Outlier/Noise-Robust Partition of Unity Implicit Surface Reconstruction

Yukie Nagai*
RIKEN, Japan

Yutaka Ohtake†
The University of Tokyo, Japan

Hiromasa Suzuki‡
The University of Tokyo, Japan

Hideo Yokota§
RIKEN, Japan

Abstract

In this paper, we propose an algorithm for outlier/noise-robust surface reconstruction based on a partition of unity (PU) approach. PU based surface reconstruction is a local method that covers an area including sampling points with spherical supports of local approximations, and then generates an approximation function whose zero-level sets approximate the surface. This algorithm has many advantages including representation of fine details, and fast and memory efficient computation. Many of these advantages are realized with the locality of PU however, it is also the reason of outlier/noise-instabilities. Unfortunately, scanned data generally contains much amount of noise, and hence improving the robustness of PU based algorithm is required. We achieved an outlier/noise-robust algorithm with integrating Graph-cut and diffusion of local approximations. Since the characteristics of outliers and noise are fundamentally different, overcoming these two with different approaches is reasonable. In our algorithm, first a spherical cover of an area containing input points is generated following the PU manner. And then Graph-cut is performed in order to determine spherical supports which are considered wrongly approximating affected by outliers. Finally, the PU approximation function is updated so that its gradient field smoothed. This smoothing is based on a diffusion of the local approximations. In this paper we show the effects of this integration approach for several scanned data sets.

Key words: surface reconstruction, partition of unity, Graph-cut, smoothing

1. Introduction

Surface reconstruction is a process generating a surface from sampling points. Sometimes, if necessary, generated surfaces are approximated by surface meshes. This subject is a very important in many industrial areas where shapes of real objects are handled, including mechanical engineering as CAD, CAE and computer graphics. Unfortunately surface reconstruction is an essentially hard issue in the point that original shape cannot be completely known from sampling points. Moreover, some kinds of problems such as noise, outliers, and lacks of sampling make surface reconstruction difficult.

In order to overcome these difficulties, many algorithms are proposed. Previous algorithms can be roughly classified into two groups according to their approaches: explicit methods and implicit methods.

Explicit methods involve Delaunay-based algorithms [2, 5, 8]. These algorithms generate meshes whose vertices are parts of sampling points. This approach is intuitive and can give quality guarantee for the generated surfaces. But it is also well known that the algorithms in this category are unstable for noise.

In an implicit approach, a surface is offered just as an implicit form, that is, a surface is defined as the boundary between the internal and external points. Global implicit approaches [7, 11, 13] have great advantages of handling low quality data but it is achieved at the cost of high-accuracy reconstruction. On the other hand, local implicit methods [1, 3, 17] have advantages such as highly accurate representation and low computational cost. But because of their localities, there are some difficulties in handling very

*nagai@riken.jp
†yu-ohtake@den.rcast.u-tokyo.ac.jp
‡suzuki@den.rcast.u-tokyo.ac.jp
§hyokota@riken.jp

low-quality data with large amounts of noise, outliers, lacks of sampling, and large differences in sampling density. The resulting shapes may be significantly deformed and have some extra components.

To achieve a robust reconstruction with smooth and precise representation, first we combine a partition of unity approach (a local method) and the Graph-cut (a global method), and then smooth the local approximations using a diffusion technique.

We explain the algorithm in Sec. 2 and then show several experimental results in Sec. 3, and finally conclude this paper and mention our future work in Sec. 4.

2. Algorithm

First of all, we show the overview of this integrated algorithm below.

Algorithm

1. Generate a spherical cover through partition of unity
2. Operate the Graph-cut and detect local approximations affected by outliers
3. Diffuse the local approximation functions
4. Polygonize the zero-level set

The first step will be explained in Sec. 2.1. The Graph-cut and the diffusion will be described in Sec. 2.2 and Sec. 2.3 respectively. The last step, polygonization of an isosurface, is well studied [12, 14]. We adopt the SurfaceNets [10] with uniform sampling around only the zero-level surface as proposed in [4]. We set a sampling point with a high confidence value as a seed point for detecting the zero-level surface.

