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 $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^3$ 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 $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ 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 *δ-noisy* if $\|\mathbf{e}_i\| \leq \delta$ for all i. A value for δ can be estimated in most applications (e.g., the accuracy of the laser scanner). Features of M that are small compared to δ 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 $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset M$ be a (noiseless) sample of a surface M. The sample Y is said to be *ρ-dense* if any sphere with radius ρ and center in M contains at least one sample point in Y. A δ-noisy sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^3$ of a surface M is said to be ρ-dense if there exists a noiseless ρ-dense sample $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset M$ such that $\mathbf{x}_i = \mathbf{y}_i + \mathbf{e}_i$, $\|\mathbf{e}_i\| \leq \delta$, $i = 1, \ldots, n$.

1.2 Problem Statement

The goal of *surface reconstruction* is to determine a surface 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 δ and the sampling density ρ.

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 $f : \mathbb{R}^3 \to \mathbb{R}$ such that $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ 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 Λ 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 Λ 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 Λ 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 $\{y_i \in \mathbb{R}\}$, determine a function $f : M \to \mathbb{R}$, such that $f(\mathbf{x}_i) \approx y_i$.

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 $f : D \to \mathbb{R}$, 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 3×3 positive semi-definite matrix

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

where $\otimes$ denotes the outer product vector operator[1]. 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 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 $\{\mathbf{o}_1, \ldots, \mathbf{o}_n\}$ (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 $\{\mathbf{o}_1, \ldots, \mathbf{o}_n\}$ 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 ± 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 $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is assumed to be a ρ-dense, δ-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

[1] 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.

a point of M since that would violate X being ρ-dense. Intuitively, the sample points do not leave holes of radius larger than ρ. If the sample is δ-noisy, the radius of the holes may increase, but by no more than δ. 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**
    $f(\mathbf{p}) \leftarrow (\mathbf{p} - \mathbf{o}_i) \cdot \hat{\mathbf{n}}_i$ $\quad \{= \pm\|\mathbf{p} - \mathbf{z}\|\}$
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.[2] The final result is shown in Figure 2f. Alternatively, other contouring methods exist that can guarantee bounds on the triangle aspect ratio [14].

[2]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.

Meshes : Points were randomly sampled from a number of existing simplicial surfaces[3]. 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	∞	1/50	137
mechpart	4102	12	∞	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

[3]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" λ_i^3 of the neighborhood. Another possible criterion is to compare λ_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.

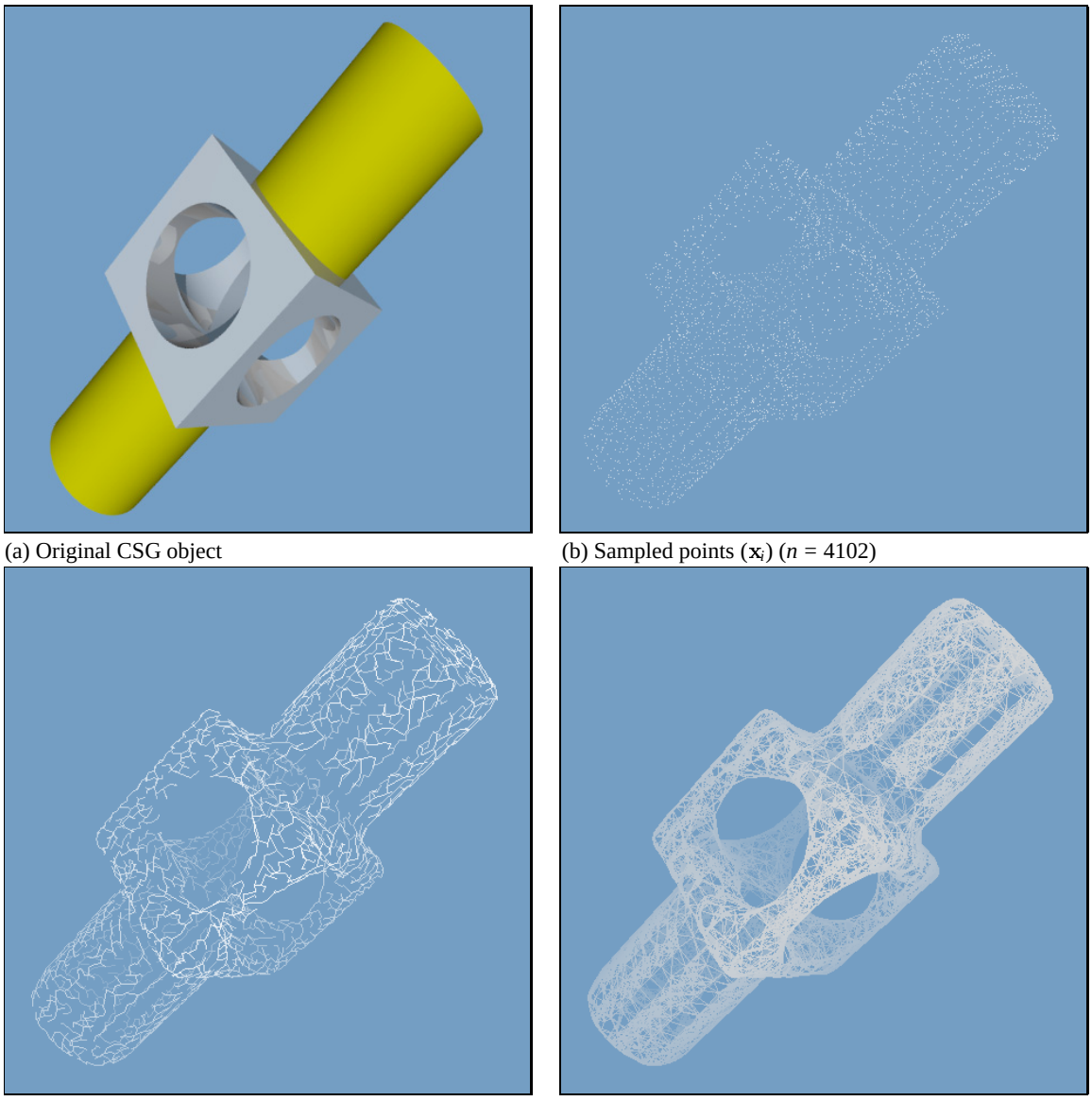
[14] Doug Moore and Joe Warren. Adaptive mesh generation ii: Packing solids. TR 90-139, Rice University, March 1991.

[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.



(a) Original CSG object

(b) Sampled points ($\mathbf{x}_i$) (n = 4102)

(c) EMST of tangent plane centers $\mathbf{o}_i$

(d) Riemannian Graph over $\mathbf{o}_i$

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.

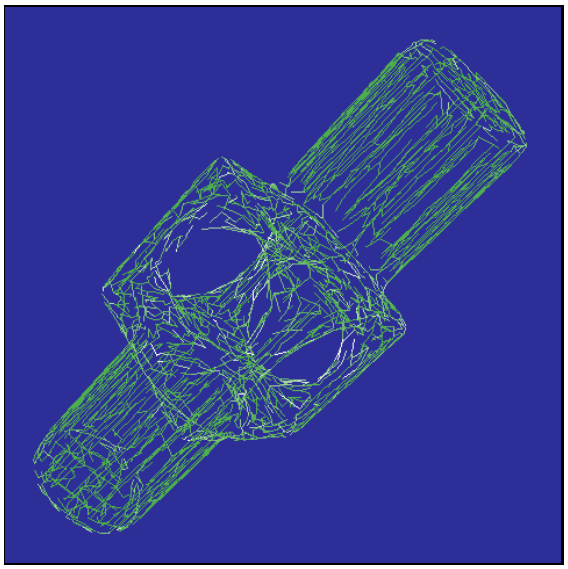
[24] Stan Sclaroff and Alex Pentland. Generalized implicit functions for computer graphics. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):247–250, July 1991.

[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.

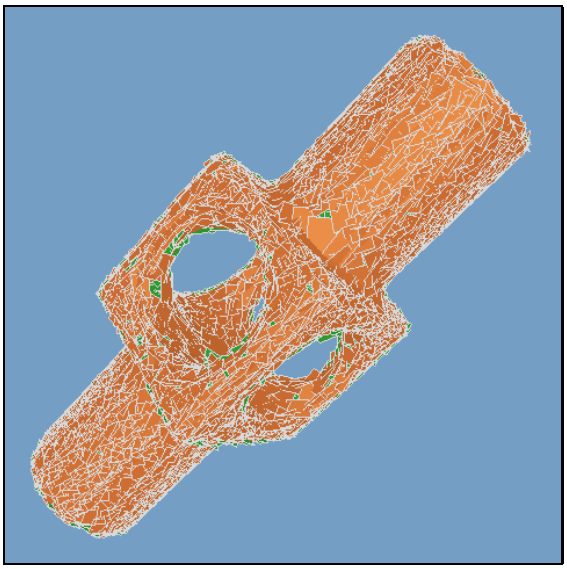
[26] B. C. Vemuri. *Representation and Recognition of Objects From Dense Range Maps*. PhD thesis, Department of Electrical and Computer Engineering, University of Texas at Austin, 1987.

[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.

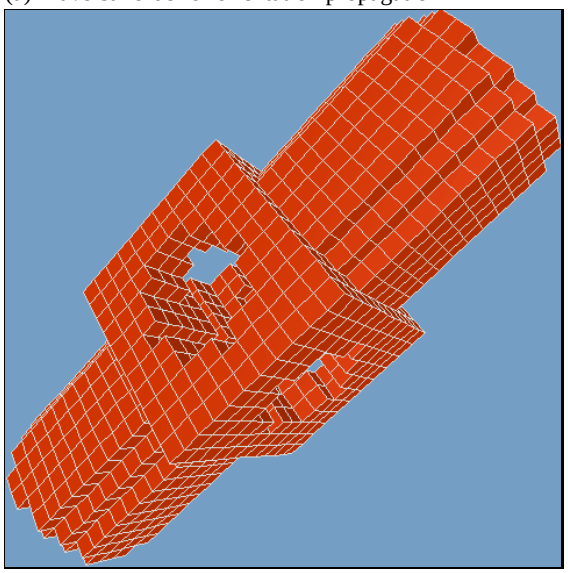
[28] G. Wyvill, C. McPheeters, and B. Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, August 1986.



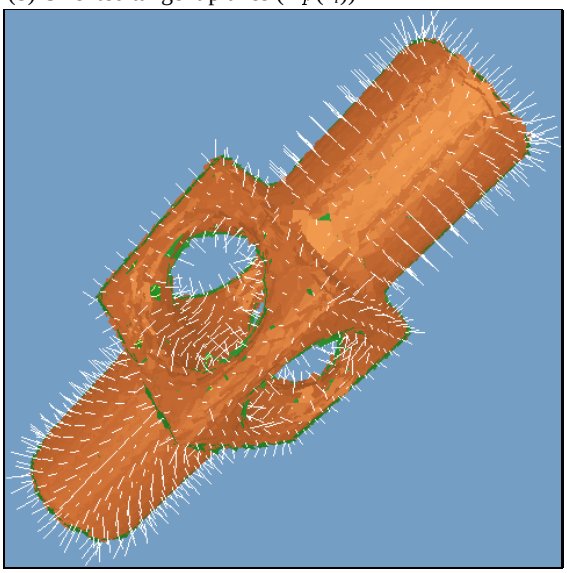
(a) Traversal order of orientation propagation



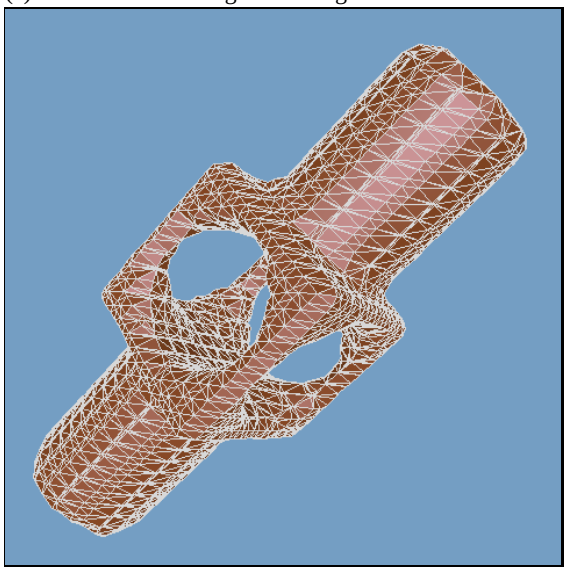
(b) Oriented tangent planes ($Tp(\mathbf{x}_i)$)



