Eurographics Symposium on Geometry Processing 2009
Marc Alexa and Michael Kazhdan
(Guest Editors)

Volume 28 (2009), Number 5

Smoothing of Partition of Unity Implicit Surfaces for Noise Robust Surface Reconstruction

Yukie Nagai, Yutaka Ohtake and Hiromasa Suzuki

The University of Tokyo, Japan

Abstract

We propose a novel method for smoothing partition of unity (PU) implicit surfaces consisting of sets of non-conforming linear functions with spherical supports. We derive new discrete differential operators and Laplacian smoothing using a spherical covering of PU as a grid-like data structure. These new differential operators are applied to the smoothing of PU implicit surfaces. First, Laplacian smoothing is performed for the vector field defined by the gradient of the PU implicit surface, which is then updated to reflect the smoothing of the gradient field. This process achieves a method for noise robust surface reconstruction from scattered points.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling, Curve, surface, solid, and object representations—surface reconstruction, implicit surface, smoothing

1. Introduction

Surface reconstruction from points sampled on the surface of an object is a very important process in mechanical engineering and computer graphics. Many algorithms have been proposed because of the various demands of the relevant areas of application. Previous algorithms can be classified into two groups based on their approach: explicit methods and implicit methods.

Explicit methods. A typical method in this category involves Delaunay-based algorithms, which are intuitive but need dense sampling with little noise. Recently, a number of algorithms with noise robustness have appeared (e.g. the Power Crust [ACK01] by Amenta et al., the Robust Cocone [DG06] by Dey and Goswami, and the Eigencrust algorithm [KSO04] by Kolluri et al.). However, even these are more sensitive to noise than implicit methods. Moreover, algorithms in this category tend to be time-intensive because of their global nature.

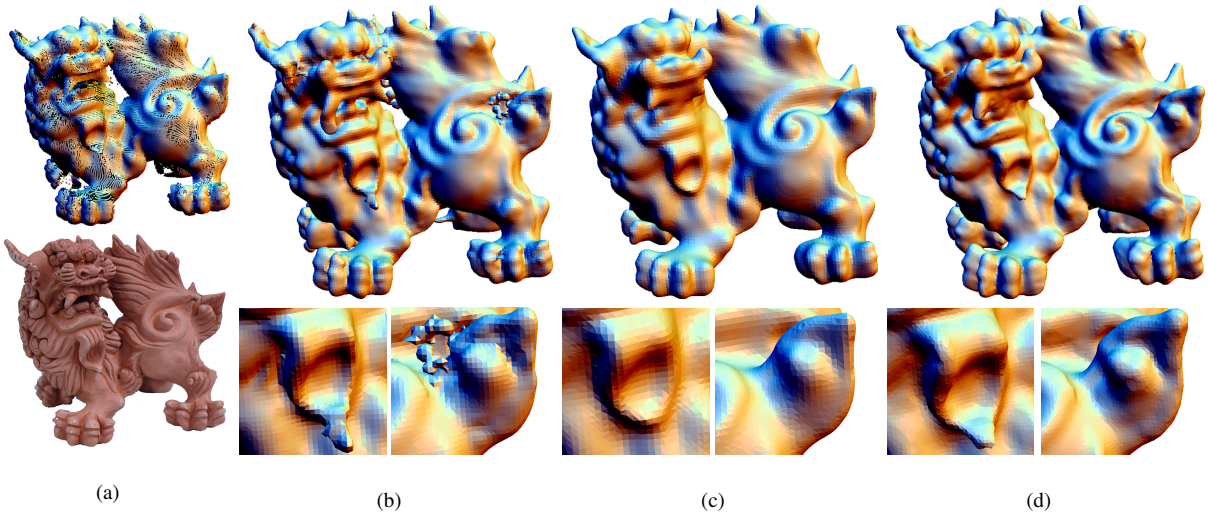
Implicit methods. Nowadays, implicit approaches are a major force in surface reconstruction because of their stability with noise and their low computational costs.

With implicit methods, the space is divided into the inside and outside of the object, and a boundary surface is then defined between the internal and external points. If necessary, the surface is approximated using a triangular mesh generated by a polygonizer such as Marching cubes [LC87], dual contouring [JLSW02], and so on. Algorithms in this group are categorized according to whether the method they are based on is local or global.

Local implicit methods. The notable advantages of methods in this category include highly accurate representation, fast computation and the ability to handle large data sets. Algorithms in this group began with VRIP [CL96], which provides high-resolution surfaces but struggles to handle misaligned surfaces. Other examples of this approach include algorithms based on moving least squares [ABCO*01, GG07] and its variants [AK04, ÖGG09]. the Multi-level Partition of Unity implicits (MPU) [OBA*03] proposed by Ohtake et al. excel in the above advantages.

Local operation offers many merits as outlined above, but also has some difficulties in handling very low-quality data such as that with large amounts of noise, outliers, lack of sampling, and large differences in sampling density. The resulting shapes are sometimes significantly deformed, and may also have holes and extra components. An example of the MPU is shown in Fig. 1(b).

Global implicit methods. Global implicit approaches have the advantage of being able to handle low quality data. Ex-



(a) (b) (c) (d)

Figure 1: *Scanned data ((a), top) for a terra-cotta Shiisa object((a), bottom); (b) model reconstructed using MPU [OBA*03]; (c) Poisson surface reconstruction [KBH06]; (d) reconstruction using our algorithm.*

amples of such methods include the Radial Basis Function approach [CBC*01] by Carr et al., the integration of Voronoi diagrams and the variational method [ACSTD07] by Alliez et al., the Graph-cut approach by Hornung and Kobbelt [HK06], and the finite element method by Sharf et al. [SLS*07]. The Poisson surface reconstruction technique proposed by Kazhdan et al. [KBH06] excels in computational speed and noise robustness. Unfortunately, however, the global implicit approach achieves noise robustness at the cost of high-accuracy reconstruction. The results of the Poisson surface reconstruction method shown in Fig. 1 (c) show this characteristic.

Our approach. Based on the discussions above, we propose a novel surface reconstruction algorithm that is essentially based on the local implicit method. Unfortunately, as outlined above, generating a smooth model using only a local method is very difficult because the techniques involved depend entirely on fitting approaches. To combat this, we accomplish reconstruction with smooth surfaces by introducing smoothing terms to the local implicit method.

