Published in final edited form as:

J Comput Sci Technol. 2012 January 1; 27(1): 163–173. doi:10.1007/s11390-012-1214-3.

Feature-Preserving Mesh Denoising via Anisotropic Surface Fitting

Jun Wang[Member, ACM, IEEE] and Zeyun Yu*[Member, ACM, IEEE]

Department of Computer Science, University of Wisconsin, Milwaukee, WI 53211, U.S.A.

Abstract

We propose in this paper a robust surface mesh denoising method that can effectively remove mesh noise while faithfully preserving sharp features. This method utilizes surface fitting and projection techniques. Sharp features are preserved in the surface fitting algorithm by considering an anisotropic neighborhood of each vertex detected by the normal-weighted distance. In addition, to handle the mesh with a high level of noise, we perform a pre-filtering of surface normals prior to the neighborhood searching. A number of experimental results and comparisons demonstrate the excellent performance of our method in preserving important surface geometries while filtering mesh noise.

Keywords

mesh denoising; feature-preserving; surface fitting; anisotropic filtering

1 Introduction

Surface meshes are widely used in geometric modeling, computer graphics and many other application fields such as engineering design and bio-medicines. These meshes may be obtained from a variety of sources, including computer-aided designing, digital scanning, 3D imaging data, and so on. However, the acquired real-life data inevitably introduce noise that lead to noisy or bumpy surface meshes. Such noises often impose problematic effects on subsequent applications, such as model reconstruction, visualization and numerical simulation. It is thus critical to filter the noise prior to further processes. While the noise removal may be conducted at different stages of the modeling pipeline, it is our interest in the current paper to filter the noise after the surface mesh is constructed but before the subsequent processes are carried out. This is commonly known as surface mesh denoising.

Surface meshes of many real models contain sharp features such as edges and corners with relatively high curvatures. The ultimate goal of surface mesh denoising is to remove noise while preserving the sharp features in a mesh.

1.1 Related Work

A wide variety of mesh denoising algorithms have been proposed^[1-9]. Laplacian iterative smoothing is one of the fastest and simplest techniques for mesh denoising^[1]. However, this method often leads to feature blurring and significant volume shrinkage. Taubin^[2] proposed a mesh smoothing method by using an isotropic scheme to improve the smoothness of a

surface mesh, while alleviating the shrinkage. Desbrun *et al.*^[3] extended Taubin's work to smooth irregular mesh by using geometric flows and re-scaling the mesh to preserve its volume. However, features are often blurred or filtered out in both methods.

Recently, anisotropic diffusion^[10] and bilateral filtering^[11-12] techniques had been used to remove mesh noise. Jones *et al.* developed a feature-preserving smoothing algorithm by adopting local first-order predictors statistically defined on triangulated surface meshes^[13]. Fleishman *et al.* introduced a similar method based on iterative bilateral filtering, a nonlinear variation of Gaussian smoothing that weighs sample points based on their similarity to the one being processed^[14]. These two methods can preserve features to certain degree, but fail to retain sharp features. Also, the volume shrinkage problem is not addressed in both methods.

Yagou *et al.*^[15] took advantage of mean and median filtering techniques to carry out mesh denoising by smoothing mesh normals. Ohtake *et al.*^[16] proposed another method by minimizing an error function to update vertex positions, and they also designed the diffusion-type smoothing method on the normal field^[17]. Similarly, Sun *et al.*^[18] presented a fast feature-preserving mesh denoising approach by normal-filtering and vertex-updating, in addition, they took advantage of the idea of random walks to filter mesh noise^[19]. Hilderbrandt and Polthier^[20] adopted mean curvature flows to remove noise with features and volumes well preserved. All these methods are able to preserve sharp features when there is not much noise present in a mesh. When there is a high level of noise, however, the sharp features are frequently corrupted.

Moving least squares (MLS)^[21] is a popular method for functional approximation of irregular data. Levin^[22] had pioneered extending MLS to the reconstruction of manifolds, where the surface is defined as the set of stationary points of an iterative projection operator^[23]. In recent years, the MLS technique had been used for the smoothing and reconstruction of point-based surface model. Fleishman *et al.*^[24] proposed an MLS-based method for reconstructing a piecewise smooth surface from sampled points, where they fit quadratic polynomials locally to points and used standard techniques from robust statistics in the fitting process. As a result, sharp features can be preserved to some extent. However, this method is not capable of handling high-level noise, as it relies on an initially found local region with low-level noise to obtain a reliable estimation of the quadratic fit. Furthermore, the method requires very dense sampling and special and complex processes to locally combine different patches, making the approach relatively expensive. By performing normal-based filtering, our method has the ability to cope with a relatively high level of noise, and it is independent of the density of the input mesh.