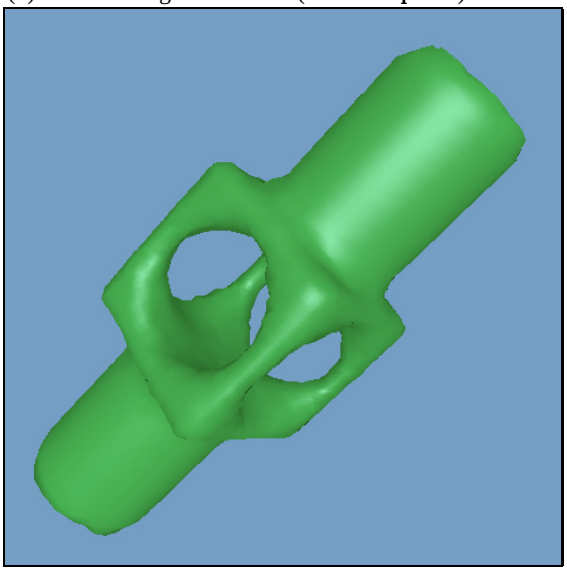
(c) Cubes visited during contouring



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

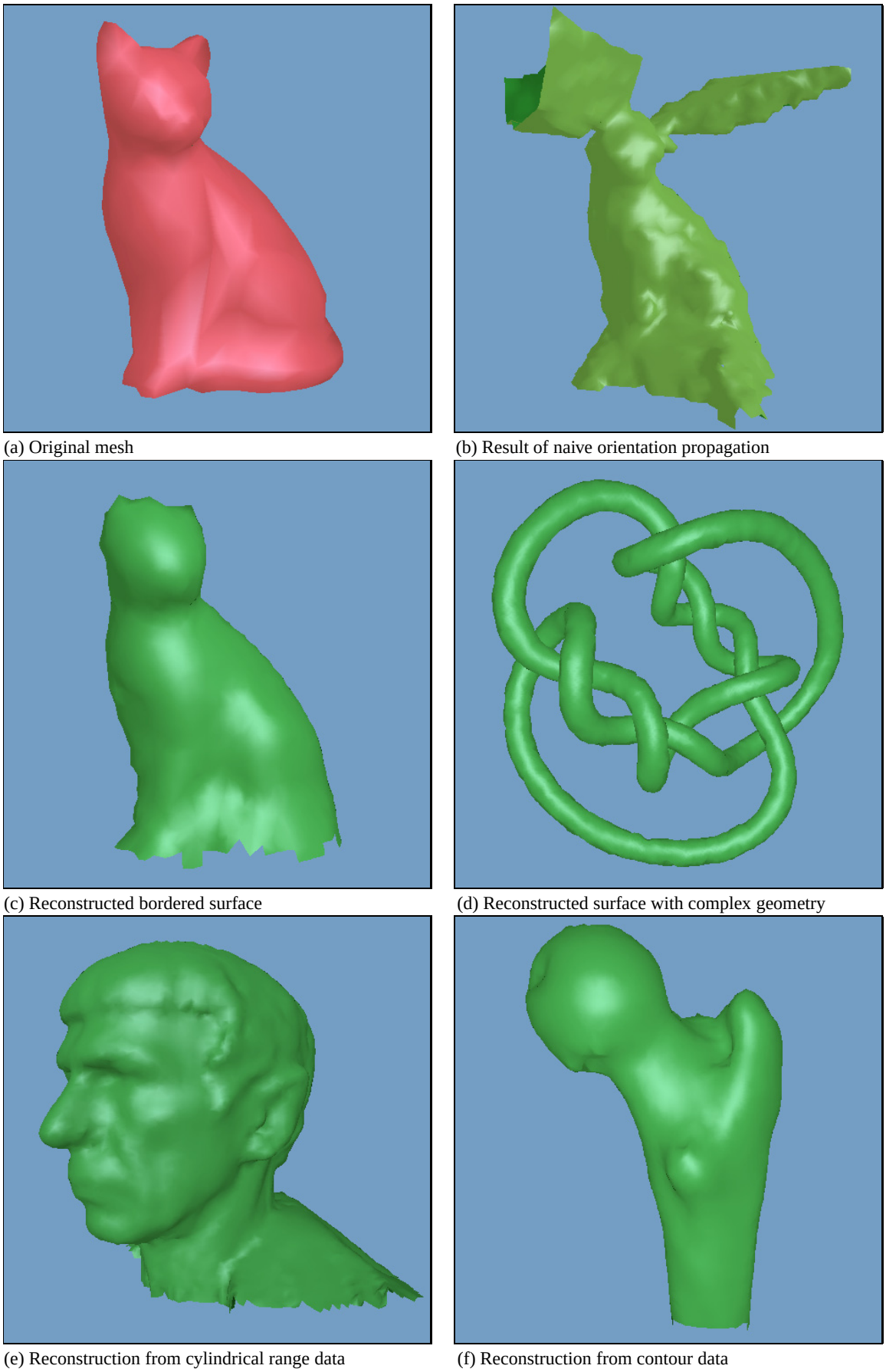


(e) Output of modified marching cubes



(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.

Eurographics Symposium on Geometry Processing (2006)
Konrad Polthier, Alla Sheffer (Editors)

Poisson Surface Reconstruction

Michael Kazhdan[1], Matthew Bolitho[1] and Hugues Hoppe[2]

[1]Johns Hopkins University, Baltimore MD, USA
[2]Microsoft Research, Redmond WA, USA

Abstract

We show that surface reconstruction from oriented points can be cast as a spatial Poisson problem. This Poisson formulation considers all the points at once, without resorting to heuristic spatial partitioning or blending, and is therefore highly resilient to data noise. Unlike radial basis function schemes, our Poisson approach allows a hierarchy of locally supported basis functions, and therefore the solution reduces to a well conditioned sparse linear system. We describe a spatially adaptive multiscale algorithm whose time and space complexities are proportional to the size of the reconstructed model. Experimenting with publicly available scan data, we demonstrate reconstruction of surfaces with greater detail than previously achievable.

1. Introduction

Reconstructing 3D surfaces from point samples is a well studied problem in computer graphics. It allows fitting of scanned data, filling of surface holes, and remeshing of existing models. We provide a novel approach that expresses surface reconstruction as the solution to a Poisson equation.

Like much previous work (Section 2), we approach the problem of surface reconstruction using an implicit function framework. Specifically, like [Kaz05] we compute a 3D *indicator function* χ (defined as 1 at points inside the model, and 0 at points outside), and then obtain the reconstructed surface by extracting an appropriate isosurface.

Our key insight is that there is an integral relationship between oriented points sampled from the surface of a model and the indicator function of the model. Specifically, the gradient of the indicator function is a vector field that is zero almost everywhere (since the indicator function is constant almost everywhere), except at points near the surface, where it is equal to the inward surface normal. Thus, the oriented point samples can be viewed as samples of the gradient of the model's indicator function (Figure 1).

The problem of computing the indicator function thus reduces to inverting the gradient operator, i.e. finding the scalar function χ whose gradient best approximates a vector field $\vec{V}$ defined by the samples, i.e. $\min_\chi \|\nabla\chi - \vec{V}\|$. If we apply the divergence operator, this variational problem transforms into a standard Poisson problem: compute the scalar function χ whose Laplacian (divergence of gradient) equals the divergence of the vector field $\vec{V}$,

$$\Delta\chi \equiv \nabla\cdot\nabla\chi = \nabla\cdot\vec{V}.$$

We will make these definitions precise in Sections 3 and 4.

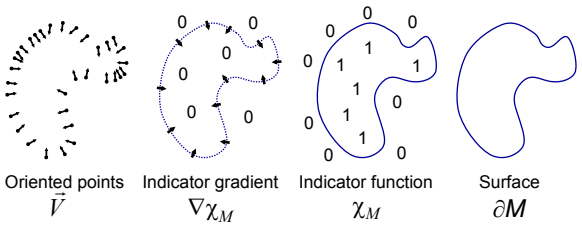


Figure 1: *Intuitive illustration of Poisson reconstruction in 2D.*

Formulating surface reconstruction as a Poisson problem offers a number of advantages. Many implicit surface fitting methods segment the data into regions for local fitting, and further combine these local approximations using blending functions. In contrast, Poisson reconstruction is a global solution that considers all the data at once, without resorting to heuristic partitioning or blending. Thus, like radial basis function (RBF) approaches, Poisson reconstruction creates very smooth surfaces that robustly approximate noisy data. But, whereas ideal RBFs are globally supported and non-decaying, the Poisson problem admits a hierarchy of *locally supported* functions, and therefore its solution reduces to a well-conditioned sparse linear system.

Moreover, in many implicit fitting schemes, the value of the implicit function is constrained only near the sample points, and consequently the reconstruction may contain spurious surface sheets away from these samples. Typically this problem is attenuated by introducing auxiliary "off-surface" points (e.g. [CBC*01, OBA*03]). With Poisson surface reconstruction, such surface sheets seldom arise because the gradient of the implicit function is constrained at *all* spatial points. In particular it is constrained to zero away from the samples.

Poisson systems are well known for their resilience in the presence of imperfect data. For instance, "gradient domain" manipulation algorithms (e.g. [FLW02]) intentionally modify the gradient data such that it no longer corresponds to any real potential field, and rely on a Poisson system to recover the globally best-fitting model.

There has been broad interdisciplinary research on solving Poisson problems and many efficient and robust methods have been developed. One particular aspect of our problem instance is that an accurate solution to the Poisson equation is only necessary near the reconstructed surface. This allows us to leverage adaptive Poisson solvers to develop a reconstruction algorithm whose spatial and temporal complexities are proportional to the size of the reconstructed surface.

2. Related Work

Surface reconstruction The reconstruction of surfaces from oriented points has a number of difficulties in practice. The point sampling is often nonuniform. The positions and normals are generally noisy due to sampling inaccuracy and scan misregistration. And, accessibility constraints during scanning may leave some surface regions devoid of data. Given these challenges, reconstruction methods attempt to infer the topology of the unknown surface, accurately fit (but not overfit) the noisy data, and fill holes reasonably.

Several approaches are based on combinatorial structures, such as Delaunay triangulations (e.g. [Boi84, KSO04]), alpha shapes [EM94, BBX95, BMR*99]), or Voronoi diagrams [ABK98, ACK01]. These schemes typically create a triangle mesh that interpolates all or a most of the points. In the presence of noisy data, the resulting surface is often jagged, and is therefore smoothed (e.g. [KSO04]) or refit to the points (e.g. [BBX95]) in subsequent processing.

Other schemes directly reconstruct an approximating surface, typically represented in implicit form. We can broadly classify these as either global or local approaches.

Global fitting methods commonly define the implicit function as the sum of radial basis functions (RBFs) centered at the points (e.g. [Mur91, CBC*01, TO02]). However, the ideal RBFs (polyharmonics) are globally supported and nondecaying, so the solution matrix is dense and ill-conditioned. Practical solutions on large datasets involve adaptive RBF reduction and the fast multipole method [CBC*01].

Local fitting methods consider subsets of nearby points at a time. A simple scheme is to estimate tangent planes and define the implicit function as the signed distance to the tangent plane of the closest point [HDD*92]. Signed distance can also be accumulated into a volumetric grid [CL96]. For function continuity, the influence of several nearby points can be blended together, for instance using moving least squares [ABCO*01, SOS04]. A different approach is to form point neighborhoods by adaptively subdividing space, for example with an adaptive octree. Blending is possible over an octree structure using a multilevel partition of unity, and the type of local implicit patch within each octree node can be selected heuristically [OBA*03].

Our Poisson reconstruction combines benefits of both global and local fitting schemes. It is global and therefore does not involve heuristic decisions for forming local neighborhoods, selecting surface patch types, and choosing blend weights. Yet, the basis functions are associated with the ambient space rather than the data points, are locally supported, and have a simple hierarchical structure that results in a sparse, well-conditioned system.

Our approach of solving an indicator function is similar to the Fourier-based reconstruction scheme of Kazhdan [Kaz05]. In fact, we show in Appendix A that our basic Poisson formulation is mathematically equivalent. Indeed, the Fast Fourier Transform (FFT) is a common technique for solving *dense*, *periodic* Poisson systems. However, the FFT requires $O(r^3 \log r)$ time and $O(r^3)$ space where r is the 3D grid resolution, quickly becoming prohibitive for fine resolutions. In contrast, the Poisson system allows adaptive discretization, and thus yields a scalable solution.

Poisson problems The Poisson equation arises in numerous applications areas. For instance, in computer graphics it is used for tone mapping of high dynamic range images [FLW02], seamless editing of image regions [PGB03], fluid mechanics [LGF04], and mesh editing [YZX*04]. Multigrid Poisson solutions have even been adapted for efficient GPU computation [BFGS03, GWL*03].

The Poisson equation is also used in heat transfer and diffusion problems. Interestingly, Davis et al [DMGL02] use diffusion to fill holes in reconstructed surfaces. Given boundary conditions in the form of a clamped signed distance function d, their diffusion approach essentially solves the homogeneous Poisson equation $\Delta d = 0$ to create an implicit surface spanning the boundaries. They use a local iterative solution rather than a global multiscale Poisson system.

Nehab et al [NRDR05] use a Poisson system to fit a 2.5D height field to sampled positions and normals. Their approach fits a given parametric surface and is well-suited to the reconstruction of surfaces from individual scans. However, in the case that the samples are obtained from the union of multiple scans, their approach cannot be directly applied to obtain a connected, watertight surface.

3. Our Poisson reconstruction approach

The input data S is a set of samples $s \in S$, each consisting of a point $s.p$ and an inward-facing normal $s.\vec{N}$, assumed to lie on or near the surface ∂M of an unknown model M. Our goal is to reconstruct a watertight, triangulated approximation to the surface by approximating the indicator function of the model and extracting the isosurface, as illustrated in Figure 2.

The key challenge is to accurately compute the indicator function from the samples. In this section, we derive a relationship between the gradient of the indicator function and an integral of the surface normal field. We then approximate this surface integral by a summation over the given oriented point samples. Finally, we reconstruct the indicator function from this gradient field as a Poisson problem.

Defining the gradient field Because the indicator function is a piecewise constant function, explicit computation of its gradient field would result in a vector field with unbounded values at the surface boundary. To avoid this, we convolve the indicator function with a smoothing filter and consider the gradient field of the smoothed function. The following lemma formalizes the relationship between the gradient of the smoothed indicator function and the surface normal field.

