Surface Reconstruction from High-Density Points Using Deformed Grids

Keisuke Fujimoto
Department of Computer Science,
The University of
Electro-Communications, Chofu,
Tokyo 182-8585, Japan.
fujimo-k@igo.cs.uec.ac.jp

Toshio Moriya
Advanced Research
Laboratory, Hitachi Ltd.,
Kawasaki, Kanagawa
215-0013, Japan.
toshio.moriya.tm@hitachi.com

Yasuichi Nakayama
Department of Computer Science,
The University of
Electro-Communications, Chofu,
Tokyo 182-8585, Japan.
yasu@cs.uec.ac.jp

ABSTRACT

We present a method of surface reconstruction from high-density scatter points $P = \{p1, p2, ..., pN\}$ without a normal vector in $\mathbb{R}^3$. This method first sets a uniform grid and deforms each cell of the grid by fitting the vertex of each cell to the nearest of the input points. It then constructs triangles according to the pattern of the vertices's state in each cell. Our method can work fast with little memory. We show that our method generated several polygon meshes from real-world range data.

Keywords: Surface reconstruction, Scatter point, Range data.

1 INTRODUCTION

Many surface reconstruction algorithms are used to model real-world objects by computer. Range scanning devices are often used to digitize real-world objects. Because most current scanning devices create scatter points, a surface reconstruction technique that uses scatter points is required.

One of the problems with reconstructing a surface is handling the scatter points on the surface. If there are many such points, reconstructing it fast is difficult.

In this paper, we present a method of surface reconstruction from unorganized high-density points without a normal vector. The method needs a set of points, $P = \{p1, p2, ..., pN\}$, sampled on a surface in $\mathbb{R}^3$. It divides the space into uniform grids, and vertices of each cell are moved to the nearest input points, i.e., each cell is deformed.

Each cell has eight vertices, and because these vertices have two states whether they are moved or not, each cell has $2^8 = 256$ patterns. After rotation is considered, the number of patterns becomes 14. Finally, meshes are constructed according to the pattern of each cell.

Our method is based on the marching cubes algorithm[Lor87]. This algorithm involves using a surface reconstruction method from an implicit function. It find intersections of the uniform grid and the surface and connect these points mutually. Our method targets unorganized points, not an implicit function. The method moves the vertices of each cell to the scatter points, then connects the moved vertices.

Some methods can be used for a surface reconstruction algorithm from the points. One of them connects the points by Voronoi/Delaunay techniques. However the computational complexity increases significantly as the number of points increases. Another method uses the implicit function to approximate the surface. It requires large memory and the computational complexity increases as the number of points increases. Our method is easier to implement and can work faster with little memory than the other method.

2 RELATED WORKS

Since the early 1990s, a substantial amount of work has been done on surface reconstruction from unorganized points. In this section, we present earlier work on this problem. The techniques for reconstructing a surface can be classified roughly by the internal representations. The first technique connects points by the Voronoi or Delaunay algorithm. The second uses the implicit function to approximate the surface, and the mesh is constructed by using the marching cubes method.

2.1 Delaunay triangulations

Delaunay triangulations is a method that connects the points. It maximizes the minimum angle of all the angles of the triangles, and it tends to avoid "sliver" triangles. Amenta approached the reconstruction problem from a computational geometry point of view, focusing on topology reconstruction [Ame98],[Ame01]. Boissonnat proposed a robust method combining the Delaunay approach and implicit function approach[Boi00]. Mederost[Med05] later proposed an algorithm for handling noisy input.

2.2 Implicit function

One of the approaches uses the implicit function to approximate the surface. This approach can handle damaged points and can allow for a complex shape. Hoppe[Hop92] locally estimated the signed distance function as the distance to the tangent plane of the closest point. Then, the method reconstructs the surface as the zero-levelset of the implicit function by using the marching cubes algorithm. Savchenko[Sav95] and Turk[Tur02] used a globally supported RBF(Radial Basis Function) to express by an implicit function. The complexity can be reduced by using a fast multiple method[Car01]. Morse[Mor01] used compactly supported RBF to reconstruct a large data. Ohtake[Oht03] defined the surface locally via quadratic functions that are blended together globally by weights summed to one. Spatial subdivision is used to adapt the resolution to data; sharp features are detected by normal clustering and represented using multiple sets of coefficients.

