An Efficient Approach for Feature-preserving Mesh Denoising

Xuequan Lu[a], Xiaohong Liu[a], Zhigang Deng[b], Wenzhi Chen[a]

[a]*X. Lu, X. Liu and W. Chen* are with the College of Computer Science and Technology, Zhejiang University.*
[b]*Z. Deng is with the Department of Computer Science, University of Houston.*
E-mails: xuequanlu@zju.edu.cn, 21421187@zju.edu.cn, zdeng4@uh.edu, chenwz@zju.edu.cn

Abstract

With the growing availability of various optical and laser scanners, it is easy to capture different kinds of mesh models which are inevitably corrupted with noise. Although many mesh denoising methods proposed in recent years can produce encouraging results, most of them still suffer from their computational efficiencies. In this paper, we propose a highly efficient approach for mesh denoising while preserving geometric features. Specifically, our method consists of three steps: initial vertex filtering, normal estimation, and vertex update. At the initial vertex filtering step, we introduce a fast iterative vertex filter to substantially reduce noise interference. With the initially filtered mesh from the above step, we then estimate face and vertex normals: an unstandardized bilateral filter to efficiently smooth face normals, and an efficient scheme to estimate vertex normals with filtered face normals. Finally, at the vertex update step, by utilizing both the filtered face normals and estimated vertex normals obtained from the previous step, we propose a novel iterative vertex update algorithm to efficiently update vertex positions. The qualitative and quantitative comparisons show that our method can outperform selected state of the art methods, in particular, its computational efficiency (up to about 32 times faster).

Keywords: Efficient mesh denoising, Feature preserving

1. Introduction

Mesh denoising, which aims at achieving clean and quality results from input noisy meshes, has recently attracted increasing interests in graphics and engineering community. With the growing availability of various optical and laser scanners, it is easy to capture different kinds of shapes. However, the captured data are always accompanied with noise. Even using high-fidelity laser scanners, the scanned models are inevitably polluted with noise, to certain extent. As a routine step, 3D scanned models need to be processed by mesh denoising before they are further applied to various applications, including computer-aided industrial design, reverse engineering, animation, rendering, prototyping and so on.

Promising mesh denoising results have been demonstrated recently [1, 2, 3, 4, 5]. However, these methods mainly focus on preserving geometric features, without taking the computational efficiency into consideration. Specifically, in terms of the computational efficiency, the scheme in [1] is probably the most efficient among all the methods, since it only includes bilateral filtering and vertex update, both of which are iterative and fast. The L_0-minimization method [2] is complex, thus leading to a slow speed. The method [3] is quite slow due to the involved large matrix manipulation. The method proposed by Wei et al. [4], which introduces additional steps, is a variant of [1]. On the one hand, the neighboring faces clustering is inefficient since it is applied to all classified feature vertices. Moreover, the clustering would sometimes be inaccurate, especially when the noise level is high. On the other hand, the face normal filter and vertex update are formulated as least-squares problems. Furthermore, the least-squares problem of vertex update needs to be solved multiple times. The scheme in [5] consists of three stages, among which the initial estimation stage accounts for considerable computation due to the iterative least-squares optimization. Hence, it is less efficient from the computational perspective.

To address the above efficiency issue, we propose a highly efficient approach for mesh denoising while preserving geometric features. Specifically, our method consists of three steps: initial vertex filtering, normal estimation, and vertex update. An example for demonstrating our mesh denoising approach is shown in Figure 2. Figure 1 shows that our method outperforms the selected state of the art methods in terms of preserving eyelids (refer to zoomed regions).

Initial vertex filtering. Since positional and normal information are noisy and cluttered in the noisy input, we present an initial vertex filter to generate a sound initialized mesh for the follow-up steps. The proposed initial vertex filter is local, iterative and fast.

Normal estimation. As reported in the work [4], considering only face normals would overlook some geometric information, we thus take both face and vertex normal information into account. We introduce an unstandardized bilateral filter (§4.1) to efficiently smooth face normals of the initialized mesh obtained above. We can adjust the employed parameters to achieve a desired face normal field. Then we present an efficient and effective algorithm for vertex normal estimation (§4.2). Specifically, vertices are first classified into two types: corners and non-corners. Then, we estimate vertex normals according to the corresponding vertex types.

Vertex update. By taking advantage of both the filtered face normals and estimated vertex normals from the above step, we propose a novel iterative vertex update algorithm to efficiently update vertex positions (§5).

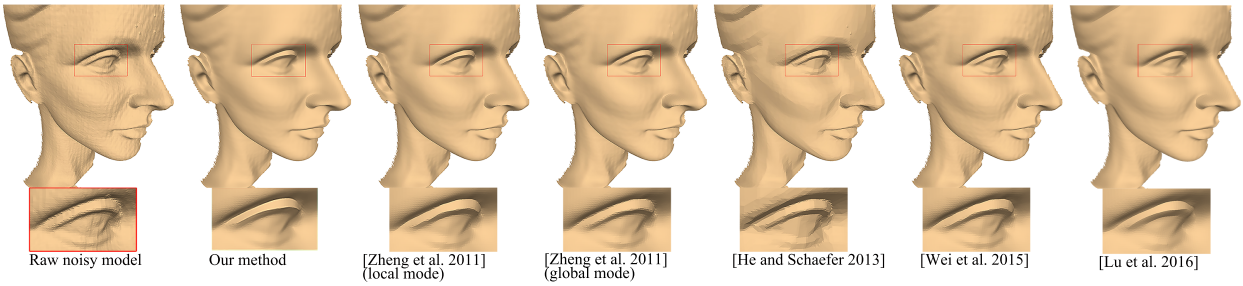


Figure 1: The denoised results of the Wilhelm model (comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

The specific contributions of this work include:

- **Initial vertex filtering.** Previous preprocessing [5] is time-consuming due to the global optimization. Thus we formulate this problem in a local sense and derive a fast initial vertex filter, which has similar performance to [5].
- **Normal estimation.** Bilateral filtering is fast [1]; however, the division computation consumes a large portion of the total computation. Therefore, we introduce an unstandardized bilateral filter to generate similar smoothed normals to the original version, with a higher efficiency. Previously, the vertex normals are estimated using an area-weighted average of neighboring face normals for non-feature vertices, and the average of representative normals of clustered face groups for feature vertices [4]. In this work, we present a much more efficient and accurate algorithm for vertex normal estimation.
- **Vertex update.** The vertex update in the work of [4] is time-consuming because of multiple executions of the least-squares optimization. Therefore, we present a fast iterative vertex update method by utilizing both the filtered face normals and estimated vertex normals in a local sense.

2. Related Work

The early methods for mesh smoothing are isotropic, which may wipe away high-frequency features as the filters applied to these methods are independent of surface geometry. A typical surface smoothing method is Laplacian smoothing [6], but it shrinks the surface and cannot preserve features. Taubin [7] proposed a two-step smoothing method which prevents features from shrinkage by expanding the meshes after smoothing. Desbrun et al. [8] used a fairing method based on curvature flow to extend meshes to irregular meshes. Other isotropic approaches including the low-pass/high-pass filtering framework with exaggeration and attenuation options, the volume preservation approach for triangle meshes, and the methods using differential properties were also proposed [9, 10, 11, 12, 13].

Many researchers have paid attentions to anisotropic approaches because such methods can preserve features of input meshes. Different diffusion-based methods were introduced [14, 15, 16, 17, 18, 19] to preserve or even

sharpen geometric features during denoising. Hildebrandt et al. [20] introduced mean curvature flows to preserve features. Inspired by the bilateral filter [21], a vertex-based bilateral filter was presented [22]. Tsuchie and Higashi [23] developed a novel method for feature-preserving mesh denoising based on the normal tensor framework. He and Schaefer [2] presented a L_0-optimization framework to minimize the curvatures of a surface except at sharp features. Other mesh denoising methods were also proposed [3, 5].