Lemma: Given a solid M with boundary ∂M, let χ_M denote the indicator function of M, $\vec{N}_{\partial M}(p)$ be the inward surface normal at $p \in \partial M$, $\tilde{F}(q)$ be a smoothing filter, and $\tilde{F}_p(q) = \tilde{F}(q-p)$ its translation to the point p. The gradient of the smoothed indicator function is equal to the vector field obtained by smoothing the surface normal field:

$$\nabla \left(\chi_M * \tilde{F}\right)(q_0) = \int_{\partial M} \tilde{F}_p(q_0)\vec{N}_{\partial M}(p)dp. \qquad (1)$$

Proof: To prove this, we show equality for each of the components of the vector field. Computing the partial derivative of the smoothed indicator function with respect to x, we get:

$$\begin{aligned}
\left.\frac{\partial}{\partial x}\right|_{q_0} \left(\chi_M * \tilde{F}\right) &= \left.\frac{\partial}{\partial x}\right|_{q=q_0} \int_M \tilde{F}(q-p)dp \\
&= \int_M \left(-\frac{\partial}{\partial x}\tilde{F}(q_0-p)\right) dp \\
&= -\int_M \nabla \cdot \left(\tilde{F}(q_0-p),0,0\right) dp \\
&= \int_{\partial M} \left\langle \left(\tilde{F}_p(q_0),0,0\right), \vec{N}_{\partial M}(p) \right\rangle dp.
\end{aligned}$$

(The first equality follows from the fact that χ_M is equal to zero outside of M and one inside. The second follows from the fact that $(\partial/\partial q)\tilde{F}(q-p) = -(\partial/\partial p)\tilde{F}(q-p)$. The last follows from the Divergence Theorem.)

A similar argument shows that the y-, and z-components of the two sides are equal, thereby completing the proof. □

Approximating the gradient field Of course, we cannot evaluate the surface integral since we do not yet know the surface geometry. However, the input set of oriented points provides precisely enough information to approximate the integral with a discrete summation. Specifically, using the point set S to partition ∂M into distinct patches $\mathscr{P}_s \subset \partial M$, we can approximate the integral over a patch $\mathscr{P}_s$ by the value at point sample $s.p$, scaled by the area of the patch:

$$\begin{aligned}
\nabla(\chi_M * \tilde{F})(q) &= \sum_{s\in S} \int_{\mathscr{P}_s} \tilde{F}_p(q)\vec{N}_{\partial M}(p)dp \\
&\approx \sum_{s\in S} |\mathscr{P}_s|\, \tilde{F}_{s.p}(q)\, s.\vec{N} \;\equiv\; \vec{V}(q).
\end{aligned} \qquad (2)$$

It should be noted that though Equation 1 is true for any smoothing filter $\tilde{F}$, in practice, care must be taken in choosing the filter. In particular, we would like the filter to satisfy two conditions. On the one hand, it should be sufficiently narrow so that we do not over-smooth the data. And on the other hand, it should be wide enough so that the integral over $\mathscr{P}_s$ is well approximated by the value at $s.p$ scaled by the patch area. A good choice of filter that balances these two requirements is a Gaussian whose variance is on the order of the sampling resolution.

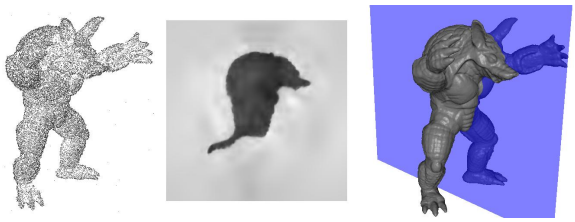


Figure 2: *Points from scans of the "Armadillo Man" model (left), our Poisson surface reconstruction (right), and a visualization of the indicator function (middle) along a plane through the 3D volume.*

Solving the Poisson problem Having formed a vector field $\vec{V}$, we want to solve for the function $\tilde{\chi}$ such that $\nabla\tilde{\chi} = \vec{V}$. However, $\vec{V}$ is generally not integrable (i.e. it is not curl-free), so an exact solution does not generally exist. To find the best least-squares approximate solution, we apply the divergence operator to form the Poisson equation

$$\Delta\tilde{\chi} = \nabla \cdot \vec{V}.$$

In the next section, we describe our implementation of these steps in more detail.

4. Implementation

We first present our reconstruction algorithm under the assumption that the point samples are uniformly distributed over the model surface. We define a space of functions with high resolution near the surface of the model and coarser resolution away from it, express the vector field $\vec{V}$ as a linear sum of functions in this space, set up and solve the Poisson equation, and extract an isosurface of the resulting indicator function. We then extend our algorithm to address the case of non-uniformly sampled points.

4.1. Problem Discretization

First, we must choose the space of functions in which to discretize the problem. The most straightforward approach is to start with a regular 3D grid [Kaz05], but such a uniform structure becomes impractical for fine-detail reconstruction, since the dimension of the space is cubic in the resolution while the number of surface triangles grows quadratically.

Fortunately, an accurate representation of the implicit function is only necessary near the reconstructed surface. This motivates the use of an adaptive octree both to represent the implicit function and to solve the Poisson system (e.g. [GKS02, LGF04]). Specifically, we use the positions of the sample points to define an octree $\mathcal{O}$ and associate a function F_o to each node $o \in \mathcal{O}$ of the tree, choosing the tree and the functions so that the following conditions are satisfied:

1. The vector field $\vec{V}$ can be precisely and efficiently represented as the linear sum of the F_o.
2. The matrix representation of the Poisson equation, expressed in terms of the F_o can be solved efficiently.
3. A representation of the indicator function as the sum of the F_o can be precisely and efficiently evaluated near the surface of the model.

Defining the function space Given a set of point samples S and a maximum tree depth D, we define the octree $\mathcal{O}$ to be the minimal octree with the property that every point sample falls into a leaf node at depth D.

Next, we define a space of functions obtained as the span of translates and scales of a fixed, unit-integral, base function $F : \mathbb{R}^3 \to \mathbb{R}$. For every node $o \in \mathcal{O}$, we set F_o to be the unit-integral "node function" centered about the node o and stretched by the size of o:

$$F_o(q) \equiv F\left(\frac{q - o.c}{o.w}\right)\frac{1}{o.w^3}.$$

where $o.c$ and $o.w$ are the center and width of node o.

This space of functions $\mathcal{F}_{\mathcal{O},F} \equiv \text{Span}\{F_o\}$ has a multiresolution structure similar to that of traditional wavelet representations. Finer nodes are associated with higher-frequency functions, and the function representation becomes more precise as we near the surface.

Selecting a base function In selecting a base function F, our goal is to choose a function so that the vector field $\vec{V}$, defined in Equation 2, can be precisely and efficiently represented as the linear sum of the node functions $\{F_o\}$.

If we were to replace the position of each sample with the center of the leaf node containing it, the vector field $\vec{V}$ could be efficiently expressed as the linear sum of $\{F_o\}$ by setting:

$$F(q) = \tilde{F}\left(\frac{q}{2^D}\right).$$

This way, each sample would contribute a single term (the normal vector) to the coefficient corresponding to its leaf's node function. Since the sampling width is 2^{-D} and the samples all fall into leaf nodes of depth D, the error arising from the clamping can never be too big (at most, on the order of half the sampling width). In the next section, we show how the error can be further reduced by using trilinear interpolation to allow for sub-node precision.

Finally, since a maximum tree depth of D corresponds to a sampling width of 2^{-D}, the smoothing filter should approximate a Gaussian with variance on the order of 2^{-D}. Thus, F should approximate a Gaussian with unit-variance.

For efficiency, we approximate the unit-variance Gaussian by a compactly supported function so that (1) the resulting Divergence and Laplacian operators are sparse and (2) the evaluation of a function expressed as the linear sum of F_o at some point q only requires summing over the nodes $o \in \mathcal{O}$ that are close to q. Thus, we set F to be the n-th convolution of a box filter with itself resulting in the base function F:

$$F(x,y,z) \equiv (B(x)B(y)B(z))^{*n} \quad \text{with} \quad B(t) = \begin{cases} 1 & |t| < 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Note that as n is increased, F more closely approximates a Gaussian and its support grows larger; in our implementation we use a piecewise quadratic approximation with $n = 3$. Therefore, the function F is supported on the domain $[-1.5, 1.5]^3$ and, for the basis function of any octree node, there are at most $5^3\text{-}1 = 124$ other nodes at the same depth whose functions overlap with it.

4.2. Vector Field Definition

To allow for sub-node precision, we avoid clamping a sample's position to the center of the containing leaf node and instead use trilinear interpolation to distribute the sample across the eight nearest nodes. Thus, we define our approximation to the gradient field of the indicator function as:

$$\vec{V}(q) \equiv \sum_{s \in S} \sum_{o \in \text{Ngbr}_D(s)} \alpha_{o,s} F_o(q) s.\vec{N} \qquad (3)$$

where $\text{Ngbr}_D(s)$ are the eight depth-D nodes closest to $s.p$ and $\{\alpha_{o,s}\}$ are the trilinear interpolation weights. (If the neighbors are not in the tree, we refine it to include them.)

Since the samples are uniform, we can assume that the area of a patch $\mathcal{P}_s$ is constant and $\vec{V}$ is a good approximation, up to a multiplicative constant, of the gradient of the smoothed indicator function. We will show that the choice of multiplicative constant does not affect the reconstruction.

4.3. Poisson Solution

Having defined the vector field $\vec{V}$, we would like to solve for the function $\tilde{\chi} \in \mathcal{F}_{\mathcal{O},F}$ such that the gradient of $\tilde{\chi}$ is closest to $\vec{V}$, i.e. a solution to the Poisson equation $\Delta\tilde{\chi} = \nabla \cdot \vec{V}$.

One challenge of solving for $\tilde{\chi}$ is that though $\tilde{\chi}$ and the

coordinate functions of $\vec{V}$ are in the space $\mathcal{F}_{\mathcal{O},F}$ it is not necessarily the case that the functions $\Delta\tilde{\chi}$ and $\nabla\cdot\vec{V}$ are.

To address this issue, we need to solve for the function $\tilde{\chi}$ such that the projection of $\Delta\tilde{\chi}$ onto the space $\mathcal{F}_{\mathcal{O},F}$ is closest to the projection of $\nabla\cdot\vec{V}$. Since, in general, the functions F_o do not form an orthonormal basis, solving this problem directly is expensive. However, we can simplify the problem by solving for the function $\tilde{\chi}$ minimizing:

$$\sum_{o\in\mathcal{O}}\left\|\langle\Delta\tilde{\chi}-\nabla\cdot\vec{V},F_o\rangle\right\|^2=\sum_{o\in\mathcal{O}}\left\|\langle\Delta\tilde{\chi},F_o\rangle-\langle\nabla\cdot\vec{V},F_o\rangle\right\|^2.$$

Thus given the $|\mathcal{O}|$-dimensional vector v whose o-th coordinate is $v_o=\langle\nabla\cdot\vec{V},F_o\rangle$, the goal is to solve for the function $\tilde{\chi}$ such that the vector obtained by projecting the Laplacian of $\tilde{\chi}$ onto each of the F_o is as close to v as possible.

To express this in matrix form, let $\tilde{\chi}=\sum_o x_o F_o$, so that we are solving for the vector $x\in\mathbb{R}^{|\mathcal{O}|}$. Then, let us define the $|\mathcal{O}|\times|\mathcal{O}|$ matrix L such that Lx returns the dot product of the Laplacian with each of the F_o. Specifically, for all $o,o'\in\mathcal{O}$, the (o,o')-th entry of L is set to:

$$L_{o,o'}\equiv\left\langle\frac{\partial^2F_o}{\partial x^2},F_{o'}\right\rangle+\left\langle\frac{\partial^2F_o}{\partial y^2},F_{o'}\right\rangle+\left\langle\frac{\partial^2F_o}{\partial z^2},F_{o'}\right\rangle.$$

Thus, solving for $\tilde{\chi}$ amounts to finding

$$\min_{x\in\mathbb{R}^{|\mathcal{O}|}}\|Lx-v\|^2.$$

Note that the matrix L is sparse and symmetric. (Sparse because the F_o are compactly supported, and symmetric because $\int f''g=-\int f'g'$.) Furthermore, there is an inherent multiresolution structure on $\mathcal{F}_{\mathcal{O},F}$, so we use an approach similar to the multigrid approach in [GKS02], solving the restriction L_d of L to the space spanned by the depth d functions (using a conjugate gradient solver) and projecting the fixed-depth solution back onto $\mathcal{F}_{\mathcal{O},F}$ to update the residual.

Addressing memory concerns In practice, as the depth increases, the matrix L_d becomes larger and it may not be practical to store it in memory. Although the number of entries in a column of L_d is bounded by a constant, the constant value can be large. For example, even using a piecewise quadratic base function F, we end up with as many as 125 non-zero entries in a column, resulting in a memory requirement that is 125 times larger than the size of the octree.

To address this issue, we augment our solver with a block Gauss-Seidel solver. That is, we decompose the d-th dimensional space into overlapping regions and solve the restriction of L_d to these different regions, projecting the local solutions back into the d-dimensional space and updating the residuals. By choosing the number of regions to be a function of the depth d, we ensure that the size of the matrix used by the solver never exceeds a desired memory threshold.

4.4. Isosurface Extraction

In order to obtain a reconstructed surface $\partial\tilde{M}$, it is necessary to first select an isovalue and then extract the corresponding isosurface from the computed indicator function.

We choose the isovalue so that the extracted surface closely approximates the positions of the input samples. We do this by evaluating $\tilde{\chi}$ at the sample positions and use the average of the values for isosurface extraction:

$$\partial\tilde{M}\equiv\{q\in\mathbb{R}^3\mid\tilde{\chi}(q)=\gamma\}\quad\text{with}\quad\gamma=\frac{1}{|S|}\sum_{s\in S}\tilde{\chi}(s.p).$$

This choice of isovalue has the property that scaling $\tilde{\chi}$ does not change the isosurface. Thus, knowing the vector field $\vec{V}$ up to a multiplicative constant provides sufficient information for reconstructing the surface.

