1. Introduction

Mesh denoising, which aims at achieving high-quality 3D models from meshes corrupted with noise, has recently attracted increasing interest in graphic applications, e.g., animation, rendering, reverse engineering and so on. The emergence of noise and uneven faces is unavoidable in those data captured from scanning real physical models. The sources of noise are numerous including measuring, algorithmic errors and limited sampling resolution (Levoy et al., 2000, Rusinkiewicz et al., 2002). Therefore, an effective mesh denoising algorithm is indispensable for further mesh processing.

Different from the fairing and smoothing algorithms, the main technical challenge of robust mesh denoising is to remove noise while maximally preserving geometric features. One of the major obstacles is that sharp features are often corrupted by noise. There exist challenging technical problems we need to solve. First, in the presence of noise, how can we identify feature and non-feature vertices? Second, how should we effectively perform noise removal for feature vertices while preventing feature blurring? Third, how do we remove the sparse strong noise points on flat region? Anisotropic treatment is often adopted to denoise the mesh while preserving features such as sharp edges and corners which are very important for the underlying models.
Researchers have made noticeable progresses in mesh denoising (Lee et al., 2005, Zheng et al., 2011, Zhang et al., 2015, Wei et al., 2015, Wei et al., 2017, Lu et al., 2017). Most recently, the bilateral normal filter (Lee et al., 2005, Zheng et al., 2011) and the guided bilateral normal filter (Zhang et al., 2015) have been successfully applied to mesh denoising to yield desirable results. But denoising is far from mature and remains active due to its complexity. As for the guided bilateral normal filter, the guidance geometry is not easily available compared to the case of images. Due to the nature of the bilateral filter, it cannot entirely eliminate Gaussian noise. The inherent deficiency is caused by the introduction of the signal domain weight and cannot fully prevent the side effects from geometrically inconsistent facets. The method in (Wei et al., 2015) proposed a binormal method for mesh denoising. The authors empirically set some parameters and they work well for the experimental models. However, they may not be suitable for all models with different structures and/or different levels of noise. Automatically finding an optimal set of parameters for a given model is still a challenge in the feature. And because this method didn’t pretreat the noise before vertex classification, if confronted with high level of noise, it would detect a large number of pseudo-features, which will degrade the performance of the method.

Fig. 1. The denoising results of Block model corrupted by Gaussian noise with standard deviation $s = 0.15$ mean edge length. From left to right: the ground truth (with wireframe), the noisy input, Sun et al.’s result (2007), result by Zheng et al.’s local method (2011), result by Zheng et al.’s global method (2011), Zhang et al.’s result (2015), and our method’s result. The bottom row shows the close-ups of the red rectangular region.

To tackle the problems of removing large-scale noise and preserving geometric features, an efficient approach for mesh denoising is proposed in this paper. The key idea of our approach is to use different denoising techniques on the different types of vertex region. More specifically, the main steps of our method are as follows. First, we preprocess the noisy model globally to reduce the interference of the high level of noise. Next, we classify the vertices of the mesh into feature ones and non-feature ones. Then we adopt different denoising methods for different types of vertex region. For non-feature areas, we apply Sun et al.’s method (2007) which is fast to remove noise and effective to overcome outliers. For feature zones, we design a special filter to prevent the side effects of facets with very different normals. It can avoid the selection of parameter values and achieve the local optimal. Finally, the vertex positions are updated according to the filtered face normal and vertex normal.

The main contributions of this work. We present a kind of collaborative filters which overcomes the main technical challenge of mesh denoising, i.e., removing noise while maximally preserving geometric features. Specifically, first of all, preprocessing is very valid when the level of noise is high. Second, we select different filters to estimate the face normals according to the detected category of vertices. For feature region, our method adaptively averages the most consistent face normals, which can achieve local optimum and avoid searching for the optimal solutions of parameters which is very difficult due to the coexistence of feature and non-feature vertices. For non-feature region, a truncated weight function is adopted to effectively eliminate the interference of strong noise points.

