Noise Robust Surface Reconstruction by Combining PU and Graph-cut

Yukie Nagai, Yutaka Ohtake and Hiromasa Suzuki

The University of Tokyo, Japan

Abstract

We present a novel method of reconstructing surfaces from 3D scattered points by combining Partition of Unity (PU) and a Graph-cut approach. PU is a local approximation technique, meaning that the surfaces obtained have high accuracy but are sensitive to noise. Graph-cut, on the other hand, is a global algorithm that is robust to noise but produces low-accuracy results because it is a discrete binary operation. Our algorithm combines these two methods to achieve robust, high accuracy surface reconstruction. First, a PU implicit function is constructed by covering a space containing a point cloud with spherical supports of linear polynomials. Graph-cut is then performed to separate the covered domain into inside and outside areas of the object to be reconstructed. Finally, we extract the zero-level of PU using the marching tetrahedra approach.

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

1. Introduction

Surface reconstruction from sets of points obtained by scanning devices is a very important subject in many areas including computer graphics, CAD and CAE. Many algorithms have been proposed to achieve such reconstruction, and an implicit approach is a good solution. One of the major advantages of such an approach is the ability to handle low-quality data: data with noise, areas with few or no sampling points and registration gaps. Algorithms in this category can be classified as local or global. One of the former is [OBA*03, GG07]; locally supported implicit functions can precisely represent details and features on the surfaces of objects. As an example of the latter, we refer to [KBH06,HK06,SLS*07].The algorithms in this class are essentially noise-robust, but are not good at dealing with precise features.

In this paper, we propose a new algorithm with the ability to represent details and robustness even for low-quality data. In Fig. 1, we show the results for the Stanford dragon raw data that is very noisy and has areas of undersampling. This is achieved by combining Partition of Unity (a local implicit approach) and Graph-cut (a global method).

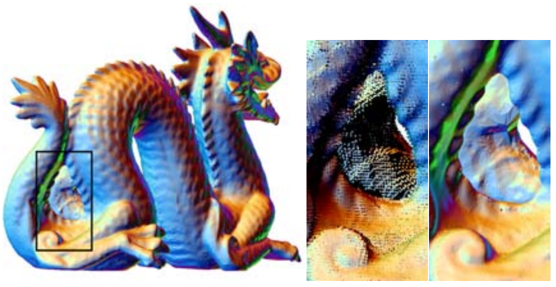
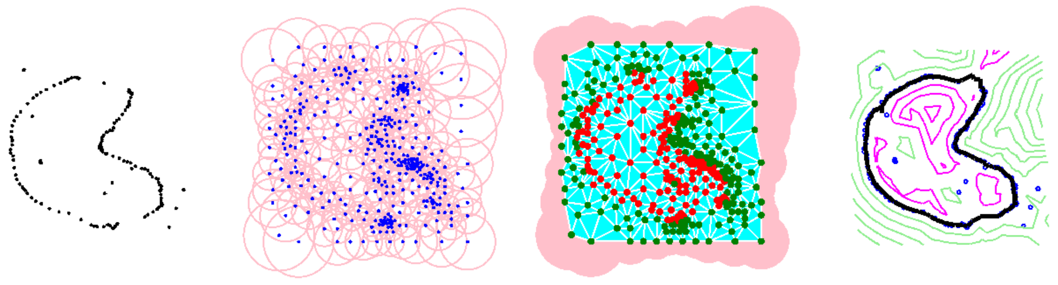


Figure 1: *A triangular mesh of the Stanford dragon reconstructed from raw data (left and right). A lack of sampling is seen on the left hind leg (middle). The reconstructed mesh for this area is shown on the right.*

2. Algorithm

Here we overview the whole algorithm, with 2D-version examples shown in Fig. 2. As input, we take a set of points equipped with oriented normals sampled from the surface of an object and possibly including noise, outliers and areas with a lack of sampling (Fig. 2(a)). These input points are referred to below as *sampling points*.



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

Figure 2: *(a) Sampling points. (b) Generated supports of approximation function $f(\boldsymbol{x})$ (pink circles) and their centers (blue points). (c) The results of Graph-cut; the red points are judged as being inside the object, while the green ones are deemed to be outside. The edges of the graph are shown with white lines. (d) The finally obtained surface of the object (black line). The contours of $f(\boldsymbol{x})$ are drawn with pink and green lines according to their sign of $f(\boldsymbol{x})$. The sampling points are indicated in blue.*

1. Generate a cover for the bonding box with support spheres of local approximation functions through Partition of Unity (Fig. 2(b)).
2. Create a tetrahedral mesh whose vertices are a subset of the support centers. Classify the mesh vertices into those representing the object (red points) and others (green points) by Graph-cut (Fig. 2(c)).
3. Extract a triangular mesh representing the object surface using the marching tetrahedra approach (Fig. 2(d)).