Smoothing itself is a well-studied subject, and many methods have been proposed. Among them, Laplacian smoothing is intuitive and simple, so is followed by many extensions such as Taubin Smoothing [Tau95], bi-Laplacian smoothing [KCVS98] by Kobbelt et al., and mean-curvature flow [DMSB99] by Desbrun et al. Unfortunately, mesh smoothing cannot change the topology of a mesh, so extra components in the target mesh (see Fig. 6(b)) are not removed.

With this proposal, we achieve a precise and noise robust surface reconstruction algorithm that inherits the benefits of local implicit methods and smoothing. We adopt a partition of unity implicit as a local implicit method along with Laplacian smoothing. The partition of unity implicit generates an implicit function as a weighted average of spherically supported linear approximation functions. At this point, a spherically covered space has no differential operators, so we define such operators in a domain covered by spherical supports associated with linear functions. Our smoothing is based on the diffusion of a gradient field, and acts to smooth local approximations. The main idea is illustrated in Fig. 2. This smoothing can handle adaptively sampled data, and enables topological change.

The concept is inspired by normal-based smoothing methods [Tau01, OBS02, CT03] that smooth normals then move vertices. A similar method for grids is proposed by Tasdizen et al. [TWBO02] based on level set methods [Set99]. The novelty of our approach lies in its definition of differential operators on a set of spherical supports for linear functions.

Paper organization. In Section 2, we explain the spherical cover generated by partition of unity. Then in Section 3, differential operators defined on the spherical cover are derived. In Section 4, we apply these operators to smooth PU implicit functions. The preservation of detail and an overview of our algorithm are outlined in the same section. Section 5 presents our experimental results and discusses them and Section 6 concludes the paper and outlines future work.

2. Partition of Unity

In this section, we describe the details of a spherical cover made using the PU method. The structure of such a spherical

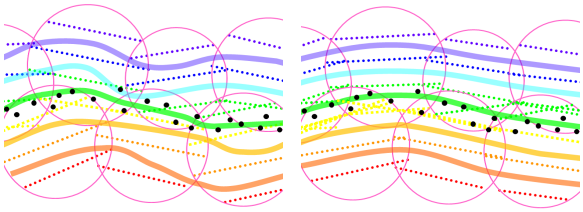


Figure 2: *Left: Isosurfaces generated using PU. Isosurfaces of local approximations are shown with dotted lines. Right: Isosurfaces after the smoothing of local approximations.*

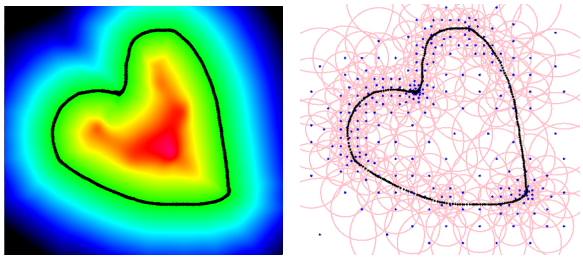


Figure 3: *An example of PU in 2D. Left: The scalar field determined by $f(\boldsymbol{x})$ and the reconstructed image (black line). Values of $f(\boldsymbol{x})$ indicated in red are high, and those in blue are low. Right: The supports of local approximations $\{g_i(\boldsymbol{x})\}$ are indicated with pink circles. The blue dots represent the centers of supports, and input points are shown with black dots.*

cover is described in Section 2.1, and its method of generation is explained in Section 2.2.

2.1. Spherical Cover

Given a set of points $\{\boldsymbol{p}_j\}$ sampled from a surface associated with oriented normals $\{\boldsymbol{n}_j\}$, we approximate the surface with the zero-level surface of signed distance function $f(\boldsymbol{x})$, as shown in the image on the left of Fig. 3. A domain including all the sampling points is covered by a set of spheres as shown in the image on the right of Fig. 3.

Spherical support s_i is defined by center $\boldsymbol{c}_i$ and radius r_i. s_i has the implicit linear function $g_i(\boldsymbol{x})$, whose support is s_i itself. $g_i(\boldsymbol{x})$ is represented as coefficient vector $\boldsymbol{\alpha}_i$, and constant term β_i as $g_i(\boldsymbol{x}) = \boldsymbol{\alpha}_i \cdot (\boldsymbol{x} - \boldsymbol{c}_i) + \beta_i$. The value of $g_i(\boldsymbol{x})$ locally approximates the signed distance of point $\boldsymbol{x}$ inside s_i from the surface. Isosurface $g_i(\boldsymbol{x}) = 0$ therefore approximates the sampling points inside s_i. $g_i(\boldsymbol{x})$ is obtained by least square fitting for sampling points inside s_i, then $\boldsymbol{\alpha}_i$ and β_i are described as

$$\boldsymbol{\alpha}_i = \frac{\sum_j w_i(\boldsymbol{p}_j)\boldsymbol{n}_j}{\|\sum_j w_i(\boldsymbol{p}_j)\boldsymbol{n}_j\|}, \quad \beta_i = \boldsymbol{\alpha}_i \cdot \left(\frac{\sum_j w_i(\boldsymbol{p}_j)(\boldsymbol{c}_i - \boldsymbol{p}_j)}{\sum_j w_i(\boldsymbol{p}_j)} \right). \tag{1}$$

These are the weighted averages of normals and coordinates of sampling points inside s_i. In this work, weighting function $w_i(\boldsymbol{x})$ is the second-degree B-spline $b_2\left(3\|\boldsymbol{x} - \boldsymbol{c}_i\|/2r_i\right)$.

Weighted average $f(\boldsymbol{x})$ of local approximations $\{g_i(\boldsymbol{x})\}$ represents the signed distance from arbitrary point $\boldsymbol{x}$ inside the domain to the surface of the object.

$$f(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})}{\sum_i w_i(\boldsymbol{x})} \tag{2}$$

This is the algorithm known as *Partition of Unity*, which is hereafter abbreviated as *PU*. A specific process to determine a spherical covering is described in Section 2.2.

