Robust Mesh Denoising via Vertex Pre-filtering and L_1-Median Normal Filtering

Xuequan Lu[a], Wenzhi Chen[a], Scott Schaefer[b]

[a] *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
[b] *S. Schaefer is with the Department of Computer Science, Texas A&M University, Texas, USA.*
E-mail: schaefer@cs.tamu.edu

Abstract

We propose a robust and effective mesh denoising approach consisting of three steps: vertex pre-filtering, L_1-median normal filtering, and vertex updating. Given an input noisy mesh model, our method generates a high quality model that preserves geometric features. Our approach is more robust than state of the art approaches when denoising models with different levels of noise and can handle models with irregular surface sampling.

Keywords: Mesh denoising, Feature preserving, Vertex pre-filtering, L_1-median normal filtering

1. Introduction

Mesh denoising has been widely used for geometry modeling and processing. Some model acquisition methods such as laser range scanners or vision-based reconstruction algorithms inherently produce noisy models due to noise in the input data. Before these raw noisy 3D models can be used for many applications, we need to produce cleaned versions via mesh denoising.

The problem of anisotropic mesh denoising has attracted lots of attention in recent years. However, the main problem faced by these methods is distinguishing sharp features from noise. This issue becomes even more challenging as the noise level increases or if the surface contains an irregular triangulation. In addition, features with small dihedral angles (i.e., shallow features) such as the zoomed in region of Figure 10 can be difficult to reconstruct.

Recent mesh denoising methods (Zheng et al., 2011; He and Schaefer, 2013; Wei et al., 2015), designed to preserve sharp features of noisy input models, have achieved noticeable successes. However, these methods can have difficulty denoising surfaces that contain different levels of noise or irregular surface sampling. Some methods are less robust to noise such as Zheng et al. (2011) and Wei et al. (2015) (see Section 7), while other methods (He and Schaefer, 2013) may be robust but can over-sharpen the result.

In this paper, we propose a robust and effective mesh denoising approach. Our method consists of three steps: vertex pre-filtering, L_1-median normal filtering, and vertex updating. The use of a *vertex pre-filtering* step significantly reduces the noise of the input model and handles irregular triangulations through a region-based pre-filter. The second step is to filter face normals using our *L_1-median filter*, which preserves both strong and shallow features. Finally, the *vertex update* step updates vertex positions to be compatible with the filtered normals. As shown in Figure 1, our approach can generate a higher quality result than current state-of-the-art approaches (He and Schaefer, 2013; Wang et al., 2014; Wei et al., 2015) particularly in the neck region.

2. Related Work

Mesh denoising is a vast field. We review only the most related works here but refer readers to Botsch et al. (2010) for a comprehensive review of mesh denoising and smoothing techniques.

Isotropic mesh denoising. Isotropic methods attempt to create smooth surfaces everywhere and, consequently, do not preserve sharp features in the input. Laplacian smoothing (Vollmer et al., 1999) is a simple and fast smoothing algorithm but suffers from surface shrinkage and feature blurring. Desbrun et al. (1999) proposed an implicit fairing method for irregular meshes using diffusion and curvature flow. Later, researchers have proposed a volume-preserving

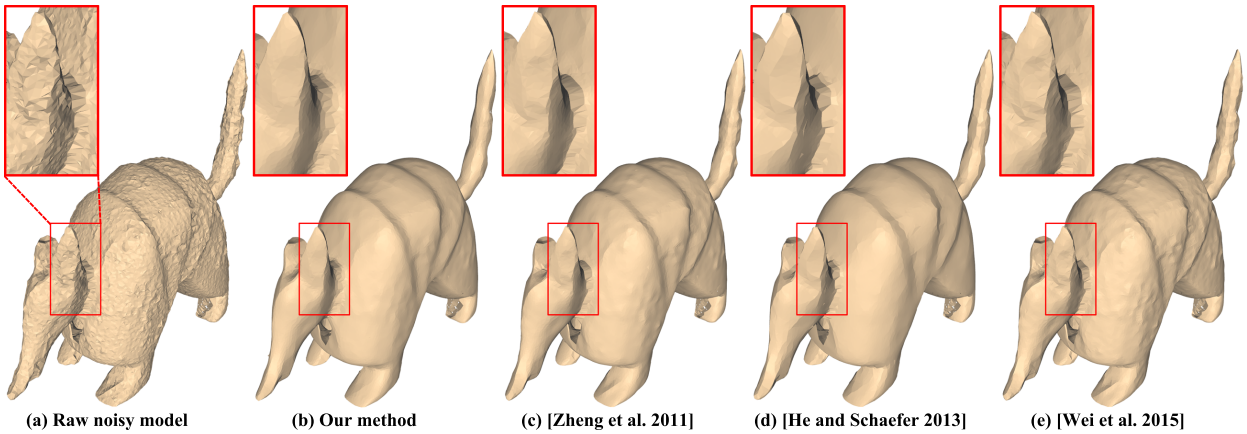


Figure 1: The denoised results of a raw Armadillo model, which is reconstructed from scanned point data. Note that the state of the art methods generate visually hackly results in the neck region.

smoothing method (Liu et al., 2002) and a mesh filtering framework including low-pass, high-pass, bandpass, and notch filters with various exaggeration and attenuation options (Kim and Rossignac, 2005). Recently, researchers proposed several isotropic methods based on global differential information (Nealen et al., 2006; Su et al., 2009).

Anisotropic mesh denoising. Given that isotropic methods cannot preserve sharp features, many researchers have turned their attention to anisotropic approaches. These anisotropic approaches can be divided into two categories. The first category consists of methods that only filter the vertices of the model. Many of these anisotropic denoising algorithms are based on differential information (Desbrun et al., 2000; Clarenz et al., 2000; Tasdizen et al., 2002; Bajaj and Xu, 2003; Hildebrandt and Polthier, 2004; He and Schaefer, 2013). Fleishman et al. (2003) presented a bilateral mesh denoising method to filter the vertices of input meshes in the normal direction using local neighborhoods. El Ouafdi et al. (2008) proposed a probabilistic approach for one-stage mesh smoothing by formulating the problem as tracking the transition probability density functions of an underlying random process. Solomon et al. (2014) described a generalization of the bilateral filter that can be applied to feature-preserving smoothing of signals on images, meshes, and other domains within a unified framework. Recently, Wang et al. (2014) proposed a denoising technique via weighted L_1-analysis based on compressed sensing. Lu et al. (2016) perform mesh denoising by explicitly identifying features to preserve.

The second category of methods (Taubin, 2001; Yagou et al., 2002; Jones et al., 2003; Shen and Barner, 2004; Sun et al., 2007, 2008; Zheng et al., 2011; Zhang et al., 2015a,b) first filter normals and then update vertex positions using those filtered normals. Taubin (2001) introduced an early two-stage method. Later, researchers used the mean-median (Yagou et al., 2002) and fuzzy median (Shen and Barner, 2004) filters to estimate face normals. Researchers have also proposed averaging neighboring face normals in a weighted manner with data dependent weights (Sun et al., 2007; Zheng et al., 2011) or probabilities derived from random walks (Sun et al., 2008). Recently two different normal filters were proposed using total variation (Zhang et al., 2015a) and the joint bilateral filter (Zhang et al., 2015b).

Vertex/Face classification. To better preserve geometric features, several approaches have been proposed to classify either the vertices or the faces of an input noisy model into different types before the mesh denoising process (Fan et al., 2010; Bian and Tong, 2011; Wang et al., 2012; Wei et al., 2015). However, the classification results are typically sensitive to the noise level, and thus potentially lead to fragility. Lipman et al. (2007) introduced a locally optimal projection operator used to denoise point sets.