As shown in Fig. 1, the utilization of the collaborative filters makes the denoising results visually appealing. Extensive experiments reported in Section 7 further demonstrate that our algorithm can achieve higher quality results than previous approaches on noisy meshes.
2. Related work

Our review here focuses only on the existing works that are most related to ours. Recently, an iterative, two-step method is very popular which filters normals followed by updating vertices. Yagou et al. proposed to use the mean, median, and alpha-trimming filters to estimate face normals (Yagou et al., 2002, 2003). In general, these approaches are accurate in relatively flat regions. But these filters may lead to feature blurring. Later both (Shen and Barner, 2004) and (Sun et al., 2007, 2008) have adopted the two-stage framework to filter face normals. Compared with one-stage methods, i.e., directly adjusting vertex positions, two-stage methods are more effective for recovering features.

In recent years, most of the mesh denoising approaches adopt anisotropic methods than isotropic ones in order to preserve sharp features (Ohtake et al., 2001, 2002, Bajaj and Xu, 2002, Desbrun et al., 2000, Clarenz et al., 2000, Hildebrandt and Polthier, 2004). Many denoising approaches are inspired by image processing. For instance, diffusion based methods extend anisotropic diffusion for 2D grids to anisotropic geometric diffusion on surfaces. (Hildebrandt and Polthier, 2004) introduced mean curvature flows to preserve features. Although the results have high quality, these methods rely on shock formation to preserve details, which suffer from the numerical instability of the diffusion equations. In fact, diffusion method and bilateral filtering have an essential relationship (Barash, 2002).

The bilateral filter has proved to be a very effective edge-preserving filter for image processing (Tomasi and Manduchi, 1998). Jones et al. (2003) and Fleishman et al. (2003) successfully extended the bilateral filter from image denoising to mesh denoising by directly adjusting vertex positions. In view that face normals can better represent local surface geometry than vertex positions, Zheng et al. (2011) proposed bilateral filters to handle surface normal instead of vertex positions. Later, Zhang et al. (2015) presented guided bilateral normal filter, because the noise in the input mesh makes the signal from a noisy data unreliable for recovering the output signal.

On the other hand, several sparse optimization methods have been proposed (He and Schaefer, 2013, Wang et al., 2014, Zhang et al., 2015). It is considered that sharp features are usually sparse on general mesh surfaces. He and Schaefer (2013) employ $\ell_0$ norm to minimize energies. The method can achieve good denoising results. All these global methods are numerically more robust. However, for meshes with fine local details, they could often over-smooth the meshes.

In order to achieve better denoised results, several mesh denoising techniques employ vertex/face classification before denoising processing (Chen and Cheng, 2005, Bian and Tong, 2011, Wang et al., 2012, Park et al., 2013, Wei et al., 2015, 2017, Zheng et al., 2017). Various curvature-based methods for classification have been applied to denoising algorithms. Chen and Cheng (2005) used Bayesian discriminant analysis to determine filters for separating potential feature vertices from non-feature in curvature space. As known, higher order derivatives are sensitive to noise, and lead to unsatisfactory results, even if the pre-filtering step is adopted. Some efforts (Sun et al., 2002, Kim et al., 2009, Wei et al., 2015, 2017) devote themselves to the normal tensor voting for vertex classification. However, the normals have been contaminated by noise, therefore, the clustering results in terms of original normals are not reliable. Lu et al. (2017) pre-filtered the noisy input mesh to substantially decrease the noise influence for the subsequent steps. In a similar way, we execute the normal tensor voting on the pretreated model rather than directly on the noisy input, which can improve the accuracy of vertex classification.

3. Overview

In this paper, we present a feature-preserving mesh denoising algorithm that combines global optimization, vertex classification, normal estimation, and vertex position updating. We perform the following steps during the denoising procedure.

1) Noise preprocessing. The noise in the input signal can lead to erroneous vertex classification. Preprocessing is essential for distinguishing features from noise when the noise level is high. Initial estimation can reduce the noise level to some extent and generate a sound initialized mesh for the follow-up steps.

2) Vertex classification. It is an arduous task to preserve geometric features when indiscriminately treating feature vertices from non-feature ones. To solve this problem, in this step, we classify the vertices into the face, and edge and corner clusters by the normal tensor voting (Kim et al., 2009). The k-means clustering is chosen for the classification. Benefited from vertex classification, we can perform different denoising methods for different types of vertices.