2.1. Local Approximation by Sphere Covering

In this step, a scalar field whose zero-level set represents the surface of an object is generated inside the bounding box of the object. Such a scalar field is constructed by Partition of Unity (PU) similar to the algorithm proposed in [OBA*03], with locally and spherically supported functions in numbers as few as possible. The set of all supports of these functions becomes a cover of the bounding box.

The input is a set of points equipped with oriented normals: $\{(\boldsymbol{p}_i, \boldsymbol{n}_i)\}$. Below the coordinates of a point x are indicated with a bold letter $\boldsymbol{x}$. Using an adaptive octree as the centers of supports is a good way of ensuring the accuracy of reconstruction. We generate an octree whose maximum depth is user-specified through division of the bounding box so that all sampling points are in the maximum-level cell, as in [KBH06]. The difference of the division level with a neighboring cell is one at most. The center candidates of supports are selected from among the cell centers of the octree.

The approximation function is described as a weighted average of local approximations $\{g_i(\boldsymbol{x})\}$ and weight functions $\{\varphi_i(\boldsymbol{x})\}$:

$$f(\boldsymbol{x}) = \frac{\sum_i \varphi_i(\boldsymbol{x}) g_i(\boldsymbol{x})}{\sum_i \varphi_i(\boldsymbol{x})}.$$

$\varphi_i(\boldsymbol{x})$ is a radially supported quadratic B-spline, $b_2(3\|\boldsymbol{x} - \boldsymbol{c}_i\|/2r_i)$, whose center c_i and support size r_i are equal to the center and radius of the support of $g_i(\boldsymbol{x})$. As the local approximation function $g_i(\boldsymbol{x})$, we adopt a linear function because the density of supports directly becomes the density of the resulting surface mesh. $g_i(\boldsymbol{x})$ is determined by weighted least squares fitting to the sampling points inside the support.

$$g_i(\boldsymbol{x}) = \boldsymbol{m}_i \cdot (\boldsymbol{x} - \boldsymbol{d}_i),$$

$$\boldsymbol{m}_i = \frac{\sum_j \varphi_i(\boldsymbol{p}_j)\boldsymbol{n}_j}{|\sum_j \varphi_i(\boldsymbol{p}_j)\boldsymbol{n}_j|},\ \boldsymbol{d}_i = \frac{\sum_j \varphi_i(\boldsymbol{p}_j)\boldsymbol{p}_j}{\sum_j \varphi_i(\boldsymbol{p}_j)}.$$

Below, the support of $g_i(\boldsymbol{x})$ is denoted as s_i. The surface of the object is approximated by the zero-level set of $f(\boldsymbol{x})$. The value of $f(\boldsymbol{x})$ approximates the signed distance from point x to the surface.

The algorithm to construct $f(\boldsymbol{x})$ is summarized below:

1. Initialize the set of support center candidates $\mathscr{C}$ as all the cell centers in the octree.
2. Randomly select a point c_i from among $\mathscr{C}$ as the center of s_i. Decide the local approximation $g_i(\boldsymbol{x})$ and s_i.
3. From $\mathscr{C}$, remove c_i and points where all eight corners of the corresponding cells are inside s_i.
   If $\mathscr{C}$ is empty, the algorithm terminates. Otherwise go back to the Step 2.

We define the local approximation error as the maximum distance to the approximated plane from the sampling points: $\max_{|\boldsymbol{p}_j - \boldsymbol{c}_i| < r_i} |g_i(\boldsymbol{p}_j)|$. The support size r_i is maximized without exceeding user-specified approximation error tolerance ε. As the experimental results show the approximation error increasing monotonically, we can find r_i through a simple binary search.

2.2. Global Inside/Outside Classification

Since PU is based on a local fitting, it enables high-accuracy representation of objects but is very sensitive to noise, outliers and lack of sampling. Therefore, we make our algorithm more stable with a global algorithm such as Graph-cut [BK04] or FEM approach [SLS*07]. We adopt Graph-cut, the reason is that just a global in/out classification is enough for outlier removal since local approximation has already generated. Moreover, Graph-cut is practically faster and more memory efficient than FEM approach. By combining PU and a Graph-cut approach, surface reconstruction can be conducted more stably than with the PU-only approach (see Fig. 3 for the advantages of Graph-cut combining for very noisy data). The PU-only method fails to detect an appropriate surface because of many outliers, but this algorithm generates surfaces robustly.