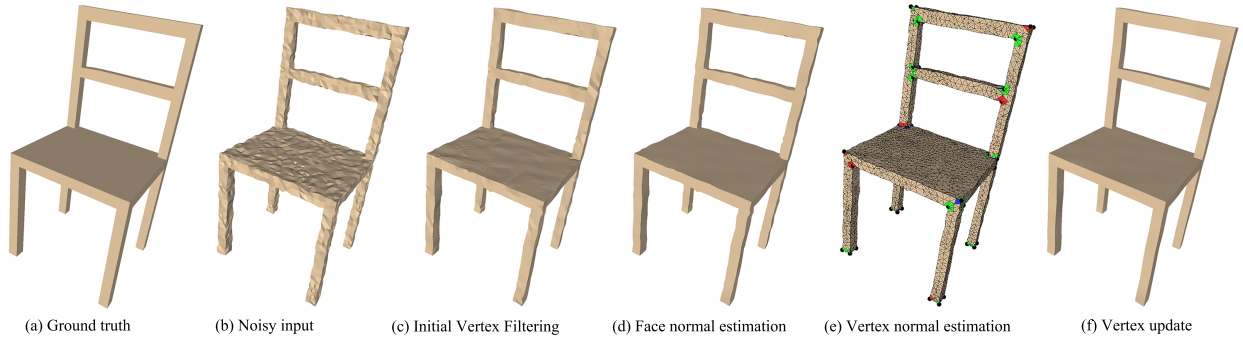


Figure 2: An example for demonstrating the pipeline of our mesh denoising approach. (a) The ground truth model. (b) The noisy model. (c) The model after initial vertex filtering. (d) The model after face normal filtering. (e) The wireframe display after vertex normal estimation. The black points denote corners, and the triangles with same color belong to the same patch (please zoom in to clearly see them). (f) The result after vertex update based on both the filtered face normals and estimated vertex normals.

Instead of directly filtering vertex coordinates, the idea of first filtering facet normals followed by vertex update has attracted lots of attentions in recent years [24, 25, 26, 27, 28, 29, 30, 1, 31, 32]. Taubin [24] and Ohtake et al. [25] introduced early two-stage methods. Yagou et al. [26, 27] proposed mean, median, and alpha-trimming filters to filter face normals. Shen et al. [28] introduced a fuzzy vector median filter. Sun et al. [29, 30] presented two different methods both by first smoothing normals and then updating vertices. More recently, three different normal filters were proposed [1, 31, 32], followed by updating vertex positions using the algorithm in the work of [29].

To better preserve geometric features, several mesh denoising techniques [33, 34, 35, 4] employ vertex/face classification before the denoising process. However, in general the classification results are sensitive to the level of noise.

3. Initial Vertex Filtering

As suggested by latest mesh denoising work[5], the original noisy input should be preprocessed, to generate a sound initialized mesh by removing folded faces, degenerate triangles and so on. Inspired by the preprocessing notion, we also present a preprocessing method — initial vertex filtering. Due to the inefficiency of the preprocessing step in [5], we attempt to achieve a *locally* fast algorithm for initial vertex filtering.

Problem formulation. Different from the global formulation [5], we define the initial vertex filtering problem for the i-th vertex in a local sense.

$$\min \sum_{e \in NE(i)} w(e) \|D(e)\|_2^2 + \alpha \sum_{e \in NE(i)} w(e) \|R(e)\|_2^2, \tag{1}$$

where e denotes edge and $NE(i)$ is the neighboring edges of the i-th vertex. We use $w(e)$ introduced in [5]: $w(e) = e^{-\left(\frac{\beta}{\sigma_\beta}\right)^2}$. β is the angle between two edge-sharing face normals, and σ_β is the angle threshold parameter. α is the weighting parameter, balancing the importance between the two terms. $D(e)$ and $R(e)$ are the area-based edge operator (to measure the edge sharpness) and the regularizer (to eliminate spurious overshoots and fold-backs), respectively [2].

$$D(e) = \begin{bmatrix} \frac{\Delta_{123}((p_4-p_3)\cdot(p_3-p_1))+\Delta_{134}((p_1-p_3)\cdot(p_3-p_2))}{\|p_3-p_1\|^2(\Delta_{123}+\Delta_{134})} \\ \frac{\Delta_{134}}{\Delta_{123}+\Delta_{134}} \\ \frac{\Delta_{123}((p_3-p_1)\cdot(p_1-p_4))+\Delta_{134}((p_2-p_1)\cdot(p_1-p_3))}{\|p_3-p_1\|^2(\Delta_{123}+\Delta_{134})} \\ \frac{\Delta_{123}}{\Delta_{123}+\Delta_{134}} \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix},$$

where Δ_{123} denotes the area of the triangle formed by p_1, p_2 and p_3, and Δ_{134} is the area of the triangle formed by p_1, p_3 and p_4.

$$R(e) = p_1 + p_3 - p_2 - p_4$$

Figure 3 illustrates the notion of neighboring edges of p_1 and the edge structure formed by p_1, p_2, p_3 and p_4. Specifically, the neighboring edges of a vertex described in this work are the union set of all edges of triangles around this vertex (without repeated edges). Each edge owns an edge structure which consists of two edge-sharing faces, i.e., four vertices in total.

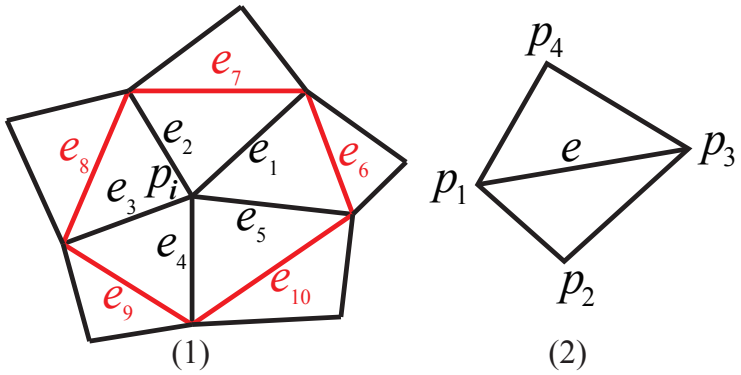


Figure 3: (1) the neighboring edges (from e_1 to e_{10}) of a vertex p_i and (2) the structure of an edge e.

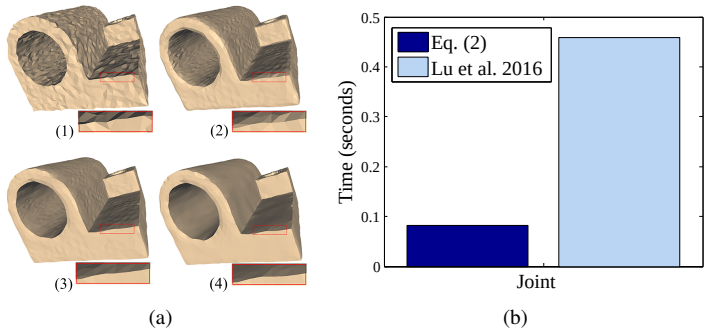


Figure 4: (a) Comparisons among the initial vertex filtering with and without $w(e)$, as well as the preprocessing step in [5]. (b) Efficiency comparison between Eq. (2) and the first stage of [5]. Specifically, (1) noisy model, (2) the initially filtered result without $w(e)$, (3) the initially filtered result with $w(e)$, (4) the initial estimation of [5].

Taking the derivative of Eq. (1) with respect to p_i (position of the i-th vertex) and equating it to 0, we obtain a local initial vertex filter, as follows.

$$p_i^{t+1} = \sum_{j \in NV(i)} b_j p_j^t, \tag{2}$$

where p_i^{t+1} is the p_i at the $(t+1)$-th iteration. $NV(i)$ is the set of neighboring vertices of the i-th vertex, and the neighboring vertices here, i.e., p_j^t, are the vertices (excluding p_i^t) of the neighboring edge structures at the t-th iteration. b_j, the coefficients of neighboring vertices, can be easily computed, given a vertex and its neighboring edges and σ_β. Interested readers can refer to the supplemental document for more details on how to derive Eq. (2) from Eq. (1).

