Multi-level Partition of Unity Implicits

Yutaka Ohtake
MPI Informatik

Alexander Belyaev *
MPI Informatik

Marc Alexa
TU Darmstadt

Greg Turk
Georgia Tech

Hans-Peter Seidel
MPI Informatik

Abstract

We present a new shape representation, the *multi-level partition of unity* implicit surface, that allows us to construct surface models from very large sets of points. There are three key ingredients to our approach: 1) piecewise quadratic functions that capture the local shape of the surface, 2) weighting functions (the partitions of unity) that blend together these local shape functions, and 3) an octree subdivision method that adapts to variations in the complexity of the local shape.

Our approach gives us considerable flexibility in the choice of local shape functions, and in particular we can accurately represent sharp features such as edges and corners by selecting appropriate shape functions. An error-controlled subdivision leads to an adaptive approximation whose time and memory consumption depends on the required accuracy. Due to the separation of local approximation and local blending, the representation is not global and can be created and evaluated rapidly. Because our surfaces are described using implicit functions, operations such as shape blending, offsets, deformations and CSG are simple to perform.

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

Keywords: partition of unity approximation, error-controlled subdivision, adaptive distance field approximation, implicit modeling.

1 Introduction

There are many applications that rely on building accurate models of real-world objects such as sculptures, damaged machine parts, archaeological artifacts, and terrain. Techniques for digitizing objects include laser rangefinding, mechanical touch probes, and computer vision techniques such as depth from stereo. Some of these techniques can yield millions of 3D point locations on the object that is being digitized. Once these points have been collected, it is a non-trivial task to build a surface representation that is faithful to the collected data. Some of the desirable properties of a surface reconstruction method include speed, low memory overhead, the creation of surfaces that approximate rather than interpolate the data (when noise or mis-registration is present), faithful reproduction of sharp features, and robustness in the presence of holes and low sampling density.

In this paper we introduce a new class of implicit models that was specifically designed to meet these requirements for rapidly and accurately creating surfaces from large collections of points. We use the name *Multi-level Partition of Unity* implicits (MPU)

Figure 1: The Stanford Lucy, consisting of 14 million points, is reconstructed as an MPU implicit with a 0.01% max-norm approximation accuracy; the left part of the model is colored according to the subdivision level which increases from blue to red. The four models in the back are reconstructed from the point set with increasing approximation error.

because at the heart of our method is a set of weighting functions that sum to one at all points in the domain. Given a set of points $\mathscr{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ sampled from a surface in $\mathbb{R}^3$, an MPU implicit $f : \mathbb{R}^3 \to \mathbb{R}$ provides an adaptive error-controlled approximation of the signed distance function from the surface. The approximation is accurate near the surface and rough far from the surface. The surface itself is then approximated by the zero level set of the distance function. We assume that the points of $\mathscr{P}$ are equipped with unit normals $\mathscr{N} = \{\mathbf{n}_1, \ldots, \mathbf{n}_N\}$ that indicate the surface orientation. In practice, these normals can be estimated either from initial scans during the shape acquisition phase or by local least-squares fitting to $\mathscr{P}$. We also consider the case when the surface is approximated by a mesh and $\mathscr{P}$ is the set of mesh vertices. Then the normals $\mathscr{N}$ are the mesh vertex normals.

To create our implicit representation, we start with a box that bounds the point set and create an octree-based subdivision of this box. At each cell of the octree, a piecewise quadratic function (the *local shape function*) is created that fits the points in the cell. These shape functions act much like a signed distance function, and take on the value zero near the data points and become positive (inside)

* Currently with the University of Aizu, Aizu-Wakamatsu, Japan.

or negative (outside) away from the data points. The approximate normals of the points are used to distinguish this inside/outside orientation locally. If the shape function approximation is not accurate enough (doesn't match the points well), the cell is subdivided and the procedure is repeated until a certain accuracy is achieved. Figure 1 shows how the octree-levels adapt according to the relation between local shape complexity and desired accuracy. In locations near the common boundary of two or more cells, the shape functions are blended together according to weights from the *partition of unity* functions. The global implicit function of the surface is given by this partition of unity blending of the local shape approximations at the octree leafs.

MPU implicits are conceptually simple, easy to implement, and are capable of providing a fast, accurate, and adaptive reconstructions of complex shapes from scattered point data containing millions of points. The complexity of the approach is output sensitive in the sense that the creation time and memory consumption depend on the complexity of the reconstructed shape rather than the number of data points. Since MPU implicits can deliver high-accuracy shape approximations, function-based operations such as shape blending, offsets, deformations and CSG can easily be applied. All of these same operations can be performed for data that was originally in a parametric or polygonal form simply by converting these shape descriptions to the MPU representation.

Previous work. Implicit shape representations are attractive because they allow a complex shape to be described by one formula, they unify surface and volume modeling, and several complex shape editing operations are easy to perform on such models [Bloomenthal et al. 1997]. On the other hand, traditional pure implicit surface modeling techniques lack local shape control. This drawback has become especially noticeable with the development of modern shape acquisition techniques that can generate data sets consisting of thousands, millions, or even billions of points (see, e.g., [Levoy et al. 2000]). The main advantages of using implicits for shape reconstruction from scattered data are data repairing capabilities and opportunities to edit the resulting objects using standard implicit modeling operations.

Most implicit shape reconstructions from point sets are based on Blinn's idea of blending local implicit primitives [Blinn 1982]. Muraki [1991] uses a linear combination of Gaussian blobs to fit an implicit surface to a point set. Hoppe et al. [1992] locally estimate the signed distance function as the distance to the tangent plane of the closest point. Lim et al. [1995] use blended union of spheres for implicit reconstruction of solids from scattered data. They obtain an initial configuration of spheres from the Delaunay tetrahedralization and a nonlinear optimization is then applied. Bajaj et al. [1995] and Bernardini et al. [1999] combine algebraic fitting with point data triangulation by adaptive α-shapes [Edelsbrunner and Mücke 1994]. A volumetric approach of Curless and Levoy [1996] introduced for shape reconstruction from range scans is based on estimating the distance function from a reconstructed model. Savchenko et al. [1995], Carr et al. [2001], and Turk and O'Brien [2002] use globally supported radial basis functions (RBFs) while Morse et al. [2001], Kojekine et al. [2003], and Ohtake et al. [2003] employ compactly supported RBFs to reconstruct smooth surfaces from point cloud data. It seems that the state-of-the-art in constructing implicit functions from large sets of scattered points are RBF-based methods [Carr et al. 2001; Dinh et al. 2002; Turk and O'Brien 2002]. While RBF-based methods are especially useful for the repair of incomplete data, they face serious difficulties in accurate reconstruction of sharp features [Dinh et al. 2001], may require a user intervention like choosing an appropriate carrier solid [Kojekine et al. 2003], and can generate extra zero-level sets [Ohtake et al. 2003]. In addition, since RBF solutions are global in nature, processing millions of points seems to be beyond the capabilities of most present day PCs.

