Anisotropic Feature-Preserving Smoothing of 3D Mesh

Tang Jie, Zhang Fuyan
State Key Laboratory for Novel Software Technology, Nanjing University,
Department of computer science and technology, Nanjing University, Nanjing 210093
jietang@graphics.nju.edu.cn, fyzhang@graphics.nju.edu.cn

Abstract

We present an anisotropic 3D mesh smoothing method which is effective and feature-preserving. The algorithm is originated from bilateral mesh denoising and improved in two aspects. Firstly, the geodesic distance, instead of Euclidean distance, is adopted for the selection of neighboring vertices. Secondly, the way to calculate smoothing offsets is adjusted and a normal weight is introduced. Compared with previous smoothing methods, our method doesn't need to repeat the smoothing operations and could preserve the features of original model with relatively faster speed. Finally, we provide a series of examples to graphically and numerically demonstrate the quality of our results.

1. Introduction

In many computer graphics applications, polygonal meshes, especially triangle meshes, deliver a simple and flexible way to represent and handle complex geometric objects. Dense triangle meshes are often obtained from points sampled over real world objects. With the advent and advances in 3D scanning and acquisition technology, models obtained this way are becoming widely available. But due to the inevitable physical noise added by a scanning device, points sampled from a 3D object often do not reflect their correct locations, resulting in meshes containing undesirable rough features. Thus, mesh smoothing, or denoising, is utilized to improve mesh quality by adjusting the locations of grid points in the mesh without changing the mesh topology. The ultimate goal of mesh smoothing is to produce highly smooth meshes efficiently, for rendering, modeling, and visualization, while still preserving the basic overall shape and important features of the original model.

The most commonly used smoothing technique is Laplacian smoothing, which moves a given node to the geometric center of its incident nodes. Various weighted Laplacian smoothing algorithms have been developed to improve the performance of the original smoothing technique. Laplacian smoothing is computationally inexpensive but does not guarantee improvement in mesh quality. The recent development in mesh smoothing shows that the feature-preserving smoothing methods are getting more and more attentions.

The bilateral mesh denoising (BMD), introduced by Fleishman [1], is a nonlinear filter derived from Gaussian blur, with a feature preservation term that operates on the geometric component of the mesh. The algorithm is practical, clear, simple and can deals with irregular meshes. Furthermore, since the algorithm only modifies vertices in the normal direction, thus no reparameterization is performed.

However, during implementing BMD method, we found that it has some disadvantages:

- It uses Euclidean distance to measure the distance between two points on the mesh, which could introduce distortion where the model is thin.
- Distortion appears at areas with sharp features.
- BMD implement the smoothing identically over the whole mesh surface, no matter the local shape is flat or steepy.

In this paper, we present an anisotropic 3D mesh smoothing method which is effective and feature-preserving. The algorithm is originated from BMD method and improved in two aspects. Firstly, the geodesic distance, instead of Euclidian distance, is adopted for the selection of neighboring vertices. Secondly, the way to calculate smoothing offsets is adjusted and a normal weight is introduced. Compared with previous smoothing methods, our method doesn't need to repeat the smoothing operations and could preserve the features of original model with relatively faster speed.

The rest of this paper is organized as follows: Section 2 will present a brief overview of related work in the literature. Section 3 describes the bilateral mesh denoising algorithm briefly. In Section 4, we introduce our improvement of neighboring vertices selection. The process to calculate smoothing offsets is described in Section 5. In Section 6, a normal weight function is introduced. We compare the smoothing results of our method to several other methods in Section 7. And we draw a conclusion in section 8.

2. Related work

The most common approaches of mesh smoothing are variants on Laplacian smoothing [2]. Laplacian smoothing usually leads to shrinkage and visible shape distortion. Taubin [3] remedied this problem using λ - μ filter, where pass-band frequencies can be retained, but they are also amplified slowly as the degree of the λ - μ polynomial increases. Vollmer et al. [4] tackled the shrinkage problem in the spatial domain. Using their HC algorithm, each vertex is moved back towards a weighted average of its original position and its previous position, after each step of Laplacian smoothing. A common drawback of HC algorithm and λ - μ filtering is their slow smoothing speed for large mesh models. A further development of Laplacian smoothing method, mesh smoothing by mean curvature flow was introduced by Desbrun et al. [5], they observed that fairing surfaces can be performed in the normal direction. Zhang [6] and Yagou [7] made further improvement on Laplacican smoothing.