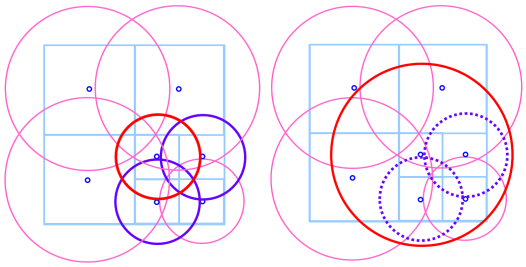


Figure 4: *Sphere expansion. Left: the spherical cover generated using MPU. The red sphere is to be expanded. Right: after expansion. The spheres inside the red sphere (shown in purple) are eliminated.*

2.2. Generation of Initial Spherical Cover

Here we outline the generation of a spherical cover for a domain including sampling points $\{\boldsymbol{p}_j\}$. We assume that the domain is rescaled so that the length of the longest edge of the bounding box of $\{\boldsymbol{p}_j\}$ is one. In this step, a spherical cover is generated essentially based on MPU, but the number of spheres is lower than the result produced by MPU itself.

In a similar way to MPU, the generation process begins by dividing the domain with an adaptive octree measuring local approximation errors. An octant is assigned a spherical support s_i whose center $\boldsymbol{c}_i$ is the center of the octant and whose radius r_i is 0.75 times the length of the octant's diagonal. Then, $g_i(\boldsymbol{x})$ is determined by (1). Let the approximation error of an octant be the maximum value of $|g(\boldsymbol{p}_j)|$ for sampling points inside the corresponding support. Each octant is divided recursively until its approximation error does not exceed user-specified tolerance ε.

After this step, we modify the resulting spherical cover

by expanding the radii of randomly selected spheres as long as their local approximation errors do not exceed ε. Spheres completely included inside an expanded sphere are eliminated. In Fig. 4, a 2D example shows the effect of this expansion, which helps to reduce the number of spheres. In our experiments, the final number of spheres is about half the number of octants.

After spherical covering using PU, to improve the smoothness of the reconstructed surface, implicit function $f(\boldsymbol{x})$ is smoothed by iteratively updating $\{\boldsymbol{\alpha}_i\}$ and $\{\beta_i\}$. New differential operators suitable for a spherical cover are defined in the next section.

3. Differential Operators for Partition of Unity

For smooth shape representation, Laplacian smoothing is used for local approximation functions. Since we are working on a set of spherical supports, using discrete differential operators specialized for a spherical cover is preferred than using general differential operators for a traditional orthogonal grid or a tetrahedral mesh [TLHD03]. Unfortunately, no differential operators for PU have yet been proposed. Hence, in this section we aim to define differential operators for a domain covered by a set of supports associated by linear functions.

The following calculations are performed only at the centers of spherical supports, so new operators are defined at their centers. To distinguish our discrete operators from traditional continuous ones, hereafter we denote new operators for centers as $\mathrm{Grad}(\cdot)$, $\mathrm{Div}(\cdot)$ and $\mathrm{Lap}(\cdot)$ corresponding to the traditional $\nabla(\cdot)$, $\nabla\cdot(\cdot)$ and $\triangle(\cdot)$, respectively.

We assume that each spherical support s_i is assigned a constant vector $\boldsymbol{v}_i$. Then, the vector at point $\boldsymbol{x}$ is evaluated using PU as follows:

$$\boldsymbol{v}(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x})\boldsymbol{v}_i}{\sum_i w_i(\boldsymbol{x})}. \tag{3}$$

Let us define $\mathrm{Div}(\boldsymbol{v}_i)$ as the average integration of $\nabla\cdot\boldsymbol{v}(\boldsymbol{x})$ inside s_i:

$$\frac{4\pi r_i^3}{3}\mathrm{Div}\,(\boldsymbol{v}_i) \equiv \int_{\boldsymbol{x}\in s_i} \nabla\cdot\boldsymbol{v}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{4}$$

By the divergence theorem, the right side of (4) is represented as an integration over sphere ∂s_i:

$$\frac{4\pi r_i^3}{3}\mathrm{Div}\,(\boldsymbol{v}_i) = \int_{\boldsymbol{x}\in\partial s_i} \boldsymbol{v}(\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \tag{5}$$

where $\boldsymbol{n}(\boldsymbol{x})$ is a unit normal at point $\boldsymbol{x}$ on the sphere. Here, we define a *neighbor* of s_i as a support intersecting with s_i. Let $\delta_{i,j}$ be the part of ∂s_i inside a neighbor s_j (indicated by the green dotted line in Fig. 5, which shows an illustration of a 2D version). For simplicity, we assume that $\delta_{i,j}$ is covered only by s_i and s_j, then $\boldsymbol{v}(\boldsymbol{x})$ in (5) is replaced with $\boldsymbol{v}_j$. Now,

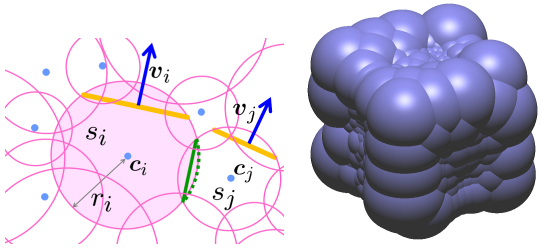


Figure 5: *Left: intersecting 2D spheres (considering the diffusion at center $\boldsymbol{c}_i$ of support s_i). The orange lines are local approximations, and the blue arrows are their normals. $\delta_{i,j}$ is indicated by the green dotted line, and the intersection disk with area $D_{i,j}$ is indicated by the solid green line. Right: an example of the adaptive spherical cover in 3D.*

the left side of equation (5) is approximated as follows:

$$\frac{4\pi r_i^3}{3}\mathrm{Div}(\boldsymbol{v}_i) \approx \frac{4\pi r_i^2}{\sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} d\boldsymbol{x}} \sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} \boldsymbol{v}_j\cdot\boldsymbol{n}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{6}$$

The value of this approximated area is greater than the true value of $4\pi r_i^2$ when neighbors overlap each other. Denominator $\int_{\boldsymbol{x}\in\delta_{i,j}}\mathrm{d}\boldsymbol{x}$ on the right-hand side normalizes this approximated area.