The level set method [Zhao and Osher 2002] is another good candidate for reconstructing the signed distance function. However, its current implementation becomes expensive in time and memory if high accuracy reconstruction is required (although this might be improved if adaptive grids are used). Projection-based approaches to shape approximation [Alexa et al. 2001; Fleishman et al. 2003] have the advantage that they are local (i.e. independent of the number of data points) and directly yield a point on the surface. However, the projection step requires the solution of a non-linear moving least squares problem, which makes most practical shape operations expensive.

Our approach can be seen as a blend of several known techniques that, together, result in an attractive method for reconstructing an implicit function. One common RBF technique is to first divide the data domain into several cells so that the data is broken into manageable pieces [Beatson et al. 2000; Schaback and Wendland 2000; Iske 2001; Iske and Levesley 2002; Wendland 2002]. As a particular method for domain decomposition, the partition of unity approach (PU) of Franke and Nielson [1980] has been used as a general FEM method in computational mechanics [Babuška and Osborn 1994] and recently has become popular because it avoids the topological overhead of constructing a mesh [Babuška and Melenk 1997; Griebel and Schweitzer 2000; Griebel and Schweitzer 2002]. Our strategy for avoiding extra zero level sets is reminiscent of [Moore and Warren 1991; Warren 1992], where an adaptive and recursive volumetric subdivision was used. One could view our MPU representations as being similar to adaptively sampled distance fields [Frisken et al. 2000], with the difference that the MPU approach uses continuous rather than sampled functions.

When used with an appropriate choice of local shape approximations, our approach has the following attractive features: the ability to create high quality implicit surfaces from very large point datasets, the accurate reconstruction of sharp features, and fast and easy local shape access.

2 Partition of Unity Approach

The partition of unity approach is typically used to integrate locally defined approximants into a global approximation. Important properties such as the maximum error and convergence order are inherited from the local behavior. The basic idea of the partition of unity approach is to break the data domain into several pieces, approximate the data in each subdomain separately, and then blend the local solutions together using smooth, local weights that sum up to one everywhere on the domain.

More specifically, consider a bounded domain Ω in a Euclidean space (we will work in 3D) and a set of nonnegative compactly supported functions $\{\varphi_i\}$ such that

$$\sum_i \varphi_i \equiv 1 \text{ on } \Omega.$$

Let us associate a local approximation set of functions V_i with each subdomain $\text{supp}(\varphi_i)$. Now an approximation of a function $f(\mathbf{x})$ defined on Ω is given by

$$f(\mathbf{x}) \approx \sum_i \varphi_i(\mathbf{x}) Q_i(\mathbf{x}), \tag{1}$$

where $Q_i \in V_i$.

Given a set of nonnegative compactly supported functions $\{w_i(\mathbf{x})\}$ such that

$$\Omega \subset \bigcup_i \text{supp}(w_i),$$

partition of unity functions $\{\varphi_i\}$ can be generated by

$$\varphi_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{j=1}^{n} w_j(\mathbf{x})}. \tag{2}$$

Approximation by means of Eqs. 1 and 2 constitutes the core of the partition of unity finite element methods [Babuška and Osborn 1994]. They resemble the Modified Shepard's method of Franke

and Nielson [1980] (see also [Renka 1988]), where polynomial local approximations $Q_i(\mathbf{x})$ are used in combination with "inverse-distance" singular weights $\{w_i(\mathbf{x})\}$ for interpolation purposes.

Given a set of scattered points $\mathscr{P}$ equipped with normals $\mathscr{N}$, we approximate the signed distance function $f(\mathbf{x})$ from $\mathscr{P}$. In contrast to the approaches mentioned above, we introduce an adaptive procedure for generating the subdivision and problem-specific approximation sets. First, we use an octree-based adaptive space subdivision of Ω. This allows us to control the error of the approximation while adapting the complexity of the representation to the complexity of the shape (see Section 3). Second, we use *piecewise* quadratic functions resulting from Boolean operations for the accurate representation of sharp features. The classification of local shapes and appropriate approximation sets are discussed in Section 4.

For approximation purposes we use the quadratic B-spline $b(t)$ to generate weight functions

$$w_i(\mathbf{x}) = b\left(\frac{3\left|\mathbf{x}-\mathbf{c}_i\right|}{2R_i}\right) \tag{3}$$

centered at $\mathbf{c}_i$ and having a spherical support of radius R_i.

If an interpolation of $\mathscr{P}$ is required, we use the inverse-distance singular weights [Franke and Nielson 1980; Renka 1988]

$$w_i(\mathbf{x}) = \left[\frac{\left(R_i - \left|\mathbf{x}-\mathbf{c}_i\right|\right)_+}{R_i\left|\mathbf{x}-\mathbf{c}_i\right|}\right]^2, \text{ where } (a)_+ = \begin{cases} a & \text{if } a>0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

3 Adaptive Octree-based Approximation

The algorithm for constructing an MPU implicit is driven by repeated subdivision of the region of space that is occupied by the input set of points. First, the points in $\mathscr{P}$ are rescaled so that an axis-aligned bounding cube has a unit-length main diagonal. We then apply an adaptive octree-based subdivision to the bounding cube. Consider a cubic cell that was generated during the subdivision process, and let $\mathbf{c}$ be the center and d the length of the main diagonal of the cell.

We define the support radius R for the weight function (3) for the cell to be proportional to its diagonal:

$$R = \alpha d. \tag{5}$$

We typically use $\alpha = 0.75$. A larger value for α yields better (smoother) interpolation and approximation results at the expense of computation time. Time complexity is roughly quadratic in α. Figure 11 illustrates the effect of α, especially for the accurate approximation of the distance function away from the zero level set.

For each cell generated during the subdivision process, a local shape function $Q(x)$ is built using a least-squares fitting procedure, as shown in the left drawing of Figure 2.

Sometimes (especially if the density of $\mathscr{P}$ is not uniform) the ball of radius R for a cell does not contain enough points for a robust estimation of $Q(x)$. If the number of points is less than $N_{\min}$ (in our implementation we set $N_{\min} = 15$), a ball with increased radius $\hat{R}$ is determined that contains at least $N_{\min}$ points. This is done by starting with $\hat{R} = R$ and then iterating

$$\hat{R} = \hat{R} + \lambda R \tag{6}$$

until the ball contains the minimum number of points (in our implementation we set $\lambda = 0.1$). Now the points enclosed by the ball of radius $\hat{R}$ are used to estimate a local shape function $Q(\mathbf{x})$, as demonstrated in the right drawing of Figure 2.