3) Face normal updating. We update face normals based on vertex classification. We adopt Sun et al.’s fast normal
filter at non-features which can remove outliers effectively. At features, we propose a method which can average the most consistent face normals to avoid the influence from face normals with great difference. The collaborative filters are more effective than a single filter.

4) Vertex position updating. After adjusting the face normals, the vertex positions are iteratively updated with elaborately designed weights. We consider the influence from surface sampling rate, spatial distance and signal difference. Stopping criterion for iteration is controlled by the specified error or the maximum number of iterations.

4. Noise preprocessing

When the noise level is high, vertex classification method does not work well enough, as the model may be distorted very severely. So it is necessary to preprocess noise. The global mesh optimization method is numerically robust for large-scale noise. We formulate the initial estimation of the input noisy model as a least square optimization problem

$$\arg\min_{\hat{V}} \|V' - \hat{V}\|_2 + \alpha \|LV\|_2^2,$$

where $V'$ are the vertices of the reconstructed model, $V$ are their initial positions, and $L$ is the Laplacian matrix. $LV' = [\delta_1, \delta_2, \cdots, \delta_n]^T$, where $\delta_i$ can be described by the difference between vertex $v_i$ and the weighted average of its neighbors. Fig. 2 shows an example to illustrate the necessity of the noise preprocessing. Without this stage, the input model could hardly be effectively denoised afterwards since many pseudo-features are detected.

![Fig. 2. An example for illustrating the preprocessing efficacy.](image)

(a) Ground truth  (b) Noisy input  (c) Preprocessing result  (d) Result w/o preprocess.  (e) Result w/ preprocess

5. Vertex classification

In this section, we introduce the k-means clustering to find the feature vertices based on the normal voting tensor. The k-means clustering is good at capturing feature and is of high computational efficiency.

5.1 Normal voting tensor

(Sun et al., 2002) and (Kim et al., 2009) defined the normal voting tensor of a vertex on a triangular mesh as a weighted sum of covariance matrices of the unit normal vectors of its neighboring triangles

$$T_v = \sum_{f_i \in N_f(v)} \mu_i V_{f_i} = \sum_{f_i \in N_f(v)} \mu_i n_i n_i^T,$$

where $N_f(v)$ denotes the set of one-ring faces around $v$, $V_{f_i}$ is the covariance matrix corresponding to face $f_i$, $n_i$ represents the unit normal of face $f_i$, and $\mu_i$ is a weight function decided by the area of $f_i$ and the distance
between the barycenter of triangle $f_i$ and the central vertex $v$. The normal voting tensor $T_v$ is a symmetric positive semi definite second-order tensor, so it can be diagonalized into a diagonal matrix composed of eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$). According to the three decreasing eigenvalues, we can classify vertices on the mesh into face-, edge- and corner- vertices.

- Face (non-feature) vertex: $\lambda_1$ is dominant, and $\lambda_2$ and $\lambda_3$ are close to 0.
- Sharp edge vertex: $\lambda_1$ and $\lambda_2$ are dominant, and $\lambda_3$ is close to 0.
- Corner vertex: $\lambda_1, \lambda_2, \lambda_3$ are dominant.

It is accurate to differentiate feature vertex from non-feature vertex using the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ on high quality mesh. As the result of the existence of noise, it is difficult to precisely categorize the vertices. The k-means algorithm is adopted for clustering vertices according to eigenvalues $\lambda_1, \lambda_2, \lambda_3$, which are normalized to a vector. The characteristics of each vertex can be expressed as $\lambda_n = \text{normalize} (\lambda_{1n}, \lambda_{2n}, \lambda_{3n})$.

Table 1 shows three typical normalized vectors of eigenvalues for face-, edge- and corner- vertices.

