Surface Reconstruction from Points

William Y. Chang[*]
Department of Computer Science and Engineering
University of California, San Diego

Abstract

This report surveys recent techniques for reconstructing surfaces from points. We describe four main ideas in the graphics literature: signed distance estimation, Voronoi-based reconstruction, implicit surface fitting, and moving least squares surfaces. The main challenges include reconstruction without surface normals, robustness to noise, accuracy to sharp features, and provable reconstruction guarantees. We compare various techniques and discuss possible avenues for future work.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: surface reconstruction, function approximation, signed distance function, voronoi diagram, implicit surface, moving least squares

1 Introduction

In recent years, there has been an increasing demand for realistic 3D computer models in a wide variety of fields and applications, including industrial manufacturing, computer animation, medical visualization and manufacturing, machine learning, and much more. One may model these surfaces by hand, using tools from the CAD/CAM community (Computer-Aided Design / Computer-Aided Manufacturing) or the graphics community. Fortunately, it is becoming more affordable and reliable to acquire a real-world object using a 3D scanning device. The following are some examples:

- Range scanning is used as a non-invasive method for scanning the surface of an object. In range scanning, a camera captures the surface as a structured pattern of light illuminates the object[1].
- Destructive scanning is used to inspect molded parts by machining and scanning thin slices of the object.
- Computer vision techniques, such as photometric stereo, multi-view geometry, shape from shading, etc. generate point samples using photographs from multiple viewpoints or lighting conditions.
- Contact probe measurement machines measure locations on a surface by directly touching the surface with a probe.

The key ingredient here is an excellent *surface reconstruction* algorithm, which allows us to construct a continuous surface given the scanned point samples of an unknown surface. This would be useful for many applications:

- Industrial applications and reverse engineering: engineers can inspect the quality of their manufactured parts by scanning them and comparing with the original 3D CAD model.
- Animation, film production, special effects: animators and artists can acquire real-world surfaces to create and reproduce realistic scenes and environments.
- Scientific simulation: physical objects can be simulated efficiently by representing them using point samples and deducing the behavior of the rest of the surface.
- Medical reconstruction and visualization: doctors and scientists can visualize medical data obtained from CAT/MRI scans or microscopy data. The data can also be used to design and manufacture medical devices and prosthetics. for example, dentists can automatically manufacture teeth to replace cracked or damaged ones by scanning and reconstructing the surface of the original tooth.
- Machine learning: examples are treated as points in a high-dimensional space, and a generalization of the reconstruction technique can be used to infer the surface or manifold which interpolates the examples.

The main challenge in designing a surface reconstruction algorithm is to guarantee that the topology of the original surface is preserved, while sharp features and surface boundaries and reproduced accurately in the reconstructed surface. In addition, the algorithm would ideally be robust to sampling density and measurement error.

This problem has been studied extensively in the literature, and there are several surveys that describe the work in this area [Bolle and Vemuri 1991; Mencl and Müller 1997; Cazals and Giesen 2006]. In this report, we will focus on recent techniques developed in the computer graphics literature. The earliest techniques for solving the problem used parametric surfaces such as NURBS or B-splines to fit and stitch together local surface patches, but we do not discuss these methods here. In addition, we will not discuss work in the vision literature that utilizes level set methods or active contours.

We examine four main ideas for point-based surface reconstruction. In section 2, we discuss techniques to estimate the signed distance function of a surface from sample points. In section 3, we will examine Voronoi-based methods, which provide a provable reconstruction guarantee. Next, in section 4 we describe implicit surface fitting techniques that produce high-quality surfaces that accommodate sharp features and boundaries. In section 5 we discuss moving least squares surfaces which are robust to noise. Finally, we compare these methods and describe areas for future work.

2 Estimating the Signed Distance Function

A key idea in finding a surface is to reliably estimate the signed distance function of the surface; i.e. the distance from a point to the surface, with positive or negative sign indicating whether the point is inside or outside if the surface is closed. Given this function, we can easily recover the surface by extracting its zero set. In this section, we examine in detail two techniques that estimate the signed distance function. Although these methods do not guarantee preserving topology and sharp features, they perform well in practice and provide a basis for which later ideas are developed.

[*]e-mail: wychang@cs.ucsd.edu

[1]This technique additionally requires a global *registration* step to align multiple scans

2.1 Tangent Plane Based Estimation

Hoppe et al. [1992] consider using oriented tangent planes to estimate the signed distance function. Given a set of point samples $X = \{x_1, x_2, \dots, x_N\}, x_i \in \mathbb{R}^3$, of an unknown surface S (where N is the number of points), we would like to compute the signed distance $d(x)$ from an arbitrary location $x \in \mathbb{R}^3$. A naive approach would be to take $d(x)$ to be the distance to the closest point sample. However, in this case the estimate would be quite inaccurate without a sufficient sampling density, and the resulting zero set would just consist of the point samples themselves. To resolve this problem, the authors approximate the surface using a set of tangent planes computed at each sample point. The tangent planes are computed as follows:

1. For each point $x_i \in X$, find the k nearest sample points. Denote this set as $N(x_i)$.

2. Compute the centroid o_i (average) of the points in $N(x_i)$.

3. Find a *tangent plane* $\mathscr{T}(x_i)$ that goes through o_i which best fits $N(x_i)$ in the least squares sense. We only need to estimate the surface normal n_i by minimizing the squared distance to the points

$$n_i = \underset{n \in \mathbb{R}^3, \|n\|=1}{\operatorname{argmin}} \sum_{y \in N(x_i)} \left(n \cdot (y - o_i)\right)^2 .$$

Given the tangent planes, they obtain an estimate of the signed distance at x by finding the closest centroid o_c and computing the distance to the tangent plane $\mathscr{T}(x_c)$ (with normal n_c):

$$d(x) = (x - o_c) \cdot n_c .$$

A problem with this method is that there is no guarantee that the normal vectors n_i have a consistent orientation. To fix this problem, they propagate the orientation of the tangent planes by traversing an augmented Euclidean minimum spanning tree graph on the sample points. Starting at an initial vertex, the orientation of the normal is propagated by visiting the neighboring vertex with the closest (most parallel) normal. Finally, the marching cubes algorithm [Lorensen and Cline 1987] is used to extract the zero set of the surface as a triangle mesh. In this step the distance function is sampled at the vertices of a cubical lattice, and the zero intersection is computed for each cube. The steps of this technique are shown in Figure 1, which illustrates the process of computing the set of tangent planes and extracting the zero contour in a cubical lattice. Notice that the sharp features in the original surface are smoothed out in the reconstructed surface; this behavior is influenced by the tangent plane estimate at the corner, which straddles both sides at the corner instead of preserving two distinct sides. In addition, although this algorithm works well with the clean datasets in this paper, it would not perform as well with a noisy data set. This is because the tangent plane estimation is not robust to outliers, as we see in the case with sharp corners.

In a series of two follow-up improvements to this method, Hoppe et al. [1993; 1994] further optimize the output to fit a subdivision surface that allows for sharp features. They incrementally tweak the mesh by a series of edge collapses, edge splits, and edge swaps to better the data points while maintaining a compact representation. This method depends on having a good initial estimate of the final surface, so it shares some of the limitations of the tangent plane based method. However, the optimization process is able to recover sharp features and boundaries.

2.2 Volumetric Estimation

A different method, known as VRIP (Volumetric Range Image Processing), is given by Curless and Levoy [1996] to estimate the signed distance from a set of range scans. Unlike the tangent plane estimation used above, rays are casted from the point of view of the scanner to determine the distance to the surface. This method is able to also combine a set of weighted range images (see Figure 2) to produce a single distance function. Their basic approach is as follows:

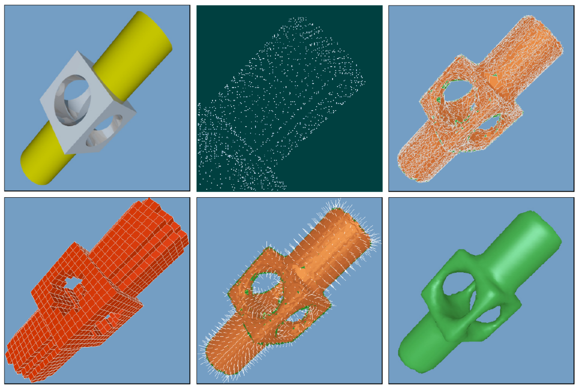
Figure 1: Reconstruction of a widget. Top left: the original surface. Top middle: a closeup of the sampled points x_i. Top right: the oriented tangent planes $\mathscr{T}_i$. Bottom left: cubes visited during contouring. Bottom middle: the signed distance function shown as lines from the cube vertices to the surface. Bottom right: final surface. From [Hoppe et al. 1992].

1. Generate range meshes by constructing triangles, connecting nearest neighbors in the range images.

2. Discretize the space surrounding the object into voxels. The voxels will store the signed distance function.

3. Cast a ray from the sensor through each voxel and intersect with the range mesh. The ray gives the distance from the voxel to the surface. The voxel is inside the surface it is closer to the sensor than the intersection point; otherwise it is labeled as outside the surface.

4. Repeat for each range mesh. This gives an array of signed distance values for each voxel, which are combined into a weighted sum.

5. The line of sight between the sensor and the mesh is an empty region, so label these as empty voxels. This approach is referred to as "space carving."

6. The zero isosurface is extracted using the marching cubes technique [Lorensen and Cline 1987].

The weights used in step 4 are determined by the dot product between each vertex normal and the viewing direction. Since the measurement has more uncertainty at grazing angles to the surface [Turk and Levoy 1994], the weights decrease as the normal becomes perpendicular to the viewing direction. In Figure 3, a 2D voxel grid is labeled as empty (black), unseen (green), or near surface (gradient) by thresholding the distance to the surface. Here, a backdrop surface behind the object can be used to reliably mark empty regions outside the object. The right side of Figure 3 is an example where a backdrop has been used to detect holes which are incorrectly filled due to the limitation of the data.

This method has a number of advantages over the tangent-plane based method. First, it is able to extract the surface from range data without the need for registration. Second, it takes into account the

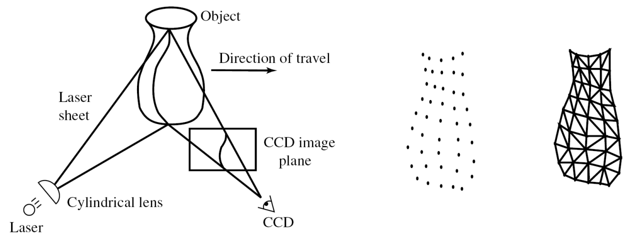

Figure 2: Acquiring a range scan. Left: a sheet of laser light illuminates the surface, while a CCD observes the reflected stripe and computes a depth profile. Right: the set of surface points generated as the laser sweeps the object from left to right. The points are connected to form a piecewise linear range surface. From [Curless and Levoy 1996].

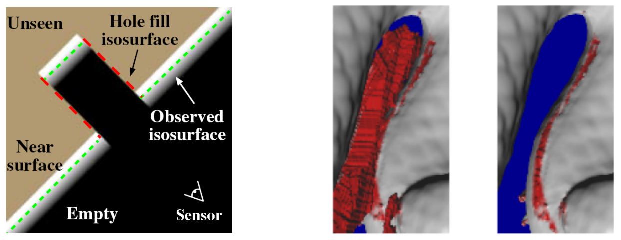

Figure 3: Left: a labeled 2D voxel grid. The green dashed lines denote the isosurface, and the red lines denote the hole fill surfaces. Right: the benefit of using a backdrop surface while scanning. The two pictures show the reconstruction result before and after using a backdrop. Notice that the incorrectly filled hole is easily detected. From [Curless and Levoy 1996].

uncertainty of the measurements; so it is robust to outliers and is able to detect and fill holes in the reconstruction. However, there are some limitations in reproducing sharp corners (limited to the resolution of the voxel grid), and the method has trouble with thin surfaces, where the range images on each side of the surface can interfere with each other. Nevertheless, this technique is quite practical, so it is cited frequently in the literature and used as a preprocessing step in a number of algorithms.