3. Approach Overview

Figure 2 provides an overview of our method's pipeline. Our first step is pre-filtering (Section 4), which provides a good initialization for our L_1-median filter of the surface normals (Section 5). Given those filtered normals, our method then updates the vertex positions to create a smoothed version of the input surface (Section 6). Note that the proposed vertex pre-filtering technique distinguishes between noisy meshes and meshes with both irregular surface

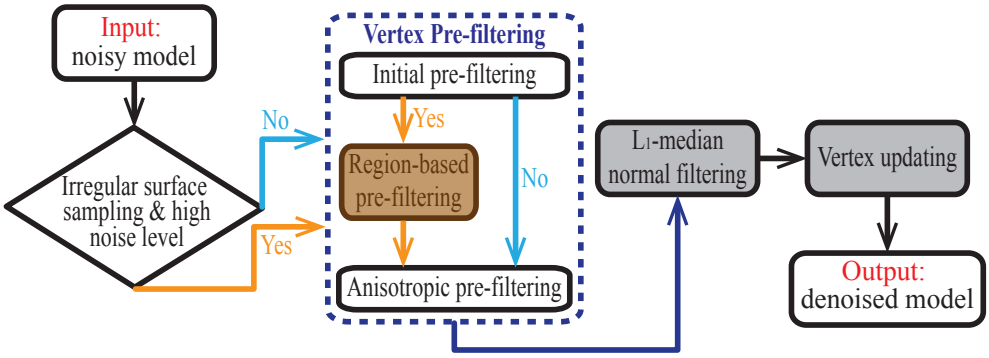


Figure 2: The pipeline of our mesh denoising approach.

sampling and large noise (see details in Section 4.2). The latter case involves an extra region-based pre-filtering step (the "Yes" branch in Figure 2).

In following sections, we assume the input data is a noisy mesh model obtained from scanning devices or by adding synthetic noise (Gaussian noise with zero mean and standard deviation σ measured as a multiple of the average edge length ℓ) to a model. We normalize the input model to a unit box. We denote by p_i the position of the i-th vertex and by n_j, c_j the normal and the centroid of the j-th face respectively.

4. Vertex Pre-filtering

Our first step is to pre-filter the noisy input mesh to substantially decrease the noise, thus ensuring a good initialization for the subsequent steps leading to more accurate face normals. Our pre-filtering method differentiates between noisy surfaces and surfaces with both irregular sampling and high noise. For the latter case, we use an additional region-based pre-filtering step.

4.1. Initial Pre-Filtering

We begin our pre-filtering stage by attempting to remove folded faces through an isotropic smoothing step. These folded faces appear in nearly flat areas of the surface and have large dihedral angles, which are typically undesirable. We formulate this step as a least squares minimization

$$\arg\min_{\{\tilde{p}_i\}} \sum_i \|\tilde{p}_i - p_i\|_2^2 + \alpha \sum_e \|S(e)\|_2^2, \tag{1}$$

where $\tilde{p}_i$ is the unknown position of the i-th vertex, p_i is the position of the i-th vertex in the input, α is a user-specified weight that is empirically set in the range of 0.05 to 0.2 in our experiments, and $S(e)$ is a shaping term applied to an edge e. The goal of $S(e)$ is to help unfold very noisy faces and reshape them to be more regular. Assume the edge e is shared by two triangles: one with vertices p_{e1}, p_{e2}, and p_{e3}, and the other with vertices p_{e1}, p_{e3}, and p_{e4}. We use the triangle regularization term from He and Schaefer (2013) as our shaping metric, which is defined as $S(e) = p_{e1} - p_{e2} + p_{e3} - p_{e4}$. He and Schaefer (2013) use this regularization term in the context of an L_0 minimization whereas our optimization is only a prefilter and we preserve geometric features through our normal filter (see Section 5).

4.2. Region-Based Pre-Filtering

For very noisy meshes with irregular sampling, the simple isotropic smoothing step (Section 4.1) is not sufficient to remove enough noise from the input. Such irregular sampling can occur in real-world meshes when multiple scans of an object overlap, which can increase sampling density in those overlapping locations. To determine if a surface requires our additional, region-based pre-filtering, we examine the connectivity to determine if the sampling is irregular. If so, we estimate the noise level of the original, noisy input surface by calculating the proportion of

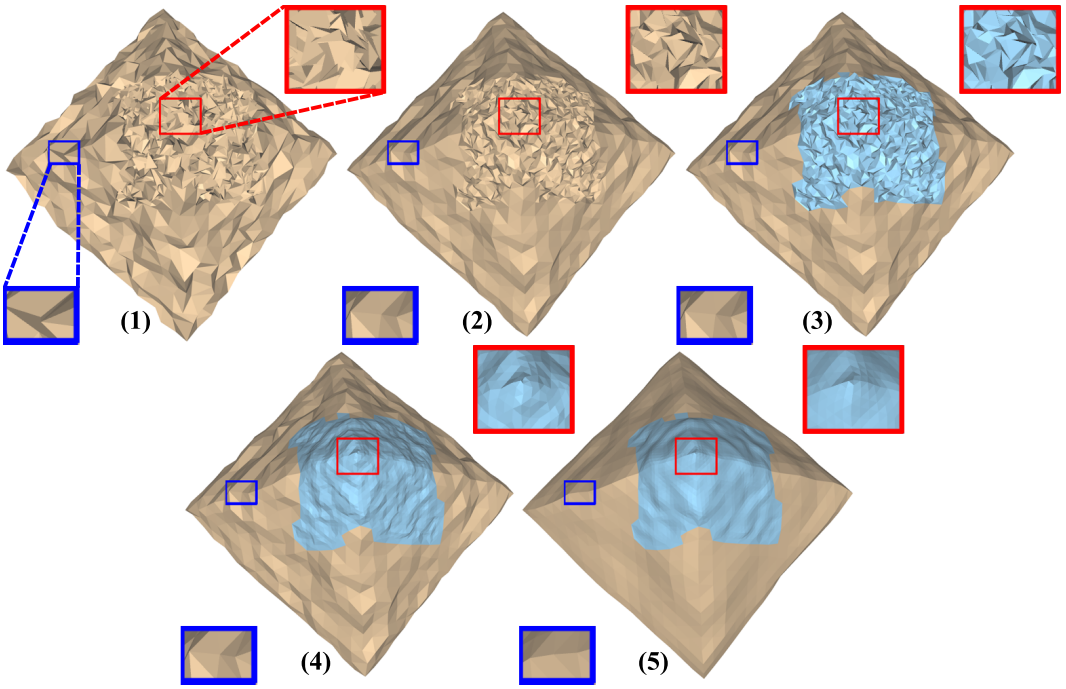


Figure 3: Example illustration of region-based pre-filtering (Section 4.2): (1) a noisy model with irregular surface sampling (see the red and blue rectangles), (2) a rough initialization using our initial pre-filter (Section 4.1), (3) region-based clustering for (2), (4) Laplacian smoothing for noisier places (blue) of (3), (5) anisotropic pre-filtering (Section 4.3) for (4).

edges whose dihedral angles are greater than a threshold, which is empirically set to 130° in our work. If this ratio is reasonably large (e.g., $\geq 10\%$), then we perform this additional region-based pre-filtering step.

First we begin with the output of the initial pre-filtering in Section 4.1 and perform a K-means clustering to cluster faces into regions based on the similarity of the area of the triangles and the distance between face centroids. Hence, we cluster triangles in a high dimensional space by appending the triangle area onto the coordinates of the triangle centroid. Figure 3-(3) shows an example result of region-based clustering, where two clusters are generated. Ideally, the number of clusters (K) in this step should be consistent with the number of different sampling densities of the model. Note that various mesh segmentation algorithms (Shamir, 2008) could be used as the alternative for the region-based clustering, but we choose the K-means clustering due to its efficiency and effectiveness.