<table>
<thead>
<tr>
<th>Type</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>$\lambda_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Edge</td>
<td>0.7071</td>
<td>0.7071</td>
<td>0</td>
</tr>
<tr>
<td>Corner</td>
<td>0.5774</td>
<td>0.5774</td>
<td>0.5774</td>
</tr>
</tbody>
</table>

5.2 k-means clustering

We use the normalized vectors composed of eigenvalues of normal voting tensors of all vertices as categorical data $\{\lambda_i\}$. Here, we introduce the k-means clustering to complete the classification, since it is computationally efficient. The k-means clustering aims to partition each data point into one of the $k$ clusters $S_k = \{S_k(1), S_k(2), \ldots, S_k(k)\}$ to minimize the sum of within-cluster squared distances

$$\arg \min_{S_k} \left\{ \sum_{j=1}^{k} \sum_{\lambda_j \in S_k(j)} \| \lambda_j - \mu_{S_k(j)} \|^2 \right\},$$

(3)

where $S_k(j)$ denotes a set of data points in the $j$-th cluster and $\mu_{S_k(j)}$ is the centroid of $S_k(j)$, i.e., the mean of points in $S_k(j)$. We specify $k = 3$, which means $S_k$ consists of 3 clusters, that is, face (non-feature) vertex cluster $S_F$, sharp edge vertex cluster $S_E$ and corner vertex cluster $S_C$. Instead of starting from random initial centers of clusters, we give better starting points to speed up the convergence:

$$\mu^n_{S_F} = (1,0,0), \quad \mu^n_{S_E} = (0.7071,0.7071,0), \quad \mu^n_{S_C} = (0.5774,0.5774,0.5774).$$

6. Mesh denoising algorithm

In this section, we present collaborative filters on the classified vertices for mesh denoising. Our denoising algorithm is an iterative, two-step method. The procedure for the algorithm is as follows:

1) Adjust the normal for each polygon face. According to the vertex classification results from the above section, we choose different filters to update the face normals.

a) For non-feature regions, we adopt Sun et al.’s method (2007) to achieve smoothing effect, as it runs fast and can remove large outliers.


b) For feature regions (edges and corners), we present a filter which can only average the normals of the neighboring faces that have the most similar normal direction as the central face to inhibit the inconsistence of geometry.

2) After filtering the normals, the vertex positions are updated to match the new normals.

### 6.1 Face normal estimation

The underlying surface of a mesh is piecewise smooth, rather than smooth everywhere. For example, a surface is divided into two smooth regions by sharp edges. The normal estimation by averaging its neighboring face normals will suffer from significant error due to the inconsistence of geometry.

Unlike the previous methods, such as bilateral normal filtering (Zheng et al., 2011) and guided bilateral normal filtering (Zhang et al., 2015), which use the same type of filter with unified standard deviation parameter $\sigma$ of Gaussian kernel function on the whole mesh, we adopt collaborative filters to discriminately deal with different types of regions, and obtain a more accurate face normal field. In fact, the standard deviation $\sigma_y$ of Gaussian kernel function measuring signal difference used therein would affect the filtering result, and should be set different according to the level of noise and the geometric features of model. Empirically, $\sigma_y$ should be large when the noise level is high, and small when the feature degree is high. But Zheng and Zhang et al. adopt a constant $\sigma_y$ for the whole model. Moreover, it is not easy to find an optimal value for Gaussian standard deviation $\sigma_y$.

Although the bilateral filter inhibits the side effect from normal direction inconsistent facets by taking advantage of the Gaussian kernel function, the scheme cannot entirely eliminate the unfavorable effect from the inconsistent neighbors. So the idea of truncation at the feature was introduced to achieve a better result by Sun et al. (2007). And Wang et al. (2012) adopted a bilateral filter which takes a truncated function as a weight for getting better results.

A face is considered as a feature face if it contains a feature vertex, i.e., corner or sharp edge vertex. We estimate the face normals based on the classified results of vertices as follows.

1) Non-feature: We adopt Sun et al.’s method (2007) to smooth the non-feature regions, because the method defines a truncated weight function which is effective for removal of strong outliers. There often exist some strong noise points on the noisy input model, and the bilateral filter can only inhibit the influence of strong noise points to some extent, but it cannot get rid of them completely.