Calculating the integrals of (6), we obtain the divergence at $\boldsymbol{c}_i$

$$\mathrm{Div}(\boldsymbol{v}_i) = \frac{3}{r_i S_i}\sum_j D_{i,j}\,\boldsymbol{v}_j\cdot\frac{\boldsymbol{c}_j-\boldsymbol{c}_i}{\|\boldsymbol{c}_j-\boldsymbol{c}_i\|}. \tag{7}$$

In the above equation, $S_i = \sum_j \int_{\boldsymbol{x}\in\delta_{i,j}}\mathrm{d}\boldsymbol{x}$ and $D_{i,j}$ is the area of the disk whose boundary is the circle where ∂s_i and ∂s_j intersect. We define *intersection disk* as such a disk (see Fig. 5). For simpler notation, by introducing

$$\phi_{i,j} = \frac{D_{i,j}}{\|\boldsymbol{c}_j-\boldsymbol{c}_i\|}, \tag{8}$$

equation (7) can be rewritten as

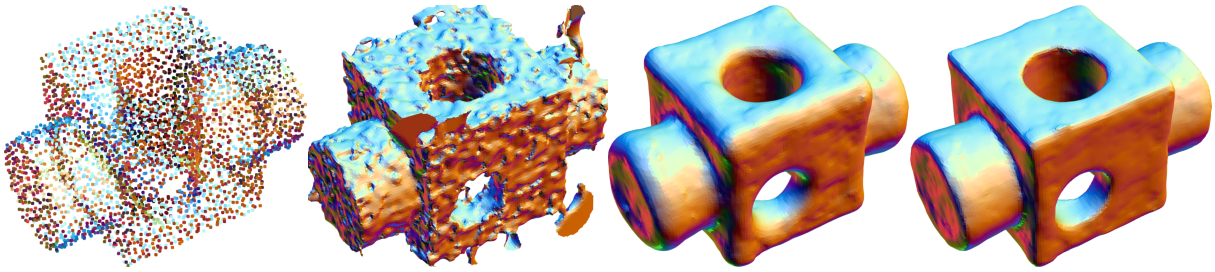
$$\mathrm{Div}(\boldsymbol{v}_i) = \frac{3}{r_i S_i}\sum_j \phi_{i,j}\boldsymbol{v}_j\cdot(\boldsymbol{c}_j-\boldsymbol{c}_i). \tag{9}$$

This is an isotropic operator, and we introduce an anisotropic operator in Section 4.3 by multiplying $\phi_{i,j}$ by an anisotropic factor.

With this new divergence operator, we also obtain the Laplacian operator through $\mathrm{Div}(\mathrm{Grad}(\boldsymbol{v}_i)) = \mathrm{Lap}(\boldsymbol{v}_i)$ in a way similar to the continuous form $\nabla\cdot\nabla(\boldsymbol{v}_i) = \triangle(\boldsymbol{v}_i)$. First, by replacing $\boldsymbol{v}$ with $\nabla\boldsymbol{v}$ in (7), the equation can be represented as

$$\mathrm{Div}(\nabla\boldsymbol{v}_i) = \frac{3}{r_i S_i}\sum_j \phi_{i,j}\nabla\boldsymbol{v}_j\cdot(\boldsymbol{c}_j-\boldsymbol{c}_i). \tag{10}$$

Let us define operator $\mathrm{Grad}(\boldsymbol{v}_i)$ as the first-order Taylor



(a) (b) (c) (d)

Figure 6: *(a) Noisy data points; (b) initial zero-level surface approximated using PU; (c) results of isotropic smoothing by (13) and (16); (d) results of the anisotropic Laplacian smoothing by (17). The sharp features are well preserved.*

expansion at $\boldsymbol{c}_j$, which is $\boldsymbol{v}_i \approx \boldsymbol{v}_j + \mathrm{Grad}(\boldsymbol{v}_j)\cdot(\boldsymbol{c}_i - \boldsymbol{c}_j)$. Then, (10) is expressed as

$$\mathrm{Lap}(\boldsymbol{v}_i) = \frac{3}{r_i S_i}\sum_j \phi_{i,j}(\boldsymbol{v}_j - \boldsymbol{v}_i). \tag{11}$$

As $\mathrm{Lap}(f(\boldsymbol{c}_i)) = \mathrm{Div}(\mathrm{Grad}(f(\boldsymbol{c}_i)))$, the Laplacian for function $f(\boldsymbol{c}_i)$ is derived in the same manner. The Laplacian of $f(\boldsymbol{c}_i)$ on the domain covered by spherical supports is

$$\mathrm{Lap}(f(\boldsymbol{c}_i)) = \frac{3}{r_i S_i}\sum_j \phi_{i,j}(f(\boldsymbol{c}_j) - f(\boldsymbol{c}_i)). \tag{12}$$

The proposed differential operators seem to be applicable to PU consisting of any type of local function. However, these operators are limited to use in cases where local functions are linear, since we simplify gradients by assuming that they are constant in each support.

Approximation with linear local functions is sufficient for surface reconstruction because its output takes the form of a polygonal mesh. Using local linear approximations tends to generate more supports than the quadratic approximations of MPU [OBA*03], but linear approximation makes the algorithm more noise robust, while high-order approximations are more noise-sensitive.

4. Smoothing of Partition of Unity

After local approximations are obtained by the spherical covering, the gradient field $\{\boldsymbol{v}_i\}$ of $f(\boldsymbol{x})$ is smoothed to give smoother surface representation. Each vector $\boldsymbol{v}_i$ is smoothed by Laplacian smoothing with the new discrete Laplacian operators. Then, each local approximation is updated to have the smoothed vector as its gradient.

Here we show an overview of each step of iterative PU smoothing.

1. Initialize the vector field with the continuous gradient of $f(\boldsymbol{x})$: set $\boldsymbol{v}_i \equiv \nabla f(\boldsymbol{c}_i)$ for each sphere center $\boldsymbol{c}_i$
2. Update $\{\boldsymbol{\alpha}_i\}$: smooth $\boldsymbol{v}_i$ by Laplacian smoothing. Set updated vector $\hat{\boldsymbol{v}}_i$ to $\hat{\boldsymbol{\alpha}}_i$
3. Update $\{\beta_i\}$: calculate $\hat{\beta}_i$ so that $\hat{\boldsymbol{v}}_i = \mathrm{Grad}\,(g_i(\boldsymbol{c}_i))$. Update β_i to $\hat{\beta}_i$

The result of this smoothing is shown in Fig. 6. The details of Steps 2 and 3 are described in Section 4.1 and 4.2, respectively.