We use a *kd*-tree partitioning for efficient solving of these range searching problems.

If the ball around an octree cell with initial radius $R = \alpha d$ is empty, an approximation of the distance function is computed as explained above and it will not be subdivided further. Otherwise, a local max-norm approximation error is estimated according to the Taubin distance [Taubin 1991]

$$\varepsilon = \max_{|\mathbf{p}_i-\mathbf{c}|<R} \left|Q\left(\mathbf{p}_i\right)\right| / \left|\nabla Q\left(\mathbf{p}_i\right)\right|. \tag{7}$$

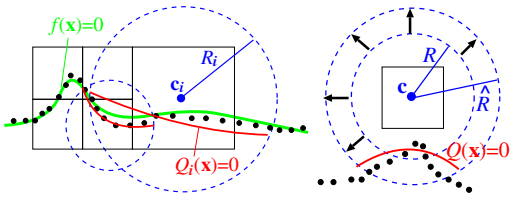

Figure 2: Left: adaptive subdivision coupled with least-squares fitting. Right: enlarging the spherical domain for the local approximation to make it more robust.

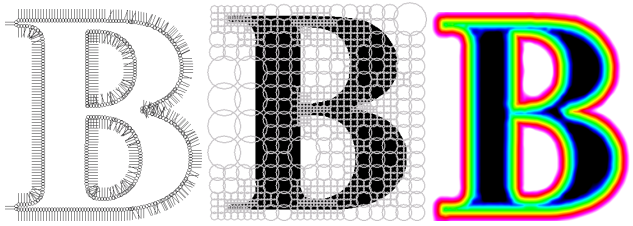

Figure 3: Left: a set of points equipped with normals. Middle: the circles (2D balls) corresponding to the adaptive subdivision are shown here at 50% of their real size. Right: the distance function is reconstructed, and the zero level set is located between the yellow and green bands.

If ε is greater than a user-specified threshold ε_0, the cell is subdivided and the fitting process is performed for the child cells.

Figure 3 demonstrates how our adaptive subdivision scheme works in 2D.

The following pseudocode describes a recursive procedure for assembling an MPU approximation at point $\mathbf{x}$ with precision ε_0.

```
EvaluateMPUapprox(x, ε0)
  S_wQ = S_w = 0;
  root->MPUapprox(x, ε0);
  return S_wQ / S_w;

MPUapprox(x, ε0)
  if (|x − c_i| > R_i) then return;
  if (Q_i is not created yet) then
    Create Q_i and compute ε_i;
  if (ε_i > ε0) then
    if (No childs) then Create childs;
    for each child
      child->MPUapprox(x, ε0);
  else
    S_wQ = S_wQ + w_i(x) * Q_i(x);
    S_w = S_w + w_i(x);
```

Here, S_w and S_{wQ} denote $\sum w_i(\mathbf{x})$ and $\sum Q_i(\mathbf{x})w_i(\mathbf{x})$, respectively, see Eqs. 1 and 2.

This procedure is easy to implement. We hope that this will make our approach and its implementation accessible to a wide range of users.

Note that we abandon the local approximations that are constructed at the non-leaf cells of the octree. This allows us to use different local approximations of the distance function far from $\mathscr{P}$ and near to $\mathscr{P}$, as well as specific sharp feature approximations, without compensating for the effects of coarse approximations. Inheriting coarse approximations would also require us to counteract already generated zero-sets in empty balls. In addition, avoiding coarse approximations saves memory and results in faster evaluation of the implicit function.

4 Estimating Local Shape Functions

Our local fitting strategy depends on the number of points in the ball of a given cell and the distribution of normals of those points. At a given cell we use the most appropriate one of these three local approximations:

(a) a general 3D quadric,
(b) a bivariate quadratic polynomial in local coordinates,
(c) a piecewise quadric surface that fits an edge or a corner.

Roughly speaking, (a) is used to approximate larger parts of the surface, which could be unbounded or contain more than one sheet, (b) is used to approximate a local smooth patch, and (c) is employed

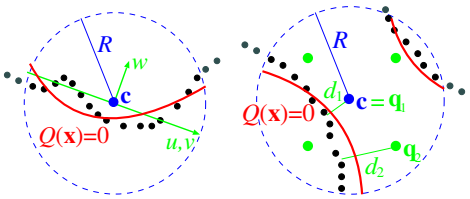

Figure 4: Left: fitting a bivariate quadratic polynomial. Right: local fit of a general quadric; auxiliary points located at the cell center and corners are used in order to achieve an accurate approximation of the distance function.

to reconstruct sharp features. Actually (c) consists of a number of tests (an edge test and corner tests) in order to determine the type of approximation surface that should be used.

A few simple tests are performed to select from among the three types of local shape functions. If there are more than $2N_{\min}$ points in the local ball, we use a function of type (a) or (b). An average normal direction is computed for the points and if the maximum deviation of normals to the average is more than $\pi/2$ then we fit with (a), otherwise we use type (b). If there are fewer than $2N_{\min}$ points associated with a cell, more detailed checks are made to see if an edge or corner is present, and details of this are given below. Why don't we look for sharp features when there are more than $2N_{\min}$ points? Because the sharp feature detection adds computational complexity and the octree subdivision procedure takes care of this, anyway. Should the surface actually contain a corner or edge near such a cell, then the quality-of-fit measure (7) will cause the cell to be divided, and the sharp feature will be fit by one or more child cells.

In the following sub-sections we will describe each of the three local shape functions in more detail. The notation we use in these sections is as follows. Let $\mathbf{c}$ be the center of a subdivision cubic cell where we want to construct a local shape function $Q(\mathbf{x})$. We denote by $\mathscr{P}'$ the points of $\mathscr{P}$ that are inside the ball of the cell. Let $\mathbf{n}$ be a unit normal vector assigned to $\mathbf{c}$. This normal $\mathbf{n}$ is computed from the normalized weighted arithmetic mean of the normals of $\mathscr{P}'$ taken with the weights defined by (3). Let θ be the maximal angle between $\mathbf{n}$ and the normals $\mathscr{N}'$ assigned to the points of $\mathscr{P}'$.

(a) Local fit of a general quadric. If

$$|\mathscr{P}'| > 2N_{\min} \quad \text{and} \quad \theta \geq \pi/2$$

a 3D quadratic surface is fitted. Usually this corresponds to a situation sketched in the right drawing of Figure 4. A local shape function is given by

$$Q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\,\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \tag{8}$$

where $\mathbf{A}$ is a symmetric 3×3 matrix, $\mathbf{b}$ is a vector of three components, and c is a scalar. In order to determine the unknowns $\mathbf{A}$, $\mathbf{b}$, and c we make use of auxiliary points $\{\mathbf{q}_i\}$ to help orient the local shape function. These points are chosen as the corners and the center of the subdivision cell, as demonstrated in the right picture of Figure 4.