1.2 Overview of Our Approach

In this paper, we propose a novel, feature-preserving mesh denoising approach by utilizing a combination of the surface fitting^[25-27] and anisotropic neighborhood-searching techniques. According to the surface theory^[28], the local geometry around a surface point can be regarded as a height field over its tangent plane. To faithfully characterize the local geometric properties in this field, surface patches with at least second-order are required. Therefore, we choose quadratic surfaces to approximate the local geometry around a surface point. For each vertex, its neighbors are searched and fitted with a quadratic surface, and then the vertex is projected onto the fitted surface. The vertex is updated with the new position to achieve the mesh denoising. The procedure above applies to all vertices of the mesh and repeats iteratively until the Hausdorff distance between the two smoothed meshes at the i -th and $(i+1)$ -th iterations is less than a pre-defined threshold. The non-sharp (or low curvature) features can be preserved very well using the surface fitting and projection scheme. To preserve sharp features, the anisotropic neighborhood technique is exploited

using the normal-weighted distance. When there is a high level of noise in a surface mesh, the normal filtering is performed prior to the neighborhood-searching. This combined strategy can preserve both sharp (high curvature) and non-sharp (low curvature) features. Fig.1 gives the 2D illustration of the process of mesh denoising.

2 Mesh Denoising via Local Surface Fitting

2.1 Vertex Normal Filtering

The vertex normals of a mesh can be seriously corrupted due to mesh noise. To improve the quality of surface fitting, normal filtering is often needed, especially when there is a very high level of noise in the input mesh. Here, we adopt the bilateral technique^[11] to filter vertex normals.

Consider a triangular mesh $M = \{V, E, F, X\}$, where $V = \{i | i = 0, 1, \dots, n-1\}$ is the vertex set, and E and F are the edge and face sets respectively. The set $X = \{(x_i, y_i, z_i) | i \in V\}$ is the vertex coordinates associated with V . Let $NV_k(i)$ be the k -ring neighboring vertex set of vertex v_i , $NF_k(i)$ the k -ring neighboring faces of vertex v_i . We chose k as 2 in our actual implementation. The normals of vertices in V , and faces in F are denoted by $N = \{n(i) | i \in V\}$, $NF = \{nf(i) | i \in F\}$, respectively. The areas of faces in F are given as $A = \{a(i) | i \in F\}$. Based on the formulation of Tomasi and Manduchi^[11], the updated normal $n'(i)$ of vertex v_i by the bilateral filtering can be represented by:

$$\begin{cases} n'(i) = n(i) + \Delta n(i) \\ \Delta n(i) = \frac{\sum_{j \in NV_1(i)} (n(j) - n(i)) \cdot W_c(\|v_j - v_i\|) \cdot W_s(\|n(j) - n(i)\|)}{\sum_{j \in NV_1(i)} W_c(\|v_j - v_i\|) \cdot W_s(\|n(j) - n(i)\|)} \end{cases}, \quad (1)$$

where the spatial smoothing term is the standard Gaussian filtering with parameter

$\sigma_c: W_c(x) = \exp(-x^2 / (2\sigma_c^2))$, and a feature preserving weight function with parameter σ_s is:

$W_s(x) = \exp(-x^2 / (2\sigma_s^2))$. The initial normal of vertex v_i is given by a weighted average:

$$n(i) = \text{normalize} \left(\frac{\sum_{j \in NF_1(i)} n f(j) \cdot a(j)}{\sum_{j \in NF_1(i)} a(j)} \right). \quad (2)$$

In our implementation, the following parameters give good smoothing results: $\sigma_c = \sigma_s = 0.025 \cdot l_{avg}$ (l_{avg} is the mean edge length) and the number of iterations applied is 15.

2.2 Local Surface Fitting and Projection

After the vertex normals of a mesh are filtered, we then perform local quadratic surface fitting. A quadratic surface S with a parametric representation in a local coordinate system (LCS) is illustrated in Fig.2 and can be expressed as:

$$S(u, v) = (u, v, h(u, v)). \quad (3)$$

The LCS for each vertex v_i on the mesh is constructed in such a way that v_i is mapped to p , the origin of the LCS, and the normal of v_i corresponds to h , the third axis of the LCS. The first two coordinate axes u and v are arbitrarily chosen from the plane determined by v_i and its normal. Then we transform all the k -ring neighboring vertices $NV_k(i)$ of v_i from the global

coordinate system (GCS) to its LCS as $NV'_k(i) = \{v'_j(x'_j, y'_j, z'_j) | 0 \leq j \leq m\}$ by homogeneous transformation and then fit them with an analytical quadratic surface:

$$h(u, v) = au^2 + buv + cv^2 + eu + fv + g. \quad (4)$$

The least square fitting method is adopted. The objective function of the least square quadratic surface fitting can be expressed as:

$$F = \sum_{0 \leq j \leq m} |h(x'_j, y'_j) - z'_j|^2 \cdot w_j, \quad (5)$$

where w_j is the weighting function defined in such a way that the closer the neighboring vertex to v_i , the more it affects the surface shape. Therefore, w_j can be chosen as

$$w_j = e^{-\|v_j - v_i\| / \max(\|v_1 - v_i\|, \dots, \|v_m - v_i\|)}. \quad (6)$$

Minimizing F gives rise to a set of linear equations, which can be solved with the Gaussian elimination method^[29]. The resulting coefficients are utilized to define the quadratic surface $h(u, v)$ in (4).