In another work, Turk and Levoy [1994] give a technique to reconstruct a surface from range scans by explicitly registering the range scans and combining the range meshes by "zippering" the boundaries together. This technique is not robust to outliers, so VRIP is usually preferred.

3 Voronoi / Delaunay Based Methods

In this section, we examine surface reconstruction techniques that use Voronoi diagrams and Delaunay complexes. These methods generate triangle meshes whose vertices are the sample points, and the goal is to determine the connectivity of the sample points that preserve the topology of the original surface. These methods come with a provable guarantee of producing a topologically correct surface. However, the quality of the output surface depends directly on the quality of the input data set, so noisy data will produce noisy surfaces.

Before discussing these methods, let us review some of the key geometric concepts. We use the definitions used by Amenta et al. [1998b]. A *Voronoi diagram* is a partition of $\mathbb{R}^d$ (where d is the dimension of the space) into *Voronoi cells* for each x_i, where every cell consists of a region closer to its sample point x_i than any other x_j, $j \neq i$ (Figure 4). Each Voronoi cell is a convex polytope, with *Voronoi vertices* equidistant from exactly $d+1$ sample points. There is a connection with the Delaunay complex: these $d+1$ points form a *Delaunay simplex* with the property that each face of the simplex has a circumsphere containing no other sample points. The set of Delaunay simplices form the Delaunay simplicial complex. It is also the *graph dual* of the Voronoi diagram. The Delaunay complex can be computed in $O(n \log n)$ for n input points using the Quickhull algorithm [Barber et al. 1996].

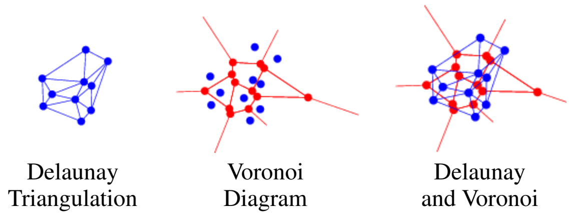

Figure 4: The Voronoi diagram and Delaunay triangulation for a 2D dataset. The blue points are the sample points, and the red points are the Voronoi vertices. From www.mathworld.com.

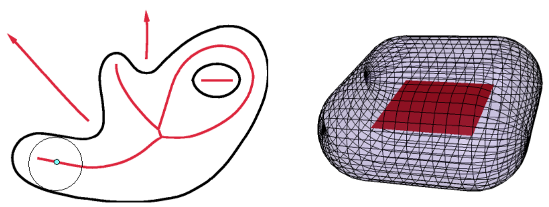

Figure 5: The medial axis. For a curve, the medial axis is also a curve, but for a surface, the medial axis can be both a surface and a curve. Left: A medial ball is shown for a point equidistant to two points on the surface. From [Amenta et al. 1998b].

3.1 Alpha Shapes

We first examine an earlier method that does not preserve the topology of the surface, but has inspired the development of the subsequent techniques in this section. This is the α-*shape* representation developed by Edelsbrunner and Mücke [1994]. Intuitively, given a point set and its Delaunay complex, an α-shape is a subset of the Delaunay complex such that the size of the simplices are less than α. In other words, we include a simplex as part of the shape if the radius of the circumsphere enclosing the simplex is less than α. Computing the α-shape is straightforward: we compute the Delaunay complex of the data set, and for each simplex we test to see if its circumsphere has radius less than α. Note that α is not determined automatically but chosen manually by the user. Figure 6 shows the different α-shapes obtained by varying the parameter α. When $\alpha = \infty$, the resulting shape is the convex hull of the point set, and as α decreases, large triangles are discarded. Notice in the figure that the resulting shape is not necessarily a connected surface. The hope is that the surface is sufficiently well-sampled so that a user can find an α which preserves the shape of the original surface.

There has been further work to extend this technique for use in surface reconstruction. Edelsbrunner et al. [1998] describe a software package called *Wrap* which requires some user input to carve the surface from the 3D Delaunay complex. Cazals et al. [2005] discusses a new approach called *conformal alpha shapes* where in-

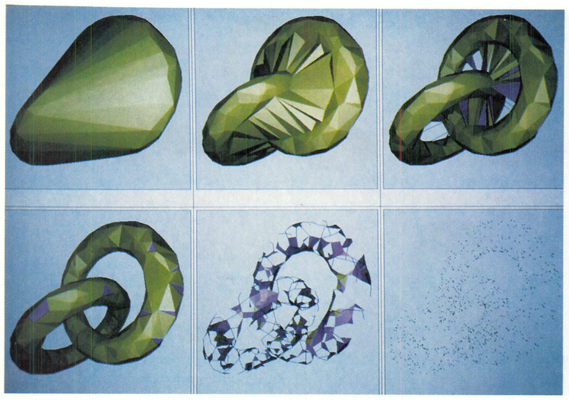

Figure 6: Reconstructing two tori. On the top right, $\alpha = \infty$ and the resulting shape is exactly the convex hull. Subsequent images show the effect of decreasing α (decreasing left to right). From [Edelsbrunner and Mücke 1994].

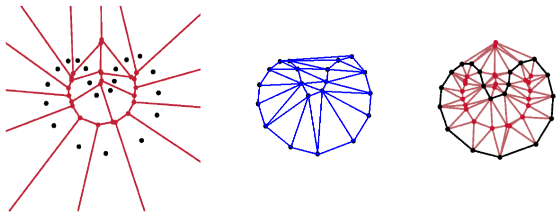

Figure 7: The two-dimensional crust algorithm. Left: the black points are the sample points, and the red points and lines show the Voronoi diagram. Middle: the Delaunay triangulation of the sample points. Right: the Delaunay triangulation of the sample points + Voronoi vertices. The sample points are connected to form the reconstructed surface. From [Amenta et al. 1998b].

stead of a global α parameter, the α is chosen on a local scale.

3.2 Crust Algorithm

Amenta et al. [1998b] show that given a sufficient sampling of the surface, the correct topology of a surface is guaranteed to be recovered based on its point samples. The method is called the *crust algorithm* and it is based on the Delaunay complex of the sample points. The main idea is illustrated in Figure 7, which shows on the left the Voronoi diagram (red) and Delaunay triangulation (blue) for a 2D data set. The observation used here is that a subset of the Voronoi diagram approximates the *medial axis*, which is the set of points that are equidistant to more than one point on the surface (Figure 5). The Voronoi vertices and lines in the figure are equidistant from at least two points in the dataset, and so it forms an approximation of the medial axis for the shape outlined by the sample points. When the Voronoi vertices are added into the dataset (right side of Figure 7, the resulting Delaunay complex does not have any edges crossing the medial axis. Therefore, the edges adjacent to the Voronoi vertices are removed, and we obtain the reconstructed surface, called the *crust*.

To obtain a guarantee that this procedure gives the correct topology, Amenta et al. devise a sampling condition on the data set. Let p be a point on the sampled surface S, and define the *local feature size* (lfs) to be the distance between p and the closest point on the medial axis. The ε-sampling condition states that the distance from any point $p \in S$ and the closest sample point x_c is less than $\varepsilon \cdot \text{lfs}(p)$. This is saying that for every point on the surface, a sample point can be found which is a factor of ε closer than the medial axis. Going back to the algorithm, adding the Voronoi vertices to the sample points, is akin to adding an approximate medial axis to the data set. Intuitively, these Voronoi vertices prevent Delaunay simplices from crossing the medial axis; by the ε-sampling, a triangle violating this condition would contain another sample point in its circumcircle. The full proof, for $\varepsilon \leq 0.06$, was obtained by Amenta and Bern [1998a].

A slight modification to this basic algorithm is needed to reconstruct 3D surfaces. The caveat with 3D Voronoi diagrams is that not all Voronoi vertices approximate the medial axis. Figure 8 is an example where we can see a Voronoi vertex that actually lies close to the surface because it is equidistant from the four sample points in the center. However, many of the Voronoi vertices do approximate the medial axis, and we select those points by picking two Voronoi vertices (from either side of the surface) that are farthest from the cell's sample point. These vertices, called *poles*, are added to the dataset to obtain the three-dimensional crust. Like the 2D case, Amenta and Bern [1998] prove that the crust is topologically equivalent to the original surface.

To summarize the algorithm:

1. Compute the Voronoi diagram of the sample points X.
2. Find the two poles for each Voronoi vertex; let P be the set of all poles.
3. Compute the Deluanay complex of $X \cup P$.
4. Keep only the simplices for which all vertices are sample points in X.

This algorithm works well in practice for ε-sampled surfaces. Figure 8 shows an example reconstruction. The running time of this algorithm depends on the total number of Voronoi vertices and sample points. Since we add two poles for each sample point, the number of points at the start of the second pass is at most $3n$. Thus, the complexity of this algorithm corresponds to the worst case time required for computing a three-dimensional Delaunay triangulation, which is $O(n^2)$.

This method produces a mesh topologically equivalent to the original surface, but it does not necessarily produce a manifold, and sometimes contains very flat tetrahedra. Also, note that it is sometimes difficult and impractical for the datasets to meet the sampling condition. For example, if there is a sharp corner in the surface, then the medial axis touches the corner, and this requires the ε-sample to be infinitely dense at the corner. In addition, the output surface does not always produce a watertight surface, and there may be undesirable holes in the output mesh.

3.3 Improvements and Extensions

There have been a number of improvements to address these deficiencies discussed above. Amenta et al. [2000] give an alternative algorithm which finds a surface with a single pass, while still guaranteeing that the reconstructed surface preserves the original topology. This algorithm, called the *cocone* algorithm, selects a subset of Voronoi edges whose dual simplices form the reconstructed surface. The intuition is to select those Voronoi edges that are approximately normal to the surface. Figure 10 illustrates this algorithm with a 2D dataset. To select the Voronoi edges, they first approximate the normal at each sample point p by connecting with the furthest Voronoi vertex within p's Voronoi cell. The cocone, also shown in Figure 9, is the complement of the double cone with apex p making an angle of $\pi/2 - \theta$ with the estimated normal (for small θ). Using this information, the Voronoi edges that are incident with cocones on

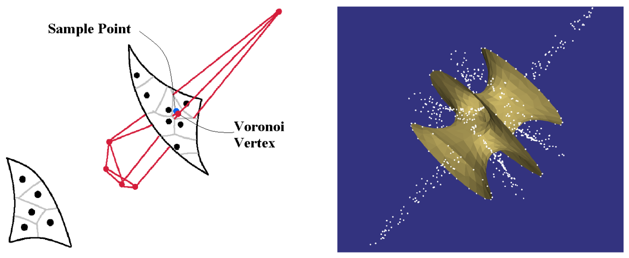

Figure 8: Left: the problem with three dimensions. The Voronoi cell of the blue point is shown; one of the Voronoi vertices lies close to the surface. Right: example of a reconstructed surface. The white dots are the poles used to compute the crust. From [Amenta et al. 1998b].

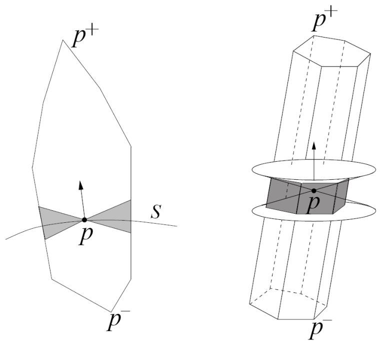

Figure 9: Illustration of the 2D and 3D cocone. The Voronoi cell of sample point p is shown with its two poles p^+ and p^-. The estimated normal direction is shown as an arrow from p to the farthest Voronoi vertex. The cocone is the shaded region, marking the directions nearly perpendicular to the normal up to a small angle θ.

all sides are selected, and their dual edges form the reconstructed surface. This method is simpler than the crust algorithm, because it only requires computing the Voronoi diagram and Delaunay complex once for the original set of sample points. In addition, given a sufficient sampling, this algorithm guarantees that the output surface is a manifold and is homeomorphic to the original surface, which did not hold in the crust algorithm.