A disadvantage of isotropic smoothing method is that sharp features such as creases and corners are usually diffused and lost after isotropic smoothing. To preserve sharp features, anistropic smoothing schemes have been proposed recently. The main idea of anistropic smoothing consists of non-uniform smoothing in different directions. The first anisotropic smoothing scheme for height fields was introduced by Desbrun et al. [8]. Peng et al. [9] applied locally adaptive Wiener filtering to meshes. Clarenz et al. [10] formulated and discretized anisotropic diffusion for meshes. A principle curvature threshold is used to detect edges explicitly and locally. Recently, Bajaj and Xu [11] achieved impressive results by combining the limit function of loop subdivision scheme with anisotropic diffusion. Tasdizen et al. [12] applied anisotropic diffusion to normals of the level-set representation of the surface, and in the final step, the level-set is converted to a mesh representation. Another impressive smoothing method was put forward by Jones et al. [13]. It is non-iterative and feature preserving, but it contains both geometric and parameter smoothing.

3. Bilateral Mesh Denoising

BMD filters a mesh using local neighborhoods. For a mesh with m vertices, the main idea of BMD is to shift each vertex v_i an offset d_i along its normal direction n_i, which is described as follows:

$$v_i' = v_i - n_i \cdot d_i \qquad i=0, 1, \ldots, \text{n-1}$$

The offset d_i is calculated using local neighborhoods as follows:

$$d_i = \frac{\sum_{v_j \in N(v_i)} n_i \cdot (v_i - v_j) W_c(\| v_i - v_j \|) W_s(n_i \cdot (v_i - v_j))}{\sum_{v_j \in N(v_i)} W_c(\| v_i - v_j \|) W_s(n_i \cdot (v_i - v_j))}$$

where $N(v_i)$ is the neighborhood of v_i. The closeness smoothing filter is a standard Gaussian filter with parameter σ_c: $W_c(x) = e^{-x^2/\sigma_c^2}$, and a feature-preserving weight function, which they refer to as a *similarity weight function*, with parameter σ_s that penalizes large variation in intensity, is: $W_s(x) = e^{-x^2/\sigma_s^2}$. In practice, $N(v_i)$ is defined by the set of points $\{ v_k \}$, where $\| v_i - v_k \| < \rho = 2\sigma_c$, which is the Euclidean distance of two vertices. The parameters, σ_c and σ_s, are specified by the user. Larger σ_s means more neighborhood vertices, and larger σ_s means bigger offset.

The geometrical explanation of BMD is shown in figure 1, which is a 2D analog. t is the tangent plane at vertex v_i, $\{ v_{i-2}, v_{i-1}, v_{i+1}, v_{i+2}\}$ is the neighborhood of v_i. $n_i \cdot (v_{i+1} - v_i)$ is the projection of vector $(v_{i+1} - v_i)$ onto the normal vector n_i, which is represented by a dash line in the figure. Therefore we could see that the offset of vertex v_i is actually the weighted summation of the lengths of those dash lines.

