A Robust Scheme for Feature-Preserving Mesh Denoising

Xuequan Lu, Zhigang Deng, *Senior Member, IEEE,* and Wenzhi Chen, *Member, IEEE*

Abstract—In recent years researchers have made noticeable progresses in mesh denoising, that is, recovering high-quality 3D models from meshes corrupted with noise (raw or synthetic). Nevertheless, these state of the art approaches still fall short for robustly handling various noisy 3D models. The main technical challenge of robust mesh denoising is to remove noise while maximally preserving geometric features. In particular, this issue becomes more difficult for models with considerable amount of noise. In this paper we present a novel scheme for robust feature-preserving mesh denoising. Given a noisy mesh input, our method first estimates an initial mesh, then performs feature detection, identification and connection, and finally, iteratively updates vertex positions based on the constructed feature edges. Through many experiments, we show that our approach can robustly and effectively denoise various input mesh models with synthetic noise or raw scanned noise. The qualitative and quantitative comparisons between our method and the selected state of the art methods also show that our approach can noticeably outperform them in terms of both quality and robustness.

Index Terms—Mesh denoising, feature preserving, robust denoising, feature operations.

1 Introduction

Mesh denoising, that is, recovering high-quality 3D models from meshes corrupted with noise (raw or synthetic), has attracted a lot of attentions in computer graphics community in recent years. Arguably, one of the main driving forces is, 3D datasets acquired by various scanners, in particular, those consumer-grade sensors, are still polluted with noise, and the scanned 3D models need to be properly post-processed including denoising before they can be used for graphics applications. The main technical challenge of robust mesh denoising is to remove noise while maximally preserving geometric features. This problem, in particular, becomes more difficult for models with considerable amount of noise.

Researchers have made noticeable progresses in mesh denoising [1], [2], [3], [4]. Nevertheless, these approaches still fall short for robustly handling various noisy 3D models, having the following limitations. First, the methods of [1], [4] are less robust when handling low-quality noisy meshes, such as degenerate triangles, pseudo-overshoots and severe fold-backs caused by comparatively large noise [2]. Therefore, for such cases the two methods could generate unsatisfactory denoised results (refer to Figures 9, 10, and 12). Second, the method of [2] is capable of handling low-quality noisy meshes but cannot well preserve certain geometric features and fine details. Finally, the method of [3] iteratively detects feature locations, which relies greatly on the detection accuracy and may introduce pseudo-features. Also, no further refining strategy has been proposed to identify potential pseudo-features. Due to the above limitations, these state of the art approaches are limited in terms of robustness and effectiveness when handling mesh models corrupted with substantial noise.

Motivated by the above challenges, in this paper we present a novel robust mesh denoising approach. Taking a noisy mesh model as the input, our method can robustly produce its corresponding denoised model with high quality. Specifically, our method has the following main contributions.

- **Initial estimation.** In general it is difficult to distinguish features from noise, especially when noise level is high. Therefore, we introduce an initial estimation stage (§4) to generate an improved mesh from the input noisy model. This strategy can largely reduce the noise level and better shape the input triangular mesh, and thus can significantly facilitate follow-up feature operations.
- **Feature operations.** To preserve features during the denoising process, we propose a string of feature operations including feature detection (§5.1), identification (§5.2), and connection (§5.3). We present a sparsity-inspired L_1-norm regularization to first detect all the features in one single iteration that may contain a few pseudo-features. Then, we define several criteria to effectively identify the detected features. Finally, the remaining continuous features are connected to form feature edges which can further remove remaining pseudo-features and provide accurate neighboring information for vertex update.
- **Vertex update.** To achieve desired results, we design a new feature-aware vertex update algorithm (§6) that is iterative, effective and fast. This algorithm takes advantage of local geometric neighboring information of each vertex and updates vertices in a feature-aware manner.

- *X. Lu and W. Chen* are with the College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang Province, P. R. China.*
  E-mails: xuequanlu@zju.edu.cn, chenwz@zju.edu.cn
- *Z. Deng* is with the Department of Computer Science, University of Houston, Houston, Texas, USA.*
  E-mail: zdeng4@uh.edu

Manuscript received on April 3rd, 2015; revised on July 22nd, 2015 and September 24th, 2015; accepted on November 5th, 2015.

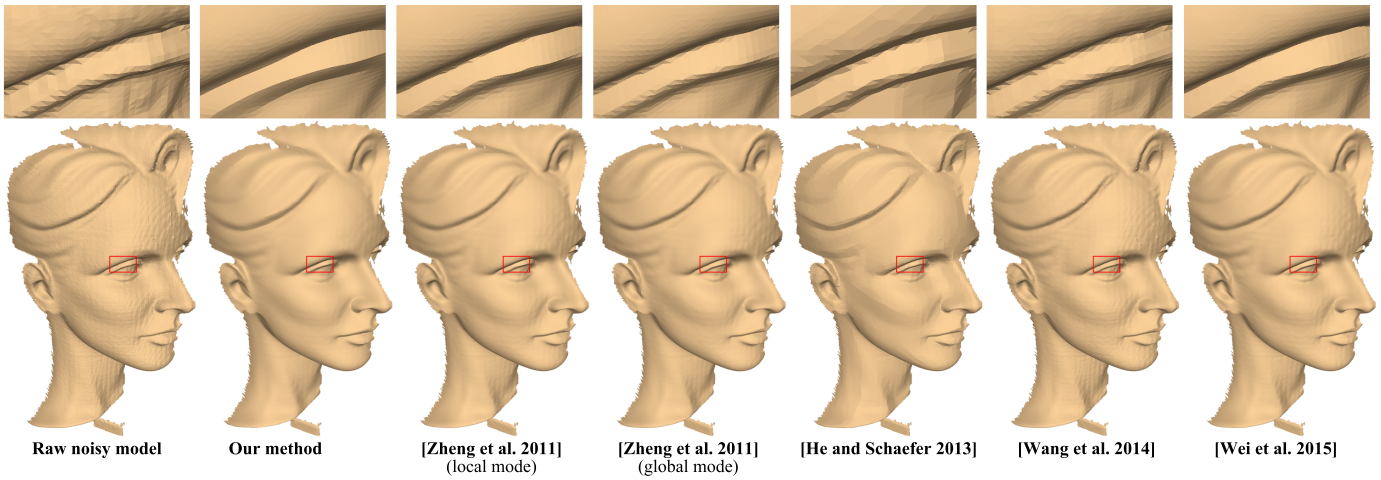


Fig. 1. Denoised results of a raw Wilhelm model. Our approach can soundly recover feature edges, while the results by the selected state of the art methods are visually hackly (refer to the top zoomed region).

In our experiments, our approach can outperform the state of the art mesh denoising methods [1], [2], [3], [4] in terms of robustness and effectiveness. Through various experiments and comparisons (§7), we show that our method can achieve noticeably better denoised results than the previous techniques. One of comparison examples is shown in Figure 1. Finally, we discuss the three stages of our method, the potential use of our method, and its limitations (§8).

2 Related Work

In this section, we only review recent research efforts that are most relevant to this work. For a comprehensive review on this topic, interested readers are referred to the articles by Botsch et al. [5], [6]. Specifically, we first review traditional mesh denoising/smoothing methods, and then focus on sparsity inspired mesh denoising techniques.

2.1 Traditional Mesh Denoising

Most early mesh smoothing methods cannot preserve geometric features since they are *isotropic* (i.e., independent of surface geometry). A non-shrinking, two-step smoothing method via signal processing was proposed by Taubin [7]. Vollmer et al. [8] presented a simple and fast Laplacian smoothing algorithm. However, it generates surface shrinkage and cannot preserve sharp features. Meanwhile, a fairing method [9] based on diffusion and curvature flow was put forward for irregular meshes. Later, researchers proposed a variety of isotropic denoising methods on the basis of volume preservation, pass frequency or differential properties [10], [11], [12], [13], [14].

Due to the fact that isotropic methods have difficulty in preserving geometric features, *anisotropic* denoising methods have attracted considerable attention. Various methods based on diffusion or differential information have been proposed [15], [16], [17], [18], [19], [20], [21], [22]. Researchers successfully extended bilateral filtering in the domain of image denoising [23] to 3D mesh denoising. A bilateral mesh denoising method, directly applied to vertices, was also proposed by Fleishman et al. [24]. Later, researchers presented various bilateral filters to handle surface normals instead of vertex positions [1], [25], [26].

Besides directly updating vertices, the idea of first filtering normals followed by updating vertices has also caught much attention in recent years [1], [25], [27], [28], [29], [30], [31], [32], [33], [34], [35]. Taubin [27] and Ohtake et al. [28] made seminal contributions to such two-step methods. Yagou et al. [29], [30] proposed mean, median and alpha-trimming filters to estimate face normals. Later, a fuzzy median filter was introduced by Shen and Barner [32]. Bilateral filters have been extended to filter surface normals as well [1], [25], [26]. Sun et al. [33], [34] presented two different two-step methods by first weighted averaging neighboring face normals and then updating vertex positions accordingly. Recently, Zhang et al. [35] presented a variational method for face normal filtering and vertex update using the algorithm in [33].

A few recent techniques [4], [36], [37], [38], [39] have focused on vertex/face classification before mesh denoising, in order to achieve better denoised results. Since the classification step in these methods is generally performed on noisy models, their results are sensitive to the noise level. For example, some vertices or faces could be misclassified.

2.2 Sparsity Inspired Mesh Denoising

If the underlying signal indeed has a sparse representation, in theory it can be reconstructed by solving a sparse problem [40]. Several recent works have introduced the concedpt of sparsity into mesh denoising applications [2], [3]. For example, He and Schafer [2] presented a L_0 minimization framework by introducing a discrete differential operator. In their work, the area-based edge operator is sparse for a mesh model (i.e., there are only a small number of edges whose operator norms are large). Later, Wang et al. [3] proposed a method to decouple noise and features via weighted L_1-analysis compressed sensing. Their work asserts that the pseudo-inverse matrix of the Laplacian of a mesh is a coherent dictionary for sparsely representing sharp feature signals on the shape.