Dey and Giesen [2001] further improve the cocone algorithm to make it robust to undersampled point sets. This method detects undersampled regions by checking that each Voronoi cell is elongated

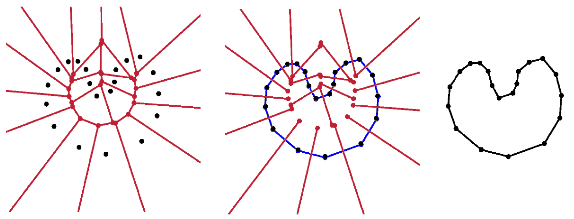

Figure 10: The cocone algorithm in 2D. Left: the Voronoi diagram of the sample points. Middle and Right: the cocone algorithm selects the Voronoi edges that are nearly normal to the surface. Their dual edges are marked as blue, and they form the reconstructed surface shown on the right. Adapted from [Amenta et al. 1998b].

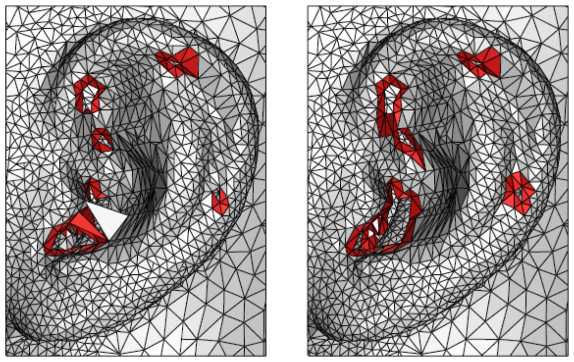

Figure 11: Removing undersampled triangles. Triangles incident to boundary sample points are shaded darker. Left: the inner structure of the ear is occluded by triangles spanning an undersampled region. Right: the improved result. From [Dey and Giesen 2001].

in the normal direction and consistent with its cocone neighbors. Points that fail this criterion are marked as boundary points, and triangles whose vertices consist only of boundary points are removed in the reconstructed surface. In Figure 11, garbage triangles that occlude the inner structure of the ear are removed after boundary detection.

Amenta et al. [2001] improve upon the original crust algorithm to produce a topologically correct, watertight surface even in the presence of moderate noise. This method uses the pole vertices to approximate the medial axis as in the crust method. However, instead of adding the Voronoi vertices to the point set to compute the crust, this method computes the *power diagram* on the pole vertices. Figure 12 illustrates the power crust algorithm for a 2D dataset; from the Voronoi diagram a set of *polar balls*–spheres centered at each Voronoi vertex and touch the equidistant points–are extracted at each pole vertex. The power diagram is just a Voronoi diagram with a different distance metric: instead of the normal Euclidean metric, a weight value at each point is subtracted from the usual Euclidean distance. Here, the radii of the polar balls (which are equivalent to the circumsphere of each Delaunay simplex) define weights on the poles. They show that the vertices of this power diagram are the sample points themselves, so the power diagram produces a simplicial complex, which is similar but different from the Delaunay complex. Finally, each power diagram cell is marked as either interior or exterior to the surface by propagating the interior / exterior label at each pole based on how much their polar balls intersect. This heuristic works well because a dense set of samples well-separates polar balls on each side of the sample, causing a shallow intersection. The boundary of the interior region is the reconstructed surface, called the *power crust*. In Figure 12, (d) shows the power diagram with each cell labeled as interior or exterior, and (e) shows the reconstructed surface by selecting the boundary of the interior cells. Because they explicitly label a region as interior to a surface, this algorithm produces watertight models. In addition, the power diagram allows sharp corners to be recovered even with sparse samples by discarding Voronoi cells that are not elongated outwards in the normal direction of the surface.

4 Implicit Surface Constructions

In this section, we discuss a different type of surface representation called *implicit surfaces*. For rendering applications, implicits are usually converted to triangle meshes, but they are useful for modeling and simulation. The main advantage is the ability to easily

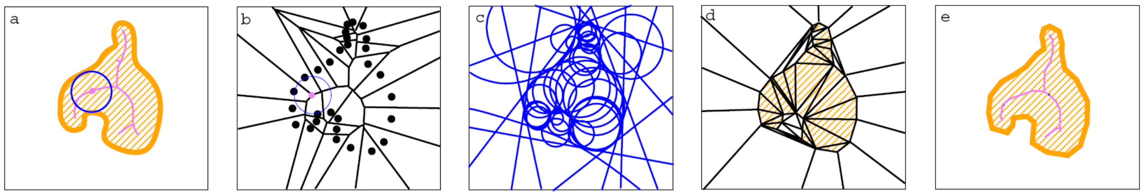

Figure 12: Power crust algorithm. (a) shows the medial axis, (b) the Voronoi diagram and a polar ball, (c) the set of polar balls, (d) the resulting power diagram with cells labeled as interior/exterior to the shape, and (e) the reconstructed power crust. From [Amenta et al. 2001].

test if a point is inside or outside the surface. Constructive solid geometry (CSG) modeling operations are particularly easy to implement, where we compute the addition, intersection, and subtraction of multiple implicit surfaces. In addition, collision detection can be implemented efficiently using implicit surfaces [Pentland 1990]. For surface reconstruction, implicits provide a compact representation of a surface that can allow either smooth interpolation or approximation to the data points.

An implicit surface is defined as the isocontour of a scalar function. Given a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, the isosurface at scalar c is the set of all $x \in \mathbb{R}^3$ such that

$$f(x) = c.$$

Note that the signed distance function discussed in section 2 is a special case of the class of implicit functions, where the surface is defined at $c = 0$.

Given a set of scattered data points, we would like to directly fit an implicit surface which interpolates or approximates the data points. Mathematically, we can express this as an optimization problem as follows: given a scattered data set $\mathscr{P} = \{x_i \in \mathbb{R}^3\}$, find an implicit function $f(x)$ such that for all x_i,

$$f(x_i) = c. \tag{1}$$

Alternatively, we can cast this as an approximation problem by minimizing the quantity

$$e_{\text{fit}}(f) = \sum_{x_i \in \mathscr{P}} (f(x_i) - c)^2. \tag{2}$$

There is a trivial solution to this problem, when $f(x)$ is the constant function with value c, i.e.

$$f(x) = c \text{ for all } x \in \mathbb{R}^3.$$

In order to avoid this solution, we need additional constraints which say that $f(x) \neq c$ for points x inside or outside the surface. This can be specified explicitly using off-surface points and assigning them values f_i which are less than or greater than c, or implicitly by specifying a normal vector n_i for each data point x_i. We can easily obtain normals for the implicit surface by computing normalized gradient evaluated at the surface point x:

$$n(x) = \frac{\nabla f(x)}{\|\nabla f(x)\|}.$$

This is indeed the surface normal because the gradient denotes the direction of greatest change, and therefore it must be perpendicular to the isosurface which has no change in value. In addition to the normal and positional constraints, there usually is a regularization term that helps to control the smoothness of fit.

The main challenge in this method is choosing the type of function to represent the surface, and then solving the resulting optimization problem in (1) and (2). In the following sections, we discuss global and local fitting techniques, and discuss the robustness of each method with respect to noise and sharp features. Unlike the Voronoi-based techniques in the previous section, most of these methods have no provable guarantee to preserve the topology of the original surface. However, implicits are better able to deal with noise and can produce a wide variety of smooth surfaces that fit the points. The main limitation of these techniques is that they usually require sampling surface normals in addition to sampling surface locations. It is possible to measure surface normals when sampling real-world objects, but often the normals contain noise and are unreliable.

4.1 Radial Basis Function Techniques

4.1.1 Fitting the Blobby Model

Muraki considers applying the “Blobby Model” [Blinn 1982] to fit a scattered dataset [Muraki 1991]. This model $f(x)$ consists of a weighted sum of M “primitive” functions $f_i(x)$,

$$f(x) = \sum_{j=1}^{M} b_j e^{-a_j f_j(x)}$$

where each $f_j(x)$ is a spherically symmetric function centered at $p_j \in \mathbb{R}^3$,

$$f_j(x) = \|x - p_j\|^2$$

and parameters a_j, b_j determine the size and weight of function f_j, respectively. The goal is to automatically determine the number of primitives M and the parameters a_j, b_j, and p_j that best fit the data points. The resulting minimization problem has the form

$$\underset{f}{\operatorname{argmin}} \; \frac{1}{N}\sum_{i=1}^{N}\left(\underbrace{(f(x_i)-c)^2}_{\text{Constraint 1}} + \alpha \underbrace{\left\| \frac{\nabla f(x_i)}{\|\nabla f(x_i)\|} - n_j \right\|^2}_{\text{Constraint 2}}\right) + \beta \underbrace{\left(\sum_{j=1}^{M} a_j^{-3/2} |b_j|\right)^2}_{\text{Constraint 3}} \tag{3}$$

where n_i is the measured surface normal at each x_i (given as input), and α and β are weighting parameters specified by the user to control the strength of the constraints 2 and 3. The constraints in this equation have the following roles:

- Constraint 1. Fits the given point locations.
- Constraint 2. Matches the surface normals to the given normals.
- Constraint 3. Limits the influence of each primitive to prevent forming shapes far away from the data points.

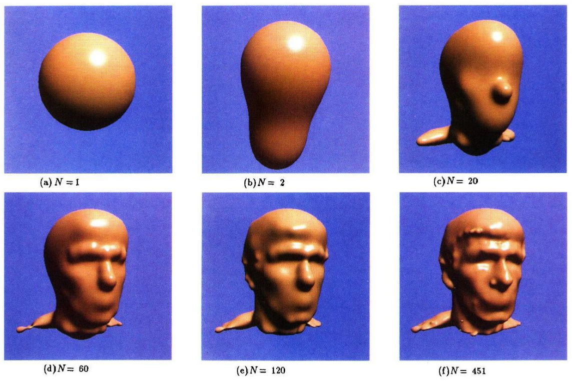

Figure 13: Approximating a head model with increasing number of primitives. From [Muraki 1991].

Name	Formula
Gaussian	$\phi(\|x\|) = e^{-c\|x\|^2}$
Multiquadric	$\phi(\|x\|) = \sqrt{\|x\|^2 + c^2}$
Inverse Multiquadric	$\phi(\|x\|) = 1/\sqrt{\|x\|^2 + c^2}$
$\mathbb{R}^3$ Biharmonic Spline	$\phi(\|x\|) = \|x\|$
$\mathbb{R}^3$ Triharmonic Spline	$\phi(\|x\|) = \|x\|^3$
Multivariate Spline	$\phi(\|x\|) = \begin{cases} \|x\|^{2k-d} & d \text{ odd} \\ \|x\|^{2k-d} \ln \|x\| & d \text{ even} \end{cases}$

Table 1: Commonly used basis functions $\phi(\|x\|) : \mathbb{R}^d \to \mathbb{R}$ in (4).

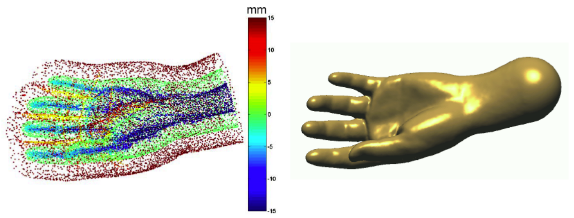

Figure 14: A hand dataset with on-surface (green) and off-surface(outside red, inside blue) points and resulting reconstruction. From [Carr et al. 2001].

This is a non-linear optimization problem involving $5M$ unknowns total (5 unknowns $(a_j, b_j, p_{j_x}, p_{j_y}, p_{j_z})$ for each primitive $f_j(x)$). Since it is difficult to solve for all unknowns simultaneously, the objective function is repeatedly optimized with two copies of a single primitive, choosing a different primitive at each step. This process continues until the approximation error falls below a chosen threshold (see Figure 13). To solve (3) they use the downhill simplex method [Press et al. 1988] to obtain an initial value for the quasi-Newton method [Press et al. 1988]. Although this method produces reasonable results, the optimization is expensive to evaluate because there are too many degrees of freedom. Also, there is no bound on the number of primitives needed to accurately approximate a surface.