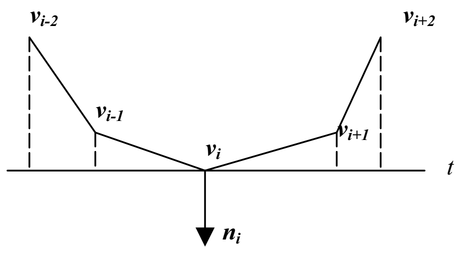

Figure 1. **Geometric explanation of BMD**

BMD method is simple, effective, and since the vertices are moved along its normal direction, thus no vertex-drifting or re-parameterization happens. However, during the implementation, we find that it still contains some drawbacks. First, it uses Euclidian distance to measure the distance between two points on the mesh, which could introduce distortion at the sharp features, since vertices that happen to be geodesicaly far from the smoothed vertex may be geometrically close. Furthermore, the assumption from differential geometry that a neighborhood of a point on a surface can be evaluated by a function over the tangent plane to that point may not be satisfied. Secondly, distortion appears at areas with sharp features. Thirdly, BMD implement the smoothing uniformly over the whole mesh surface, no matter the local shape is flat or steepy. For example, in those areas that are relatively flat, more neighborhood vertices could be selected, and in those steepy areas, less vertices should be selected. Also, as shown in figure 2, when we want to smooth the vertex $\boldsymbol{v}_i$, we should adopt more vertices along the direction $\boldsymbol{n}_1$, less vertices along the direction $\boldsymbol{n}_2$.

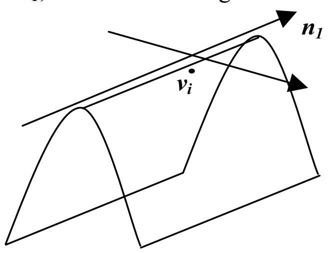

Figure 2. **Anisotropic smoothing**

4. Neighborhood

4.1. Geodesic distance

Geodesic distance between two points on a mesh is the shortest distance along the surface of the mesh. Many mesh processing algorithms require geodesic distances between vertices in a triangular mesh. However, the algorithms to find an exact shortest path on a mesh (including the non-convex case) usually involve high time and space costs. The time complexity is usually $O(n^2)$ [14]or $O(n\log^2 n)$ [15]. Therefore, it is not practical to apply these algorithms to a dense triangle mesh in most cases. Approximate geodesic's computing is much faster and if the error is under control, it is a good substitute.

Novotni [16] proposed an approximation method to compute geodesic distances on triangulated domains in the three dimensional space. Their particular approach is based on the Fast Marching Method for solving the Eikonal equation on triangular meshes. When computing the geodesic distance between two points, the algorithm proceeds by propagating a wavefront outwards from the start points. The advancing front can be thought of as a brush-fire advancing with constant velocity in all directions in which the mesh has not yet been ”burnt”. This is accomplished in a fashion very similar to the well known Dijkstra algorithm [17] for computation of shortest paths in a graph. The geodesic distances of vertices are calculated propagatedly until the target vertex is met or all vertices of the mesh have their own geodesic distances. When computing the geodesic distance of a vertex $\boldsymbol{v}_i$ from the start point, if the other two vertices of the triangle in which $\boldsymbol{v}_i$ lies both have effective geodesic distances, the geodesic distance of $\boldsymbol{v}_i$ is computed using virtual start point (as shown in figure 3), which is implemented as follows: Given a triangle $\boldsymbol{v}_i\boldsymbol{v}_{i+1}\boldsymbol{v}_{i+2}$, if the geodesic distances of $\boldsymbol{v}_{i+1}$ and $\boldsymbol{v}_{i+2}$ are already calculated, say d_{i+1} and d_{i+2} respectively, we could compose two circles using $\boldsymbol{v}_{i+1}$ and $\boldsymbol{v}_{i+2}$ as center, d_{i+1} and d_{i+2} as radii respectively. Denoting the intersect point far off from the $\boldsymbol{v}_i$ as $\boldsymbol{v}_s$, the geodesic distance of $\boldsymbol{v}_i$ is the distance between $\boldsymbol{v}_i$ and $\boldsymbol{v}_s$, which is $\|\boldsymbol{v}_i\boldsymbol{v}_s\|$.

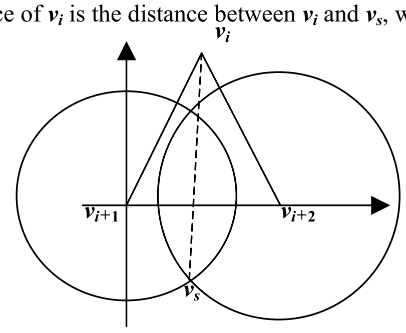

Figure 3. **Virtual start point**

Novotni's method is fast and effective, but sometimes, the error is too big to accept. We find that the algorithm ignored two other cases when calculating the distance from virtual start point. As shown in figure 4, Novotni's method only considered the case of figure 3, while missing the two cases of figure 4a and figure 4b. Therefore, we improve the method as follows:

- When the intersect falls in between and $\boldsymbol{v}_{i+2}$, the geodesic distance is $\|\boldsymbol{v}_i\boldsymbol{v}_s\|$
- When the intersect falls left to $\boldsymbol{v}_{i+1}$, the geodesic distance is $d_{i+1} + \|\boldsymbol{v}_i\boldsymbol{v}_{i+1}\|$
- When the intersect falls right to $\boldsymbol{v}_{i+2}$, the geodesic distance is $d_{i+2} + \|\boldsymbol{v}_i\boldsymbol{v}_{i+2}\|$.