To produce a good initialized mesh from the original noisy model, Eq. (2) should be iteratively called. Since the face normals of the original noisy input are cluttered and unreliable, at the first iteration, we call the unweighted version (i.e., without $w(e)$). Figure 4 shows that our iterative initial vertex filter can also generate a sound initialized mesh while preserving features. More importantly, our initial vertex filter is much more efficient (around 6 times faster) than theirs [5].

4. Normal Estimation

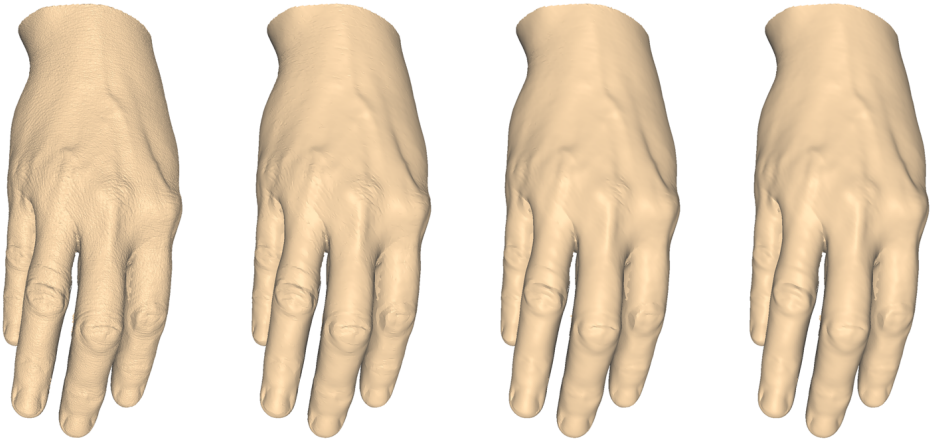
In this section, we start with the face normal filtering since vertex normals are less accurate and more difficult to deal with. Then we estimate vertex normals with the filtered face normals.

4.1. Face Normal Filtering

The traditional bilateral filtering form [1] is shown as follows.

$$n_i^{F'} = \frac{\sum_{j \in NF(i)} a_j \phi(\theta_{ij}) \psi(\| c_i - c_j \|) n_j^F}{\sum_{j \in NF(i)} a_j \phi(\theta_{ij}) \psi(\| c_i - c_j \|)}, \tag{3}$$

where $n_i^{F'}$ is the new (to be solved) normal for the i-th face, n_j^F is the original (known) normal for the j-th face, $NF(i)$ is the one-ring neighboring faces of the i-th face, and a_j is the area of the j-th face, accounting for the influence from surface sampling rate. θ_{ij} denotes the angle between the face normals of the i-th and j-th faces, c_i and c_j denotes the centroids of the i-th and j-th faces, $\| c_i - c_j \|$ represents the Euclidean distance between c_i and c_j. $\phi(\theta_{ij})$ and $\psi(\| c_i - c_j \|)$ are the normal weight function and spatial weight function, respectively.



(a) Raw noisy model (b) Denoised result (c) Denoised result (d) Denoised result

Figure 5: Different σ_θ used in face normal filtering (given the same raw input). (a) Raw noisy model. (b) $n_2 = 5$, $\sigma_\theta = 10$, $n_3 = 30$. (c) $n_2 = 5$, $\sigma_\theta = 15$, $n_3 = 30$. (d) $n_2 = 5$, $\sigma_\theta = 20$, $n_3 = 30$.

When the value of θ_{ij} is large, it means the normals of these two neighboring faces differ greatly. This typically implies that the two faces lie on two different sides of a sharp edge. In this work, we take the angle θ_{ij} instead of $\| n_i - n_j \|$ in [1] as the variable of the normal weight function $\phi(\theta_{ij})$, as suggested in [36].

$$\phi(\theta) = e^{-\left(\frac{1-\cos(\theta)}{1-\cos(\sigma_\theta)}\right)^2},$$

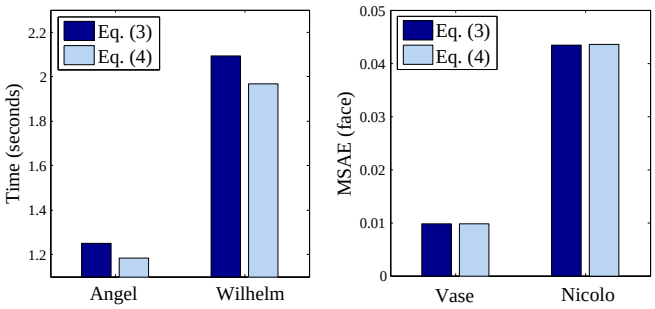
where σ_θ is the angle threshold parameter, which should be tuned by users, a greater σ_θ will lead to a smoother mesh, as shown in Figure 5. Similar to [1], the spatial weighting term is defined as follows:

$$\psi(\theta) = e^{-\left(\frac{x}{\sigma_c}\right)^2},$$

where σ_c is the parameter to scale the spatial distance of pairwise faces. We empirically set $\sigma_c = \frac{3}{2}d$, where d is the average distance between the centroid of the i-th face and those of its neighboring faces.

Despite the bilateral filter in Eq. (3) is iterative and fast, we found that the speed can be further improved by introducing an unstandardized bilateral filter. This unstandardized filter is more efficient than the traditional version, since the division computation is not involved, shown as follows.

$$n_i^{F'} = \sum_{j \in NF(i)} a_j \phi(\theta_{ij}) \psi(\| c_i - c_j \|) n_j^F \tag{4}$$



(a) Efficiency (b) Error

Figure 6: The efficiency and error comparisons between the unstandardized bilateral filter (Eq. (4)) and the original bilateral filter (Eq. (3)), with the same input and parameters. MSAE is the abbreviation of mean square angular error [1, 4].

Our unstandardized bilateral filter smoothes the face normals of the initialized mesh in the following way: it makes isotropic normals (i.e., on the same piecewise surface) to be close and similar, while preserving anisotropic features. There are two major reasons to employ Eq. (4) instead of Eq. (3): (i) the introduced unstandardized filter is more concise and efficient (Figure 6(a)); (ii) Eq. (4) can generate highly approximate filtered results to Eq. (3) (Figure 6(b)).

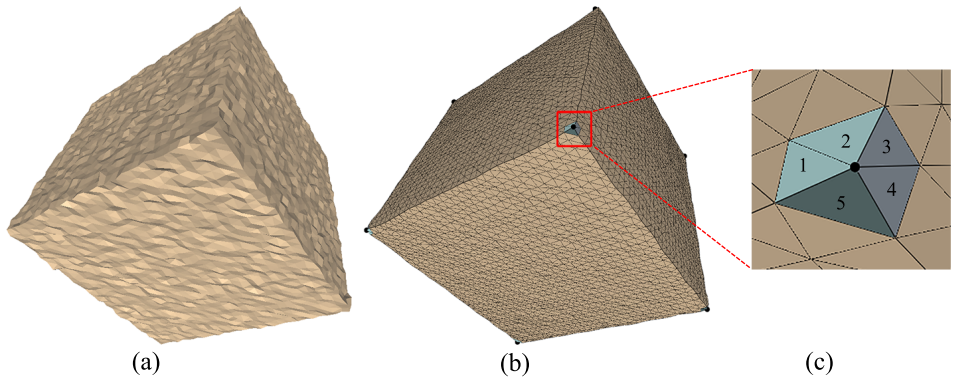
Our proposed unstandardized bilateral filter is also iterative, similar to the traditional bilateral filter [1]. The number of iterations can be tuned by users, to achieve a desired filtered result for face normals. However, as with the traditional bilateral filter [1], the unstandardized bilateral filter cannot either guarantee convergence.

4.2. Vertex Normal Estimation

After filtering face normals, we present an efficient and accurate algorithm to estimate vertex normals. Specifically, we first classify vertices into corners and non-corners, and then we estimate vertex normals in accordance with the classified vertex types.

