Received: 9 March 2018 | Accepted: 21 March 2018

DOI: 10.1002/cav.1827

SPECIAL ISSUE PAPER

WILEY

# Mesh denoising via total variation and weighted Laplacian regularizations

Saishang Zhong[1,2] | Zhong Xie[1,2] | Weina Wang[3] | Zheng Liu[1,2] | Ligang Liu[3]

[1]Faculty of Information Engineering, China University of Geosciences, Wuhan, China

[2]National Engineering Research Center of Geographic Information System, China University of Geosciences, Wuhan, China

[3]School of Mathematical Sciences, University of Science and Technology of China, Hefei, China

**Correspondence**
Zheng Liu, Faculty of Information Engineering and National Engineering Research Center of Geographic Information System, China University of Geosciences, Wuhan, 430074, China.
Email: liu.zheng.jojo@gmail.com

**Funding information**
National Natural Science Foundation of China, Grant/Award Number: 61702467 and 41671400

**Abstract**

Mesh denoising is a fundamental problem in geometry processing. The main challenge is to preserve sharp features (such as edges and corners) and smooth regions (such as smoothly curved regions and fine details) while removing the noise. State-of-the-art denoising methods still struggle with this issue. In this paper, we first propose a new variational model combining total variation and anisotropic Laplacian regularization to filter the normal vector field of the mesh. This model can preserve sharp features and simultaneously handle smooth regions well. Then, a new vertex updating scheme is presented to reconstruct the mesh according to the filtered face normals. It prevents the orientation ambiguity problem introduced by existing schemes. Experiments show that our denoising method outperforms all compared methods visually and quantitatively, especially for meshes consisting of both sharp features and smooth regions.

**KEYWORDS**

feature preserving, Laplacian, mesh denoising, total variation

## 1 | INTRODUCTION

Triangulated meshes are wildly used to represent shapes in a variety of fields, such as computer graphics, virtual reality, and computer vision. Recently, meshes are usually derived by scanner devices from the real world. However, even with high-fidelity scanners, the scanning process inevitably generates noise, which degrades the quality of meshes and causes errors in downstream graphics applications.[1] Thus, mesh denoising becomes an important task in graphics processing. The main challenge in this task is to remove noise while preserving both sharp features and smooth regions well.

State-of-the-art mesh denoising methods, such as that in the method proposed by Sun et al.,[2] bilateral weighting filtering,[3] TV,[4] and $\ell_0$[5] minimization methods, achieve greatly successes. However, these methods are either not effective to deal with sharp features or less robust to handle smooth regions. Specifically, the recent TV[4] and $\ell_0$[5] minimization methods preserve sharp features well but inevitably suffer staircase effects in smooth regions. In other words, if surfaces have smooth regions, the two methods tend to flatten smoothly curved regions and sharpen fine details. The staircase effect[6] is caused by their sparsity regularization. In contrast, the method proposed by Sun et al.[2] and the bilateral weighting method[3] can efficiently handle smooth regions. However, they usually blur sharp features, especially when the noise level is high. Therefore, typical existing methods can deal with either sharp features or smooth regions.

In this paper, we propose a variational model combining TV and anisotropic Laplacian regularization term to filter the normal vector field. This model has crucial advantage in preserving sharp features and smooth regions and in substantially suppressing staircase effects. It is numerically solved by an iterative algorithm with the operator splitting and

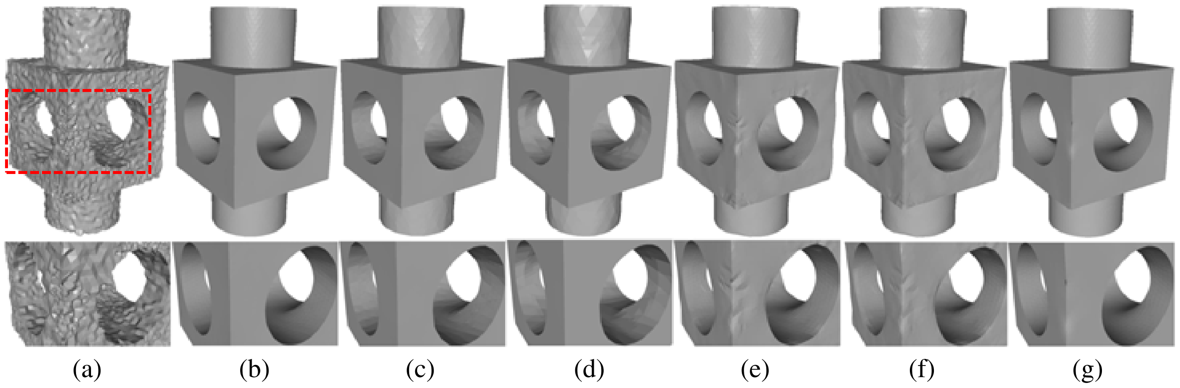

**FIGURE 1** Denoising results of Block (corrupted by Gaussian noise with standard deviation $\sigma = 0.2$, the mean edge length of the clean mesh). From left to right: (a) noisy mesh; denoising results produced by (b) our proposed iterative combining normal filtering (ICNF), (c) TV method,[4] (d) $\ell_0$ minimization,[5] (e) local bilateral filtering,[3] (f) global bilateral filtering,[3] and (g) the method proposed by Sun et al.[2] The second row is the zoomed-in view of Block

augmented Lagrangian method (ALM). After restoring the face normals, we introduce a new vertex updating scheme to perfectly reconstruct the vertex positions of the mesh, which solves the orientation ambiguity problem. To summarize, the contributions of the paper are listed as follows:

- We present a novel normal filtering model combining TV and anisotropic Laplacian term, which can simultaneously preserve sharp features and smooth regions.
- We propose a new vertex updating scheme to reconstruct the mesh. Compared with existing vertex updating schemes, it significantly improves the mesh quality and reduces foldovers.

