Computer-Aided Design and Applications

http://www.cadanda.com

Feature-Preserving Anisotropic Smoothing for Meshes with Large-Scale Noise

Masatake Higashi[1], Tetsuo Oya[2], Tetsuro Sugiura[1] and Masakazu Kobayashi[1]

[1]Toyota Technological Institute, {higashi, sd07410, kobayashi}@toyota-ti.ac.jp
[2]The University of Tokyo, t-oya@iis.u-tokyo.ac.jp

ABSTRACT

This paper presents a smoothing method which preserves features for a triangular mesh even when large-scale noise are included because of measurement errors. First, scale-dependent discrete Laplacian is introduced along with boundary Laplacian to deal with an open mesh. Then, a method for feature detection which uses the values by these Laplacians is constructed. Furthermore, anisotropic diffusion is proposed which determines suitable parameters from the values for preserving features. Finally a method is presented which discriminates features from large-scale noise by generating graph of feature lines. Effectiveness of the methods is shown by the experiment results of well-smoothed meshes with their features preserved.

Keywords: anisotropic smoothing, feature-preserving, diffusion flow, Laplacian.
DOI: 10.3722/cadaps.2009.365-374

1. INTRODUCTION

Recently, reverse engineering, which generates mathematical models in the computer from real objects, is getting popular according to the advance of the measuring machine. When these mathematical models are used for CAD or CAE, it takes much time for obtaining necessary data for a complicated shape. Further, since the data include noise, a technique for smoothing them is required. The algorithm for the smoothing should be simple and efficient, because the size of the data obtained from a measuring machine is usually large.

Methods [10-11], which minimize the elastic energy of a plate, have been proposed to generate smooth surfaces. However, they include problems such as difficulty to set the boundary conditions and reliability for real data. On the other hand, in the field of computer graphics, Taubin [17] has proposed a method which applies a signal processing technique to removal of noise in a mesh by discrete Laplacian. Kobelt [13] has also proposed a method which corrects positions of points locally by minimizing discrete energy function of a plate. There are many methods which apply discrete Laplacian to smoothing problems; Desbrun et al. [3] have extended the Laplacian technique to a diffusion equation on a mesh. On representing shapes, these studies involve drawbacks in preserving sharp features exactly: Sharp features such as edges and corners are rounded or lost along with noise. For designing industrial products, the smoothing methods are strongly required to preserve sharp features as the designer intends.

In this paper, we propose a smoothing method which preserves sharp features and obtains smooth shapes efficiently for the objects composed of triangular meshes. This method extends the method proposed by the authors [7-8], which obtains smooth surfaces even with inner holes or imperfect

boundaries by applying fourth divided difference to rectangular meshes. The object mesh is expanded from rectangle to triangle meshes which are widely used in CAD and CG. The method uses anisotropic smoothing by employing anisotropic coefficients in a diffusion flow, which was proposed [1][9] for preserving features by setting smaller weights to the direction across them. We obtain results efficiently by detecting sharp features from *smoothing factors* calculated using discrete Laplacians. We have also introduced a graph technique for treating data with large-scale noise.

In this paper, we describe some definitions of discrete Laplacians along with smoothing process by diffusion equations in Section 2. Next, in Section 3, we introduce a new type of Laplacian and a method for smoothing a mesh using it for boundaries of an open shell. Then, in Section 4, we propose a method for extracting sharp features and anisotropic smoothing. In Section 5, we treat a graph technique to obtain correct features when large-scale noise is given, and conclude the paper.

2. SMOOTHING PROCESS

We describe a smoothing process first. We adopt a diffusion flow method [17],[3], which applies a diffusion equation discretely to points on a mesh. At each time step, the coordinates of the points are updated and the surface becomes smooth according to the iteration. Let a state quantity be X, then the diffusion process is expressed as

$$\frac{\partial X}{\partial t} = \lambda \boldsymbol{L}(X). \tag{2.1}$$

Here, t is a time variable, λ is a diffusion coefficient, and $\boldsymbol{L}$ is a differential operator, that is Laplacian. To solve this equation by integrating it, there are two methods: explicit and implicit. An explicit method gets a solution quickly, but it is unstable. On the other hand, an implicit method needs to solve a simultaneous equation and obtains a solution without fail taking a longer computation time.
Let a coordinate value on a mesh, its present state and the state at the next step be v, v^n and v^{n+1}, then we get an explicit equation:

$$v^{n+1} = (I + \lambda dt \boldsymbol{L}) v^n. \tag{2.2}$$

Here, I is an identity matrix. We directly obtain v^{n+1} by calculating the right side. The stable condition is $\lambda dt < 1$. An implicit equation is deduced [3] as

$$(I - \lambda dt \boldsymbol{L}) v^{n+1} = v^n. \tag{2.3}$$

We obtain a solution by solving a simultaneous equation, but we have to use coordinate values at the nth step because we cannot get coefficients at the left side. This method is called semi-implicit and supposes that the lengths of the edges are not changed much before and after the calculation [1],[3],[9]. In either case, we move the coordinate of a point according to the obtained discrete Laplacian value by the diffusion coefficient λ. We use the simple explicit method if there is no problem of stable convergence. We stop iteration when the sum of absolute Laplacian values does not decrease.

3. INTRODUCTION OF DISCRETE LAPLACIAN

3.1 Formula of Discrete Laplacian

A basic diffusion formula is represented by differential operator $\boldsymbol{L}$ for point v_i on a mesh, and expressed as follows:

$$\boldsymbol{L}(v_i) = \sum_{j \in N_i} \omega_{i,j} (v_j - v_i). \tag{3.1}$$

Here, $\omega_{i,j}$ is a weight coefficient which is operated to one–neighbor vertex j, and N_i denotes one-neighbor vertices of vertex i. The simplest weight is $\omega_i = 1/|N_i|$, where $|N_i|$ is a valence of vertex i. In this case, the length and angle of edges from vertex i are supposed to be equal, but these are not satisfied for a general mesh. Therefore, Fujiwara [5] has introduced a scale dependent formula:

$$L(v_i) = \frac{1}{E} \sum_{j \in N_i} \frac{(v_j - v_i)}{|v_j - v_i|}, \quad E_i = \sum_{j \in N_i} |v_j - v_i|. \tag{3.2}$$

This can be applied to a mesh with non-uniform edges.

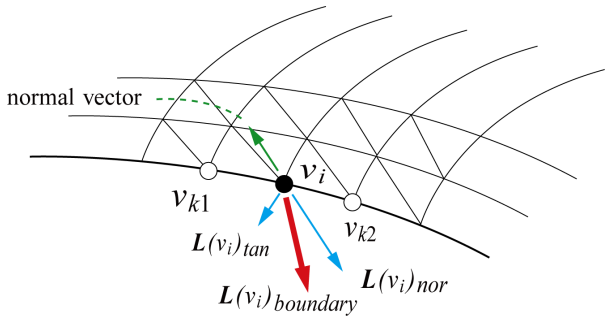


Fig. 1: Calculation of discrete Laplacian of an open Mesh.

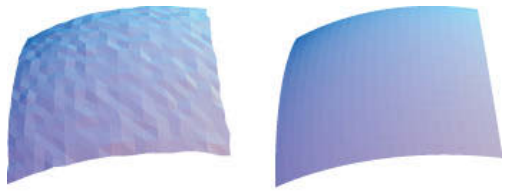


Fig. 2: An open surface (441 vertices and 800 faces). Left: With noise, Right: Smoothed.

To consider curvature variation, we need second-order Laplacian. Second–order Laplacian $\boldsymbol{L}^2$ can obtain precise results because it calculates two-neighbor vertices, that are one-neighbor of vertices *j*'s of one neighbor of *i*. But a higher order than two is not necessarily effective because there is a trade-off between computation time and quality. Second order Laplacian is expressed using one-order Laplacian:

$$\boldsymbol{L}^2(v_i) = \sum_{j \in N_i} \frac{L(v_j) - L(v_i)}{|\, v_j - v_i \,|}. \tag{3.3}$$

3.2 Simplified Discrete Laplacian

To get solutions efficiently, we introduce a simplified Laplacian operator as follows:

$$\boldsymbol{L}(v_i) = \sum_{j \in N_i} \frac{(v_j - v_i)}{|\, v_j - v_i \,|}. \tag{3.4}$$

This formula is similar to that, introduced by the authors [7-8] as the divided difference. This formula is equal to Eqn. (3.1) when $\omega_{i,j}$ is set to $(v_j - v_i)^{-1}$, and $1/E_i$ is removed from Eqn. (3.2). Second-order Laplacian is calculated using this one-order Laplacian. Since it normalizes the length of each vector from vertex *i* to the one neighbor vertex, it can treat a non-uniform mesh. From our experiments, almost meshes are well smoothed. When we treat an extremely irregular mesh, Laplacian-Beltrami operator [3][14][21] is necessary which counts angles around the vertex in the weight.