In general, the regions with small triangles will have larger amounts of noise in the normals than large triangles since small geometric deviations lead to large deviations in orientation (refer to Figure 3-(3)). In this case, uniformly smoothing the whole surface tends to produce unsatisfactory results (e.g., flipped triangles). Instead, our method handles such clusters (noiser regions) using a form of Laplacian smoothing (Chuang and Kazhdan, 2011), defined as follows:

$$\arg\min_{\tilde{P}} \left\|\tilde{P} - P\right\|_F^2 + \beta \left\|L\tilde{P}\right\|_F^2, \tag{2}$$

where $\tilde{P} = (\tilde{p}_1, \cdots, \tilde{p}_n)^T$ is the vectorized form of unknown vertex positions within a single cluster, $P = (p_1, \cdots, p_n)^T$ is the vectorized form of the corresponding vertex positions of the rough initialized mesh, L is the uniformly weighted Laplacian matrix, and β (we use $\beta = 6$) is a user-defined smoothness parameter. Figure 4 demonstrates that only initial pre-filtering and anisotropic pre-filtering fails to produce decent pre-filtering results when handling very noisy meshes with irregular sampling.

4.3. Anisotropic Pre-Filtering

While our initial pre-filtering and region-based pre-filtering can remove folded faces, these isotropic filters tend to blur geometric features, which provides a poor initialization for subsequent steps (see Figure 5). Therefore, we

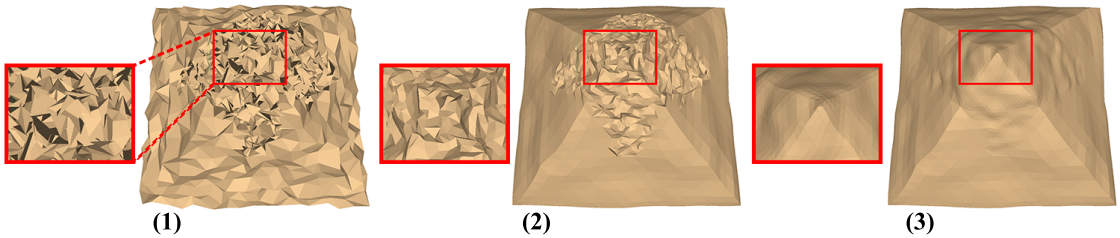


Figure 4: The importance of region-based pre-filtering: (1) noisy input with irregular surface sampling, (2) only initial pre-filtering (Section 4.1) and anisotropic pre-filtering (Section 4.3), (3) combined initial pre-filtering, region-based pre-filtering and anisotropic pre-filtering.

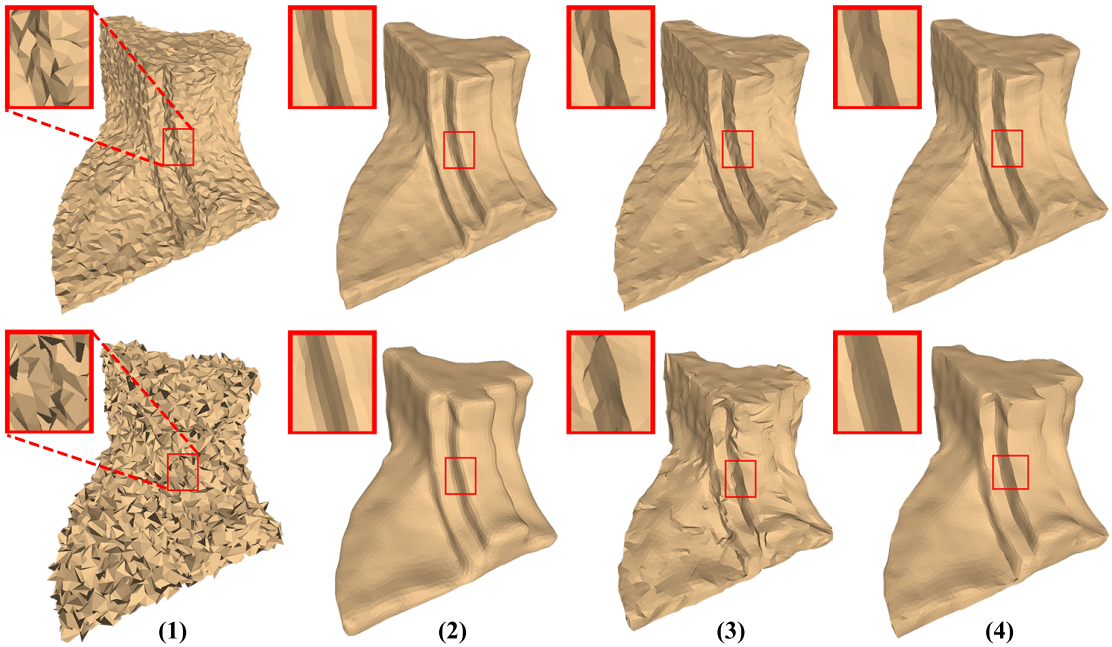


Figure 5: Vertex pre-filtering with different levels of noise: (1) noisy input ($\sigma = 0.2\ell$ top, $\sigma = 0.6\ell$ bottom), (2) only initial pre-filtering (Section 4.1), (3) only anisotropic pre-filtering (Section 4.3), (4) combined initial pre-filtering and anisotropic pre-filtering. As shown in (4), using a combination of our initial pre-filter (Section 4.1) and anisotropic pre-filter (Section 4.3) preserves features the best.

perform a subsequent anisotropic filtering step to help preserve geometric features written as

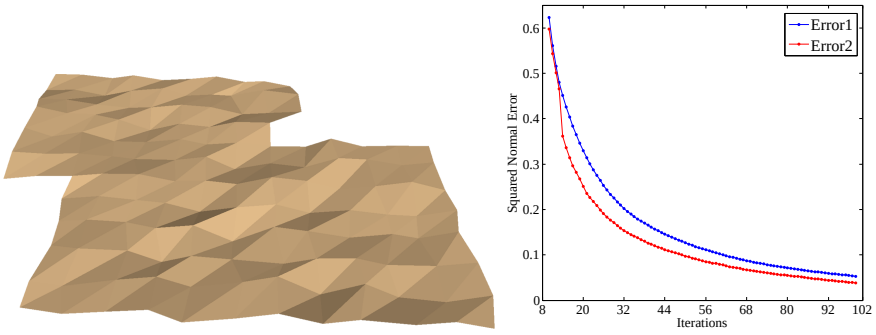
$$\arg\min_{\{\tilde{p}_i\}} \sum_i \|\tilde{p}_i - p_i\|_2^2 + \alpha \sum_e w_e \|S(e)\|_2^2, \tag{3}$$

where p_i are the input vertices that come either from the output of the initial pre-filter (Section 4.1) or the region-based pre-filter (Section 4.2), and $S(e)$ is the shape term from Section 4.1.

w_e, the weight for edge e, is a non-negative, decreasing function with respect to the angle between the normals of the edge-sharing faces. In this work, we design w_e as an exponential function, defined as follows:

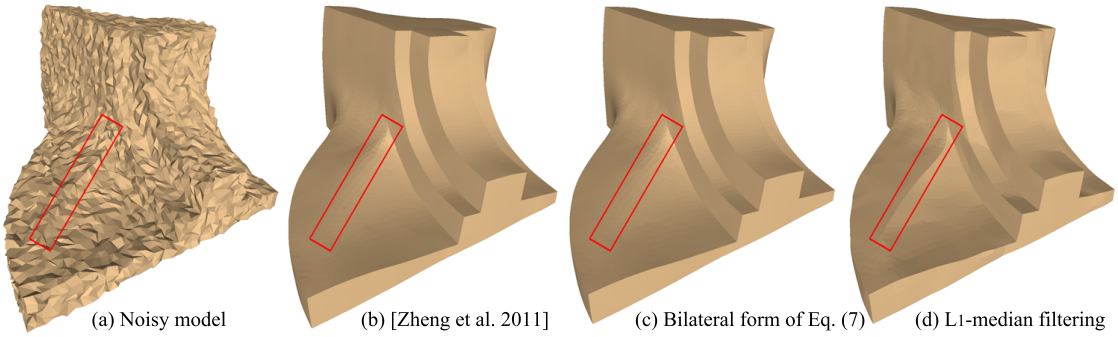
$$w_e = b^{-\left(\frac{1-\cos(\theta)}{1-\cos(\sigma_\theta)}\right)} \tag{4}$$

where b is a constant number (we have empirically found that $b = \sqrt{3}$ works well), θ is the angle between the normals of two edge-sharing faces, and σ_θ is used to scale the similarity of the face normals (in our experiments, σ_θ is usually set to 30°). Note that the weights w_e are small for edges with large dihedral angles (possible features) and large for edges with small dihedral angles. Hence, this optimization will smooth the mesh while retaining its features. We



(a) (b)

Figure 6: Isotropic normal smoothing when using two different ways: (1) an isotropic patch corrupted with noise, (2) squared error ($\sum_i \left\|n_i - n_{avg}\right\|^2$) computed at each iteration, where n_{avg} is the average normal of this patch. "Error1" and "Error2" are computed with adding a small constant to the denominator in ω_{ij} and our strategy, respectively. Smaller error values mean faster isotropic normal smoothing.