From a sparsity perspective, feature vertices on a mesh model are sparse and the number of them is typically much

smaller than non-feature vertices. We thus formulate a novel sparse problem in order to detect features. L_0 optimization is in fact the sparsest solution, e.g., the work of [2], but it is highly non-convex and requires to solve a combinatorial problem. At certain cases the convex L_1 minimization can be used to replace L_0 to find a sparse approximation of L_0, e.g., the method in [3]. However, the method in [3] contains additional constraints; thus, it is less robust, which could lead to unsatisfactory results for some cases.

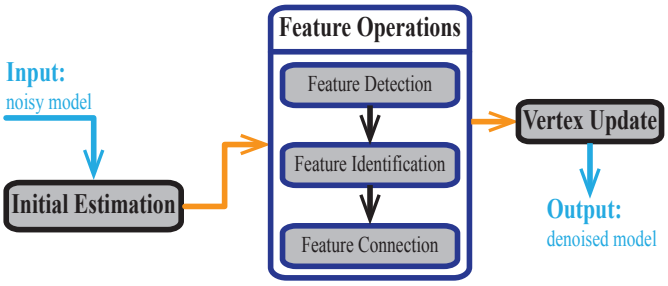


Fig. 2. The pipeline of our feature-preserving approach for mesh denoising.

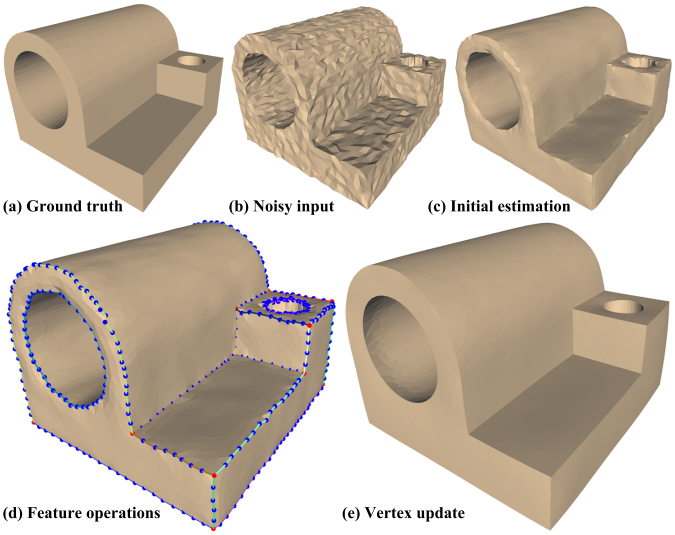


Fig. 3. An example for demonstrating the pipeline of our mesh denoising approach. (a): The ground truth. (b): The input noisy model. (c): The initial estimation from (b). (d): The connected feature edges (rendered with various colors) after feature operations (feature detection, identification and connection) are applied. Here, corner points (described in §5.3) are colored with red, and edge points (§5.3) are colored with blue. (e): The final denoised result after the vertex update step is applied to (d). Note the model is zoomed in for clear observation.

3 Method Overview

Our method consists of three stages: initial estimation; feature detection, identification and connection; and feature-aware vertex update, as illustrated in Figure 2. An example for demonstrating our mesh denoising approach is shown in Figure 3.

Initial estimation. First, we estimate an initial mesh from the input noisy model by solving a weighted least squares optimization problem. At this stage, users can adjust the employed parameters to achieve a desired estimation.

Feature detection, identification and connection. Based on the estimated mesh from the previous stage, we first detect all of its features by optimizing a L_1-norm regularized formula. Then, we identify the detected features and remove the pseudo-features by introducing several criteria. Finally, we connect the identified features to form feature edges, which are used at the follow-up vertex update stage.

Feature-aware vertex update. At the final stage of our method, we introduce a novel iterative vertex update algorithm to produce denoised results efficiently.

4 Initial Estimation

The basic idea of this initial estimation step is to smooth an input noisy model to some extent while preserving its features. Inspired by the L_0 minimization framework [2], we formulate the initial estimation of the input noisy model as the following weighted least squares optimization problem.

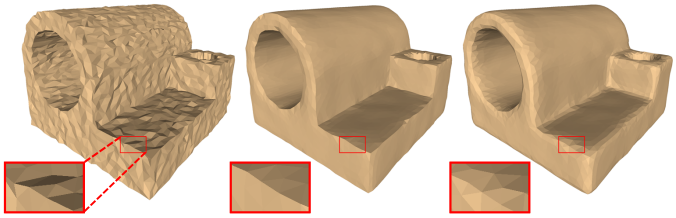
$$\min \sum_i \|\tilde{p}_i - p_i\|_2^2 + \alpha \sum_e w(e)\|D(e)\|_2^2 + \beta \sum_e w(e)\|R(e)\|_2^2, \tag{1}$$

where $\tilde{p}_i$ is the unknown (to be solved) position of the i-th vertex, p_i is the position of the i-th vertex in the input noisy mesh, α and β are weighting coefficients which balance the three terms. $D(e)$ is an area-based edge operator to measure the edge sharpness, and $R(e)$ is a regularizer to eliminate spurious overshoots and fold-backs [2]. $w(e)$ is the weight for edge e, and it is expected to be small when both the two vertices of an edge are features in order not to be smoothed. Thus, it can be defined as an exponential function, using the pairwise face normals of the two faces sharing edge e.

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

where θ is the angle between a pair of face normals whose corresponding faces share the same edge e, and σ_θ is the angle threshold to scale the pairwise normal similarity. It is noteworthy that as shown in Eq. 1, we formulate the initial estimation problem as a weighted least squares optimization problem, which is different from the formulated L_0 minimization problem in the work of [2]. The main reasons are: (1) L_0 minimization is computationally intensive and is hard to optimize due to its non-convex nature; (2) L_0 minimization is prone to generate results with planar regions (e.g., lost details or weak features), such as Figure 15. Thus it is inappropriate to use such an algorithm at the pre-processing step. In contrast, our formulation is fast and can produce a satisfactory initial estimate for the follow-up stages.

Eq. 1 estimates the input noisy mesh in the following way: it automatically offers large weights to non-feature vertices and small weights to feature vertices, and therefore smoothes non-features and preserves features. Such a way significantly reduces noise interferences and ensures that this optimization can estimate a sound mesh from the noisy input. Figure 4 shows that encoding edge weight $w(e)$ to Eq. 1 can estimate an initial mesh with preserved features. We also test the involved parameters in Figure 5, and found that a larger σ_θ or α tends to produce smoother estimation.



(a) Noisy model **(b)** with edge weight **(c)** without edge weight

Fig. 4. The initial estimations with and without the edge weight $w(e)$. (a) an input noisy model, (b) the estimated result with the edge weight $w(e)$, (c) the estimated result without the edge weight $w(e)$.

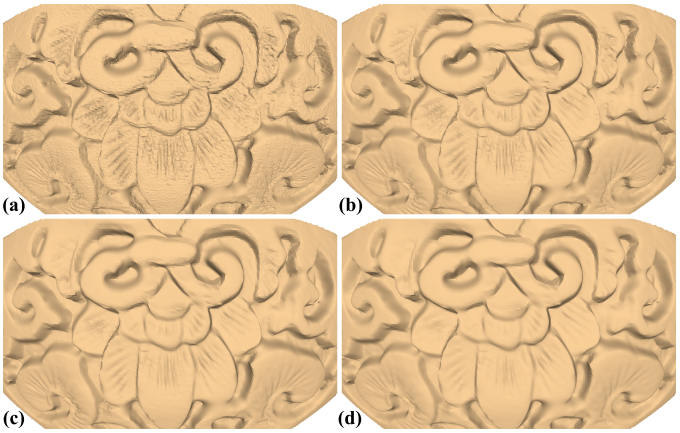


Fig. 5. The values of parameters used in the initial estimation stage. (a): raw noisy model. (b): $\alpha = 0.3$, $\sigma_\theta = 15$. (c) $\alpha = 0.3$, $\sigma_\theta = 30$. (d) $\alpha = 0.8$, $\sigma_\theta = 30$. For (b) (c) and (d): $\alpha = \beta = 0.03$ at the first unweighted (without $w(e)$) iteration; $\beta = 0.03$ for three weighted iterations.

Note that β (by default $\beta = 0.1\alpha$ for weighted optimizations) is the coefficient of the triangle shape regularizer $R(e)$ and should be set based on the noise level. Hence, parameters of this stage should be carefully tuned to preserve weak features or details. Figure 6 shows the necessity of the initial estimation stage: without this stage, the noisy model could hardly be effectively denoised afterwards since many features are misdetected.

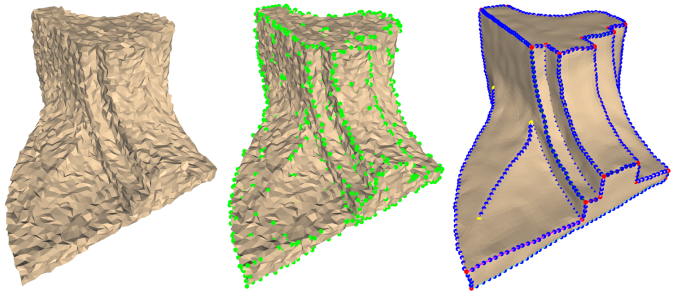
To generate a sound initial estimation for follow-up stages, our method optimizes Eq. 1 for several times. For the three parameters employed in Eq. 1, in our experiments, we empirically set $\alpha = \beta = 0.1$ and optimize the unweighted form (i.e., without $w(e)$) of Eq. 1 at the first iteration. Then, at follow-up iterations (3 iterations by default), we set $\alpha = 0.5$, $\beta = 0.05$, and $\sigma_\theta = 30°$. Note that Eq. 1 can be straightforwardly solved as a sparse system of linear equations [41].

5 Feature Detection, Identification and Connection

With the estimated mesh from the above stage, we then perform a series of operations on features, including feature detection, feature identification, and feature connection, described below.

5.1 Feature Detection

We observe that if the i-th vertex and its 1-ring neighbors are co-planar, its uniformly-weighted Laplacian (i.e., differential coordinates) [42] is perpendicular to the surface normal at this vertex. The underlying causality is: if a vertex and its 1-ring neighboring vertices are co-planar, its uniformly-weighted Laplacian is expected to be parallel to this plane, since this vertex normal is perpendicular to this plane, and it is also perpendicular to the uniformly-weighted Laplacian. Our observation can also be applied to the scenario where the Laplacian is a zero vector.