2.3 Other techniques

Boissonnat[Boi84] made meshes by starting from an initial one and using the greedy approach. Szeliski[Sze93] proposed an approach that uses a dynamic, self-organizing oriented particle system and an efficient triangulation scheme that connects the particles. Jenke[Jen06] proposed a Bayesian statistics-based technique.

3 OUR METHOD

In this section, we present the details of our technique. It makes meshes from points $P \subseteq \mathbb{R}^3$ distributed on the surface without a normal vector. The method has two steps.

1. **Set and deforme cells**
   Set a uniform grid in the space. Each cell is deformed by fitting a vertex of the cell to the nearest input point.

2. **Reconstruct meshes**
   Each cell has eight vertices, and because these vertices have two states whether they were moved or not, each cell has $2^8 = 256$ patterns. After rotation is considered, the number of patterns becomes 14. Then, meshes are constructed according to the pattern of each cell.

We shall now describe each step in detail.

3.1 Setting and deforming grids

In the first step, we set up a uniform grid. Then, each cell of the grid is deformed according to input points. We define h as the cell size, and we give index (i, j, k) to each cell. The cell $C_{i,j,k}(h)$ and the vertices of the cell are at the position shown in Fig.1 It is given by Eq.(1).

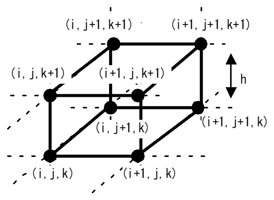

Figure 1: The cell in the grid

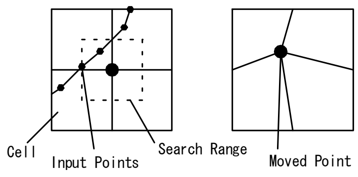

Figure 2: Deformed cell in 2D

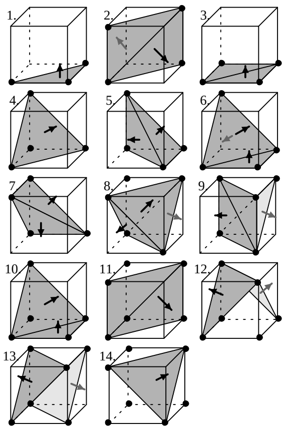

Figure 3: Mesh pattern

$$C_{i,j,k}(h) = [ih, ih+h] \times [jh, jh+h] \times [kh, kh+h] \quad (1)$$

Then, we define $v_{i,j,k} \in \mathbb{R}^3$ as the vertex of each cell in Eq.(2).

$$\mathbf{v}_{i,j,k} = (ih, jh, kh) \quad (2)$$

The vertices are scattered at where grid intersects in the beginning. In the next step, we deform the grid by moving the vertices. We define $\Omega_{i,j,k}$ as the search range of the vertex $v_{i,j,k}$ (Fig.2). We define $q_n \in Q_{i,j,k}$ as points included within the area of $\Omega_{i,j,k}$. Then, we select the nearest point of $Q_{i,j,k}$. The vertex $v_{i,j,k}$ of the cell is moved on the selected point.

$$\min_n(||\mathbf{q}_n - \mathbf{v}_{i,j,k}||)) \qquad (\mathbf{q}_n \in Q_{i,j,k}) \quad (3)$$

Because each cell shares their vertices with the next cell, these two cells sharing the vertex are deformed by moving the vertex, v. When we implement them, we should input the index of vertices v or their address in each cell, not the coordinate values in each cell.

There are possibility of meshes intersecting when it is created by the scatter points. Meshes may intersect when they are created by the scatter points. In our method, we set the range $\Omega_{i,j,k}$ to the vertex, so the intersection of the meshes does not occur. The left half of Fig. 2 shows the state before deforming the cell in 2D. The vertex is sheared by nearby cells, and it is moved to the nearest input point. Then, these cells are deformed as they are on the right half of Fig. 2.