(a) Noisy model (b) [Zheng et al. 2011] (c) Bilateral form of Eq. (7) (d) L1-median filtering

Figure 7: Applying different normal filters to the same vertex pre-filtering result of (a). Our L_1-median filter better preserves shallow features (highlighted in red) compared with the bilateral filters in (b) and (c).

perform this optimization iteratively (typically between 0-12 times depending on the noise level) using the output of one iteration as the input to the next iteration to smooth the surface.

5. L_1-Median Normal Filter

After pre-filtering, we now estimate the normals of the denoised surface using an L_1-median filter. The L_1 **median** of a set of data points $\{y_i\}$ is the point x that minimizes the sum of Euclidean distances to the y_i.

$$\arg\min_x \sum_{y_i} \|x - y_i\|_2, \tag{5}$$

Unlike a typical average, the L_1 median is robust to outliers and noise in the data (Lipman et al., 2007). Since the face normals still contain noise after vertex pre-filtering, we propose to filter these normals using the L_1-median filter.

We measure the face normal difference between the i-th face and its neighboring faces (we use vertex adjacency to determine neighbors, as suggested by Zheng et al. (2011); Wei et al. (2015)) with a localized version of Equation (5).

$$\sum_{j \in N(i)} a_j \phi\left(\frac{1-\cos(\gamma_{ij})}{1-\cos(\sigma_\gamma)}\right) \phi\left(\frac{\|c_i - c_j\|_2}{\sigma_c}\right) \left\|n_i - n_j\right\|_2, \tag{6}$$

where $N(i)$ denotes the neighboring faces of the i-th face, a_i/c_i are the area/centroid of the i-th face, and γ_{ij} is the angle between n_i and n_j. σ_γ is the angle threshold parameter (30° by default) which can be adjusted by users. $\phi(x) = e^{-x^2}$ is the Gaussian function and we set $\sigma_c = \frac{3}{2}d$, where d is the average distance between centroids of adjacent faces.

Optimization. The nonlinear nature of Equation (6) makes optimization with respect to n_i difficult. Therefore, we develop an iterative form of the optimization. Differentiating Equation (6) with respect to n_i and setting the derivative equal to 0 yields a local update formula (see Appendix for the derivation procedure).

$$n_i^{t+1} = \frac{\sum_{j \in N(i)} \omega_{ij} n_j^t}{\sum_{j \in N(i)} \omega_{ij}} \tag{7}$$

where n_i^{t+1} is the normal of the i-th face at the $t+1$-th iteration and $\omega_{ij} = \frac{a_j \phi\left(\frac{1-\cos(\gamma_{ij})}{1-\cos(\sigma_\gamma)}\right) \phi\left(\frac{\|c_i - c_j\|_2}{\sigma_c}\right)}{\|n_i - n_j\|_2}$. While Equation (7) works well, ω_{ij} is undefined when $n_i = n_j$. Hence, when $\|n_i - n_j\| < 10^{-3}$ we switch to the bilateral weights $\omega_{ij} = a_j \phi\left(\frac{1-\cos(\gamma_{ij})}{1-\cos(\sigma_\gamma)}\right) \phi\left(\frac{\|c_i - c_j\|_2}{\sigma_c}\right)$, which we found leads to faster isotropic normal smoothing than adding a small constant to the denominator in ω_{ij} (Figure 6).

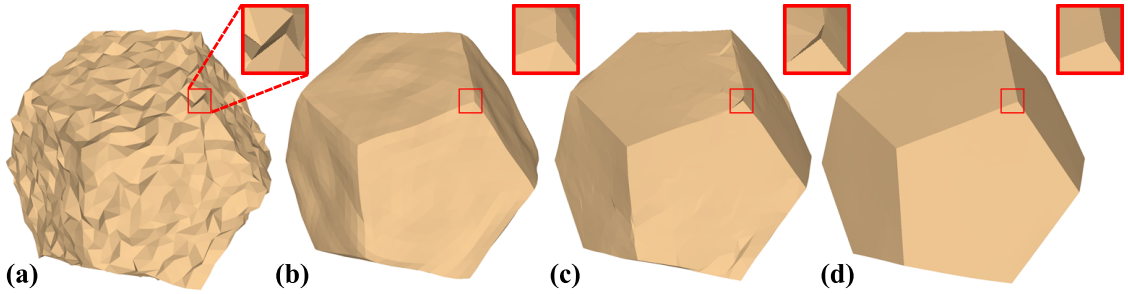


Figure 8: The importance of both pre-filtering and L_1-median filtering. (a) a noisy dodecahedron, (b) the result of only applying pre-filtering, (c) the result of only applying L_1-median filtering, (d) the result of applying both filters. Applying both filters produces better results than either filter individually.

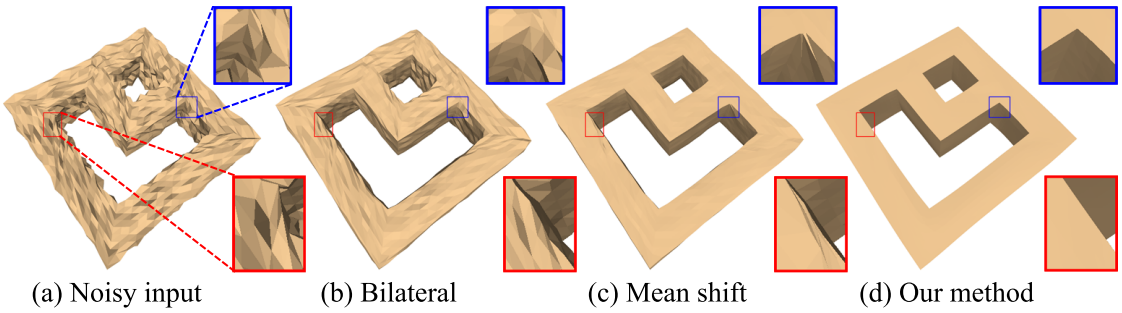


Figure 9: Denoising results of Solomon et al. (2014) and our method. (b) and (c) are the results of the generalized bilateral and mean shift filtering of Solomon et al. (2014).

Equation (7) can be regarded as a regularized mean shift for robustly filtering normals. ω_{ij} acts as a kernel function. The mean shift filter results from iterative bilateral filtering (Solomon et al., 2014; Weijer and Boomgaard, 2001). Note that Equation (7) bears some similarity to a bilateral filter if $\|n_i - n_j\|_2$ is set to 1 (indeed we use this form when $\|n_i - n_j\| < 10^{-3}$). We tested always using these bilateral weights versus our L_1-median filter as well as the bilateral filter in Zheng et al. (2011). Figure 7 shows that our L_1-median filter outperforms both bilateral filters when attempting to preserve shallow features.