4.1. Smoothing of Gradient Field

For vector $\boldsymbol{v}_i$, the new smoothed vector $\hat{\boldsymbol{v}}_i$ is obtained by solving $\mathrm{Lap}(\boldsymbol{v}_i) = 0$. It derives the smoothed vector

$$\hat{\boldsymbol{v}}_i = \frac{\sum_j \phi_{i,j}\boldsymbol{v}_j}{\sum_j \phi_{i,j}}. \tag{13}$$

4.2. Update of Local Functions

In this step, we update $\{g(\boldsymbol{x})\}$ with equation (15) reflecting the smoothing of $\{\boldsymbol{v}_i\}$. In the previous step, the update of $\boldsymbol{\alpha}_i$ was performed with $\hat{\boldsymbol{\alpha}}_i = \hat{\boldsymbol{v}}_i$, so in this step, β_i (the constant term of $g_i(\boldsymbol{x})$) is updated. Our aim is equivalent to updating $f(\boldsymbol{x})$ so that it satisfies

$$\mathrm{Grad}\,(f(\boldsymbol{c}_i)) = \hat{\boldsymbol{v}}_i. \tag{14}$$

For simple computation, here we introduce the assumption of $f(\boldsymbol{c}_i) \approx g_i(\boldsymbol{c}_i) = \beta_i$. This implies that the center of a sphere is not included in other spheres. We observed good results for all examples in this paper obtained under this assumption.

By calculating the divergence for the both sides of equation (14), we obtain the weak solution

$$\mathrm{Lap}\,(f(\boldsymbol{c}_i)) = \mathrm{Div}\,(\hat{\boldsymbol{v}}_i)\,. \tag{15}$$

By solving equation (15), β_i is updated to

$$\hat{\beta}_i = \frac{\sum_j \phi_{i,j}\left(\hat{\boldsymbol{\alpha}}_j\cdot(\boldsymbol{c}_i - \boldsymbol{c}_j) + \beta_j\right)}{\sum_j \phi_{i,j}}. \tag{16}$$

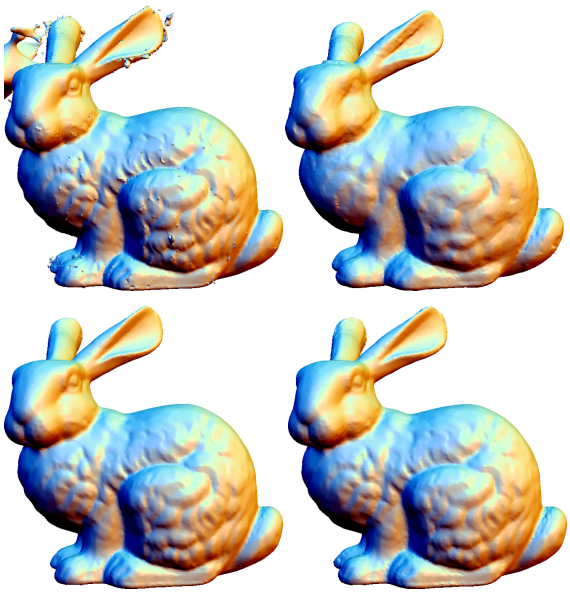


Figure 7: *The effects of fitting terms and convergence. Top-left: The initial zero-level using PU; top-right: results after five iterations without fitting terms; bottom-left: results after five iterations with fitting terms; bottom-right: results after twenty iterations with fitting terms.*

4.3. Anisotropic Laplacian Smoothing

To preserve highly curved regions, the anisotropy factor proposed in [PSM94] is introduced into the smoothing of $\boldsymbol{v}$ (equation (13)) and the update of $f(\boldsymbol{x})$ (equation (16)). We propose the use of the anisotropic factor

$$\psi_{i,j} = \frac{1}{1+\theta_{i,j}^2} \tag{17}$$

where $\theta_{i,j}$ is the angle between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. The weighting function $\phi_{i,j}$ in equations (13) and (16) is replaced with $\phi_{i,j}\psi_{i,j}$.

An example of the effect of this anisotropic factor $\psi_{i,j}$ is shown in Fig. 6. For noisy points (a), results obtained using PU (b), isotropic smoothing (c) and anisotropic smoothing (d) are presented. By introducing the anisotropy factor (17), the sharp edges and corners are preserved.

4.4. Detail Preservation in Smoothing

In surface reconstruction, preserving detail is just as important as smooth representation. To avoid the loss of detail by oversmoothing, we introduce terms that make the approximated surface fit the sampling points. The effects of this modification are shown in Fig. 7. After applying several iterations of PU smoothing without the fitting terms, the details are smoothed out, as shown in the top-right image in Fig. 7. On the other hand, the result with the fitting terms shown in the bottom-left image in Fig. 7 demonstrates that the details are well represented.

By introducing a fitting term to the smoothing of $\boldsymbol{v}$, $\boldsymbol{v}$ is obtained as the minimizer of the following minimization problem:

$$\|\mathrm{Lap}\,(\hat{\boldsymbol{v}}_i)\|^2 + \lambda_n \sum_k \sigma_k w_i(\boldsymbol{p}_k)\,\|\hat{\boldsymbol{v}}_i - \boldsymbol{n}_k\|^2 \to \min. \tag{18}$$

Coefficient λ_n is a sufficiently large number. We set the value at about 10^7. σ_k is the confidence level of $\boldsymbol{p}_k$ (details below). The minimizer of this problem is

$$\hat{\boldsymbol{v}}_i = \left((3/r_i S_i)^2 \textstyle\sum_j \phi_{i,j} \sum_j \phi_{i,j}\boldsymbol{v}_j + \lambda_n \sum_k \sigma_k w_i(\boldsymbol{p}_k)\boldsymbol{n}_k\right) \Big/ \left((3/r_i S_i)^2 (\textstyle\sum_j \phi_{i,j})^2 + \lambda_n \sum_k \sigma_k w_i(\boldsymbol{p}_k)\right). \tag{19}$$

In the same way, the update of $f(\boldsymbol{x})$ (equation (14)) becomes

$$(\mathrm{Lap}\,(f(\boldsymbol{c}_i)) - \mathrm{Div}\,(\hat{\boldsymbol{v}}_i))^2 + \lambda_p \sum_k \sigma_k w_i(\boldsymbol{p}_k) g_i^2(\boldsymbol{p}_k) \to \min. \tag{20}$$

