International Journal of Multimedia and Ubiquitous Engineering
Vol. 9, No. 11 (2014), pp. 11-20
http://dx.doi.org/10.14257/ijmue.2014.9.11.02

Feature-preserving, Adaptive and Anisotropic Smoothing Algorithm for Triangular Mesh Models

Liu Xu-min[1], Yang Li-xin[2] and Li Cai-ling[1]

[1]*College of Information Engineering, Capital Normal University, Beijing 100048, China*

[2]*College of Computer, Beijing University of Post&Telecommunication, Beijing 100876, China*

liuxumin@126.com

Abstract

Despite the great success in smoothing of triangular mesh, the classical methods mostly require human interactions. In order to reduce the parameter settings, an adaptive and anisotropic smoothing algorithm for removing noise and preserving features of triangular mesh model was proposed. First calculate the expected normal vector of the model triangles, the expected normal vector should be an accurate representation of the desired movement direction and be similar to the original surface normal vector, and then we compute the value of the offset value. We develop an adaptive coefficient scheme which can avoid the parameter settings to obtain the coefficient value for each vertex. Finally, we updated every vertex's position of the model by formula. Experimental results show that the algorithm can be adaptive to preserve sharp features and avoid shrinkages by comparing with classical methods.

Keywords: *smoothing; anisotropic; adaptive; triangular mesh; feature-preserving*

1. Introduction

With the development of the theory and technology of computer-aided geometric design, reverse engineering has become an important approach to represent three-dimensional model. The mock-ups are usually obtained by 3D scanner, the point cloud are collected from the three-dimensional data measurement tools, while on account of human factors and other reasons, often lead to data noise , data redundancy and other issues. The noises should be removed before 3D data processing. Mesh smoothing is to eliminate the noise, to maintain its topology and geometry, to avoid oversmoothing and volume shrinkage.

Existing triangular mesh smoothing algorithms have two main categories: local iterative method and global optimization method. Field [1] proposed a classical Laplace algorithm that mesh vertices move towards its neighboring's center direction. The algorithm is simple and fast, yet the model is prone to be shrinking and oversmoothing with the increasing number of iterations. Taubin [2] applied the knowledge of signal processing to remove noise, and presented a weighted Laplacian algorithm with two alternative scale factors of opposite signs. Such smoothing can suppress high frequency and enhance low frequency slightly without volume shrinkage. Mingqiang Wei *et al.*, [3] present a feature-preserving optimization for noisy mesh algorithm by using joint bilateral filter and constrained Laplacian smoothing, yet a large computation is needed when the mesh is very large. Desbrun *et al.*, [4] proposed the mean curvature flow algorithm, each mesh vertex moves towards its normal vector direction with the speed of mean curvature. While in some special cases, the grid is easy to cause distortion. A mesh regularization and adaptive smoothing was proposed by Ohtake *et al.*, [5], it used a coupled nonlinear diffusion of the mesh normal and vertices. The algorithm increased regularity of the mesh, reduced over-smoothing, and enhanced crease lines. Its drawback was to manually set some interaction parameters. Zhanheng Gao *et al.*, [6], proposed a quality tetrahedral mesh smoothing via boundary-optimized Delaunay triangulation, the method can be readily adapted to preserve sharp features.

Mean filter and median filter method of image processing are applied to the field of mesh smoothing by Yagou [7], while mean filter is not able to maintain the original sharp features, median filter may increase irregular grid. Tang [8] improves bilateral filtering to a fast smoothing algorithm based on CUDA, but it has a high computation. The most common energy function is global optimization method; this method first presented an energy function which can update the mesh vertex position by solving the function. It's computationally intensive, hasn't been widely used. In 2010, Zhang *et al.*, presents a smoothing algorithm by solving the discrete linear quadratic energy function [9], low complexity without shrinkage but a large number of calculations.