2.1. Spherical Cover Generation

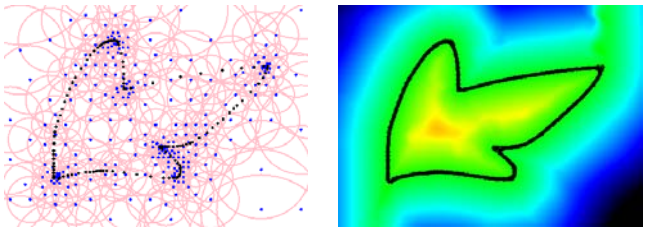


Figure 1. Examples of a spherical cover and a scalar field in 2D. Left: the supports of local approximation functions $\{g_i(\boldsymbol{x})\}$ (pink circles) and those centers (blue dots). Black dots means the sampling points. Right: the scalar field of the approximation functions $f(\boldsymbol{x})$ and the reconstructed surface (black line). Values of $f(\boldsymbol{x})$ are indicated in red for high and blue for low.

Given a set of points $\{\boldsymbol{p}_j\}$ sampled from a surface of an object associated with oriented normals $\{\boldsymbol{n}_j\}$, we approximate the surface of the object with the zero-level sets of a signed distance function $f(\boldsymbol{x})$. An example of a scalar field of $f(\boldsymbol{x})$ is shown in the image on the right of Fig. 1. In our implementation, inside of the object is assigned positive value. Approximation function $f(\boldsymbol{x})$ can be realized with a weighted average of local approximations $\{g_i(\boldsymbol{x})\}$:

$$f(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})}{\sum_i w_i(\boldsymbol{x})}. \tag{1}$$

A local approximation function $g_i(\boldsymbol{x})$ has a support s_i that is a sphere defined with the center $\boldsymbol{c}_i$ and the radius r_i. We adopt a linear function as a local function; $g_i(\boldsymbol{x}) = \boldsymbol{\alpha}_i \cdot \boldsymbol{x} + \beta_i$. Local approximation $g_i(\boldsymbol{x})$ can be determined with the least square fitting for $\{\boldsymbol{p}_k\}$, the sampling points inside s_i. The weighting function $w_i(\boldsymbol{x})$ is the second degree B-spline $b_2\left(3\|\boldsymbol{x} - \boldsymbol{c}_i\|/2r_i\right)$. A domain including all the sampling points is covered with spherical supports as shown in the image on the left of Fig. 1.

Our spherical covering algorithm completely follows [16]. We assume that the area which should be covered is rescaled so that the length of the longest edge of the bounding box of $\{\boldsymbol{p}_j\}$ equals to one. The

covering starts with dividing the domain with an adaptive octree measuring local approximation errors. Each octant is assigned a spherical support s_i of $g_i(\boldsymbol{x})$. The center $\boldsymbol{c}_i$ of s_i is the center of the octant and the radius r_i is 0.75 times the length of the octant's diagonal. And then, $g_i(\boldsymbol{x})$ is determined with the least square fitting for $\{\boldsymbol{p}_k\}$. We measure the approximation error of $\{g_i(\boldsymbol{p}_k)\}$ with the maximum value of $|g_i(\boldsymbol{p}_k)|$. Each octant is divided recursively until the approximation error does not exceed a user-specified tolerance ε. After this division terminated, the distribution of supports and those size reflects the density of the input points.

And Then, in order to reduce the number of supports, the radii of randomly selected supports are expanded without exceeding ε, and then supports completely included in the expanded supports are eliminated. For more details, authors recommend to refer [16].

2.2. Graph-cut

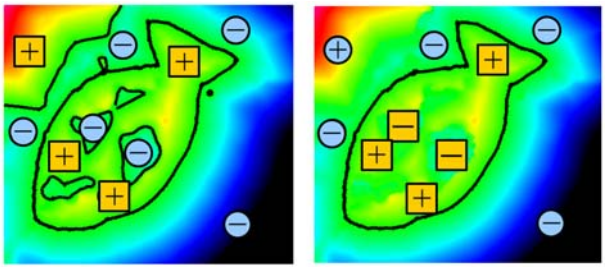


Figure 2. Inside/outside classification only through partition of unity (left) and through combination of partition of unity and the Graph-cut (right).

Since PU is a local fitting approach, it achieves high-accuracy representation of objects but sensitive to low-quality data involving outliers and lacks of sampling. As shown in Fig. 2, overcoming these difficulties with introducing a global approach is a good solution. In order to determine supports affected by outliers and lacks of sampling, we use the Graph-cut approach as proposed in [15]. We consider a support whose sign of $f(\boldsymbol{x})$ is not coincident with the results of Graph-cut is affected by outliers and lacks of sampling.