To demonstrate the necessity of both vertex pre-filtering and L_1-median normal filtering, we compared the denoised results by only applying one of the filters and by applying both filters. As shown in Figure 8, applying both filters produces better results than either filter individually.

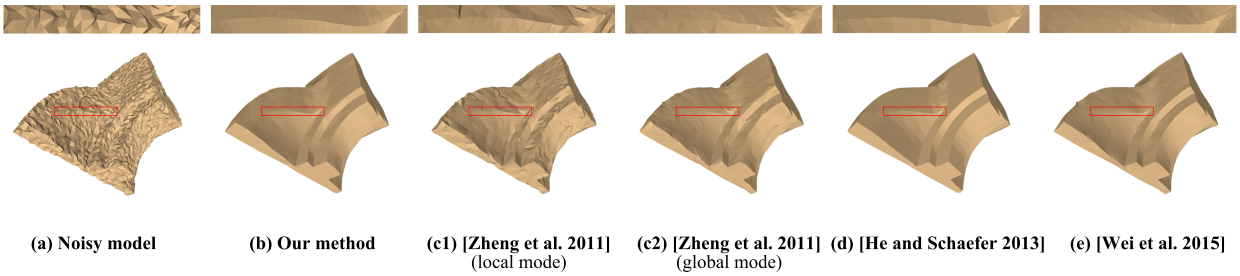


Figure 10: Shallow feature preservation during the mesh denoising process ($\sigma = 0.2\ell$). Note that the result of our method is smoother than that of He and Schaefer (2013) (refer to the top zoomed region).

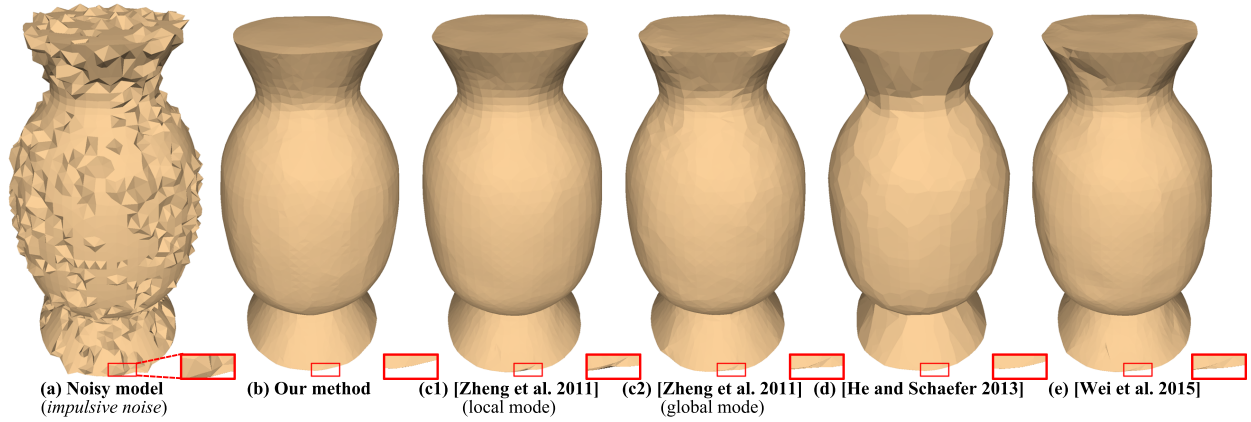


Figure 11: Denoised results of the Vase model whose 20% vertices are corrupted with 0.6 mean edge length impulsive random noise.

6. Vertex Update

After normal filtering, we use the vertex update algorithm from Sun et al. (2007) to iteratively update vertex positions based on the filtered normals.

$$p_i^{k+1} = p_i^k + \frac{1}{|NF(i)|} \sum_{j \in NF(i)} n_j (n_j \cdot (c_j^k - p_i^k)), \tag{8}$$

where p_i^k is the vertex at the (k)-th iteration, $NF(i)$ is the neighboring faces of the i-th vertex, and $|NF(i)|$ is the number of neighboring faces of the i-th vertex. See Sun et al. (2007) for proof of convergence.

7. Experimental Results

We tested our approach on a variety of 3D mesh models with either raw or synthetic noise (different levels of Gaussian noise with zero mean and standard deviation σ), including irregularly sampled meshes. While synthetic noise is artificial, it allows us to easily generate different levels of noise and obtain ground-truth models for quantitative analysis, similar to almost all previous mesh denoising works. We also compared our approach with a number of state of the art mesh denoising approaches including Zheng et al. (2011), He and Schaefer (2013), Wang et al. (2014), Solomon et al. (2014), and Wei et al. (2015). We obtained the source code from the authors of Zheng et al. (2011) for the comparisons; several results (in Figures 9, 12 and 13) were provided by those authors as well; finally, we implemented He and Schaefer (2013) and Wei et al. (2015) based on their original papers.

Parameter choices. In our comparison experiments, we used the following parameter sets for the above methods: ours = (α, σ_θ, iterations of anisotropic pre-filtering, σ_γ, iterations of normal filtering, iterations of vertex update); Zheng et al. (2011) (local) = (normal filtering iterations, σ_s, vertex update iterations); Zheng et al. (2011) (global) =

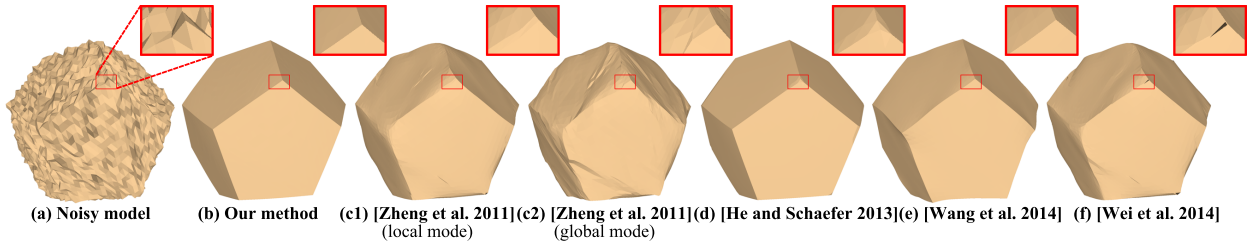


Figure 12: Denoised results of the noisy Dodecahedron model ($\sigma = 0.3\ell$). Note that the sharp edges generated by Wang et al. (2014) are not straight.

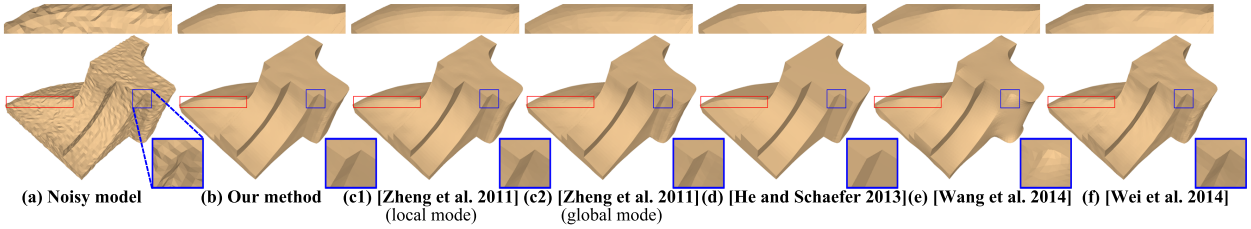


Figure 13: Denoised results of the Fandisk model ($\sigma = 0.1\ell$). Wang et al. (2014) over-smooth some features (see blue box).