In order to reduce the amount of human interaction and computation, effectively remove noise and keep the sharp features, this paper presents an adaptive anisotropic smoothing algorithm. First calculate expected normal vector of surface according to its 1-ring surface normal vector using surface area as weights. Then calculate the offset of each vertex according to the expected normal vector. Finally, we get umbrella operator [10] to compute the offset coefficient, then update the vertex's new position. This algorithm is more adaptive and efficient.

2. Adaptive and Anisotropic Smoothing Algorithm

2.1. Expect Normal Vector Optimization

Graph Φ is a triangular mesh as Figure 1 shown, the target triangle R's adjacent triangles are distinguished into two categories: the former which shares the same edge with R, as the set of $S_i(i = 1,2,3)$ shown in Fig. 1, which called 1-ring neighborhood; the later which shares the same point with R, as the set of $Q_j(j = 1,2,3 \dots n)$ shown in Figure 1, which called 2-ring neighborhood.

Considering to accurately calculate the expected normal vector, the desired weights determined by the area of the triangle adjacent triangles. The area is larger, the impact on the target surface is greater.

The excepted normal vector is obtained by Eq.1:

$$\mathrm{m(R)} = \frac{1}{\sum A(S)} \sum_{s \in N(R)} A(S) n(S) \quad (1)$$

Where $A(S)$ is the area of triangle S, $N(R)$ is the set of neighborhood of triangle R. In order to make the desired normal represents the real moving direction, we analysis the impact of 1-ring and 2-ring adjacent triangles' normal vector on the expected normal vector calculation. As Figure 2 shown, two intersecting planes Π_1 and Π_2, assuming $\Phi - \{ABCD\}$ is on the Π_1 plane which contains the target triangle R, $ABCD$ is on the plane Π_2. $ABCD$ are the points connected with the target triangle R. If considering both point and edge connected surface, the expected normal vector will deviate from the plane Π_1 while the surface R is in a relatively flat area in real situation. According to the experiment result, the method using 1-ring can keep sharp features and avoid uncertain mesh structural aberrations.

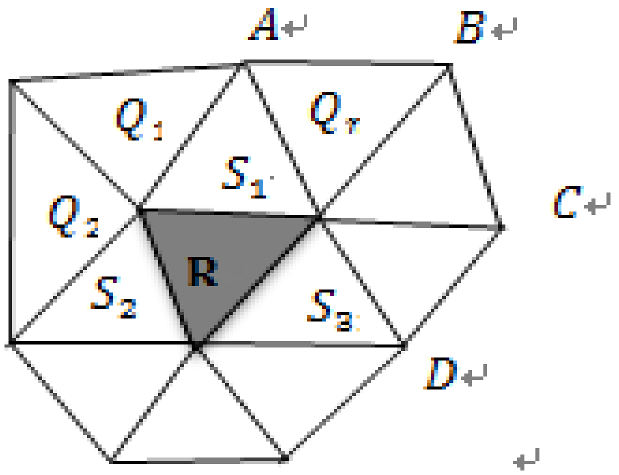


Figure 1. Target Triangle R's Neighborhood

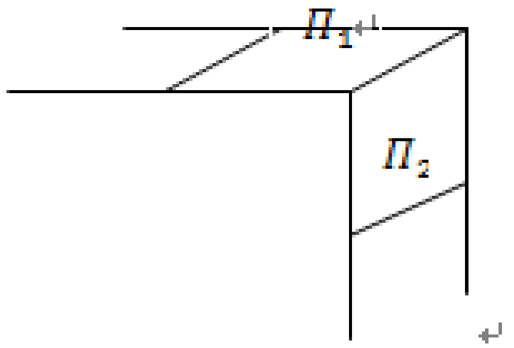


Figure 2. Two Intersecting Planes Π_1 and Π_2

2.2. Vertex Offset Optimization

Literature [5] provides a function to obtain offset value, assuming $V(R)$ is the offset value of vertex P, $V(R)$ is defined by Eq.2:

$$V(R) = \lfloor \overrightarrow{PC} \cdot m(R) \rfloor m(R) \tag{2}$$

Where R is the triangle containing vertex P, C is the triangle-center of R, $m(R)$ is the expected normal vector of R. As Figure 3 shows, $m(R)$ is close to the original normal vector $n(R)$ if the area is flatter, and so $\overrightarrow{PC} \cdot m(R) \to 0$, $V(R)$ will be smaller, the vertex moves relatively small which can obtain the sharp features. If $\overrightarrow{PC} \cdot m(R)$ is larger, it means the vertex is in an uneven region and it requires more adjustments to achieve the purpose of smoothing.