To extract the isosurface from the indicator function, we use a method similar to previous adaptations of the Marching Cubes [LC87] to octree representations (e.g. [WG92, SFYC96, WKE99]). However, due to the nonconforming properties of our tree, we modify the reconstruction approach slightly, defining the positions of zero-crossings along an edge in terms of the zero-crossings computed by the finest level nodes adjacent to the edge. In the case that an edge of a leaf node has more than one zero-crossing associated to it, the node is subdivided. As in previous approaches, we avoid cracks arising when coarser nodes share a face with finer ones by projecting the isocurve segments from the faces of finer nodes onto the face of the coarser one.

4.5. Non-uniform Samples

We now extend our method to the case of non-uniformly distributed point samples. As in [Kaz05], our approach is to estimate the local sampling density, and scale the contribution of each point accordingly. However, rather than simply scaling the *magnitude* of a *fixed*-width kernel associated with each point, we additionally adapt the kernel width. This results in a reconstruction that maintains sharp features in areas of dense sampling and provides a smooth fit in sparsely sampled regions.

Estimating local sampling density Following the approach of [Kaz05], we implement the density computation using a kernel density estimator [Par62]. The approach is to estimate the number of points in a neighborhood of a sample by "splatting" the samples into a 3D grid, convolving the "splatting" function with a smoothing filter, and evaluating the convolution at each of the sample points.

We implement the convolution in a manner similar to Equation 3. Given a depth $\hat{D}\leq D$ we set the density estimator to be the sum of node functions at depth $\hat{D}$:

$$W_{\hat{D}}(q)\equiv\sum_{s\in S}\sum_{o\in\mathrm{Ngbr}_{\hat{D}}(s)}\alpha_{o,s}F_o(q).$$

Since octree nodes at lower resolution are associated with functions that approximate Gaussians of larger width, the parameter $\hat{D}$ provides away for specifying the locality of the density estimation, with smaller values of $\hat{D}$ giving sampling density estimates over larger regions.

Computing the vector field Using the density estimator, we modify the summation in Equation 3 so that each sample's contribution is proportional to its associated area on the surface. Specifically, using the fact that the area is inversely proportional to sampling density, we set:

$$\vec{V}(q) \equiv \sum_{s \in S} \frac{1}{W_{\hat{D}}(s.p)} \sum_{o \in \mathrm{Ngbr}_D(s)} \alpha_{o,s} F_o(q).$$

However, adapting only the magnitudes of the sample contributions results in poor noise filtering in sparsely sampled regions as demonstrated later in Figure 7. Therefore, we additionally adapt the width of the smoothing filter $\tilde{F}$ to the local sampling density. Adapting the filter width lets us retain fine detail in regions of dense sampling, while smoothing out noise in regions of sparse sampling.

Using the fact that node functions at smaller depths correspond to wider smoothing filters, we define

$$\vec{V}(q) \equiv \sum_{s \in S} \frac{1}{W_{\hat{D}}(s.p)} \sum_{o \in \mathrm{Ngbr}_{\mathrm{Depth}(s.p)}(s)} \alpha_{o,s} F_o(q).$$

In this definition, $\mathrm{Depth}(s.p)$ represents the desired depth of a sample point $s \in S$. It is defined by computing the average sampling density W over all of the samples and setting:

$$\mathrm{Depth}(s.p) \equiv \min\left(D, D + \log_4(W_{\hat{D}}(s.p)/W)\right)$$

so that the width of the smoothing filter with which s contributes to $\vec{V}$ is proportional to the radius of its associated surface patch P_s.

Selecting an isovalue Finally, we modify the surface extraction step by selecting an isovalue which is the weighted average of the values of $\tilde{\chi}$ at the sample positions:

$$\partial\tilde{M} \equiv \{q \in \mathbb{R}^3 \mid \tilde{\chi}(q) = \gamma\} \quad \text{with} \quad \gamma = \frac{\sum \frac{1}{W_{\hat{D}}(s.p)} \tilde{\chi}(s.p)}{\sum \frac{1}{W_{\hat{D}}(s.p)}}.$$

5. Results

To evaluate our method we conducted a series of experiments. Our goal was to address three separate questions: How well does the algorithm reconstruct surfaces? How does it compare to other reconstruction methods? And, what are its performance characteristics?

Much practical motivation for surface reconstruction derives from 3D scanning, so we have focused our experiments on the reconstruction of 3D models from real-world data.

5.1. Resolution

We first consider the effects of the maximum octree depth on the reconstructed surface.

Figure 3 shows our reconstruction results for the "dragon" model at octree depths 6, 8, and 10. (In the context of reconstruction on a regular grid, this would correspond to resolutions of 64^3, 256^3, and 1024^3, respectively.) As the tree depth is increased, higher-resolution functions are used to fit the indicator function, and consequently the reconstructions capture finer detail. For example, the scales of the dragon, which are too fine to be captured at the coarsest resolution begin appearing and become more sharply pronounced as the octree depth is increased.

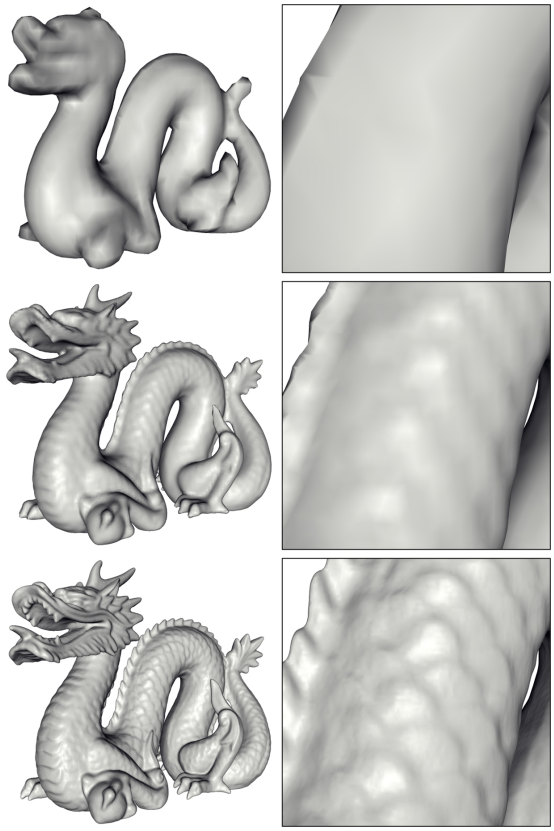


Figure 3: *Reconstructions of the dragon model at octree depths 6 (top), 8 (middle), and 10 (bottom).*

5.2. Comparison to Previous Work

We compare the results of our reconstruction algorithm to the results obtained using Power Crust [ACK01], Robust Cocone [DG04], Fast Radial Basis Functions (FastRBF) [CBC*01], Multi-Level Partition of Unity Implicits (MPU) [OBA*03], Surface Reconstruction from Unorganized Points [HDD*92], Volumetric Range Image Processing (VRIP) [CL96], and the FFT-based method of [Kaz05].

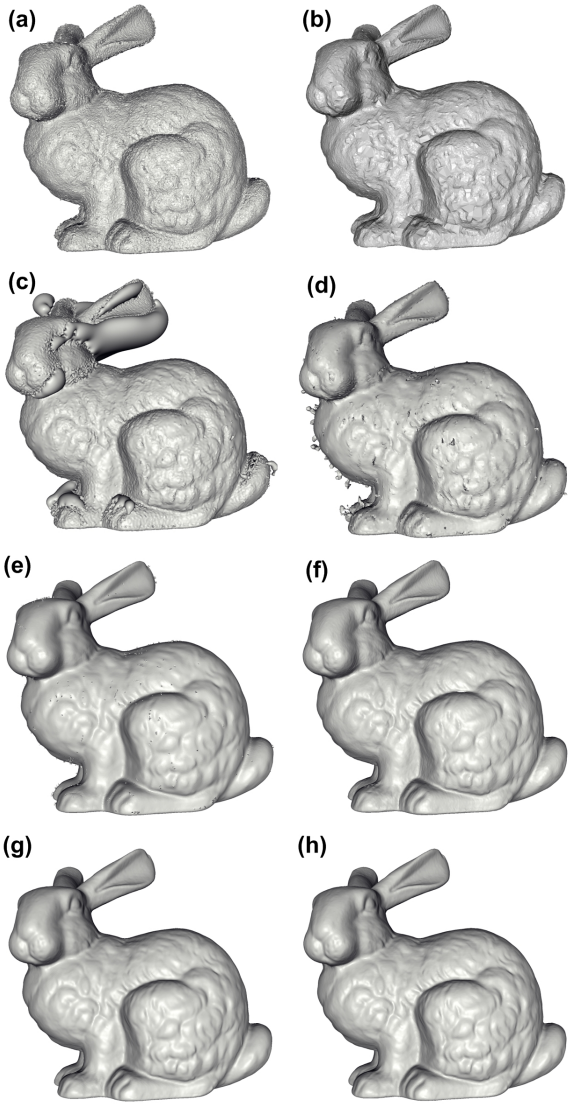


Figure 4: *Reconstructions of the Stanford bunny using Power Crust (a), Robust Cocone (b), Fast RBF (c), MPU (d), Hoppe* et al.*'s reconstruction (e), VRIP (f), FFT-based reconstruction (g), and our Poisson reconstruction (h).*

Our initial test case is the Stanford "bunny" raw dataset of 362,000 points assembled from ten range images. The data was processed to fit the input format of each algorithm. For example, when running our method, we estimated a sample's normal from the positions of the neighbors; Running VRIP, we used the registered scans as input, maintaining the regularity of the sampling, and providing the confidence values.

Figure 4 compares the different reconstructions. Since the scanned data contains noise, interpolatory methods such as Power Crust (a) and Robust Cocone (b) generate surfaces that are themselves noisy. Methods such as Fast RBF (c) and MPU (d), which only constrain the implicit function near the sample points, result in reconstructions with spurious surface sheets. Non-interpolatory methods, such as the approach of [HDD*92] (e), can smooth out the noise, although often at the cost of model detail. VRIP (f), the FFT-based approach (g), and the Poisson approach (h) all accurately reconstruct the surface of the bunny, even in the presence of noise, and we compare these three methods in more detail.

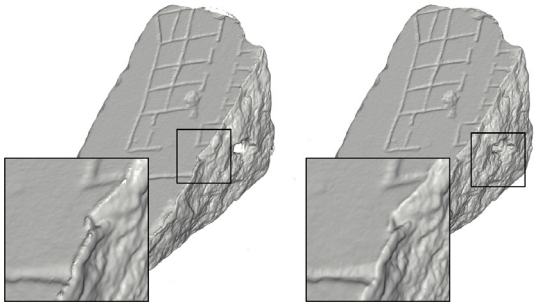


Figure 5: *Reconstructions of a fragment of the Forma Urbis Romae tablet using VRIP (left) and the Poisson solution (right).*

Comparison to VRIP A challenge in surface reconstruction is the recovery of sharp features. We compared our method to VRIP by evaluating the reconstruction of sample points obtained from fragment 661a of the Forma Urbis Romae (30 scans, 2,470,000 points) and the "Happy Buddha" model (48 scans, 2,468,000 points), shown in Figures 5 and 6. In both cases, we find that VRIP exhibits a "lipping" phenomenon at sharp creases. This is due to the fact that VRIP's distance function is grown perpendicular to the view direction, not the surface normal. In contrast, our Poisson reconstruction, which is independent of view direction, accurately reconstructs the corner of the fragment and the sharp creases in the Buddha's cloak.

Comparison to the FFT-based approach As Figure 4 demonstrates, our Poisson reconstruction (h) closely matches the one obtained with the FFT-based method (g). Since our method provides an adaptive solution to the same problem, the similarity is a confirmation that in adapting the octree to the data, our method does not discard salient, high-frequency information. We have also confirmed that our Poisson method maintains the high noise resilience already demonstrated in the results of [Kaz05].

Though theoretically equivalent in the context of uniformly sampled data, our use of adaptive-width filters (Section 4.5) gives better reconstructions than the FFT-based method on the non-uniform data commonly encountered in 3D scanning. For example, let us consider the region around the left eye of the "David" model, shown in Figure 7(a). The area above the eyelid (highlighted in red) is sparsely sampled due to the fact that it is in a concave region and is seen only by a few scans. Furthermore, the scans that do sample

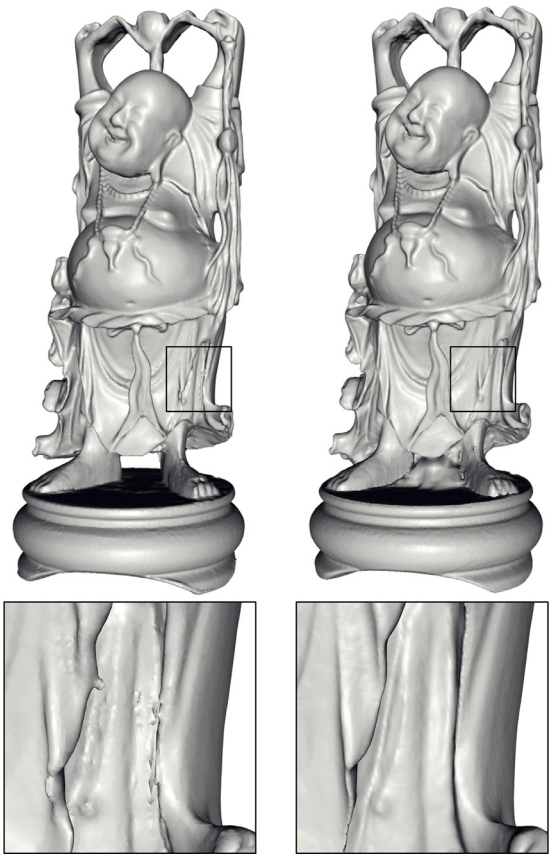


Figure 6: *Reconstructions of the "Happy Buddha" model using VRIP (left) and Poisson reconstruction (right).*

the region tend to sample at near-grazing angles resulting in noisy position and normal estimates. Consequently, fixed-resolution reconstruction schemes such as the FFT-based approach (b) introduce high-frequency noise in these regions. In contrast, our method (c), which adapts both the scale and the variance of the samples' contributions, fits a smoother reconstruction to these regions, without sacrificing fidelity in areas of dense sampling (e.g. the region highlighted in blue).