4.1.2 Fitting Radial Basis Functions

The blobby model discussed in the previous section belongs to a family of functions called “radial basis functions (RBF)”. Radial basis functions have the general form

$$f(\cdot) = p(x) + \sum_{j=1}^{N} \lambda_j \phi(\| \cdot - x_j \|), \tag{4}$$

where $p(x) = \sum_{l=1}^{t} c_l p_l(x)$, is a polynomial of degree m and $\phi(\| \cdot - x_j \|)$ is called a *basic function* centered at $x_j \in \mathbf{R}^3$. Here, the polynomial is expressed as a linear combination of polynomial basis functions $\{p_1, \ldots, p_t\}$ with coefficients $c_1, \ldots, c_t \in \mathbb{R}$. Example basic functions are shown in Table 1 [Reuter 2003]. If we have a basic function for each data point x_i (total N) and a specific $\phi(x)$, the data interpolation problem (1) reduces to a linear system, where we solve for the unknown weights λ_j and polynomial $p(x)$:

$$\begin{pmatrix} A & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{5}$$

where

$A_{i,j} = \phi(\|x_i - x_j\|)$	(ϕ centered at x_j evaluated at x_i)
$P_{i,l} = p_l(x_i)$	(evaluation of polynomial p_l at x_i)
$\lambda = (\lambda_1, \ldots, \lambda_N)^T$	(RBF coefficients)
$c = (c_1, \ldots, c_t)^T$	(polynomial coefficients)
$f = (f_1, \ldots, f_N)^T$	(function values at x_i)

In this system, the size of A is $N \times N$, and the size of P is $N \times t$, so the size of the entire system scales quadratically ($O(N^2)$) with the number of points N. $P^T \lambda = 0$ is called the “orthogonality” or ”side condition” which ensures that $f(x)$ is well-behaved. To avoid a degenerate solution, the linear system requires off-surface constraints that specify implicit function values away from the surface. Usually off-surface points are generated by displacing the surface points along the normal direction and offsetting the function value by the displacement distance.

Carr et al. [2001] give a practical technique to fit these radial basis functions to a scattered data set. This technique requires the use of off-surface points, as shown in Figure 14, where the off-surface points are generated by displacing the points along the positive and negative normal directions. This is a clear limitation of this method, along with the inability to fit sharp features in the dataset.

The main contribution of this paper is to use the Fast Multipole Method (FMM) to efficiently evaluate the radial basis function [Greengard and Rokhlin 1987]. This is an important contribution because the linear system of (5) is dense and difficult to solve quickly with millions of input points. The FMM technique evaluates RBFs rapidly up to any accuracy by first hierarchically clustering the RBF centers and then using approximate evaluation (“far-field”) or direct evaluation (“near-field”) based on the distance to the evaluation point. The basic mathematical tool behind this clustering technique is a series expansion of the basic function $\phi(x_i, x_j)$ into the form $\sum_k \varphi_k(x_i) \psi_k(x_j)$. This allows us to rewrite the latter part of (4) in the form

$$\sum_j \lambda_j \phi(x_i, x_j) = \sum_k \varphi_k(x_i) \left(\sum_j \lambda_j \psi_k(x_j) \right).$$

The latter part of the RHS is indpendent of i, so it can be precomputed and reused in subsequent evaluations. This series expansion has bounded error within a specific radius from the center of expansion, which is why the RBF centers x_j are clustered into regions approximated with a single expansion. Further details on the fast multipole method and the fitting methods can be found in [Beatson and Greengard 1997] and [Beatson et al. 1999; Beatson et al. 2000]. Using these techniques reduces the evaluation time of a single sample point from $O(N)$ to $O(1)$ after $O(N \log N)$ preprocessing time, and linear solving time complexity from $O(N^3)$ to $O(N \log N)$.

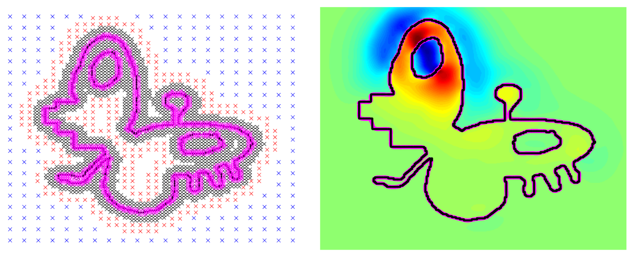

Figure 15: Two-dimensional reconstruction. On the left, the data points are black dots, and the basis function centers are crosses color-coded based on their width. On the right is the resulting implicit function and the surface rendered as a magenta line. From [Walder et al. 2005].

4.1.3 Fitting RBFs Without Normals

Walder et al. [2005] demonstrate that it is actually possible to fit RBFs without using normal vectors or off-surface points. They formulate an optimization problem that enforces some desirable properties for the resulting fit $f(x)$: the surface should fit the data, avoid a degenerate solution $f(x) = 0$, have a consistent sign inside/outside the surface, and be smooth everywhere. The resulting problem is

$$\begin{aligned} \operatorname*{argmin}_f \; & \underbrace{\sum_{i=1}^{N} f(x_i)^2}_{\text{Data Term}} - \underbrace{\lambda_e \int_{u\in\mathbb{R}^d} f(u)^2 \, d\mu(u)}_{\text{Energy Term}} \\ & \underbrace{-\lambda_\nabla \sum_{i=1}^{N} \|(\nabla f)(x_i)\|^2}_{\text{Gradient Term}} + \underbrace{\lambda_\Omega \|f\|^2}_{\text{Regularization Term}} \end{aligned} \tag{6}$$

where $\lambda_e, \lambda_\nabla, \lambda_\Omega$ are weights that control the strength of each term. The role of each item is:

- Data Term: constrains values at data points $|f(x_i)|$ to be small (since we are fitting a zero isosurface $f(x) = 0$)
- Energy Term: constrains other values $|f(x)|$ to be large
- Gradient Term: constrains the gradients $\|(\nabla f)(x_i)\|$ to be large
- Regularization Term: constrains the smoothness of f using thin plate energy [Duchon 1976]

Each term in this problem is quadratic which allows them to formulate it as an eigenvalue problem. Moreover, using compactly supported B-splines, the matrices are sparse and the problem can be solved efficiently. To set up the linear system, a grid of basis functions covers the entire domain. The resolution of the grid is dependent on the proximity to the data points. Figure 15 shows the system solved for a two-dimensional dataset. Although this method relieves the use of normals for fitting RBFs, a parameter search is required to find reasonable values for λ-weights of each term to ensure that the solution is both stable and accurate. In addition, sharp features tend to smoothed out in the reconstruction, and the algorithm provides no control over the accuracy for sharp features (see Figure 16).

4.2 Multi-level Partition of Unity Implicits

The multi-level partition of unity approach (MPU) by Ohtake et al. solves the fitting problem by blending local shape functions

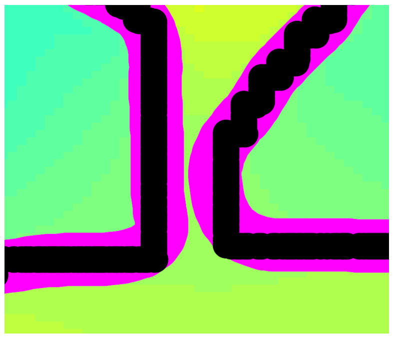

Figure 16: Closeup of the two-dimensional reconstruction. The reconstructed implicit smooths out sharp features. From [Walder et al. 2005].

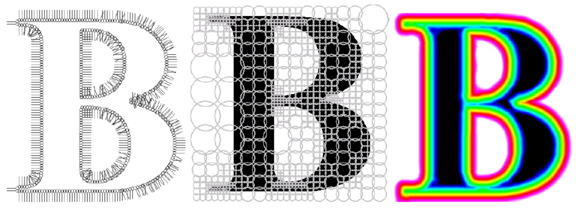

Figure 17: Adaptive subdivision of the domain. Left: the original dataset with normals. Middle: subdividing the domain. The circles denote the support of each cell. Right: the resulting implicit function. From [Ohtake et al. 2003].

[Ohtake et al. 2003]. Unlike the previous RBF approaches that fit the entire dataset at once, this technique fits small subsets of the data separately and blends these fits together using partitions of unity. This blending preserves the fitted shape in each cell, so sharp features can be detected and fit independently without sacrificing smoothness in other regions of the surface.

On a high level, this algorithm maintains an adaptive octree which subdivides the domain into cells based on fitting error. Starting by considering the entire domain as a single cell, the algorithm

1. Fits a local shape implicit $Q_i(x)$ in each cell,
2. Computes the approximation error (ε) of the fitted implicit surface,
3. If ε is greater than a user-specified threshold, discards the fit and subdivides the cell into multiple regions,
4. Repeats the above procedure until the error meets the threshold criterion.

A partition of unity, used to blend the local implicits, is a set of non-negative compactly supported functions $\{\varphi_i\}$ in a bounded domain Ω such that

$$\sum_{i=1}^{M} \varphi_i \equiv 1 \text{ on } \Omega.$$

These functions can be generated using any non-negative compactly supported functions $w_i(x)$ by normalization:

$$\varphi_i(x) = \frac{w_i(x)}{\sum_{j=1}^{M} w_j(x)}.$$

This method uses quadratic B-splines to generate partitions of unity. These functions are assigned to each cell in the domain such that its

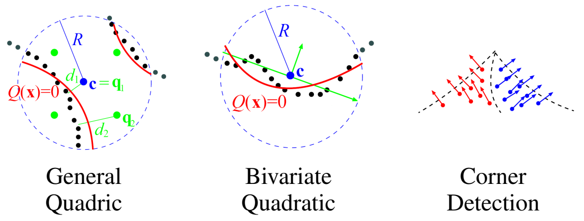

Figure 18: Fitting local shape functions. The dotted circle centered at c with radius R denotes the support of the blending function. $Q(x) = 0$ is the locally fitted implicit, using the off surface constraints q_i each with estimated distance d_i. Left: fitting a general quadric. Middle: fitting a bivariate quadratic on a smooth patch. Right: Clustering normals to detect corners. From [Ohtake et al. 2003].

support covers the entire cell. In regions where the supports overlap, the local implicits $Q_i(x)$ are blended by computing a weighted sum

$$f(x) = \sum_{i=1}^{M} \varphi_i(x) Q_i(x).$$

Three types of local shape functions are used in this technique: general quadrics, bivariate quadratics, and piecewise quadric surfaces. Figure 18 shows examples of the three cases. General quadrics are used to approximate a large number of points or multiple sheets, bivariate quadratics are used to approximate local smooth patches, and piecewise quadrics are used to model sharp features which are detected by clustering the normal vectors in the cell. To avoid fitting degenerate functions, off-surface constraints are generated at the center and corners of each cell. The value of the constraint is determined by averaging the projected distance to the six nearest sample points along each sample normal.

This method differs significantly from all of the methods described so far in that it is a *local* method as opposed to a *global* method. Instead of fitting a surface to the entire dataset at once, this method divides the dataset into manageable regions and locally fits surfaces which are stitched together using partitions of unity. This allows the fitted surface to have sharp features in the dataset, unlike the RBF-based methods discussed in the previous two sections. In addition, MPU runs considerably faster than RBF techniques such as [Carr et al. 2001]. Unfortunately, MPU requires surface samples equipped with surface normals, and the local shape functions are fitted based on heuristics with no theoretical guarantees. It is unclear that these heuristics would be robust in the presence of measurement error in the sample positions and normals.

4.3 Moving-Least Squares Implicits

Closely related to the partition of unity method above, Shen et al. [2004] discuss a moving-least squares technique to create implicit surfaces that fit polygonal data. Their goal is to convert polygon soup (an unstructured set of polygons) to an implicit surface using points and normals sampled from the polygons. Like the MPU method, this method fits local shape functions to the data, but it does not explicitly subdivide the domain and blend the shapes together. Note that this method is not a surface reconstruction method by interpolating/approximating scattered *points;* it constructs an implicit surface from an unstructured *polygon soup.* However, this method is a good application of the moving-least squares idea, which is the main theme for the next section.

