial axis as a simplicial complex M, which we call the *power shape*. The vertices of M are the poles themselves. Inner poles whose cells are adjacent in the power diagram are connected by simplices in M, as are adjacent outer poles. The power shape is a subset of the *weighted Delaunay triangulation* (also known as the *regular triangulation*) dual to the power diagram, just as the Delaunay triangulation is dual to the usual unweighted Voronoi diagram. While the medial axis of F is a two-dimensional surface, the power shape generally contains some very flat, but solid, tetrahedra.

4 Theory

The analysis of the power crust is given in the companion paper [3]. Here we state some key lemmas from the analysis which turn out to be useful in developing an implementation which is robust when the sampling assumptions do not hold.

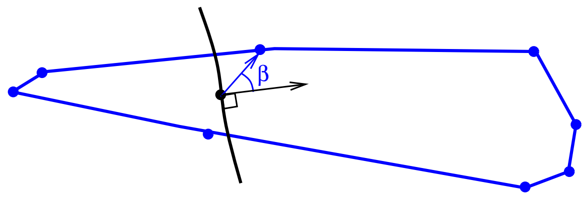

Figure 6. The Voronoi cell of sample s is long and skinny and perpendicular to the surface. Lemma 1 states that if the vector from s to a point x in its Voronoi cell is at a large angle β to the surface normal, x must be close to s.

One key idea driving the algorithm is that when S is sufficiently dense, the Voronoi cell of any sample $s \in S$ is long and skinny and perpendicular to the surface F. The 'skinnyness' is formalized in the following lemma by saying that any point x in the Voronoi cell such that the angle β between vector sx and the surface normal is large must be close to s; see Figure 6. For convenience, we write

$r' = r/(1-r) = O(r)$, where r is the constant describing the distance between samples in the sampling assumption.

Lemma 1. **([3], Corollary 12)** *Let S be an r-sample from a smooth surface F, for small enough r. For any x such that $\beta > \arcsin r'$, we have $d(s,x) \leq \kappa LFS(s)$ with*

$$\kappa = \frac{r'}{\sin(\beta - \arcsin r')}$$

The idea that the Voronoi cell has to be long is formalized simply as follows.

Lemma 2. **([3], page 16)** *The distance to either pole of a sample s is at least $LFS(s)$.*

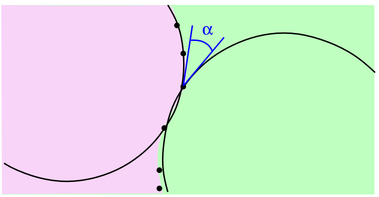

Figure 7. An inner and an outer polar ball can intersect shallowly, if at all, because the dense set of samples separates them. Angle α measures the depth of the intersection.

Another key idea is that when S is an r-sample, for small enough r, any inner polar ball is almost entirely contained in the interior solid, and any outer polar ball is almost entirely exterior. The following lemma formalizes this idea by saying that any inner polar ball and any outer polar ball can only intersect shallowly, as in Figure 7.

Lemma 3. **([3], Lemma 5)** *Let S be an r-sample from a smooth surface F, for small enough r. The angle α at which an inner and an outer polar ball can intersect is $O(r)$.*

5 Algorithm

The basic algorithm is a straightforward reflection of our strategy: first estimate the MAT, and then use it to define the surface approximation.

1. Compute the Voronoi diagram of the sample points S.
2. For each sample point, compute its poles.
3. Compute the power diagram of the poles.
4. Label each pole either inside or outside.
5. Output the power diagram faces separating the cells of inside and outside poles as the power crust.
6. Output the regular triangulation faces connecting inside poles as the power shape.

Given a good program for Voronoi diagram and power diagram, only Steps 2 and 4 require some elaboration.

5.1 Selecting poles

In Step 1, we compute the Voronoi diagram of the sample points S, augmented with the eight vertices Z of the surrounding box. For each sample point $s \in S$, we find the farthest Voronoi vertex, p_1, which is the first pole of s. We then find the farthest Voronoi vertex p_2 such that the vectors $s,\vec{p}_1$ and $s,\vec{p}_2$ have negative dot product. When F is an r-sample for small enough r, p_1 and p_2 are indeed the farthest Voronoi vertices of s on either side of F [1]. The correctness of this procedure depends on Lemma 1, and it breaks down when the assumption of the Lemma fails.