3.3 Boundary Treatment for Open Mesh

When we treat an open mesh, we have to consider the condition of vertices on boundaries. The discrete Laplacian on a boundary is difficult to get a suitable value because of a large tangential component. No good concrete solutions have been proposed, although Desbrun and et al. [3] proposed that the discrete Laplacian is defined using two adjacent vertices or that virtual vertices are used for the calculation. Taubin [18] takes out only normal component for vertices on a boundary, but some troubles occur when tangential components are large. Hence we use both components of Laplacian, normal component $\boldsymbol{L}(v_i)_{nor}$ and tangential one $\boldsymbol{L}(v_i)_{\tan}$, on a boundary as shown in Fig. 1.

$$\boldsymbol{L}(v_i)_{nor} = \left(\boldsymbol{n_i} \cdot \sum_{j \in N_i} \frac{(v_j - v_i)}{|\, v_j - v_i \,|} \right) \boldsymbol{n_i} \tag{3.5}$$

$$\boldsymbol{L}(v_i)_{\tan} = \boldsymbol{K_i} - (\boldsymbol{n_i} \cdot \boldsymbol{K_i})\boldsymbol{n_i} \tag{3.6}$$

Here, $\boldsymbol{n}_i$ is a normal vector at vertex *i*. We obtain $\boldsymbol{n}_i$ from the normalized average of outer products of edge vectors for all triangles at the vertex. $\boldsymbol{K}_i$ is calculated as follows using vertex *i* and its adjacent vertices v_{k1} and v_{k2}.

$$\boldsymbol{K}_i = \frac{v_{k1} - v_i}{|v_{k1} - v_i|} + \frac{v_{k2} - v_i}{|v_{k2} - v_i|} \tag{3.7}$$

Using these values, we calculate Laplacians on a boundary as follows:

$$\boldsymbol{L}(v_i)_{nor} = \begin{cases} \boldsymbol{L}(v_i)_{nor} & |\boldsymbol{L}(v_i)_{\tan}| \geq \varepsilon, \\ \boldsymbol{L}(v_i)_{nor} + \boldsymbol{L}(v_i)_{\tan} & |\boldsymbol{L}(v_i)_{\tan}| < \varepsilon. \end{cases} \tag{3.8}$$

Here, ε is an arbitrary constant, and it is a threshold which determines whether a tangent component is included or not. Fig. 2 shows a result of an applied example for a simple mesh. We have obtained a smooth mesh by the second Laplacian and implicit diffusion flow. In this example, we set $\varepsilon = 0.3$. It shows the effectiveness of our method.

4. SMOOTHING METHOD FOR PRESERVING SHARP FEATURES

4.1 Extracting Sharp Features

We propose a method for extracting sharp features. Here, we define a sharp feature as an edge or a corner on a mesh. We introduce two kinds of *fairness factors* to distinguish existing features from large noise. First one is $\boldsymbol{F}_i$ for detecting unevenness of one-neighbor vertices of vertex i, which is defined by the normal component of a normalized discrete Laplacian. This is quite the same to Eqn. (3.5). When unevenness around a vertex is small, this factor is small, and it becomes large for a large unevenness. The second factor is $\boldsymbol{G}_i$ which represents unevenness of one neighbor vertices j's. This factor is expressed by a vector which is the sum of $\boldsymbol{F}_i$ and the sum-up of $\boldsymbol{F}_j$'s of its one-neighbor vertices:

$$\boldsymbol{G}_i = \sum_{j \in N_i} \boldsymbol{F}_j + \boldsymbol{F}_i. \tag{4.1}$$

This factor does not include the unevenness of vertex i, because the vectors of $\boldsymbol{F}_i$ from vertex i to one-neighbor vertices are cancelled by the vectors of $\boldsymbol{F}_j$'s from one-neighbor vertices to vertex i. Hence $\boldsymbol{G}_i$ includes only out-going vectors from one-neighbor vertices, and it is not affected by large-scale noise at vertex i and represents only the unevenness of the one-neighbor. On the other hand, at a vertex on a feature, both $\boldsymbol{F}_i$ and $\boldsymbol{G}_i$ becomes a large value, because curvature is not continuous there and a direction of normal vector changes abruptly as shown in Fig. 3. Therefore, we can judge that a vertex is on a feature when both the factors of the vertex are large, and it has noise when only either of the factors is large.

