

# Surface Reconstruction from Unorganized Points

Hugues Hoppe\*   Tony DeRose\*   Tom Duchamp†  
John McDonald‡   Werner Stuetzle‡

University of Washington  
Seattle, WA 98195

## Abstract

We describe and demonstrate an algorithm that takes as input an unorganized set of points  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^3$  on or near an unknown manifold  $M$ , and produces as output a simplicial surface that approximates  $M$ . Neither the topology, the presence of boundaries, nor the geometry of  $M$  are assumed to be known in advance — all are inferred automatically from the data. This problem naturally arises in a variety of practical situations such as range scanning an object from multiple view points, recovery of biological shapes from two-dimensional slices, and interactive surface sketching.

**CR Categories and Subject Descriptors:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling.

**Additional Keywords:** Geometric Modeling, Surface Fitting, Three-Dimensional Shape Recovery, Range Data Analysis.

## 1 Introduction

Broadly speaking, the class of problems we are interested in can be stated as follows: Given partial information of an unknown surface, construct, to the extent possible, a compact representation of the surface. Reconstruction problems of this sort occur in diverse scientific and engineering application domains, including:

- *Surfaces from range data:* The data produced by laser range scanning systems is typically a rectangular grid of distances from the sensor to the object being scanned. If the sensor and object are fixed, only objects that are “point viewable” can be fully digitized. More sophisticated systems, such as those produced by Cyberware Laboratory, Inc., are capable of digitizing cylindrical objects by rotating either the sensor or the object. However, the scanning of topologically more

complex objects, including those as simple as a coffee cup with a handle (a surface of genus 1), or the object depicted in Figure 1a (a surface of genus 3), cannot be accomplished by either of these methods. To adequately scan these objects, multiple view points must be used. Merging the data generated from multiple view points to reconstruct a polyhedral surface representation is a non-trivial task [11].

- *Surfaces from contours:* In many medical studies it is common to slice biological specimens into thin layers with a microtome. The outlines of the structures of interest are then digitized to create a stack of contours. The problem is to reconstruct the three-dimensional structures from the stacks of two-dimensional contours. Although this problem has received a good deal of attention, there remain severe limitations with current methods. Perhaps foremost among these is the difficulty of automatically dealing with branching structures [3, 12].
- *Interactive surface sketching:* A number of researchers, including Schneider [21] and Eisenman [6], have investigated the creation of curves in  $\mathbb{R}^2$  by tracing the path of a stylus or mouse as the user sketches the desired shape. Sachs et al. [19] describe a system, called 3-Draw, that permits the creation of free-form curves in  $\mathbb{R}^3$  by recording the motion of a stylus fitted with a Polhemus sensor. This can be extended to the design of free-form surfaces by ignoring the order in which positions are recorded, allowing the user to move the stylus arbitrarily back and forth over the surface. The problem is then to construct a surface representation faithful to the unordered collection of points.

Reconstruction algorithms addressing these problems have typically been crafted on a case by case basis to exploit partial structure in the data. For instance, algorithms solving the surface from contours problem make heavy use of the fact that data are organized into contours (i.e., closed polygons), and that the contours lie in parallel planes. Similarly, specialized algorithms to reconstruct surfaces from multiple view point range data might exploit the adjacency relationship of the data points within each view.

In contrast, our approach is to pose a unifying general problem that does not assume any structure on the data points. This approach has both theoretical and practical merit. On the theoretical side, abstracting to a general problem often sheds light on the truly critical aspects of the problem. On the practical side, a single algorithm that solves the general problem can be used to solve any specific problem instance.

---

\*Department of Computer Science and Engineering, FR-35

†Department of Mathematics, GN-50

‡Department of Statistics, GN-22

This work was supported in part by Bellcore, the Xerox Corporation, IBM, Hewlett-Packard, the Digital Equipment Corporation, the Department of Energy under grant DE-FG06-85-ER25006, the National Library of Medicine under grant NIH LM-04174, and the National Science Foundation under grants CCR-8957323 and DMS-9103002.

## 1.1 Terminology

By a *surface* we mean a “compact, connected, orientable two-dimensional manifold, possibly with boundary, embedded in  $\mathbb{R}^3$ ” (cf. O’Neill [17]). A surface without boundary will be called a *closed surface*. If we want to emphasize that a surface possesses a non-empty boundary, we will call it a *bordered surface*. A piecewise linear surface with triangular faces will be referred to as a *simplicial surface*. We use  $\|\mathbf{x}\|$  to denote the Euclidean length of a vector  $\mathbf{x}$ , and we use  $d(X, Y)$  to denote the Hausdorff distance between the sets of points  $X$  and  $Y$  (the Hausdorff distance is simply the distance between the two closest points of  $X$  and  $Y$ ).

Let  $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  be sampled data points on or near an unknown surface  $M$  (see Figure 1b). To capture the error in most sampling processes, we assume that each of the points  $\mathbf{x}_i \in X$  is of the form  $\mathbf{x}_i = \mathbf{y}_i + \mathbf{e}_i$ , where  $\mathbf{y}_i \in M$  is a point on the unknown surface and  $\mathbf{e}_i \in \mathbb{R}^3$  is an error vector. We call such a sample  $X$   $\delta$ -noisy if  $\|\mathbf{e}_i\| \leq \delta$  for all  $i$ . A value for  $\delta$  can be estimated in most applications (e.g., the accuracy of the laser scanner). Features of  $M$  that are small compared to  $\delta$  will obviously not be recoverable.

It is also impossible to recover features of  $M$  in regions where insufficient sampling has occurred. In particular, if  $M$  is a bordered surface, such as a sphere with a disc removed, it is impossible to distinguish holes in the sample from holes in the surface. To capture the intuitive notion of sampling density we need to make another definition: Let  $Y = \{\mathbf{y}_1, \dots, \mathbf{y}_n\} \subset M$  be a (noiseless) sample of a surface  $M$ . The sample  $Y$  is said to be  $\rho$ -dense if any sphere with radius  $\rho$  and center in  $M$  contains at least one sample point in  $Y$ . A  $\delta$ -noisy sample  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^3$  of a surface  $M$  is said to be  $\rho$ -dense if there exists a noiseless  $\rho$ -dense sample  $\{\mathbf{y}_1, \dots, \mathbf{y}_n\} \subset M$  such that  $\mathbf{x}_i = \mathbf{y}_i + \mathbf{e}_i$ ,  $\|\mathbf{e}_i\| \leq \delta$ ,  $i = 1, \dots, n$ .