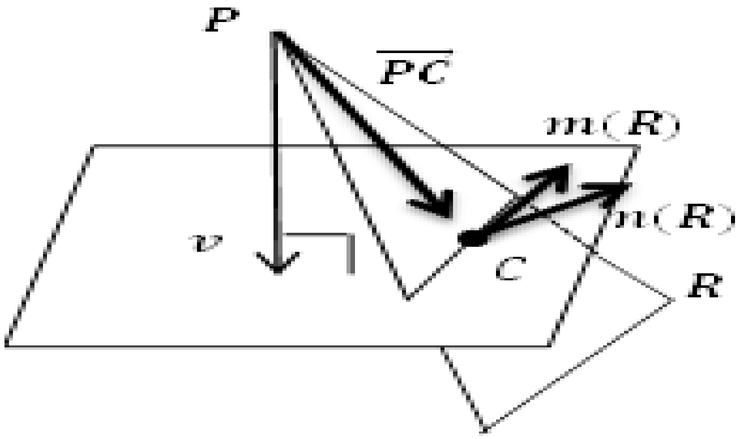


Figure 3. Vertex Offset Direction

2.3. The Offset Coefficient Optimization

This algorithm proposes a conception called offset coefficient to reduce human interaction, remove noise quickly and improve the adaptability. The offset coefficient is a value which changes based on the features of v_i regions. Supposed the offset coefficient is d_i, d_i should have the following properties:

1） When v_i is in a flat area, $d_i \to 0$, which preserves the original features.

2） When v_i is in an uneven area, d_i increases with the frequency of the noise.

In the experiments, offset coefficient function $g(x)$' s argument is $U(i)^2$ which is calculated by Eq.(4), $U(i)$ is the umbrella operator which is computed by Eq. (3):

$$U(i) = \frac{1}{\sum_{j \in NV(i)} w_{ij}} \sum_{j \in NV(i)} w_{ij} v_j - v_i \qquad (3)$$

$$U(i)^2 = (x_i^2 + y_i^2 + z_i^2) \qquad (4)$$

$NV(i)$ is the set of v_i's adjacent vertexes, which are directly connected with v_i. w_i is the weight of v_i, 1) Let $w_i = 1$, all the vertexes have the same weight. 2) Using reciprocal of the distance e_{ij} of vertex v_i and v_j, evidently a close vertex has a greater weight. In our experiment, we choose the former one for simple computation.

As Figure 4 shown, when $U(i) < 0$, v_i move downward, the high frequency information is suppressed; as Figure 5 shown, when $U(i) > 0$, v_i move upward, the high frequency information is suppressed. $U(i)^2$ is a good representation of the size of the noise.

For the function $g(x)$, this paper attempts the following functions:

$$g(x) = x \qquad (5)$$

$$g(x) = \exp(x) \qquad (6)$$

$$g(x) = 1 - e^{-(x-u)^2/\sigma^2}, \text{ where } \mu = 0, \sigma \text{ is a small positive variable} \qquad (7)$$

The above function are all monotonically increasing function in the definition of domain-wide $(x \geq 0)$ which meet the requirements of the offset coefficient d_i. Experiments show that: Eq.5 can keep the sharp features but needs multiple iterations. Eq. 7 requires multiple experimental σ' value. Eq. 6 can quickly and efficiently remove the noise and preserve the features.