(a) Noisy model (b) without initial estimation (c) with initial estimation

Fig. 6. The importance of the initial estimation stage. (a) A noisy input model, (b) the detected features (colored with green, described in §5.1) on the noisy model, (c) feature operations (§5) on the initial estimated mesh. Here, feature edges (described in §5.3) are rendered with different colors. Corner points, end points and edge points (§5.3) are colored with red, yellow and blue, respectively.

In the following, we first define f_i, a function used to determine whether the i-th vertex is a feature or not. In general, the absolute value of f_i of feature vertices should be significantly larger or smaller than those of non-feature vertices. Based on this observation, it is intuitive to define f_i as the dot product of the i-th vertex normal n_i^V and its Laplacian L_iP for the i-th vertex, as follows.

$$f_i = n_i^V (L_i P)^T, \tag{2}$$

where the superscript V denotes vertex, L_i is the i-th row of the uniformly-weighted Laplacian matrix L, and P is the vectorized form of all vertex positions. The superscript T stands for transposition.

It is obvious that in a model there exist only a small (i.e., sparse) number of vertices (i.e., features) whose $|f_i|$ is large. In other words, most vertices (i.e., non-features) have a close-to-zero $|f_i|$; features are indeed sparse in a model. L_0-norm minimization is theoretically the sparsest solution [40]. However, L_0-norm problems are non-convex and difficult to optimize, and current approximate optimizations generally take long computational time [2].

To alleviate the above issue, we introduce a simple L_1-norm regularized formula that can be efficiently optimized. The convex L_1-norm regularization is used to find a sparse approximation of L_0 minimization and given by,

$$\min \|x - f\|_2 + \lambda \|x\|_1, \tag{3}$$

where $f = (f_1, \cdots, f_n)^T$, x is the vectorized form of the unknowns, and λ is the regularization parameter to control the number of non-zero elements (i.e., sparsity) of the vector x, by regulating the trade-off between $\|x - f\|_2$ and $\|x\|_1$. A larger λ would result in a smaller number of non-zero elements; and vice versa. The vertex normal n_i^V is computed by normalizing the area-weighted average

among its neighboring face normals. In this work, we use the CVX package [43] to solve Eq. 3 and interpret x_i as a non-zero entry if $|x_i| > 10^{-5}$.

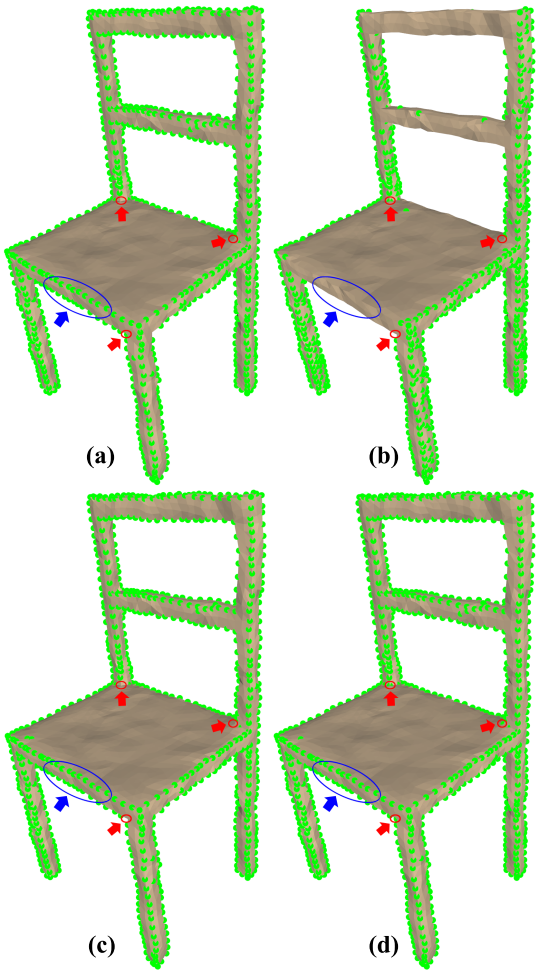


Fig. 7. Feature detection results using different f_i. We tune λ to let the number of non-zero entries be the same for the four cases. (a) Using f_i in Eq. 2, (b) using f_i with an intuitive standardization (described in §5.1), (c) using the mean curvature [44], (d) using f_i in Eq. 4. The form of f_i in Eq. 4 clearly produces the best detection result (refer to the blue and red circled regions, pointed by arrows). Note that, the red circled regions pointed by red arrows are zoomed in to clearly show the differences.

However, we found that the defined f_i in Eq. 2 does not take into account surface sampling density, which might be unrobust (Figure 7(a)) for irregularly sampled meshes [45]. In other words, f_i does not include any kind of standardization. Lacking of a standardization would lead to some potential failure cases. For example, if a non-feature vertex is far from its neighbors or the face area formed by a vertex and its neighbors is large, then this vertex might be misidentified as a feature since its f_i could be large. Besides, the feature vertex normals may be inaccurately estimated using the general area-weighted computation. As a result, some features might be missed via optimizing Eq. 3 (an example is shown in Figure 7(a)). The premise is to use the general area-weighted computation to estimate vertex normals. So, efforts need to be paid to find an appropriate standardization.

A straightforward way would be to divide f_i by the total area of the 1-ring neighboring faces of the current vertex. But our experiments show that the detected results are poorer (shown in Figure 7(b)) than those without standardization (i.e., using Eq. 2). This is not surprising since the Laplacian of a vertex is closely related to the vertex and its 1-ring neighboring vertices. Therefore, we extend Eq. 2 to a new form in Eq. 4 by including area standardization, as follows:

$$f_i = \frac{1}{\sum_{j \in NF(i)} a_j} \sum_{j \in NF(i)} a_j (n_i^V - n_j^F)(L_i P)^T, \quad (4)$$

where $NF(i)$ is the 1-ring neighboring faces of the i-th vertex, a_j is the j-th face area in $NF(i)$, the superscript F denotes face, and n_j^F is the corresponding face normal. Optimizing Eq. 3 by substituting Eq. 4 to Eq. 3 can generate better detection results than using Eq. 2 (shown in Figure 7(c)). Besides the comparisons among the four forms of f_i (Eq. 2, the intuitive standardization, the mean curvature [44] and Eq. 4, illustrated in Figure 7), we also tested the cotangent-weighted Laplacian instead of the uniformly-weighted Laplacian, and we found the latter involved in Eq. 4 has a better performance for feature detection.

It is noteworthy that a weighted L_1-analysis technique has been recently proposed by Wang et al. [3]. The main differences between their method and our approach are: (1) they apply L_1-analysis on residual, $n_i^V(p_i - \tilde{p}_i)^T$: the dot product of the surface normal n_i^V and the direction by subtracting the Laplacian smoothing position $\tilde{p}_i$ from its original noisy position p_i, while our formula on f_i is more intuitive: the Laplacian of a vertex is perpendicular to both the vertex normal and the surrounding face normals of this vertex if it and its neighbors are co-planar, that is, $f_i = 0$ (Eq. 4); (2) their method has a weighted L_1-norm constraint, while our method introduces a simple unweighted L_1-norm regularization without any additional constraints; (3) their method is iterative (i.e., optimizing L_1 for multiple times), while our method is non-iterative, which is much more computationally efficient.

5.2 Feature Identification

We can obtain detected features from the above step. Nevertheless, some of the detected features may be pseudo-features that should be identified and discarded. Recent sparsity-inspired methods [2], [3] do not take special care of this aspect. We design effective feature identification strategies based on the following criteria.

- **Isolation criterion.** We first adopt the normal tensor voting algorithm [46] to the isolated features to identify corners. The identified corners are immediately excluded from isolated features and marked as true features. Afterwards, if isolated single features exist in the detected features, we mark them as pseudo-features and remove them accordingly. In a similar way, we can identify and remove isolated coupled features that are only adjacent to each other (i.e., not having other neighboring detected features).
- **Threshold criterion.** Even if there exist multiple features around a detected feature, it may also be a pseudo-feature. To identify those pseudo-features, we first define a measurement function for the k-th detected feature as follows. This function measures

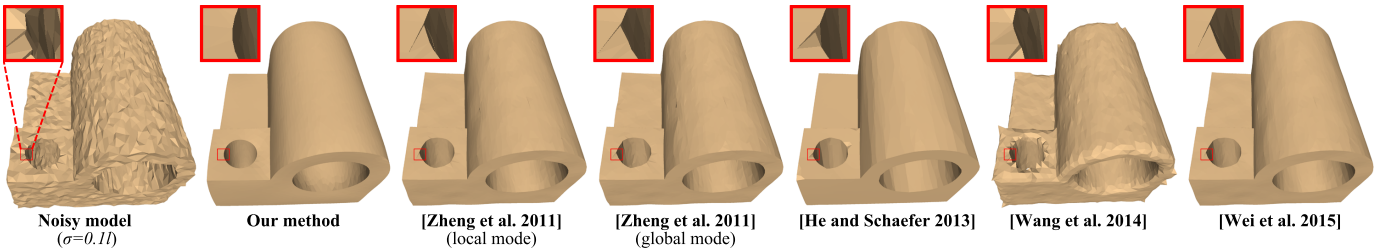


Fig. 8. Denoising results of the Joint model (comparison among our method and the selected start of the art methods). Please refer to the zoomed red rectangular windows.