Limitation of our approach A limitation of our method is that it does not incorporate information associated with the acquisition modality. Figure 6 shows an example of this in the reconstruction at the base of the Buddha. Since there are no samples between the two feet, our method (right) connects the two regions. In contrast, the ability to use secondary information such as line of sight allows VRIP (left) to perform the space carving necessary to disconnect the two feet, resulting in a more accurate reconstruction.

5.3. Performance and Scalability

Table 1 summarizes the temporal and spatial efficiency of our algorithm on the "dragon" model, and indicates that the

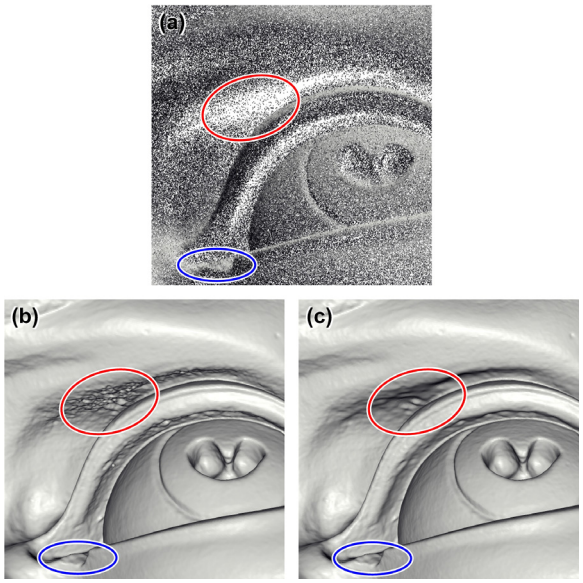


Figure 7: *Reconstruction of samples from the region around the left eye of the David model (a), using the fixed-resolution FFT approach (b), and Poisson reconstruction (c).*

memory and time requirements of our algorithm are roughly quadratic in the resolution. Thus, as we increase the octree depth by one, we find that the running time, the memory overhead, and the number of output triangles increases roughly by a factor of four.

Tree Depth	Time	Peak Memory	# of Tris.
7	6	19	21,000
8	26	75	90,244
9	126	155	374,868
10	633	699	1,516,806

Table 1: *The running time (in seconds), the peak memory usage (in megabytes), and the number of triangles in the reconstructed model for the different depth reconstructions of the dragon model. A kernel depth of 6 was used for density estimation.*

The running time and memory performance of our method in reconstructing the Stanford Bunny at a depth of 9 is compared to the performance of related methods in Table 2. Although in this experiment, our method is neither fastest nor most memory efficient, its quadratic nature makes it scalable to higher resolution reconstructions. As an example, Figure 8 shows a reconstruction of the head of Michelangelo's David at a depth of 11 from a set of 215,613,477 samples. The reconstruction was computed in 1.9 hours and 5.2GB of RAM, generating a 16,328,329 triangle model. Trying to compute an equivalent reconstruction with methods such as the FFT approach would require constructing two voxel grids at a resolution of 2048^3 and would require in excess of 100GB of memory.

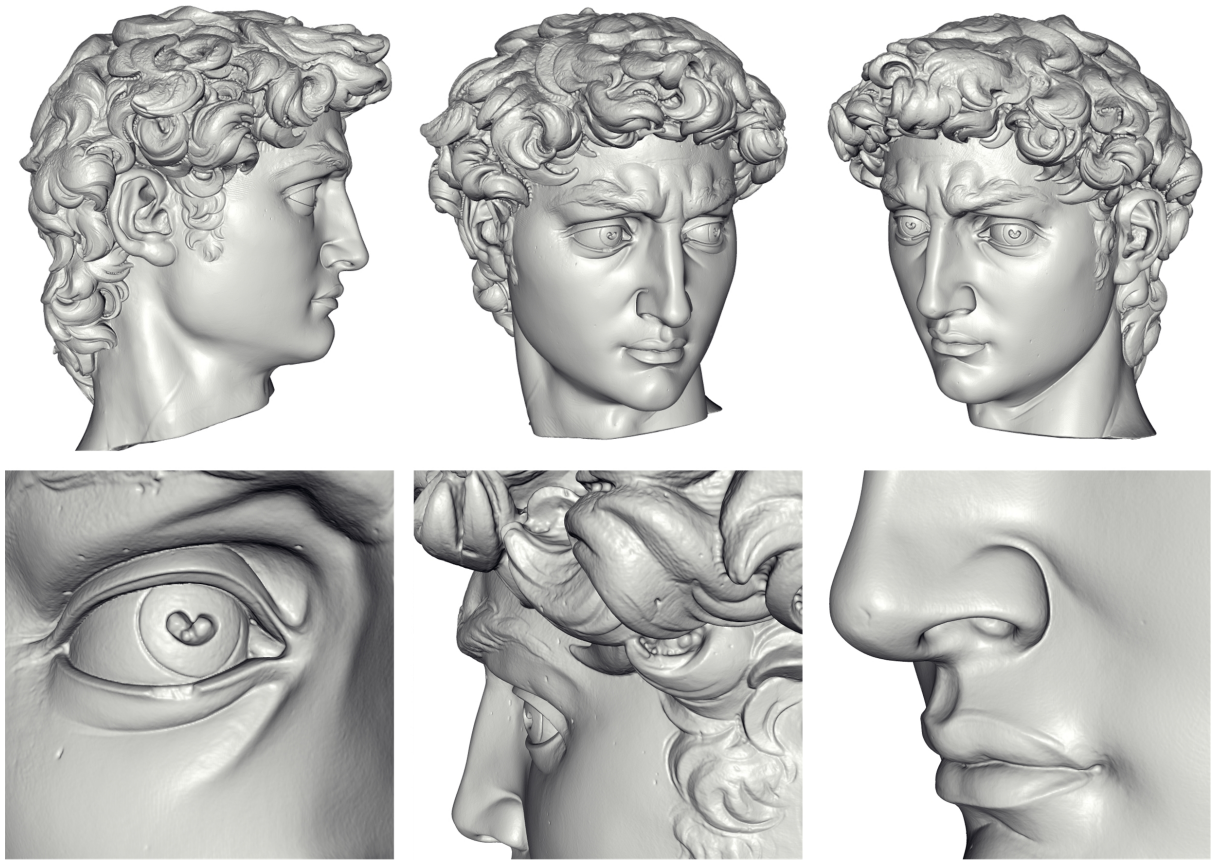


Figure 8: *Several images of the reconstruction of the head of Michelangelo's David, obtained running our algorithm with a maximum tree depth of 11. The ability to reconstruct the head at such a high resolution allows us to make out the fine features in the model such as the inset iris, the drill marks in the hair, the chip on the eyelid, and the creases around the nose and mouth.*

Method	Time	Peak Memory	# of Tris.
Power Crust	380	2653	554,332
Robust Cocone	892	544	272,662
FastRBF	4919	796	1,798,154
MPU	28	260	925,240
Hoppe et al 1992	70	330	950,562
VRIP	86	186	1,038,055
FFT	125	1684	910,320
Poisson	263	310	911,390

Table 2: *The running time (in seconds), the peak memory usage (in megabytes), and the number of triangles in the reconstructed surface of the Stanford Bunny generated by the different methods.*

6. Conclusion

We have shown that surface reconstruction can be expressed as a Poisson problem, which seeks the indicator function that best agrees with a set of noisy, non-uniform observations, and we have demonstrated that this approach can robustly recover fine detail from noisy real-world scans.

There are several avenues for future work:

- Extend the approach to exploit sample confidence values.
- Incorporate line-of-sight information from the scanning process into the solution process.
- Extend the system to allow out-of-core processing for huge datasets.

Acknowledgements

The authors would like to express their thanks to the Stanford 3D Scanning Repository for their generosity in distributing their 3D models. The authors would also like to express particular gratitude to Szymon Rusinkiewicz and Benedict Brown for sharing valuable experiences and ideas, and for providing non-rigid body aligned David data.

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C.: Point set surfaces. In *Proc. of the Conference on Visualization '01* (2001), 21–28.

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new Voronoi-based surface reconstruction algorithm. *Computer Graphics (SIGGRAPH '98)* (1998), 415–21.

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications 19* (2001), 127–153.

[BBX95] BAJAJ C., BERNARDINI F., XU G.: Automatic reconstruction of surfaces and scalar fields from 3d scans. In *SIGGRAPH* (1995), 109–18.

[BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖDER P.: Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *TOG 22* (2003), 917–924.

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE TVCG 5* (1999), 349–359.

[Boi84] BOISSONNAT J.: Geometric structures for three dimensional shape representation. *TOG* (1984), 266–286.

[CBC*01] CARR J., BEATSON R., CHERRIE H., MITCHEL T., FRIGHT W., MCCALLUM B., EVANS T.: Reconstruction and representation of 3D objects with radial basis functions. *SIGGRAPH* (2001), 67–76.

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. *Computer Graphics (SIGGRAPH '96)* (1996), 303–312.

[DG04] DEY T., GOSWAMI S.: Provable surface reconstruction from noisy samples. In *Proc. of the Ann. Symp. Comp. Geom.* (2004), 428–438.

[DMGL02] DAVIS J., MARSCHNER S., GARR M., LEVOY M.: Filling holes in complex surfaces using volumetric diffusion. In *Int. Symp. 3DPVT* (2002), 428–438.

[EM94] EDELSBRUNNER H., MÜCKE E.: Three-dimensional alpha shapes. *TOG* (1994), 43–72.

[FLW02] FATTAL R., LISCHINKSI D., WERMAN M.: Gradient domain high dynamic range compression. In *SIGGRAPH* (2002), 249–256.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: a simple framework for adaptive simulation. In *SIGGRAPH* (2002), 281–290.

[GWL*03] GOODNIGHT N., WOOLLEY C., LEWIN G., LUEBKE D., HUMPHREYS G.: A multigrid solver for boundary value problems using programmable graphics hardware. In *Graphics Hardware* (2003), 102–111.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *Computer Graphics 26* (1992), 71–78.

[Kaz05] KAZHDAN M.: Reconstruction of solid models from oriented point sets. *SGP* (2005), 73–82.

[KSO04] KOLLURI R., SHEWCHUK J., O'BRIEN J.: Spectral surface reconstruction from noisy point clouds. In *SGP* (2004), 11–21.

[LC87] LORENSEN W., CLINE H.: Marching cubes: A high resolution 3d surface reconstruction algorithm. *SIGGRAPH* (1987), 163–169.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *TOG (SIGGRAPH '04) 23* (2004), 457–462.

[Mur91] MURAKI S.: Volumetric shape description of range data using "blobby model". *Computer Graphics 25* (1991), 227–235.

[NRDR05] NEHAB D., RUSINKIEWICZ S., DAVIS J., RAMAMOORTHI R.: Efficiently combining positions and normals for precise 3D geometry. *TOG (SIGGRAPH '05) 24* (2005).

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.: Multi-level partition of unity implicits. *TOG* (2003), 463–470.

[Par62] PARZEN E.: On estimation of a probability density function and mode. *Ann. Math Stat. 33* (1962), 1065–1076.

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *TOG (SIGGRAPH '03) 22* (2003), 313–318.

[SFYC96] SHEKHAR R., FAYYAD E., YAGEL R., CORNHILL J.: Octree-based decimation of marching cubes surfaces. In *IEEE Visualization* (1996), 335–342.

[SOS04] SHEN C., O'BRIEN J., SHEWCHUK J.: Interpolating and approximating implicit surfaces from polygon soup. *TOG (SIGGRAPH '04) 23* (2004), 896–904.

[TO02] TURK G., O'BRIEN J.: Modelling with implicit surfaces that interpolate. In *TOG* (2002), 855–873.

[WG92] WILHELMS J., GELDER A. V.: Octrees for faster isosurface generation. *TOG 11* (1992), 201–227.

[WKE99] WESTERMANN R., KOBBELT L., ERTL T.: Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer 15* (1999), 100–111.

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.: Mesh editing with Poisson-based gradient field manipulation. *TOG (SIGGRAPH '04) 23* (2004), 641–648.

Appendix A:

The solution to surface reconstruction described in this paper approaches the problem in a manner similar to the solution of [Kaz05] in that the reconstructed surface is obtained by first computing the indicator function and then extracting the appropriate isosurface.

While the two methods seem to approach the problem of computing the indicator function in different manners ([Kaz05] uses Stokes' Theorem to define the Fourier coefficients of the indicator function while we use the Poisson equation), the two methods are in fact equivalent.