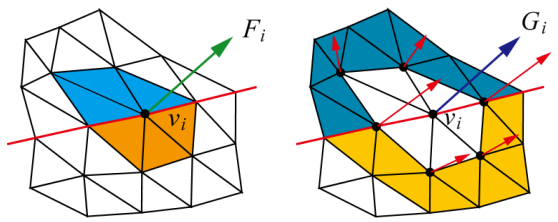


Fig. 3: Vectors of $\boldsymbol{F}_i$ and $\boldsymbol{G}_i$ on feature line. Left: $\boldsymbol{F}_i$, Right: $\boldsymbol{G}_i$.

Next, using these factors, we introduce two functions which have standard deviations σ_F and σ_G:

$$W_F = 1 - \exp\left(-\frac{\boldsymbol{F}_i^2}{2\sigma_F^2}\right), \quad W_G = 1 - \exp\left(-\frac{\boldsymbol{G}_i^2}{2\sigma_G^2}\right). \tag{4.2}$$

Furthermore, we calculate geometrical values for vertices' pair, v_L and v_R, of an edge; we multiply W_F's and W_G's at both ends of each edge and create a factor pair: $(W_{FL} \times W_{FR}, W_{GL} \times W_{GR})$. Then, we define an *edge factor*, multiplying this pair by Ψ which is the sine of the included angle at the edge. By plotting these pairs of the factors as a distribution map, we extract feature edges. When the value of W_F is near 0 for both the vertices, the multiplied value also becomes near 0: $W_{FL} \times W_{FR} \approx 0$. And only

when both the values of W_F are near 1, $W_{FL} \times W_{FR} \approx 1$ is satisfied. This is strengthened by adding the effect of Ψ.

When we make a distribution map of the factors on a space with vertical axis $\Psi \times W_{FL} \times W_{FR}$ and horizontal axis $\Psi \times W_{GL} \times W_{GR}$ as shown in Fig. 4, plotted points for feature edges exist in an upper right region. Setting a threshold value, we extract feature edges by separating them from the other edges with the threshold. Corner points are obtained as the intersection of feature edges.

Fig. 5 shows an example of the extraction of feature edges for an octahedron; Left figure shows input data with noise and right figure shows an output. This method is demonstrated effectively by extracting all the boundary edges of faces in spite of included noise.

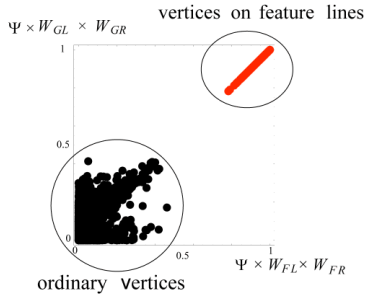


Fig. 4: Distribution of calculated pair of $(\Psi \times W_{FL} \times W_{FR}, \Psi \times W_{GL} \times W_{GR})$ for feature detection.

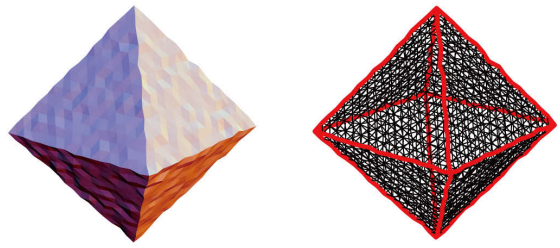


Fig. 5: An octahedron model (1026 vertices and 2048 faces). Left: With noise, Right: Extracted result.

4.2 Feature Preserving Smoothing

In early isotropic smoothing methods such as by Taubin [17] and by Kobbelt[13], sharp edges are rounded and features are lost. Hence many methods to prevent loss of features have been proposed. Bilateral filtering [12],[4] is application of image processing to three dimensional surfaces: Fleishman [4] has executed filtering of noise by regarding a distance from a surface point to a tangential plane at a vertex as a brightness value of images. When the distance is large, it is regarded as a feature. They succeeded in filtering noise, but the obtained smoothness is not so high. Ohtake [16] has introduced a method which uses adaptive Gaussian filter, but its computational cost is high.