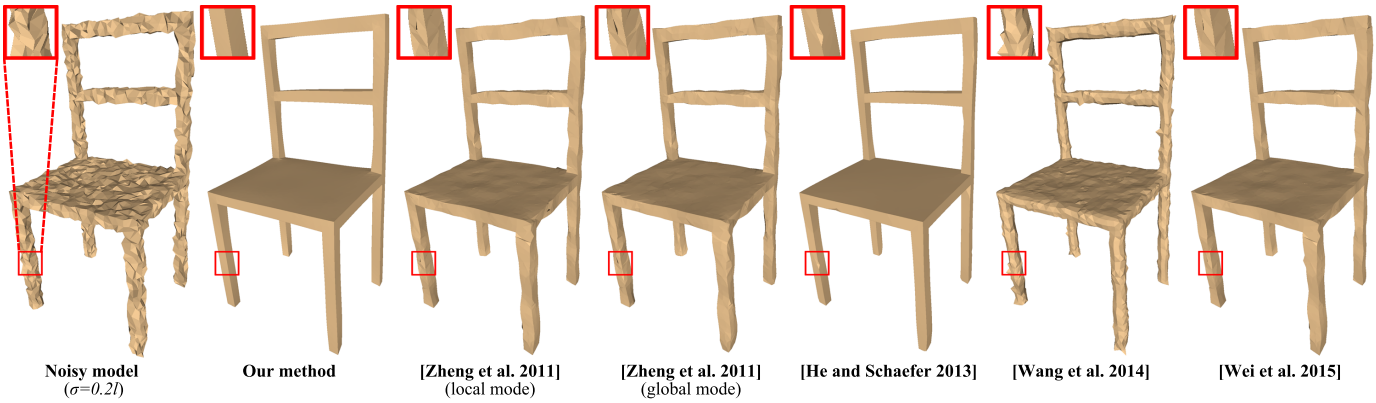


Fig. 9. Denoised results of the Chair model (comparison among our method and the selected start of the art methods). Please refer to the zoomed red rectangular windows.

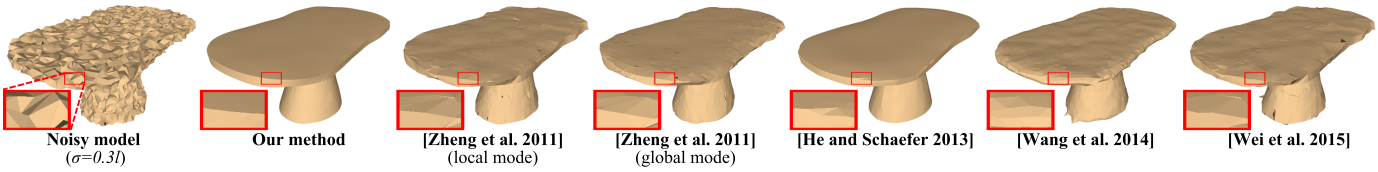


Fig. 10. Denoised results of the Table model (comparison among our method and the selected start of the art methods). Please refer to the zoomed red rectangular windows.

the energy of a detected feature, and the energies of true feature vertices should be generally greater than those of pseudo-feature vertices.

$$M_k = \frac{1}{\sum_{j\in NF(k)} a_j} \sum_{j\in NF(k)} a_j n_k^V (n_k^V - n_j^F)^T \quad (5)$$

Adaptive threshold. Then, we compute the average of the measurements of all other neighboring detected features around the k-th feature vertex as follows:

$$M_k^{Avg} = \frac{1}{|NV_1(k)|} \sum_{j_1\in NV_1(k)} M_{j_1}, \quad (6)$$

where $NV_1(k)$ is the neighboring detected features of the k-th feature vertex, and $|NV_1(k)|$ is the number of elements in $NV_1(k)$. We can observe that the average measurement varies since each detected feature has different neighboring detected features. Therefore, M_k^{Avg} acts as a self-adaptive threshold. In our work, if $M_k < \gamma M_k^{Avg}$, we identify the k-th feature as a pseudo-feature. γ is the only parameter at this step. In our experiments, it is typically set in the range of 0.25 to 0.55.

The two criteria defined above can be applied alternately several times (2 times empirically), and can reliably identify most of pseudo-features from the detected features. To the end, the remaining pseudo-features (if existing) can only impose limited impact on the next step, and actually they can be further removed.

5.3 Feature Connection

At the feature connection step, we connect continuous features to form distinct edges called *feature edges* (refer to Figure 3(c)). We first classify the resulting features from the above step into three categories: *corner points*, *edge points*, and *end points*. Note that an end point is the start or end of a feature edge, but it is not a corner point. We use the following steps for feature connection.

(1) We employ the normal tensor voting algorithm [46] to identify corners. We also further check whether a feature point is an end point, by checking whether it has only a single neighboring feature point. The remaining feature points are edge points.

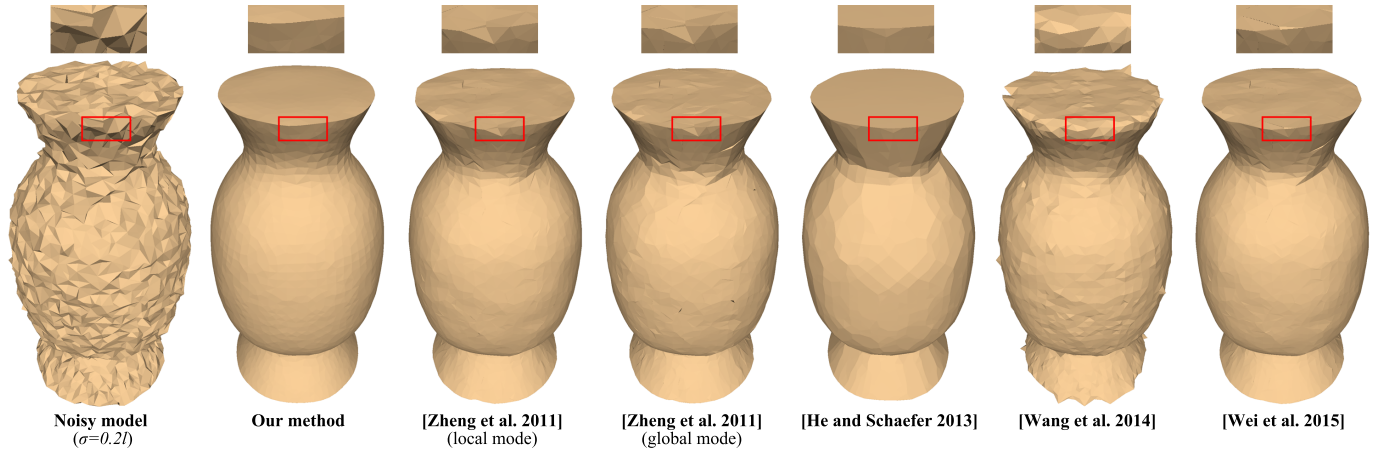


Fig. 11. Denoising results of the Vase model (comparison among our method and the selected start of the art methods). Note that the denoised result by He and Schaefer [2] is also clean (without folded faces), but it looks unnatural, especially at the neck of the vase model.

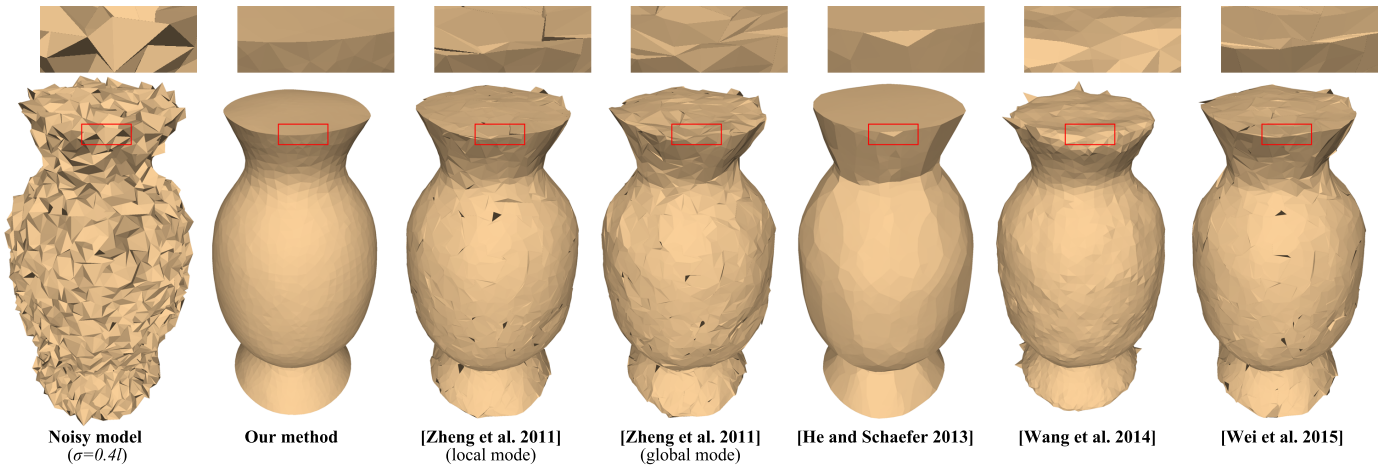


Fig. 12. Denoised results of the Vase model contaminated with heavy noise. Our method still robustly achieves a clean and high-quality result which is clearly better than the selected state of the art methods.

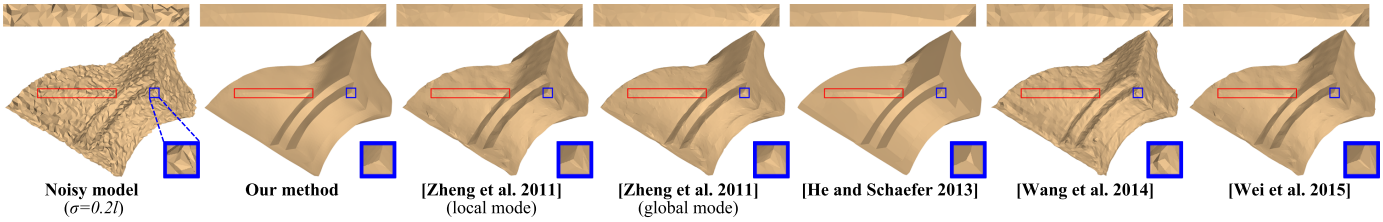


Fig. 13. Denoising results of the Fandisk model (comparison among our method and the selected start of the art methods). Please refer to the two zoomed regions.

(2) In general, the first point and last point of a feature edge are corner points or end points, unless the feature edge is a loop. We thus start at a corner point or an end point, and put it and one of its neighboring features into a new feature edge list. To determine if the next neighboring feature point can be added to the current feature edge, we compute the angle, κ, between the first line segment (formed by the last vertex and its previous vertex in the current edge list) and the other line segment (formed by the last vertex in the current edge list and its next neighboring feature vertex). If $\kappa > \kappa_{thr}$ and only one neighboring feature point exists, we add this neighboring feature to the current feature edge. If there are multiple neighboring features, we first select the one with the largest κ and then check whether $\kappa > \kappa_{thr}$. If so, we also add it to the current feature edge. We repeat this operation until encountering a corner point or an end point, or $\kappa \leq \kappa_{thr}$. We empirically set $\kappa_{thr} = 130^{\circ}$, and users may loose this constraint by choosing a smaller κ_{thr}.