To show this, we use the fact that the Poisson equation $\Delta u = f$ where f is periodic can be solved using the Fourier transform. The Fourier series expansion is $-|\zeta|^2 \hat{u}(\zeta) = \hat{f}(\zeta)$, or equivalently $\hat{u}(\zeta) = \frac{-1}{|\zeta|^2}\hat{f}(\zeta)$.

Thus, our Poisson equation $\Delta\chi = \nabla \cdot \vec{V}$ can be solved using $\hat{\chi} = \frac{-1}{|\zeta|^2}\widehat{\nabla \cdot \vec{V}}$. With the well known identity $\hat{f}' = -i\zeta\hat{f}$ and its generalization $\widehat{\nabla \cdot \vec{V}} = -i\zeta \cdot \hat{\vec{V}}$, we get $\hat{\chi} = \frac{i}{|\zeta|^2}\zeta \cdot \hat{\vec{V}}$, which is *identical* to [Kaz05].

PolyFit: Polygonal Surface Reconstruction from Point Clouds

Liangliang Nan
Visual Computing Center, KAUST
liangliang.nan@gmail.com

Peter Wonka
Visual Computing Center, KAUST
pwonka@gmail.com

Abstract

We propose a novel framework for reconstructing lightweight polygonal surfaces from point clouds[1]*. Unlike traditional methods that focus on either extracting good geometric primitives or obtaining proper arrangements of primitives, the emphasis of this work lies in intersecting the primitives (planes only) and seeking for an appropriate combination of them to obtain a manifold polygonal surface model without boundary.*

We show that reconstruction from point clouds can be cast as a binary labeling problem. Our method is based on a hypothesizing and selection strategy. We first generate a reasonably large set of face candidates by intersecting the extracted planar primitives. Then an optimal subset of the candidate faces is selected through optimization. Our optimization is based on a binary linear programming formulation under hard constraints that enforce the final polygonal surface model to be manifold and watertight. Experiments on point clouds from various sources demonstrate that our method can generate lightweight polygonal surface models of arbitrary piecewise planar objects. Besides, our method is capable of recovering sharp features and is robust to noise, outliers, and missing data.

1. Introduction

Reconstructing 3D models from sampled points has been a major problem in both computer vision and computer graphics. Although it has been extensively researched in the past few decades [8, 13, 22, 11, 3, 17, 27, 25, 12, 2, 15, 20, 26, 16], obtaining faithful reconstructions of real-world objects from unavoidable noisy and possibly incomplete point clouds remains an open problem.

In this work, we focus on reconstructing piecewise planar objects (i.e., man-made objects such as buildings). Our inputs are point clouds sampled from real-world objects that could be captured by various means (e.g., drones, handheld scanners, and depth cameras). Our goal is to achieve lightweight polygonal surface models of the objects.

We consider a method to be effective to this problem if it meets the following requirements. First, the reconstructed model should be an oriented 2-manifold and watertight, ensuring physically valid models. This is essential for many applications, e.g., simulation and fabrication. Second, it should be able to recover sharp features of the objects. Third, the method should not closely follow surface details due to imperfections (i.e., noise, outliers, and missing data). Instead, a lightweight representation is more preferred and beneficial for many applications (e.g., Augmented Reality/Virtual Reality) that may involve many objects and large scenes. Besides, it is also helpful to provide a way to control the detail levels of the reconstructed models within the reconstruction pipeline. We proposed a novel reconstruction framework that all the above requirements are satisfied in a single optimization process.

Instead of fitting dense smooth polygonal surfaces [8, 13, 22, 11], we seek for a more compact representation (i.e., simple polygonal surfaces) for piecewise planar objects. Our method is based on a *hypothesizing and selection* strategy. Specifically, we chose an optimal set of faces from a large number of candidate faces to assemble a compact polygonal surface model. The optimal set of faces is selected through optimization that encourages good point support and compactness of the final model. The manifold and watertight requirements are enforced as hard constraints. Figure 1 shows an example of our reconstruction. Our work makes the following contributions:

- we cast polygonal surface reconstruction from point clouds as a binary labeling problem based on a *hypothesizing and selection* strategy.
- a surface reconstruction framework that is dedicated to reconstructing piecewise planar objects.
- a binary linear programming formulation for face selection that guarantees lightweight, manifold, and watertight reconstructions and meanwhile recovers sharp features of the objects.

[1]The source code of PolyFit is available under https://github.com/LiangliangNan/PolyFit.

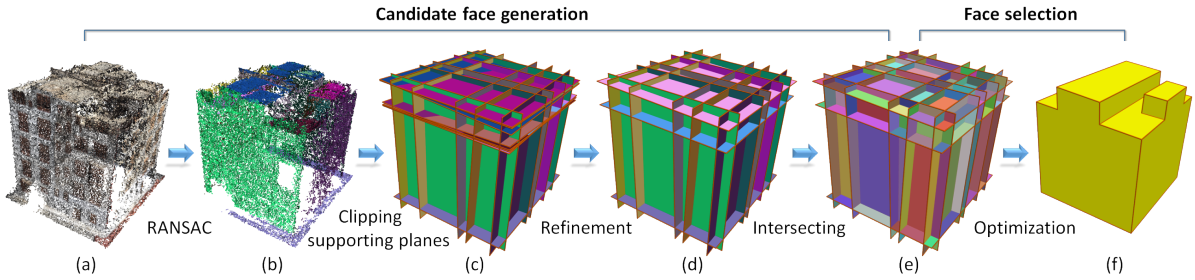


Figure 1. Pipeline. (a) Input point cloud. (b) Planar segments. (c) Supporting planes of the initial planar segments. (d) Supporting planes of the refined planar segments. (e) Candidate faces. (f) Reconstructed model. All planar segments and faces are randomly colored.

2. Related Work

In literature, a large body of research on polygonal surface reconstruction from point clouds aim at obtaining dense polygonal surfaces [8, 13, 22, 11]. In the last decades, extracting geometric primitives and identifying their combination to infer higher-level structures have become the most popular technique for reconstructing piecewise planar objects. In this section, we mainly review approaches that focus on primitive extraction, primitive regularization, and primitive- and hypothesis-based reconstruction.

Primitive extraction. Researches falling in this category aim at extracting high-quality instances of basic geometric primitives (e.g., plane, cylinder) from point clouds corrupted by noise and outliers. The common practice for this particular task is the Random Sample Consensus (RANSAC) algorithm [6] and its variants [28, 24, 14]. These methods are reliable when the input data is contaminated by noise and outliers. So in our work, we use an efficient implementation of the RANSAC algorithm [24] to extract initial planar primitives.

Primitive regularization. By exploiting the prior knowledge about the structure of an object, researchers further regularize the extracted primitives. Li et.al. [17] discover global mutual relations between basic primitives and use such information as constraints to refine the initial primitives base on local fitting and constrained optimization. Monszpart et. al. [19] further exploit this idea to extract regular arrangements of planes. These methods focus on inferring and regularizing potential relationship between structures. However, obtaining manifold and watertight surface models from the regularized primitives remains unsolved. In our work, we provide a promising framework to this challenging problem based on a *hypothesizing and selection* strategy.

Primitive-based reconstruction. In contrast with approaches that focus on obtaining dense polygonal surfaces, exploiting high-level primitives for man-made objects became popular in the last years [25, 12, 15, 20, 26, 16]. Arikan et al. [1] proposed an optimization-based interactive tool that can reconstruct architectural models from a sparse point cloud. Lin et al. [18] reconstruct coarse building models by decomposing and fitting a set of piecewise building blocks to the point clouds. By structuring and resampling planar primitives, Lafarge and Alliez [12] consolidate the point clouds and then obtain surface models by solving a graph-cut problem. Using a min-cut formulation, Verdie et. al. [26] reconstruct watertight buildings from an initial arrangement of planar segments. These approaches specialize in reconstructing urban buildings. Thus, they may not be suitable to handle general piecewise planar objects. Based on space partitioning and volumetric presentations, [3] and [2] obtain piecewise planar reconstruction by determining if cells are occupied or not. These two techniques require visibility information from the acquisition process as input while our approach does not require this information.

Hypothesis-based reconstruction. By making stronger assumptions on the objects, researchers further regularize the reconstruction problem and fit compound shapes (i.e., a combination of multiple basic primitives) to the point clouds [9, 16]. In Li et.al. [16], a set of axis-aligned boxes is assembled to approximate the geometry of a building. Their strategy is generating a non-uniform grid and then selecting a subset of its cells that have good data support and are smooth. In our work, we generalize this idea to reconstruct general piecewise planar objects, and our reconstruction is based on optimization under hard constraints that guarantee manifold and watertight polygonal surface models.

3. Overview

Our method takes as input a point cloud (possibly noisy, incomplete, and with outliers) of a piecewise planar object and outputs a lightweight polygonal surface model. Our method consists of two main parts (see Figure 1):

Candidate face generation. We first extract a set of planar segments from the point cloud using RANSAC [24]. Considering the detected planar segments may contain undesired elements due to noise, outliers, and missing data, we refine these planar segments by iteratively merging plane pairs and fitting new planes. We then clip these

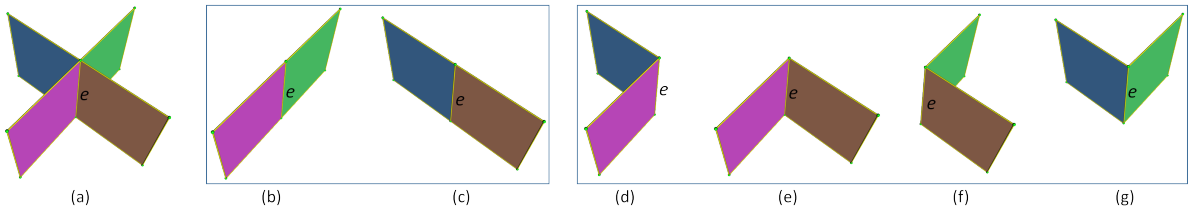


Figure 2. Two planes intersect with each other resulting in four parts (a). (b) to (g) show all possible combinations to ensure a 2-manifold surface. The edge in (b) and (c) connects two co-planar parts, while in (d) to (g) it introduces sharp edges in the final model.

planes by an enlarged bounding box of the point cloud and compute pairwise intersections of the clipped planes to hypothesize of the object's faces.

Face selection. We choose an optimal subset of the candidate faces to assemble a manifold and watertight polygonal surface model. To do so, we formulate the face selection as a binary linear programming problem. Our objective function combines three terms that favor data fitting, point coverage, and model complexity, respectively. We also formulate hard constraints that ensure the final model is manifold and watertight.

4. Candidate Face Generation

The input to this step is a point cloud P of a piecewise planar object, and the goal is to generate a reasonably large number of candidate faces.

Plane extraction. We use the RANSAC-based primitive detection method proposed by Schnabel et al. [24] to detect a set of initial planar segments $S = \{s_i\}$ from the point cloud P. Here s_i is a set of points whose distances are smaller than a threshold ε to a plane and each points can be assigned to no more than one plane. This plane is denoted as the *supporting plane* of s_i.

Plane refinement. Due to the presence of noise and outliers (especially for point clouds computed from Multi-view Stereo), RANSAC may produce a few undesired planar segments. We observed that the undesired planar segments usually have arbitrary orientations and poor point support. Although the later optimization-based face selection procedure is designed to be capable of handling such outliers, there may still cause problems. First, these arbitrarily oriented planes may result in long but very thin faces and small bumps in the final model. In some extreme cases, it may also introduce degenerate faces due to the limit of floating point precision. This degeneracy usually makes the *manifold* and *watertight* hard constraints (see Section 5.2) impossible to be satisfied. Second, the undesired candidate faces results in a larger optimization problem that is more expensive to be solved.

To tackle this issue, we iteratively refine the initial planar segments using an improved plane refinement algorithm proposed in [15]. Specifically, we first compute the angle of the supporting planes for each pair of planar segments. Then, starting from the pair (s_i, s_j) with the smallest angle, we test if the following two conditions are met. First, the angle between the two planes is lower than a threshold, i.e., $angle(s_i, s_j) < \theta_t$. Second, more than a specified number (denoted as N_t) of points lie on the supporting planes of both segments. If both conditions are satisfied, we merge the two planar segments and fit a new supporting plane using PCA [10]. We iterate this process until no more segment pair can be merged. In our experiments, we choose $\theta_t = 10°$ and $N_t = min(|s_i|, |s_j|)/5$, where $|s_i|$ denotes the number of supporting points of s_i. Figure 1 (c) and (d) show the extracted planes before and after refinement respectively. It should be noted that an alternative approach, e.g., [17], can also be used to extract planar segments.

Pairwise intersecting. To hypothesize the object's faces, we crop the supporting planes of all planar segments by the bounding box of the point cloud. Then, the candidate faces (i.e., a superset of the faces of the object) can be obtained by intersecting the cropped planes. For simplicity, we compute pairwise intersections of the cropped planes (see Figure 1 (e)). It should be noted that pairwise intersections introduce redundant candidate faces. Since most of the redundant faces do not represent actual structures of an object, they are supported by no or very few (due to noise and outliers) point samples. The subsequent optimization-based face selection is designed to favor choosing the most confident faces while satisfying certain constraints.

The pairwise intersections maintain incidence information of the faces and edges. Each edge of a candidate face is either connecting four neighboring candidate faces or representing a boundary. For example in Figure 2 (a), edge e connects four faces while others are boundaries. We rely on such incidence information to formulate our *manifold* and *watertight* constraints for face selection (see Section 5.2).

5. Face Selection

Given N candidate faces $F = \{f_i | 1 \leq i \leq N\}$ generated in the previous step, we select a subset of these candidate faces that can best describe the geometry of the object and ensure that the chosen faces form a manifold and watertight polygonal surface. This is achieved through op-

timization. We define multiple energy terms that constitute our objective function.