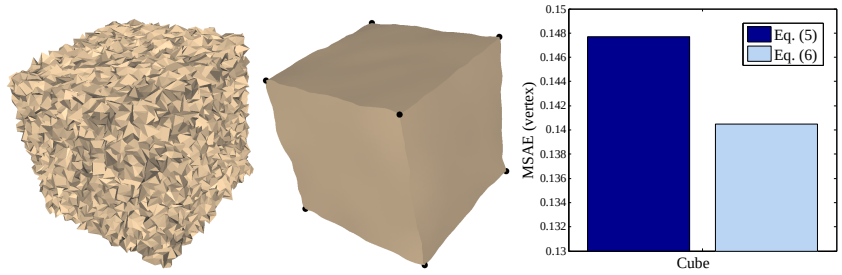
We employ a tensor voting technique [37] to classify vertices based on the filtered face normals. Compared to directly classifying vertices on the initialized mesh from the first step, the classification results after face normal filtering are more accurate, as normal noise interference can be significantly lowered by the above step. According to the eigenvalues of the voting tensor, vertices can be classified into three types: corners, sharp edges and faces. In this work, vertices of sharp edges and faces are summarized to a single type—*non-corners*.

Then, we estimate vertex normals based on the classified vertex types, as described below.



(a) (b) (c)

Figure 7: An example for illustrating piecewise smooth patches around a corner. (a) The raw noisy model. (b) The wireframe display after initial vertex filtering and face normal filtering. (c) The zoomed region of the corner in (b): the triangles 1 and 2 are in the same patch (the same color), the triangles 3 and 4 are in a separate patch, and the triangle 5 is in the same patch.



(a) (b) (c)

Figure 8: The comparison of corner normal errors between Eq. (5) and Eq. (6). (a) noisy input, (b) corner normal estimation after initial vertex filtering and face normal filtering, (c) corner normal errors.

Non-corners. For non-corners, we compute vertex normals by normalizing the angle weighted average of their 1-ring neighboring face normals, inspired by the work [38].

$$n_i^V = \urcorner\left(\sum_{j \in VNF(i)} \delta_j n_j^F\right), \tag{5}$$

where n_i^V represents the vertex normal of the i-th vertex. n_j^F represents the face normal of the j-th face. δ_j is the angle of the j-th face sharing the i-th vertex. $VNF(i)$ is the 1-ring neighboring faces of the i-th vertex. $\urcorner$ is the normalization operator.

Corners. We aim to understand the underlying piecewise smooth patches around a corner, in order to accurately calculate vertex normals. We adopt the following strategy to divide the neighboring faces of corners into three or more piecewise smooth patches: first, the order of these neighboring faces are sorted clockwise or counterclockwise; then, if the angle of the adjacent face normals is greater than the clustering angle threshold (15° by default), it denotes these two faces are not on the same piecewise smooth patch, and should add one new cluster; by repeating the comparisons of such angles until the starting face is reached, we can obtain a few patches which contain one or more piecewise smooth faces. An example is shown in Figure 7.

The vertex normal of a corner vertex is estimated by normalizing the average of the representative normals of all clustered patches around it, as described below.

$$n_i^V = \urcorner\left(\sum_{k\in NUM(i)} \urcorner\left(\sum_{j\in VNF_k(i)} n_j^F\right)\right), \tag{6}$$

where $NUM(i)$ denotes the set of clustered piecewise smooth patches around the i-th vertex, and $VNF_k(i)$ represents the k-th piecewise smooth patch consisting of neighboring faces for the i-th vertex. Figure 8 shows Eq. (6) is more accurate than Eq. (5) when estimating corner normals.

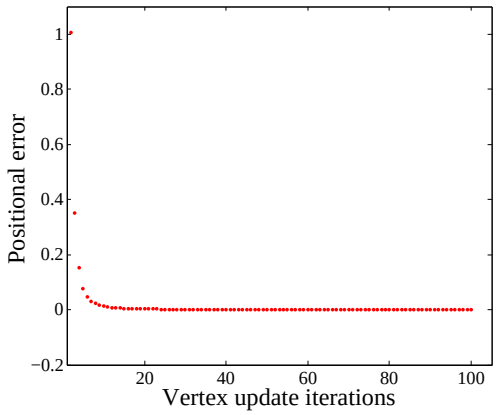


Figure 9: The positional errors ($\sum_i \left\|p_i^{t+1} - p_i^t\right\|^2$) with growing vertex update iterations. The plotted numbers are scaled by 10^3.

5. Vertex Update

Inspired by Wei et al. [4], we define the vertex update problem by taking advantage of both the filtered face normals and estimated vertex normals, as follows.

$$\min \sum_{k\in VNF(i)} \left(n_k^F \cdot (c_k - \tilde{p}_i)\right)^2 - \left(n_i^V \cdot (\tilde{p}_i - p_i)\right)^2, \tag{7}$$

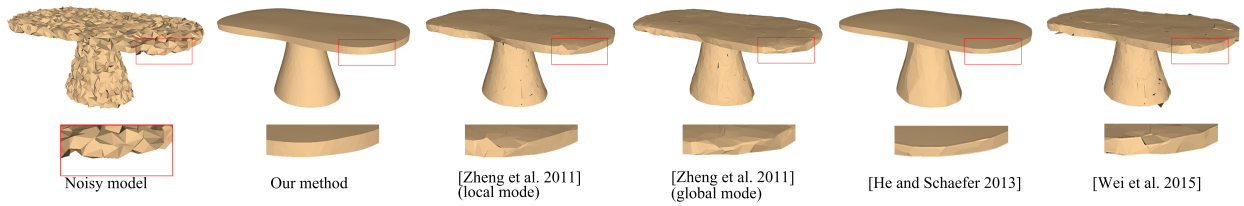


Figure 10: Denoised results of the Table model ($\sigma = 0.3\ell$, comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

where $VNF(i)$ denotes the neighboring faces of the i-th vertex, and c_k is the centroid of a face in $VNF(i)$. $\tilde{p}_i$ (unknown) and p_i (known) are the positions of the i-th vertex, respectively. $n_k^F \cdot (c_k - \tilde{p}_i)$ describes the projection from $(c_k - \tilde{p}_i)$ to n_k^F. The first term encodes the sum of squared projection distances to n_k^F. $n_i^V \cdot (\tilde{p}_i - p_i)$ is the projection from $(\tilde{p}_i - p_i)$ to n_i^V. The second term describes the squared projection distance to n_i^V.

We can achieve an iterative update algorithm from Eq. (7), by employing the gradient descent algorithm. It is fast and effective, and shown as follows.

$$p_i^{t+2} = p_i^{t+1} + \frac{1}{|\ VNF(i)\ |}\left(\sum_{k\in VNF(i)} n_k^F\left(n_k^F \cdot (c_k^{t+1} - p_i^{t+1})\right) + n_i^V\left(n_i^V \cdot (p_i^{t+1} - p_i^t)\right)\right), \tag{8}$$

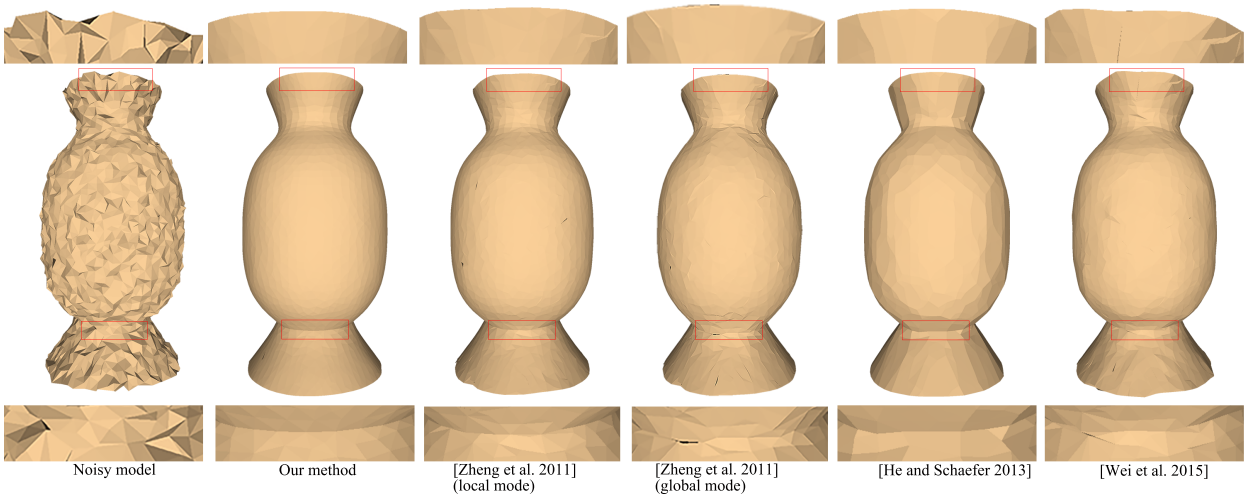


Figure 11: Denoised results of the Vase model ($\sigma = 0.2\ell$, comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