Recall that our overall goal is to find a function $f(x)$ which approximates the values at the data points. Given a set of scattered data values $\{f_i\}$ each at locations $\{x_i\}$, we want $f(x_i) \approx f_i$ for all i. The approximation technique used by Shen et al. is called *moving least squares,* also known as *local regression* in the statistics literature [Cleveland and Loader 1996]. In this method, for each point x we find a function $\tilde{g}$ such that

$$\tilde{g} = \underset{g \in B}{\operatorname{argmin}} \sum_{i=1}^{n} (g(x_i) - f_i)^2 \, \theta(\|x - x_i\|),$$

and assign the value $f(x) = \tilde{g}(x)$. Here, B denotes a family of basis functions (constant, linear, polynomial, etc.) and θ a local weighting function. The role of θ here is to weight each data point according to the distance from x. When the weight function is a Gaussian $\theta(x) = e^{-x^2/h^2}$, data points close by have large weights, but points far away have smaller weights. In this sense, $f(x)$ is a local approximation of the data. The important feature of this technique is that the local fit varies based on where we evaluate the function. Figure 19 shows an example of moving least square approximation to a 1D data set. The top example shows a global least-squares fit of a linear function. Note the striking difference in the second figure which uses moving least squares. Note that the approximating behavior of the function can be controlled by changing the local weighting function.

We now describe the approach used by Shen et al. Let $b(x)$ be the vector of basis functions we use for a local fit; for example $b(x) = [1]$ if we fit a constant function, or $b(x) = [1, x, y, z]$ if we fit a plane. Now, given the dataset $\{x_i\}$ with function values $\{f_i\}$ as before, to evaluate the implicit function $f(x)$ we first solve the weighted least squares problem

$$\underset{c}{\operatorname{argmin}} \; e_{\text{MLS}}(x, c) = \underset{c}{\operatorname{argmin}} \sum_{i=1}^{N} \theta(\|x - x_i\|)(b^T(x_i)c - f_i)^2$$

where $\theta(x, x_i)$ is the weight function and c is the vector of basis coefficients. Solving the resulting linear system for c produces a local fit, and we assign the function value $f(x) = b^T(x)c$. Here, the authors use the weight function $\theta(\|x - x_i\|) = 1/(\|x - x_i\|^2 - \varepsilon^2)$, because this function can produce varying degrees of interpolating and approximating behavior. When $\varepsilon = 0$ then $\theta \longrightarrow \infty$ as $x \longrightarrow x_i$, causing the fit to interpolate the data points, whereas a nonzero ε will cause the fit to approximate the data points, as we have seen in Figure 19.

Because the goal of this technique is to fit a set of polygons accurately, more samples lead to a better approximation. Figure 20 demonstrates that a finite sampling results in dents and dimples in the resulting surface. To simulate an infinite distribution of point samples, they analytically integrate the objective function over each polygon Ω_k:

$$e_{MLS}(x, c) = \sum_k \int_{p \in \Omega_k} \theta(\|x - p\|)(b^T(p)c - f_k)^2 dp.$$

Here, the function values f_k are equal to zero at each point $p \in \Omega_k$, but they can be iteratively modified to fit approximating surfaces that completely enclose the surface. The integral constraints here are numerically evaluated using a modified quadrature rule that avoids the singularity of the weighting function near zero.

In addition to the point constrains, Shen et al. use the normal information to avoid a degenerate solution. Unlike the RBF methods which used off-surface constraints, they formulate a continuous off-surface constraint by interpolating linear functions at each surface point along the normal. So for each triangle k and a fixed point q_k and normal n_k on this triangle, the points f_k are replaced by functions $F_k(x)$:

$$F_k(x) = f_k + (x - q_k)^T n_k.$$

Here, each triangle is able to "predict" the implicit function value based on the projected distance to the triangle along the normal.

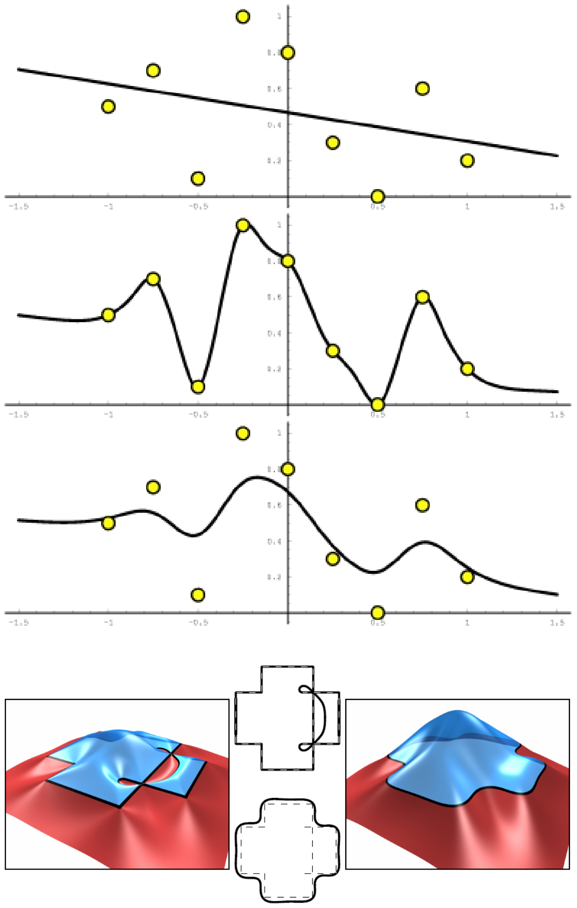
Figure 19: On top: Comparison of least squares, moving least squares interpolation, and moving least squares approximation. On bottom: height field generated using a 2D dataset. On the left is interpolation behavior, on the right approximation behavior. The extracted isocontours are shown in the middle. From [Shen et al. 2004].

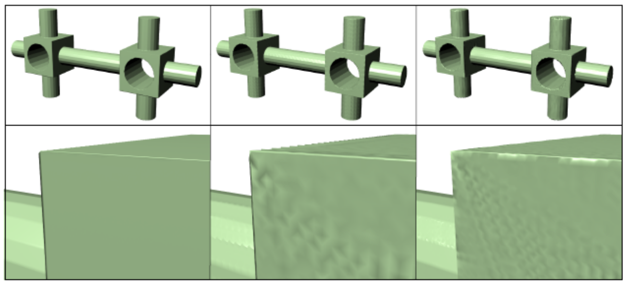
Figure 20: Dimples resulting from a finite data set. Left: integrated polygon constraints. Middle and Right: coarse and fine densities of finite point constraints. From [Shen et al. 2004].

Adding this constraint gives the least squares minimization problem

$$e_{MLS}(x,c) = \sum_k \int_{p \in \Omega_k} \theta(\|x-p\|)(b^T(p)c - F_k(x))^2 dp.$$

This method produces nice implicit surfaces that exactly fit the

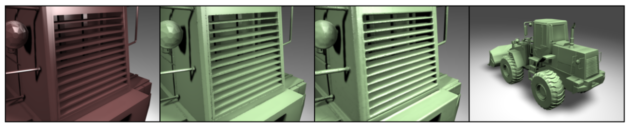
Figure 21: Result of fitting a polygon mesh. Left: the original polygon mesh. Middle left: close-up of an interpolating implicit surface. Note the reproduction of sharp edges from the original polygon mesh. The dented appearance near edges is a polygonization artifact. Middle right: close-up of a slightly approximating surface. Right: the entire interpolating surface. From [Shen et al. 2004].

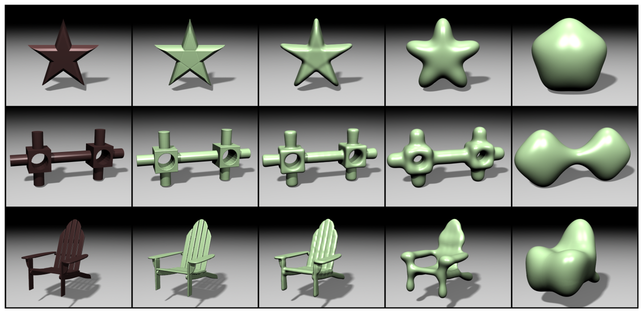
Figure 22: Approximating various surfaces with varying ε. On the far left is the original polygonal model, followed by an interpolating implicit, and increasing ε to the right. From [Shen et al. 2004].

given polygonal models, including boundaries and sharp features. Converting such models into implicit form is useful because an unstructured set of polygons can be converted into a mesh by contouring the implicit representation. In addition, the implicits can be used for CSG operations, physical simulation, or efficient collision detection. Moreover, as Figure 22 demonstrates, with increasing ε the method can generate coarse approximating implicit surfaces that completely enclose the surface.

As mentioned before, the f_k values are modified to guarantee the enclosure. Initially if the approximating surface does not tightly enclose the original mesh, the function values are incremented or decremented depending on whether the implicit lies inside or outside the mesh. This procedure is iterated until the entire mesh is enclosed within the implicit. These enclosures, and their corresponding polygonizations, can be used as "simulation envelopes" for creating efficient and practical physical simulations.

This method produces very high-quality implicit surfaces that precisely fit a polygon mesh, reproducing sharp features, holes, and boundaries. However, it has the unfair advantage that the input is a polygon mesh, not point samples. Certainly this method is applicable to a finite point set, but the quality of the surface degrades as we have seen in Figure 20. Another drawback is that the entire point set must be stored to evaluate the implicit surface, because this technique does not use parameterized functions such as RBFs or local shape functions. Recent work has shown that it is possible to prove reconstruction guarantees for the moving least squares implicit surface. Kolluri analyzes a similar implicit surface that computes the moving signed distance function of the sampled surface and shows that for a sufficiently dense sampling, the zero-set U is homeomorphic to the original surface F [Kolluri 2005]. Dey and Sun show a similar result with a normal estimation procedure in the absence of normals [Dey and Sun 2005].

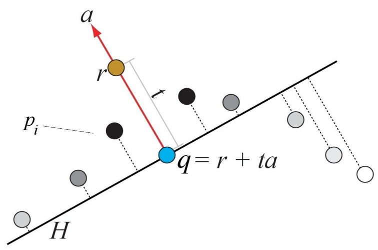

Figure 23: Projecting a point using MLS. From [Amenta and Kil 2004b].

5 Moving Least-Squares Surfaces

The moving least-squares surface (MLS for short) was first developed by Levin [2003] as a way to apply the moving least squares function approximation to surface reconstruction. In the previous section, we discussed a moving least squares technique to approximate an implicit function that fits the sample points. However, this required specifying function values and normal vectors at each sample point. Without this information, we cannot apply this *function approximation* technique directly to our surface reconstruction problem.

Instead, we could attempt to use the coordinates of the points as function values above a two dimensional plane, but in general we cannot hope to find such a global reference domain. The key idea that was introduced by Levin is to use locally fitted planes as *local* reference domains in which the coordinates resemble a function. In the following sections, we present in more detail the definition of the MLS surface and examine work that improves the surface definition by addressing non-smooth surfaces, surfaces with boundary, and numerical issues such as convergence and stability.

5.1 MLS Surface Definition

The MLS surface is a generalization of MLS function approximation discussed in the beginning of Section 4.3. The basic idea is to first find a local reference plane and then fit a polynomial function to the data points [Levin 2003]. Both the local reference plane and the polynomial fit depend on *where* we evaluate the MLS surface, and like the functional setting there is no notion of "stitching together" fitted functions. This type of approach is *non-parametric,* because the model is inferred directly from the data without the need for estimating parameters.

To determine the MLS surface, given a query point we fit a surface patch in a local neighborhood and move the query point to the patch. This is similar to the functional setting, where we assign the value of the locally fitted polynomial. Thus we "project" query points to the surface.