5.1. Energy terms

Let variable x_i encode if a candidate face f_i is chosen (i.e., $x_i = 1$) or not chosen (i.e., $x_i = 0$), our objective function consists of three energy terms: data-fitting, model complexity, and point coverage.

Data-fitting. This term is intended to evaluate the fitting quality of the faces to the point cloud while accounting for an appropriate notion of confidence [21]. It is defined to measure a confidence-weighted percentage of points that do not contribute to the final reconstruction

$$E_f = 1 - \frac{1}{|P|}\sum_{i=1}^{N} x_i \cdot support(f_i), \tag{1}$$

where $|P|$ is the total number of points in P. $support(f_i)$ accounts for a notion of confidence and is defined at each point considering its local neighborhood

$$support(f) = \sum_{p,f|dist(p,f)<\varepsilon} (1 - \frac{dist(p,f)}{\varepsilon}) \cdot conf(p), \tag{2}$$

where $dist(p, f)$ measures the Euclidean distance between a point p and a face f. We consider only points that are ε-close to f, i.e., points p satisfying $dist(p, f) < \varepsilon$. The confidence term $conf(p)$ measures the local quality of the point cloud at a point p. It is computed by examining the local covariance matrices defined at p, as in [23]. In this work, we compute for p the covariance matrices of its local neighbors at three static scales (i.e., different neighborhood sizes). Then, $conf(p)$ is defined as

$$conf(p) = \frac{1}{3}\sum_{i=1}^{3}(1 - \frac{3\lambda_i^1}{\lambda_i^1 + \lambda_i^2 + \lambda_i^3}) \cdot \frac{\lambda_i^2}{\lambda_i^3}, \tag{3}$$

where $\lambda_i^1 \leq \lambda_i^2 \leq \lambda_i^3$ are the three eigenvalues of the covariance matrix at scale i. $conf(p)$ encode two geometric properties of the point samples near point p. The first property $1 - 3\lambda^1/(\lambda^1 + \lambda^2 + \lambda^3)$ evaluates the quality of fitting a local tangent plane at p, whose value of 0 indicates the worst point distribution and value of 1 indicates a perfectly fitted plane. The second property λ^2/λ^3 evaluates the local sampling uniformity. Each eigenvalue ratio takes on a value in the range $[0, 1]$ with boundary values 0 and 1 corresponding to a perfect line distribution and a uniform disc distribution correspondingly.

Intuitively, the *data-fitting* term favors choosing faces that are close to the input points and are supported by densely sampled uniform regions. This term has a value in the range [0, 1] where a value of 1 indicates a noise-free and outlier-free input data.

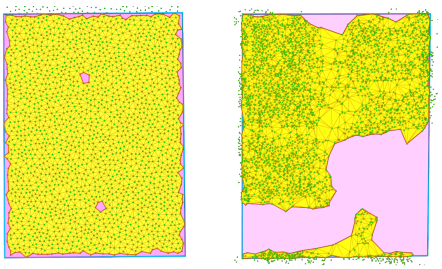


Figure 3. Two examples of point coverage. The α-shape meshes are in yellow and the candidate faces are in purple. The value below each figure indicates the coverage ratio of each face.

Model complexity. Given incomplete point clouds due to occlusions, the *data-fitting* term defined in Equation 1 tends to stubbornly comply with the incomplete data, resulting in gaps in the final model. Besides, noise and outliers also tend to introduce gaps and protrusions in the reconstructed models. To avoid these defects, we introduce a *model complexity* term to encourage simple structures (i.e., large planar regions). Considering gaps and protrusions result in additional sharp edges, we define the *model complexity* term as the ratio of sharp edges in the model

$$E_m = \frac{1}{|E|}\sum_{i=1}^{|E|} corner(e_i), \tag{4}$$

where $|E|$ denotes the total number of pairwise intersections in the candidate face set. $corner(e_i)$ is an indicator function whose value is determined by the configuration of the two selected faces connected by an edge e_i. The intersecting edges can be either planar or sharp. For example, the intersecting edges e in Figure 2 (b) and (c) are planar edges yielding larger planar polygons, but edges in (d) to (g) are sharp edges that will introduce sharp features (if they are selected in the final model). So $corner(e_i)$ will have a value of 1 if the faces associated with e_i introduce a sharp edge in the final model. Otherwise, $corner(e_i)$ has a zero value meaning the two faces are coplanar.

Point coverage. To handle missing data caused by occlusions, the unsupported regions (i.e., regions not covered by points) of the final model should be as small as possible. To measure the coverage of a face f_i, we first project all ε-close points onto f_i. We then extract a mesh M_i^α by constructing a 2D α-shape [5] from the projected points (see Figure 3). We use only the points whose projections are inside f_i for α-shape construction. We call M_i^α an α-shape mesh. Intuitively, the α-shape mesh of a set of points ensures that any three points with a circumradius $r_c \leq \sqrt{\alpha}$ are spanned by a triangle face. Given an appropriate value of r_c, the area of the α-shape mesh surface can provide us a reliable measure of the coverage of a candidate face by the

input points. Thus, our *point coverage* energy is defined as the ratio of the uncovered regions in the model

$$E_c = \frac{1}{area(M)} \sum_{i=1}^{N} x_i \cdot (area(f_i) - area(M_i^\alpha)), \quad (5)$$

where $area(M)$, $area(f_i)$, and $area(M_i^\alpha)$ denote the surface areas of the final model, a candidate face f_i, and the α-shape mesh M_i^α of f_i, respectively. In our implementation, we chose the radius r_c to be $5 \cdot density(P)$, where $density(P)$ denotes the average spacing of all the points to their k nearest neighbors, k being set to 6.

Using the exact model area, i.e., $\sum_{i=1}^{N} x_i \cdot area(f_i)$, as the denominator ensures that the value of the *point coverage* term is within the range [0, 1]. However, this results in a non-linear objective function that is difficult to optimize. Considering the actual surface area of the final model is comparable to the area of the point cloud's bounding box, we replace the exact model area with the area of the point cloud's bounding box, i.e., $area(M) \approx area(bbox(P))$.

5.2. Optimization

With the above energy terms, the optimal set of faces can be obtained by minimizing a weighted sum of these terms under certain hard constraints enforcing the final model to be manifold without boundary.

Remember that the candidate faces are obtained by the pairwise intersection of planes. Thus an edge is connected to either one (for boundary faces) or four faces (for inner faces). See Figure 2 (a) for an example. It is quite obvious that the necessary and sufficient condition for manifold and watertight polygonal surfaces is that each edge of the model connects only two adjacent faces. Thus, the final formulation for face selection can be written as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \lambda_f \cdot E_f + \lambda_m \cdot E_m + \lambda_c \cdot E_c \\ \text{s.t.} \quad & \begin{cases} \sum_{j \in \mathcal{N}(e_i)} x_j = 2 \quad \text{or} \quad 0, & 1 \le i \le |E| \\ x_i \in \{0, 1\}, & 1 \le i \le N \end{cases} \end{aligned} \quad (6)$$

where $\sum_{j \in \mathcal{N}(e_i)} x_j$ counts the number of faces connected by an edge e_i. This value is enforced to be either 0 or 2, meaning none or two of the faces are selected (see Figure 2 (b) - (g) for possible selections). These hard constraints guarantee that the final model is manifold and closed.

The problem defined in Equation 6 is a binary linear program. We solve it using the Gurobi solver [7]. After optimization, the union of the selected faces (i.e., faces having a corresponding variable $x_i = 1$) comprises a polygonal surface model approximating the object.

6. Results and Discussion

We implemented our method using C++. The α-shapes are constructed using the CGAL library [5]. We tested our method on various real-world objects of different complexity. Experiments demonstrated the advantages of our *hypothesizing and selection* strategy.

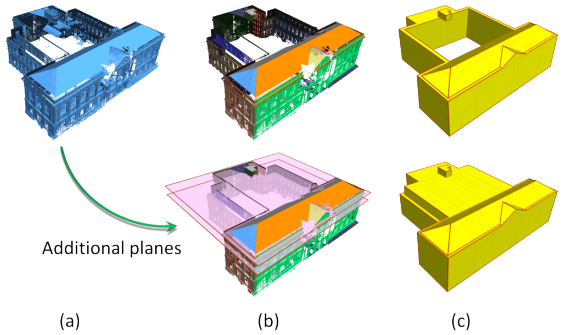


Figure 5. Reconstruction of a building with completely missing planes (top row). By adding an extra plane, model with different structures can be obtained (bottom row).

Reconstruction results. Our method can reconstruct piecewise planar objects such as buildings and other man-made objects. Figures 1 and 4 (a) - (d) show the reconstruction of five buildings of different styles. The input of these buildings are point clouds computed from images using Multi-View Stereo techniques. Although the point clouds are quite noisy and contain significant amounts of outliers and missing data, our method faithfully reconstructed these buildings. In Figures 4 (e) and (f), two indoor scenes consisting of multiple rooms are reconstructed. These two scenes were captured by Google Tango tablets. Due to the cluttered nature of indoor scenes, the datasets contains large holes. Our *hypothesizing and selection* strategy fills in the missing regions and reconstructed the permanent structures (i.e., walls and roofs) of these scenes.

Besides the buildings, we also tested our method on other man-made objects. In Figure 4 (g), our method took the laser scan of a small packing foam box as input and faithfully reconstructed all its structures with sharp features. We consider this as the *first* advantage of our approach. In (h) and (i), two sofas of different styles are reconstructed. These pieces of furniture were scanned by PrimeSense RGB-D cameras in the form of dense triangular meshes [4]. Our method took the vertices of the meshes as input and produced lightweight polygonal surface models.

In rare cases where few planes are completely missing, our method may still generate plausible reconstructions. In Figure 5 (top), a building with its back roof completely missing is reconstructed. It is interesting to see that by adding few extra planes, models with distinct structures can also be obtained (bottom row). Thanks to the manifold and watertight constraints, our method guarantees the reconstructed models to be manifold without boundary. We consider this as the *second* advantage of our approach.

In Table 1, we give statistics of our quantitative results.

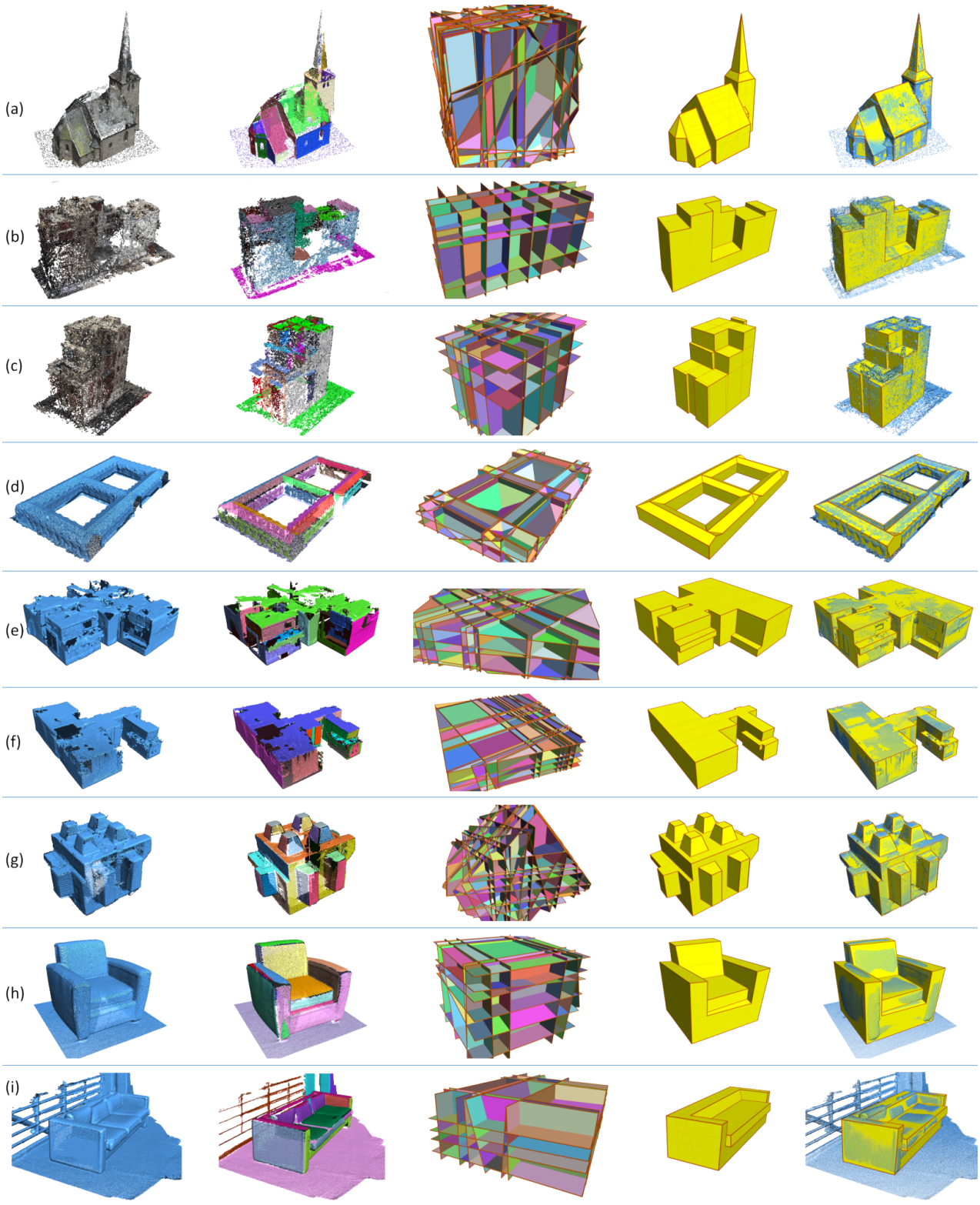


Figure 4. Reconstruction of a set of piecewise planar objects from various data sources. The input for (a) - (d) are computed from images using Multi-View Stereo; (e) and (f) are acquired by Google Tango tablets; (g) is captured by a laser scanner [17]; (h) and (i) are acquired by PrimeSense Carmine RGB-D cameras [4]. From left to right: input point clouds, extracted planar segments, candidate faces (randomly colored), reconstructed models, and models overlaid with the original point clouds (rendered with a smaller point size).