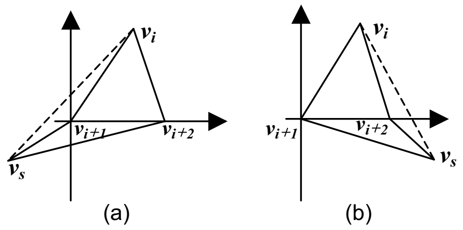

Figure 4. **Virtual start point**

Figure 5 to 7 are some experiment results of our improved algorithm to calculate the geodesic distance on meshes. Figure 5 is a dense planar rectangle mesh, which has 10000 vertices and the edge length of the rectangle is 1. We calculated all geodesic distances from the left bottom vertex to others. Since the geodesic distance on a plane is just its Euclidean distance, we could easily find out the error of the algorithm. The result shows that the maximum error is 10^{-15}, the root mean square error is 10^{-31}, which is quite precise. Figure 5a is the iso-distance line from the left bottom vertex and figure 5b is the colorful result.

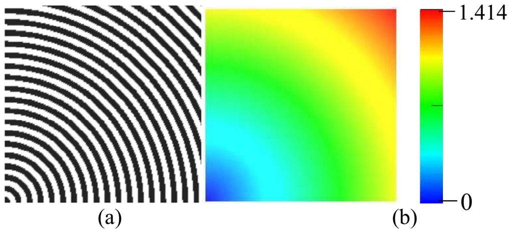

Figure 5. **Geodesic distance on a planar mesh**

Figure 6 shows another more complicated result of our algorithm, in which we calculated all geodesic distance from the top vertex on the nose of the horse to all other vertices. Again figure 6a shows the iso-distance line and figure 6b shows the colorful result.

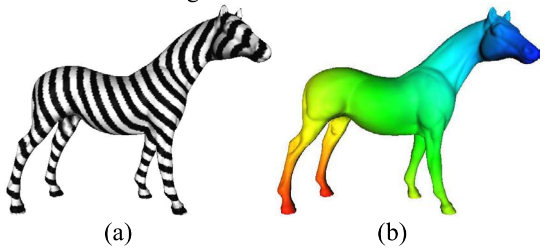

Figure 6. **Geodesic distance on a horse**

4.2. Neighborhood decision

Based on the algorithm of calculating geodesic distance on meshes, we improve the BMD algorithm as follows: when selecting the neighborhood of $\boldsymbol{v}_i$, we no longer select all vertices the Euclidean distance of which to $\boldsymbol{v}_i$ is greater than ρ, but select all vertices the geodesic distance of which to v_i is greater than ρ.

$$N(v_i) = \{\boldsymbol{v}_j | \, d(\boldsymbol{v}_j, \boldsymbol{v}_i) < \rho \}$$

Where $d(\boldsymbol{v}_j, \boldsymbol{v}_i)$ is the function to calculate the geodesic distance between two vertices on a mesh.

5. Offsets computing

From the section 3, we know that the BMD algorithm computes an offset for each vertex $\boldsymbol{v}_i$ using its neighborhood information, which is calculated as the weighted summation of the lengths of those dash lines in figure 1. However, we find that when calculating the offsets of those vertices on sharp features, the resultant offsets are usually over big, which leads to a recess. In order to avoid this flaw, we adjust the projecting direction. Instead of project the vector $\boldsymbol{v}_j$-$\boldsymbol{v}_i$ to the normal vector of $\boldsymbol{v}_i$, i.e. $\boldsymbol{n}_i$, we project $\boldsymbol{v}_j$-$\boldsymbol{v}_i$ to the normal vector of $\boldsymbol{v}_j$, which is explained graphically in figure 7. The adjusted offset is the weighted summation of those thickened lines, while BMD uses those dash lines. Once again, t is the tangent plane at $\boldsymbol{v}_i$. From the figure, we could see that the offset computed by our method is much smaller than that computed by BMD method, which is much more reasonable and verified by the experimental results.