After a quadratic surface is fitted at a vertex, the new position for the vertex must be chosen somewhere on the surface. We project the vertex along its normal vector onto the quadratic surface and update it with the projection position. This method is very easy to implement and works well in reducing the bumpiness of the surface mesh, while preserving features of low curvatures. Fig.3 gives the denoising result of the Stanford bunny model embossed with the letters “i” and “H”. Note that those small wavy features on the bunny body have been preserved quite well. However, sharp features, such as the boundaries of “i” or the mouth of the bunny, have been blurred.

3 Anisotropic Mesh Denoising

From the smoothing results in Fig.3, we can see that many sharp features, including edges and corners, are smoothed out. Specifically, if a vertex is located near an edge or a corner, its local quadratic surface will blur these features. An ideal way to preserve features at a vertex is to partition surface segments sharing the vertex, fit the connected segments into different surface patches, and finally project the vertex to the intersection curve of the feature patches. However, it is nontrivial to accurately segment feature patches of a mesh and compute the intersection of patches when the surface shape is highly complicated. In our method, we take a more feasible and robust way to preserve the features during mesh denoising. For each vertex to be smoothed, we consider part of its neighboring vertices (so-called anisotropic neighborhood) that will be used to fit the local quadratic surface. The subset of neighboring vertices is chosen so that the fitted surface faithfully mimics the original surface feature and thus the sharp feature is preserved after the mesh denoising.

3.1 Face Normal Filtering

It often helps to perform normal-based mesh smoothing as a pre-processing, especially when there is a high level of noise in a model. There have been many approaches to mesh normal filtering, most of which update the normals directly from the original face normals. The mean, median, and alpha-trimming filters were adopted by Yagou *et al.*^[15]. Shen *et al.*^[5] gave a fuzzy vector median filter to smooth face normals. Sun *et al.*^[18] presented a fast

method to filter face normals using the thresholding technique. In the present paper, we employ Sun *et al.*'s approach^[18] with some modifications. Suppose $NFF(i)$ is the union of a face f_i and the set of faces that have a common vertex or edge with f_i , then the updated normal of f_i is given in Sun *et al.*^[18] as

$$\mathbf{n}f(i) = \text{normalize} \left(\sum_{j \in NFF(i)} h_j \mathbf{n}_j \right), \quad (7)$$

where h_j is a weighted function defined by

$$h_j = \begin{cases} f(\mathbf{n}_i \cdot \mathbf{n}_j - T), & \text{if } \mathbf{n}_i \cdot \mathbf{n}_j > T, \\ 0, & \text{otherwise,} \end{cases} \quad (8)$$

where $0 < T < 1$ is a pre-defined threshold, and $f(x)$ is monotonically increasing function for $x \geq 0$. In [18], $f(x) = x^2$ is adopted, while we choose $f(x) = x^4$, which produces more preferable results in our experiments.

3.2 Anisotropic Neighborhood Selection

In the standard surface fitting method described in Section 2, we consider an isotropic neighborhood around each vertex. As shown in Fig.3, the isotropic neighborhood can cause feature-blurring problems. In general, sharp features can be considered as the intersections of multiple local surface patches. Accordingly, the points on a sharp feature are located on these joint surface patches. In other words, the vertices in a neighborhood of a surface point P should contribute differently to the quadratic fitting function according to their “similarity” to the center point P . Similar to the bilateral filtering as we briefly explained in Section 2, we will consider two types of “similarities”: one is spatial similarity (or distance to the center point) and the other is property similarity (or the angle between the normal vectors of two triangles). To combine these two terms, we adopt the normal-weighted distance between two triangles in a mesh, as introduced by Ohtake *et al.*^[16]. The normal-weighted distance is defined for each surrounding face around a seed face. As illustrated in Fig.4, let f_i be a seed face and f_j, f_k are two adjacent faces within the neighborhood of f_i . $\mathbf{n}(i)$, $\mathbf{n}(j)$ and $\mathbf{n}(k)$ are the unit normals of f_i, f_j and f_k , respectively. c_j, c_k are the centroids of f_j, f_k , and mp_{jk} is the middle point of the common edge of f_j and f_k . The distance between two adjacent faces, f_j and f_k , is defined as $\Delta(f_j, f_k) = \|c_j - mp_{jk}\| + \|c_k - mp_{jk}\|$. Then, the normal-weighted distance from the face f_k to the seed face f_i is calculated as