For any models corrupted by Gaussian noise of mean $\mu$ and standard deviation $s$, more than 95% of noise lie within two standard deviations of the mean: $[\mu-2s, \mu+2s]$, and over 99.7% of noise lie within three standard deviations of the mean: $[\mu-3s, \mu+3s]$. For the normal distribution of zero mean, noise with intensity over $2s$ is less than 5%. Although being “small probability event”, strong noise points with intensity more than $2s$ even $3s$ still possibly appear on the model. Since the features are sparse throughout the model, these strong noise points are most likely distributed in the non-feature regions.

In this paper we perform normal updating at non-feature region by using

$$
n'_i = \text{normalize} \left( \sum_{j \in N_f(i)} h_{ij} n_{fj} \right),
$$

where $n_{fi}$ and $n'_{fi}$ are the normal of face $f_i$ before and after updating respectively, $N_f(i)$ is the one-ring face neighborhood of face $f_i$, and $h_{ij}$ is a truncated weight function defined as

$$
h_{ij} = \begin{cases} f(n_{fi}, n_{fj}, T) & \text{if } n_{fi} \cdot n_{fj} > T \\ 0 & \text{if } n_{fi} \cdot n_{fj} \leq T \end{cases}
$$

where $f(\cdot)$ is a function of $\cdot$.
Here, \(-1 \leq T \leq 1\) is a threshold determined by the user, and \(f(x)\) is a monotonically increasing function for \(x \geq 0\). \(T = 0.5\) and \(f(x) = x^2\) is a good choice for our experiments.

2) Feature: In general, the face normal is estimated by weighted averaging the face normals in the neighborhood. One-ring neighboring face normals at feature vertices have great difference. The normal estimation of the feature face based on the weighted average of its one-ring neighboring face normals will suffer from significant errors due to the geometric inconsistence. So we should avoid the unfavorable interference of faces with large normal difference from the pending face normal as much as possible.

Based on these observations, for a more precise estimate of the feature face normal field, we attempt to figure out the underlying surface geometry around a feature face. With the help of geometric information, a feature face normal is computed by only weighted averaging the neighboring face normals which have the most similar direction as the pending feature face normal. We need search for a geometrically consistent face set in the neighborhood of the pending face. We propose a simple and effective method to solve the problem.

The set of candidate faces \(N_f(i)\) for a feature face \(f_i\) consists of the faces that share a common vertex or edge with \(f_i\). First, we formulate the distance function for each candidate face \(f_j \in N_f(i)\) from the pending face \(f_i\) as follows:

\[
\phi_{ji} = d(n_{f_j}, n_{f_i}),
\]

where \(d(n_{f_j}, n_{f_i})\) is a distance function between \(n_{f_j}\) and \(n_{f_i}\), using either the \(L_2\) norm \(\|n_{f_j} - n_{f_i}\|\) or the angle \(\angle(n_{f_j}, n_{f_i})\). For the neighborhood of \(f_j\), \(\{\phi_{ji}\}\) denotes the normal difference between each face \(f_j \in N_f(i)\) and the face \(f_i\). Then, we can get \(|N_f(i)|\) number of difference values, on which the k-means clustering method is used again to search the optimal subset \(\{\phi_{ji} \in S_{\phi}(1)\}\) whose values are small and whose corresponding faces constitute the geometrically consistent patch. The k-means clustering is performed to find the partition \(S_{\phi} = \{S_{\phi}(1), \ldots, S_{\phi}(k)\}\) by minimizing the sum of within-cluster variance

\[
\arg \min_{S_{\phi}} \left\{ \sum_{m=1}^{k} \sum_{\phi_{ji} \in S_{\phi}(m)} \|\phi_{ji} - \mu_{S_{\phi}(m)}\|^2 \right\},
\]

where \(k = 2\), \(S_{\phi}(m)\) denotes a set of data points in the \(m\)-th cluster and \(\mu_{S_{\phi}(m)}\) is the centroid of \(S_{\phi}(m)\). The minimum and maximum values of \(\{\phi_{ji}\}\), which are good starting points, are chosen to speed up the convergent procedure. By k-means, we classify the neighboring faces of \(f_j\) into two clusters \(S_{\phi}(1)\) and \(S_{\phi}(2)\) according to their corresponding \(\phi_{ji}\). \(S_{\phi}(1)\) consists of the faces with small normal differences. We compute the new feature face normal of face \(f_j\) by weighted averaging the face normals with small \(\phi_{ji}\) as follows:

\[
n'_{f_j} = \frac{\sum_{f_j \in S_{\phi}(1)} A_j n_{f_j}}{\sum_{f_j \in S_{\phi}(1)} A_j n_{f_j}},
\]

where \(A_j\) is the area of face \(f_j\).