In this strategy, basically we follow [15] however, the graph is slightly different from the one in [15]. In [15], the graph is constructed based on the weighted Delaunay tetrahedrization obtained from the sphere intersection graph (SIG) [9]. In this mesh, several supports may do not appear as vertices. In our algorithm, it is needed for the all supports to be classified since the classification results are used in the following smoothing step. In [15], the tetrahedral mesh also serves as a grid for the extraction of a surface mesh. In this algorithm, it can be replaced with another grid structure.

The weighting functions for graph edges are the same as the ones in [15]. We assign the following cost to a terminal edge that one end is a terminal node (source or sink) and the other is a vertex v of SIG. Note that the terminal nodes are symbolic nodes and are not realized with vertices of SIG.

$$\begin{cases} w(v,\text{source}) = 0, & w(v,\text{sink}) = k\frac{|f(\boldsymbol{v})|}{l_v} & (f(\boldsymbol{v}) < 0) \\ w(v,\text{sink}) = 0, & w(v,\text{source}) = k\frac{|f(\boldsymbol{v})|}{l_v} & (f(\boldsymbol{v}) \geq 0) \end{cases} \tag{2}$$

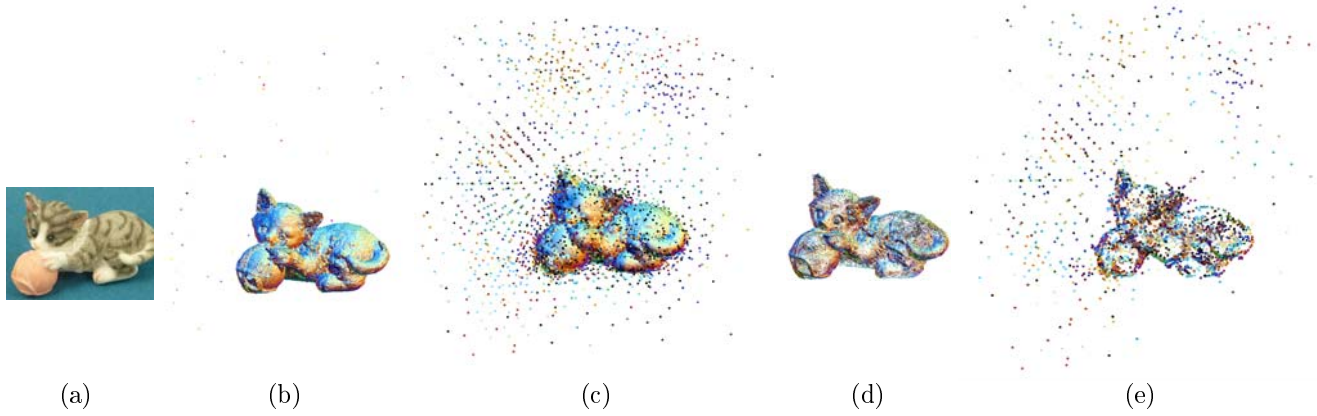
where l_v is the average length of edges incident to v. A constant k means a ratio of cost for terminal edges to cost for tetrahedral mesh edges (we set k to about $27 \sim 30$). The cost for a tetrahedral mesh edge with end points v_i and v_j is

$$w(v_i, v_j) = \frac{|f(\boldsymbol{v}_i) + f(\boldsymbol{v}_j)|}{l_{v_i,v_j}} \tag{3}$$

where l_{v_i,v_j} is the length of the edge. Since $f(\boldsymbol{x})$ means an approximation of the signed distance from the surface, this cost increases for edges far from the surface. On the other hand, it decreases for edges crossing the surface since the signs of $f(\boldsymbol{x})$ at end points are different. As a result, crossing edges tend to belong to a minimal cut-set. For the implementation of Graph-cut, we utilized codes distributed by Boykov and Kolmogorov based on [6].

With performing the Graph-cut can detect the *inconsistent* supports $\{s_i\}$ that the signs of $f(\boldsymbol{c}_i)$ and the results of inside/outside classification through the Graph-cut are different.

The images in Fig. 3 show the effects of the Graph-cut with the centers of supports. (b) shows input points which consists of scanned points of a figure shown in (a) and 50 outliers. The image (c) shows the centers of the all supports. With applying the Graph-cut, the centers of supports inside-classified are shown in (d). The centers of the inconsistent supports are shown in (e).



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

Figure 3. Effects of the Graph-cut. (a) original object. (b) input points. (c) the centers of the all supports. (d) the centers of supports classified into inside supports. (e) the centers of inconsistent supports.