Another effective method is anisotropic smoothing which sets different weights on vertices and smoothes a mesh in an anisotropic way for diffusion tensors according to whether a vertex is on a feature edge or not. The weighting function is shown below.

$$\omega(a) = \begin{cases} 1 & \text{for } |a| \le \tau, \\ \dfrac{\tau^2}{\tau(\tau - |a|)^2 + \tau^2} & \text{for } |a| > \varepsilon. \end{cases} \tag{4.3}$$

Here, τ is a threshold which is a constant to determine whether the vertex is on a feature or not. Variable a is quantity concerning a vertex; Clarenz et al. [1] used principal curvature for this value. This method also needs a high computational cost to calculate principal curvature for a large mesh. Hildebrandt et al. [9] have proposed a two-step smoothing method in anisotropic smoothing; First, they calculate mean curvature and smooth its scalar surface, then generate a smooth mesh by converging it to have the mean curvature surface. This also requires much computation time.

We propose a method which makes use of merits of the anisotropic smoothing and reduces a computational cost by applying discrete Laplacian in Eqn. (3.4) to the weight in Eqn. (4.3):

$$\omega(\boldsymbol{L}(v_i)) = \frac{\tau^2}{r(\tau - |\boldsymbol{L}(v_i)|)^2 + \tau^2}, \quad \text{for } |\boldsymbol{L}(v_i)| > \tau. \tag{4.4}$$

We can reduce the computational time using $|\boldsymbol{L}(v_i)|$ also in determining features. We can select an appropriate value for threshold τ, which is decided by users in the conventional method by trial and error. Instead, we execute feature extraction in the preprocessing: We set τ so that $|\boldsymbol{L}(v_i)|_{Feature} > \tau$ for all the set of feature vertices. Fig. 6 shows an example of setting for the octahedron in Fig. 5. A horizontal axis is index number for vertices and a vertical axis is $|\boldsymbol{L}(v_i)|$, and big black circles show vertices on features. We set $\tau = 0.9$ in this example, and we exactly separate feature vertices from ordinary ones.

When a vertex is judged to be on a feature according to the method described in Section 4.1, the weight is set to 1 for the edges along the feature and set to ω for those across the feature, and it is given to all the edges when the vertex is at the corner. This can reserve the features and also smooth the feature edges themselves. One-order Laplacian operators for the anisotropic smoothing at the feature vertices become

$$\boldsymbol{L}(v_i) = \begin{cases} \sum_j^{N_i} \dfrac{v_l - v_i}{|v_j - v_i|} + \omega_i \sum_j^{N_i} \dfrac{v_j - v_i}{|v_j - v_i|} & \text{for edge,} \\ \omega_i \sum_j^{N_i} \dfrac{v_j - v_i}{|v_j - v_i|} & \text{for corner.} \end{cases} \tag{4.5}$$

Here, v_l is a vertex of one neighbor of v_i and also on a feature, and v_j is a vertex of one neighbor of v_i but not on a feature.

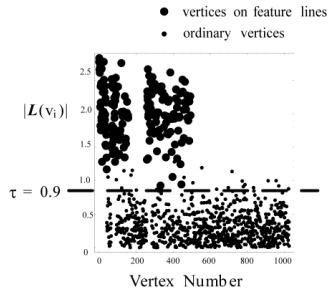
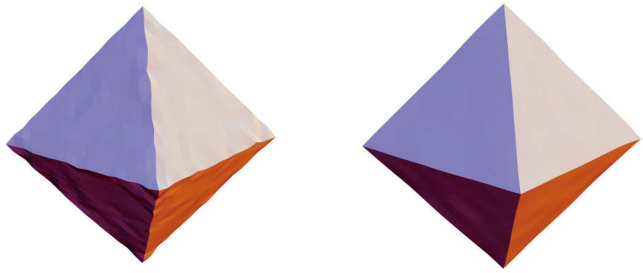


Fig.6: Values of $|\boldsymbol{L}(v_i)|$ on each vertex.