5.2 Labeling algorithm

We first compute the power diagram of all the polar balls, and then label the poles as inner or outer by examining the power diagram. We define a natural graph on the power diagram cells: two cells are connected in the graph if they share a two-dimensional face. In addition, two cells are connected if they belong to the two poles of the same sample s. We traverse this graph, labeling poles `inner` or `outer` as we go. When S is well-sampled the simple algorithm below can be proved correct, using two facts. The first is Lemma 3, that an inner polar ball and an outer polar ball can only intersect shallowly. The second is that one of the two poles of every sample is an inner pole and the other is an outer pole.

The naive traversal algorithm begins by labeling poles adjacent to points forming the bounding box Z as `outer` and then propagating labels as follows. For any pole p labeled `outer`, if it has an unlabeled neighbor q such that the polar balls of p and q intersect deeply (that is, the angle α in Figure 7 is large), we give q label `outer` as well. And for each sample s for which p is a pole (there might be more than one), we give the other pole of s the label `inner`. We propagate the labels of inner poles similarly: deeply intersecting neighbors get labeled `inner`, and the opposite pole of the same sample gets labeled `outer`.

But because the sampling assumption is not met everywhere, a naive implementation of this graph-traversal algorithm could fail dramatically - once an error is made, it propagates. Instead, we choose which labels to propagate using the following greedy heuristic. We keep track of the "belief" that an unlabeled ball is inner or outer, based on the labels already assigned, and we label and propagate the labels of the poles for which we are most confident first.

Specifically, each ball keeps track of two values, *in* and *out*, which lie between 0 ("unknown") and 1 ("certain"). We start by giving all poles far away from the bounding box of the original samples an *out* value of 1 and an *in* value of zero, and initialize all other balls' *in*/*out* values to zero.

We put all the unlabeled poles in a priority queue, with the priority determined by the *in* and *out* values. If only one of the *in* or *out* values is non-zero, we use the non-zero value as the priority. If both *in* and *out* values are non-zero, it means that the pole is "confused"; we would like to label such poles as late in the process as possible, so we give them the priority $|in - out| - 1$, which is between zero and -1.

The algorithm is then to repeatedly remove the top element of the queue and label it *in* or *out*, whichever has the bigger value. We then propagate the newly assigned label to the *in* and *out* values of the remaining unlabeled poles, changing their priority in the queue.

We use the local geometry to weight the effect of a newly labeled pole on its neighbors. For a sample s of which p is the pole, let β denote the angle formed by p,s and the other pole q of s, so that we have $\pi/2 \leq \beta \leq \pi$. According to Lemma 1, the denser the sampling is, as measured by the parameter r, the larger β should be. So the bigger β is, the more "likely" is it that q should get the opposite label from p. We use $0 \leq -\cos(\beta) \leq 1$ as the weight of the connection between p and q.

According to Lemma 3, two balls with different labels should intersect shallowly, as measured by angle α in Figure 7. So the deeper the intersection, the more "likely" q will have the same label as p. We set the weight of the connection between p and q to $0 \leq -\cos(\alpha) \leq 1$.

We summarize the labeling algorithm with the pseudo-code in Figure 8. Note that once a label is assigned, it is never changed. An algorithm which toggles labels to find a locally optimal labeling might be better, but we have not found it necessary.

```
Label_Poles() {
    For all poles p,
        initialize in(p) = out(p) = 0.
        insert p in the queue.
    For each pole p adjacent to points of Z,
        out(p) = 1.
        Update_Priority(p)
    while (queue is not empty) {
        Remove the top element p of the priority queue
        If in(p) > out(p), label(p) = in, tmp(p) = in(p)
        Otherwise, label(p) = out, tmp(p) = out(p)
        For each sample s of which p is the pole,
            let q be the other pole of s,
            opp(label(p))(q) = max(tmp(p) * w_pq, opp(label(p))(q))
    /* opp(in) = out, opp(out) = in, w_pq = -cos(∠psq)*/
            Update_Priority(q)
        For each deeply intersecting neighboring poles q,
            (label(p))(q) = max(tmp(p) * w_pq, (label(p))(q))
    /* w_pq = -cos(α), α is angle between balls p and q*/
            Update_Priority(q)
    }
}

Update_Priority(pole p) {
    If in(p) > 0 and out(p) > 0, pri(p) = |in(p) - out(p)| - 1.
    Otherwise, pri(p) = max(in(p), out(p)).
}
```

Figure 8. The labeling algorithm we implemented. This is a special case of the naive labeling algorithm, which is provably correct when S and F meet the sampling assumptions.

6 Omitting poles

The quality of the power crust depends on how well polar balls approximate the MAT. When Lemma 1 fails, because F is not smooth or because S is too sparse, our procedure for choosing poles breaks down and the MAT approximation suffers. Detecting the failure of Lemma 1 and omitting the poles of badly-shaped Voronoi cells leaves us with an approximation of the portion of the MAT which *can* be estimated from the input sample. We find empirically that computing the power crust from this partial MAT as usual produces good models.