Table 1. Statistics on the examples presented in Figure 4.

Model index in Figure 4	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
# points	129 K	101 K	73 K	577 K	186 K	176 K	382 K	756 K	911 K
# planar segments	36	21	22	22	28	33	52	17	19
# candidate faces	6239	472	946	959	1778	2770	7540	580	523
# faces of the final model	38	18	25	29	26	35	52	16	14
Primitive extraction (sec)	0.21	0.14	0.09	1.17	0.45	0.13	0.93	1.04	1.54
Candidate generation (sec)	0.05	0.01	0.01	0.02	0.01	0.02	0.06	0.02	0.03
Face selection (optimization) (sec)	2.73	1.46	1.17	8.26	2.90	3.05	7.39	13.26	16.01

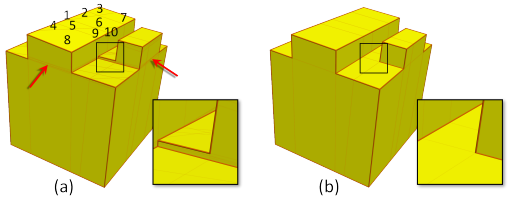


Figure 6. Result without (a) and with (b) plane refinement step. In (a), the top comprises ten faces originating from three different planes. The red arrows indicate long but very thin polygonal faces.

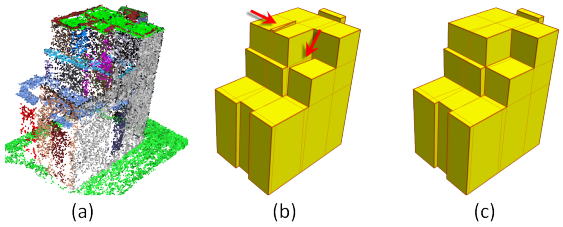


Figure 7. A building (a) reconstructed without (b) and with (c) the complexity term. The red arrows indicate bumps and gaps.

As can be seen from the number of faces in the final models, our method can produce lightweight 3D models. We consider this as the *third* main advantage of our approach.

Timings. Table 1 shows the running times of each step for the examples presented in Figure 4. We can see that the face selection step is slower compared to primitive extraction and candidate face generation. It should be noted that the construction of the α-shape meshes dominated this step. Down-sampling of input point clouds may speed up this process.

Effect of plane refinement. We tested our algorithm on a few data sets by omitting the refinement step. One such example is shown in Figure 6. We can see that even without the plane refinement step, our optimization still recovers the main structure of this building. However, we observe that the two models have some differences in the details. First, the top-most face of the reconstructed polyhedron in (a) is composed of ten candidate faces lying on three different planes, while in (b) it comprises fewer faces originating from the same plane. Second, some undesired bumpy structures can be avoided with the plane refinement step. This is because some initial faces are quite close to each other, making the energy terms less distinguishable. Another benefit of the plane refinement step is the gain in efficiency. With plane refinement, the total number of candidate faces is reduced from 1185 to 348. Accordingly, the run-time of the optimization decreased from 1.25 seconds to 0.31 seconds, i.e., more than four times faster.

Effect of the energy terms. The core of our reconstruction method lies in the optimization-based face selection that is designed to favor different aspects necessary for high-quality reconstructions. Both data fitting and coverage are essential requirements. We observed that having only these two terms works perfectly for reasonably complete datasets, but it is still not sufficient for point clouds with significant imperfections (i.e., noise, outliers, and missing data). Figure 7 (b) shows such an example. We can see that the reconstructed model demonstrates some desired bumps and gaps. In contrast, with our model complexity term, the reconstructed model is cleaner and compact (c). To further evaluate the behavior of the model complexity term, we ran our face selection on a dataset by gradually increasing the weight (see Figure 9). Not surprisingly, increasing the influence of the model complexity term resulted in less detailed 3D models. Thus, the model complexity term also provides control over the model details.

Parameters. The parameters for most examples are as follows: $\lambda_f = 0.46$, $\lambda_c = 0.27$, and $\lambda_m = 0.27$ (Note that weights in a wide range can produce the same results). Slightly different weights ($\lambda_f = 0.3$, $\lambda_c = 0.4$, and $\lambda_m = 0.3$) are used for the sofa example in Figure 4 (i), where the background (ground plane) has a much higher density than the object (sofa), thus the smaller data fitting weight.

Robustness and accuracy. We evaluated the robustness of our approach by reconstructing from synthetic data with an increasing amount of noise (see Figure 10). In (c), the standard deviation of the noise distribution is 0.6 meters high, our method still obtained topologically accurate reconstruction. However, a much higher level

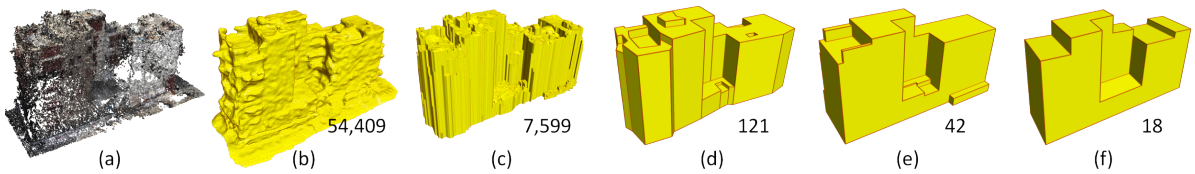


Figure 8. Comparison with four state-of-the-art methods on a building dataset. (a) Input point cloud. (b) Model reconstructed by the Poisson surface reconstruction algorithm [11]. (c) The result of the 2.5D Dual Contouring approach [27]. (d) The result of [15]. (e) The result of [16]. (f) Our result. The number under each sub-figure indicates the total number of faces in the corresponding model.

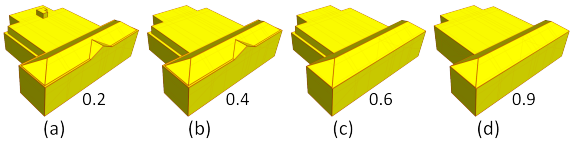


Figure 9. Reconstruction of the building shown in Figure 5 by gradually increasing the influence of the model complexity term. The values are the weights used in the corresponding optimization.

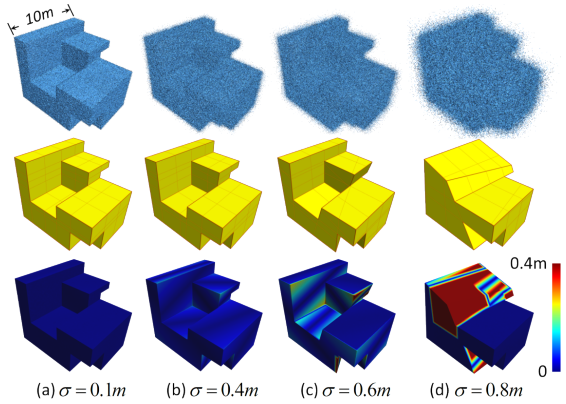


Figure 10. Reconstruction from synthetic data with increasing noise. Top: input. Middle: reconstruction. Bottom: reconstruction error. σ indicates the standard deviation of the Gaussian noise.

of noise may further pollute the structures of the object, resulting in large errors in the final model.

Comparison. Figure 8 shows the comparison of our approach with four other methods on a building dataset. We can see that both Poisson [11] and 2.5D Dual Contouring [27] generate dense surfaces with large numbers of bumps. The method in [15] uses only the roof information and also produce a 2.5D reconstruction. By making the Manhattan-World assumption, the result of [16] is more regularized. However, these two methods both produce undesired structures. In contrast, our approach generates the most compact and clean model. Besides, we can also observe that our result is less regular than that of [16]. However, this can be improved by a post-processing step using the coarse model optimization technique proposed in [20].

Limitations. Our *hypothesizing and selection* based reconstruction strategy is intended for reconstructing simple polygonal surfaces. It may encounter computation bottlenecks for large complex objects (e.g., urban scenes of many buildings). Running on such objects results in a huge number of candidate faces and the computation may not be affordable. In our experiments, we did not encounter such situations because we only tested our method on problems of manageable scales.

7. Conclusions and Future Work

We introduced a novel framework that casts polygonal surface reconstruction from point clouds as a binary labeling problem. Our framework is based on a *hypothesizing and selection* strategy, and we proposed a novel optimization formulation to obtain lightweight, watertight polygonal surface models. We demonstrated the effectiveness of our method on datasets captured by a range of devices. Since our approach seeks to find an optimal combination of the intersected planes under manifold and watertight constraints, it is guaranteed to produce lightweight, manifold models without boundary.

Future direction. Our current implementation exploits planar primitives and it is suitable for reconstructing piecewise planar objects. However, the *hypothesizing and selection* strategy is general and has the potential to handle various types of basic primitives. In future work, we would like to extend our method to incorporate more types of geometric primitives, such as cylinders and spheres. To handle data with completely missing planes, we plan to infer the missing planes by exploiting structural priors (e.g., symmetry, parallelism, and orthogonality) to obtain complete reconstructions.

Acknowledgements

We would like to thank Florent Lafarge, Michael Wimmer, and Pablo Speciale for providing us the data used in Figures 4 (d), (a) , and (e)-(f), respectively. We also thank Tina Smith for recording the voice-over for the video. This research was supported by the KAUST Office of Sponsored Research (award No. OCRF-2014-CGR3-62140401) and the Visual Computing Center (VCC) at KAUST.

References

[1] M. Arikan, M. Schwärzler, S. Flöry, M. Wimmer, and S. Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, 2013.

[2] A. Boulch, M. de La Gorce, and R. Marlet. Piecewise-planar 3d reconstruction with edge and corner regularization. In *Computer Graphics Forum*, volume 33, pages 55–64, 2014.

[3] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *CVPR 2010*, pages 1261–1268.

[4] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun. A large dataset of object scans. *arXiv preprint:1602.02481*, 2016.

[5] T. K. F. Da. 2D alpha shapes. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.9 edition, 2016.

[6] M. A. Fischler and R. C. Bolles. Readings in computer vision: Issues, problems, principles, and paradigms. chapter Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, pages 726–740. 1987.

[7] Gurobi. Gurobi optimization. `http://www.gurobi.com/`.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH 1992*, pages 71–78.

[9] H. Jiang and J. Xiao. A linear approach to matching cuboids in rgbd images. In *CVPR 2013*.

[10] I. Jolliffe. Principal component analysis. *Encyclopedia of Statistics in Behavioral Science*.

[11] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. Symposium on Geometry Processing 2006, pages 61–70.

[12] F. Lafarge and P. Alliez. Surface reconstruction through point set structuring. In *Computer Graphics Forum*, volume 32, pages 225–234, 2013.

[13] 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. In *SIGGRAPH 2000*, pages 131–144.

[14] B. Li, R. Schnabel, J. Shiyao, and R. Klein. Variational surface approximation and model selection. *Computer Graphics Forum*, 28(7), 2009.

[15] M. Li, L. Nan, N. Smith, and P. Wonka. Reconstructing building mass models from uav images. *Computers & Graphics*, 54(C):84–93, 2016.

[16] M. Li, P. Wonka, and L. Nan. Manhattan-world urban reconstruction from point clouds. In *ECCV 2016*, pages 54–69.

[17] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. In *SIGGRAPH 2011*, volume 30, page 52.

[18] H. Lin, J. Gao, Y. Zhou, G. Lu, M. Ye, C. Zhang, L. Liu, and R. Yang. Semantic decomposition and reconstruction of residential scenes from lidar data. *SIGGRAPH 2013*, 32(4):66.

[19] A. Monszpart, N. Mellado, G. J. Brostow, and N. J. Mitra. Rapter: Rebuilding man-made scenes with regular arrangements of planes. *SIGGRAPH 2015*, 34(4):103:1–103:12.

[20] L. Nan, C. Jiang, B. Ghanem, and P. Wonka. Template assembly for detailed urban reconstruction. In *Computer Graphics Forum*, volume 34, pages 217–228, 2015.

[21] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen. Smartboxes for interactive urban reconstruction. In *SIGGRAPH 2010*, volume 29, page 93.

[22] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. In *SIGGRAPH 2003*, pages 463–470.

[23] M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, and L. J. Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing 2005*, pages 23–32.

[24] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226, 2007.

[25] C. A. Vanegas, D. G. Aliaga, and B. Benes. Automatic extraction of manhattan-world building masses from 3d laser range scans. *IEEE transactions on visualization and computer graphics*, 18(10):1627–1637, 2012.

[26] Y. Verdie, F. Lafarge, and P. Alliez. Lod generation for urban scenes. *ACM Transactions on Graphics*, 34(3), 2015.

[27] Q.-Y. Zhou and U. Neumann. 2.5d dual contouring: A robust approach to creating building models from aerial lidar point clouds. In *ECCV 2010*, pages 115–128.

[28] M. Zuliani, C. S. Kenney, and B. Manjunath. The multiransac algorithm and its application to detect planar homographies. In *ICIP 2005*, volume 3, pages III–153.