2.3. Diffusion

After the Graph-cut, in order to reduce the effects of noise, we smooth the gradient field with a diffusion technique. This scheme basically follows the detail preserving smoothing algorithm in [16]. Here we show an overview of the smoothing and brief explanations. For more details, see the explanations in [16].

Algorithm: smoothing partition of unity implicits

Iterate the following steps

1. Initialize the vector field $\boldsymbol{v}_i$ with the gradient of $f(\boldsymbol{x})$ on each sphere center $\boldsymbol{c}_i$
2. Update a vector $\boldsymbol{v}_i$ into $\hat{\boldsymbol{v}}_i$ with Laplacian smoothing
3. Update the local approximations to satisfy $\hat{\boldsymbol{v}}_i = \mathrm{Grad}\,(f(\boldsymbol{c}_i))$

In order to avoid affections of the inconsistent supports, local approximations of the inconsistent supports are changed into zero functions. In our implementation, the number of the iteration in the above algorithm is set to five. In Step 2, the gradients of $f(\boldsymbol{x})$ is smoothed through Laplacian smoothing with the discrete Laplacian operator on a spherical cover. And then, each local approximation function is updated with solving the Poisson equation in Step 3, so that the smoothed vectors become the gradients of local approximations. The discrete differential operators on a spherical cover are derived in [16].

Initializing the gradient field. First the gradient field is initialized with assigning $\boldsymbol{v}_i = \nabla f(\boldsymbol{c}_i)$ for each support center $\boldsymbol{c}_i$. The left image in Fig. 4 shows the initial state of the scalar field. The directions of gradient vectors and the values of the isosurfaces of the local approximations are different. As a result, the isosurface of $f(\boldsymbol{x})$ (a solid green line) is bumpy.

Smoothing the gradient field. In Step 2, vectors $\{\boldsymbol{v}_i\}$ are smoothed through the Laplacian smoothing, that is, the smoothed vector $\hat{\boldsymbol{v}}_i$ satisfies

$$\mathrm{Lap}(\hat{\boldsymbol{v}}_i) = 0. \tag{4}$$

In the following of this paper, $\mathrm{Grad}(*), \mathrm{Div}(*), \mathrm{Lap}(*)$ mean the discrete gradient, the discrete divergence, and the discrete Laplacian operator on a spherical cover respectively.

Update the local approximation functions. In the final step, local approximation functions are updated reflecting the smoothing of the gradient field. It is performed solving a Poisson equation. First the coefficient vectors $\{\boldsymbol{\alpha}_i\}$, then constant terms $\{\beta_i\}$ are updated into $\{\hat{\boldsymbol{\alpha}}_i\}$ and $\{\hat{\beta}_i\}$ respectively.

The update of a coefficient vector $\boldsymbol{\alpha}_i$ is realized with a substitution $\hat{\boldsymbol{\alpha}}_i = \hat{\boldsymbol{v}}_i$. The image in the center of Fig. 4 shows the state after the coefficient vectors are updated. The directions of the gradient vectors become consistent and the isosurface of $f(\boldsymbol{x})$ becomes smooth.

Updating of a constant term is equivalent to updating $f(\boldsymbol{x})$ so that it satisfies $\mathrm{Grad}\left(f(\boldsymbol{c}_i)\right) = \hat{\boldsymbol{v}}_i$. The left hand side is only composed of a divergence term, so it implies that the smoothed vector field must be represented with no curl-term. This strong constraint makes solving this equation difficult. So we solve the weak form obtained with applying the discrete divergence operator on the both hand sides. The weak form is a Poisson equation:

$$\mathrm{Lap}\left(f(\boldsymbol{c}_i)\right) = \mathrm{Div}\left(\hat{\boldsymbol{v}}_i\right). \tag{5}$$

The effects of this updating is shown in the right image in Fig. 4. The values of the isosurfaces become consistent and the isosurface of $f(\boldsymbol{x})$ becomes much smoother than the surface updated only the gradients.