6.1 Noise

One situation in which Lemma 1 fails is when there is significant noise in the data, comparable to or greater than the distance between samples on the surface. As on the left in Figure 9, some Voronoi cells are roughly round, and others extend a long way perpendicularly on one side of the surface but not on the other. This kind of noise is typical when several laser range scans are combined; while the individual scans are pretty clean, alignment errors between the scans produce a scattering of samples near the surface.

Instead of just assuming that Lemma 1 is always satisfied, our implementation tests the Voronoi cell of every sample $s \in S$. Given a user-defined estimate λ of r' we get a lower bound on $LFS(s)$ by evaluating the following 'skinnyness' formula at every Voronoi vertex x:

$$LFS(s) \geq \min_{\text{vertex } x} \frac{d(s,x)\sin(\beta-\lambda)}{\lambda}$$

This is derived from Lemma 1, estimating $\arcsin\lambda$ by λ for efficiency, since λ is small.

According to Lemma 2, for a pole p, $d(s,p) \geq LFS(s)$. If the distance from a pole to s violates this lower bound, the Voronoi cell is misshapen on that side of the surface, and we discard the pole. As on the right in Figure 9, the resulting power crust lies between the remaining inner and outer polar balls. In this case the power crust lies near, but does not necessarily contain, the samples.

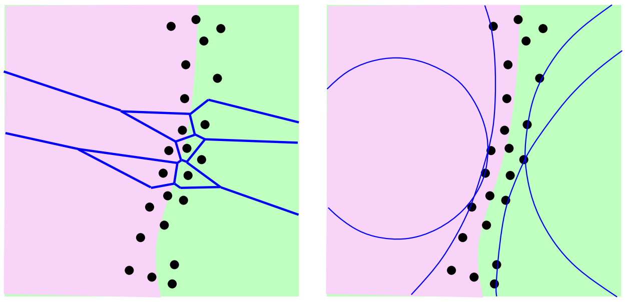

Figure 9. When the input sample lies near, but not on, the surface, not all Voronoi cells are well shaped. We take a pole from the Voronoi cell of sample s only if the cell is well shaped and the pole is far from s. The resulting sets of polar balls roughly approximate the MAT.

While we get watertight models from the power crust in the presence of this characteristic noise in laser range data, as in Figures 1,18, and 19, and even with added Gaussian noise as in Figure 10, clearly we are not producing the *optimal* reconstructions based on all the data.

Figure 10. To demonstrate robustness in the presence of noise, we added Gaussian noise to laser range scanner data. The resulting model is still watertight.

6.2 Sharp corners

Near a sharp corner, the poles on the inside of the corner will fail the above 'skinnyness' test, while those on the outside will pass. But discarding only the poles on the inside of the corner causes the power crust to collapse and round off the corner.

Instead, if the user indicates that the model contains sharp corners, we discard *both* p_1 and p_2 for any sample that fails the 'skinnyness' test. Nearby power crust faces extend into the region which is left uncovered by the discarded poles, extending adjacent smooth surfaces linearly into the empty region, until they meet at a sharp angle. Figure 12 shows a three-dimensional example of this behavior. Notice that a sharp edge can be reconstructed nicely even though there are no sample points on the edge itself.

Sharp cornered models of mechanical parts can be produced from fairly sparse samples, such as the Renault steering knuckle data shown in Figure 13, which was reconstructed from only 6002 points.

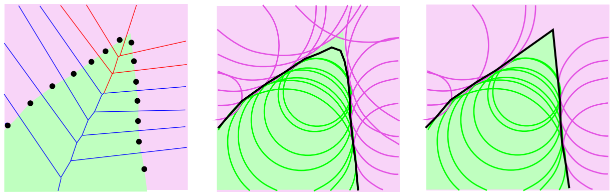

Figure 11. The inner poles of the samples near a sharp corner do not pass the 'skinny-ness' test. But just discarding the inner poles collapses the power crust at the corner. Discarding both poles of the badly-shaped Voronoi cells allows the power crust faces formed in nearby well-sampled regions to extend into the region of uncertainty and meet at a sharp corner.