(λ, σ_s, vertex update iterations); He and Schaefer (2013) = (μ, α_0, λ); Wang et al. (2014) = (τ); and Wei et al. (2015) = (σ_{s1}, n_1, σ_{s2}, n_2). We used the default parameter values for He and Schaefer (2013) and Wang et al. (2014) specified in their papers. For the remaining methods, we carefully tuned the parameters to generate the best visual results for each model in our experiments. Specifically, in our method, α, σ_θ, the iterations of anisotropic pre-filtering, σ_γ, the iterations of normal filtering and the iterations of vertex update are within [0.05, 0.2], [20, 50], [0, 12], [20, 50], [5, 100] and [5, 100], respectively. In Zheng et al. (2011), σ_s and the vertex update iterations are in the ranges of [0.2, 0.5] and [5, 100] for both local and global modes. The normal filtering iterations in Zheng et al. (2011) (local) is within [3, 60] and λ in Zheng et al. (2011) (global) is within [0.001, 0.1]. In Wei et al. (2015), σ_{s1}, n_1, σ_{s2} and n_2 are in the ranges of [0.2, 0.5], [3, 45], [0.2, 0.5] and [3, 50]. In these methods, the parameters generally increase as the noise level increases. The specific parameter values of each method for some models are summarized in Table 1. In addition to the denoised results in this paper, we have included more results in the supplemental document.

Denoising meshes with different levels of noise. Most existing anisotropic methods can produce reasonable results on models with moderate noise (e.g., high-quality scanned data or the standard deviation of the noise $\sigma \leq 0.3\ell$) but struggle with large noise (e.g., low-quality scanned data or $\sigma > 0.3\ell$). Figure 9 shows a comparisons with the generalized bilateral and mean shift filter (Solomon et al., 2014) where our method can remove the noise from the input effectively. Figures 10 and 13 show that our approach can preserve both sharp and shallow features on input noisy meshes when the noise level is comparatively low ($\sigma \leq 0.2\ell$). Figure 12 demonstrates our results on a platonic solid where the desired shape is planar everywhere except at sharp features. Figure 11 shows that our method is also robust to impulse noise. Figure 14 demonstrates the performance of our method on denoising models with high levels of noise ($\sigma \geq 0.5\ell$) compared to other methods that fail to achieve similar quality results.

Denoising meshes with irregular surface sampling. Meshes with irregular polygon densities pose a difficult problem for most denoising methods. Figure 16 shows a comparison of our method on a model with irregular sampling and relatively small amounts of noise ($\sigma \leq 0.2\ell$). While most methods perform poorly with irregular sampling, our method and He and Schaefer (2013) perform much better. However, the method of He and Schaefer (2013) tends to produce sharp features even in smooth areas like the eye-lids. Figure 15 shows another example of irregular sampling with large amounts of noise ($\sigma \geq 0.4\ell$). Again, the majority of methods fail to handle the irregular sampling well, and our method creates the highest quality results.

Denoising real 3D scanned data. We also compared these approaches on real scanned data. Figure 1 shows the results of the methods applied to a raw scan of an armadillo model. Our method produces a higher quality result than the other methods particularly around the neck where the triangle density is non-uniform due to missing data in the

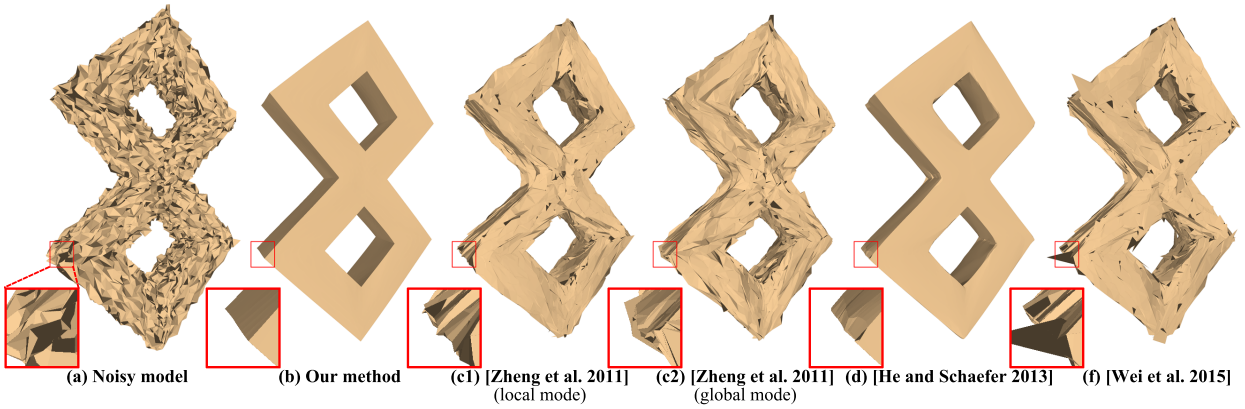


Figure 14: Denoised results of the Double-torus model contaminated with large noise ($\sigma = 0.5\ell$).

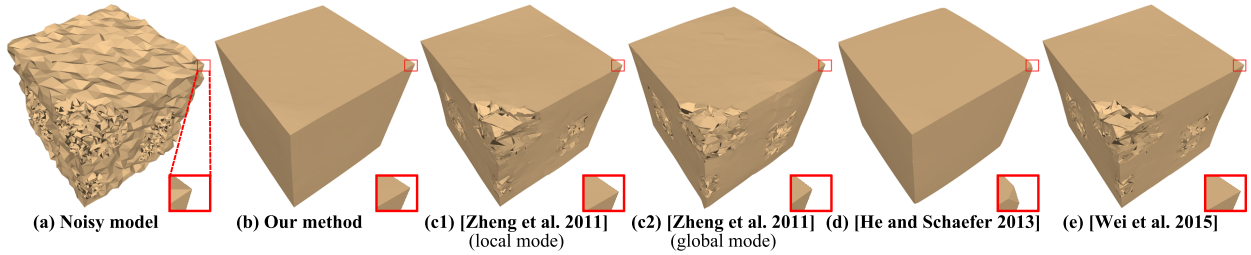


Figure 15: Denoised results of an irregularly sampled cube with large noise ($\sigma = 0.4\ell$). Note that (d) does not recover the corners exactly.

original scan. Figure 17 shows another example of a scanned model denoised where our method produces cleaner sharp features than the other techniques.

Quantitative comparisons. We also compared our approach to other methods using two quantitative metrics, shown in Table 1. We use E_v (the L_2 vertex-based error) and MSAE (the mean squared angular error) advocated by previous work (Zheng et al., 2011; Wei et al., 2015). E_v is a numerical error measurement between the denoised mesh and the ground-truth. E_v generally, but not always, is in agreement with visual comparison results (Sun et al., 2007). MSAE measures the mean square angular error between the face normals of the denoised mesh and those of the ground truth.

As shown in Table 1, in terms of E_v, the results are mixed. The reason is that our method repositions vertices using our triangle shaping metric. Hence, in relatively flat areas vertices may move along those planes without significantly affecting the visual appearance of the shape. However, the MSAE numbers indicate that our method significantly outperforms most methods and even improves upon the error numbers of He and Schaefer (2013).

Table 2 shows the computation time of our method for each step and the whole pipeline, as well as the selected state of the art methods. Vertex pre-filtering generally occupies a significant portion of the total computation time since it involves multiple iterations of solving a global, linear system of equations. However, overall, our denoising method is generally faster than Zheng et al. (2011) (global), He and Schaefer (2013) and Wei et al. (2015) but slower than Zheng et al. (2011) (local).

8. Conclusions and Limitations

In this paper, we introduced a robust and effective approach to denoise 3D models with various levels of noise, with or without irregular surface sampling, while preserving geometric features. By utilizing both a pre-filtering step followed by an L_1-median normal filter, our approach outperforms recent anisotropic mesh denoising methods.

Despite the robustness exhibited by our work, our approach does have some limitations. Very large amounts of noise can cause errors in the result (see Figure 18). Similar to other methods (He and Schaefer, 2013; Wei et al., 2015),

Table 1: Quantitative comparisons among different methods. E_v and MSAE are scaled by 10^{-3} and 10^{-2} respectively.