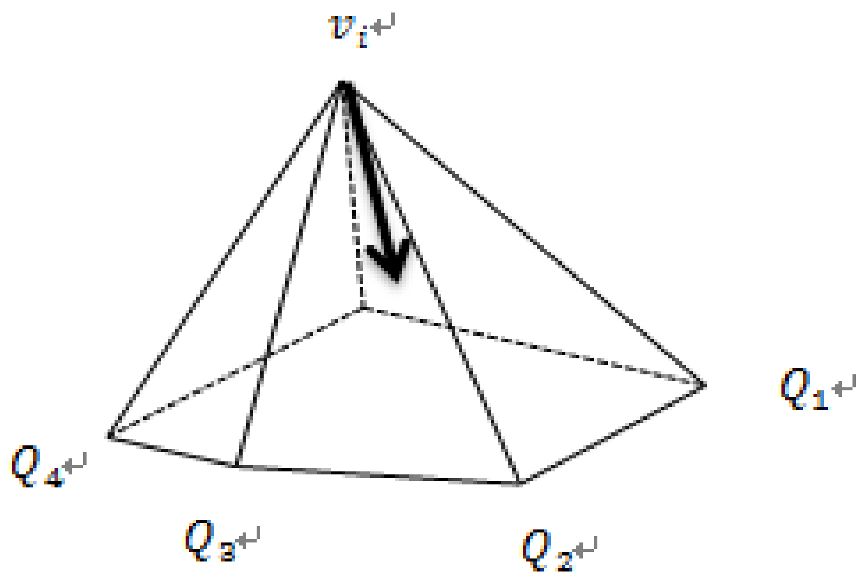


Figure 4. Umbrella Operator $U(i) < 0$

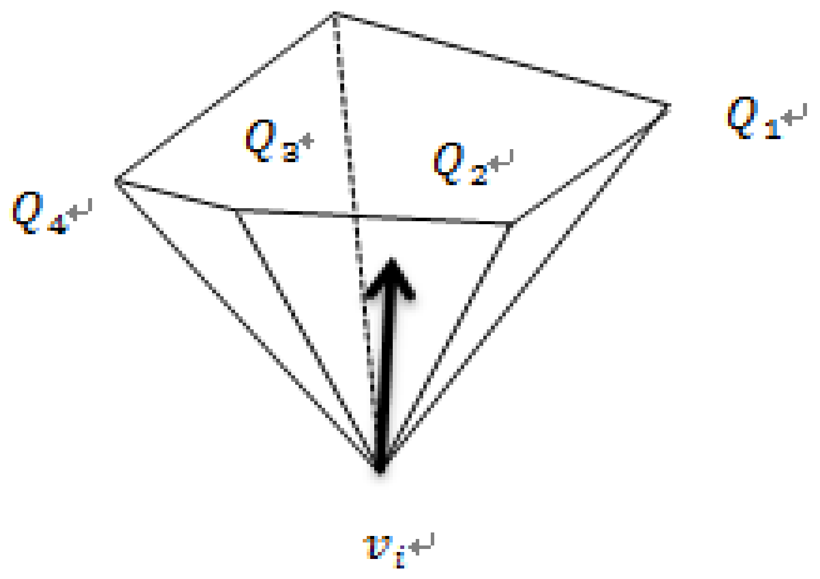


Figure 5. Umbrella Operator $\mathbf{U(i)} > 0$

a) Update Vertex Position

We get updated vertex position by the Eq. (8) after calculate the offset and offset coefficient:

$$v_i' = v_i + d_i V(R) \qquad (8)$$

Where v_i represents the coordinate before iterations and v_i' represents the coordinate after updating.

b) Fairing Error Analysis

In order to compare the results between the classical algorithms and this algorithm, using literature [5] proposed vertex error method. Assuming M is the original mesh, M' is the mesh model after smoothing. We calculate the error by Eq.9:

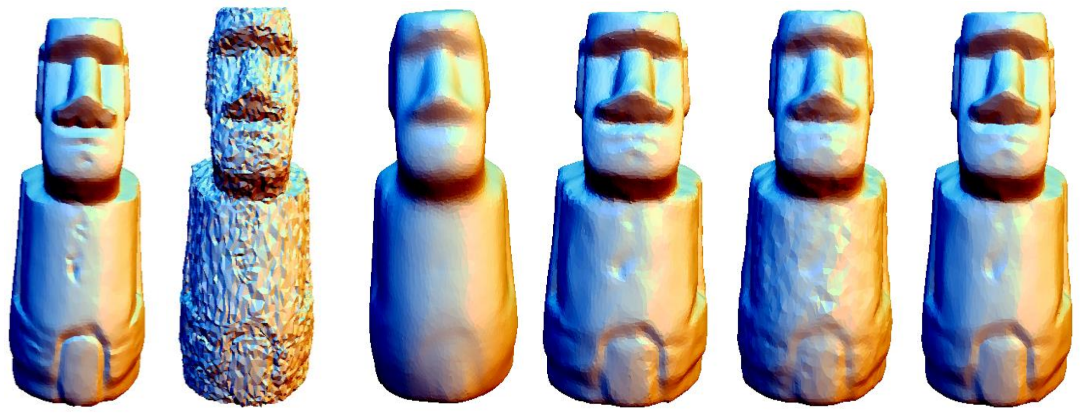
$$E_v = \sqrt{\frac{1}{3A(M')}\sum_{P\in M'} A(P')dist(P',M)^2} \qquad (9)$$

P' is one vertex in mesh M', $dist(P',M)$ is the distance between vertex P' and the triangle which is nearest vertex P' in mesh M. $A(P')$ is the total area of triangles with vertex P' in model M'. $A(M')$ is the area of the model M'. This error can generally reflect the volume change of the model.

3. Experimental Results

The experiment run on a PC which frequency is 2.10GHz, RAM 4.00GB (2.96GB available) and Pentium Dual-Core CPU. Programming platform is Microsoft Visual Studio 2010 with OpenGL graphics library. Comparative results are as follows:

Choosing moai model and add 20% random noise, compare this algorithm with Laplacian [1], Taubin's λ/μ algorithm and mean curvature flow algorithm for smoothing results. As Figure 6 shown, the data comparing results show in Table 1.



(a) original model (b) add 20% noise (c) Laplacian (d) Taubin (e) mean curvature flow (f)this algorithm

Figure 6. Comparing this Algorithm with Classical Algorithms Using Model Moai

Table 1. Moai Data Comparison

algorithm	Vertex num	Triangles num	step	iterations	error	Running time
Laplacian	10002	20000	0.5	20	0.022616	78ms
Taubin λ/μ	10002	20000	0.5	20	0.006112	172ms
mean curvature flow	10002	20000	——	20	0.010658	1060ms
this algorithm	10002	20000	——	20	0.005699	5601ms

Choosing Octa-flower model and add 20% random noise, compare this algorithm with Laplacian [1], Taubin's λ/μ algorithm and mean curvature flow algorithm for smoothing results. As Figure 7 shown, the data comparing results show in Table 2.

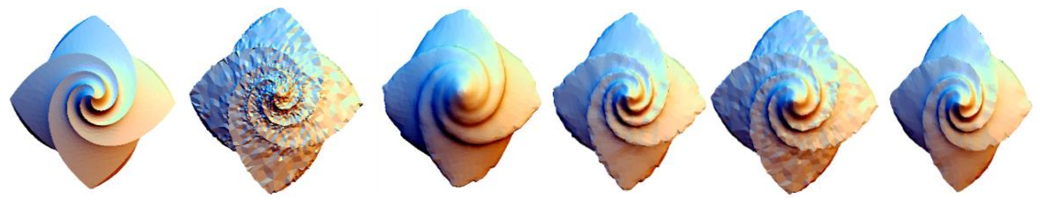

(a) original model (b) add 20% noise (c) Laplacian (d) Taubin (e) mean curvature flow (f) this algorithm