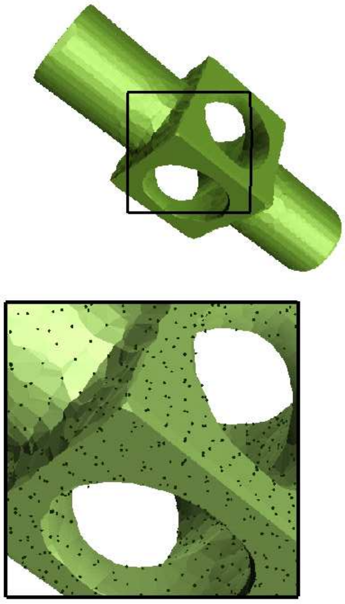

Figure 12. Sharp corners with no nearby features can be reconstructed without any samples on the edge itself. The corner between the cylinder and the top of the cube is not resolved as well because the polar balls on both sides of the surface are not very large compared to the sampling density.

Notice that when we try to resolve sharp corners on a noisy input, both poles of every sample could be discarded.

7 Extensions

We present a few simple extensions of our algorithm which seem to be of practical importance.

7.1 Holes

A 'watertight' surface representation is desirable in some contexts, for instance as input to a layered manufacturing system or for CSG. But we would also like the flexibility to produce surfaces which are not closed manifolds. In general, we would like to be able to fill in small holes in the data, but not cover over large ones: for instance, we might want to leave a hole at the bottom of the hand in Figure 1, while still filling in the hard-to-scan gaps between the fingers. Big holes in the data need not lie on the convex hull or even on any silhouette of the object; for instance the hole inside the shell in Figure 18 where the scanner could not reach the visible surface.

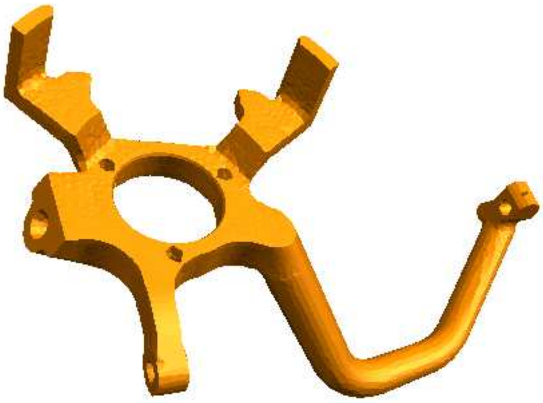

Figure 13. All the holes, and even the small notch at the end of the arm, are correct. This data was an example of something beyond the capabilities of the algorithm of [2]

We characterize holes using the idea behind Lemma 3: on well-sampled regions of the surface, inner and outer polar balls cannot intersect deeply. At a hole, inner polar balls can bulge out of the object, as in Figure 5, and outer polar balls can bulge into the interior. A power crust face formed by a deeply intersecting pair of polar balls, one inner and one outer, fills in the hole in the surface. When the intersecting pair of balls is large (as defined by a user-specified parameter), we can choose to omit the face from the power crust. Examples are the sea shell in Figure 18 and the foot in Figure 5.

7.2 Approximate offset surfaces

An ε-*offset surface* from F is a surface F', formed by the points x such that the distance from x to the nearest point on F is exactly ε. There is an inside and an outside offset surface from F for every ε. In terms of the MAT, the inside offset surface is formed by adding ε to the radius of every ball in the exterior MAT, and subtracting ε from every ball in the interior MAT; the outer offset surface can be defined analogously.

Computing an exact offset surface is difficult, in part because it can differ topologically from F. When F is represented by an approximate MAT, we can construct an approximate inside offset surface of P by increasing the radius of every outer polar ball by ε and decreasing the radius of every inner polar ball by ε, and then computing the power crust as usual. Since the power crust is always the watertight boundary of a solid, the output cannot suffer from cracks or self-intersections. Figure 14 shows an example.

7.3 Medial axis simplification

The power shape accurately reflects the topology of the power crust, and it is geometrically correct at least in the sense that an accurate reconstruction of the surface can be found from the (inner and outer) power shape by the inverse transform. But the power shape, like the medial axis, of a natural three-dimensional object tends to be complicated. This is because small bumps on the surface produce large "spikes" in the medial axis. Another way to say this is that introducing small perturbations in the surface introduces large features in the medial axis. A simplification of the medial axis, preserving stable shape features but removing those corresponding to surface details, can be more useful in some applications such as feature recognition or shape decomposition.

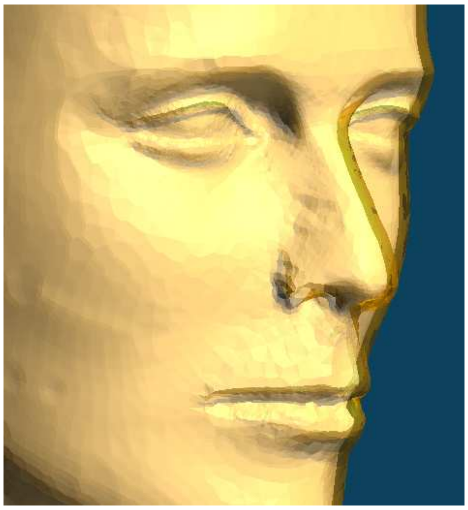

Figure 14. An approximate inner offset surface. The transparent yellow surface is the original. Like the power crust, the offset surface is always the watertight boundary of a solid.