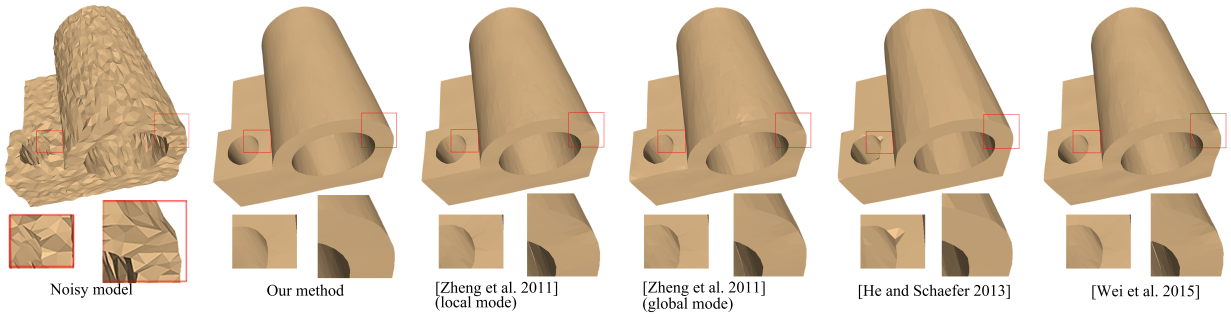


Figure 12: Denoised results of the Joint model ($\sigma = 0.1\ell$, comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

where p_i^t denotes p_i in the t-th iteration, p_i^0 denotes the initial position of the i-th vertex.
At the first iteration, we use the following expression:

$$p_i^1 = p_i^0 + \frac{1}{|\ VNF(i)\ |} \sum_{k \in VNF(i)} n_k^F \left(n_k^F \cdot \left(c_k^0 - p_i^0 \right) \right), \tag{9}$$

The number of iterations, which should be increased with the growing noise level of the input mesh, can be practically controlled by users. Our experiments (Figure 9) show the convergence of the proposed vertex update algorithm.

6. Results and Discussion

We tested our approach on a variety of mesh models corrupted with either synthetic or raw noise. Synthetic noise is generated by a zero-mean Gaussian function with standard deviation σ proportional to the mean edge length ℓ of the input mesh. Meanwhile, we also tested state-of-the-art methods on the same mesh models as comparisons. For implementation, we used the source code provided by Zheng et al. [1], and implemented the other methods [2, 4] by strictly following their algorithms. Besides, Zhang et al. [31] and Lu et al. [5] provided some denoised results, respectively.

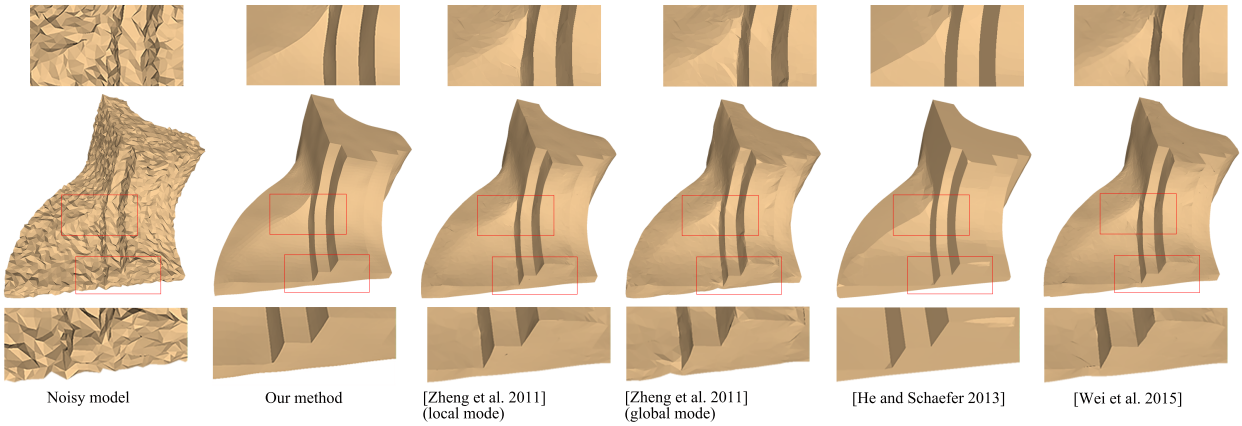


Figure 13: Denoised results of the Fandisk model ($\sigma = 0.2\ell$, comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

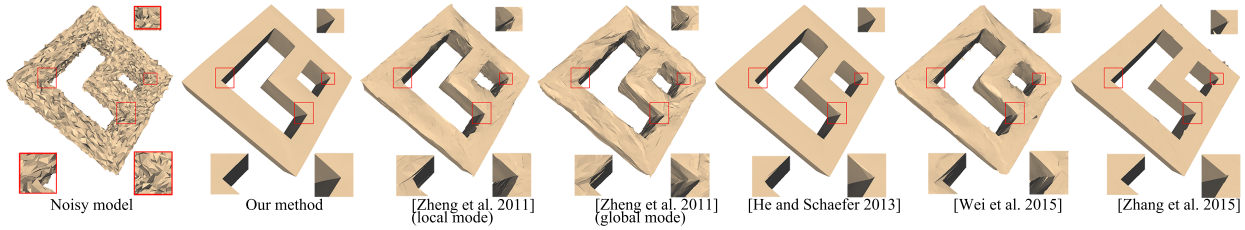


Figure 14: Denoised results of the Double-torus model contaminated with large noise ($\sigma = 0.4\ell$, comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

For the sake of simplicity and clearness, we use *Zheng et al. 2011*, *He et al. 2013* and *Wei et al. 2015* to denote the method [1], the method [2] and the method [4], respectively. Since Zheng et al. 2011 has two versions of solutions, we use Zheng et al. 2011 (local) and Zheng et al. 2011 (global) to denote the local and global versions, respectively.

6.1. Parameter Sets and Selection

Different mesh denoising methods have different sets of parameters. Concisely, we used the following parameter sets for the above approaches: Our method=(initial vertex filtering iterations (n_1), σ_β, α, normal filtering iterations (n_2), σ_θ, vertex update iterations (n_3)); Zheng et al. 2011 (local)=(normal filtering iterations, σ_s, vertex update iterations); Zheng et al. 2011 (global) = (λ, σ_s, vertex update iterations); He et al. 2013 = (μ, α_0, λ); Wei et al. 2015 = (σ_{s1}, n_1, σ_{s2}, n_2).