Coefficient λ_p is set at a sufficiently large value, e.g. 10^9. The minimizer of (20) is

$$\hat{\beta}_i = \left((3/r_i S_i)^2 \textstyle\sum_j \phi_{i,j} \sum_j \phi_{i,j}(\hat{\boldsymbol{\alpha}}_j \cdot (\boldsymbol{c}_i - \boldsymbol{c}_j) + \beta_j) + \lambda_p \hat{\boldsymbol{\alpha}}_i \cdot \left(\textstyle\sum_k \sigma_k w_i(\boldsymbol{p}_k)(\boldsymbol{c}_i - \boldsymbol{p}_k)\right)\right) \Big/ \left((3/r_i S_i)^2 (\textstyle\sum_j \phi_{i,j})^2 + \lambda_p \sum_k \sigma_k w_i(\boldsymbol{p}_k)\right). \tag{21}$$

Confidence level σ_k indicates the reliability of a sampling point $\boldsymbol{p}_k$. In cases where confidence values are obtained with a scanner, using these values is one solution. In cases where such values are not available, we assign confidence value σ_k as defined below for each sampling point $\boldsymbol{p}_k$:

$$\sigma_k = \frac{\sum_i w_i(\boldsymbol{p}_k)\tau_i}{\sum_i w_i(\boldsymbol{p}_k)}. \tag{22}$$

τ_i represents the distribution of sampling points inside support s_i. The definition of τ_i is

$$\tau_i = \exp\left(-2\omega_i^2\right) \tag{23}$$

where ω_i is the angle between eigenvector $\boldsymbol{N}_i$ of the minimum eigenvalue of PCA for the sampling points inside s_i and the vector from centroid $\boldsymbol{G}_i$ of the sampling points inside s_i to center $\boldsymbol{c}_i$ of s_i (see Fig. 8).

σ_k becomes smaller when $\boldsymbol{p}_k$ is near an area with a lack of sampling. σ_k is based on the confidence of supports $\{s_i\}$ including $\boldsymbol{p}_k$.

When the sampling point distribution is uniform (as in the

part on the left of Fig. 8), vector $\boldsymbol{N}_i$ and $\boldsymbol{c} - \boldsymbol{G}_i$ have almost the same direction, and the value of τ_i increases. If the sampling density changes inside s_i, their directions are different and τ_i decreases. The distribution of the value of σ_k is indicated in Fig. 9. Note that outliers are detected.

Moreover, the fitting terms help smoothing to converge quickly. It can be seen that the results from five iterations of smoothing (the bottom-left image in Fig. 7) and those from twenty iterations (the bottom-right image in Fig. 7) show no notable difference.

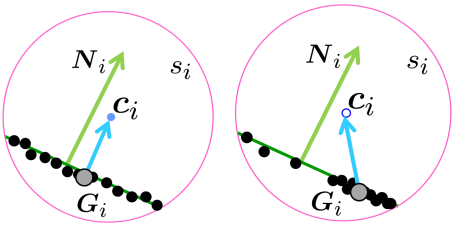


Figure 8: *Examples of 2D spheres with points (black dots) uniformly sampled (left) and non-uniformly sampled (right). The line approximated using PCA for the sampling points is shown in dark green, and its normal $\boldsymbol{N}_i$ is shown with a light green arrow. The centroid of the sampling points, $\boldsymbol{G}_i$, is indicated with a large gray point.*

4.5. Algorithm Summary

Here we summarize the whole algorithm as follows.

1. Approximate the surface using PU
2. Evaluate the confidence of the sampling points
3. Iteratively apply PU smoothing preserving detail
4. Polygonize the zero-level surface

Step 1 has already been explained in Section 2, and Steps 2 and 3 are outlined in Section 4.4. The number of the iterations in Step 3 is five in our experiments for all models. In the last step, we polygonize the zero-level surface using the SurfaceNets [Gib98] with uniform sampling around only the zero-level surface as proposed in [Blo88]. We set use a sampling point with a high confidence value as a seed point for detecting the zero-level surface.

5. Results and Discussion

Parameter setting. All input data in this paper (except Fig. 6, 11, and 15) are noisy raw scanned data. All parameters except ε are fixed unless the settings are described. The only free parameter required by the proposed algorithm is approximation error tolerance ε, which depends on the noise level of the scanning device. A comparison of results with different values of ε is shown in Fig. 10 (the details of ε are described in Section 2.2).

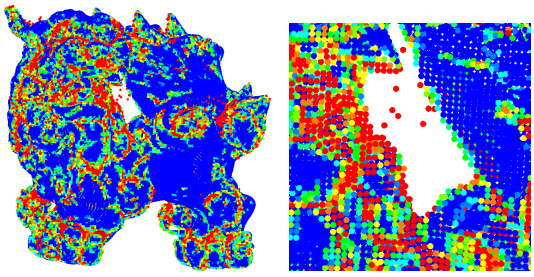


Figure 9: *Confidence levels are indicated with coloring; blue is high (reliable) and red is low (unreliable). Low confidence values are assigned to outliers.*

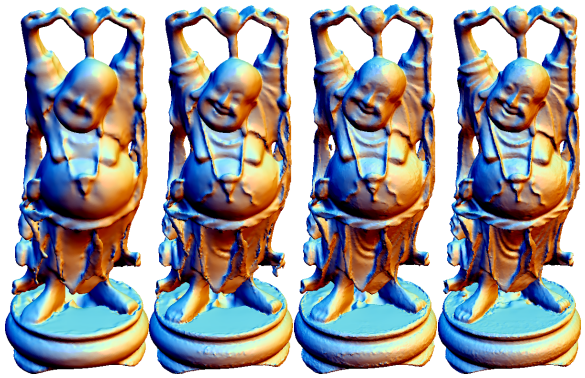


Figure 10: *The reconstructed mesh with different values of ε. From left to right, $\varepsilon = 1.0\times10^{-2}, 5.0\times10^{-3}, 2.5\times10^{-3}$, and 1.25×10^{-3}.*

Noise robustness. The robustness for noise is demonstrated in Fig. 11. The sampling points are obtained from a noisy mesh. The noise level on the right model is twice as large as that on the left model. The reconstructed meshes are topologically correct even for data with large amount of noise.