As described in Section 1, several groups of researchers have independently given similar characterizations of the stable parts of the medial axis. The method we adopt is similar to that used in [6], but a little simpler.

Our approach is to assume that the position of each surface point is perturbed from its "true position" by at most distance ε, which represents the noise level. We classify a point p of the medial axis as stable or unstable with respect to ε by examining the medial ball B centered at p. If all the points at which B touches the surface are within distance ε of each other, then it is possible that their "true positions" coincide, causing B to disappear; in this case p is unstable. Certainly p is unstable when the diameter of B is less than ε, but notice that p might also be unstable when B is quite large, as in Figure 15, if the maximum angle γ spanned by points of contact with the surface is small.

An alternative would be to define p as unstable when γ is small, regardless of the diameter of B. This is appealing in that it is a scale invariant function of shape, but it is not very useful for removing the effects of noise; small surface perturbations do indeed introduce balls with small diameter and large values of γ, like the one on the right in Figure 15.

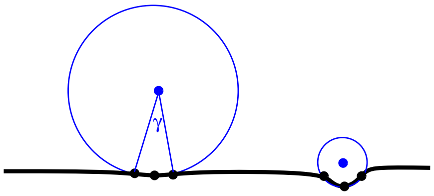

Figure 15. When the samples corresponding to a point of the medial axis are all close together, either the medial ball is small or the angle γ is small. Either way the absolute size of the surface feature represented by the ball is small.

We idea leads to the following simple procedure to eliminate unstable polar balls. For each pole p let S_p be the set of sample points on the surface of its polar ball. Let d_p be the maximum distance between any two samples in S_p. We remove p if $d_p \leq \varepsilon$.

Even after the removal of unstable features, the power shape can still be quite redundant, since S might be arbitrarily dense irrespective of the noise level. We try to reduce this redundancy by removing poles whenever we can guarantee that the accuracy of the union of the remaining polar balls as a representation of the object shape is preserved, again to within a user-specified parameter δ, usually less than ε.

We use a greedy method to eliminate redundant polar balls. We sort the polar balls by radius and examine them in order from largest to the smallest. For each ball $B_{c,\rho}$ we consider all neighboring balls $B_{p,\mu}$ in the power shape such that $\rho > \mu$. We remove $B_{p,\mu}$ if $B_{p,\mu}$ is contained in the slightly larger ball $B_{c,\rho+\delta}$. If $B_{c,\mu}$ is removed, we recursively consider removing its neighbors as well.

After polar balls have been eliminated by simplification and redundancy checking, we recompute the power diagram of the remaining balls, retaining the original `inner` and `outer` labels, and extract the new power shape.

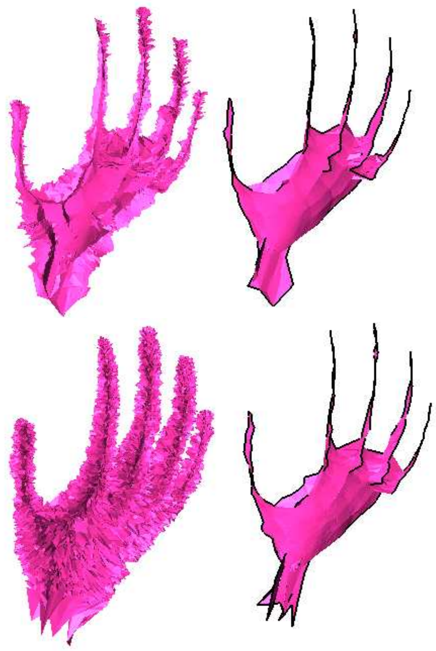

Figure 16. Above, the power shape of the original hand model and its simplification with 352,985 and 7805 faces respectively. Below, the power shape of the hand model with added Gaussian noise and its simplification, with 438,855 and 7003 faces respectively. Notice that the two simplified models are very similar; this is because the simplification procedure removes unstable features which might be due to small surface perturbations.

To demonstrate the stability of the parts of the power shape left after simplification we simplified the power shapes of the hand data and the hand data with added Gaussian noise from Figure 10, using the same parameters ε and δ. Figure 16 shows the results. Notice that the simplified power shape of the noisy model is visually almost identical to that of the clean model; both represent the stable parts of the medial axis, and sketch out the general shape.