## 1.2 Problem Statement

The goal of *surface reconstruction* is to determine a surface  $\hat{M}$  (see Figure 2f) that approximates an unknown surface  $M$  (Figure 1a), using a sample  $X$  (Figure 1b) and information about the sampling process, for example, bounds on the noise magnitude  $\delta$  and the sampling density  $\rho$ .

We are currently working to develop conditions on the original surface  $M$  and the sample  $X$  that are sufficient to allow  $M$  to be reliably reconstructed. As that work is still preliminary, we are unable to give guarantees for the algorithm presented here. However, the algorithm has worked well in practice where the results can be compared to the original surface (see Section 4).

# 2 Related Work

## 2.1 Surface Reconstruction

Surface reconstruction methods can be classified according to the way in which they represent the reconstructed surface.

Implicit reconstruction methods attempt to find a smooth function  $f : \mathbb{R}^3 \rightarrow \mathbb{R}$  such that  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  is close to the zero set  $Z(f)$ . They differ with respect to the form of  $f$  and the measure of closeness. Pratt [18] and Taubin [25] minimize the sum of squared Hausdorff distances from the data points to the zero set of a polynomial in three variables. Muraki [15] takes  $f$  to be a linear combination of three-dimensional Gaussian kernels with different means and spreads. His goodness-of-fit function measures how close the values of  $f$  at the data points are to zero, and how well the unit normals to the zero set of  $f$  match the normals estimated from the data. Moore and Warren [13] fit a piecewise polynomial recursively and then enforce continuity using a technique they call *free form blending*.

In contrast to implicit reconstruction techniques, parametric reconstruction techniques represent the reconstructed surface as a topological embedding  $f(\Lambda)$  of a 2-dimensional parameter domain  $\Lambda$  into  $\mathbb{R}^3$ . Previous work has concentrated on domain spaces with simple topology, i.e. the plane and the sphere. Hastie and Stuetzle [9] and Vemuri [26, 27] discuss reconstruction of surfaces by a topological embedding  $f(\Lambda)$  of a planar region  $\Lambda$  into  $\mathbb{R}^3$ . Schudy and Ballard [22, 23] and Brinkley [4] consider the reconstruction of surfaces that are slightly deformed spheres, and thus choose  $\Lambda$  to be a sphere. Sclaroff and Pentland [24] describe a hybrid implicit/parametric method for fitting a deformed sphere to a set of points using deformations of a superquadric.

Compared to the techniques mentioned above, our method has several advantages:

- It requires only an unorganized collection of points on or near the surface. No additional information is needed (such as normal information used by Muraki’s method).
- Unlike the parametric methods mentioned above, it can reconstruct surfaces of arbitrary topology.
- Unlike previously suggested implicit methods, it deals with boundaries in a natural way, and it does not generate spurious surface components not supported by the data.

## 2.2 Surface Reconstruction vs Function Reconstruction

Terms like “surface fitting” appear in reference to two distinct classes of problems: surface reconstruction and function reconstruction. The goal of surface reconstruction was stated earlier. The goal of function reconstruction may be stated as follows: Given a surface  $M$ , a set  $\{\mathbf{x}_i \in M\}$ , and a set  $\{\mathbf{y}_i \in \mathbb{R}\}$ , determine a function  $f : M \rightarrow \mathbb{R}$ , such that  $f(\mathbf{x}_i) \approx \mathbf{y}_i$ .

The domain surface  $M$  is most commonly a plane embedded in  $\mathbb{R}^3$ , in which case the problem is a standard one considered in approximation theory. The case where  $M$  is a sphere has also been extensively treated (cf. [7]). Some recent work under the title *surfaces on surfaces* addresses the case when  $M$  is a general curved surface such as the skin of an airplane [16].

Function reconstruction methods can be used for surface reconstruction in simple, special cases, where the surface to be reconstructed is, roughly speaking, the graph of a function over a *known* surface  $M$ . It is important to recognize just how limited these special cases are — for example, not every surface homeomorphic to a sphere is the graph of a function over the sphere. The point we want to make is that function reconstruction must not be misconstrued to solve the general surface reconstruction problem.

# 3 A Description of the Algorithm

## 3.1 Overview

Our surface reconstruction algorithm consists of two stages. In the first stage we define a function  $f : D \rightarrow \mathbb{R}$ , where  $D \subset \mathbb{R}^3$  is a region near the data, such that  $f$  estimates the signed geometric distance to the unknown surface  $M$ . The zero set  $Z(f)$  is our estimate for  $M$ . In the second stage we use a contouring algorithm to approximate  $Z(f)$  by a simplicial surface.

Although the *unsigned* distance function  $|f|$  would be easier to estimate, zero is not a regular value of  $|f|$ . Zero is, however, a regular value of  $f$ , and the implicit function theorem thus guarantees that our approximation  $Z(f)$  is a manifold.

The key ingredient to defining the signed distance function is to associate an oriented plane with each of the data points. These

*tangent planes* serve as local linear approximations to the surface. Although the construction of the tangent planes is relatively simple, the selection of their orientations so as to define a globally consistent orientation for the surface is one of the major obstacles facing the algorithm. As indicated in Figure 2b, the tangent planes do not directly define the surface, since their union may have a complicated non-manifold structure. Rather, we use the tangent planes to define the signed distance function to the surface. An example of the simplicial surface obtained by contouring the zero set of the signed distance function is shown in Figure 2e. The next several sections develop in more detail the successive steps of the algorithm.

### 3.2 Tangent Plane Estimation

The first step toward defining a signed distance function is to compute an oriented tangent plane for each data point. The tangent plane  $Tp(\mathbf{x}_i)$  associated with the data point  $\mathbf{x}_i$  is represented as a point  $\mathbf{o}_i$ , called the center, together with a unit normal vector  $\hat{\mathbf{n}}_i$ . The signed distance of an arbitrary point  $\mathbf{p} \in \mathbb{R}^3$  to  $Tp(\mathbf{x}_i)$  is defined to be  $\text{dist}_i(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_i) \cdot \hat{\mathbf{n}}_i$ . The center and normal for  $Tp(\mathbf{x}_i)$  are determined by gathering together the  $k$  points of  $X$  nearest to  $\mathbf{x}_i$ ; this set is denoted by  $Nbhd(\mathbf{x}_i)$  and is called the  $k$ -neighborhood of  $\mathbf{x}_i$ . (We currently assume  $k$  to be a user-specified parameter, although in Section 5 we propose a method for determining  $k$  automatically.) The center and unit normal are computed so that the plane  $\{\text{dist}_i(\mathbf{p}) = 0\}$  is the least squares best fitting plane to  $Nbhd(\mathbf{x}_i)$ . That is, the center  $\mathbf{o}_i$  is taken to be the centroid of  $Nbhd(\mathbf{x}_i)$ , and the normal  $\hat{\mathbf{n}}_i$  is determined using principal component analysis. To compute  $\hat{\mathbf{n}}_i$ , the covariance matrix of  $Nbhd(\mathbf{x}_i)$  is formed. This is the symmetric  $3 \times 3$  positive semi-definite matrix