6.2 Vertex updating

Every vertex would be moved to a new position to match the new normals \(\{n'_{f_j}\}\). Inspired by Jones et al. (2003), we formulate the new position of vertex as follows:
\[ v' = v + \frac{1}{U_i} \sum_{f_i \in F_i} A_g(\|n_{i_k} - n_{i_k}\|) n'_{i_k} \cdot (c_k - v_i) , \]  
(9)

where \( v \) and \( v' \) are the vertex position before and after the update, \( A_g \) is the area of face \( f_i \), \( g(x) = \exp(-\frac{x^2}{2\sigma^2}) \) is the Gaussian function, \( F_i \) is the set of faces that share a common vertex \( v_i \), \( n_i = \text{normalize}(\sum_{f_i \in F_i} A_i n_{f_i}) \) is the normal of vertex \( v_i \) and \( U_i = \sum_{f_i \in F_i} A_g(\|n_{i_k} - n_{i_k}\|) \) is a normalization factor.

Eq. (9) can be rewritten equivalently

\[ v' = \frac{1}{U_i} \sum_{f_i \in F_i} A_g(\|n_{i_k} - n_{i_k}\|) P_i(i) , \]  
(10)

where \( P_i(i) = v_i + n'_{i_k} (n_{i_k} \cdot (c_k - v_i)) \) is the projection of vertex \( v_i \) onto the plane \( f'_i \) defined by \( n'_{i_k} \cdot (c_k - x) = 0 \), \( f'_i \) is a modification of the plane \( f_i \) whose normal has been changed from \( n_{i_k} \) to \( n'_{i_k} \). The new vertex is computed as a weighted average of the projections of the vertex onto the modified planes whose original faces have the common vertex \( v_i \).

7. Experimental results and discussion

To validate the effectiveness of our approach in removing noises and simultaneously preserving sharp features, we have tested our denoising scheme on a variety of models with either raw or synthetic noises. Synthetic noise is generated by a zero mean Gaussian function with standard deviation \( \sigma \) proportional to the mean edge length of the input mesh. We compare our results with a number of state-of-the-art denoising approaches including Sun et al.’s fast normal filtering (2007), Zheng et al.’s local bilateral normal filtering and global bilateral filtering (2011), Zhang et al.’s guided normal filtering (2015), Wei et al.’s binormal filtering (2015), Wei et al.’s piecewise moving least squares surface fitting (2017), and Lu et al.’s vertex pre-filtering and L1-median normal filtering (2017). Besides the visual qualitative comparisons, we employ the mean angular error to evaluate the fidelity of the denoised results to the ground truth in a quantitative way. Each mesh denoising method has its own parameters. For a fair comparison, we fine tune the parameters of the approaches to produce visually good results, or ask the authors to provide the results. Unlike other approaches, our method does not require frequent tuning of parameters.

7.1 Visual comparisons

In this section we visually compare our results with those obtained by the state of the art approaches on a variety of models with either synthetic or raw noises. We first test our approach on CAD-type models (Figs. 1,3,4,5,6,10), on which both corners and sharp edges are hard to be preserved exactly. However, our method can robustly generate sound results. Figs. 1,3,4,5,6,10 show the denoising results of different methods on different models corrupted by Gaussian noise with mean \( \mu = 0 \) and standard deviation \( \sigma = 0.15, 0.2, 0.3, 0.5 \) respectively, where \( \lambda \) is the mean edge length of mesh. It can be observed that our approach performs better with the increase of noise, and outperforms the state-of-the-art methods when dealing with a high level of noises. Fig. 5 demonstrates the robustness of our approach on severely destroyed models in terms of sharp edge recovery. Our approach outperforms other approaches. For SharpSphere model in Fig. 10, it is observed that all the methods can effectively remove noises, but the preservation degrees of geometry structure are different. Our result is comparable to Wei et al.’s result (2015), and better than Wei et al.’s result (2017) and Lu et al.’s result (2017).