8 Implementation

Our implementation uses robust computational geometry software. We computed 3D Delaunay triangulation and weighted Delaunay triangulation using Ken Clarkson's `hull`, an open-source exact-arithmetic convex hull program.

To compute the power diagram using `hull`, we used a well-known transformation from weighted Delaunay triangulation to convex hull (see eg. [22]), which required a slight modification of the code.

r	S is an $r - sample$ for F
user controllable parameters	
λ	user estimate of r'
α	deep intersection angle between polar balls
ε	for removing unstable poles
δ	for removing redundant poles

Figure 17. Table of parameters

The output of `hull` is a set of Delaunay tetrahedra, and we need Voronoi vertices and power diagram vertices as well. Computing these robustly is difficult since, for instance, when a tetrahedron is nearly flat the circumcenter computation used for the Voronoi vertex is inherently unstable. We used functions based on Jonathan Shewchuk's robust adaptive precision determinant subroutines. For the Voronoi vertex computations, the input points were given with fixed precision, and as a result the determinants we computed were always either sufficiently precise or identically zero. To get the same effect for the power diagram vertices, we rounded the locations of the poles to a fixed precision as well. This produced no errors that we noticed.

We gave the program several user-specified parameters (see Figure 17) but found that we never needed to change the default values for most of them. For example, the user can set the angle α at which two balls are considered to intersect deeply; we generally left this at a default value of $\arccos(-0.4)$. The one frequently used parameter is the estimate λ of sample spacing r', which should vary depending on the input data.

The simplification of the power shape is implemented as a separate post-processing step.

Moderately sized inputs, eg. 30,000 samples, required about six minutes on a 400 MHz Sun.

9 Outputs

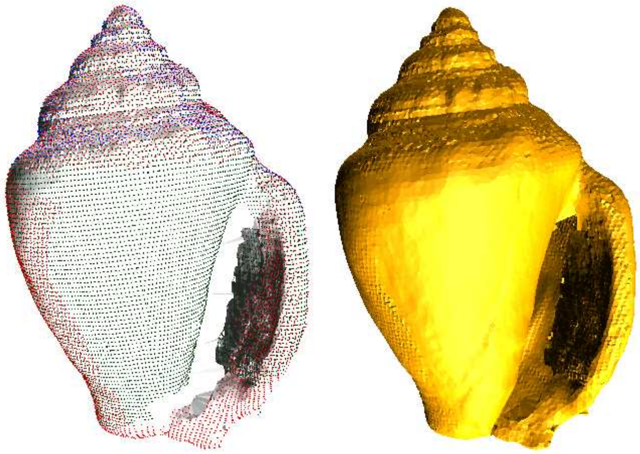

Figure 18. A reconstruction from four different scans, 37,073 samples total. The area in the interior, where there are no samples, is detected using the idea in Section 7.1 and intentionally left as a hole. A silhouette-based hole filling algorithm would have trouble filling this hole properly, but our method does not (it is filled in the solid model under the samples on the left).

We tested the power crust using both well-known models and data we collected using a Cyberware M15 (tabletop) scanner. Almost all of the inputs immediately produced perfect surface reconstructions, requiring no tweaking. Exceptions were the small holes in the sparsely sampled steering knuckle in Figure 13, which required a careful choice of α to optimize the labeling algorithm, and the sharp corners on the noisy rubber stamp in Figure 19, where we had to tune r' to balance the noise adaptation with the sharp corner detection.

While any power crust output is guaranteed to be the boundary of a solid, all our outputs were *regular* solids, that is, their boundaries are piecewise-linear manifolds. Holes in the data seem to be filled in an appropriate and predictable way, for example the hard-to-scan spaces between the fingers, and the end of the wrist, in Figure 1.

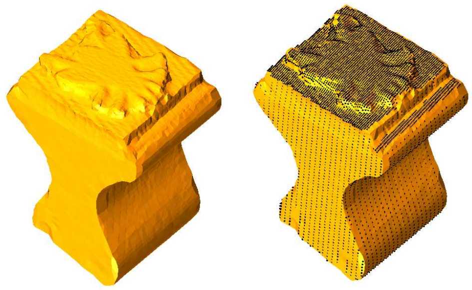

Figure 19. A rubber stamp model, and the imperfect laser range data (9755 points) used to produce it. Six scans were combined, with the top sampled at a higher resolution than the sides and bottom. Note the sharp corners on the sides, and the fill-in where scans fail to overlap.

Power crust faces are not triangles, and although the power crust interpolates the input samples, not all input samples are power crust vertices and not all power crust vertices are input samples. The shapes of the faces seem to conform to the shape of the object more nicely than polygonal models which are constrained to have triangular faces with the samples as vertices.