$$CV = \sum_{\mathbf{y} \in Nbhd(\mathbf{x}_i)} (\mathbf{y} - \mathbf{o}_i) \otimes (\mathbf{y} - \mathbf{o}_i)$$

where  $\otimes$  denotes the outer product vector operator<sup>1</sup>. If  $\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3$  denote the eigenvalues of  $CV$  associated with unit eigenvectors  $\hat{\mathbf{v}}_i^1, \hat{\mathbf{v}}_i^2, \hat{\mathbf{v}}_i^3$ , respectively, we choose  $\hat{\mathbf{n}}_i$  to be either  $\hat{\mathbf{v}}_i^3$  or  $-\hat{\mathbf{v}}_i^3$ . The selection determines the orientation of the tangent plane, and it must be done so that nearby planes are “consistently oriented”.

### 3.3 Consistent Tangent Plane Orientation

Suppose two data points  $\mathbf{x}_i, \mathbf{x}_j \in X$  are geometrically close. Ideally, when the data is dense and the surface is smooth, the corresponding tangent planes  $Tp(\mathbf{x}_i) = (\mathbf{o}_i, \hat{\mathbf{n}}_i)$  and  $Tp(\mathbf{x}_j) = (\mathbf{o}_j, \hat{\mathbf{n}}_j)$  are nearly parallel, i.e.  $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j \approx \pm 1$ . If the planes are consistently oriented, then  $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j \approx +1$ ; otherwise, either  $\hat{\mathbf{n}}_i$  or  $\hat{\mathbf{n}}_j$  should be flipped. The difficulty in finding a consistent global orientation is that this condition should hold between all pairs of “sufficiently close” data points.

We can model the problem as graph optimization. The graph contains one node  $N_i$  per tangent plane  $Tp(\mathbf{x}_i)$ , with an edge  $(i, j)$  between  $N_i$  and  $N_j$  if the tangent plane centers  $\mathbf{o}_i$  and  $\mathbf{o}_j$  are sufficiently close (we will be more precise about what we mean by sufficiently close shortly). The cost on edge  $(i, j)$  encodes the degree to which  $N_i$  and  $N_j$  are consistently oriented and is taken to be  $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j$ . The problem is then to select orientations for the tangent planes so as to maximize the total cost of the graph. Unfortunately, this problem can be shown to be NP-hard via a reduction to MAX-CUT [8]. To efficiently solve the orientation problem we must therefore resort to an approximation algorithm.

Before describing the approximation algorithm we use, we must decide when a pair of nodes are to be connected in the graph. Since

<sup>1</sup>If  $\mathbf{a}$  and  $\mathbf{b}$  have components  $a_i$  and  $b_j$  respectively, then the matrix  $\mathbf{a} \otimes \mathbf{b}$  has  $a_i b_j$  as its  $ij$ -th entry.

the surface is assumed to consist of a single connected component, the graph should be connected. A simple connected graph for a set of points that tends to connect neighbors is the Euclidean Minimum Spanning Tree (EMST). However, the EMST over the tangent plane centers  $\{\mathbf{o}_1, \dots, \mathbf{o}_n\}$  (Figure 1c) is not sufficiently dense in edges to serve our purposes. We therefore enrich it by adding a number of edges to it. Specifically, we add the edge  $(i, j)$  if either  $\mathbf{o}_i$  is in the  $k$ -neighborhood of  $\mathbf{o}_j$ , or  $\mathbf{o}_j$  is in the  $k$ -neighborhood of  $\mathbf{o}_i$  (where  $k$ -neighborhood is defined over  $\{\mathbf{o}_1, \dots, \mathbf{o}_n\}$  as it was for  $X$ ). The resulting graph (Figure 1d), called the *Riemannian Graph*, is thus constructed to be a connected graph that encodes geometric proximity of the tangent plane centers.

A relatively simple-minded algorithm to orient the planes would be to arbitrarily choose an orientation for some plane, then “propagate” the orientation to neighboring planes in the Riemannian Graph. In practice, we found that the order in which the orientation is propagated is important. Figure 3b shows what may result when propagating orientation solely on the basis of geometric proximity; a correct reconstruction is shown in Figure 3c. Intuitively, we would like to choose an order of propagation that favors propagation from  $Tp(\mathbf{x}_i)$  to  $Tp(\mathbf{x}_j)$  if the unoriented planes are nearly parallel. This can be accomplished by assigning to each edge  $(i, j)$  in the Riemannian Graph the cost  $1 - |\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j|$ . In addition to being non-negative, this assignment has the property that a cost is small if the unoriented tangent planes are nearly parallel. A favorable propagation order can therefore be achieved by traversing the *minimal spanning tree* (MST) of the resulting graph. This order is advantageous because it tends to propagate orientation along directions of low curvature in the data, thereby largely avoiding ambiguous situations encountered when trying to propagate orientation across sharp edges (as at the tip of the cat’s ears in Figure 3b). In the MST shown in Figure 2a, the edges are colored according to their cost, with the brightly colored edges corresponding to regions of high variation (where  $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j$  is somewhat less than 1).

To assign orientation to an initial plane, the unit normal of the plane whose center has the largest  $z$  coordinate is forced to point toward the  $+z$  axis. Then, rooting the tree at this initial node, we traverse the tree in depth-first order, assigning each plane an orientation that is consistent with that of its parent. That is, if during traversal, the current plane  $Tp(\mathbf{x}_i)$  has been assigned the orientation  $\hat{\mathbf{n}}_i$  and  $Tp(\mathbf{x}_j)$  is the next plane to be visited, then  $\hat{\mathbf{n}}_j$  is replaced with  $-\hat{\mathbf{n}}_j$  if  $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j < 0$ .

This orientation algorithm has been used in all our examples and has produced correct orientations in all the cases we have run. The resulting oriented tangent planes are represented as shaded rectangles in Figure 2b.