Models	Methods	E_v	MSAE	Parameters
Fandisk (Figure 10) $\|V\|$: 6475 $\|F\|$: 12946	Our	8.974	**0.337**	(0.1, 30, 2, 30, 20, 10)
	Zheng et al. (2011) (local)	8.809	10.41	(3, 0.3, 10)
	Zheng et al. (2011) (global)	**8.670**	7.797	(0.01, 0.3, 10)
	He and Schaefer (2013)	9.373	0.655	Default
	Wei et al. (2015)	8.694	8.891	(0.3, 3, 0.3, 6)
Double-torus (Figure 14) $\|V\|$: 8702 $\|F\|$: 17408	Our	**17.19**	**1.828**	(0.2, 30, 12, 30, 50, 80)
	Zheng et al. (2011) (local)	21.15	183.6	(50, 0.35, 80)
	Zheng et al. (2011) (global)	21.37	161.6	(0.01, 0.35, 80)
	He and Schaefer (2013)	17.63	3.384	Default
	Wei et al. (2015)	28.28	207.4	(0.35, 45, 0.35, 50)
Nicolo (Figure 16) $\|V\|$: 31433 $\|F\|$: 62254	Our	4.257	**1.842**	(0.15, 30, 0, 50, 10, 8)
	Zheng et al. (2011) (local)	3.881	29.56	(3, 0.4, 8)
	Zheng et al. (2011) (global)	**3.870**	28.61	(0.07, 0.4, 8)
	He and Schaefer (2013)	4.409	3.521	Default
	Wei et al. (2015)	3.871	27.82	(0.4, 3, 0.4, 5)
Cube (Figure 15) $\|V\|$: 4808 $\|F\|$: 9612	Our	**46.16**	**0.024**	(0.15, 20, 1, 40, 100, 100)
	Zheng et al. (2011) (local)	47.75	150.7	(50, 0.3, 100)
	Zheng et al. (2011) (global)	48.25	143.2	(0.001, 0.25, 100)
	He and Schaefer (2013)	48.55	0.374	Default
	Wei et al. (2015)	47.31	150.4	(0.3, 30, 0.3, 30)

Table 2: Timing statistics of different approaches. Steps 1, 2 and 3 indicate vertex pre-filtering, L_1-median filtering and vertex update, respectively. The running time (in seconds) was recorded on an Intel Core i7-3770 3.40-GHz CPU.

Models	Fandisk (Figure 10)	Vase (Figure 11)	Double-torus (Figure 14)	Nicolo (Figure 16)	Cube (Figure 15)
Step 1	0.399	0.412	1.752	0.584	0.415
Step 2	0.342	0.263	1.137	0.931	1.249
Step 3	0.018	0.023	0.123	0.096	0.084
Total	0.759	0.698	3.012	1.611	1.748
Zheng et al. (2011) (local)	0.146	0.246	2.200	0.975	1.245
Zheng et al. (2011) (global)	1.183	0.682	2.621	9.017	0.776
He and Schaefer (2013)	3.235	2.089	4.871	18.668	2.484
Wei et al. (2015)	1.118	0.985	2.696	5.1	1.139

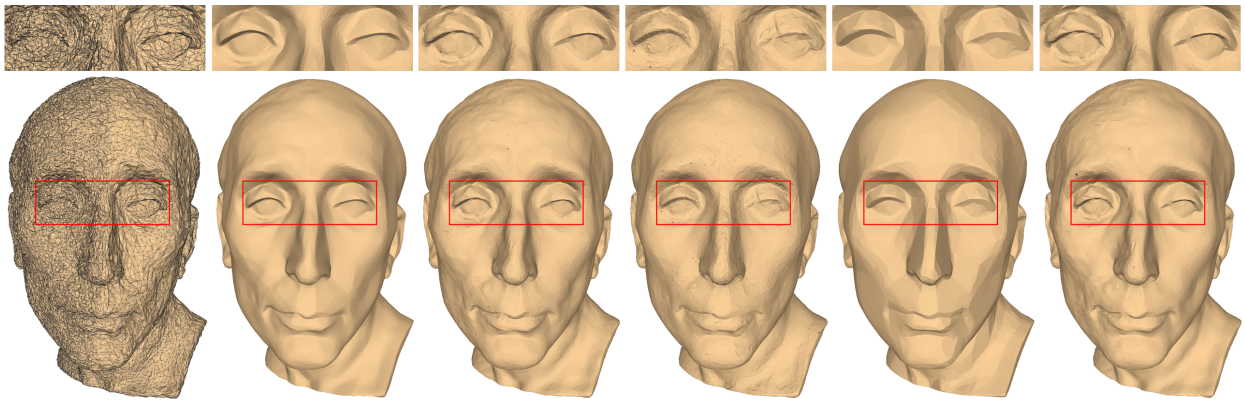


Figure 16: Denoised results of an irregularly sampled Nicolo model ($\sigma = 0.2\ell$). The input noisy model (a) is generated by resampling to decrease the density of the right side of the original Nicolo model and then adding Gaussian noise.

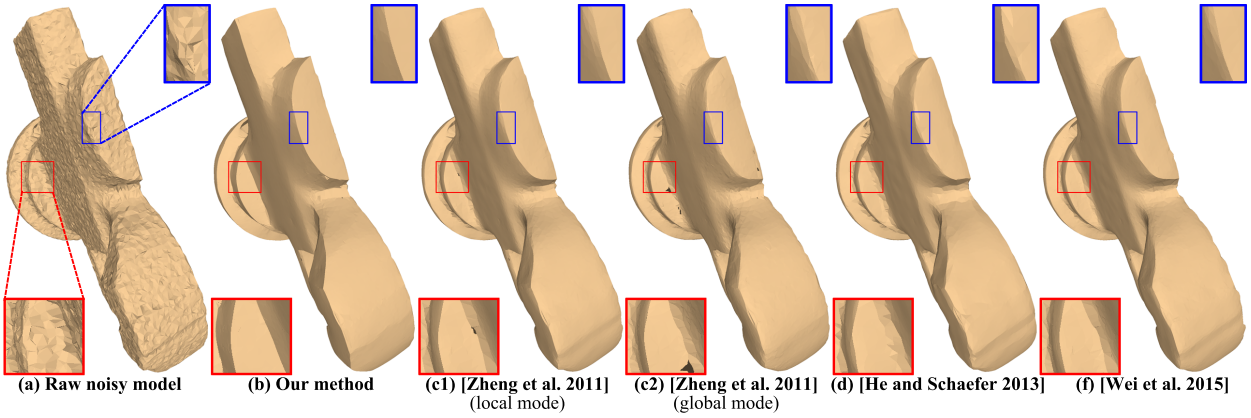


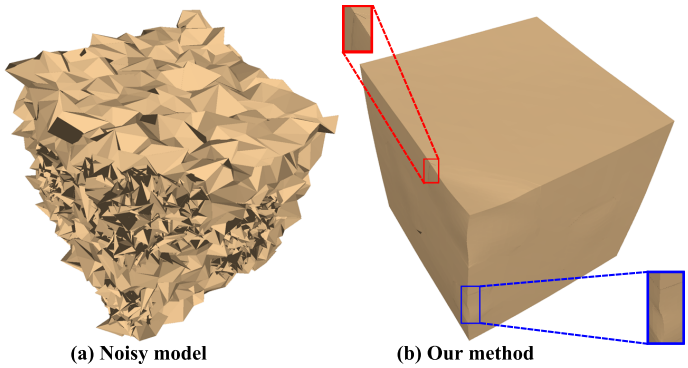
Figure 17: Denoised results of a scanned Iron model.

our method can fail when the sampling is extremely irregular. Like many other methods, our method also requires a few seconds to minutes to find a decent set of parameters that depend on the input model. While we have provided parameter ranges and examples for the models we used, automatically finding an optimal set of parameters for a given model is still a challenge.

Acknowledgments

This work was supported by NSF Career award IIS 1148976.