As suggested by the original authors [2] for He et al. 2013, we use the default parameter values ($\sqrt{2}, 0.1\overline{\gamma}, 0.01\ell^2\overline{\gamma}$) - abbreviated as *Default*. The parameter values used in the other methods are elaborately tuned to achieve best visual results for all the test models. Specifically, in our method, the initial vertex filtering iterations (n_1) is between 2 and 10; α is in the range of [0.1, 1.1]; σ_β is restricted in the range of [20.0, 50.0] and a smaller value will produce sharper initializations; the number of normal filtering iterations (n_2) is in the range of [5, 30] in our experiments, and a larger value will result in smoother face normals; σ_θ is in the range of [15.0, 40.0], with a larger value for a higher level of noise; the number of vertex update iterations (n_3) is in the range of [10, 100] and a greater number will lead to smoother results. In Zheng et al. 2011 (local), the number of normal filtering iterations is in the range of [5, 40], with more iterations for larger noise; σ_s is suggested to be in the range of [0.2, 0.5]; the number of vertex update iterations is in the range of [10, 100], with a greater value for a higher level of noise. In Zheng et al. 2011 (global), λ is in the range of [0.01, 0.1], and a smaller value leads to smoother results; σ_s and vertex update iterations are in the same range as Zheng et al. 2011 (local). In Wei et al. 2015, both σ_{s1} and σ_{s2} are in the range of [0.20, 0.65], with larger values for heavier noise; n_1 is in the range of [3, 20], with greater numbers for smoother normal initialization; n_2 is in

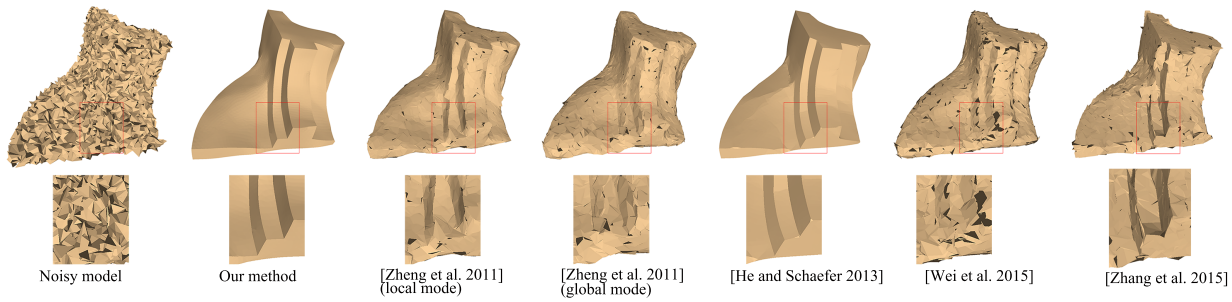


Figure 15: Denoised results of the Fandisk model corrupted with large noise ($\sigma = 0.6\ell$, comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

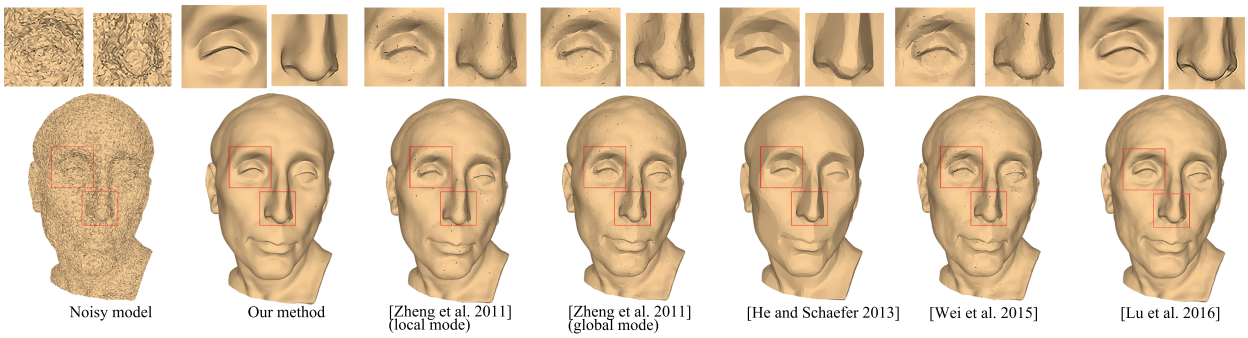


Figure 16: Denoised results of the Nicolo model ($\sigma = 0.3\ell$, comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

the range of [4, 40], with greater numbers for larger noise. For fair comparisons, the normal filtering iterations in our method and Zheng et al. 2011 (local) are set to be equivalent; the vertex update iterations are set to be equal in our method and Zheng et al. 2011.

6.2. Test Models

Models with synthetic noise. From Figures 10 to 16, we observe that our approach can generate better results than the selected state of the art methods, especially when denoising models with relatively large noise ($\sigma >= 0.3\ell$). The original inputs (Figures 13, 11, and 12) are corrupted with comparatively small noise ($\sigma <= 0.2\ell$), and the denoised results by our method have higher-quality than other selected methods. Our approach can still generate best denoised results among all methods even when encountering inputs contaminated with large noise ($\sigma >= 0.3\ell$), for example, Figures 14, 15, 10 and 16.

Models with raw noise. In addition to the models corrupted with synthetic noise, we also tested all the methods on real 3D scanned data. For the Angel Model (Figure 17), the result by our method is noticeably better than all the other methods, in terms of details and features preservation (refer to the zoomed regions). The Wilhelm Model (Figure 1) shows that our approach can generate a better result than the state of the art methods in terms of preserving features. Figure 18 shows that our method can generate a desired denoised result. The raw scanned cube model (Figure 19) also demonstrates the superiority of our method over the state of the art methods.

6.3. Quantitative Comparisons

Besides the above visual qualitative comparisons, we also compare these methods in a quantitative way, as shown in Table 1. Also, we recorded the computational time for all the approaches on some test models (as shown in Table 2). Table 1 also shows the used parameter values of all the methods on some test 3D models.

Table 1: Quantitative comparisons among all the four mesh denoising methods. MSAE (face) and MSAE (vertex) are both with 10^{-1}. E_v is with 10^{-3}.

Models	Methods	MSAE (face)	MSAE (vertex)	E_v	Parameters
Double-torus	Our method	0.274	**1.353**	**7.797**	(6, 30, 1.1, 30, 35, 100)
(Fig. 14)	Zheng et al. 2011(local)	14.284	-	9.52	(40, 0.35, 100)
$\mid V \mid$:10750	Zheng et al. 2011(global)	11.998	-	9.576	(0.01, 0.35, 100)
$\mid F \mid$:21504	He et al. 2013	**0.253**	-	8.46	Default
	Wei et al. 2015	15.423	9.243	10.402	(0.45, 15, 0.35, 40)
Fandisk	Our method	**0.038**	**0.469**	8.527	(3, 35, 0.3, 5, 25, 30)
(Fig. 13)	Zheng et al. 2011(local)	1.087	-	**8.115**	(10, 0.35, 30)
$\mid V \mid$:6475	Zheng et al. 2011(global)	0.948	-	8.116	(0.03, 0.35, 30)
$\mid F \mid$:13946	He et al. 2013	0.065	-	8.925	Default
	Wei et al. 2015	1.078	2.324	8.118	(0.35, 5, 0.35, 12)
Nicolo	Our method	0.434	**1.651**	**2.858**	(10, 30, 0.1, 5, 25, 30)
(Fig. 16)	Zheng et al. 2011(local)	7.075	-	3.161	(15, 0.4, 30)
$\mid V \mid$:49886	Zheng et al. 2011(global)	6.359	-	3.149	(0.02, 0.4, 30)
$\mid F \mid$:98971	He et al. 2013	**0.356**	-	3.149	Default
	Wei et al. 2015	6.514	5.881	3.179	(0.35, 10, 0.4, 12)
Table	Our method	0.376	**1.429**	8.028	(5, 35, 0.3, 15, 30, 30)
(Fig. 10)	Zheng et al. 2011(local)	4.933	-	7.135	(30, 0.35, 30)
$\mid V \mid$:4556	Zheng et al. 2011(global)	4.603	-	**6.698**	(0.01, 0.35, 30)
$\mid F \mid$:9108	He et al. 2013	**0.188**	-	8.3	Default
	Wei et al. 2015	5.484	5.367	8.289	(0.3, 20, 0.3, 5)
Vase	Our method	**0.098**	**0.712**	0.122	(5, 30, 0.3, 5, 30, 30)
(Fig. 11)	Zheng et al. 2011(local)	1.617	-	**0.109**	(10, 0.35, 30)
$\mid V \mid$:3827	Zheng et al. 2011(global)	1.487	-	0.11	(0.01, 0.35, 30)
$\mid F \mid$:7650	He et al. 2013	0.125	-	0.125	Default
	Wei et al. 2015	1.678	2.944	0.11	(0.35, 3, 0.35, 12)
Fandisk	Our method	**0.428**	**1.505**	**15.196**	(4, 50, 0.4, 20, 25, 30)
(Fig. 15)	Zheng et al. 2011(local)	15.997	-	18.533	(20, 0.5, 30)
$\mid V \mid$:6475	Zheng et al. 2011(global)	17.770	-	19.362	(0.01, 0.4, 30)
$\mid F \mid$:13946	He et al. 2013	0.512	-	16.527	Default
	Wei et al. 2015	21.090	11.085	20.006	(0.65, 20, 0.65, 10)