3.2 Generating meshes

In the second step, we construct polygon meshes. Meshes are made by a pattern of the vertices' state of each cell. Each cell is constructed from eight vertices. If an input point around the vertex of each cell, the vertex will move towards the input point. Each cell has eight vertices, and because these vertices have two states whether there are moved or not, each cell has $2^8 = 256$ patterns. After rotation is considered, the number of patterns becomes 14, as shown in Fig. 3. Finally, the meshes are constructed according to the pattern of each cell. In Fig. 3, the black point is moved. If v_i, j, k, $v_i + 1, j, k$, and $v_i + 1, j + 1, k$ are moved, pattern 1 is applied, and these points are connected.

4 EFFICIENCY IMPROVEMENT OF MEMORY

If all cells and the position of points are stored in the memory, the amount of required memory become enormous (it requires $O(n^3)$). In our method, because each grid can be independently calculated, the required memory size can be reduced by copying the results of the previous steps onto the results of the next steps. It requires $O(n^2)$ memory size. In practice, because n is from 50 to 500, the necessary memory size can be obtained.

The next step, only needs a copy of memory, so the overhead of the calculation is small. Pseudocode in Algorithm 1. describes the whole process of surface fitting.

```
function **construct_mesh**()
for $j = 0$ to $Y + 1$ do
    for $i = 0$ to $X + 1$ do
        Set initial position($v_{i,j,0}$)
        Move position($p_{i,j,0}$)
for $k = 1$ to $Z$ do
    Set initial position($v_{0,0,k}$)
    Move position($p_{0,0,k}$)
    for $j = 1$ to $Y + 1$ do
        for $i = 1$ to $X + 1$ do
            Set initial position($v_{i,j,k}$)
            Move position($v_{i,j,k}$)
            Generate mesh($v_{i-1,j-1,k-1}$ to $v_{i,j,k}$)
    Copy from $k+1$ to $k$
```

Algorithm 1: Pseudocode

5 EXPERIMENT

In this section, we describe how we applied our method to several data sets. Because the number of points was very high, We divided the space into 10^3 and allocated the kd-tree in each area so that the search time was shortened,

The experiments were executed on a 1.70-GHz Mobile Pentium with 1.2 GB RAM. The results are shown in Fig.4-6. We compeared our method with the Multilevel Partition of Unity(MPU)[Oht03]. Table 1 shows the computational time and RAM memory for our method and the MPU, We excluded the time for the reading and writing of the file. As a result of the experiment, our method yielded a good result on the computational time and the required memory. The mesh is constructed according to the pattern of each cell of grid in both methods. The MPU have to convert the point set to the implicit function as preprocessing, it becomes overhead. Because Our method generate the mesh directly, it can work fast. And the required memory of our method is less than of MPU. Because it only has to use one step of grid in our method (if the number of grid is 100^3, it use only 100^2 cells.), the size of the memory that our method use is little as described in section 4, while the MPU must have the octree structure of whole region, it requires large memory.

However there is a problem that a loss of meshes can occure when the vertex of grid cannot find any near input point, if the input points are partially space(Fig.7). Therefore, we have to set the grid size according to the distance of the point to the neighbor point or we prepare it correct a hole-filling algorithm [Tek04].

Table 1: Results

Model	Our method		MPU	
point, grid	RAM	time(s)	RAM	time(s)
Bunny 35K, 50^3	4.7 MB	0.3	16 MB	2
Dragon 437K, 130^3	19.6 MB	2.8	41 MB	23
Gargoyle 863K, 300^3	38 MB	26.0	98 MB	48

6 CONCLUSION

We presented a method for reconstructing a surface from high-density unorganized points. Our method uses deformed grids, enabling it to work fast with little memory. In addition, it is simple and easy to implement. However, a loss of meshes can occur if the input