To evaluate the generality of the proposed approach, we also test our approach on non-CAD models (Figs. 7-9).
These models have varieties of features and abundant details. The model Nicolo in Fig. 7 and Atenea in Fig. 9 are contaminated by synthetic noise. The above mentioned denoising techniques are tested on the noisy input models. We find that Sun et al.’s scheme (2007) over-smoothes some weak features, such as Nicolo’s hair, eyes and lip, and Zheng et al.’s local method (2011) and Wei et al.’s method (2015) filter out some details too, like Atenea’s eyes and lip. In contrast, our approach can produce the more desirable result in which geometric features at various sizes are better preserved. In addition to the models damaged by synthetic noise, we also verify the effectiveness of our approach on real scanned data, as shown in Fig. 8. It is observed that our result is similar to Zhang et al.’s result (2015) and result by Zheng et al.’s local method (2011), and obviously better than Sun et al.’s result (2007) which looks blurred.

Fig. 3. The denoising results of Dodecahedron model corrupted with Gaussian noises with $s = 0.2$ mean edge length. From left to right: Noisy model, the result by Sun et al.’s method (2007), the result by Zheng et al.’s local method (2011), the result by Zheng et al.’s global method (2011), the result by Zhang et al.’s method (2015), and our method’s result. The bottom row shows the close-ups of the red rectangular region.

Fig. 4. The denoising results of Fandisk model corrupted with Gaussian noises with $s = 0.3$ mean edge length. From left to right: Noisy model, result by Sun et al.’s method (2007), result by Zheng et al.’s local method (2011), result by Zheng et al.’s global method (2011), result by Zhang et al.’s method (2015), and our method’s result. The bottom row shows the close-ups of the red rectangular region.

Fig. 5. The denoising results of Fandisk model corrupted with Gaussian noises with $s = 0.5$ mean edge length. From left to right: Noisy model, Sun et al.’s result (2007), result by Zheng et al.’s local method (2011), result by Zheng et al.’s global method (2011), Zhang et al.’s result (2015), and our result.
Fig. 6. The denoising results of Joint model corrupted with Gaussian noises with $s = 0.3$ mean edge length. From left to right: Noisy model, result by Sun et al.’s method (2007), result by Wei et al.’s method (2015), result by Wei et al.’s method (2017), result by Zhang et al.’s method (2015), and our method’s result. The bottom row shows the close-ups of the red rectangular region.

Fig. 7. The denoising results of Nicolo model corrupted with Gaussian noises with $s = 0.2$ mean edge length. From left to right: Noisy model, Sun et al.’s result (2007), result by Zheng et al.’s local method (2011), result by Zheng et al.’s global method (2011), Zhang et al.’s result (2015), and our method’s result. The bottom row shows the close-ups of the red rectangular region.

Fig. 8. The denoising results of raw Rabbit with scanning noises. From left to right: Noisy model, Sun et al.’s result (2007), result by Zheng et al.’s local method (2011), Zhang et al.’s result (2015), and our method’s result.
Fig. 9. The denoising results of Atenea model corrupted with Gaussian noises with $s = 0.1$ mean edge length. From left to right: Noisy model, result by Zheng et al.’s local method (2011), Wei et al.’s result (2015), Wei et al.’s result (2017), and our method’s result.

Fig. 10. The denoising results of SharpSphere model corrupted with Gaussian noises with $s = 0.3$ mean edge length. From left to right: Noisy model, Wei et al.’s result (2015), Wei et al.’s result (2017), Lu et al.’s result (2017), and our method’s result.