Each auxiliary point $\mathbf{q}$ is tested for whether it can be used to obtain a reliable estimate of the signed distance function. For each $\mathbf{q}$, its six nearest neighbors $\mathbf{p}^{(1)},\mathbf{p}^{(2)},\ldots,\mathbf{p}^{(6)}$ from $\mathscr{P}'$ are found and the scalar products

$$\mathbf{n}^{(i)}\cdot\left(\mathbf{q}-\mathbf{p}^{(i)}\right), \quad i=1,2,\ldots,6, \tag{9}$$

are computed. If not all the scalar products have the same sign, $\mathbf{q}$ is removed from the set of auxiliary points. The geometric idea behind this test is explained by the left drawing of Figure 5.

If the set of remaining auxiliary points is empty, the cell is subdivided.

For each remaining auxiliary point $\mathbf{q}$, an average distance d, the arithmetic mean of the scalar products (9), is computed

$$d = \frac{1}{6}\sum_{i=1}^{6}\mathbf{n}^{(i)}\cdot\left(\mathbf{q}-\mathbf{p}^{(i)}\right). \tag{10}$$

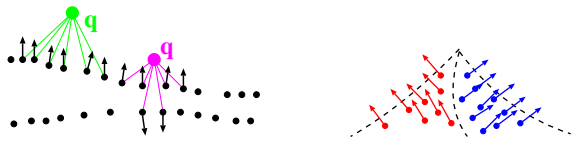

Figure 5: Left: Testing whether an auxiliary point $\mathbf{q}$ can be used to obtain a reliable estimate of the signed distance function. The green $\mathbf{q}$ is reliable, but the magenta $\mathbf{q}$ is not. Right: Detection of sharp features is done by clustering of point normals.

Finally the unknowns in (8) are found by minimizing

$$\frac{1}{\sum w(\mathbf{p}_i)}\sum_{\mathbf{p}_i\in\mathscr{P}'} w(\mathbf{p}_i)\,Q(\mathbf{p}_i)^2 + \frac{1}{m}\sum_{i=1}^{m}\left(Q(\mathbf{q}_i)-d_i\right)^2, \tag{11}$$

where m is the number of remaining points $\mathbf{q}$.

(b) Local fit of a bivariate quadratic polynomial. If

$$|\mathscr{P}'| > 2N_{\min} \quad \text{and} \quad \theta < \pi/2$$

a bivariate quadratic polynomial is locally fitted. Let us introduce local coordinates (u,v,w) with the origin of coordinates at $\mathbf{c}$ such that the plane (u,v) is orthogonal to $\mathbf{n}$ and the positive direction of w coincides with the direction of $\mathbf{n}$. A quadratic shape function at $\mathbf{c}$ is given by

$$Q(\mathbf{x}) = w - \left(Au^2 + 2Buv + Cv^2 + Du + Ev + F\right), \tag{12}$$

where (v,u,w) are the coordinates of $\mathbf{x}$ in the new coordinate system. The unknown coefficients A, B, C, D, E, and F are determined by minimizing

$$\sum_{\mathbf{p}_i\in\mathscr{P}'} w(\mathbf{p}_i)\,Q(\mathbf{p}_i)^2. \tag{13}$$

The left drawing of Figure 4 illustrates the geometric idea behind local fitting of a bivariate quadratic polynomial.

(c) Local approximation of edges and corners. The quadratic functions (12) and (8) considered above are not capable of accurately capturing sharp edges and corners. If there are just a few points associated with a cell ($|\mathscr{P}'| \leq 2N_{\min}$), we try to fit a *piecewise* quadratic function instead of a quadratic approximation.

We perform automatic recognition of edges and corners using a simple but effective procedure proposed by Kobbelt et al. [2001]. The idea is based on clustering of normals, as demonstrated by the right drawing of Figure 5.

Following [Kobbelt et al. 2001] we assume that the surface has a sharp feature if

$$\min_{i,j}\left(\mathbf{n}_i\cdot\mathbf{n}_j\right) < 0.9. \tag{14}$$

If (14) is not satisfied, we go to (b) and local bivariate polynomial (12) is fitted to $\mathscr{P}'$. Otherwise we check whether the detected feature is a corner. We consider $\mathbf{n}_3 = \mathbf{n}_1\times\mathbf{n}_2$, the normal vector to the plane determined by the normals $\mathbf{n}_1$ and $\mathbf{n}_2$ enclosing the maximal angle. If the deviation of the normals $\mathbf{n}_i$ from the plane is sufficiently large

$$\max_i\left|\mathbf{n}_i\cdot\mathbf{n}_3\right| > 0.7 \tag{15}$$

the feature is recognized as a corner.

If (14) is satisfied and (15) is not, we expect the surface to have an edge and subdivide the set of normals $\{\mathbf{n}_i\}$ into two clusters according to their angles with $\mathbf{n}_1$ and $\mathbf{n}_2$ (two spherical Voronoi subsets corresponding to $\mathbf{n}_1$ and $\mathbf{n}_2$). Denote by $\mathscr{P}'_1$ and $\mathscr{P}'_2$, $\mathscr{P}' = \mathscr{P}'_1\cup\mathscr{P}'_2$, two subsets corresponding to the clusters of the normals. Now the quadratic fit procedure is applied separately to $\mathscr{P}'_1$ and $\mathscr{P}'_2$ and a non-smooth local shape function approximation $\mathscr{P}'$ is constructed via the max / min Boolean operations of Ricci [1973].

If (14) and (15) are satisfied, we subdivide $\mathscr{N}'$ into three sets. First $\mathscr{N}'_1$ and $\mathscr{N}'_2$ are constructed as above. Next we check whether $\left|\mathbf{n}_{1,2}\cdot\mathbf{n}_i\right| < \left|\mathbf{n}_3\cdot\mathbf{n}_i\right|$ and add point $\mathbf{p}_i$ to the third cluster if the inequality is satisfied.

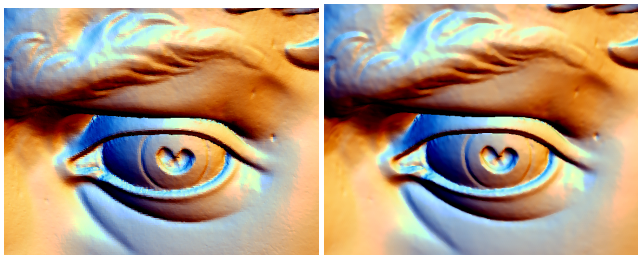

Figure 6: Left: Eye from Stanford's reconstruction of Michalangelo's David (scanned at 1mm resolution). Right: The eye is reconstructed as an MPU implicit with relative accuracy $\varepsilon_0 = 10^{-4}$.