Bajaj, C. L., Xu, G., Jan. 2003. Anisotropic diffusion of surfaces and functions on surfaces. ACM Trans. Graph. 22 (1), 4–32.
Bian, Z., Tong, R., 2011. Feature-preserving mesh denoising based on vertices classification. Computer Aided Geometric Design 28 (1), 50 – 64.
Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B., 2010. Polygon mesh processing. CRC press.
Chuang, M., Kazhdan, M., Jul. 2011. Interactive and anisotropic geometry processing using the screened poisson equation. ACM Trans. Graph. 30 (4), 57:1–57:10.
Clarenz, U., Diewald, U., Rumpf, M., 2000. Anisotropic geometric diffusion in surface processing. In: Proc. of IEEE Conference on Visualization '00. pp. 397–405.
Desbrun, M., Meyer, M., Schröder, P., Barr, A. H., 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. of SIGGRAPH'99. pp. 317–324.
Desbrun, M., Meyer, M., Schröder, P., Barr, A. H., 2000. Anisotropic feature-preserving denoising of height fields and bivariate data. In: Graphics Interface'00. pp. 145–152.
El Ouafdi, A., Ziou, D., Krim, H., 2008. A smart stochastic approach for manifolds smoothing. Computer Graphics Forum 27 (5), 1357–1364.
Fan, H., Yu, Y., Peng, Q., 2010. Robust feature-preserving mesh denoising based on consistent subneighborhoods. IEEE Transactions on Visualization and Computer Graphics 16 (2), 312–324.
Fleishman, S., Drori, I., Cohen-Or, D., Jul. 2003. Bilateral mesh denoising. ACM Trans. Graph. 22 (3), 950–953.
He, L., Schaefer, S., Jul. 2013. Mesh denoising via l0 minimization. ACM Trans. Graph. 32 (4), 64:1–64:8.



(a) Noisy model (b) Our method

Figure 18: A failure example of our approach when handling an input mesh with a very high level of noise ($\sigma = 0.9\ell$). As shown in this figure, our method could bring defects to the result (refer to the red and blue rectangles).

Hildebrandt, K., Polthier, K., 2004. Anisotropic filtering of non-linear surface features. Computer Graphics Forum 23 (3), 391–400.

Jones, T. R., Durand, F., Desbrun, M., Jul. 2003. Non-iterative, feature-preserving mesh smoothing. ACM Trans. Graph. 22 (3), 943–949.

Kim, B., Rossignac, J., 2005. Geofilter: Geometric selection of mesh filter parameters. Comput. Graph. Forum, 295–302.

Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H., Jul. 2007. Parameterization-free projection for geometry reconstruction. ACM Trans. Graph. 26 (3).

Liu, X., Bao, H., Shum, H.-Y., Peng, Q., May 2002. A novel volume constrained smoothing method for meshes. Graph. Models 64 (3-4), 169–182.

Lu, X., Deng, Z., Chen, W., 2016. A robust scheme for feature-preserving mesh denoising. IEEE Trans. Vis. Comput. Graph. 22 (3), 1181–1194.

Nealen, A., Igarashi, T., Sorkine, O., Alexa, M., 2006. Laplacian mesh optimization. In: Proceedings of GRAPHITE'06. pp. 381–389.

Shamir, A., 2008. A survey on mesh segmentation techniques. Computer graphics forum 27 (6), 1539–1556.

Shen, Y., Barner, K., May 2004. Fuzzy vector median-based surface smoothing. IEEE Transactions on Visualization and Computer Graphics 10 (3), 252–265.

Solomon, J., Crane, K., Butscher, A., Wojtan, C., 2014. A general framework for bilateral and mean shift filtering. CoRR abs/1405.4734.

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

Sun, X., Rosin, P., Martin, R., Langbein, F., Sept 2007. Fast and effective feature-preserving mesh denoising. IEEE Transactions on Visualization and Computer Graphics 13 (5), 925–938.

Sun, X., Rosin, P. L., Martin, R. R., Langbein, F. C., Oct. 2008. Random walks for feature-preserving mesh denoising. Comput. Aided Geom. Des. 25 (7), 437–456.

Tasdizen, T., Whitaker, R., Burchard, P., Osher, S., 2002. Geometric surface smoothing via anisotropic diffusion of normals. In: Proceedings of IEEE Conference on Visualization '02. pp. 125–132.

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

Vollmer, J., Mencl, R., Müller, H., 1999. Improved Laplacian smoothing of noisy surface meshes. Computer Graphics Forum 18 (3), 131–138.

Wang, J., Zhang, X., Yu, Z., 2012. A cascaded approach for feature-preserving surface mesh denoising. Computer-Aided Design 44 (7), 597 – 610.

Wang, R., Yang, Z., Liu, L., Deng, J., Chen, F., Apr. 2014. Decoupling noise and features via weighted L1-analysis compressed sensing. ACM Trans. Graph. 33 (2), 18:1–18:12.

Wei, M., Yu, J., Pang, W., Wang, J., Qin, J., Liu, L., Heng, P., Jan 2015. Bi-normal filtering for mesh denoising. IEEE Transactions on Visualization and Computer Graphics 21 (1), 43–55.

Weijer, J. V. D., Boomgaard, R. V. D., 2001. Local Mode Filtering. In: Computer Vision and Pattern Recognition. Vol. 2. pp. 428–433.

Yagou, H., Ohtake, Y., Belyaev, A., 2002. Mesh smoothing via mean and median filtering applied to face normals. In: Proc. of Geometric Modeling and Processing 2002. pp. 124–131.

Zhang, H., Wu, C., Zhang, J., Deng, J., July 2015a. Variational mesh denoising using total variation and piecewise constant function space. IEEE Transactions on Visualization and Computer Graphics 21 (7), 873–886.

Zhang, W., Deng, B., Zhang, J., Bouaziz, S., Liu, L., 2015b. Guided mesh normal filtering. Computer Graphics Forum 34 (7), 23–34.

Zheng, Y., Fu, H., Au, O.-C., Tai, C.-L., Oct 2011. Bilateral normal filtering for mesh denoising. IEEE Transactions on Visualization and Computer Graphics 17 (10), 1521–1530.

Appendix

Taking the derivative of Equation (6) with respect to n_i and equating it to 0, we can obtain

$$\frac{\partial}{\partial n_i} = \sum_{j \in N(i)} a_j \phi\left(\frac{1-\cos(\gamma_{ij})}{1-\cos(\sigma_\gamma)}\right) \phi\left(\frac{\left\|c_i - c_j\right\|_2}{\sigma_c}\right) \frac{n_i - n_j}{\left\|n_i - n_j\right\|_2} = 0$$

$$\Leftrightarrow$$

$$n_i = \frac{\sum_{j \in N(i)} \frac{a_j \phi\left(\frac{1-\cos(\gamma_{ij})}{1-\cos(\sigma_\gamma)}\right) \phi\left(\frac{\left\|c_i - c_j\right\|_2}{\sigma_c}\right)}{\left\|n_i - n_j\right\|_2} n_j}{\sum_{j \in N(i)} \frac{a_j \phi\left(\frac{1-\cos(\gamma_{ij})}{1-\cos(\sigma_\gamma)}\right) \phi\left(\frac{\left\|c_i - c_j\right\|_2}{\sigma_c}\right)}{\left\|n_i - n_j\right\|_2}}$$

We denote $\omega_{ij} = \frac{a_j \phi\left(\frac{1-\cos(\gamma_{ij})}{1-\cos(\sigma_\gamma)}\right)\phi\left(\frac{\left\|c_i - c_j\right\|_2}{\sigma_c}\right)}{\left\|n_i - n_j\right\|_2}$, and n_i can be finally expressed by the following iterative form (same as Eq. (7)).

$$n_i^{t+1} = \frac{\sum_{j \in N(i)} \omega_{ij} n_j^t}{\sum_{j \in N(i)} \omega_{ij}}$$