### 3.4 Signed Distance Function

The signed distance  $f(\mathbf{p})$  from an arbitrary point  $\mathbf{p} \in \mathbb{R}^3$  to a known surface  $M$  is the distance between  $\mathbf{p}$  and the closest point  $\mathbf{z} \in M$ , multiplied by  $\pm 1$ , depending on which side of the surface  $\mathbf{p}$  lies. In reality  $M$  is not known, but we can mimic this procedure using the oriented tangent planes as follows. First, we find the tangent plane  $Tp(\mathbf{x}_i)$  whose center  $\mathbf{o}_i$  is closest to  $\mathbf{p}$ . This tangent plane is a local linear approximation to  $M$ , so we take the signed distance  $f(\mathbf{p})$  to  $M$  to be the signed distance between  $\mathbf{p}$  and its projection  $\mathbf{z}$  onto  $Tp(\mathbf{x}_i)$ ; that is,

$$f(\mathbf{p}) = \text{dist}_i(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_i) \cdot \hat{\mathbf{n}}_i.$$

If  $M$  is known not to have boundaries, this simple rule works well. However, the rule must be extended to accommodate surfaces that might have boundaries. Recall that the set  $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  is assumed to be a  $\rho$ -dense,  $\delta$ -noisy sample of  $M$ . If there was no noise, we could deduce that a point  $\mathbf{z}$  with  $d(\mathbf{z}, X) > \rho$  cannot be

a point of  $M$  since that would violate  $X$  being  $\rho$ -dense. Intuitively, the sample points do not leave holes of radius larger than  $\rho$ . If the sample is  $\delta$ -noisy, the radius of the holes may increase, but by no more than  $\delta$ . We therefore conclude that a point  $\mathbf{z}$  cannot be a point of  $M$  if  $d(\mathbf{z}, X) > \rho + \delta$ . If the projection  $\mathbf{z}$  of  $\mathbf{p}$  onto the closest tangent plane has  $d(\mathbf{z}, X) > \rho + \delta$ , we take  $f(\mathbf{p})$  to be undefined. Undefined values are used by the contouring algorithm of Section 3.5 to identify boundaries.

Stated procedurally, our signed distance function is defined as:

```

 $i \leftarrow$  index of tangent plane whose center is closest to  $\mathbf{p}$ 

{ Compute  $\mathbf{z}$  as the projection of  $\mathbf{p}$  onto  $Tp(\mathbf{x}_i)$  }
 $\mathbf{z} \leftarrow \mathbf{o}_i - ((\mathbf{p} - \mathbf{o}_i) \cdot \hat{\mathbf{n}}_i) \hat{\mathbf{n}}_i$ 

if  $d(\mathbf{z}, X) < \rho + \delta$  then
     $f(\mathbf{p}) \leftarrow (\mathbf{p} - \mathbf{o}_i) \cdot \hat{\mathbf{n}}_i$      $\{ = \pm \|\mathbf{p} - \mathbf{z}\| \}$ 
else
     $f(\mathbf{p}) \leftarrow$  undefined
endif

```

The simple approach outlined above creates a zero set  $Z(f)$  that is piecewise linear but contains discontinuities. The discontinuities result from the implicit partitioning of space into regions within which a single tangent plane is used to define the signed distance function. (These regions are in fact the Voronoi regions associated with the centers  $\mathbf{o}_i$ .) Fortunately, the discontinuities do not adversely affect our algorithm. The contouring algorithm discussed in the next section will discretely sample the function  $f$  over a portion of a 3-dimensional grid near the data and reconstruct a *continuous* piecewise linear approximation to  $Z(f)$ .

### 3.5 Contour Tracing

Contour tracing, the extraction of an isosurface from a scalar function, is a well-studied problem [1, 5, 28]. We chose to implement a variation of the marching cubes algorithm (cf. [1]) that samples the function at the vertices of a cubical lattice and finds the contour intersections within tetrahedral decompositions of the cubical cells.

To accurately estimate boundaries, the cube size should be set so that edges are of length less than  $\rho + \delta$ . In practice we have often found it convenient to set the cube size somewhat larger than this value, simply to increase the speed of execution and to reduce the number of triangular facets generated.

The algorithm only visits cubes that intersect the zero set by pushing onto a queue only the appropriate neighboring cubes (Figure 2c). In this way, the signed distance function  $f$  is evaluated only at points close to the data. Figure 2d illustrates the signed distance function by showing line segments between the query points  $\mathbf{p}$  (at the cube vertices) and their associated projected points  $\mathbf{z}$ . As suggested in Section 3.4, no intersection is reported within a cube if the signed distance function is undefined at any vertex of the cube, thereby giving rise to boundaries in the simplicial surface.

The resulting simplicial surface can contain triangles with arbitrarily poor aspect ratio (Figure 2e). We alleviate this problem using a post-processing procedure that collapses edges in the surface using an aspect ratio criterion.<sup>2</sup> The final result is shown in Figure 2f. Alternatively, other contouring methods exist that can guarantee bounds on the triangle aspect ratio [14].

<sup>2</sup>The edges are kept in a priority queue; the criterion to minimize is the product of the edge length times the minimum inscribed radius of its two adjacent faces. Tests are also performed to ensure that edge collapses preserve the topological type of the surface.

## 4 Results

We have experimented with the reconstruction method on data sets obtained from several different sources. In all cases, any structure (including ordering) that might have been present in the point sets was discarded.

**Mesheres** : Points were randomly sampled from a number of existing simplicial surfaces<sup>3</sup>. For instance, the mesh of Figure 3a was randomly sampled to yield 1000 unorganized points, and these in turn were used to reconstruct the surface in Figure 3c. This particular case illustrates the behavior of the method on a bordered surface (the cat has no base and is thus homeomorphic to a disc). The reconstructed knot (original mesh from Rob Scharein) of Figure 3d is an example of a surface with simple topology yet complex geometrical embedding.

**Ray Traced Points** : To simulate laser range imaging from multiple view points, CSG models were ray traced from multiple eye points. The ray tracer recorded the point of first intersection along each ray. Eight eye points (the vertices of a large cube centered at the object) were used to generate the point set of Figure 1b from the CSG object shown in Figure 1a. This is the point set used in Section 3 to illustrate the steps of the algorithm (Figures 1a-2f).

**Range Images** : The bust of Spock (Figure 3e) was reconstructed from points taken from an actual cylindrical range image (generated by Cyberware Laboratory, Inc.). Only 25% of the original points were used.