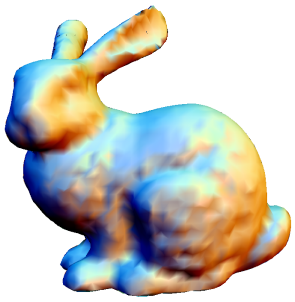
Figure 4: Bunny model

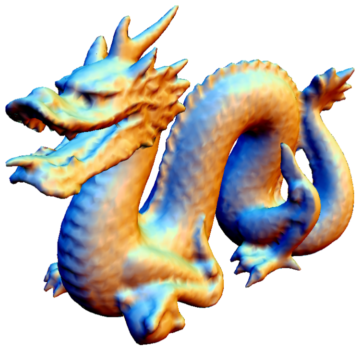
Figure 5: Dragon model

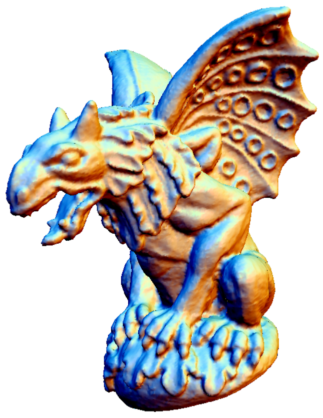
Figure 6: Gargoyle model

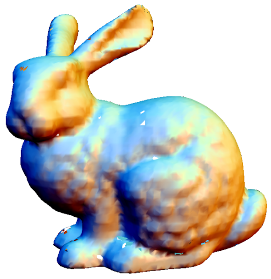
Figure 7: Error result

points are partially sparse. In future work, our method will handle sparse points by applying an octree structure to deal with a multi-level grid.

ACKNOWLEDGEMENT

We would like to thank Stanford University for providing the models of Bunny and Dragon, and the AIM@SHAPE project for providing the models of Gargoyle.

REFERENCES

[Ame98] N.Amenta, M.Bern, and M.Kamvysselis, "A new Voronoi-based surface reconstruction algorithm," In Proceedings of ACM SIGGRAPH, pp.415-421, 1998.

[Ame01] N.Amenta, S.Choi, and R.Kolluri, "The Power Crust," In Proceedings of the 6th ACM Symposium on Solid Modeling, pp.249-260, 2001.

[Boi84] J.D.Boissonnat, "Geometric structures for three-dimensional shape reconstruction," ACM Transactions on Graphics, Vol.3, No.4, pp.266-289, 1984.

[Boi00] J.D.Boissonnat, and F.Cazals, "Smooth Surface Reconstruction via Natural Neighbour," In Proceedings of the 16th Annual ACM Symposium on Computational Geometry, pp.223-232, 2000.

[Car01] 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 Proceedings of ACM SIGGRAPH, pp.67-76, 2001.

[Hop92] H.Hoppe, T.Derose, T.Duchamp, J.McDonald, and W.Stuetzle, "Surface reconstruction from unorganized point," In Proceedings of ACM SIGGRAPH, pp.71-78, 1992.

[Jen06] P.Jenke, M.Wand, M.Bokeloh, A.Schilling, and W. Strasser, "Bayesian point cloud reconstruction," In Proceedings of EUROGRAPHICS, Vol.25, No.3, 2006.

[Lor87] W.E.Lorensen, and H.E.Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," Computer Graphics Vol.21, No.4, pp163-170, 1987.

[Med05] B. Mederos, N. Amenta, L. Velho, and L.H. Figueiredo, "Spectral urface reconstruction from noisy point clouds," In Proceedings of EUROGRAPHICS, pp.53-62, 2005.

[Mor01] B.S.Morse, T.S.Yoo, P.Rheingans, D.T.Chen, and K.Subramanian, "Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions," In Shape Modeling International, pp.89-98, 2001.

[Oht03] Y.Ohtake, A.G.Belyaev, M.Alexa, G.Turk, and H.P.Seidel, "Multilevel Partition of Unity Implicits," In Proceedings of ACM SIGGRAPH, pp.463-470, 2003.

[Sav95] V. V. Savchenko, A. A. Pasko, O. G. Okunev, and T. L. Kunii, "Function representation of solids reconstructed from scattered surface points and contours," Computer Graphics Forum, Vol.14, No.4, pp.181-188, 1995.

[Sze93] R.Szeliski, D.Tonnesen, and D.Terzopoulos, "Modeling surfaces of arbitrary topology with dynamic particles," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp.140-152, 1993.

[Tek04] LS. Tekumalla and E. Cohen, "A Hole-Filling Algorithm for Triangular Meshes," Technical Report UUCS-04-019, School of Computing, University of Utah, 2004.

[Tur02] G.Turk, and J.Obrien, "Modelling with implicit surfaces that interpolate," ACM Transactions on Graphics, Vol.21, No.4, pp.855-873, 2002.