On the other hand, power crusts have more faces than comparable triangulated surfaces. Using standard decimation techniques - progressive mesh [33] with a quadratic error metric [25] - we can reduce the polygon count to about 1/4 of the original with no visible loss of detail, as in Figure 20.

Figure 20. The original club on top has 112,669 faces. The middle club with 32,000 faces is visually indistinguishable (the word "ESTEEM" on the face of the club is still legible). The club on the bottom, with 5,000 faces, begins to show some degradation.

10 Software

The power crust and medial axis simplification software is available at:
`http://www.cs.utexas.edu/users/amenta/powercrust/welcome.html`.

11 Acknowledgments

We thank Ken Clarkson for his advice on modifying `hull`, and for publishing the source code. We thank Jonathan Shewchuk for his determinant routines and for advance information on his new three-dimensional tetrahedralization package `pyramid`. Without these major computational geometry software projects this work would not be possible. We also thank The Geometry Center for `Geomview`, and Michael Garland and Paul Heckbert for the mesh simplification code. We thank Thomas Wahl for Figure 20, and for his careful reading of this text.

References

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry* **22**, pp. 481–504, (1999).

[2] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Proc. SIGGRAPH '98*, pp. 415–421.

[3] N. Amenta, S. Choi and R. Kolluri. The power crust, unions of balls and the medial axis transform, *Int. J. Computational Geometry and its Applications*, to appear.

[4] N. Amenta, S. Choi, T. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction, *ACM Symposium on Computational Geometry*, 2000, pages 213-222. Also *Int. J. Computational Geometry and its Applications*, to appear.

[5] D. Attali and A. Montanvert. Computing and simplifying 2D and 3D continuous skeletons. *Computer Vision and Image Understanding* **67**:3 (1997) pp. 261–273.

[6] D. Attali and J-O. Lachaud. Delaunay constrained iso-surface, skeleton extraction and noise removal, *Int. J.Computational Geometry and its Applications*, to appear.

[7] J. August, A. Tannenbaum, and S.W. Zucker. On the evolution of the skeleton, *International Conference on Computer Vision* (1999).

[8] F. Bernardini, J. Mittleman, H. Rushmeir, C. Silva and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* **5**:4, pp. 349–359. Also IBM Tech. Report RC21463(96842).

[9] F. Bernardini, H. Rushmeier. Strategies for registering range images from unknown camera positions. *Proc. of SPIE Three-Dimensional Image Capture and Applications III*, 2000.

[10] R. Blanding, C. Brooking, M. Ganter and D. Stori. A skeletal-based solid editor. *Solid Modeling '99*.

[11] J-D. Boissonnat. Geometric structures for three-dimensional shape reconstruction, *ACM Trans. Graphics* 3 (1984) 266–286.

[12] J-D. Boissonnat and F. Cazals. Natural coordinates of points on a surface. *Proceedings of the 16th Annual ACM Symposium on Computational Geometry* pp. 223–232, (2000).

[13] J. W. Brandt. Describing a solid with the three-dimensional skeleton, *Proc. the International Society for Optical Engineering*, vol. 1830, Curves and Surfaces in Computer Vision and Graphics III, SPIE, Boston Mass, 1992.

[14] H.I. Choi, S.W. Choi and H.P. Moon. Mathematical theory of medial axis transform, *Pacific Journal of Mathematics* 181(1), pp. 57–88, (1997).

[15] H. Choset, *Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph*, PhD Thesis, CMU, (1996).

[16] K. Clarkson and P. Shor. Applications of random sampling in computational geometry II, *Discrete and Computational Geometry* **4** (1989), pp. 387–421.

[17] Tim Culver, John Keyser, Dinesh Manocha. Accurate computation of the medial axis of a polyhedron. *Solid Modeling '99*, pp. 179–190.

[18] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Proc. SIGGRAPH '96*, pp. 303–312.

[19] T. Dey, personal communication.

[20] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes, *ACM Trans. Graphics* 13 (1994) 43–72.

[21] H. Edelsbrunner. The union of balls and its dual shape, *Technical report*. A version appeared in *Proceedings of the 9th Annual ACM Symposium on Computational Geometry*, pp. 218–231, (1993).

[22] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer Verlag, (1987). See pages 327–328.

[23] M. Etzion and A. Rappoport. Computing the Voronoi diagram of a 3-D polyhedron by separate computation of its symbolic and geometric parts. *Solid Modeling '99*

[24] M. Etzion and A. Rappoport. Computing Voronoi skeletons of a 3D polyhedron by space subdivision, *Technical Report*, Hebrew University, 1999.