In this step, we classify the support centers into two groups: points inside the object and points outside it. First, a graph is constructed by the weighted Delaunay tetrahedrization for the support centers as vertices and their squared radii as weights. Two terminal nodes are introduced as the source node and the sink node; these are purely symbolic rather than representing the vertices of the tetrahedral mesh. Each of them has edges to all vertices.

In our algorithm, vertices labeled as source represent the area inside the object, and other vertices (labeled as sink) represent the outside. We set weak constraints so that vertices with the values of $f(\boldsymbol{v}) \geq 0$ are classified into the source side and otherwise to the sink side.

We assign a cost to an edge incident to one terminal node and vertex v of the tetrahedral mesh as follows: for an edge where one end is a terminal node,

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

k is the ratio of cost for terminal-incident edges to that for tetrahedral mesh edges. According to our experiments, we recommend to set k to about 15. l_v is the average length of edges incident to v. The cost of a tetrahedral mesh edge with end points v_i and v_j is

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

where l_{v_i,v_j} is the length of the edge. Since value $f(\boldsymbol{x})$ represents an approximation of the signed distance of x from the surface, these costs decrease for edges crossing the surface of the object, but increase for edges far from the surface. For the implementation of Graph-cut, we utilize codes distributed by Boykov and Kolmogorov based on [BK04].

The image on the left of Fig. 4 shows a cross section of a resulting tetrahedral mesh of the Stanford dragon. The image on the right of Fig. 4 is a 2D-version image of the results of this operation. Blue round points indicate outside nodes, while yellow squared ones are inside nodes. The signs inside the nodes are those of the approximation function $f(\boldsymbol{x})$ at the nodes. If approximation by $f(\boldsymbol{x})$ is completely correct, the distribution of the sign of $f(\boldsymbol{x})$ should coincide with classification by Graph-cut. In fact, there are non-consistent

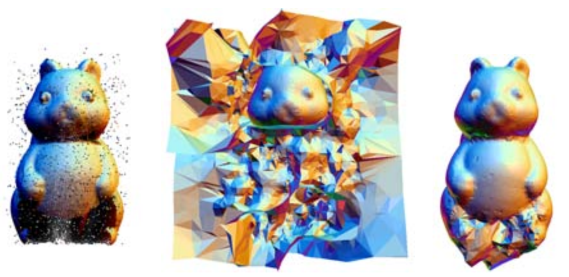


Figure 3: *Left: A set of sampling points including many outliers. Middle: Reconstruction without Graph-cut has undesired extra surfaces. Right: Graph-cut helps to generate stable results.*

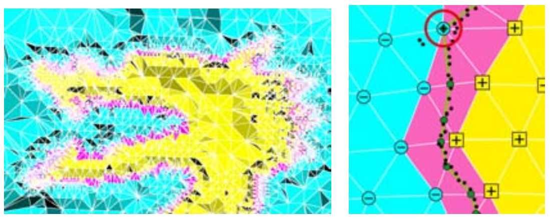


Figure 4: *Left: A cross section of a tetrahedral mesh for the head of the Stanford dragon. The coloring rules follow Right: a 2D-version result. Triangular mesh vertices are depicted as green points, and black points are sampling points.*

nodes; an example is indicated by the red-circled point in the image on the right of Fig. 4. In these areas, approximations have failed because of outliers. Graph-cut provides reliable classification even for such areas.

2.3. Triangular Mesh Extraction

In this step, we extract a triangulated surface using the marching tetrahedra approach. Tetrahedra that have object-inside point(s) and object-outside point(s) intersect with the surface. Intersections which become triangular mesh vertices are calculated on each edge connecting inside and outside points v_i and v_j. In the right image of Fig. 4, vertices are shown with green points. We determine a vertex as a point satisfying $f(\boldsymbol{x}) = 0$ on an edge. Such points are obtained by regula falsi.

If the signs of $f(\boldsymbol{v}_i)$ and $f(\boldsymbol{v}_j)$ are the same, we temporally determine the coordinates of a vertex as one end point whose absolute value of $f(\boldsymbol{x})$ is less than the other. After triangular mesh generation, such vertices are smoothed by bi-Laplacian mesh smoothing. In addition, under-sampled areas are also smoothed. In our current implementation, such areas are notified as vertices on edges; at least one of end points is the center of a support including less than ten sampling points.