$$DFF_i(k) = DFF_i(j) + \Delta(f_j, f_k) \cdot [1 + \omega \cdot (A + B + C)], \quad (9)$$

where $A = \frac{\|\mathbf{n}(j) - \mathbf{n}(k)\|^2}{6}$, $B = \frac{\|\mathbf{n}(i) - \mathbf{n}(j)\|^2}{2}$, $C = \frac{\|\mathbf{n}(i) - \mathbf{n}(k)\|^2}{2}$, and ω is a positive constant. Since the face f_i is the seed face, the normal vector difference from $\mathbf{n}(j)$ and $\mathbf{n}(k)$ to $\mathbf{n}(i)$ should weigh more than that between $\mathbf{n}(j)$ and $\mathbf{n}(k)$. Our experiments have shown that the ratio used above yields good results. Note that there are typically more than one path going from f_i to f_k . In this case, the distance is calculated for each path and the shortest one can be chosen using Dijkstra's algorithm. After the normal-weighted distance is calculated for every triangle in the neighborhood of the face f_i , the anisotropic neighborhood is chosen as those triangles whose distances to the seed face are less than a pre-specified distance threshold.

The size of neighborhoods could be determined by the distance threshold. If the threshold value is too high, the size of neighborhoods would be relatively large, while the size of neighborhoods would be small if the threshold is too low. In our method, the quadratic

surface is chosen for local surface fitting, which requires at least six neighboring vertices in the neighborhoods. On the other hand, too many neighboring vertices will increase the computational time and ruin the local surface features as well. For these reasons, in our implementation we specify a maximal size δ of neighborhoods, i.e., the maximal number of neighboring faces, as an additional condition of anisotropic neighborhood selection. Our experiments show that $\delta = 10$ is a good choice.

3.3 Fitting-Based Mesh Denoising

The surface fitting described in Section 2 is performed on each vertex, but the anisotropic neighborhood searching given in Subsection 3.2 is carried out on triangles. To combine the anisotropic neighborhood selection into the fitting-based mesh denoising algorithm, we check each mesh vertex v and find one of the incident triangles, denoted by t , which has the most similar normal direction to the normal vector of the vertex v (see (2) on the approximation of the normal of a vertex). The selected triangle t is used as the seed face in the anisotropic neighborhood searching algorithm. All vertices in the neighborhood found are considered as the data points, with which the local surface fitting and projection are carried out to get the new position of the vertex v . Locally, the neighborhood of a vertex is approximated by a quadratic surface. By repeating the fitting and projection processes, the vertices are expected to converge to the corresponding fitted surfaces. Accordingly, the iterative procedure is performed repeatedly until convergency, i.e., the Hausdorff distance between the two sequential denoised meshes at the i -th and $(i+1)$ -th iterations is less than a user-defined small threshold d . In our experiments, $d = 0.05 \times \text{mean edge length}$ generally produces good results.

4 Results and Discussion

We have implemented all algorithms described in this paper with Visual C++ and OpenGL, running on a Pentium IV PC with 2.0 GHz, and tested them on scanned data and CAD mesh models with different levels of noise.

Parameter Selection

In the comparisons shown below, we attempt to get the best results for each method by fine tuning their parameters. In our algorithm, the parameters mainly consist of the iterations of face normal filtering: n_1 , the angle threshold for face normal filtering: T , the number of anisotropic neighboring vertices for fitting: n_2 , and the distance threshold: d .

Among those parameters, the number of iterations n_1 for face normal filtering is set according to the level of noise. If there is a high level of noise, n_1 should be relatively big; otherwise, it is set with a small value. The angle threshold T in the face normal filter is related to the “sharpness” of the features. If the mesh contains very sharp features that need to be preserved, T should be a small value; otherwise, it is assigned with a relatively high value. For example, the fandisk model contains much sharper features than the bunny model and thus T should be set smaller in the fandisk model than the bunny model. The size of neighborhood n_2 is selected according to the curvature of the mesh at a vertex. If the curvature is small, indicating the surrounding region of the vertex is relatively flat, the size of neighborhood n_2 could be relatively big; otherwise, it should be set smaller.

Comparison with Related Methods

To demonstrate the effectiveness of our method, we compare it with several related methods, including Ohtake *et al.*'s^[16], Yagou *et al.*'s^[15], Fleishman *et al.*'s^[14], Jones *et al.*'s^[13], Hilderbrandt and Polthier's^[20], Sun *et al.*'s^[18] methods.