[25] M. Garland, and P. Heckbert. Surface simplification using quadric error metrics, *SIGGRAPH '97* pp. 209–216.

[26] C. Gold. Crust and anti-crust: a one-step boundary and skeleton extraction algorithm, *ACM Symp. on Computational Geometry*, (1999).

[27] M. Gopi and S. Krishnan. A fast and efficient projection based approach for surface reconstruction. *International Journal of High Performance Computer Graphics, Multimedia and Visualization*, to appear.

[28] L. Guibas, R. Holleman and L. E. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach, in *IEEE/RSJ Proc. of the Int. Conf. on Intelligent Robots and Systems* (1999).

[29] L. Guibas, D. Knuth and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams, *Algorithmica* **7** (1992), pp. 381–413.

[30] K. Hoff, T. Culver, J. Keyser, M. Lin and D. Manocha. Fast computation of generalized Voronoi diagrams using graphics hardware, *Proc. SIGGRAPH '99*, pp. 277–285.

[31] C. Hoffman, How to construct the skeleton of CSG objects, in *The Mathematics of Surfaces, IV*A. Bowyer and J. Davenport, Eds., Oxford University Press, 1990.

[32] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface Reconstruction from Unorganized Points. *Proc. SIGGRAPH '92*, pp. 71–78.

[33] H. Hoppe. Progressive meshes, *Proc. SIGGRAPH '96* , pp. 99–108.

[34] P. Hubbard. Approximating polyhedra with spheres for time-critical collision detection, *ACM Transactions on Graphics*, 15(3), pp. 179–210, (1996).

[35] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade and D. Fulk. The Digital Michelangelo project: 3D scanning of large statues. *SIGGRAPH 2000*, pages 131–144.

[36] V.J. Milenkovic. Robust construction of the Voronoi diagram of a polyhedron, *5th Canadian Conference on Computational Geometry* (1993) pp. 473–478.

[37] R.L. Ogniewicz. Skeleton-space: A multiscale shape description combining region and boundary information, *Proceedings of Computer Vision and Pattern Recognition* (1994) pp. 746-751.

[38] S.M. Pizer, C.A. Burbeck, J.M. Coggins, D.S. Fritsch and B.S Morse. Object shape before boundary shape: Scale-space medial axes, *Journal of Mathematics Imaging and Vision*, **4**:3, (1994), pp. 303–313.

[39] G. Stetten and S. Pizer. Automated identification and measurement of objects via populations of medial primitives, *Information Processing in Medical Imaging*, 1999.

[40] V. Ranjan and A. Fournier. Matching and interpolation of shapes using unions of circles, *Computer Graphics Forum*, 15(3), pp. 129–142, (1996).

[41] D. Sheehy, C. Armstrong and D. Robinson. Shape description by medial axis construction, *IEEE Transactions on Visualization and Computer Graphics*, 2(1), pp. 62–72, (1996).

[42] A. Sheffer, M. Etzion, A. Rappoport and M. Bercovier. Hexahedral mesh generation using the embedded Voronoi graph, *Technical Report*, Hebrew University (1999).

[43] E.C. Sherbrooke, N.M. Patrikalakis and E. Brisson. An algorithm for the medial axis transform of 3D polyhedral solids, *IEEE Transactions on Visualization and Computer Graphics*, **2**:1 (1996) pp. 44-61.

[44] J.R. Shewchuk, personal communication.

[45] D. Storti, G. Turkiyyah, M. Ganter, C. Lim and D. Stal. Skeleton-based modeling operations on solids. *Solid Modeling '97*, pp. 141–154.

[46] M. Teichman and S. Teller. Assisted articulation of closed polygonal models, *Proc. 9th Eurographics Workshop on Animation and Simulation*, (1998).

[47] G.M. Turkiyyah, D.W.Storti, M. Ganter, H. Chen and M. Vimawala. An accelerated triangulation method for computing the skeletons of free-form solid models, *Computer Aided Design*, **29**:1 (1997) pp. 5–19.

[48] S.P. Uselton. Three dimensional surface descriptions from sensed data, *Proc. of the 16th Hawaii International Conference on System Sciences*, vol. 2, (1983) pp. 387–385.

[49] P. Vermeer. *Medial axis transform to boundary representation conversion*, PhD. Thesis, Purdue University, (1994).

[50] J. Vleugels and M. Overmars. Approximating generalized Voronoi diagrams in any dimension, *Int. Jour. Computational Geometry and its Applications*, (1998).

[51] S. A. Wilmarth, N. M. Amato and P. F. Stiller. Motion planning for a rigid body using random networks on the medial axis of the free space, *ACM Symp. on Computational Geometry* (1999).