**Contours** : Points from 39 planar (horizontal) slices of the CT scan of a femur were combined together to obtain the surface of Figure 3f.

The algorithm's parameters are shown in the next table for each of the examples. The execution times were obtained on a 20 MIPS workstation. The parameter  $\rho + \delta$  and the marching cube cell size are both expressed as a fraction of the object's size. The parameter  $\rho + \delta$  is set to infinity for those surfaces that are known to be closed.

Object	$n$	$k$	$\rho + \delta$	cell size	time (seconds)
cat	1000	15	.06	1/30	19
knot	10000	20	$\infty$	1/50	137
mechpart	4102	12	$\infty$	1/40	54
spock	21760	8	.08	1/80	514
femur	18224	40	.06	1/50	2135

## 5 Discussion

### 5.1 Tangent Plane Approximation

The neighborhood  $Nbhd(\mathbf{x}_i)$  of a data point  $\mathbf{x}_i$  is defined to consist of its  $k$  nearest neighbors, where  $k$  is currently assumed to be an input parameter. In the case where the data contains little or no noise,  $k$  is not a critical parameter since the output has been empirically observed to be stable over a wide range of settings. However, it would be best if  $k$  could be selected automatically. Furthermore, allowing  $k$  to adapt locally would make less stringent the requirement that the data be uniformly distributed over the surface. To select and adapt  $k$ , the algorithm could incrementally gather points while monitoring the changing eigenvalues of the covariance matrix (see Section 3.2). For small values of  $k$ , data noise tends to dominate, the eigenvalues are similar, and the eigenvectors do not reveal the surface's true tangent plane. At the other extreme, as  $k$  becomes

<sup>3</sup>Discrete inverse transform sampling [10, page 469] on triangle area was used to select face indices from the mesh, and uniform sampling was used within the faces.

large, the  $k$ -neighborhoods become less localized and the surface curvature tends to increase the “thickness”  $\lambda_i^3$  of the neighborhood. Another possible criterion is to compare  $\lambda_i^3$  to some local or global estimate of data noise. Although we have done some initial experimentation in this direction, we have not yet fully examined these options.

If the data is obtained from range images, there exists some knowledge of surface orientation at each data point. Indeed, each data point is known to be visible from a particular viewing direction, so that, unless the surface incident angle is large, the point’s tangent plane orientation can be inferred from that viewing direction. Our method could exploit this additional information in the tangent plane orientation step (Section 3.3) by augmenting the Riemannian Graph with an additional pseudo-node and  $n$  additional edges.

## 5.2 Algorithm Complexity

A spatial partitioning Abstract Data Type greatly improves performance of many of the subproblems discussed previously. The critical subproblems are (with their standard time complexity):

- EMST graph ( $O(n^2)$ )
- $k$ -nearest neighbors to a given point ( $O(n + k \log n)$ )
- nearest tangent plane origin to a given point ( $O(n)$ )

Hierarchical spatial partitioning schemes such as octrees [20] and  $k$ -D trees [2] can be used to solve these problems more efficiently. However, the uniform sampling density assumed in our data allows simple spatial cubic partitioning to work efficiently. The axis-aligned bounding box of the points is partitioned by a cubical grid. Points are entered into sets corresponding to the cube to which they belong, and these sets are accessed through a hash table indexed by the cube indices. It is difficult to analyze the resulting improvements analytically, but, empirically, the time complexity of the above problems is effectively reduced by a factor of  $n$ , except for the  $k$ -nearest neighbors problem which becomes  $O(k)$ .

As a result of the spatial partitioning, the Riemannian Graph can be constructed in  $O(nk)$  time. Because the Riemannian Graph has  $O(n)$  edges (at most  $n+nk$ ), the MST computation used in finding the best path on which to propagate orientation requires only  $O(n \log n)$  time. Traversal of the MST is of course  $O(n)$ .

The time complexity of the contouring algorithm depends only on the number of cubes visited, since the evaluation of the signed distance function  $f$  at a point  $\mathbf{p}$  can be done in constant time (the closest tangent plane origin  $\mathbf{o}_i$  to  $\mathbf{p}$  and the closest data point  $\mathbf{x}_j$  to the projected point  $\mathbf{z}$  can both be found in constant time with spatial partitioning).

## 6 Conclusions and Future Work

We have developed an algorithm to reconstruct a surface in three-dimensional space with or without boundary from a set of unorganized points scattered on or near the surface. The algorithm, based on the idea of determining the zero set of an estimated signed distance function, was demonstrated on data gathered from a variety of sources. It is capable of automatically inferring the topological type of the surface, including the presence of boundary curves.

The algorithm can, in principle, be extended to reconstruct manifolds of co-dimension one in spaces of arbitrary dimension; that is, to reconstruct  $d - 1$  dimensional manifolds in  $d$  dimensional space. Thus, essentially the same algorithm can be used to reconstruct curves in the plane or volumes in four-dimensional space.

The output of our reconstruction method produced the correct topology in all the examples. We are trying to develop formal guarantees on the correctness of the reconstruction, given constraints

on the sample and the original surface. To further improve the geometric accuracy of the fit, and to reduce the space required to store the reconstruction, we envision using the output of our algorithm as the starting point for a subsequent spline surface fitting procedure. We are currently investigating such a method based on a nonlinear least squares approach using triangular Bézier surfaces.