(3) We search for a limited distance to check if two feature edges can be further connected as a longer one. Specifically, we start at the beginning or the ending feature point of a feature edge E_{t1}, and put its neighboring

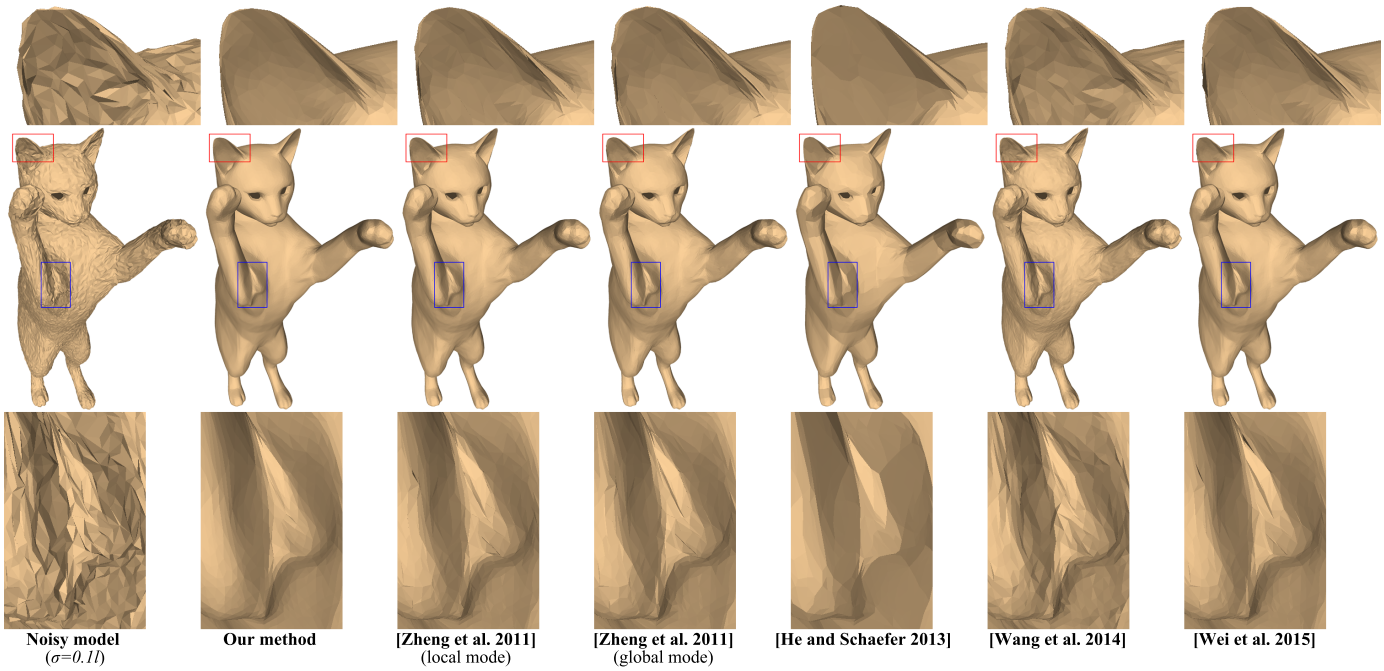


Fig. 14. Denoised results of the Cat model (comparison among our method and the selected start of the art methods). Please refer to the two zoomed windows for comparing differences.

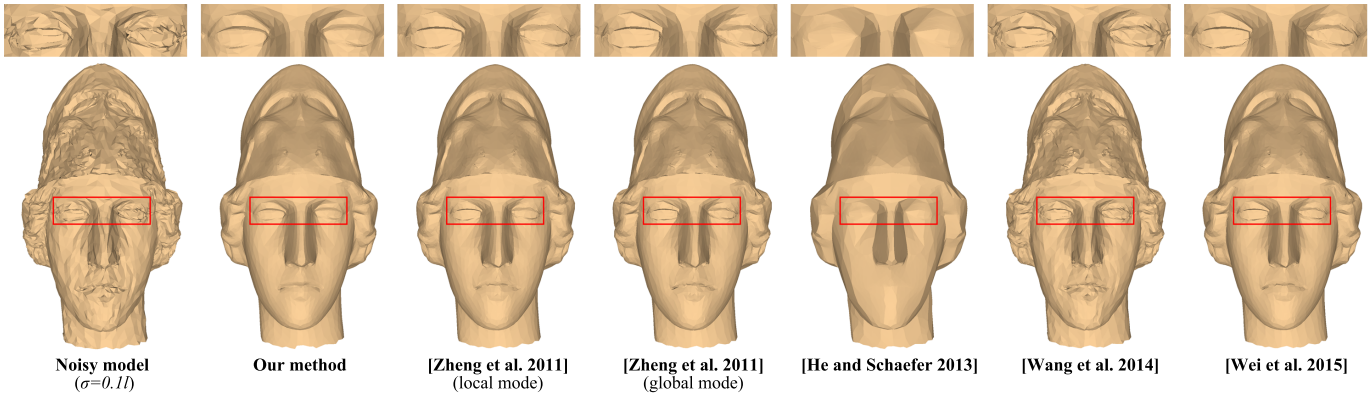


Fig. 15. Denoised results of the Ateneam model (comparison among our method and the selected start of the art methods). Please refer to the zoomed regions.

feature point in E_{t1} and it into a search list, as the first and second elements, respectively. We then search one of the 1-ring neighboring vertices of the last point in the search list that has the largest angle, κ_1, between two line segments (one is formed by the last point and its previous point in this search list, and the other is formed by the last point in the search list and this searching point). If $\kappa_1 > \kappa_{thr}$, we move this searched point into the search list and further seek the next neighboring point. If the distance is less than or equal to the user-specified distance threshold (5 points by default), which means the search meets the beginning or the end of another feature edge E_{t2}, then we add the searched points and E_{t2} onto E_{t1}, and also delete E_{t2}. Otherwise, we discard the search and leave E_{t1} and E_{t2} unchanged.

(4) A 3D model may contain looping feature edges (e.g., Figures 8, 10 and 11). We randomly select a feature point from the remaining features and mark it as a virtual corner point. Then, we perform the above step (2). We set this virtual point as an edge point if the newly connected feature edge has the same beginning and end, which indicates it is a looping feature edge. We repeat this operation until no new feature edges can be created.

After the feature connection step, the remaining unconnected features are identified as pseudo-features, and we just simply eliminate them.

6 Feature-aware Vertex Update

From the above stage we can obtain feature edges. Now we need to make non-feature regions smooth while preserving features. Our basic idea is to smooth non-feature places by considering the 1-ring neighbors of each non-feature vertex, and smooth the constructed feature edges by taking the two bilateral neighbors (along the corresponding feature edge)

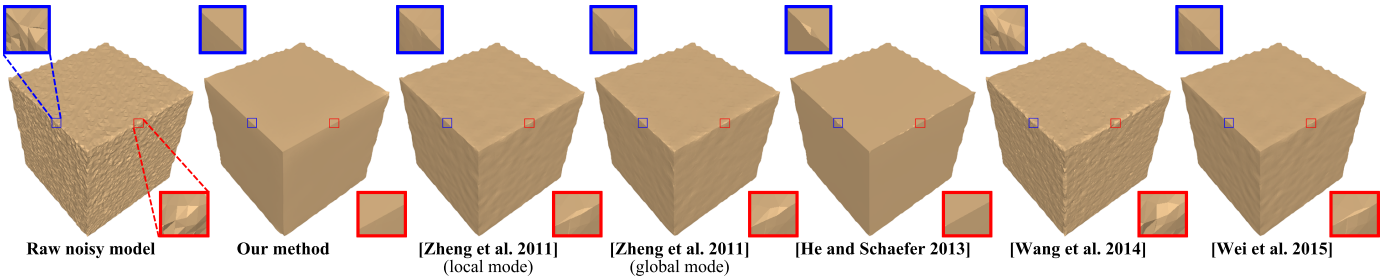


Fig. 16. Denoised results of a scanned Cube model (comparison among our method and the selected start of the art methods). Please refer to the zoomed red and blue regions.

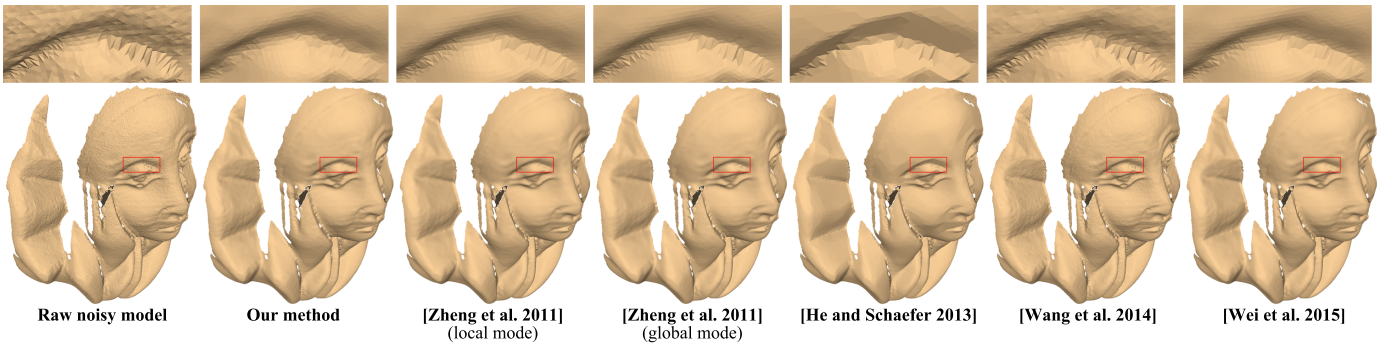


Fig. 17. Denoising results of a scanned Pierrot model. The selected state of the art methods generally produce undesired results, such as the zigzag pattern (refer to the zoomed rectangular window). By contrast, our approach can generate a desired result which significantly reduces the zigzag pattern.