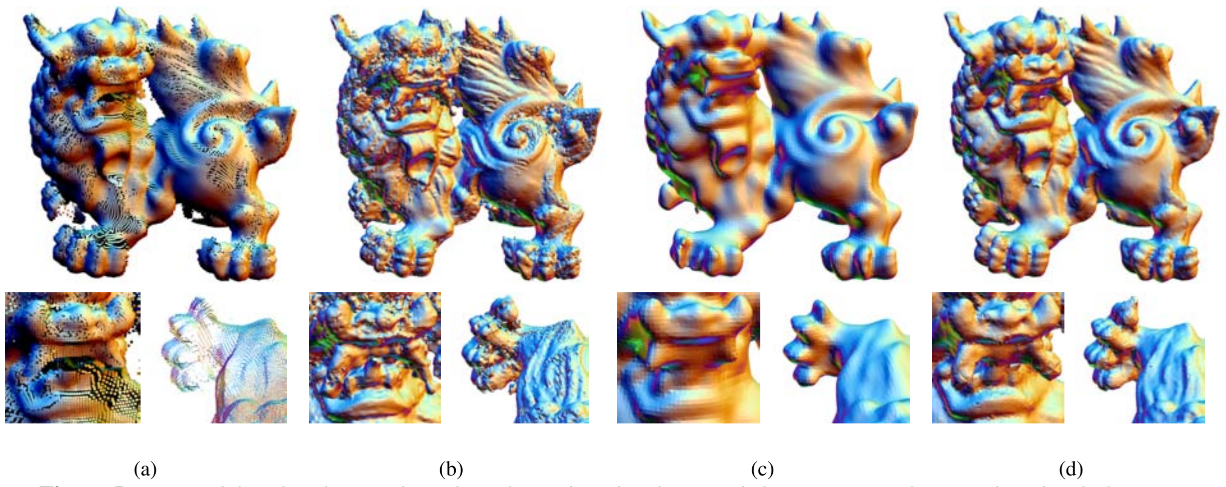


Figure 5: *Scanned data for Shiisa (a lion-shaped guardian dog deity) and close-up views of its mouth and right foot (a). Results of MPU [OBA*03] (b), Poisson surface reconstruction [KBH06] (c) and our algorithm (d). Fangs in the mouth and details of the fingers are successfully reconstructed.*

3. Results and Discussion

In Fig. 5, we compared our results with Poisson surface reconstruction proposed by Kazhdan et al. [KBH06] and with MPU by Ohtake et al. [OBA*03] for data including sparsely sampled areas (Fig. 5(a)). The picture on the left shows the original object. For Poisson surface reconstruction we set the maximum octree level as 10, for MPU error tolerance is set as $1.0\text{x}10^{-3}$, and for ours, the octree level as 8 and the error tolerance ε as $1.0\text{x}10^{-3}$. As an assumption, the input data is scaled with the length of the longest edge of its bounding box as one.

MPU generates very precise results because it adopts quadratic equations, but the result meshes unfortunately contain many tiny extra components (Fig. 5(b)). The accuracy is high but fitting 2-nd order approximation is sensitive to noise. Poisson surface reconstruction can generate watertight meshes, but the results are slightly shrunk (Fig. 5(c)). Moreover, especially for sparse and noisy data, the results tend to be too smooth and many important details are lost. Our algorithm generates precise surface meshes even for such poor scanned data (Fig. 5(d)). The reason is that Poisson surface reconstruction is a 0-th order approximation (a scalar value is assigned to each octant), on the other hand, ours is a 1-st order approximation. A drawback is a necessity of normals. If many normals are reversed, our algorithm cannot achieve a correct result.

RAM is required about 500Mbytes for processing one million input points, and it increases linearly. One bottleneck is time-consumption. For processing a few million points, proposed algorithm takes a few tens of minutes, while Poisson surface reconstruction and MPU works on several minutes. The most expensive procedure is tetrahedrization. We plan to improve this operation in our future work.

In this paper, we have proposed an accurate surface reconstruction algorithm that is robust against noise and outliers. These advantages are achieved by the combination of PU and the Graph-cut techniques.

References

[BK04] Boykov Y., Kolmogorov V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9 (2004), 1124–1137.

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

[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.

[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.

[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.

[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.