We also treat separately corners of degree four: test (14) is applied to the normals of the third cluster and if (14) is satisfied, the cluster is subdivided into two pieces. If the resulting four clusters of normals correspond to either a convex or concave corner, it is reconstructed via Boolean operations. Otherwise, we go to (a) and a general quadric (8) is fitted to $\mathscr{P}'$.

If the number of points used to estimate the coefficients of bivariate quadratic polynomial (12) is less then six, we set all the unknown coefficients in (12) equal to zero.

Given the above approach, more complex types of sharp features (for example, a saddle corner of degree 4) are approximated by smooth functions. Notice however that "generic" sharp features are obtained from the intersections of two or three surfaces, and therefore consist of edges and corners of at most degree 3. It is not a generic event for four smooth surfaces to intersect at one point.

5 Visualization

Conventional techniques for visualizing implicit models include polygonization (isosurface extraction), ray tracing, and volume rendering. From among these various visualization methods, we use Bloomenthal's polygonizer [Bloomenthal 1994] because of its nice continuation properties and the Hart sphere tracing method [Hart 1996]. These two methods both work well using the approximate distance functions of MPU implicits.

If our goal is to create a polygonal mesh, we can save time and memory by computing MPU approximations on the fly during the polygonization process. We have found that an approximation accuracy of $\varepsilon_0 = 10^{-4}$ (that is, 0.01% of the length of the main diagonal of the bounding box) is quite sufficient for the reconstruction of fine features, as demonstrated by Figure 6.

If a non-adaptive surface extraction routine is used with an implicit model that has sharp features, a fine sampling density is required to capture these features. An example of this can be seen in the top and bottom left images of Figure 7. An alternative is to use adaptive sampling and remeshing strategies such as those in [Kobbelt et al. 2001; Ohtake and Belyaev 2002; Ju et al. 2002]. We find it attractive to combine a low resolution Bloomenthal polygonization with a postprocessing mesh optimization technique developed in [Ohtake and Belyaev 2002], as shown in the top middle, top right, and bottom middle images of Figure 7.

Even higher quality rendering can be achieved using ray tracing techniques. The sphere tracing method of Hart [1996] works well together with MPU implicits since it uses the distance function representation and it is quite capable of rendering implicit models with sharp features. The bottom right image of Figure 7 shows an MPU implicit model of the fandisk model that was rendered with sphere tracing. The left image of Figure 8 shows sphere tracing of a more complex implicit model. This model was generated from a function-based operation (subtraction) applied to the dragon and the David model, both represented as MPU implicits. Notice how well the sharp features are reconstructed and rendered.

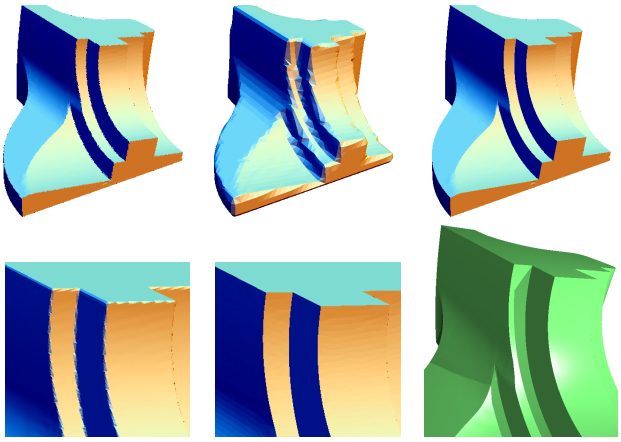

Figure 7: Visualization of the fandisk model implicitized with MPU. Top left: Bloomenthal polygonization was used; in spite of sufficiently high polygonization resolution (200K triangles) one can notice small aliasing defects along sharp features. Top middle: a low resolution polygonization is applied. Top right: an optimized mesh (17K triangles) is obtained from the low resolution mesh, the sharp features are accurately reconstructed. Bottom left: a magnified part of the high resolution mesh. Bottom middle: the same part of the optimized mesh. Bottom right: a part of the model rendered using Hart sphere tracing.

6 Results & Applications

In this section we discuss results and applications of approximating or interpolating MPU implicits for surface reconstruction from range scans and incomplete point data, and function-based operations.

Approximation and Interpolation. Most of the illustrations in this work have been generated using approximating MPU implicits as described in Section 4. However, our MPU approach can also be adapted to exact data point interpolation if we use a local interpolation method such as Franke and Nielson's singular weights (4) instead of (3).

Since MPU with (4) requires a deeper octree-based subdivision (every nonempty subdivision cell contains only one point of $\mathscr{P}$), our interpolation procedure requires more memory resources than the approximation one. For interpolation with singular weights (4) the subdivision stops only when all of points of $\mathscr{P}$ have been placed in their own cells. The ball around a nonempty octree cell is centered at the interpolated point inside the cell. We set $\alpha = 1.25$ in (5) in order to ensure that we cover the bounding box by the balls around the octree cells.

For each cell containing one interpolated point $\mathbf{p}$ of $\mathscr{P}$, its local shape function $Q(\mathbf{x})$ is defined by (12), where the origin of coordinates of local coordinate system (u,v,w) is located at $\mathbf{p}$ (hence $F = 0$ in (12)) and the positive direction of w coincides with the direction of the averaged normal at $\mathbf{p}$. We don't use the normal of $\mathscr{N}$ assigned to $\mathbf{p}$ because of stability problems common for Hermite-like interpolation schemes. The unknown coefficients are determined by minimizing quadratic energy (13) with $\mathbf{c} = \mathbf{p}$. Now (1) interpolates $\mathscr{P}$ because partition of unity functions $\{\varphi_i(\cdot)\}$ defined by (2) satisfy

$$\varphi_i(\mathbf{p}_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad \text{and} \quad \nabla\varphi_i(\mathbf{p}_j) = 0.$$

The right image of Figure 8 shows results of applying the MPU interpolation and shape modeling operations (Boolean subtraction, twisting) to the Stanford Buddha model.

Reconstruction from incomplete data. Reconstruction from scattered point data with MPU implicits is robust with respect to variations of point density, as demonstrated in Figure 9.