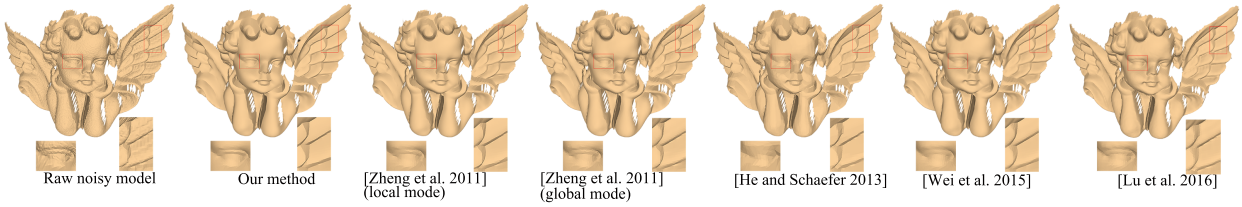


Figure 17: Denoised results of the raw scanned Angel model (comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

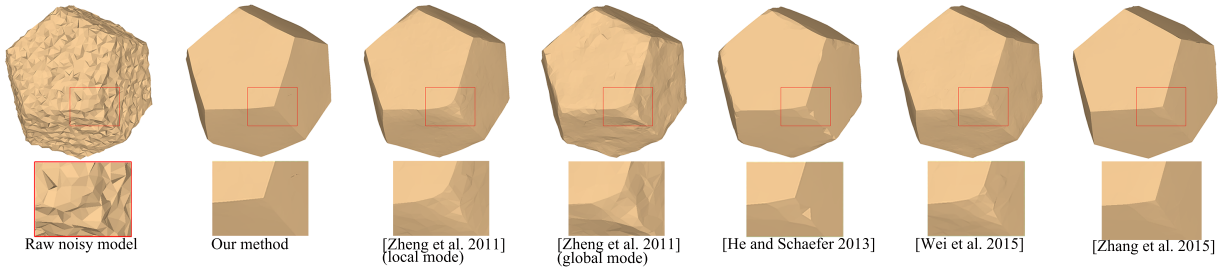


Figure 18: Denoised results of the raw scanned Dodecahedron model (comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

As suggested by [1, 4], We choose E_v and MSAE as the quantitative metrics. E_v is the L^2 vertex-based error to measure the positional error between the ground-truth mesh and the denoised mesh. MSAE (face) is to estimate the mean square angular error between the face normals of the ground-truth model and those of the denoised model. MSAE (vertex) is to estimate the mean square angular error between the vertex normals of the ground-truth model and those of the denoised model.

We can see from Table 1 that: for MSAE (face), our approach and He et al. 2013 [2] obtain similar results which are significantly smaller than other methods, as these two methods can remove folded faces. For MSAE (vertex), we only compare our method with Wei et al. 2015 [4] because only these two methods involve vertex normals. Our vertex normal estimation is more accurate than Wei et al. 2015 [4] since folded faces have been largely removed by the initial vertex filtering and face normals filtered by the second step are accurate and the vertex estimation algorithm is effective. For E_v, our method achieves mixed comparative results: for Double-torus, Nicolo and Fandisk (Figure 15), our approach generates the smallest E_v errors; for other models in Table 1, our method is not the best. The reason would be that for some models especially models corrupted with large noise, our method could produce a more similar vertex position structure to the ground truth, with the assistance of initial vertex filtering. However, for some models, our approach would generate a less similar structure to the ground truth, as the initial vertex filtering might relocate vertex positions greatly by removing folded faces.

As shown in Table 2, it is clear that our method is the fastest among all the methods: up to about 32 times faster than He et al. 2013 [2]. Specifically, our method is roughly 2 times faster than Zheng et al. 2011 (local), about 2–17 times faster than Zheng et al. 2011 (global), about 5–33 times faster than He et al. 2013, and around 2–5 times faster than Wei et al. 2015. The total runtime increases when the numbers of vertices and faces of the input mesh become larger. In general, our approach takes less than one second on an off-the-shelf computer to process a noisy mesh with about 44,000 vertices and 86,000 faces.

6.4. Limitations

Despite the proven efficiency and effectiveness of our approach, it still has the following two limitations.

- As with He et al. 2013 and Wei et al. 2015 [2, 4], our approach cannot well handle noisy models with an extreme triangulation, since vertices only exist at sharp features. Please refer to [2, 4] for illustration examples.

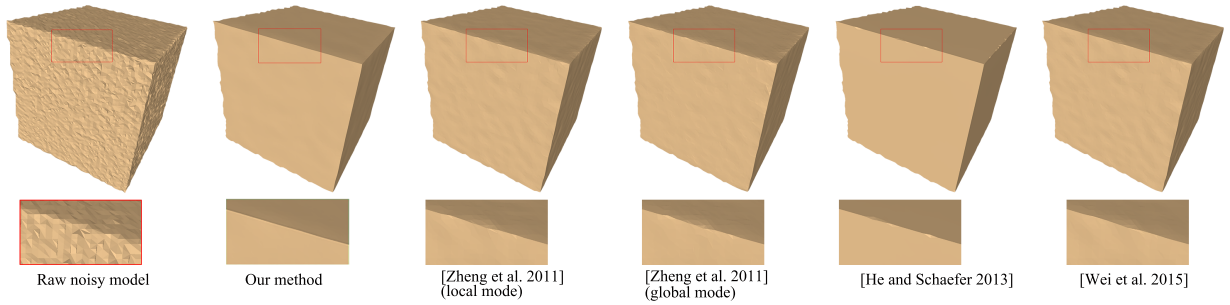


Figure 19: Denoised results of the raw scanned Cube model (comparison among our method and the selected state of the art methods). Please refer to the zoomed regions.

Table 2: Timing statistics for our method and the selected methods. The running time (seconds) was recorded on the same experimental computer with an Intel(R) Core(TM) i7-3770 CPU 3.40-GHz. 2.19X indicates our method is 2.19 times faster than Zheng et al. 2011 (local).

Models	Our method	Zheng et al. 2011 (local)	Zheng et al. 2011 (global)	He et al. 2013	Wei et al. 2015
Double-torus(Fig. 14)					
$\|V\|$:10750	1.106	2.427	3.497	6.123	2.611
$\|F\|$:21504		2.19X	3.16X	5.54X	2.36X
Fandisk(Figure 15)					
$\|V\|$:6475	0.365	0.684	1.235	3.330	1.444
$\|F\|$:13946		1.87X	3.38X	9.12X	3.96X
Vase(Fig. 11)					
$\|V\|$:3827	0.141	0.265	0.679	1.960	0.639
$\|F\|$:7650		1.88X	4.82X	13.90X	4.53X
Wilhelm(Fig. 1)					
$\|V\|$:43644	0.701	1.297	11.452	22.653	2.920
$\|F\|$:85553		1.85X	16.34X	32.32X	4.17X
Dodecahedron(Fig. 18)					
$\|V\|$:4916	0.468	0.975	1.179	2.590	0.945
$\|F\|$:9828		2.08X	2.52X	5.53X	2.02X

- Given a noisy mesh model, it is still challenging to find an optimal set of parameters for our method, similar to many other mesh denoising works (e.g., [1, 4]).