Figs. 5 and 6 show the denoising results for a twirl model and a sphere model with sharp features. All methods preserve sharp features well except Fleishman *et al.*'s method^[14] in Fig.5 and Hilderbrandt and Polthier's method^[20] in Fig.6. Ohtake *et al.*'s method^[16] and Sun *et al.*'s method^[18] can preserve most sharp features, but some of them are still corrupted, as indicated in Figs. 5 and 6. By contrast, our approach produces smoother surface meshes for these two models and preserves all important features. In the following figures, $\bar{l}_e$ stands for the mean edge length.

Fig.7 gives the denoising results of a bust model. Yagou *et al.*'s^[15] method preserves features so excessively that many non-sharp features with small curvatures are sharpened (for example, the tip of the nose). The same problem occurs occasionally by Hilderbrandt and Polthier's^[20], and Sun *et al.*'s^[18] methods (see the wavy hair). Using local surface fitting with anisotropic neighborhoods, our algorithm is able to remedy this problem by generating a smooth and faithful mesh as compared to the noisy model.

Fig.8 shows the denoising results of a scanned model – the femme with real noise. All four approaches generate good results. However, there are some disordered faces in the result by Sun *et al.*'s^[18] method and some bumpy faces by Ohtake *et al.*'s^[16] method. In this particular model, Jones *et al.*'s^[13] method seems to produce the smoothest result. The scanned data usually are dense and there are also a number of points in noisy areas. If small neighborhoods are used to fit local surfaces in our method, the noisy area will be incorrectly considered as a feature and thus preserved. On the other hand, if we choose relatively big neighborhoods, the above-mentioned problem can be fixed, but the computational time will increase considerably. In our current implementation, the maximal size of neighborhoods is pre-defined by the users. As part of our future work, we shall consider an adaptive sizing function of neighborhoods that is determined by the size of local features.

Fig.9 shows the denoising results for a fandisk mesh, corrupted by the Gaussian noise with 0.2 times mean edge length. Fleishman *et al.*'s^[14] method blurs some sharp edges, while all three other methods preserve most of the sharp features. Note that the small, non-sharp feature indicated by the rectangles in the model is preserved only by our method. Another difference between these resulting meshes is the curvy edge going from the center to the bottom of the model in Fig.9. Again, our methods preserves better than other approaches on this edge feature.

Fig.10 gives the denoising results of the bunny model. From the curvature maps, we can see that both the sharp features around the boundaries of the “i” and “H” and the non-sharp, wavy features on the bunny body are well preserved in our method.

Fig.11 gives the denoising results of the Stanford dragon mesh model, which contains many complicated features. From the comparisons, we notice that small features may be preserved better with Jones *et al.*'s^[13] and our methods than the other two methods. For more complicated features, they may be constructed with higher-order surfaces. However, locally the mesh could be represented fairly well with quadratic surface patches. We have tested higher order surfaces using cubic and even higher-order surfaces to fit the local surfaces. There is no significant improvements over the quadratic surface from the results we have seen.

The results shown above have visually demonstrated the superiority of our algorithm to other methods in terms of mesh smoothing and feature-preserving. Here we provide some quantitative comparisons between our approach and other methods. Since our algorithm essentially performs on the normal fields, we follow [18] to measure the difference between the normals of the original mesh and the denoised mesh, in which the error metric is defined with the mean square angular error. We compare the angular errors resulting from [18] and

our method on the testing meshes in Fig.12. Apparently, our method results in smaller normal errors.

To demonstrate the fidelity of the smoothed mesh relative to the original surface, the Hausdorff distance between the two meshes is calculated with the software tool called Metro^[30]. Fig.13 shows a detailed comparison of the Hausdorff distance results, where the horizontal axis is the absolute distance value between the smoothed mesh and the original mesh, and the vertical axis is the corresponding histogram (in percentage) with respect to each distance value. From this figure, we can see that our method yields small Hausdorff distances, indicating that our method produces very close surface mesh relative to the original model, in addition to its high efficiency of noise removal.

Table 1 shows the timing comparison for these models using different methods. Compared with other methods, our approach is not the fastest but the disparity is insignificant. With comprehensive consideration of the quality and timing, our method still has better performance.

5 Conclusions

We have presented a novel feature-preserving mesh denoising method using the surface fitting and projection technique, combined with the anisotropic neighborhood-searching scheme. The results demonstrated in the current paper show that by considering the anisotropic neighborhoods in the surface fitting, we are able to achieve promising mesh denoising on many types of input mesh models. The comparisons with several other popular methods also show that our approach is very effective in retaining both sharp (high-curvature) and non-sharp (low curvature) features. Furthermore, the histograms of Hausdorff distances from the denoised meshes to the original ones indicate that our method possesses an excellent performance in preserving the geometry of the given model (e.g., surface areas, volumes, local curvatures). As part of our efforts in the present paper, the software package implementing the algorithms described will be made available on a public domain for downloading.