Reconstruction from range scans. MPU implicits can be used to reconstruct 3D models from a collection of range scans. We have

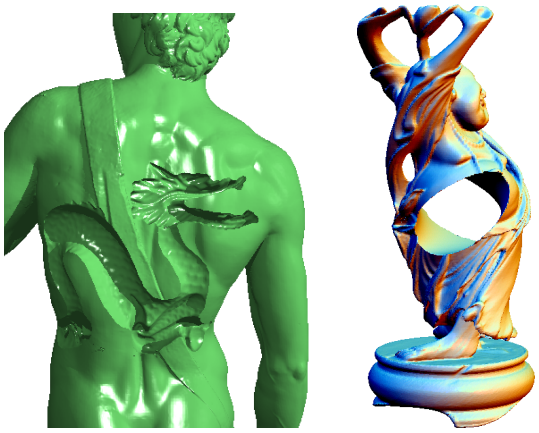

Figure 8: CSG operations applied on MPU implicits. Left: sphere tracing of the subtraction of two MPU approximations. Right: boolean subtraction and twisting operations are applied to interpolating MPU implicits.

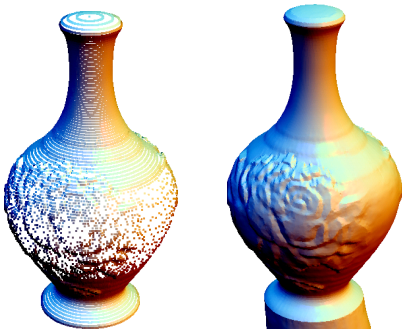

Figure 9: Reconstruction from a scattered point dataset with non-uniform density of points.

found that if several range scans overlap, better results are obtained if we take into account per-point measurement confidences during the reconstruction process. If we treat all points the same, artifacts can arise. If the accuracy threshold (7) is small, the MPU implicit approximating the scan points can have wrinkles in the overlapping regions. On the other hand, if (7) is not small enough, the MPU implicit does not capture the fine geometric details of the scanned model. In practice, a given position on the object can be measured more accurately from some scanning directions than from others. This notion of using confidence during surface reconstruction was advocated in [Turk and Levoy 1994; Curless and Levoy 1996].

Consider a collection of points from range data. Assume that each point $\mathbf{p}_i$ is assigned a confidence weight c_i, $c_i \in [0,1]$, that were computed based on scanning information according to the rules suggested in [Curless n. d.]. Now the MPU reconstruction process described in previous sections is enhanced by the modifications given below.

- For a better estimation of local shape function $Q(\mathbf{x})$, if the sum of the confidence measures of the points inside the ball is less than $N_{\min}$ then the radius growth rule (6) is applied repeatedly until the sum is above this threshold.
- Instead of (7), a weighted accuracy measure is used: $\varepsilon = \max c_i\, Q(\mathbf{p}_i)/|\nabla Q(\mathbf{p}_i)|$.
- The unit normal vector $\mathbf{n}$ of the base plane (u,v) used to fit the bivariate quadratic polynomial (12) is obtained by averaging the neighboring normals with weights $c_i w(\mathbf{p}_i)$.
- Weights $\{c_i w(\mathbf{p}_i)\}$ are used in (13) and (11) instead of $\{w(\mathbf{p}_i)\}$.
- The normals in (10) are taken with the confidence weights assigned to their corresponding points.

Figure 10 demonstrates the MPU reconstruction of the Stanford bunny from the original range scans. In one case we have used only six scans, and in the other case we have used the full ten range scans. Notice the ability of the MPU method to repair missed data.

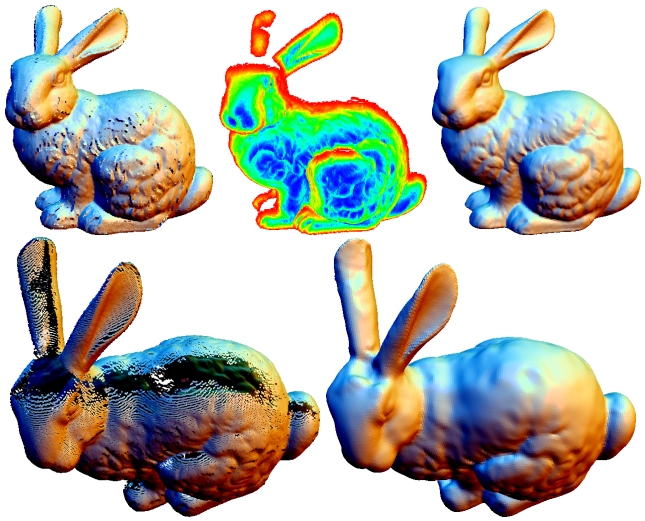

Figure 10: Reconstruction of Stanford bunny from range data. Top left: bunny scan data is rendered as a cloud of points, (all ten original range scans are used); defects caused by low accuracy of some points and normals are clearly visible. Top middle: a side range image of bunny is colored according to the confidence measure. Top right: bunny is reconstructed as an MPU implicit. Bottom left: only six range scans of the bunny scan data are rendered (an example of incomplete data). Bottom right: an MPU implicit bunny from six scans.

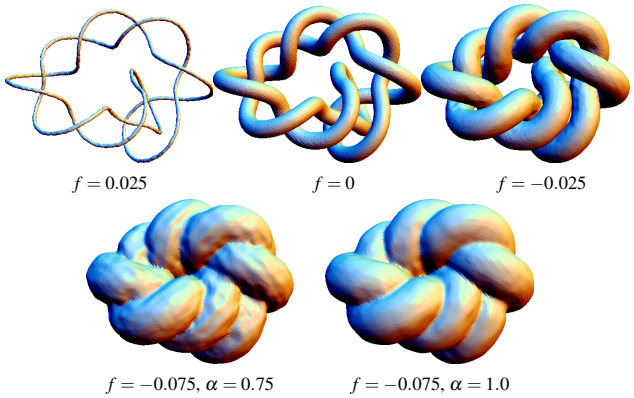

Figure 11: Offsetting of a knot model. The distance function to the knot is approximated by $w = f(x,y,z)$. The first four models were generated with $\alpha = 0.75$. For the last model $\alpha = 1$ was used and a higher quality offsetting was produced.

Function-based shape modeling operations. Using MPU implicits allow us to extend the power of function-based shape modeling operations [Bloomenthal et al. 1997] to point set surfaces. Given several MPU functions defined over the same bounding box and having possibly different octree structures, at each point of the box we evaluate the value of the result of applying functional operations to the functions. Then the level sets of the resulting function are visualized via a polygonization or ray tracing.

An example of a CSG operation applied to two large and complex point set surfaces was already demonstrated in Figure 8. Results of offsetting, smooth blending, morphing, and twisting operations [HyperFun: F-rep Library n. d.], [Pasko and Savchenko 1994] are shown in Figures 11, 12, 13, 14 and the right image of Figure 8. In particular, Figures 13 and 14 demonstrate a linear morphing of two implicit models.[1]

[1] The linear morphing of implicit models $w = f(x,y,z)$ and $w = g(x,y,z)$ is an implicit model defined by $w = (1-t)f(x,y,z) + tg(x,y,z)$.