Figure 7. Comparing this Algorithm with Classical Algorithms using Model Octa-flower

Table 2. Octa-flower Data Comparison

algorithm	Vertex num	Triangles num	step	iterations	error	Running time
Laplacian	7919	15834	0.5	20	0.135568	62ms
Taubin λ/μ	7919	15834	0.5	20	0.035798	125ms
mean curvature flow	7919	15834	——	20	0.041597	1045ms
this algorithm	7919	15834	——	20	0.030784	3604ms

Figure 6 and Figure 7 represent the models after adding noise to the original model, (a) is the original model, (b) is the model which adding 20% noise, (c)-(f) are the smoothing results after 20 time iterations. (c) is the result of Laplcian which is oversmoothing. (e) shows the result of mean curvature flow, obviously leads to an irregular grid. (d) and (f) have the similar result, from the Table 1 and 2, (f) smoothing result has a smaller error compare with (d). For a higher computation, the running time of this algorithm is longer.

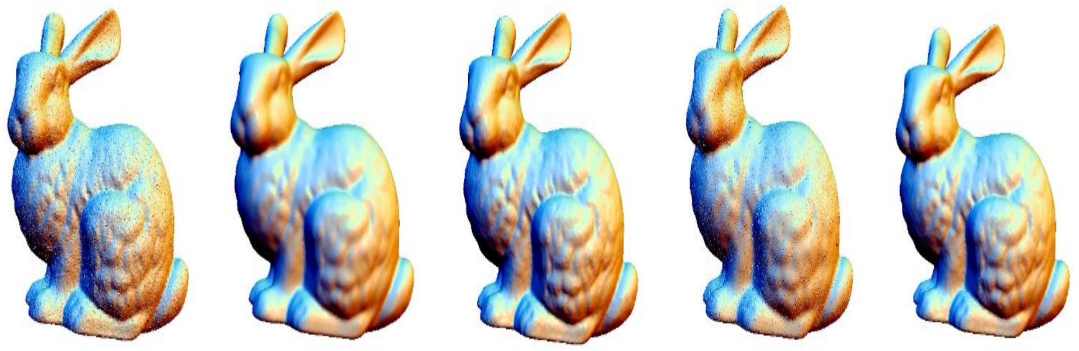

(a) original model (b) add 20% noise (c) Laplacian (d) Taubin (e) mean curvature flow (f)this algorithm

Figure 8. Comparing this Algorithm with Classical Algorithms Using Model Bunny

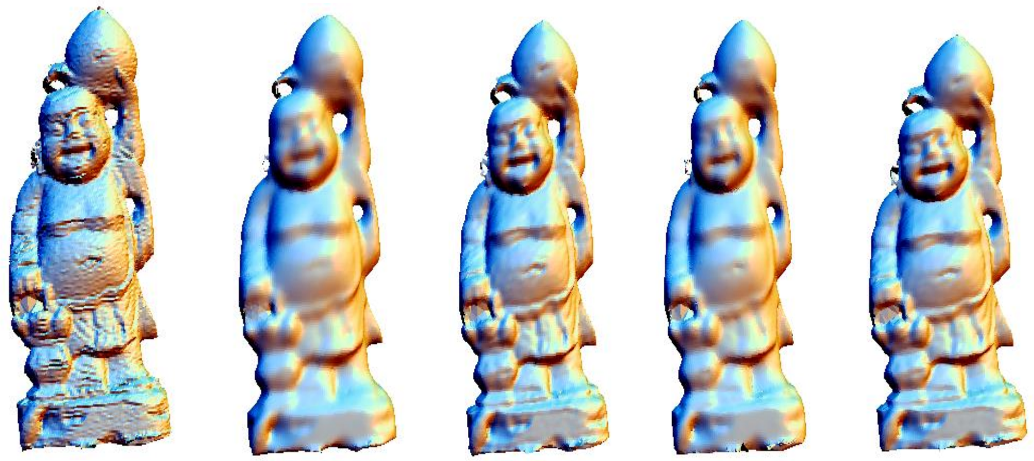

(a) original model (b) add 20% noise (c) Laplacian (d) Taubin (e) mean curvature flow (f)this algorithm