As mentioned in Section 4, the maximal number of anisotropic neighborhoods is given by the users in our algorithm. However, different features in a model may have distinct geometric properties, such as sizes. Therefore, it may be more desirable to implement an adaptive scheme to select the anisotropic neighborhoods according to the estimation of the local feature size. We plan to explore these topics in our future work.

Acknowledgments

The work described was supported in part by the National Institutes of Health of USA under Grant No. R15HL103497 from the National Heart, Lung, and Blood Institute (NHLBI) and by a subcontract of NIH Award under Grant No. P41RR08605 from the National Biomedical Computation Resource. The content is solely the responsibility of the authors and does not necessarily represent the official views of the sponsors.

Biography

Jun Wang received the B.S. and Ph.D. degrees both in mechanical engineering from Nanjing Univ. Aeronautics and Astronautics (China) in 2002 and 2007, respectively. He was a postdoctoral scholar at the University of California-Davis from April 2008 to April 2009. Since then he has been a postdoctoral scholar at the University of Wisconsin-Milwaukee. His research interests are geometric modeling, in particular, mesh segmentation and processing.

Zeyun Yu received the B.S. degree in mathematics from Peking University (China) in 1996 and the M.S. degree in pattern recognition and machine intelligence from Chinese Academy of Sciences (China) in 1999. He was awarded the Ph.D. degree in computer science from the University of Texas at Austin in 2006 and then worked as a postdoctoral scholar at the Department of Mathematics, University of California-San Diego. Since August 2008, he has been an assistant professor in the Department of Computer Science, University of Wisconsin-Milwaukee. His research interests span from algorithms in image analysis, pattern recognition, numerical simulation, and 3D visualization, to applications in structural and functional modeling of biomedical systems.

References

1. Field DA. Laplacian smoothing and Delaunay triangulations. *Commun. Applied Numerical Methods*. 1988; 4(6):709–712.
2. Taubin, G. A signal processing approach to fair surface design.. *Proc. the 22nd Conf. Computer Graphics and Interactive Techniques*; Los Angeles, USA. Aug. 6-11, 1995; p. 351-358.
3. Desbrun, M.; Meyer, M.; Schröder, P., et al. Implicit fairing of irregular meshes using diffusion and curvature flow.. *Proc. the 26th Annual Conf. Computer Graphics and Interactive Techniques*; Los Angeles, USA. Aug. 8-13, 1999; p. 317-324.
4. Choudhury, P.; Tumblin, J. The trilateral filter for high contrast images and meshes.. *Proc. the 14th Eurographics Symp. Rendering*; Leuven, Belgium. Jun. 25-27, 2003; p. 186-196.
5. Shen Y, Barnew KE. Fuzzy vector median-based surface smoothing. *IEEE Trans. Visualization and Computer Graphics*. 2004; 10(3):252–265.
6. Zhao, H.; Xu, G. Directional smoothing of triangular meshes.. *Proc. Int. Conf. Computer Graphics, Imaging and Visualization*; Beijing, China. Jul. 26-29, 2005; p. 397-402.
7. Bose NK, Ahuja NA. Superresolution and noise filtering using moving least squares. *IEEE Transactions on Image Processing*. 2006; 15(8):2239–2248. [PubMed: 16900679]