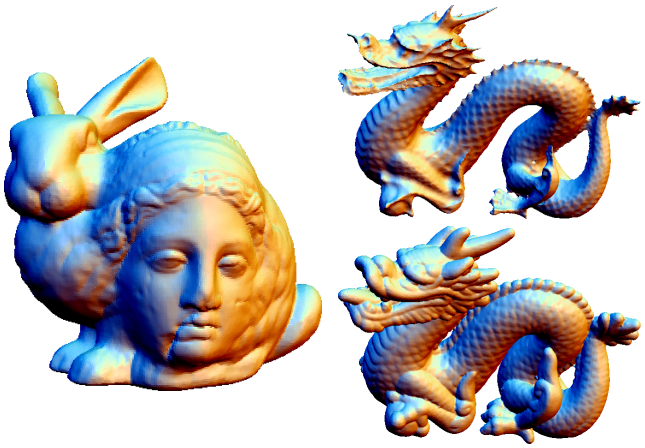

Figure 12: Left: Smooth blending of the Stanford bunny and Cyberware Igea models. Right: offsetting of the Stanford dragon model; $f = 0.0075$ (top) and $f = -0.01$ (bottom); $\alpha = 0.75$.

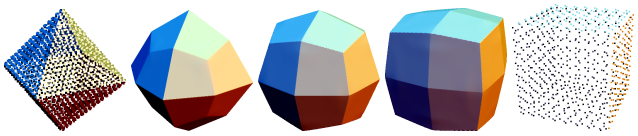

Figure 13: Linear blending of octahedron and cube.

7 Discussion

This paper describes a new implicit surface representation based on local shape functions, partitions of unity, and an octree hierarchy. Strengths of the Multi-Level Partition of Unity formulation include:

- Fast surface reconstruction and rendering.
- Representation of sharp features.
- Reconstruction from incomplete data.
- Choice of either approximation or interpolation of the data and the ability to adaptively vary the approximation accuracy.

Given a point set model processed by the MPU method with a specified accuracy, the computational time and memory usage depends on the geometric complexity of the model: the higher the geometric complexity, the deeper the octree is subdivided. This is clearly demonstrated by Figure 1 where the reconstruction of fine features requires a deeper subdivision.

Table 1 presents RAM memory usage and computational time measurements for simultaneous generating and polygonizing various point set models. Note that our method is quite fast. Our experiments with state-of-the-art RBF-based 3D surface reconstruction techniques such as FastRBF [Carr et al. 2001] and others suggest that the MPU method is considerably faster than these other techniques. [2]

Model	Number of points	Relative error	Peak RAM	Number of triangles	Comp. time
Bunny	34,611	2.5×10^{-3}	34 MB	91,104	0:07
Bunny scans	362,272	1.0×10^{-3}	110 MB	219,676	1:46
Dragon	433,375	8.0×10^{-4}	195 MB	819,704	1:39
Buddha	543,625	0.0	442 MB	648,332	6:53
David (2mm)	4,124,454	1.0×10^{-4}	810 MB	1,296,522	10:33

Table 1: Memory and computational time measurements for generation + polygonization of MPU implicits for various point set models. Computations were performed on a 1.6GHz Mobile Pentium 4 with 1GB RAM, and timings are listed as minutes:seconds.

[2] Comparing the results of Table 1 with those of Table 2 in [Carr et al. 2001] one can find that the MPU method is 20-30 times faster than the FastRBF technique [Carr et al. 2001].

Figure 14: Linear morphing of two head models approximated by MPU implicits, Max Planck Head and Head of Michelangelo's David.

Notice that processing time for the Buddha model is more than three times longer then that for the dragon model which has a similar size. This is because we reconstruct the Buddha by the MPU interpolation which requires a deeper subdivision and wider support for the corresponding partition of unity functions.

Because of its local nature, the MPU method is more sensitive to the quality of input data, especially the field of normals, to compare with the approximation and interpolation techniques based on globally-supported RBFs [Carr et al. 2001; Turk and O'Brien 2002]. Nevertheless, according to our experiments, the MPU method is sufficient for accurate shape reconstruction from a wide variety of data sets. The parameters in our current implementation of the MPU approach are adjusted for processing typical outputs of laser scanner devices. We believe that the parameter modifications needed for different classes of input are fairly intuitive in order to handle scattered data of lower (higher) quality at the expense of lower (higher) computational speed.

Unlike the crust approach [Amenta et al. 1998], the MPU method is not supported by rigorous results guaranteeing correct reconstruction of input data satisfying certain properties described mathematically. It is a price we pay for a high speed of our method.

We would like also to stress here that our method is not an RBF technique. RBF is a global approximation/interpolation method because of its global variational nature. Our method is a local one. We make use of *two* functions, the partition-of-unity weights and the local piecewise quadratic approximation functions, which is different than the single function used by an RBF approach. This two-function approach gives benefits such as sharp feature reconstruction that, to date, have not been possible using RBFs.

Smoothness properties of the MPU implicits are determined by those of weight functions (3). Choosing smoother weight functions will produce smoother MPU implicits.

Similar to other implicit function shape representation schemes, the MPU implicits are not capable of representing correctly surfaces with boundaries.

We see a number of opportunities to improve our approach. Other estimation of the distance function might be beneficial. The distance function is a ruled surface with singularities, therefore using quadratic functions to approximate the distance function is not optimal. A richer library of local shape approximations could be generated in order to reconstruct accurately more complex sharp features. Finally, the MPU approach should be well suited to an out-of-core implementation due to the local nature of the weighting functions.

Acknowledgments

The models are courtesy of the Digital Michelangelo Project 3D Model Repository (Michelangelo's David and Head of Michelangelo's David), the Stanford 3D Scanning Repository (Lucy, bunny, dragon, and Buddha), Cyberware (Igea, fandisk, and vase), Max-Planck-Institut für Informatik (Head of Max Planck), and the Imager Computer Graphics Laboratory of the University of British Columbia (knot).

References

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2001. Point set surfaces. In *IEEE Visualization 2001*, 21–28.

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of ACM SIGGRAPH 1998*, 415–421.

BABUŠKA, I., AND MELENK, J. M. 1997. The partition of unity method. *International Journal of Numerical Methods in Engineering 40*, 727–758.

BABUŠKA, CALOZ, G., AND OSBORN, J. E. 1994. Special finite element methods for a class of second order elliptic problems with rough coefficients. *SIAM J. Numerical Analysis 31*, 4, 745–981.

BAJAJ, C. L., BERNARDINI, F., AND XU, G. 1995. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Proceedings of ACM SIGGRAPH 95*, 109–118.

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, 1717–1740.

BERNARDINI, F., BAJAJ, C., CHEN, J., AND SCHIKORE, D. 1999. Automatic reconstruction of 3D CAD models from digital scans. *International Journal of Computational Geometry & Applications 9*, 4, 327–369.