7.2 Quantitative comparisons

To give a more objective comparison, we further evaluate the denoising results using quantitative measure. Since the signal that we process is normal, we measure the difference between the normal field of the denoised model and the ground truth. We take the mean angular error as the error metric. As shown in Table 2, our approach often performs best. When dealing with higher level noise, our approach outperforms other exemplary denoising methods, and mean angular error by our approach is significantly lower than others, which is consistent with the observation of qualitative (visual) comparison. In these experiments, when dealing with low level noise, Zhang et al.’s guided bilateral normal filtering technique (2015) performs well on CAD-type models (Fig. 1), and Zheng et al.’s local bilateral normal filtering method (2011) performs well on non-CAD models (Fig. 7). Wei et al.’s bi-normal filtering method (2015) outperforms others on noisy SharpSphere with medium level of noise (Fig. 10). However, our approach generates comparable outcomes in these cases.

Table 2 The error comparison among different methods
(For each model, the least error value is highlighted in bold.)

<table>
<thead>
<tr>
<th>Models (noise level)</th>
<th>Methods</th>
<th>Mean angular error</th>
<th>Models (noise level)</th>
<th>Methods</th>
<th>Mean angular error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block $(0.15)$ in Fig.1</td>
<td>Sun et al.’s (2007)</td>
<td>0.0457</td>
<td>Dodecahedron $(0.2)$ in Fig.3</td>
<td>Sun et al.’s (2007)</td>
<td>0.0915</td>
</tr>
<tr>
<td></td>
<td>Zheng (2011) local</td>
<td>0.0368</td>
<td></td>
<td>Zheng (2011) local</td>
<td>0.0449</td>
</tr>
<tr>
<td></td>
<td>Zheng (2011) global</td>
<td>0.0524</td>
<td></td>
<td>Zheng (2011) global</td>
<td>0.0496</td>
</tr>
<tr>
<td></td>
<td>Zhang et al.’s (2015)</td>
<td><strong>0.0365</strong></td>
<td></td>
<td>Zhang et al.’s (2015)</td>
<td>0.0232</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>0.0367</td>
<td></td>
<td>Ours</td>
<td><strong>0.0190</strong></td>
</tr>
</tbody>
</table>
### 7.3 Limitations

Our method has certain limitations in spite of its demonstrated superiority over the state of the art approaches. First, if the noise level is surprisingly high, it is possible to smooth out some features at the noise preprocessing stage, which brings challenges to the following steps, leading to less than satisfactory denoised results. Second, our method cannot handle models with extremely irregular surface sampling. Fig. 11 shows an example. When sharp edges and corners are formed by a lot of irregular triangles, our method cannot achieve a good denoised result.

![Fig. 11. Denoising a model with an extreme triangulation. From left to right: the ground truth with an extreme triangulation, the wireframe display, the noisy input, and our result. Our method fails to achieve a desired denoised result.](image)

### 8. Conclusion

In this paper, we have presented an effective collaborative filtering method for feature-preserving mesh denoising based on vertex classification. Given a noisy input mesh, we first reduce the noise by global optimization preprocessing, after which the feature vertices can be easily detected via normal voting tensor. Next, in the light of the vertex classification results, we design different filters to estimate the face normals. In the feature region, our method may avoid the arduous task of searching for an optimal set of parameters for a given model. Finally, we update the vertex positions according to the new estimated face normals. In the vertex updating formula, we use the normal difference weight to improve the accuracy. As a result, our approach can efficiently remove noises, overcome large outliers, and maintain sharp feature by taking advantage of geometric features. Experimental results show that our approach outperforms most of the existing algorithms on both CAD and non-CAD models with different intensities of noises. Especially when dealing with models with high level of noises, our approach can generate the best denoised results among the selected exemplary methods.

### Acknowledgments
We thank the anonymous reviewers for their comments and suggestions. We also thank Mingqiang Wei for providing the results of (Wei et al., 2015) and (Wei et al., 2017), and Xuequan Lu for providing the results of (Lu et al., 2017). This work is supported by the National Natural Science Foundation of China under Grant No.61472466 and No.11601115.

References


Sun, X. Rosin, P. L., Martin, R. R., and Langbein, F. C., Random walks for feature-preserving mesh denoising,