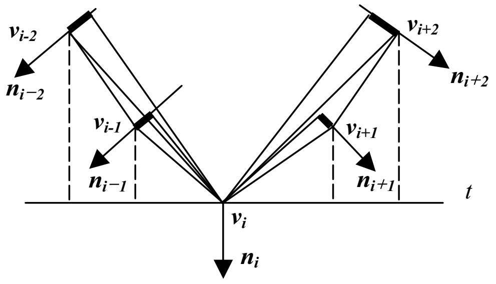

Figure 7. **Offsets computing**

6. Normal weight function

When smoothing a vertex $\boldsymbol{v}_i$, most algorithms deal with it isotropiclly, which means that the calculation of offset is carried out uniformly in all directions. However, this kind of processing ignores the fact that the shape varies in different directions, therefore the contributions to the offset should also very with the direction. For example, as shown in figure 2, when we smooth the vertex $\boldsymbol{v}_i$, we should adopt more vertices along the direction $\boldsymbol{n}_1$, less vertices along the direction $\boldsymbol{n}_2$. In such way, we smooth the vertex $\boldsymbol{v}_i$ anisotropcally.

Another problem is that most algorithms carry out the smoothing uniformly over the whole mesh, which means the same method to calculate offset and select neighborhood vertices. But the geometrical shape is various over a mesh. The ideally way is to select more neighborhood vertices in those areas that are relatively flat (figure 8a) and less neighborhood vertices in those steepy areas (figure 8b). To implement this target, the smoothing algorithm should adjust the cut-off distance ρ adaptively according to the local shape. But this is too hard to realize. Therefore we find a substitute by adding one more weight function to the equation to calculating offset. The new weight function, W_n, is also a Gaussian filter, which punishes the angle between the normal of $\boldsymbol{v_i}$ and that of its neighborhood vertex $\boldsymbol{v_j}$. Thus modified offset calculating method is as follows:

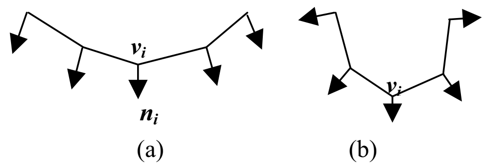

Figure 8. **Different shape of a mesh**

$$d_i = \frac{\sum_{v_j \in N(v_i)} n_j \cdot (v_i - v_j) W_c(\| v_i - v_j \|) W_s(n_j \cdot (v_i - v_j)) W_n(1 - n_i * n_j)}{\sum_{v_j \in N(v_i)} W_c(\| v_i - v_j \|) W_s(n_j \cdot (v_i - v_j)) W_n(1 - n_i * n_j)}$$

Where W_c and W_s are defined in section 3 and

$$W_n = e^{-x^2/2\sigma_n^2}$$

7. Results

We have implemented the our smoothing algorithm as described in the previous sections and compared our results to the results of Laplacian smoothing[2], the signal processing of Taubin [3], improved Laplacian smoothing [4], Jones et al. [13], and the BMD algorithm [1].

The parameters of the algorithm are: σ_c, σ_s, σ_n and ρ. ρ is the cut-off distance of selecting the neighborhood vertices, and usually is set to be $2\sigma_c$. σ_c is the standard deviation of the closeness smoothing filter W_c. Greater σ_c means more neighborhood vertices and greater contribution of vertices to the offset at same geodesic distance. Another shortcoming of too great σ_c value is the computing time. Usually we set σ_c = 2-6 average edge length of the mesh. σ_s is the standard deviation of similarity weight function W_s. Greater σ_s leads to greater offset under the same circumstance. Usually we set σ_s = 0.5-2 average edge length. σ_n is the standard deviation of normal weight function W_n. when the angle between two normals is the same, the smaller the σ_n is, the smaller the contribution to the offset is. Since the parameter is between 0 and 1, we set σ_s = 0.1-0.3.