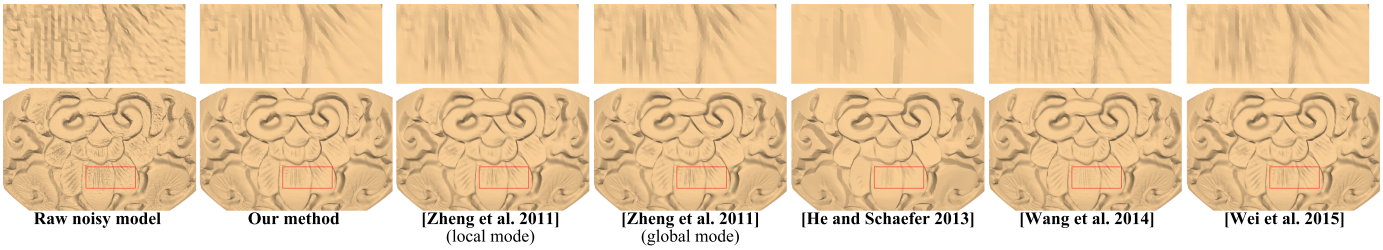


Fig. 18. Denoised results of a raw Decorative pattern embossed model. Even this model involves many weak features and details, our method can also produce a desired result, in terms of fine details and features preservation.

of each feature vertex. Specifically, we propose a novel, efficient, iterative vertex update method as follows.

$$p_i^{'} = \frac{p_i N + \frac{\sum_{j_2 \in NV_2(i)} p_{j_2}}{|NV_2(i)|}}{N+1}, \tag{7}$$

where $p_i^{'}$ and p_i are the new position and the original position of the i-th vertex, respectively, N is an integer constant that balances the importance between the vertex itself (data term) and the centroid (smooth term) of its actual neighboring vertices $NV_2(i)$, and $|NV_2(i)|$ denotes the number of elements in $NV_2(i)$. Specifically, $NV_2(i)$ is computed as the 1-ring neighbors of a non-feature point or an end point, and the two bilateral neighboring feature points of an edge point. We handle corner points in the following way: we simply fix them after the first stage because the corners of the initial estimated mesh are reasonable while the corners of the input noisy model may be still kept noisy.

In our method, we set different N for non-feature vertices and feature vertices. A larger N means a stronger bias on the data term, which will make the smoothing task more difficult. In our experiments, we set $N = 1$ for non-feature vertices. Because the curvedness of a curved feature edge is expected to be preserved, we have to further classify feature edges to two types: curved edges and straight edges. Our classification scheme is simple and efficient: we apply PCA to each feature edge and its half, and take the most significant principal component as the main direction, respectively; it is a curved edge if the angle between the two main directions is larger than a threshold (=10° in our experiments). We set $N = 6$ for the edge points of a curved feature edge, $N = 1$ for the edge points of a straight feature edge, and $N = 6$ for the end points and corner points.

7 Experimental Results

We tested our method on many mesh models corrupted with either synthetic or raw scanned noise. Meanwhile, we also tested several state-of-the-art mesh denoising methods on the same test set as comparisons. In this section, we use

Method I to denote our approach. The selected state of the art mesh denoising methods are **Method II** [1], **Method III** [2], **Method IV** [3], and **Method V** [4]. Method II has two forms of solutions (local and global), thus we further denote them as Method II (local) and Method II (global).

Parameter sets. Each mesh denoising method has its own parameters. Specifically, we used the following parameter sets for the above approaches: Method I = (λ, γ, vertex update iterations); Method II (local) = (normal filtering iterations, σ_s, vertex update iterations); Method II (global) = (λ, σ_s, vertex update iterations); Method III = (μ, α_0, λ); Method IV = (τ); and Method V = (σ_{s1}, n_1, σ_{s2}, n_2).

Parameters choosing. We used the default parameter values suggested by the original authors [2], [3] for the Methods III and IV. Specifically, the parameter values used in the Method III and IV are $(\sqrt{2}, 0.1\overline{\gamma}, 0.01l^2\overline{\gamma})$ - abbreviated as *Default1*, and $(0.1\sqrt{n})$ - abbreviated as *Default2*, respectively. The parameter values used in the Method I, II and V are elaborately tuned to achieve best visual results for all the test models. Specifically, in the Method I, a larger λ detects a smaller number of non-zero elements (i.e., features) and λ is in the range of [0.01, 0.1]; γ is used to identify the detected features and is in the range of [0.25, 0.55]; the number of vertex update iterations is in the range of [5, 30] and a greater number would lead to smoother results. In the Method II [1], the number of normal filtering iterations is in the range of [3, 25] in our experiments, and a larger number would result in smoother normals; σ_s is suggested to be in the range of [0.2, 0.6], with a greater values for a higher level of noise; λ is in the range of [0, 1], and a smaller value of it leads to smoother results; the number of vertex update iterations is in the range of [5, 50] in our experiments, and it should be larger for a higher level of noise as the vertices deviate further from their true positions [33]. In the Method V [4], both σ_{s1} and σ_{s2} are in the range of [0.2, 0.6], with larger values for heavier noises; n_1 is in the range of [3, 20], with greater numbers for smoother normal initialization; n_2 is in the range of [1, 15], with greater numbers for larger noises. Table 1 shows the used parameter values of all the methods on some test 3D models. In terms of the implementation of the selected comparison methods, we used the source code provided by Zheng et al. [1] and implement the other three methods [2], [3], [4] by strictly following their algorithms and additional responses from their original authors for our technical inquiries.

Test models. We test both our approach and the selected state of the art methods on a variety of models with either synthetic or raw noise.

- *Models with synthetic noise.* Models corrupted with synthetic noise are obtained by adding zero-mean Gaussian Noise with standard deviation σ to their corresponding ground truth. σ is proportional to l, the average edge length of the ground truth. From Figures 8 to 13, we clearly observe that our approach can generate substantially better results than the selected state of the art methods, in terms of feature preservation and cleanness. Note that the Chair model (Figure 9) and the Table model (Figure 10) have thin side faces which may bring extra difficulty for denoising, but our method can robustly generate sound results. Figure 12 further demonstrates the robustness of our approach when dealing with a high level of noise. Figures 14 and 15 show that our approach can also produce good quality results in terms of cleanness and details-preserving, even using the first stage of our pipeline alone.
- *Models with raw noise.* In addition to the models corrupted with synthetic noise, we also tested all the methods on real 3D scanned data. Figure 1 shows that when all the methods (Methods I to V) were applied to a raw scanned Wilhelm model, our method is noticeably better than all the other methods in terms of feature preservation (refer to the zoomed rectangular windows). Figure 16 shows that our approach can generate better results than the state of the art methods in terms of sharp edge recovery. From Figure 17, we found that the selected state of the art methods generate less desired results (e.g., the zigzag pattern) compared with our approach. In Figure 18, we demonstrate that our method can also generate compelling results when denoising 3D models with many weak features or details, by adopting the first stage only.

Quantitative comparisons. Besides the above visual qualitative comparisons, we also compared our approach with the state of the art methods in a quantitative way, listed in Table 1. We chose the quantitative metrics E_v (L^2 vertex-based error) and MSAE (the mean square angular error) suggested by the previous works [1], [4]. E_v is to measure the positional error between the ground-truth model and the denoised model, and MSAE is to estimate the mean square angular error between the face normals of the ground-truth model and those of the denoised model. Additionally, we recorded computational time for each stage of our approach on some test models (shown in Table 2).

From Table 1, we can observe the following: First, for all the test models in Table 1, MSAE by our approach is significantly smaller than all the selected state of the art methods, which is consistent with the qualitative (visual) comparison outcomes. Second, we found mixed comparison results in terms of E_v. In other words, E_v generally, but not invariably, is in agreement with the visual comparison outcomes, which was also reported in the previous work [33]. The justification for mixed E_v outcome is that, vertices may be repositioned towards larger or smaller E_v during denoising. Specifically, our approach generated larger E_v for some test models (i.e., Table, Vase (Figure 11), Cat and Ateneam in Table 1), since all the faces of the noisy input are better reshaped and vertices are relocated in our method, which could result in a larger positional error between the denoised model and the ground truth. Nevertheless, for some other test cases (i.e., Joint, Chair, Vase (Figure 12) and Fandisk in Table 1), our method may produce a more similar structure with that of the ground truth, thus leading to smaller E_v than the other methods. In brief, a better denoised result does not necessarily have a smaller E_v, but generally achieves a smaller MSAE. It is noteworthy that the MSAE is irrelevant in some application scenarios, where geometry accuracy is paramount.

In terms of computational time, as shown in Table 2, we

found that the first stage of our method accounts for the majority of the total computational time, since solving the global equation in Eq. 1 is time-consuming, and we empirically fixed α, β and σ_θ and ran 4 times (the unweighted form once and the weighted form three times) for each test model in our experiments. To reduce the computational time, an alternative would be to decrease the solving iterations for Eq. 1 by tuning α, β and σ_θ. Note that our approach denoised the Cat and Ateneam models (Figures 14 and 15) by skipping its last two stages.

TABLE 1
Quantitative comparisons among all the five mesh denoising methods.