Let us state this idea mathematically. We are given a point set $\mathscr{P} = \{p_i \in \mathbb{R}^3\}$, and we would like to project a query point $r \in \mathbb{R}^3$ to the surface. There are two steps in the projection procedure:

- **Step 1.** First, we compute a local reference plane in the neighborhood of r (Figure 23). More precisely, we find a plane H with normal vector $a \in \mathbb{R}^3$ passing through some point $q = r + ta$ (for some $t \in \mathbb{R}$) such that $\|a\| = 1$ and H minimizes the least-squares error:

$$\begin{aligned} e_{MLS}(a,t) &= \sum_{p_i \in \mathscr{P}} \langle a, p_i - q\rangle^2 \, \theta(\|p_i - q\|) \\ &= \sum_{p_i \in \mathscr{P}} \langle a, p_i - r - ta\rangle^2 \, \theta(\|p_i - r - ta\|), \end{aligned} \tag{7}$$

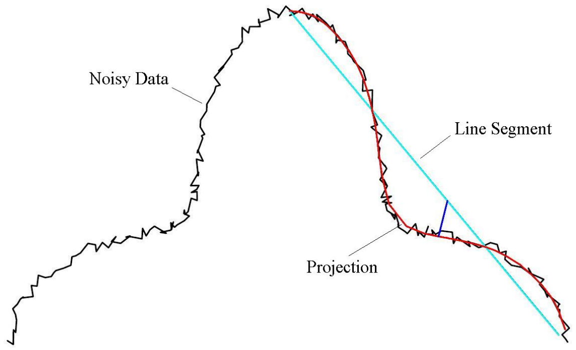

Figure 24: Noisy data, a line segment, and its MLS projection. 30 points on the line were projected to the MLS surface; one of these projections is shown as a short line segment connecting the line and the curve. From [Levin 2003].

where $\langle \cdot, \cdot \rangle$ is a dot product of two vectors and $\| \cdot \|$ is the norm of a vector. Here, $\theta(\|p_i - q\|)$ is the weighting function, and $\langle a, p_i - q\rangle^2$ is the squared distance from each point p_i to the plane H. Note that e_{MLS} is non-linear (due to $\theta(x) = e^{-x^2/h^2}$), but this optimization problem can be solved using an iterative method.

- **Step 2.** Consider the orthogonal projections of the points $\{p_i\}$ as the points $\{x_i\}$ on the plane H with projected distance $f_i = \langle a, p_i - q\rangle$. Here, the f_i's can be considered as function values at locations x_i on the plane H oriented with normal a and origin q. Thus, we can apply MLS function approximation, and this is the second step of the projection: find a polynomial $\tilde{g} \in \Pi_m^2$ (bivariate polynomial of degree m) which locally minimizes the least-squares error around q:

$$\tilde{g} = \underset{g \in \Pi_m^2}{\operatorname{argmin}} \sum_i (g(x_i) - f_i)^2 \, \theta(\|x_i - q\|). \tag{8}$$

Since $\tilde{g}(0)$ is the local approximation of the surface at q, we can project r along the normal with distance $\tilde{g}(0)$: $q + \tilde{g}(0)a = r + (t + \tilde{g}(0))a$. This is the projection of r onto the surface using a degree m polynomial fit.

Figure 24 is an example of this technique applied to a two-dimensional dataset. This method also applies for reconstructing $(d-1)$-dimensional manifold given a d-dimensional data set.

Some theoretical and practical questions remain with this work. First, there is no guarantee that the result is a surface. To ensure that the MLS surface does not have multiple overlapping sheets, we need to assume a "uniqueness domain" $U \subseteq \mathbb{R}^d$ such that for any $\mathbf{r} \in U$, step 1 has a unique solution $\mathbf{q}$. The existence of this domain and its characterization has not resolved and remains an open problem [Levin 2003]. Second, Levin conjectures that if the weighting function is infinitely smooth, the resulting MLS surface is also C^∞ smooth. In the functional case this is certainly true [Levin 1998] [Lancaster and Salkauskas 1981]. This may be a desirable property; however, if the surface has non-smooth features such as sharp corners and boundaries, a C^∞ smooth surface is undesirable. It is possible to extend MLS surfaces to accommodate sharp surface features, which we will discuss in the following sections.

Alternative Definition. Amenta and Kil [2004b] give an (equivalent) explicit definition of the MLS surface that directly determines whether a point belongs to the MLS surface. The key observation is

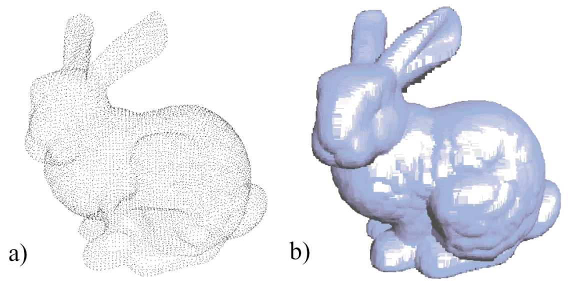

Figure 25: Sampling and splatting a bunny using MLS. From [Alexa et al. 2001].

to characterize the MLS energy function independent of the query point r:

$$e_{MLS}(x,a) = \sum_{p_i \in \mathscr{P}} \theta(\|p_i - x\|)\langle a, p_i - x\rangle^2. \tag{9}$$

Comparing (7) with this equation, we have substituted the quantity q for x and have removed the dependence on r. This gives an approximation error for each choice of plane passing through x with normal a. If we omit the polynomial fitting step from Levin's procedure (which is equivalent to fitting a constant polynomial in step 2), the point x belongs to the MLS surface if there exists some query point r which projects to x via Step 1 of Levin's MLS projection procedure. Amenta and Kil prove that this condition is satisfied if there exists unique normal n_x minimizing e_{MLS} at the point x, and if x is a local minimum of $e_{MLS}(x,n_x)$ restricted to the line with direction n_x. These types of surfaces are known as *extremal surfaces* [Guy and Medioni 1995; Medioni et al. 2000]. Associated with an extremal surface is a corresponding implicit surface, which is obtained by evaluating the local minimum criteria at every point:

$$f(x) = n_x \cdot \nabla_y \, e_{MLS}(y,n_x)|_x = 0$$

i.e. the directional derivative along n_x of e_{MLS} evaluated at x is zero. The assumption here is that x belongs to a domain where n_x is uniquely defined and consistently oriented. With this definition, Amenta and Kil introduce a modified projection procedure to search for local minima of e_{MLS}. Given a query point $r_0 \in \mathbb{R}^3$, we compute the estimated normal n_{r_0} and search along the line in the direction n_{r_0} for a local minimum of e_{MLS}. The resulting local minimum becomes the new query point r_1, and the process is repeated until it converges (i.e. $\|r_{i+1} - r_i\| < \varepsilon$ for some small $\varepsilon \in \mathbb{R}$). The key difference between this procedure and Levin's procedure is that

1. The projection is iterated until it converges to a point on the MLS surface.

2. In Step 1 of Levin's procedure, we needed to solve a nonlinear optimization problem to obtain a estimated hyperplane; in Amenta and Kil's approach the normal estimate is obtained by calculating a best-fit hyperplane through the query point r. Since the weights θ are constant in the optimization with each choice of r, this results in a linear eigenvalue problem.

5.2 Improvements and Extensions

Based upon this work, Alexa et al. [2001] explore the potential to represent surfaces entirely using points. The idea here is to use *only* the point set as the surface representation, without using a triangle mesh or an implicit function. Although this was not possible before, the MLS method allows us to structure an point set with an implicitly defined non-parametric surface representation. This underlying representation enables a resampling of the surface. First, downsampling is achieved by iteratively removing point samples with smallest contribution to the MLS surface up to an error threshold. Second, upsampling is achieved by placing additional points where there is a large spacing between points. This is accomplished by locally computing a Voronoi diagram, finding the Voronoi vertex with the largest distance to points, and projecting that vertex to the MLS surface. The process yields a point set with bounded distance between samples. We can also use a point set to visualize the surface by splatting the points (see Figure 25) [Rusinkiewicz and Levoy 2000].

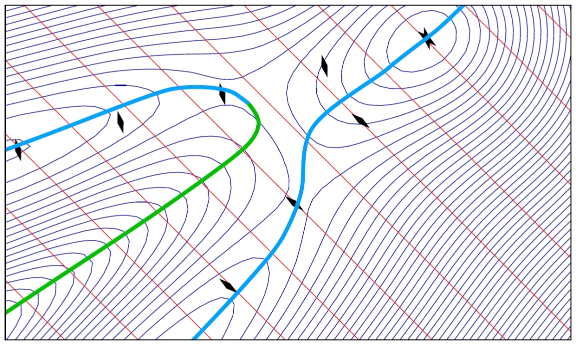

Figure 26: Trouble with a sharp corner. The blue line denotes local minima of the energy function; the green line local maxima. From [Amenta and Kil 2004b].

An important assumption in Amenta and Kil's work is the existence of a uniqueness domain $U \subseteq \mathbb{R}^3$ where n_x is uniquely determined and allows a consistent orientation [Amenta and Kil 2004b]. Recall that a similar assumption was in Levin's work [Levin 2003]. This domain seems to be affected by several factors, including:

1. The weighting function θ. The choice of h in the function $\theta(x) = e^{-x^2/h^2}$ determines the width of the neighborhood around the sample points where a good n_x can be estimated [Alexa et al. 2003]. It also determines the smoothness of the resulting surface; too large a value of h will tend to smooth fine-scale features.

2. The sampling density and noise level of the points. A poor sampling can yield holes in the domain, and a noisy point set will increase the approximation error of the normal. However, a larger h in our weighting function can be used to accommodate these problems, in exchange for smoothing out the surface [Lee 2000; Alexa et al. 2003].

3. Derivative discontinuities of the underlying surface (see Figure 26). A sharp corner has an ambiguous surface normal which is difficult to estimate [Amenta and Kil 2004b; Amenta and Kil 2004a].

A variety of techniques have been used to address these problems. Lee considered the problem of reconstructing a curve from a noisy point cloud [Lee 2000]. Noise makes it difficult to apply MLS to obtain a good estimate for the normal direction and also remove unwanted neighboring points. Lee solves this problem by constructing a Euclidean minimum spanning tree (EMST) for the point set, which provides a rough estimate of the connectivity information in the point set. The nearby points are gathered by traversing the EMST, which helps to find connected structures and discard points that are close by but actually part of a different structure. To deal with noise, Lee computes the local correlation of the dataset, and

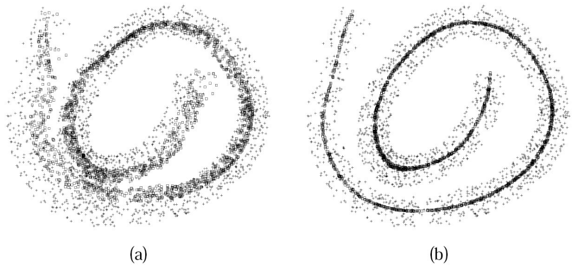

Figure 27: Curve reconstruction using MLS (a) and Lee's approach (b). The pluses are the input points and the squares are the projected points. From [Lee 2000].

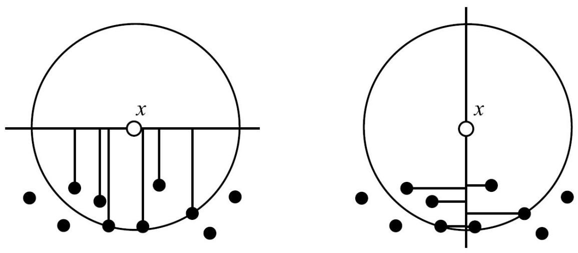

Figure 28: Problems with estimating the normal at a point. From [Amenta and Kil 2004a].

the width of the weight function h is increased until the correlation (measure of linearity of the point set) exceeds a user-defined threshold. This process is iterated until the approximation error is small. An example of applying this approach is in Figure 27.

In a follow-up paper, Amenta and Kil [2004a] attack this problem by changing the definition of the MLS error function. The observation is that the estimated normal can be parallel to the surface if the query point is sufficiently far away (see Figure 28) [Amenta and Kil 2004b]. Here, e_{MLS} is a weighted sum of the distances from the data points to their closest point on the plane, so Amenta and Kil consider an alternative definition of the error function. Intuitively, their error function integrates the distance from the plane to the the dataset along points on the plane. This would avoid a case like Figure 28 because if the plane is perpendicular to the surface, then the points of the plane far away from the dataset would be highly penalized. Mathematically we can describe this idea as:

$$e_I(x,a) = \int_{y \in H(x,a)} \delta^2(y)\theta(y,x)$$

where

$$\delta(y) = \sum_{p_i \in \mathscr{P}} \|y - p_i\|^2 \theta_N(x, p_i),$$

and θ_N is the normalized version of a weighting function θ. This variant not only gives reliable normal estimates, but also is able to deal better with corners (see Figure 29).