8. Ji Z, Liu L, Wang G. Non-iterative global mesh smoothing with feature preservation. *Int. J. CAD/CAM*. 2006; 6(1)
9. Fan H, Yu Y, Peng Q. Robust feature-preserving mesh denoising based on consistent sub-neighborhoods. *IEEE Trans. Visualization and Computer Graphics*. 2010; 16(2):312–324.
10. Tasdizen, T.; Whitaker, R.; Burchard, P., et al. Geometric surface smoothing via anisotropic diffusion of normals.. *Proc. Visualization*; Boston, USA. Oct. 22-Nov. 1, 2002; p. 125-132.
11. Tomasi, C.; Manduchi, R. Bilateral filtering for gray and color images.. *Proc. the 6th International Conference on Computer Vision*; Bombay, India. Jan. 4-7, 1998; p. 839-846.
12. Durand, F.; Dorsey, J. Fast bilateral filtering for the display of high-dynamic-range images.. *Proc. the 29th Annual Conferences on Computer Graphics and Interactive Techniques*; San Antonio, USA. Jul. 21-26, 2002; p. 257-266.
13. Jones, T.; Durand, F.; Desbrun, M. Non-iterative, feature preserving mesh smoothing.. *Proc. ACM SIGGRAPH 2003*; San Diego, USA. Jul. 27-31, 2003; p. 943-949.
14. Fleishman, S.; Drori, I.; Cohen-Or, D. Bilateral mesh denoising.. *Proc. ACM SIGGRAPH 2003*; San Diego, USA. Jul. 27-31, 2003; p. 950-953.
15. Yagou, H.; Ohtake, Y.; Belyaev, AG. Mesh smoothing via mean and median filtering applied to face normals.. *Proc. Geometric Modeling and Processing*; Wako, Japan. Jul. 10-12, 2002; p. 124-131.
16. Ohtake, Y.; Belyaev, AG.; Seidel, HP. Mesh smoothing by adaptive and anisotropic Gaussian filter applied to mesh normals.. *Proc. Vision, Modeling, and Visualization*; Erlangen, Germany. Nov. 20-22, 2002; p. 203-210.
17. Belyaev, A.; Ohtake, Y. Nonlinear diffusion of normals for crease enhancement.. *Proc. Vision Geometry X, SPIE Annual Meeting*; San Diego, USA. Jul. 29-30, 2001; p. 42-47.
18. Sun X, Rosin P, Martin R, Langbein F. Fast and effective feature-preserving mesh denoising. *IEEE Trans. Visualization and Computer Graphics*. 2007; 13(5):925–938.
19. Sun X, Rosin P, Martin R, Langbein F. Random walks for feature-preserving mesh denoising. *Computer Aided Geometric Design*. 2008; 25(7):437–456.
20. Hildebrandt K, Polthier K. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*. 2004; 23(3):391–400.
21. Shepard, D. A tow-dimensional interpolation function for irregularly-spaced data.. *Proc. the 23rd ACM National Conference*; Aug. 7-29, 1968; p. 517-524.
22. Levin, D. Mesh-independent surface interpolation.; *Geometric Modeling for Scientific Visualization*. 2003. p. 39-47.
23. Oztireli C, Guennebaud G, Gross M. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*. 2009; 28(2):493–501.
24. Fleishman S, Cohen-Or D, Silva CT. Robust moving least-squares fitting with sharp features. *ACM Transaction on Graphics*. 2005; 24(3):544–552.
25. Lin H, Wang G, et al. Parameterization for fitting triangular mesh. *Progress in Natural Science*. 2006; 16(11):1214–1221.
26. Lai YK, Hu SM, Pottmann H. Surface fitting based on a feature sensitive parameterization. *Computer-Aided Design*. 2006; 38(7):800–807.
27. Cazals F, Pouget M. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*. 2005; 22(2):121–146.
28. do Carmo, MP. *Differential Geometry of Curves and Surfaces*. Prentice-Hall; 1976.
29. Golub, GH.; van Van Loan, CF. *Matrix Computations*. 3rd edition. Johns Hopkins; 1996.
30. Cignoni P, Rocchini C, Scopigno R. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*. 1998; 17(2):167–174.

Fig.1.

2D Illustration of the process of mesh denoising. (a) Input noisy data, which can be interpreted as piecewise smooth surfaces. (b) One vertex of noisy data (in green). We search the anisotropic neighborhoods (in yellow) of the vertex in (c). The local surface (in blue) of the neighborhoods is fitted in (d), and the vertex is projected onto the fitted surface to obtain the corresponding projection vertex (in red) in (e). By this mean, all vertices are projected onto their associated fitted surfaces (g) and updated with the projection vertices (in red) (h). As a result, the noisy data are denoised.

Fig.2.

Illustration of the local coordinate system (LCS) on a surface S , where p is the origin; h -axis directs along the normal vector n_p of p on S ; and u -, v -axes are orthogonal vectors in the tangent plane of p on S .

Fig.3. Mesh denoising by surface fitting. Note that non-sharp features have been well preserved, while the boundaries of “i” and “H” with sharp features have been blurred.

Fig.4.
Illustration of the normal-weighted distance.

Fig.5.

Denoising of a twirl model. (a) Original model. (b) Noisy model (Gaussian noise with standard deviation: $0.25 \times$ mean edge length). (c) Denosing results by Fleishman *et al.*'s^[14] ($n = 5$) method. (d) By Ohtake *et al.*'s^[16] method. (e) By Sun *et al.*'s^[18] method ($n_1 = 10$, $n_2 = 30$, $T = 0.4$). (f) By our method ($n_1 = 10$, $T = 0.4$, $n_2 = 20$, $d = 5\% \times I_e$).

Fig.6.

Denoising of a sphere model. (a) Original model. (b) Noisy model (Gaussian noise with standard deviation: $0.25 \times$ mean edge length). (c) Denoising results by Hilderbrandt and Polthier's^[20] method ($n = 70$). (d) By Sun *et al.*'s^[18] method ($n_1 = 10, n_2 = 20, T = 0.4$). (e) By Ohtake *et al.*'s^[16] method. (f) By our method ($n_1 = 10, T = 0.4, n_2 = 20, d = 5\% \times l_e$). Note that the sharp features in the boxes are corrupted by all methods except ours.