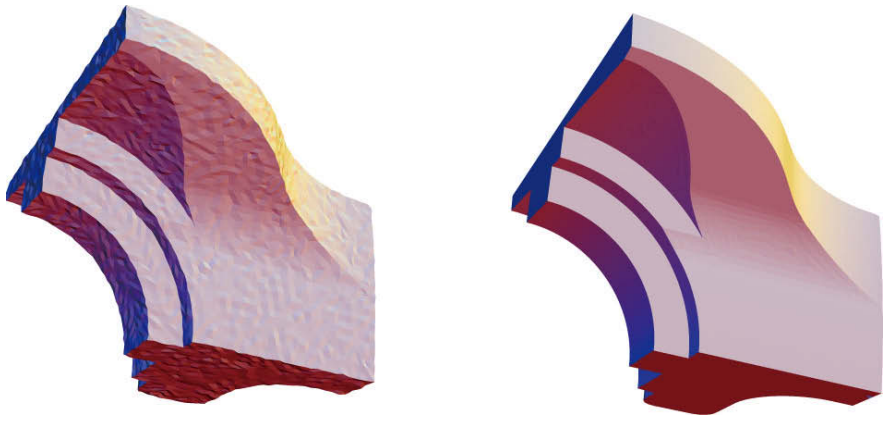
4.3 Example

We executed the proposed method for some examples. They were calculated by the explicit diffusion method and the anisotropic second Laplacian operator was applied. Fig. 7 shows an example for an octahedron; Fig. 7(a) is calculated by Fleishman's bilateral filtering [4] and Fig. 7(b) is by our method, which removes noise completely while the feature lines are preserved. Jones' method [12] shows a similar result to that of Fleishman's because both the bilateral filtering cannot smooth feature lines successfully.

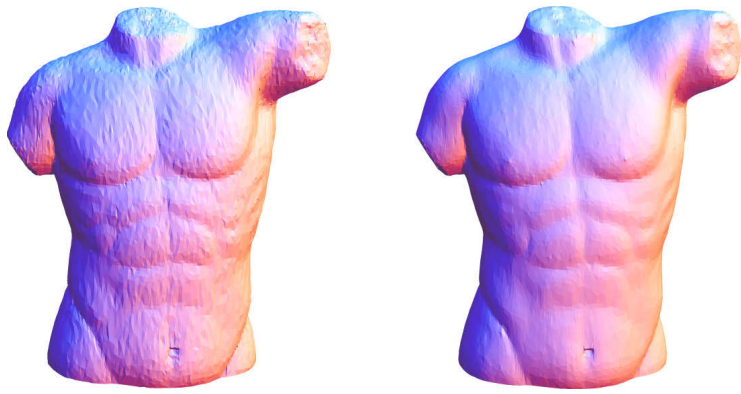
The next example is a fandisk model. Fig. 8(a) shows a shape which is given random noise according to Gaussian distribution, and the smoothed result is shown in Fig. 8(b). It is demonstrated that the method is effective for a complicated shape as well as one with an irregular mesh. The last example is a plaster, whose height is 13 cm and obtained through actual measurement; A 3D scanner is Roland PICZA LPX-1200. Fig. 9(a) shows original measured data and Fig. 9(b) the smoothed result. In this example, feature lines such as muscle lines are remained but the noise is not completely removed. The reason is that the feature lines are composed of valleys with small $\boldsymbol{G}_i$'s and it is difficult to set an



(a) Fleishman's bilateral filtering (b) Proposed anisotropic smoothing
Fig. 7: Smoothed results for octahedron.



(a) Original mesh with noise (b) Smoothed result
Fig. 8: Fandisc model (6,457 vertices and 12,946 faces).



(a) Mesured mesh with noise (b) Smoothed result
Fig. 9: Plaster model (8,368 vertices and 16,372 faces).

appropriate parameter for τ against these vertices. To detect vague and gradual features is future research work. It might be effective to use a technique for recovering sharp features such as those by Wang [19-20].

5. SMOOTHING FOR LARGE-SCALE NOISE BY GRAPH TECHNIQUE

5.1 Extraction of Feature Edges

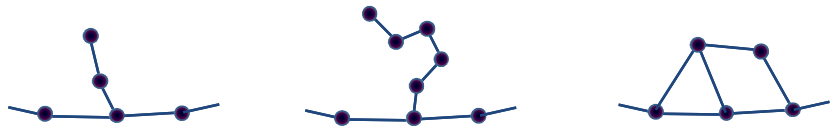
In our method, the edges which are not on feature lines are extracted when large-scale noise is given or features include edges with thin angles. Here, large-scale noise means that the standard deviation of given noise is greater than 0.2 $\times$ average edge-length in case of the fandisc model. Hence, we select edges on features not only by the fairness factors but also by connectivity of edges using a graph technique. The graph technique for extracting features has been proposed by Gumhold et al. [6] and Pauly et al. [15]. They extracted features from point data. On the other hand, Demarison et al. [2] first executed segmentation for points data and then removed small branches using the graph which were obtained from the segments. We apply a graph for extracted edges as sharp features and refine the graph according to the Demarison's approach.