Fleishman et al. [2005] extends the MLS surface to accommodate sharp features by separating the dataset into multiple smooth regions. They apply a technique called forward search [Atkinson and Riani 2000; Hadi 1992] to classify the point set into one or more smooth subsets. The surface is described as the "intersection" of these smooth regions, resulting in a piecewise smooth MLS surface that is able to describe sharp corners. The forward search technique takes an initial smooth subset and incrementally grows it by

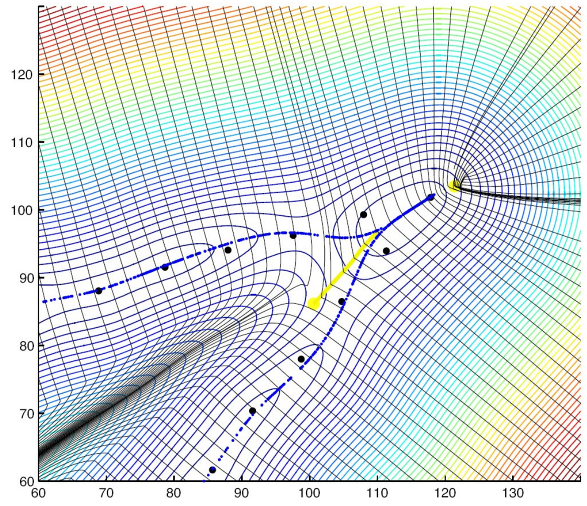

Figure 29: Projection with the new MLS error function for a sharp corner. The blue points are the projected points. From [Amenta and Kil 2004a].

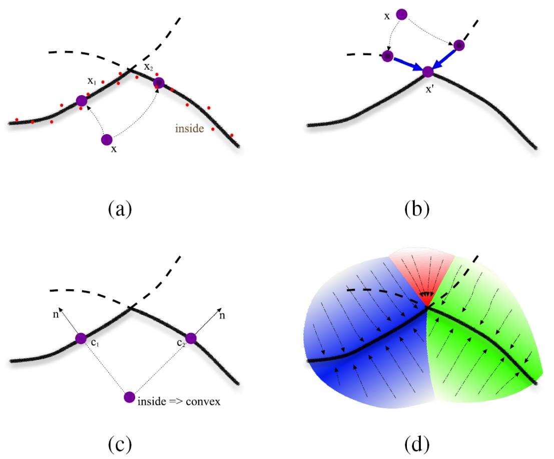

Figure 30: Projection with multiple smooth regions. A query point is projected to each of the two surfaces, and the closest projected point is chosen in (a). If both the projected points lie outside the intersection of the two regions, then the intersection of the tangent planes at the projected points are used to compute the location of intersection in (b). Inside/outside testing is done with the estimated normals in (c), and the regions where points are projected are shown in (d). From [Fleishman et al. 2005].

fitting a surface model (a bivariate polynomial of degree two) to the points, and adding the point with the smallest residual (difference between estimated and measured point) to the smooth region. If the smallest residual exceeds some threshold, then the remaining points are classified as outliers, and we can use the forward search algorithm to classify them into one or more smooth regions as well. This procedure is illustrated in Figure 30. Although this work allows for sharp features in the surface, it is not clear how well the technique deals with noisy point sets.

There has been a flurry of research activity investigating the use of point sets as an alternative surface representation. The goal is to use point sets to create and edit surfaces as well as rendering the surface without converting the point set into a triangle mesh or implicit surface. Zwicker et al. [2002] and Pauly et al. [2003] give techniques to directly model and edit point set data. Pauly et al. [2002] considers the problem of simplifying a point set, and Fleishman et al. [2003] introduce a hierarchy of point sets with increasing level of detail. Point sets can also be rendered using ray-tracing [Adamson and Alexa 2003]. The MLS surface is extended to include boundaries in the recent papers by Adamson and Alexa [2003; 2006]. Also, a mesh can be directly generated from the MLS surface by incrementally growing the mesh over the surface [Scheidegger et al. 2005] or projecting vertices of a coarse mesh [Sharf et al. 2006]. In addition, there has been some theoretical work concerning the error and a better choice of neighborhood size *h* [Lipman et al. 2006].

5.3 Future Work on MLS surfaces

Some future avenues of research include better estimation of normals, comparing surfaces resulting from different extremal surfaces, and theoretical work on the error and convergence of the method. Many methods depend on normals or off-surface points [Shen et al. 2004] [Carr et al. 2001] [Ohtake et al. 2003], and the MLS method is unique in that it does not require such information. However, the MLS method can benefit from normal information [Amenta and Kil 2004b], and a consistent, reliable way of estimating normals could improve the quality of the resulting surface. One idea is to directly fit local non-linear models to the data set, which may improve over the tangent plane estimation in the original MLS method. The challenge here is to find an appropriate non-linear model that fits the surface well without overfitting the data.

An important parameter that influences the quality of the surface is the choice of neighborhood around the query point. In previous work, the neighborhood was selected to be the *k*-nearest neighbors [Pauly et al. 2002] or the points within a certain radius of the query point [Alexa et al. 2001]. Just like how Fleishman et al. [2005] were able to accommodate sharp features by intelligently selecting point sets, a good technique to select the neighborhood in three-dimensions (perhaps by traversing a graph [Lee 2000]) could improve the quality of the resulting surface.

6 Discussion

We first summarize the user-specified parameters of the methods in Table 2. As a general trend, we observe here that there is a fundamental trade-off between the accuracy of the reconstruction and robustness of the method. Almost all of the methods have a parameter to control the smoothness of the resulting surface, which trades off with computation time, with surface fidelity (reducing artifacts and unwanted features), and with robustness to error and noise.

Kazhdan et al. [2006] have performed a direct comparison of their method with several methods we have discussed. The quality of the reconstruction varies quite a bit, which suggests that either the method is sensitive to the quality of the input (amount of noise) or sensitive to the user-specified parameter values. Among the examples shown in Figure 31, the best method seems to be VRIP, which generates a complete surface with no visible artifacts. It is curious that they have not been able to generate a correct surface with each of these methods; the Stanford Bunny is a common example and it has been successfully recovered using each of these methods. Perhaps the data input format may have been in favor of this method (a raw dataset assembled from range scans); in any case this comparison gives an idea about the practicality of these algorithms.

Comparing the running times of the methods is a difficult task, because the work spans ∼10 years of research, and the computing

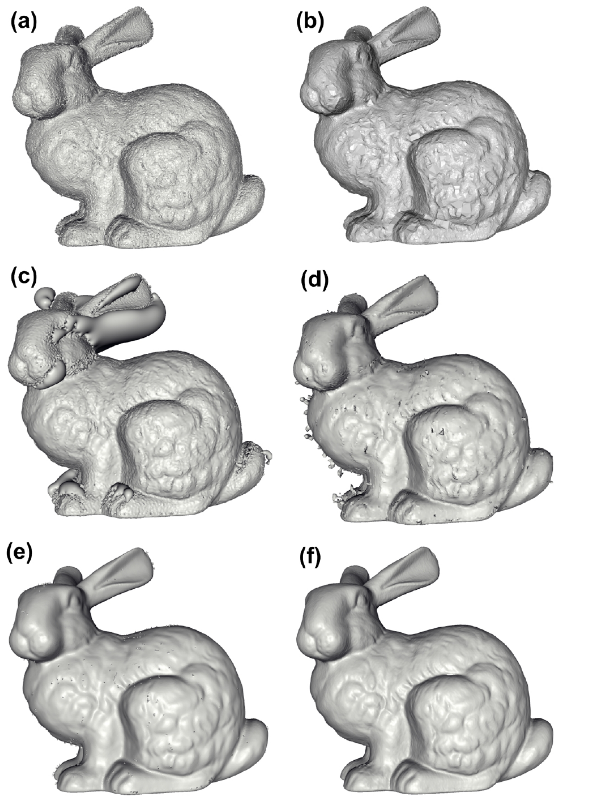

Figure 31: Reconstructing the Stanford bunny using: (a) Power Crust [Amenta et al. 2001], (b) Robust Cocone [Dey and Giesen 2001], (c) FastRBF [Carr et al. 2001], (d) MPU [Ohtake et al. 2003], (e) Hoppe et al. [1992], and (f) VRIP [Curless and Levoy 1996]. Figure from [Kazhdan et al. 2006].

environment has changed dramatically in that span of time. Although we have not implemented the algorithms and performed actual experiments, Kazhdan et al. [2006] has performed a running time comparison with the Stanford Bunny dataset. Here, the slowest method was FastRBF, where the execution time was on the order of ∼1 hour, followed by Power Crust and Robust Cocone, which ran on the order of ∼10 minutes, and the fastest being MPU, Hoppe et al. [1992], and VRIP, which ran on the order of ∼1 minute.

We also compare the methods using the following high-level criteria, summarized in Table 3:

- *Normals:* Does the method require normals?
- *Noisy Data:* Is the method robust to noisy data?
- *Approx:* Does the method produce an approximating or an interpolating surface?
- *Fill holes:* Does the method fill holes (due to insufficient sampling) in the surface?
- *Smooth:* Does the method produce a smooth surface?
- *Sharp:* Can the method preserve sharp features?
- *Guarantee:* Does the method provide a guarantee to preserve topology?
- *Global/Local:* Does the method solve for the entire surface at once (global) or does it compute localized fits to the data?

An ideal surface reconstruction algorithm would require no normals, be robust to noisy data, be able to produce both interpolating

Method	Parameters	Description	Comment
Tangent plane estimation	k	Number of nearest neighbors	Affects the smoothness of the approximation
	ρ, δ	Density and noise of the dataset, respectively	Used for determining boundaries and the voxel size for marching cubes
VRIP	w_i	Weights for each range scan	Represents variation across range surfaces, used to combine signed distances
	$D_{\min}, D_{\max}$	Range of the signed distance functions	Used for finding zero-crossing, hole-filling
α-shapes	α	Maximum size of a Delaunay simplex	"Carving" parameter that determines the shape of the complex
Crust & variants	r	Sampling density of the dataset	Provides theoretical guarantee for the reconstruction
	t	Normal filtering threshold	Threshold for throwing out triangles that differ too much from the estimated normals
Power Crust	λ	Estimate of the r-sampling density	Estimates level of noise to discard corresponding poles
	α	Intersection angle between the polar balls	Threshold for propagating pole labels
	ε	Threshold for detecting unstable poles	Used for medial axis simplification
	δ	Threshold for detecting redundant poles	Used for medial axis simplification
Blobby Model	α	Strength of the normal constraint	Controls the fit to the surface normals
	β	Strength of the shrink constraint	Constrains the resulting surface to avoid spurious features
	n	Surface normals estimated from the depth images	Used in formulating the normal constraint
	T	Isosurface value	Function value of the extracted isosurface
FastRBF	ε	Fitting accuracy	Controls the accuracy of RBF evaluation
	n	Surface normals	Required at each point to generate off-surface points
	ρ	Surface approximation parameter	Affects the smoothness of the resulting surface
EigenRBF	λ_e	Strength of the energy term	Controls the overall magnitude of the function
	λ_∇	Strength of the gradient term	Trades off smoothness vs. fidelity in the result
	λ_Ω	Strength of the regularization term	Constrains the smoothness of the function
MPU	α	Size of the support radius for the POU	Adjusted to cover the support of each octree cell
	n	Surface normals	Used for approximating signed distance (fitting local shapes) and detecting corners
	ε	Accuracy threshold	Octree continues subdivision until the fitted error is below this threshold
IMLS	ε	Weight function parameter	Controls the degree of interpolation / approximation of the surface
	γ	Adjustment rate parameter	Determines the rate at which the function values f_k are adjusted
	f_k	Function value constraints	Adjusted via γ to produce a tightly enclosing implicit function
MLS	h	Width of the weight function	Determines the support of the weight function and smoothness of the result
	m	Degree of the local approximating polynomial	Affects fitting accuracy and overfitting
	ε	Accuracy parameter for point projection	Determines the convergence limit for iterated projection onto the surface
Integral MLS variant	h	Width of the weight function	Determines the support of the weight function
	e_h	Change in h	Used to modulate the width of the weight function for searching the best value
Piecewise smooth MLS	r_t	Threshold of the maximal tolerated residual	In the forward search, determines which points are outliers
	L_s	Minimal neighborhood size for a surface	Used to select a neighborhood of points to fit the surface

Table 2: Listing of user-specified parameters for each method.