Models	Methods	MSAE ($\times 10^{-2}$)	E_v ($\times 10^{-3}$)	Parameters
Joint (Figure 8) $\|V\|$: 5636 $\|F\|$: 11276	I	**0.342**	**13.98**	(0.047, 0.5, 15)
	II (local)	4.337	14.84	(5, 0.35, 10)
	II (global)	4.413	14.84	(0.1, 0.35, 10)
	III	0.629	15.70	Default1
	IV	3.321	15.23	Default2
	V	4.507	14.85	(0.35, 3, 0.35, 7)
Chair (Figure 9) $\|V\|$: 3301 $\|F\|$: 6606	I	**0.667**	**7.732**	(0.031, 0.27, 30)
	II (local)	17.06	8.071	(5, 0.35, 20)
	II (global)	15.33	8.013	(0.06, 0.4, 20)
	III	1.620	8.430	Default1
	IV	11.85	8.137	Default2
	V	19.19	8.067	(0.35, 4, 0.3, 10)
Table (Figure 10) $\|V\|$: 4556 $\|F\|$: 9108	I	**1.123**	8.300	(0.044, 0.3, 30)
	II (local)	50.17	6.827	(10, 0.35, 20)
	II (global)	53.36	6.811	(0.01, 0.4, 20)
	III	1.810	7.971	Default1
	IV	5.591	8.173	Default2
	V	53.90	7.384	(0.35, 7, 0.4, 12)
Vase (Figure 11) $\|V\|$: 3827 $\|F\|$: 7650	I	**0.415**	11.64	(0.06, 0.4, 10)
	II (local)	13.71	11.44	(5, 0.4, 12)
	II (global)	13.03	11.48	(0.05, 0.35, 12)
	III	1.199	12.94	Default1
	IV	3.686	12.53	Default2
	V	13.19	11.47	(0.35, 4, 0.4, 8)
Vase (Figure 12) $\|V\|$: 3827 $\|F\|$: 7650	I	**0.879**	**14.42**	(0.055, 0.25, 20)
	II (local)	111.3	20.76	(15, 0.4, 30)
	II (global)	90.50	20.73	(0.05, 0.45, 30)
	III	2.187	21.33	Default1
	IV	4.910	20.44	Default2
	V	109.3	21.24	(0.4, 8, 0.35, 15)
Fandisk (Figure 13) $\|V\|$: 6475 $\|F\|$: 12946	I	**0.440**	**8.309**	(0.03, 0.4, 10)
	II (local)	8.287	8.697	(5, 0.35, 10)
	II (global)	7.891	8.669	(0.02, 0.4, 10)
	III	0.655	9.373	Default1
	IV	4.354	9.090	Default2
	V	8.522	8.716	(0.35, 4, 0.35, 8)
Cat (Figure 14) $\|V\|$: 27109 $\|F\|$: 54192	I	**0.998**	2.819	(-, -, -)
	II (local)	6.208	2.606	(4, 0.45, 6)
	II (global)	5.863	2.606	(0.1, 0.4, 6)
	III	2.314	2.830	Default1
	IV	3.190	2.833	Default2
	V	5.440	2.607	(0.4, 3, 0.35, 4)
Ateneam (Figure 15) $\|V\|$: 7311 $\|F\|$: 14552	I	**6.663**	16.36	(-, -, -)
	II (local)	18.89	15.81	(4, 0.4, 8)
	II (global)	18.69	15.78	(0.1, 0.35, 8)
	III	13.16	16.53	Default1
	IV	13.35	16.11	Default2
	V	19.81	15.78	(0.35, 3, 0.35, 5)

TABLE 2
Timing statistics of our approach. The running time (seconds) was recorded on the same experimental computer with an Intel Core(TM) i7-3770 3.40-GHz CPU.

Models	Stage 1 Initial estimation	Stage 2 Feature operations	Stage 3 Vertex update	Total
Joint (Figure 8) $\|V\|$: 5636 $\|F\|$: 11276	2.461	0.51	0.105	3.076
Vase (Figure 11) $\|V\|$: 3827 $\|F\|$: 7650	1.602	0.334	0.028	1.964
Fandisk (Figure 13) $\|V\|$: 6475 $\|F\|$: 12946	2.174	0.583	0.524	3.281
Cat (Figure 14) $\|V\|$: 27109 $\|F\|$: 54192	7.098	-	-	7.098
Wilhelm (Figure 1) $\|V\|$: 43644 $\|F\|$: 85553	7.234	3.57	0.067	10.871

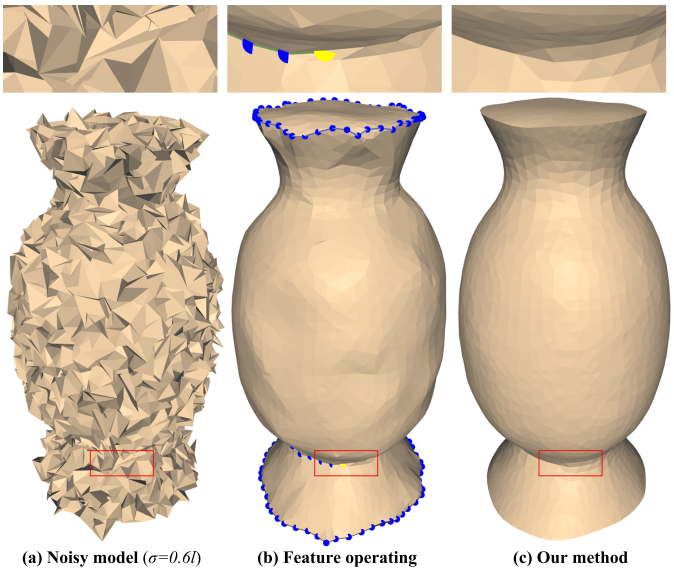


Fig. 19. Denoising a model corrupted with very heavy noise. In this case, our approach successfully forms two looping feature edges, but only connects a part of the middle looping feature edge (refer to (b)), thus leading to an unsatisfactory result (refer to (c)).

8 Discussion

As shown in the above section, our approach can generally achieve noticeably better results than the selected state of the art mesh denoising methods. In addition to the comparative results (§7), it is necessary to discuss the three stages of our method in detail.

- First, the initial estimation stage in our approach provides a sound estimated mesh from a noisy input. Though inspired by the work of [2], this strategy is new, as previous mesh denoising methods did not propose a similar one, to the best of our knowledge. It introduces a quadratic optimization which iteratively removes noise while preserving features. The whole mesh can be better shaped after this stage, meanwhile fold-backs, overshoots as well as degenerate triangles can be largely wiped out. Besides, existing mesh denoising methods can potentially take this stage as a vertex pre-filtering scheme before the methods themselves. Users can also tune the involved parameters if they found the default values cannot meet their requirements: a larger α or σ_θ would lead to smoother estimations given the same number of iterations, and β is generally smaller than α (e.g., $\beta = 0.1\alpha$) except at the first iteration.
- Second, the intermediate stage of our method consists of three steps: feature detection, identification and connection. The feature detection step is inspired by sparsity. However, our feature detection technique is widely different from the recent works [2], [3] (refer to §5.1). In fact, significant differences exist between our L_1-norm regularization and the L_1 optimization [40]: (a) the latter directly applies L_1 in optimization for point clouds, while our L_1

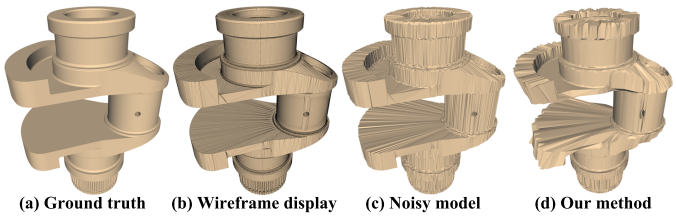


Fig. 20. Denoising a model with an extreme triangulation. Our method fails to achieve a desired denoised result.

formulation is to detect features on mesh shapes; (b) the optimized formulas are quite different. Specifically, the L_1 optimization in the work of [40] is to denoise point sets and generate well-organized points, and thus could facilitate other geometry processing, such as surface reconstruction. However, our L_1-norm regularized formula is to detect features on mesh shapes, which is one of the three operations in the feature operating stage. To the best of our knowledge, the last two steps (feature identification and connection) proposed in this work are the first-of-its-kind, since previous mesh denoising methods generally focus on detecting features and pay little attention to the refinement of the detected results. It should be noted that this stage can be potentially extended to cope with feature recognition problems on triangular meshes [46].

- Lastly, our proposed feature-aware vertex update algorithm utilizes local geometric information around each vertex. Triangles of the mesh can be further shaped by our vertex update, because the new position of a vertex is weighted averaged by itself and the centroid of its practical neighboring vertices. Similar to the vertex update in [1], [33], our algorithm is also iterative, fast and effective. Nevertheless, their vertex update relies on filtered face normals while ours needs to know which vertices are features or non-features. As a consequence, our approach directly acts on vertices, unlike methods that filter surface normals followed by vertex update (e.g., [1], [4]).

Similar to previous mesh denoising works, the input of our approach is a noisy mesh, and the noise typically has two types: raw noise and synthetic noise. Synthetic noise is used to generate different levels of noise (similar to image denoising), and raw noise arises from real scanned data. Synthetic noise is common in practice, for example, Gaussian White Noise in images. As mentioned in many previous mesh denoising works, a sound mesh denoising method is expected to well handle both synthetic and raw noise. Because: First, if a method could do a good job in suppressing synthetic noise, most likely it could reasonably suppress raw noise in scanned meshes. Second, for scanned meshes with raw noise, we typically cannot obtain their ground-truth models for comparison and quantitative analysis; by contrast, models added with synthetic noise do not need to face this issue. Hence, it is practically necessary and useful to utilize synthetic noise to test and validate a mesh denoising method. In fact, not only our work, almost all previous mesh denoising works have also done validation experiments on synthetic noise. In addition to other models with synthetic noise, we use one or two examples with large noise (Figures 12 and 19) to test the competence and limit of our method. Besides, we also validate the proposed method on raw scanned models (Figures 1, 16, 17 and 18), demonstrating the real application of our method. Raw scanned models may be obtained by first performing existing surface reconstruction techniques (e.g., Ball Pivoting [47]).

It is noteworthy that, in general our method can well handle noisy models with continuous features that can be connected into feature edges. However, noisy models with details or weak features (i.e., weak sharpness) could be challenging to denoise by employing the whole pipeline: details or weak features might be over-smoothed. Two successful examples of our approach are shown in Figures 1 and 13. An alternative is to skip the last two stages of our method, for example, Figures 14, 15 and 18 can be effectively and robustly denoised by only adopting the first stage.

Our method has certain limitations in spite of its demonstrated superiority over the state of the art approaches.

(1) Like existing methods [2], [4], our approach cannot well handle noisy models with an extreme triangulation, since vertices only exist at sharp features. A failure example is shown in Figure 20.
(2) If the noise level is very high, it is possible to smooth out some features at the first stage of our method, thus leading to challenges for the following stages. Therefore, it may produce undesired denoised results (a failure example is shown in Figure 19).
(3) Currently we fix open boundaries and corner vertices in our implementation (after the first stage). Such an assumption might be unreasonable for some cases.