In Fig. 12, the robustness for normal noise is demonstrated. For data with normals randomly rotated by 30°, our approach succeeded in reconstruction (b). For a rotation angle of 60°, the reconstructed model is affected in terms of smoothness (d). By setting λ_n to zero, even for such cases, a smooth surface is obtained (e).

Performance. Timing results are reported in the Table 1. In order to perform PU smoothing efficiently, we construct and store the graph as an adjacency list data structure.

However, following our observation, the average of degrees of a sphere intersection graph is about 100, which makes processing large data sets difficult. According to our experiments, reducing the degree by shrinking the radii of

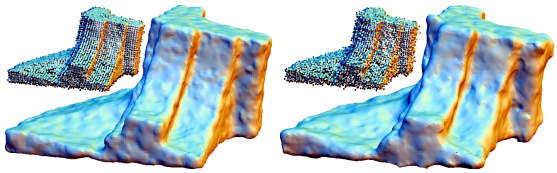
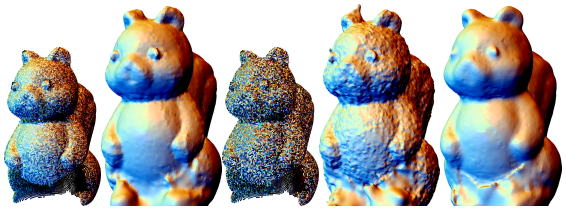


Figure 11: *Noise robustness. The sampling points are obtained from a mesh added Gaussian noise. The value of the standard deviation of the noise on the left/right model is a quarter/half of the averaged edge-lengths.*



(a) (b) (c) (d) (e)

Figure 12: *Normal noise robustness. (a) Data with normals rotated randomly by 30°; (b) reconstructed model for (a); (c) data with normals rotated randomly by 60°; (d) results for (c); (e) results for (c) with $\lambda_n = 0$.*

supports to 0.7 times the original radii reduce the computational time and space. The average of degree after this shrinking is about 40. The results obtained with this shrinking are no different from those without shrinking. This shrinking is only for constructing a sphere intersection graph, so we use the original radii in all the other process including the calculation of $\phi_{i,j}$.

Model	# pts.	ε ($\times 10^{-3}$)	# supps.	Time [min:sec] init.	smooth.	poly.	Peak RAM	# tris.
Bunny	362K	2.5	176K	0:18	0:19	0:14	74M	315K
Arma.	2.3M	1.0	7.35M	5:42	36:21	3:49	3.24G	3.30M
Buddha	3.2M	2.5	540K	2:05	2:26	0:58	234M	1.07M
Buddha	3.2M	1.25	2.94M	3:41	13:35	1:08	1.23G	1.08M

Table 1: *Computational time and performance. From left to right: the number of sampling points, value of ε, the number of supports, computation time for initialization of PU, for smoothing, and for polygonization, the maximum memory space our algorithm used, and the number of triangles of the result mesh. Arma. is an abbreviation of Armadillo.*

Comparison. We compared our results with MPU [OBA*03] and Poisson surface reconstruction [KBH06] for a Shiisa model (see Fig. 1). In the MPU results, peak stretches and extra components appear around the tail. Poisson surface reconstruction gives a mesh in which detail is lost by the smoothing effect. On the other hand, the proposed method generates a smooth mesh with the detail well preserved (see the close-ups in the lower columns of Fig. 1).

Another comparison for a large data set with Poisson surface reconstruction is shown in Fig. 13. The reason that Poisson surface reconstruction is selected as a comparative algorithm is that it has excellent reconstruction capabilities, and the software required is freely available. The parameters for Poisson surface reconstruction are 11 for the level of the octree, and for ours, $\varepsilon = 1.0 \times 10^{-3}$. In reconstruction using our method, detail is preserved well, e.g. the bumps in the hand. This is thought to be because the proposed algorithm uses first-order approximation, while Poisson surface reconstruction uses zeroth-order approximation.

Another example is shown in Fig. 14. The surface is gradually sparsely sampled. The results of Poisson surface reconstruction shrink near the tail, but the results from proposed method can express a rough shape of the tail.

One of the drawbacks of our method is that reconstructed surfaces tend to extend in the tangential directions. See the tail of the dragon in Fig. 14. A more clear comparison of this property is shown in Fig. 15. This is caused by diffusion of normals in our method. In contrast, Poisson surface reconstruction solves Laplacian equation on the region where sampling is missing. Thus the missing parts are filled with minimal surfaces.

Another drawback is that the proposed method is three–five times more time-consuming than that of the Poisson surface reconstruction. However, our PU smoothing based on iterative averaging can be easily parallelized by using multicore CPU.

6. Conclusion

We derived new differential operators for a spherically covered domain. As a related application, detail-preserving noise robust surface reconstruction algorithm is proposed based on a diffusion technique.

The local approximations in this work are linear functions. Changing the primitive functions to be quadratic and cubic functions is a future task to be performed, and we also plan to represent sharp edges. Anisotropic smoothing preserves edges, but more precise representation is needed in some applications. Other possible applications include volume processing and the smoothing of more attributed vectors such as 4D vectors and texture data.

Acknowledgement

The authors recognize the fruitful discussions involving Hiroshi Kawaharada and Ralph Martin, and express their thanks to Michael Kazhdan for making his algorithm publicly available. Thanks also go the Stanford Computer

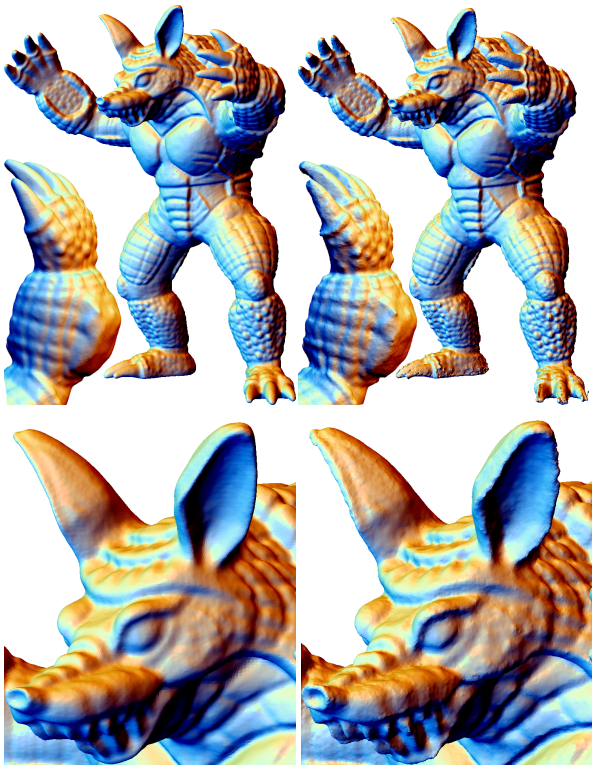


Figure 13: *Meshes reconstructed using the Poisson surface reconstruction (left) and our methods (right).*