Method	**Normals**	**Noisy Data**	**Approx**	**Fill holes**	**Smooth**	**Sharp**	**Guarantee**	**Global / Local Computation**
Ideal	*No*	*Yes*	*Both*	*Yes*	*Yes*	*Yes*	*Yes*	*Any*
Tangent plane estimation	No	Probably	Approx	No	Probably	No	No	Global
VRIP	No	Yes	Approx	Yes	Probably	No	No	Global
α-shapes	No	No	Interp	No	No	Yes	No	Global
Crust & variants	No	No	Interp	No	No	Yes	Yes	Global
Power Crust	No	No	Interp	Yes	No	Yes	Yes	Global
Blobby Model	Yes	Probably	Approx	Yes	Yes	No	No	Global
FastRBF	Yes	Yes	Both	Yes	Yes	No	No	Global
EigenRBF	No	Yes	Approx	Yes	Yes	No	No	Global
MPU	Yes	No	Approx	Probably	Piecewise	Yes	No	Local
IMLS	Yes	Probably	Both	Yes	Yes	Yes	No	Local
Point-based IMLS	Yes	Probably	Both	Yes	Yes	Yes	Yes	Local
MLS	No	Yes	Approx	Yes	Yes	No	No	Local
MLS curve reconstruction	No	Yes	Approx	Yes	Yes	No	No	Local
Integral MLS variant	No	Yes	Approx	Yes	Yes	Probably	No	Local
Piecewise smooth MLS	No	Yes	Approx	Yes	Piecewise	Yes	No	Local

Table 3: Comparison of surveyed methods.

and approximating surfaces, be able to fill holes and preserve sharp features, and provide a guarantee that the reconstructed surface is homeomorphic to the original surface. Unfortunately, none of the techniques satisfy all of these criteria.

A potential avenue for future work is to combine the strength of different techniques to approach the ideal solution. The main strength of Voronoi-based methods is their provable reconstruction guarantee, while implicit surfaces can produce accurate interpolating and approximating surfaces with sharp features, and moving least squares surfaces are robust to noisy data. It may be fruitful to pre-process a noisy point set using moving least squares to produce a clean point set, and then use a Voronoi-based method to provide a topologically accurate estimation of the surface. In the final step, an implicit surface would be fit to provide a smooth surface accommodating sharp features and boundaries.

Another interesting problem is to reconstruct surfaces from time-varying point sets. The problem is interesting because the data may not necessarily maintain point-to-point correspondence between frames, and the sampling density or noise level may vary with time as well. The first challenge is to find a good surface model that can also describe how the data deforms over time. For this part, it may be possible to adapt mesh deformation models and apply them to deform point sets. The second challenge is to gather and integrate partial information about the surface from multiple frames. In particular, for poorly sampled regions, it may be possible to incorporate more surface points as the surface evolves over time.

7 Conclusion

In this report, we have surveyed many techniques that reconstruct a surface from point samples. We focused on four key ideas for surface reconstruction: signed distance estimation, Voronoi-based reconstruction, implicit surface fitting, and moving least squares surfaces. The main challenges have been identified, which include reconstruction without normal information, robustness to noise, accuracy to sharp features, and provable reconstruction guarantees. We hope to have had informed the reader about recent trends in this field to identify possible avenues for future work.

Acknowledgements I give thanks to God, who has graciously given me the wisdom and strength to tackle this examination. I am indebted to my adviser Matthias Zwicker for his aid in navigating the literature and for providing support during the writing of this report. Also, thanks to Wojciech Jarosz, Neel Joshi, and Wan-yen Lo for proofreading a draft of this paper.

References

ADAMSON, A., AND ALEXA, M. 2003. Approximating and intersecting surfaces from points. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 230–239.

ADAMSON, A., AND ALEXA, M. 2006. Point-sampled cell complexes. *ACM Trans. Graph. 25*, 3, 671–680.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2001. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, IEEE Computer Society, Washington, DC, USA, 21–28.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1, 3–15.

AMENTA, N., AND BERN, M. 1998. Surface reconstruction by voronoi filtering. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, ACM Press, New York, NY, USA, 39–48.

AMENTA, N., AND KIL, Y. J. 2004. The domain of a point set surface. *Eurographics Symposium on Point-Based Graphics*.

AMENTA, N., AND KIL, Y. J. 2004. Defining point-set surfaces. *ACM Trans. Graph. 23*, 3, 264–270.

AMENTA, N., BERN, M., AND EPPSTEIN, D. 1998. The crust and the beta-skeleton: combinatorial curve reconstruction. *Graph. Models Image Process. 60*, 2, 125–135.

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 415–421.

AMENTA, N., CHOI, S., DEY, T. K., AND LEEKHA, N. 2000. A simple algorithm for homeomorphic surface reconstruction. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, ACM Press, New York, NY, USA, 213–222.

AMENTA, N., CHOI, S., AND KOLLURI, R. K. 2001. The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, ACM Press, New York, NY, USA, 249–266.

ATKINSON, A., AND RIANI, M. 2000. *Robust Diagnostic Regression Analysis*. Springer.

BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22*, 4, 469–483.

BEATSON, R. K., AND GREENGARD, L. 1997. A short course on fast multipole methods. In *M. Ainsworth, J. Levesley, W.A. Light, and M. Marletta, editors, Wavelets, Multilevel Methods and Elliptic PDEs*, Oxford University Press, 137.

BEATSON, R. K., CHERRIE, J. B., , AND MOUAT, C. T. 1999. Fast fitting of radial basis functions: Methods based on preconditioned gmres iteration. *Advances in Computational Mathematics 11*, 253–270.

BEATSON, R. K., LIGHT, W. A., AND BILLINGS, S. 2000. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput. 22*, 5, 17171740.

BLINN, J. F. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3, 235–256.

BOLLE, R. M., AND VEMURI, B. C. 1991. On three-dimensional surface reconstruction methods. *IEEE Trans. Pattern Anal. Mach. Intell. 13*, 1, 1–13.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 67–76.

CAZALS, F., AND GIESEN, J. 2006. Delaunay triangulation based surface reconstruction. In *Effective Computational Geometry for Curves and Surfaces*, J.-D. Boissonnat and M. Teillaud, Eds., Mathematics and Visualization. Springer-Verlag.

CAZALS, F., GEISEN, J., PAULY, M., AND ZOMORODIAN, A. 2005. Conformal alpha shapes. *Eurographics Symposium on Point-Based Graphics*.

CLEVELAND, W. S., AND LOADER, C. L. 1996. Smoothing by local regression: Principles and methods. *Statistical Theory and Computational Aspects of Smoothing*, 10–49.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 303–312.

DEY, T. K., AND GIESEN, J. 2001. Detecting undersampling in surface reconstruction. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*, ACM Press, New York, NY, USA, 257–263.

DEY, T. K., AND SUN, J. 2005. Adaptive mls surfaces for reconstruction with guarantees. *SGP*.

DUCHON, J. 1976. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *R.A.I.R.O. Analyse Numrique 10*, 5–12.

EDELSBRUNNER, H., AND MÜCKE, E. P. 1994. Three-dimensional alpha shapes. *ACM Trans. Graph. 13*, 1, 43–72.

EDELSBRUNNER, H., FACELLO, M. A., FU, P., QIAN, J., AND NEKHAYEV, D. V. 1998. Wrapping 3D scanning data. In *Proc. SPIE Vol. 3313, p. 148-158, Three-Dimensional Image Capture and Applications, Richard N. Ellson; Joseph H. Nurre; Eds.*, R. N. Ellson and J. H. Nurre, Eds., 148–158.

FLEISHMAN, S., COHEN-OR, D., ALEXA, M., AND SILVA, C. T. 2003. Progressive point set surfaces. *ACM Trans. Graph. 22*, 4, 997–1011.

FLEISHMAN, S., COHEN-OR, D., AND SILVA, C. T. 2005. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3, 544–552.

GREENGARD, L., AND ROKHLIN, V. 1987. A fast algorithm for particle simulations. *Journal of Computational Physics 73*, 2, 325–348.

GUY, G., AND MEDIONI, G. 1995. Inference of surfaces, 3-d curves, and junctions from sparse 3-d data. 599–604.

HADI, A. 1992. Identifying multiple outliers in multivariate data. *J. R. Statist. Soc. B 54*, 3, 761–771.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 71–78.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 19–26.

HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., MCDONALD, J., SCHWEITZER, J., AND STUETZLE, W. 1994. Piecewise smooth surface reconstruction. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 295–302.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson surface reconstruction. *Eurographics Symposium on Geometry Processing*.

KOLLURI, R. 2005. Provably good moving least squares. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1008–1017.

LANCASTER, P., AND SALKAUSKAS, K. 1981. Surfaces generated by moving least squares methods. *Mathematics of Computation 37*, 155, 141–158.

LEE, I. 2000. Curve reconstruction from unorganized points. *Computer Aided Geometric Design 17*, 161–177.

LEVIN, D. 1998. The approximation power of moving least-squares. *Mathematics of Computation 67*, 224, 1517–1531.

LEVIN, D. 2003. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*.

LIPMAN, Y., COHEN-OR, D., AND LEVIN, D. 2006. Error bounds and optimal neighborhoods for mls approximation. In *Eurographics Symposium on Geometry Processing*.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 163–169.

MEDIONI, G. G., LEE, M.-S., AND TANG, C.-K. 2000. *A Computational Framework for Segmentation and Grouping*. Elsevier.

MENCL, R., AND MÜLLER, H. 1997. Interpolation and approximation of surfaces from three-dimensional scattered data points. In *DAGSTUHL '97: Proceedings of the Conference on Scientific Visualization*, IEEE Computer Society, Washington, DC, USA, 223.

MURAKI, S. 1991. Volumetric shape description of range data using "blobby model". In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 227–235.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3, 463–470.

PAULY, M., GROSS, M., AND KOBBELT, L. P. 2002. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, IEEE Computer Society, Washington, DC, USA.

PAULY, M., KEISER, R., KOBBELT, L. P., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM Trans. Graph. 22*, 3, 641–650.

PENTLAND, A. 1990. Computational complexity versus virtual worlds. *SIGGRAPH Comput. Graph. 24*, 2, 185–192.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., AND VETTERLING, W. T. 1988. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA.

REUTER, P. 2003. *Reconstruction and Rendering of Implicit Surfaces from Large Unorganized Point Sets*. PhD thesis, LaBRI - Université Bordeaux.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. QSplat: A multiresolution point rendering system for large meshes. In *Siggraph 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, K. Akeley, Ed., 343–352.

SCHEIDEGGER, C., FLEISHMAN, S., AND SILVA, C. 2005. Triangulating point-set surfaces with bounded error. In *Proceedings of the third Eurographics/ACM Symposium on Geometry Processing*, Eurographics Association, 63–72.

SHARF, A., LEWINER, T., SHAMIR, A., KOBBELT, L., AND COHENOR, D. 2006. Competing fronts for coarse-to-fine surface reconstruction. In *Eurographics*.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. 23*, 3, 896–904.

TURK, G., AND LEVOY, M. 1994. Zippered polygon meshes from range images. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 311–318.

WALDER, C., CHAPELLE, O., AND SCHOLKOPF, B. 2005. Implicit surface modeling as an eigenvalue problem. *ICML*.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop 3d: an interactive system for point-based surface editing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 322–329.