The normals are first-order properties of the mesh, and they are more sensitive to noise than vertex positions. Fleishman [1] computes the normals using 2 or 3-ring of a vertex. Jones [13] improves the stability by first performing a normal filtering. Since our method has a normal weight function, it is less sensitive to the noise, and thus only the 1-ring face are used to compute the normal and no other processing is needed.

Figure 9 is the graphical results of the comparing of our method with other algorithms. The parameters used are listed in table 1. From the figure, we could see that: Laplacian smoothing preserves sharp features the worst. Improved Laplacian and Taubin's smoothing have improved feature preserving ability. Fleishman's smoothing generates concaves at sharp edges. Jones's approach preserves features much better, but it has the problem of vertex drifting, which is apparent nearby sharp edges. Our approach preserves sharp features the best and since our approach move the vertex along its normal direction, no vertex drifting happens.

Table 1. **Smoothing parameters**

algorithm	parameters
Laplacian	Iterations: 20;
Improved Laplacian	Iterations: 20; α: 0; β: 0.5;
Taubin	Iterations: 20; λ: 0.6307; μ: -0.6732;
Jones	Iterations: 1: $\sigma_f/\|e\|$: 2; $\sigma_g/\|e\|$: 1;
Fleishman	Iterations: 1; $\sigma_c/\|e\|$: 2; $\sigma_s/\|e\|$: 1;
Ours	Iterations: 1; $\sigma_c/\|e\|$: 2; $\sigma_s/\|e\|$: 1; $\sigma_f/\|e\|$: 0.3;

*$\|e\|$=average edge length of the mesh

8. Conclusions

We present a mesh smoothing algorithm based on bilateral mesh denoising. The algorithm modifies a vertices in its normal direction. Geodesic distance, instead of Euclidian distance, is adopted for the selection of neighboring vertices. To compute offsets, a new projection direction is used and a normal weight is introduced. Compared with previous smoothing methods, our method has some advantages: first it doesn't need to repeat the smoothing operations; second it could preserve the features of original model with relatively faster speed; and thirdly it has no vertex drifting problem.

When dealing with irregular mesh, especially when the edge length varies too much, our method seems to be ineffective. This is caused by the fixed neighborhood selection policy and should be improved in the future. Another future development is the smoothing with texture and color information, not just geometrical information.

9. References

[1] Fleishman S, Drori I, Cohen-Or D. Bilateral Mesh Denoising. Proceedings of ACM SIGGRAPH 2003 , 950 – 953.