## References

- [1] E. L. Allgower and P. H. Schmidt. An algorithm for piecewise linear approximation of an implicitly defined manifold. *SIAM Journal of Numerical Analysis*, 22:322–346, April 1985.
- [2] J. L. Bentley. Multidimensional divide and conquer. *Comm. ACM*, 23(4):214–229, 1980.
- [3] Y. Breseler, J. A. Fessler, and A. Macovski. A Bayesian approach to reconstruction from incomplete projections of a multiple object 3D domain. *IEEE Trans. Pat. Anal. Mach. Intell.*, 11(8):840–858, August 1989.
- [4] James F. Brinkley. Knowledge-driven ultrasonic three-dimensional organ modeling. *IEEE Trans. Pat. Anal. Mach. Intell.*, 7(4):431–441, July 1985.
- [5] David P. Dobkin, Silvio V. F. Levy, William P. Thurston, and Allan R. Wilks. Contour tracing by piecewise linear approximations. *ACM TOG*, 9(4):389–423, October 1990.
- [6] John A. Eisenman. Graphical editing of composite bezier curves. Master’s thesis, Department of Electrical Engineering and Computer Science, M.I.T., 1988.
- [7] T.A. Foley. Interpolation to scattered data on a spherical domain. In M. Cox and J. Mason, editors, *Algorithms for Approximation II*, pages 303–310. Chapman and Hall, London, 1990.
- [8] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.
- [9] T. Hastie and W. Stuetzle. Principal curves. *JASA*, 84:502–516, 1989.
- [10] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, Inc., second edition, 1991.
- [11] Marshal L. Merriam. Experience with the cyberware 3D digitizer. In *NCGA Proceedings*, pages 125–133, March 1992.
- [12] David Meyers, Shelly Skinner, and Kenneth Sloan. Surfaces from contours: The correspondence and branching problems. In *Proceedings of Graphics Interface '91*, pages 246–254, June 1991.
- [13] Doug Moore and Joe Warren. Approximation of dense scattered data using algebraic surfaces. TR 90-135, Rice University, October 1990.
- [14] Doug Moore and Joe Warren. Adaptive mesh generation ii: Packing solids. TR 90-139, Rice University, March 1991.
- [15] Shigeru Muraki. Volumetric shape description of range data using “blobby model”. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):227–235, July 1991.
- [16] Gregory M. Nielson, Thomas A. Foley, Bernd Hamann, and David Lane. Visualizing and modeling scattered multivariate data. *IEEE CG&A*, 11(3):47–55, May 1991.
- [17] Barrett O’Neill. *Elementary Differential Geometry*. Academic Press, Orlando, Florida, 1966.
- [18] Vaughan Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):145–152, July 1987.

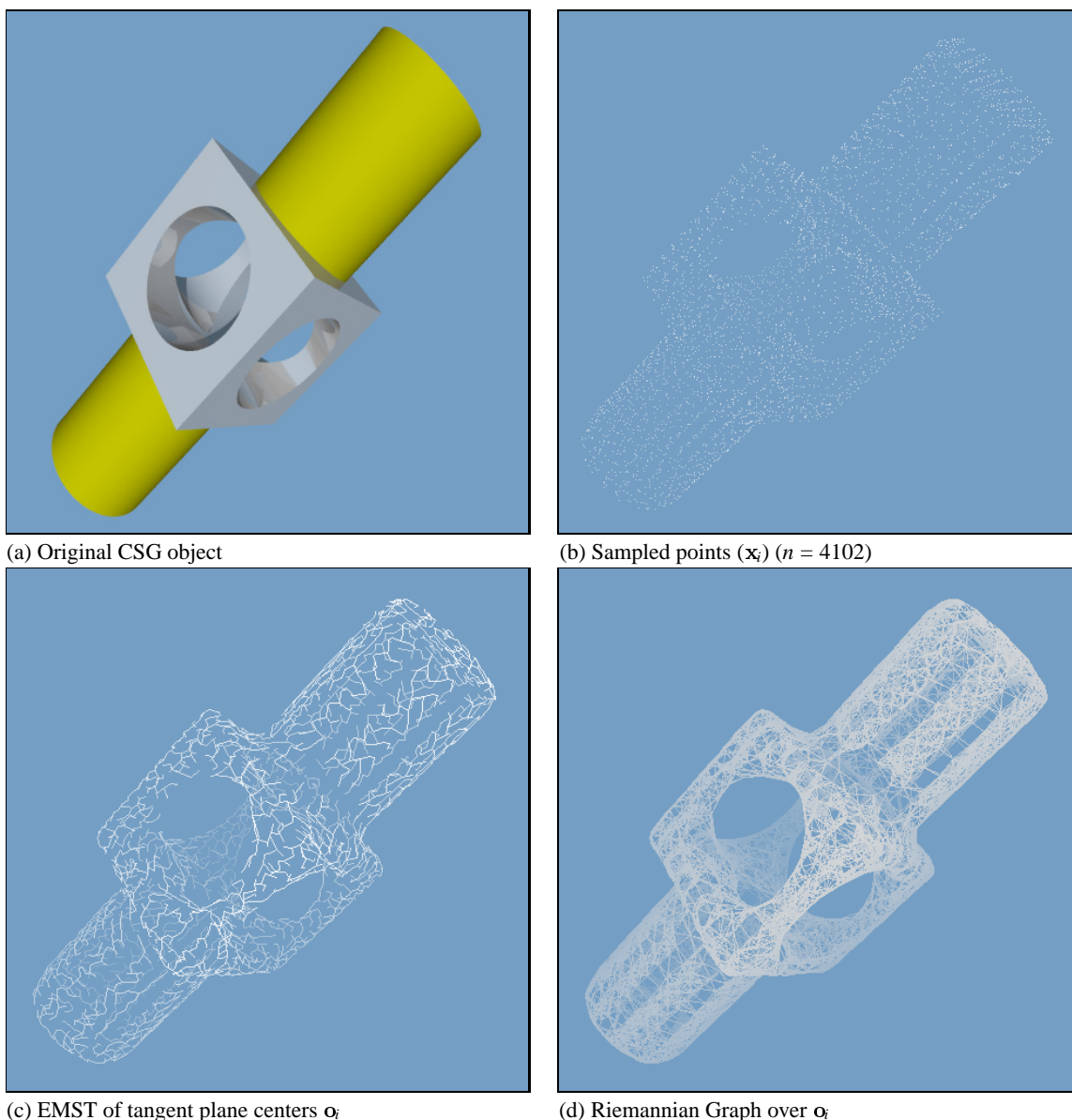
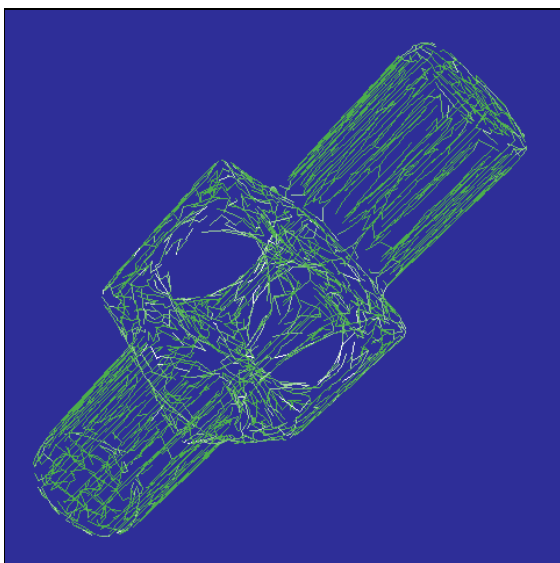