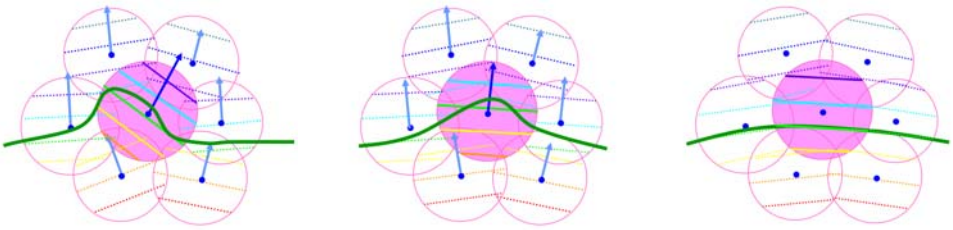


Figure 4. The effects of the smoothing of the scalar field. Bold green lines mean the zero-level sets of $f(\boldsymbol{x})$. Left: initial state. Center: after the updating of the coefficient vectors. Right: after updating of the constant terms.

3. Results

In this chapter we show some experimental results for scanned data. These show the abilities of the integrated algorithm to handle low quality data.

The first example is for points shown in the image (b) of Fig. 3. This data consists of 1161580 points including 50 outliers. The reconstructed surfaces obtained with only the diffusion technique is shown in the left image of Fig. 5. It has large extra surfaces because of the outliers. With the help of the Graph-cut (right), such surfaces do not appear. Moreover, the area around the left forefoot and the ball where clouds of outliers exist are correctly reconstructed.

Fig. 6 shows the results for gradually downsampled data with 294259 points including 100 outliers. Even the head, sparsely sampled area, is well reconstructed.

The images in Fig. 7 are the results for points with large lacks of sampling and 20 outliers. The total number of the points is 28751. In the result only with diffusion technique, extra surfaces exist near the back. With the aid of the Graph-cut, the smooth shape is reconstructed.

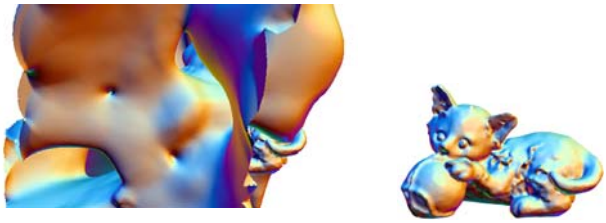


Figure 5. Results for points shown in the image (b) of Fig. 3. The surfaces obtained only with the diffusion technique (left) and with the integration of the Graph-cut and the diffusion technique (right).

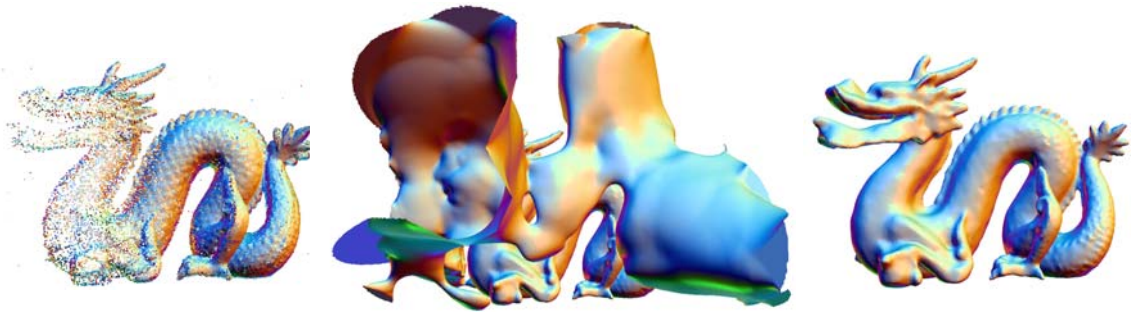


Figure 6. Results for gradually sparsely sampled and outlier-added points. Left: input points. Center: surfaces obtained with the diffusion technique. Right: surfaces obtained with the Graph-cut and the diffusion technique.