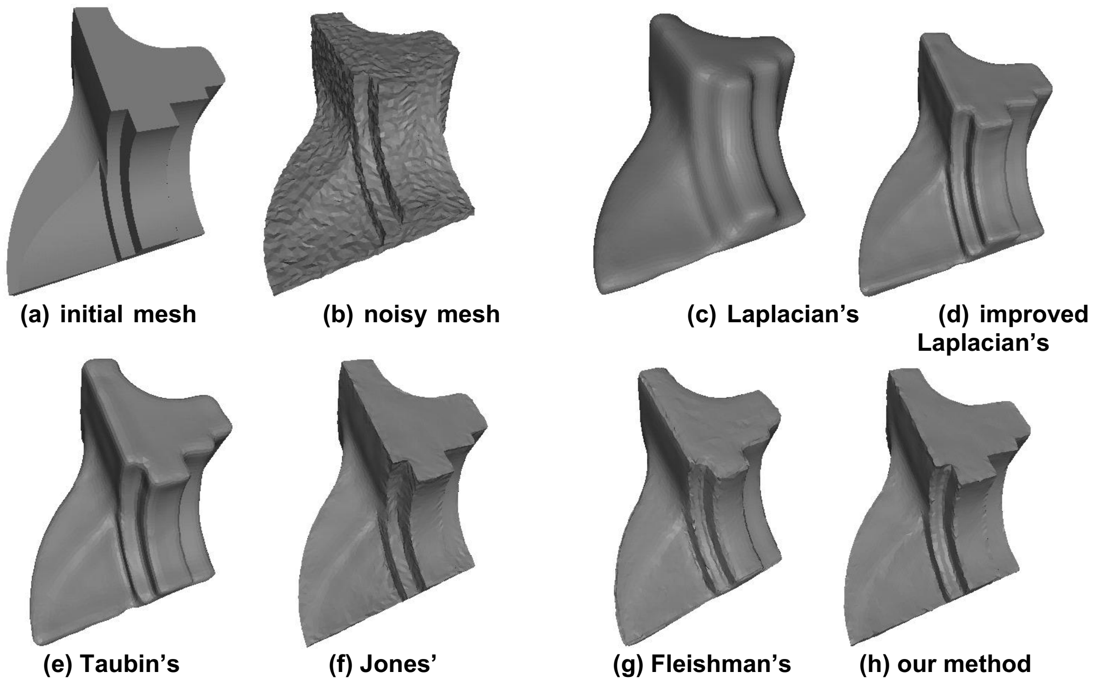

Figure 9**. Comparison of smoothing methods**

[2] Field D. Laplaceian smoothing and Delaunay triangulations. Communications in Numerical Methods in Engineering, 1988, 4: 709-712.

[3] Taubin G. A signal processing approach to fair surface design. In：Computer Graphics Proceedings，Annual Conference Series, ACM SIGGRAPH，Los Angeles，California, 1995, 351-358.

[4] Vollmer J, Mencl R, Muller H. Improved Laplacian smoothing of noisy surface meshes. In: EUROGRAPHICS 99 Conference Proceedings, 1999, pp. 131-138.

[5] M Desbrun，M Meyer，P Schröder，*et al*. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In：Computer Graphics Proceedings，Annual Conference Series, ACM SIGGRAPH，Los Angeles，California, 1999, 317-324

[6] Zhang H, Fiume E. Mesh smoothing with shape or feature preservation. In: Proceedings of Computer Graphics International 2002 *(June 2002),* published as *Advances in Modeling, Animation, and Rendering*, J. Vince and R. Earnshaw, editors, Springer-Verlag, June 2002, 167-182.

[7] Yagou H, Ohtake Y, Belyaev A. Mesh Smoothing via Mean and Median Filtering Applied to Face Normals. Geometric Modeling and Processing, RIKEN, Saitama, Japan, 10-12 July 2002, pp. 124-131.

[8] Desbrun M, Meyer M, Schroder P, Barr AH. Anisortopic feature-preserving denoising of height fields and bivariate data. In: Graphics Interface 2000, May, 2000. 145-152.

[9] Peng J，Strela V，Zorin D. A simple algorithm for surface denoising. In: Proceedings of the conference on Visualization '01, San Diego, California, October, 2001,107 – 112.

[10] Clarenz U, Diewald U, Rumpf M. Nonlinear Anisotropic Geometric Diffusion in Surface Processing. In: Proc. IEEE Visualization 2000, pages 397-405, 2000.

[11] Bajaj C L, Xu Guoliang. Adaptive Fairing of Surface Meshes by Geometric Diffusion. In: Fifth International Conference on Information Visualisation (IV'01), London, England 2001. 731-737.

[12] Tasdizen T, Whitaker R, Burchard P, Osher S. Geometric Surface Smoothing via Anisotropic Diffusion of Normals. In Proceedings of IEEE Visualization 2002, Boston, MA. 125–132.

[13] Jones T R, Durand F, Desbrun M. Non-Iterative, Feature-Preserving Mesh Smoothing. In: SIGGRAPH 2003. 943-949.

[14] Chen J, Han Y. Shortest paths on a polyhedron; part i: computing shortest paths. Int. J. Comput. Geom. & Appl. 1996, 6(2): 127–144.

[15] Kapoor S. Efficient computation of geodesic shortest paths. In Proc. 32nd Annu. ACM Sympos. Theory Comput. 1999, 770–779.

[16] M. Novotni and R. Klein. Computing geodesic distances on triangular meshes. In The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), pages 341--347, 2002.

[17] Dijkstra E W. A note on two problems in connection with graphics. Numerische Mathematik 1, 1959, 1: 269–271.