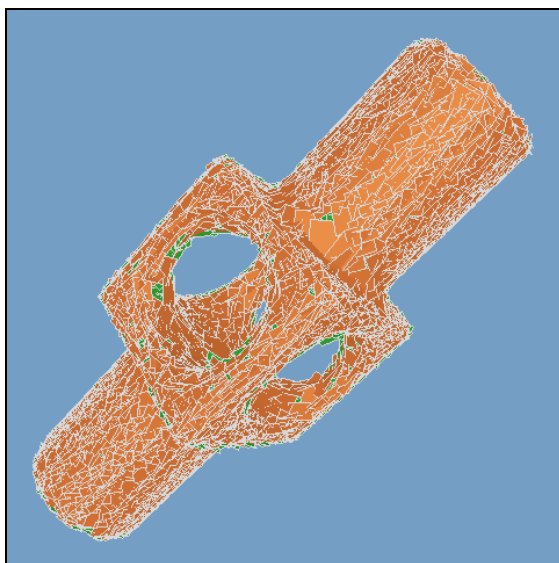
Figure 1: Reconstruction of ray-traced CSG object (simulated multi-view range data).

- [19] Emanuel Sachs, Andrew Roberts, and David Stoops. 3-Draw: A tool for designing 3D shapes. *IEEE Computer Graphics and Applications*, 11(6):18–26, November 1991.
- [20] Hanan Samet. *Applications of Spatial Data Structures*. Addison-Wesley, 1990.
- [21] Philip J. Schneider. Phoenix: An interactive curve design system based on the automatic fitting of hand-sketched curves. Master's thesis, Department of Computer Science, U. of Washington, 1988.
- [22] R. B. Schudy and D. H. Ballard. Model detection of cardiac chambers in ultrasound images. Technical Report 12, Computer Science Department, University of Rochester, 1978.
- [23] R. B. Schudy and D. H. Ballard. Towards an anatomical model of heart motion as seen in 4-d cardiac ultrasound data. In *Proceedings of the 6th Conference on Computer Applications in Radiology and Computer-Aided Analysis of Radiological Images*, 1979.
- [24] Stan Sclaroff and Alex Pentland. Generalized implicit functions for computer graphics. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):247–250, July 1991.
- [25] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. Technical Report LEMS-66, Division of Engineering, Brown University, 1990.
- [26] B. C. Vemuri. *Representation and Recognition of Objects From Dense Range Maps*. PhD thesis, Department of Electrical and Computer Engineering, University of Texas at Austin, 1987.
- [27] B. C. Vemuri, A. Mitiche, and J. K. Aggarwal. Curvature-based representation of objects from range data. *Image and Vision Computing*, 4(2):107–114, 1986.
- [28] G. Wyvill, C. McPheeters, and B. Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, August 1986.

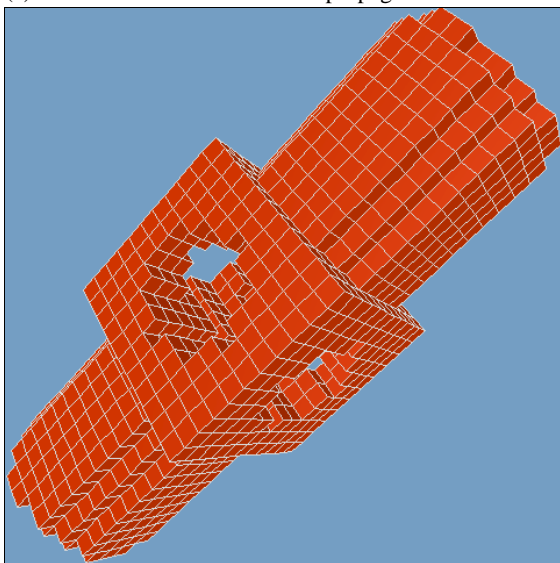




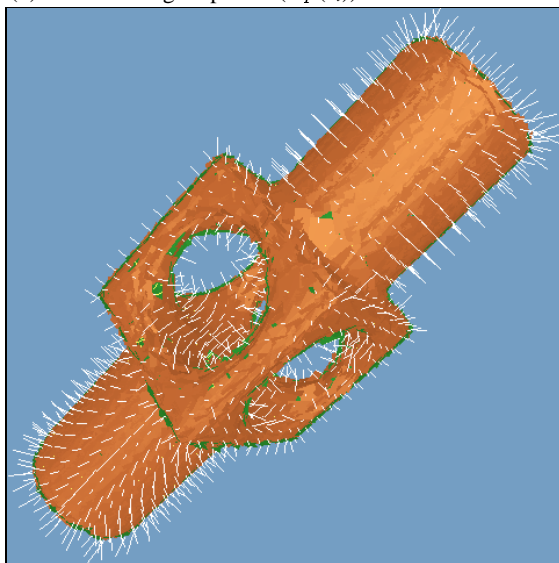
(a) Traversal order of orientation propagation



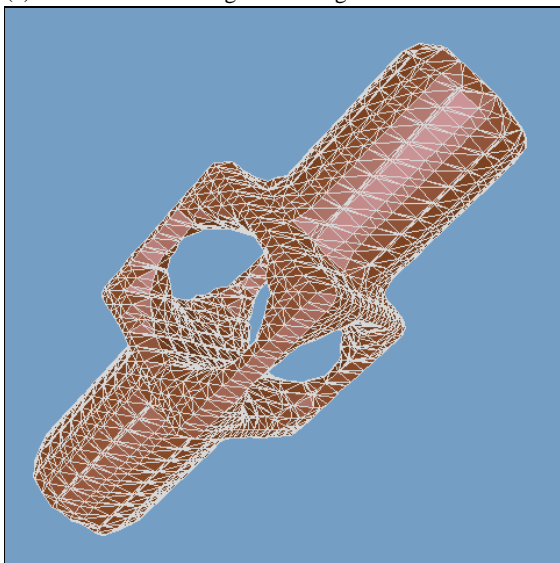
(b) Oriented tangent planes ( $Tp(\mathbf{x}_i)$ )