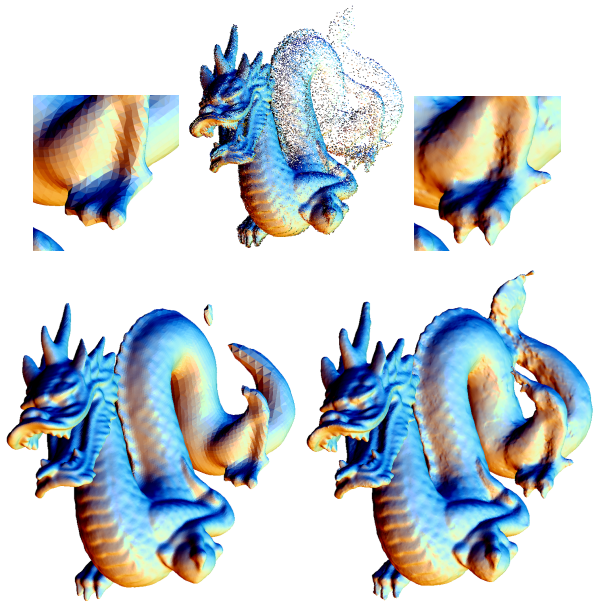


Figure 14: *Comparison of results for gradually downsampled points. Top: close-ups of the results from Poisson surface reconstruction (left), proposed method (right), and sampling points (middle). Bottom: results from Poisson surface reconstruction (left) and proposed method (right).*

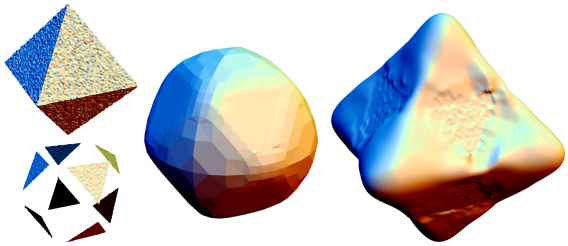


Figure 15: *Comparison of the properties of reconstructed surfaces. Left: the original noisy mesh (top) and partially sampled points with normals from the mesh (bottom). Middle: results from Poisson surface reconstruction. Right: results from our method.*

Graphics Laboratory for providing scanning data. The authors are also grateful for a range of suggestions from anonymous reviewers. This research is partly funded by Grant-in-Aid for Scientific Research (B), No. 19360070, Japan Society for the Promotion of Science (JSPS).

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01* (2001), pp. 21–28.

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The power crust. In *Proceedings of 6th ACM Symposium on Solid Modeling* (2001), pp. 249–260.

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), pp. 39–48.

[AK04] AMENTA N., KIL Y.: Defining point-set surfaces. *ACM Transactions on Graphics 23*, 3 (2004), 264–270. Proceedings of ACM SIGGRAPH 2004.

[Blo88] BLOOMENTHAL J.: Polygonization of implicit surfaces. *Comput. Aided Geom. Des. 5*, 4 (1988), 341–355.

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: 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* (2001), pp. 67–76.

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of ACM SIGGRAPH 1996* (1996), pp. 303–312.

[CT03] CHOUDHURY P., TUMBLIN J.: The trilateral filter for high contrast images and meshes. In *EGRW '03: Proceedings of*

the 14th Eurographics workshop on Rendering (2003), pp. 186–196.

[DG06] DEY T. K., GOSWAMI S.: Provable surface reconstruction from noisy samples. *Comput. Geom. Theory Appl. 35*, 1 (2006), 124–141.

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), pp. 317–324.

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), pp. 23.1–23.9.

[Gib98] GIBSON S. F. F.: Using distance maps for accurate surface representation in sampled volumes. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization* (1998), pp. 23–30.

[HK06] HORNUNG A., KOBBELT L.: Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), pp. 41–50.

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 339–346.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), pp. 61–70.

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), pp. 105–114.

[KSO04] KOLLURI R., SHEWCHUK J. R., O'BRIEN J. F.: Spectral surface reconstruction from noisy point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), pp. 11–21.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), pp. 163–169.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Transaction of Graphics 22*, 3 (2003), 463–470.

[OBS02] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Mesh smoothing by adaptive and anisotropic gaussian filter. In *Proceedings of Vision, Modeling, and Visualization VMV 2002* (2002), pp. 203–210.

[ÖGG09] ÖZTIRELI C., GUENNEBAUD G., GROSS M.: Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum 28*, 2 (2009), 493–501. Proceedings of EUROGRAPHICS 2009.

[PSM94] PERONA P., SHIOTA T., MALIK J.: Anisotropic diffusion. In *Geometry-Driven Diffusion in Computer Vision* (1994), pp. 73–92.

[Set99] SETHIAN J. A.: *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999. Second Edition.

[SLS*07] SHARF A., LEWINER T., SHKLARSKI G., TOLEDO S., COHEN-OR D.: Interactive topology-aware surface reconstruction. *ACM Transaction of Graphics 26*, 3 (2007), 43.1–43.9.

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Computer Graphics (Proceedings of SIGGRAPH 95)* (1995), pp. 351–358.

[Tau01] TAUBIN G.: *Linear Anisotropic Mesh Filters*. Tech. Rep. RC-22213 (W0110-051), IBM Research Technical Report, 2001.

[TLHD03] TONG Y., LOMBEYDA S., HIRANI A. N., DESBRUN M.: Discrete multiscale vector field decomposition. *ACM Trans. Graph. 22*, 3 (2003), 445–452.

[TWBO02] TASDIZEN T., WHITAKER R., BURCHARD P., OSHER S.: *Geometric Surface Processing via Normal Maps*. Tech. Rep. UUCS-02-003, University of Utah School of Computing, 2002.