9 Conclusion

In this paper we present a novel approach for robust feature-preserving mesh denoising. Given a noisy mesh input, our method first estimates an initial mesh, then performs feature detection, identification and connection, and finally, iteratively updates vertex positions based on the constructed feature edgs. As demonstrated in many experimental results, our approach can robustly denoise input mesh models with synthetic noise or raw scanned noise. The qualitative and quantitative comparisons between our method and the selected state of the art methods show that our approach can soundly outperform them in terms of both result quality and robustness.

Our work concentrates on mesh denoising, which aims to recover high-quality 3D models from meshes corrupted with raw or synthetic noise. An interesting research direction for future investigation would be the combination of the mesh denoising layer with the point set denoising layer and even possibly with the surface reconstruction layer. We believe that, with some non-trivial efforts, such a combination could potentially generate desirable complete shapes.

Acknowledgments

The authors would like to thank the in-part funding support from NIH 1R21HD075048-01A1, NSF IIS-1524782, and

Natural Science Foundation of China (NSFC) grant (No. 61328204).

REFERENCES

[1] Y. Zheng, H. Fu, O.-C. Au, and C.-L. Tai, "Bilateral normal filtering for mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 10, pp. 1521–1530, Oct 2011.

[2] L. He and S. Schaefer, "Mesh denoising via l0 minimization," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 64:1–64:8, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461965

[3] R. Wang, Z. Yang, L. Liu, J. Deng, and F. Chen, "Decoupling noise and features via weighted l1-analysis compressed sensing," *ACM Trans. Graph.*, vol. 33, no. 2, pp. 18:1–18:12, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2557449

[4] M. Wei, J. Yu, W. Pang, J. Wang, J. Qin, L. Liu, and P. Heng, "Bi-normal filtering for mesh denoising," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 21, no. 1, pp. 43–55, Jan 2015.

[5] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, and B. Lévy, "Geometric modeling based on polygonal meshes," in *Eurographics Tutorial*, 2008.

[6] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*. CRC press, 2010.

[7] G. Taubin, "A signal processing approach to fair surface design," in *Proc. of SIGGRAPH'95*, 1995, pp. 351–358.

[8] J. Vollmer, R. Mencl, and H. Müller, "Improved laplacian smoothing of noisy surface meshes," *Computer Graphics Forum*, pp. 131–138, 1999.

[9] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proc. of SIGGRAPH'99*, 1999, pp. 317–324.

[10] X. Liu, H. Bao, H.-Y. Shum, and Q. Peng, "A novel volume constrained smoothing method for meshes," *Graph. Models*, vol. 64, no. 3-4, pp. 169–182, May 2002. [Online]. Available: http://dx.doi.org/10.1006/gmod.2002.0576

[11] B. Kim and J. Rossignac, "Geofilter: Geometric selection of mesh filter parameters," *Comput. Graph. Forum*, pp. 295–302, 2005.

[12] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi, "Efficiently combining positions and normals for precise 3d geometry," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 536–543, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073226

[13] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proceedings of GRAPHITE'06*, 2006, pp. 381–389.

[14] Z.-X. Su, H. Wang, and J.-J. Cao, "Mesh denoising based on differential coordinates," in *Proc. of IEEE Int'l Conf. on Shape Modeling and Applications 2009*, June 2009, pp. 1–6.

[15] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Anisotropic feature-preserving denoising of height fields and bivariate data," in *In Graphics Interface*, 2000, pp. 145–152.

[16] Y. Ohtake, A. G. Belyaev, and I. A. Bogaevski, "Polyhedral surface smoothing with simultaneous mesh regularization," in *Proc. of the Geometric Modeling and Processing 2000*, 2000, pp. 229–237.

[17] U. Clarenz, U. Diewald, and M. Rumpf, "Anisotropic geometric diffusion in surface processing," in *Proc. of IEEE Conference on Visualization '00*, 2000, pp. 397–405.

[18] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher, "Geometric surface smoothing via anisotropic diffusion of normals," in *Proceedings of IEEE Conference on Visualization '02*, 2002, pp. 125–132.

[19] C. L. Bajaj and G. Xu, "Anisotropic diffusion of surfaces and functions on surfaces," *ACM Trans. Graph.*, vol. 22, no. 1, pp. 4–32, Jan. 2003. [Online]. Available: http://doi.acm.org/10.1145/588272.588276

[20] K. Hildebrandt and K. Polthier, "Anisotropic filtering of non-linear surface features," *Computer Graphics Forum*, vol. 23, no. 3, pp. 391–400, 2004. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2004.00770.x

[21] A. El Ouafdi and D. Ziou, "A global physical method for manifold smoothing," in *Proc. of IEEE International Conf. on Shape Modeling and Applications 2008*, June 2008, pp. 11–17.

[22] A. El Ouafdi, D. Ziou, and H. Krim, "A smart stochastic approach for manifolds smoothing," *Computer Graphics Forum*, vol. 27, no. 5, pp. 1357–1364, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2008.01275.x

[23] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV'98*, Jan 1998, pp. 839–846.

[24] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 950–953, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882368

[25] K.-W. Lee and W.-P. Wang, "Feature-preserving mesh denoising via bilateral normal filtering," in *Proc. of Int'l Conf. on Computer Aided Design and Computer Graphics 2005*, Dec 2005.

[26] C. C. L. Wang, "Bilateral recovering of sharp edges on feature-insensitive sampled meshes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 4, pp. 629–639, July 2006.

[27] G. Taubin, "Linear anisotropic mesh filtering," *IBM Research Report RC22213(W0110-051), IBM T.J. Watson Research*, 2001.

[28] Y. Ohtake, A. Belyaev, and I. Bogaevski, "Mesh regularization and adaptive smoothing," *Computer-Aided Design*, vol. 33, no. 11, pp. 789 – 800, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448501000951

[29] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh smoothing via mean and median filtering applied to face normals," in *Geometric Modeling and Processing, 2002. Proceedings*, 2002, pp. 124–131.

[30] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding," in *Computer Graphics International, 2003. Proceedings*, July 2003, pp. 28–33.

[31] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 943–949, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882367

[32] Y. Shen and K. Barner, "Fuzzy vector median-based surface smoothing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 3, pp. 252–265, May 2004.

[33] X. Sun, P. Rosin, R. Martin, and F. Langbein, "Fast and effective feature-preserving mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 925–938, Sept 2007.

[34] X. Sun, P. L. Rosin, R. R. Martin, and F. C. Langbein, "Random walks for feature-preserving mesh denoising," *Computer Aided Geometric Design*, vol. 25, no. 7, pp. 437 – 456, 2008, solid and Physical Modeling Selected papers from the Solid and Physical Modeling and Applications Symposium 2007 (SPM 2007) Solid and Physical Modeling and Applications Symposium 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167839608000307

[35] H. Zhang, C. Wu, J. Zhang, and J. Deng, "Variational mesh denoising using total variation and piecewise constant function space," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 21, no. 7, pp. 873–886, July 2015.

[36] H. Fan, Y. Yu, and Q. Peng, "Robust feature-preserving mesh denoising based on consistent subneighborhoods," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 2, pp. 312–324, 2010.

[37] Z. Bian and R. Tong, "Feature-preserving mesh denoising based on vertices classification," *Computer Aided Geometric Design*, vol. 28, no. 1, pp. 50 – 64, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167839610001111

[38] J. Wang, X. Zhang, and Z. Yu, "A cascaded approach for feature-preserving surface mesh denoising," *Computer-Aided Design*, vol. 44, no. 7, pp. 597 – 610, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448512000516

[39] L. Zhu, M. Wei, J. Yu, W. Wang, J. Qin, and P.-A. Heng, "Coarse-to-fine normal filtering for feature-preserving mesh denoising based on isotropic subneighborhoods," *Computer Graphics Forum*, vol. 32, no. 7, pp. 371–380, 2013. [Online]. Available: http://dx.doi.org/10.1111/cgf.12245

[40] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "L1-sparse reconstruction of sharp point set surfaces," *ACM Trans. Graph.*, vol. 29, no. 5, pp. 135:1–135:12, Nov. 2010. [Online]. Available: http://doi.acm.org/10.1145/1857907.1857911

[41] "Intel math kernel library (mkl), https://software.intel.com/en-us/intel-mkl," 2015.

[42] O. Sorkine, "Differential representations for mesh processing," *Computer Graphics Forum*, vol. 25, no. 4, pp. 789–807, 2006. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2006.00999.x

[43] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[44] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and mathematics III*. Springer, 2003, pp. 35–57.

[45] M. Desbrun, "Processing irregular meshes," in *Proc. of Shape Modeling International 2002*, 2002, pp. 157–158.
[46] H. S. Kim, H. K. Choi, and K. H. Lee, "Feature detection of triangular meshes based on tensor voting theory," *Computer-Aided Design*, vol. 41, no. 1, pp. 47 – 58, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448508002297
[47] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, Oct. 1999. [Online]. Available: http://dx.doi.org/10.1109/2945.817351

Xuequan Lu is a Ph.D. candidate in the College of Computer Science and Technology at Zhejiang University, China. He earned his B.S. degree in Computer Science from Northwest A&F University (China). His research interests include geometry processing and modeling, crowd-related simulation, and computer animation.

Zhigang Deng is currently an Associate Professor of Computer Science at the University of Houston (UH) and the Founding Director of the UH Computer Graphics and Interactive Media (CGIM) Lab. His research interests include computer graphics, computer animation, virtual human modeling and animation, and human computer interaction . He earned his Ph.D. in Computer Science at the Department of Computer Science at the University of Southern California in 2006. Prior that, he also completed B.S. degree in Mathematics from Xiamen University (China), and M.S. in Computer Science from Peking University (China). Besides the CASA 2014 general co-chair and SCA 2015 general co-chair, he currently serves as an Associate Editor of several journals including Computer Graphics Forum, and Computer Animation and Virtual Worlds Journal.

Wenzhi Chen was born in 1969, received his Ph.D. degree from Zhejiang University, Hangzhou, China. He is now a Professor and Ph.D. supervisor of college of Computer Science and Technology of Zhejiang University. His areas of research include computer graphics, computer architecture, system software, embedded system and security.