Figure 9. Comparing this Algorithm with Classical Algorithms using Model Monk

Figure 8 and Figure 9 respectively represent the bunny and monk models' smoothing results after 10 times iterations. Figure 8(a) is the bunny noise model, Figure 9 (a) is the monk noise model, 8(b) and 9(b) is the results of Laplacian which step is 0.5, 8(c) and 9(c) is the results of Taubin which step is 0.5, 8(d) and 9(d) is the results of mean curvature flow, 8(e) and 9(e) is the results of this algorithm. The result of Laplacian is oversmoothing, mean curvature flow method is oversmoothing in Figure 8 (d) and have more residual noise. This algorithm has a more efficient result compare with (b) and (d).

4. Conclusion

This paper proposed a feature-preserving, adaptive and anisotropic smoothing algorithm for triangular mesh models. By discussing the impact of adjacent surface normal vector, we calculate the expected triangles normal vector. By developing an adaptive coefficient scheme, we compute the offset coefficient values using the umbrella operator. Finally adjust the position of the vertexes. Experiments show that this algorithm can reduce human interactions, achieve a satisfactory smoothing effect, and preserve sharp features adaptively. By experimental data analysis, the algorithm has a smaller error compared with other algorithms, can effectively keep the model's shape.

Acknowledgments

This research was supported by the National Natural Science Foundation of China General Projects Grant No. 61272029.

References

[1] Field D A, Laplacian smoothing and Delaunay triangulations, Communications in Applied Numerical Methods, vol. 6, no. 4, **(1988)**, pp. 709-712.
[2] G. Taubin, "A signal processing approach to fair surface design", Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, **(1995)**, pp. 351-358.
[3] M. Q. Wei, W. Y. Shen, J. Qin, J. H. Wu, T. T. Wong and P. A. Heng, "Feature-preserving optimization for noisy mesh using joint bilateral filter and constrained Laplacian smoothing", Optics and Lasers in Engineering, vol. 11, no. 51, **(2013)**, pp. 1223-1234.

[4] M. Desbrun, M. Meyer，P. Schroder and A. H. Barr, “Implicit fairing of irregular meshes using diffusion and curvature flow”, Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, **(1999)**, pp. 317-324.

[5] Y. Ohtake, A. G. Belyaev and I. A. Bogaevski, “Mesh regularization and adaptive smoothing”, Computer-Aided Design, vol. 4, no. 33, **(2001)**, pp. 789-800.

[6] Z. H. Gao, Z. Y. Yu and M. Holst, “Quality tetrahedral mesh smoothing via boundary-optimized Delaunay triangulation”, Computer Aided Geometric Design, vol. 9, no. 29, **(2012)**, pp. 707-721.

[7] H. Yagou, Y. Ohtake and A. Belyaey, “Mesh smoothing via mean and median filtering applied to face normals”, In: Geometric Modeling and Processing Proceedings, **(2002)**，pp. 124-131.

[8] J. Tang, B. Xu, Z. L. Gong, G. S. Wu, “Fast Fairing of 3D Point Clouds Using CUDA”, Journal of System Simulation, vol. 8, no. 24, **(2012)**, pp.1633-1642.

[9] D. M. Zhang and L. G. Liu, “Feature preserving Mesh Smoothing Algorithm Based on the Weighted Least Squares”, Journal of Computer□Aided Design & Computer Graphics, vol. 9, no. 22, **(2010)**, pp. 1497-1501.

[10] L. Kobbelt, S. Campagna and J. Vorsatz and H. P. Seidel, “Interactive multi-resolution modeling on arbitrary meshes”, Computer Graphics SIGGRAPH Proceedings, **(1998)**, pp. 105-11.