We first generate a graph *G(V, E)* from the extracted edges which is composed of vertices (*V*) and edges (*E*), then we calculate *edge factors* which are distances from the origin to distributed points in Fig. 4 . Then we calculate *tree factors* by summing up the factor values of connected edges in each tree. Second, using these values, we select trees which have larger tree factors than the given constant and remove unimportant trees consisting of a small number of edges. Third, we remove mis-extracted edges according to the following patterns (see Fig. 10):

(1) Short branch edges
(2) Zigzag edges
(3) Small loops

These can be obtained using topological relation of the graph along with the directions of adjacent edges.

Last, we restore disappeared edges by connecting edges which are recognized feature edges. At the leaves of the remained trees, that are ends of sharp edges, we check if there are sharp edges to be connected. The intermediate edges to connect both the ends must have larger edge factors than half of the threshold and the number of them is less than four. The angle between the connecting edge and the adjacent existing edge is less than the tolerance.



(1) Short branch edges (2) Zigzag edges (c) Loops

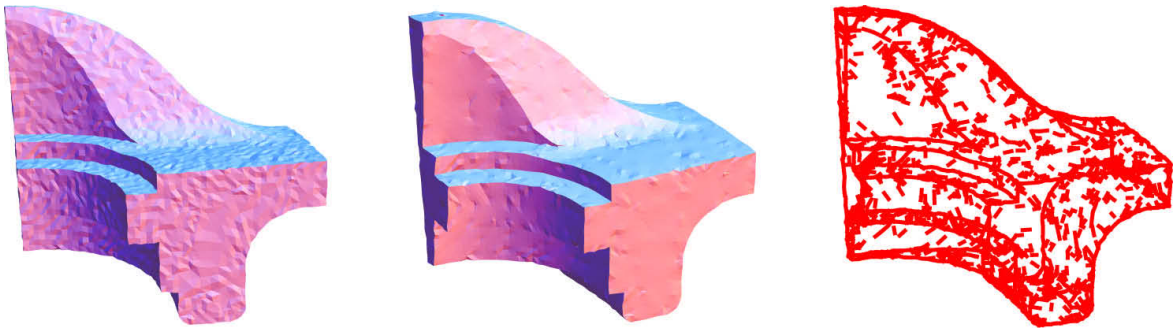
Fig. 10: Patterns for mis-extracted edges.

5.2 Applied Example

We execute our method for the same data as Fig. 8, but they have large-scale noise as shown in Fig. 11(a). The smoothed result of the method proposed in Section 4 is shown in Fig 11(b) using extracted feature edges shown in Fig. 11(c). The result is not so good because of many mis-extracted sharp edges. Fig. 12 shows smoothed result using the graph method in this section. Fig. 12(b) shows the result after removing mis-extracted edges, and Fig. 12(a) shows the result by the anisotropic smoothing with the weights of the remained features. The blue edges show restored sharp edges. The quality of the shape is much improved.

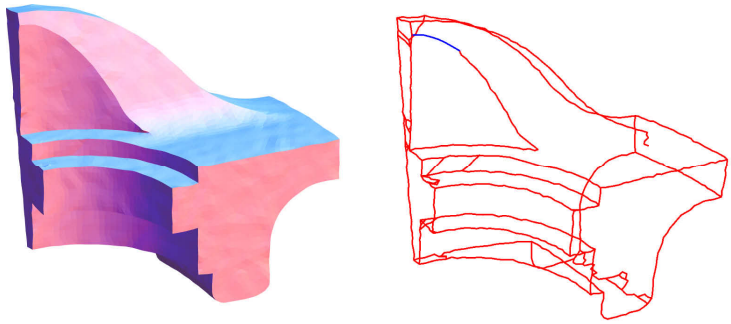
6. CONCLUSION

We have proposed methods for smoothing shapes with large-scale noise. By applying the methods to meshes with noise, we have obtained following conclusions.



(a) Original mesh with noise (b) Smoothed result (c) Extracted feature edges

Fig. 11: Smoothing of fandisc model with large noise.



(a) Smoothed result (b) Extracted feature edges

Fig. 12: Smoothing of fandisc model by removing mis-extracted feature edges.