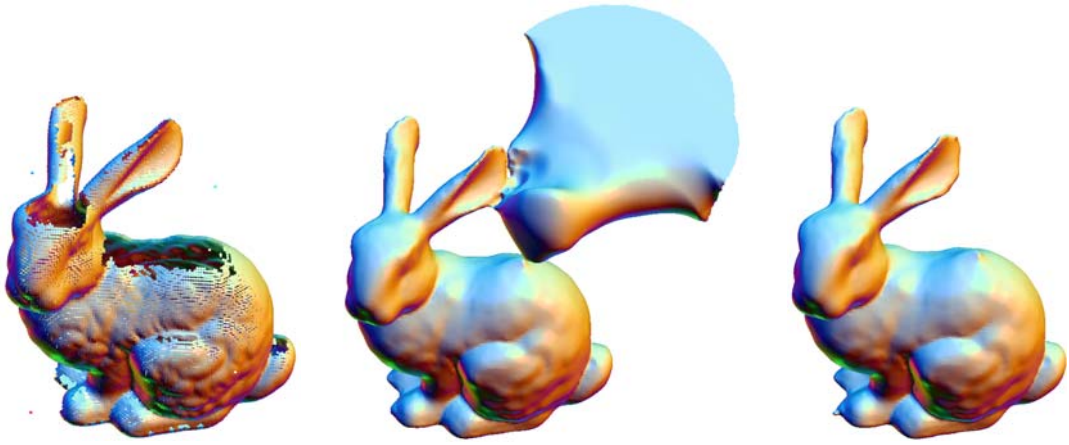


Figure 7. Result for points with lacks and outliers. Left: input points. Center: surfaces obtained with the diffusion technique. Right: surfaces obtained with the Graph-cut and the diffusion technique.

4. Conclusion and Future Work

In this paper a PU-based surface reconstruction algorithm using the Graph-cut and a diffusion technique is proposed. This algorithm has advantages of outlier/noise-robustness and detail preservation. Outlier-robustness is achieved with the Graph-cut and noise-robustness is mainly achieved with the aid of a diffusion of local approximation functions. Robustness for lacks of sampling is also achieved.

As shown in the right image of Fig. 7, lost of details may occur. There is no undesired surfaces but the details are lost. On the other hand, the results in Fig. 8 give surfaces in which the details are better represented but undesired surfaces remain. This means the trade-off between precise representation and no-extra-component reconstruction. Adaptive settings of the value of ε and fitting parameters of the diffusion (refer [16]) may help to reconstruct details with no extra surfaces.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, 2001.

[2] N. Amenta, S. Choi, and R. Kolluri. The power crust. In *Proceedings of the 6th ACM Symposium on Solid Modeling*, pages 249–260, 2001.

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

[4] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.

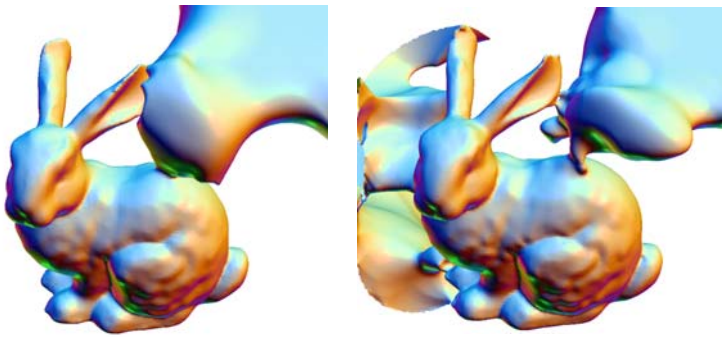


Figure 8. Results of the integrated approach with small ε for points in Fig. 7. Only changing the values of fitting parameters cannot achieve results with no extra surfaces.

[5] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, October 1984.

[6] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[7] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Computer Graphics (Proceedings of SIGGRAPH 2001)*, pages 67–76, 2001.

[8] T. K. Dey and S. Goswami. Tight cocone: A water-tight surface reconstructor. In *Proc. 8th ACM Sympos. Solid Modeling Applications*, pages 127–134, 2003.

[9] H. Edelsbrunner. The union of balls and its dual shape. In *Proceedings of the 9th annual symposium on Computational geometry*, pages 218–231, San Diego, California, United States, 1993.

[10] Sarah F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 23–30, 1998.

[11] A. Hornung and L. Kobbelt. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *SGP '06: Proceedings of the 4th Eurographics symposium on Geometry processing*, pages 41–50, 2006.

[12] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, July 2002. Proceedings of ACM SIGGRAPH 2002.

[13] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP '06: Proceedings of the 4th Eurographics symposium on Geometry processing*, pages 61–70, 2006.

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

[15] Y. Nagai, Y. Ohtake, and H. Suzuki. Noise robust surface reconstruction by combining pu and graph-cut. In *Eurographics*, pages 73–76, 2009.

[16] Y. Nagai, Y. Ohtake, and H. Suzuki. Smoothing of partition of unity implicit surfaces for noise robust surface reconstruction. *Computer Graphics Forum*, 28(5):1339–1348, 2009. Proceedings of Symposium on Geometry Processing 2009.

[17] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transaction of Graphics*, 22(3):463–470, 2003.