Fig.7.

Denoising of a bust model. (a) Original model. (b) Noisy model (Gaussian noise with standard deviation: $0.25 \times$ mean edge length). (c) Denosing results by Yagou *et al.*'s^[15] method ($n = 10$). (d) By Hilderbrandt and Polthier's^[20] method ($n = 50$). (e) By Sun *et al.*'s^[18] method ($n_1 = 30, n_2 = 50, T = 0.55$). (f) By our method ($n_1 = 30, T = 0.55, n_2 = 30, d = 5\% \times l_e$). Yagou *et al.*'s method and Sun *et al.*'s method intend to excessively sharpen non-sharp features (see the nose-tip and hair) and some bumpy faces are generated by Hilderbrandt and Polthier's method, while our method can preserve most features.

Fig.8.

Denoising of a femme model. (a) Original scanned model. (b) Denoising results by Jones *et al.*'s^[13] method. (c) By Sun *et al.*'s^[18] method ($n_1 = 10$, $n_2 = 20$, $T = 0.6$). (d) By Ohtake *et al.*'s^[16] method. (e) By our method ($n_1 = 10$, $T = 0.6$, $n_2 = 30$, $d = 5\% \times I_e$).

Fig.9.

Denoising of a fandisk model. (a) Original model. (b) Noisy model (Gaussian noise with standard deviation: $0.25 \times$ mean edge length). (c) Denosing results by Fleishman *et al.*'s^[14] ($n = 5$) method. (d) By Sun *et al.*'s^[18] method ($n_1 = 20, n_2 = 30, T = 0.5$). (e) By Ohtake *et al.*'s^[16] method. (f) By our method ($n_1 = 20, T = 0.5, n_2 = 20, d = 5\% \times l_e$). Note that the small feature enclosed in the rectangles is reserved only by our method.

Fig.10.

Denoising of an iH-bunny model. (a) Original model. (b) Noisy model (Gaussian noise with standard deviation: $0.25 \times$ mean edge length). (c) Denosing results by Ohtake *et al.*'s^[16] method. (d) By Sun *et al.*'s^[18] method ($n_1 = 5$, $n_2 = 20$, $T = 0.6$). (e) By Hilderbrandt and Polthier's^[20] method ($n = 100$). (f) By our method ($n_1 = 5$, $T = 0.6$, $n_2 = 30$, $d = 5\% \times l_e$). From the mean curvature maps on the bottom row, we can see that both sharp and non-sharp features are preserved very well with our method.

Fig.11.

Denoising on the dragon model. (a) Original model. (b) Noisy model (Gaussian noise with standard deviation: $0.2 \times$ mean edge length). (c) Denoising results by Jones *et al.*'s^[13] method. (d) By Sun *et al.*'s^[18] method ($n_1 = 5$, $n_2 = 20$, $T = 0.55$). (e) By Hilderbrandt and Polthier's^[20] method ($n = 80$). (f) By our method ($n_1 = 5$, $T = 0.5$, $n_2 = 25$, $d = 5\% \times l_e$). From the results, we can see that small features are well preserved with Jones *et al.*'s^[13] and our method.

Fig.12. Normal errors based comparisons between the method in [18] and ours.

Fig.13.

Histograms show the Hausdorff distances between the meshes obtained by each approach and the original meshes. The horizontal axis is the error (absolute distance value) from the denoised mesh to the original mesh, and the vertical axis is the corresponding percentage to each error value. (a) Data set. (b) The corresponding Hausdorff distances by Ohtake *et al.*'s^[17] method. (c) By Sun *et al.*'s^[18] method. (d) The first, fourth and fifth rows: by Hilderbrandt and Polthier's^[20] method; the second and third rows: by Fleishman *et al.*'s^[14] method. (e) By our method. Please note that the second and third models in (c) by Sun *et al.*'s method show a very good agreement on most vertices between the noisy and smoothed meshes. However, there are a few vertices where the errors are significantly large (see the spikes in both histograms).

Table 1

Timing Comparison of Mesh Denoising (s)

Method	Twirl	Fandisk	Sphere	iH-Bunny	Femme	Bust
	V_0 : 5 201	V_0 : 6 475	V_0 : 10 443	V_0 : 34 834	V_0 : 40 544	V_0 : 51 073
	T_0 : 10 402	T_0 : 12 946	T_0 : 20 882	T_0 : 69 451	T_0 : 76 522	T_0 : 102 142
Fleishman's	0.045	0.052	0.102	0.325	0.385	0.504
Ohtake's	0.335	0.341	0.714	2.648	3.014	3.854
Sun's	0.074	0.082	0.154	0.541	0.627	0.859
Ours	0.249	0.253	0.527	1.952	2.249	3.514

Note: V_0 —number of vertices; T_0 —number of triangles.