BLINN, J. F. 1982. A generalization of algebraic surface drawing. *ACM Transactions on Graphics 1*, 3 (July), 235–256.

BLOOMENTHAL, J., BAJAJ, C., BLINN, J., CANI-GASCUEL, M. P., ROCKWOOD, A., WYVILL, B., AND WYVILL, G. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann.

BLOOMENTHAL, J. 1994. An implicit surface polygonizer. In *Graphics Gems IV*. 324–349.

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 *Proceedings of ACM SIGGRAPH 2001*, 67–76.

CURLESS, B. VripPack User's Guide. http://graphics.stanford.edu/software/vrip/.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of ACM SIGGRAPH 1996*, 303–312.

DINH, H. Q., SLABAUGH, G., AND TURK, G. 2001. Reconstructing surfaces using anisotropic basis functions. In *International Conference on Computer Vision (ICCV) 2001*, 606–613.

DINH, H. Q., TURK, G., AND SLABAUGH, G. 2002. Reconstructing surfaces by volumetric regularization. *IEEE Trans. on Pattern Analysis and Machine Intelligence 24*, 10 (October), 1358–1371.

EDELSBRUNNER, H., AND MÜCKE, E. P. 1994. Three-dimensional alpha shapes. *ACM Transactions on Graphics 13*, 1 (January), 43–72.

FLEISHMAN, S., COHEN-OR, D., ALEXA, M., AND SILVA, C. T. 2003. Progressive point set surfaces. *ACM Transactions on Graphics 22*, 4 (October).

FRANKE, R., AND NIELSON, G. 1980. Smooth interpolation of large sets of scattered data. *International Journal of Numerical Methods in Engineering 15*, 1691–1704.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A., AND JONES, T. R. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of ACM SIGGRAPH 2000*, 249–254.

GRIEBEL, M., AND SCHWEITZER, M. A. 2000. A Particle-Partition of Unity Method for the solution of Elliptic, Parabolic and Hyperbolic PDE. *SIAM J. Sci. Comp. 22*, 3, 853–890.

GRIEBEL, M., AND SCHWEITZER, M. A. 2002. A Particle-Partition of Unity Method – Part III: A Multilevel Solver. *SIAM J. Sci. Comp. 24*, 2, 377–409.

HART, J. C. 1996. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 527–545.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized point. In *Proceedings of ACM SIGGRAPH 1992*, 71–78.

HYPERFUN: F-REP LIBRARY. http://cis.k.hosei.ac.jp/ F-rep/HF_lib.html.

ISKE, A., AND LEVESLEY, J. 2002. Multilevel scattered data approximation by adaptive domain decomposition. Tech. rep., University of Leicester, April.

ISKE, A. 2001. Hierarchical scattered data filtering for multilevel interpolation schemes. In *Mathematical methods for curves and surfaces (Oslo, 2000)*. Vanderbilt Univ. Press, Nashville, TN, 211–221.

JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. 2002. Dual contouring of hermite data. *ACM Transactions on Graphics 21*, 3 (July), 339–346. Proceedings of ACM SIGGRAPH 2002.

KOBBELT, L. P., BOTSCH, M., SCHWANECKE, U., AND SEIDEL, H.-P. 2001. Feature sensitive surface extraction from volume data. In *Proceedings of ACM SIGGRAPH 2001*, 57–66.

KOJEKINE, N., HAGIWARA, I., AND SAVCHENKO, V. 2003. Software tools using CSRBFs for processing scattered data. *Computers & Graphics 27*, 2 (April).

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The Digital Michelangelo Project: 3D scanning of large statues. In *Proceedings of ACM SIGGRAPH 2000*, 131–144.

LIM, C., TURKIYYAH, G. M., GANTER, M. A., AND STORTI, D. W. 1995. Implicit reconstruction of solids from cloud point sets. In *Proceedings of the third ACM symposium on Solid Modeling and Applications*, ACM Press, 393–402.

MOORE, D., AND WARREN, J. 1991. Approximation of dense scattered data using algebraic surfaces. In *Proceedings of the 24th Hawaii International Conference on System Sciences*, IEEE Computer Society Press, Kauai, Hawaii, 681–690.

MORSE, B. S., YOO, T. S., RHEINGANS, P., CHEN, D. T., AND SUBRAMANIAN, K. R. 2001. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Shape Modeling International 2001*, 89–98.

MURAKI, S. 1991. Volumetric shape description of range data using "Blobby Model". *Computer Graphics 25*, 4 (July), 227–235. Proceedings of ACM SIGGRAPH 1991.

OHTAKE, Y., AND BELYAEV, A. G. 2002. Dual/primal mesh optimization for polygonized implicit surfaces. In *7th ACM Symposium on Solid Modeling and Applications*, 171–178.

OHTAKE, Y., BELYAEV, A. G., AND SEIDEL, H.-P. 2003. A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. In *Shape Modeling International 2003*. Accepted.

PASKO, A., AND SAVCHENKO, V. 1994. Blending operations for the functionally based constructive geometry. In *Set-theoretic Solid Modeling: Techniques and Applications, CSG 94 Conference Proceedings*, Information Geometers, 151–161.

RENKA, R. J. 1988. Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software 14*, 2 (June), 139–148.

RICCI, A. 1973. A constructive geometry for computer graphics. *The Computer Journal 16*, 2 (May), 157–160.

SAVCHENKO, V. V., PASKO, A. A., OKUNEV, O. G., AND KUNII, T. L. 1995. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum 14*, 4, 181–188.

SCHABACK, R., AND WENDLAND, H. 2000. Adaptive greedy techniques for approximate solution of large RBF systems. *Numerical Algorithms 24*, 239–254.

TAUBIN, G. 1991. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence 13*, 11, 1115–1138.

TURK, G., AND LEVOY, M. 1994. Zippered polygon meshes from range images. In *Proceedings of ACM SIGGRAPH 1994*, 311–318.

TURK, G., AND O'BRIEN, J. 2002. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics 21*, 4 (October), 855–873.

WARREN, J. 1992. Free-form blending: a technique for creating piecewise implicit surfaces. In *Topics in Surface Modeling*, H. Hagen, Ed. SIAM Press, Philadelphia, 473–483.

WENDLAND, H. 2002. Fast evaluation of radial basis functions: Methods based on partition of unity. In *Approximation Theory X: Wavelets, Splines, and Applications*, Vanderbilt University Press, Nashville, L. Schumaker and J. Stöckler, Eds., 473–483.

ZHAO, H., AND OSHER, S. 2002. Visualization, analysis and shape reconstruction of unorganized data sets. In *Geometric Level Set Methods in Imaging, Vision and Graphics*, Springer, S. Osher and N. Paragios, Eds.