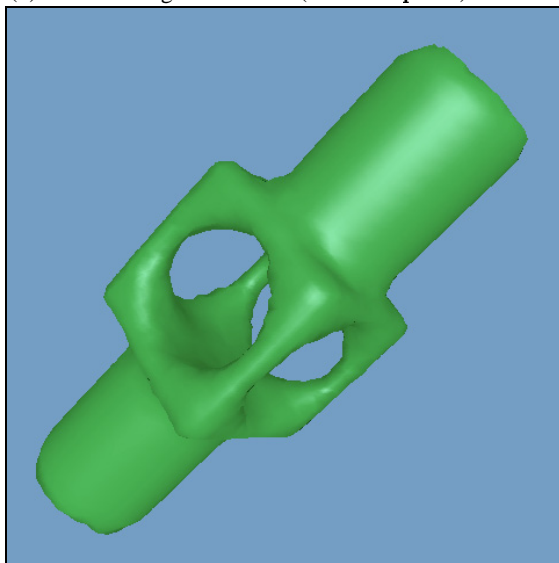
(c) Cubes visited during contouring



(d) Estimated signed distance (shown as  $\mathbf{p} - \mathbf{z}$ )

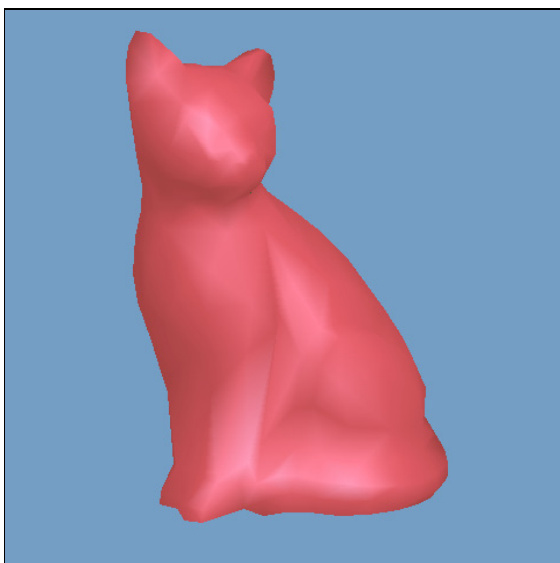


(e) Output of modified marching cubes



(f) Final surface after edge collapses

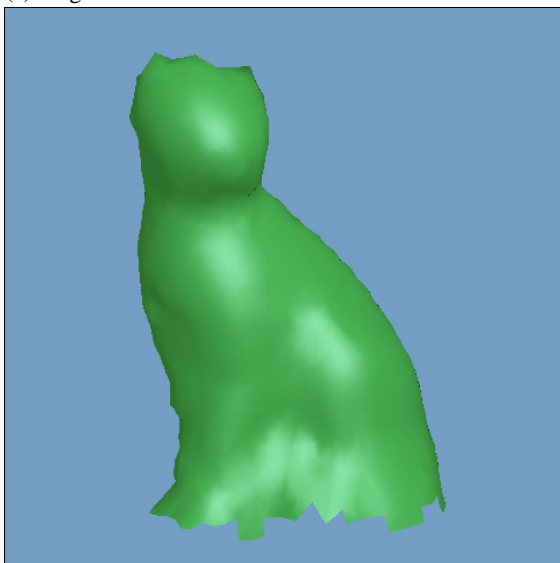
Figure 2: Reconstruction of ray-traced CSG object (continued).



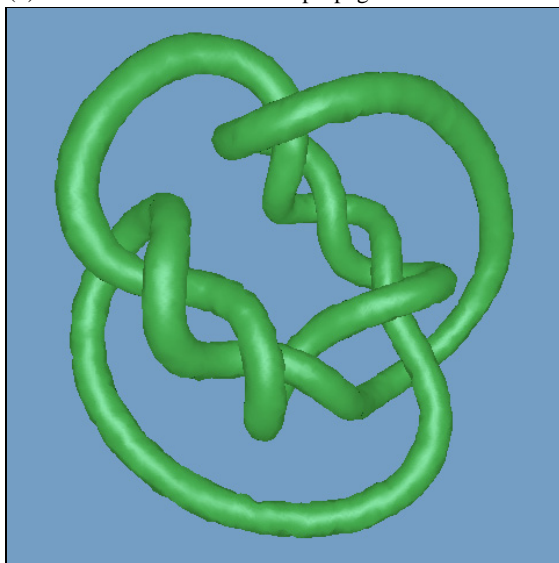
(a) Original mesh



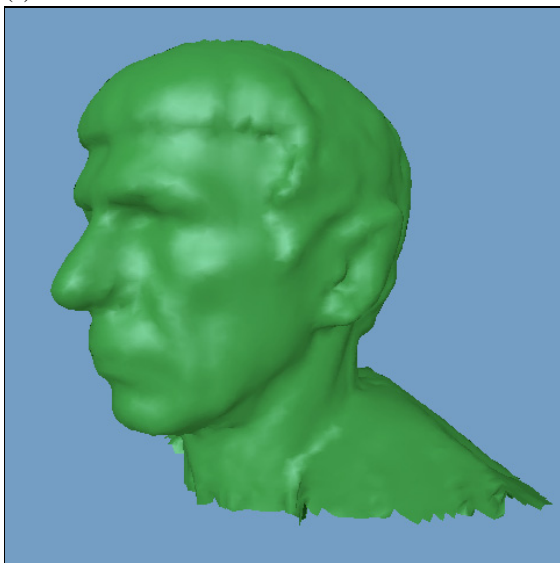
(b) Result of naive orientation propagation



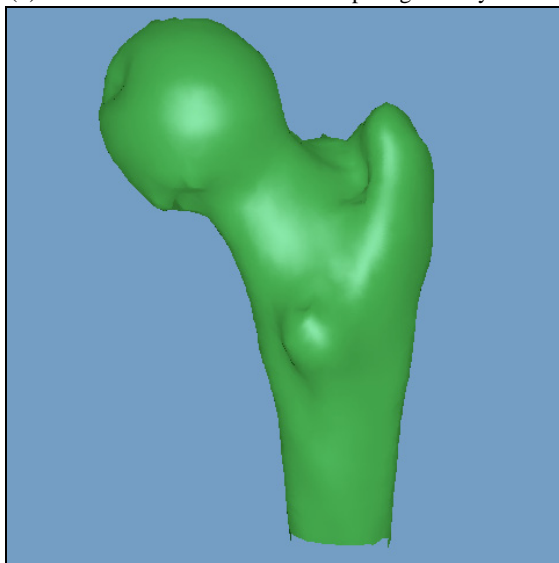
(c) Reconstructed bordered surface



(d) Reconstructed surface with complex geometry



(e) Reconstruction from cylindrical range data



(f) Reconstruction from contour data

Figure 3: Reconstruction examples.