- Extraction of sharp features is possible by fairness factors calculated with scale-dependent discrete Laplacian.
- Anisotropic smoothing can be effectively executed using weights obtained also from discrete Laplacian.
- A mesh with large-scale noise can be smoothed by restoring the graph of the extracted feature edges.

Our future research includes extending the method to treat shapes with vague sharp features and fillets.

7. ACKNOWLEDGEMENTS

This research was partly supported by Grant-in-Aid for Scientific Research (C) of MEXT, Japan government.

8. REFERENCES

[1] Clarenz, U.; Diewald, U.; Rampf, M.: Anisotropic geometric diffusion in surface processing, Proc. of IEEE Visualization, 2000, 397–405.

[2] Demarsin, K.; Vanderstraeten, D.; Volodine, T.; Roose, D.: Detection of Closed Sharp Feature Lines in Point Clouds for Reverse Engineering Applications, GMP2006, LNCS4077, 2006, 571–577.

[3] Desbrun, M.; Meyer, M.; Schröder, P.; Barr, A. H.: Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow, SIGGRAPH 99, 1999, 317–324.

[4] Fleishman, S.; Drori, I.; Cohen-Or, D.: Bilateral Mesh Denoising, ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2003), 22(3), 2003, 950–953.

[5] Fujiwara, K.: Eigenvalues of Laplacians on a closed Riemannian manifold and its nets, Proc. of

AMS 123, 1995, 2585–2594.
[6] Gumhold, S. etc.: Feature Extraction from point clouds, Proc. of the 10th IMR, 2001, 293–305.
[7] Higashi, M.; Yamada K.: Smoothing of Mesh Data Using Fourth Divided Differences, Journal of the Japan Society for Precision Engineering, 67(5), 2001, 749–753 (in Japanese).
[8] Higashi, M.; Yamada K.: Smoothing of Mesh Data Using Fourth Divided Differences (2nd Report) Application to Mesh with Defect Points and C^1 Continuity–, Journal of the Japan Society for Precision Engineering, 69(8) , 2003, 1135–1140 (in Japanese).
[9] Hildebrandt, K.; Polthier, K.: Anisotropic Filtering of Non-Linear Surface Features, Comput. Graph. Forum (Proc. EUROGRAPHICS 2004), 23(3), 2004, 391-400.
[10] Hosaka, M.: Theory of Curves and Surface Synthesis and Their Smooth Fitting, Information Processing in Japan, 10(3), 1969, 60-68.
[11] Hoscheck, J.; Lasser, D.: Fundamentals of Computer Aided Geometric Design, A K Peters, Wellesley, MA, 1993.
[12] Jones, T.; Durand, R. F.; Desbrun, M.: Non-Iterative, Feature-Preserving Mesh Smoothing, ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2003), 22(3), 2003, 943–949.
[13] Kobbelt, L.; Campagna S.; Vorsatz J.; Seidel H.-P.: Interactive multi-resolution modeling on arbitrary meshes, SIGGRAPH98, 1998, 105–114.
[14] Meyer, M.; Desbrun, M.; Schröder, P.; Barr A.: Discrete Differential-Geometry Operator for Triangulated 2-manifolds, Proc. VisMath '02, Berlin, Germany, 2002.
[15] Pauly, M. et al.: Multi-scale feature extraction on point-sampled surfaces, Comput. Graph. Forum, 22(3), 2003, 281–290.
[16] Ohtake, Y.; Belyaev, A.; Seidel, H. -P.: Mesh Smoothing by Adaptive and Anisotropic Gaussian Filter Applied to Mesh Normals, Proc. VMV2002, Erlangen, 2002, 203-210.
[17] Taubin G.: A Signal Processing Approach to Fair Surface Design, Proc. SIGGRAPH95, 1995, 351–358.
[18] Taubin, G.: Linear anisotropic mesh filtering, IBM Research Report, RC2213, 2001.
[19] Wang, C. C. L.: Bilateral recovering of sharp edges on feature-insensitive sampled meshes, IEEE Transactions on Visualization and Computer Graphics, 12(4) , 2006, 629-639.
[20] Wang, C. C. L.: Incremental reconstruction of sharp edges on mesh surfaces, Computer-Aided Design, 38(6) , 2006, 689-702.
[21] Xu, G.: Convergence of discrete Laplace-Beltrami operators over surfaces, Computers and Mathematics with Applications, 48, 2004, 347–360.