7. Conclusion

In this paper, we present a highly efficient approach for feature-preserving mesh denoising. Given a noisy mesh input, our method initially filters vertices with a fast local filter, then efficiently filters face normals with the introduced unstandardized bilateral filter, and estimates vertex normals using an efficient algorithm, and finally updates vertex positions by utilizing both the filtered face normals and estimated vertex normals. Through many experiments on various test models corrupted with raw or synthetic noise, we demonstrate that our approach is both efficient and effective in mesh denoising applications. The qualitative and quantitative comparisons further show that our method can outperform the selected state of the art methods, in particular, its computational efficiency.

In the future, it would be useful to accelerate our current framework with the availability of GPU. As a part of our future work, we also plan to resolve the issues mentioned in the limitation part (§6.4).

Acknowledgments

The authors would like to thank Mingqiang Wei for the constructive discussion. This work was supported by NSF Career award IIS 1148976.

[1] Y. Zheng, H. Fu, O.-C. Au, C.-L. Tai, Bilateral normal filtering for mesh denoising, IEEE Transactions on Visualization and Computer Graphics 17 (10) (2011) 1521–1530.
[2] L. He, S. Schaefer, Mesh denoising via l0 minimization, ACM Trans. Graph. 32 (4) (2013) 64:1–64:8.
[3] R. Wang, Z. Yang, L. Liu, J. Deng, F. Chen, Decoupling noise and features via weighted L1-analysis compressed sensing, ACM Trans. Graph. 33 (2) (2014) 18:1–18:12.
[4] M. Wei, J. Yu, W. Pang, J. Wang, J. Qin, L. Liu, P. Heng, Bi-normal filtering for mesh denoising, IEEE Transactions on Visualization and Computer Graphics 21 (1) (2015) 43–55.
[5] X. Lu, Z. Deng, W. Chen, A robust scheme for feature-preserving mesh denoising, IEEE Trans. Vis. Comput. Graph. 22 (3) (2016) 1181–1194.
[6] J. Vollmer, R. Mencl, H. Müller, Improved laplacian smoothing of noisy surface meshes, Computer Graphics Forum (1999) 131–138.
[7] G. Taubin, A signal processing approach to fair surface design, in: Proc. of SIGGRAPH'95, 1995, pp. 351–358.
[8] M. Desbrun, M. Meyer, P. Schröder, A. H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, in: Proc. of SIGGRAPH'99, 1999, pp. 317–324.
[9] X. Liu, H. Bao, H.-Y. Shum, Q. Peng, A novel volume constrained smoothing method for meshes, Graph. Models 64 (3-4) (2002) 169–182.
[10] B. Kim, J. Rossignac, Geofilter: Geometric selection of mesh filter parameters, Comput. Graph. Forum (2005) 295–302.
[11] D. Nehab, S. Rusinkiewicz, J. Davis, R. Ramamoorthi, Efficiently combining positions and normals for precise 3d geometry, ACM Trans. Graph. 24 (3) (2005) 536–543.
[12] A. Nealen, T. Igarashi, O. Sorkine, M. Alexa, Laplacian mesh optimization, in: Proceedings of GRAPHITE'06, 2006, pp. 381–389.
[13] Z.-X. Su, H. Wang, J.-J. Cao, Mesh denoising based on differential coordinates, in: Proc. of IEEE Int'l Conf. on Shape Modeling and Applications 2009, 2009, pp. 1–6.
[14] U. Clarenz, U. Diewald, M. Rumpf, Anisotropic geometric diffusion in surface processing, in: Proc. of IEEE Conference on Visualization '00, 2000, pp. 397–405.
[15] T. Tasdizen, R. Whitaker, P. Burchard, S. Osher, Geometric surface smoothing via anisotropic diffusion of normals, in: Proceedings of IEEE Conference on Visualization '02, 2002, pp. 125–132.
[16] C. L. Bajaj, G. Xu, Anisotropic diffusion of surfaces and functions on surfaces, ACM Trans. Graph. 22 (1) (2003) 4–32.
[17] Y. Zhang, A. B. Hamza, Vertex-based diffusion for 3-d mesh denoising, IEEE Transactions on Image Processing 16 (4) (2007) 1036–1045. doi:10.1109/TIP.2007.891787.
[18] A. El Ouafdi, D. Ziou, H. Krim, A smart stochastic approach for manifolds smoothing, Computer Graphics Forum 27 (5) (2008) 1357–1364.
[19] A. F. El Ouafdi, D. Ziou, Global diffusion method for smoothing triangular mesh, The Visual Computer 31 (3) (2015) 295–310. doi:10.1007/s00371-014-0922-9.
URL `http://dx.doi.org/10.1007/s00371-014-0922-9`
[20] K. Hildebrandt, K. Polthier, Anisotropic filtering of non-linear surface features, Computer Graphics Forum 23 (3) (2004) 391–400.
[21] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: ICCV'98, 1998, pp. 839–846.
[22] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, ACM Trans. Graph. 22 (3) (2003) 950–953.
[23] S. Tsuchie, M. Higashi, Surface mesh denoising with normal tensor framework, Graphical Models 74 (4) (2012) 130 – 139, {GMP2012}. doi:http://dx.doi.org/10.1016/j.gmod.2012.03.010.
URL `http://www.sciencedirect.com/science/article/pii/S1524070312000173`
[24] G. Taubin, Linear anisotropic mesh filtering, IBM Research Report RC22213(W0110-051), IBM T.J. Watson Research.
[25] Y. Ohtake, A. Belyaev, I. Bogaevski, Mesh regularization and adaptive smoothing, Computer-Aided Design 33 (11) (2001) 789 – 800.
[26] H. Yagou, Y. Ohtake, A. Belyaev, Mesh smoothing via mean and median filtering applied to face normals, in: Geometric Modeling and Processing, 2002. Proceedings, 2002, pp. 124–131.
[27] H. Yagou, Y. Ohtake, A. Belyaev, Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding, in: Computer Graphics International, 2003. Proceedings, 2003, pp. 28–33.
[28] Y. Shen, K. Barner, Fuzzy vector median-based surface smoothing, IEEE Transactions on Visualization and Computer Graphics 10 (3) (2004) 252–265.
[29] X. Sun, P. Rosin, R. Martin, F. Langbein, Fast and effective feature-preserving mesh denoising, IEEE Transactions on Visualization and Computer Graphics 13 (5) (2007) 925–938.
[30] X. Sun, P. L. Rosin, R. R. Martin, F. C. Langbein, Random walks for feature-preserving mesh denoising, Computer Aided Geometric Design 25 (7) (2008) 437 – 456.
[31] H. Zhang, C. Wu, J. Zhang, J. Deng, Variational mesh denoising using total variation and piecewise constant function space, IEEE Transactions on Visualization and Computer Graphics 21 (7) (2015) 873–886.
[32] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, L. Liu, Guided mesh normal filtering, Computer Graphics Forum 34 (7) (2015) 23–34.
[33] H. Fan, Y. Yu, Q. Peng, Robust feature-preserving mesh denoising based on consistent subneighborhoods, IEEE Transactions on Visualization and Computer Graphics 16 (2) (2010) 312–324.
[34] Z. Bian, R. Tong, Feature-preserving mesh denoising based on vertices classification, Computer Aided Geometric Design 28 (1) (2011) 50 – 64.
[35] J. Wang, X. Zhang, Z. Yu, A cascaded approach for feature-preserving surface mesh denoising, Computer-Aided Design 44 (7) (2012) 597 – 610.
[36] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, H. Zhang, Edge-aware point set resampling, ACM Transactions on Graphics 32 (2013) 9:1–9:12.

[37] H. S. Kim, H. K. Choi, K. H. Lee, Feature detection of triangular meshes based on tensor voting theory, Computer-Aided Design 41 (1) (2009) 47 – 58.
[38] G. Thürmer, C. A. Wüthrich, Computing vertex normals from polygonal facets, J. Graph. Tools 3 (1) (1998) 43–46.