Experiments illustrate that our denoising method outperforms the compared state-of-the-art methods[2,4] visually and quantitatively (see Section 6), especially for the mesh consisting of sharp features and smooth regions as showed in Figure 1. Last but not the least, when the noise level is high, our new vertex updating scheme can still perfectly reconstruct the mesh, whereas the scheme in the work of Sun et al.[2] may produce frequent foldovers; see Figure 3.

## 2 | RELATED WORK

Mesh denoising has been studied for years. Although a wide variety of denoising methods have been proposed, we only review those methods that are most relevant to this work.

Laplacian-based denoising methods can be considered as filtering schemes, which are roughly classified into isotropic and anisotropic methods. The isotropic methods[7,8] are classic and simple, which smooth meshes without considering geometric features. Thus, these methods suffer surface shrinkage and blur geometric features. Later on, to overcome this problem, many anisotropic methods have been proposed.[9–15] These anisotropic methods are more effective to deal with geometric features. However, when the noise level increases, they are difficult to produce satisfactory results, especially for meshes containing sharp features.

Recently, many feature-preserving denoising methods were proposed.[3–5,16–20] For preserving sharp features, many researchers use the conception of sparsity to remove noise from meshes. He et al.[5] proposed $\ell_0$ minimization approach to remove noise by inducing the sparsity for an edge-based operator. Zhang et al.[4] applied TV regularization on face normal field to remove noise. Lu et al.[20] designed $\ell_1$-median normal filter, which can effectively handle meshes with different levels of noise and irregular surface sampling. Although these sparsity minimization methods can effectively remove noise and preserve sharp features, they suffer undesired staircase effects in smoothly curved region. In particular, this phenomenon is more serious for $\ell_0$ minimization,[5] due to its high sparsity requirement.

## 3 | NOTATIONS

A mesh of arbitrary topology with no degenerate triangles in $\mathbb{R}^3$ is represented as $M$. The set of vertices, edges, and triangle faces of $M$ are denoted as $\{v_i : i = 1, 2, \dots, \mathrm{V}\}$, $\{e_i : i = 1, 2, \dots, \mathrm{E}\}$ and $\{\tau_i : i = 1, 2, \dots, \mathrm{T}\}$, respectively. Here V, E, and T are the numbers of vertices, edges, and faces of $M$, respectively. If $v$ is an endpoint of an edge $e$, then we denote it as $v \prec e$. Similarly, $e$ is an edge of a face $\tau$ denoted as $e \prec \tau$, and $v$ is a vertex of a face $\tau$ denoted as $v \prec \tau$.

Furthermore, let $D_1(\tau_i)$ be the 1-ring of triangle $\tau_i$, which is the set of triangles sharing some common edges with $\tau_i$. Denote the 1-disk of vertex $v_i$ as $M_1(v_i)$, which is the set of triangles containing $v_i$.

## 4 | NORMAL FILTERING USING TOTAL VARIATION AND REWEIGHTED LAPLACIAN REGULARIZATIONS

The TV method[4] preserves sharp features but tends to corrupt smooth regions. In contrast, the Laplacian-based method[3] can effectively deal with smooth regions. However, when the noise level increases, it usually fails to preserve sharp features. In this section, a normal filtering model using TV and Laplacian regularizations is proposed. It utilizes the best properties of the two methods and overcomes the weakness of both. The model is iteratively solved for the reweighted Laplacian term, which can significantly improve the quality of recovered mesh.

### 4.1 | Combining normal filtering model

Given a noisy mesh $M^{in}$, we denote its face normals as $\mathrm{N}^{in}$. To filter $\mathrm{N}^{in}$, our normal filtering model treats face normals N as variable and finds them as a solution to the following problem:

$$\min_{\mathrm{N}\in \mathrm{C}_\mathrm{N}} E_f(\mathrm{N}) + \frac{\alpha}{2}E_{tv}(\mathrm{N}) + \frac{\beta}{2}E_{wlap}(\mathrm{N}), \tag{1}$$

where $\alpha, \beta$ are two positive parameters, and $\mathrm{C}_\mathrm{N} = \{\mathrm{N} \in \mathbb{R}^{3\times \mathrm{T}} : \|\mathrm{N}_\tau\|_2 = 1, \forall \tau\}$. The variational model (1) consists of the fidelity, TV, and reweighted Laplacian term.

**Fidelity term**:

$$E_f(\mathrm{N}) = \sum_\tau s_\tau \left\| \mathrm{N}_\tau - \mathrm{N}_\tau^{in} \right\|^2,$$

where $s_\tau$ is the area, and $N_\tau$ is the normal of triangle $\tau$.

**TV term**:

$$E_{tv}(\mathrm{N}) = \sum_e l_e \left\| (\nabla \mathrm{N})_e \right\|,$$

where $l_e$ is the length of edge $e$, and $\nabla$ is the discrete gradient operator of face normal field. This operator is defined on each edge of the mesh. For its computation, refer to the work of Zhang et al.[4]

$E_{tv}$ is a TV regularization term applied to the face normal field, which enables sharp features preserving while removing noise. However, this TV term tends to optimize the face normal field to be a piecewise constant vector field. Thus, it introduces staircase effects in smooth regions. Moreover, some fine details of the mesh would be sharpened by this term.

**Reweighted Laplacian term**:

$$E_{wlap}(\mathrm{N}) = \sum_\tau s_\tau \left\| \mathrm{N}_\tau - \xi_\tau \sum_{\tau_j \in D_1(\tau)} w_{\tau\tau_j} \mathrm{N}_{\tau_j} \right\|^2,$$

where $w_{\tau\tau_j}$ is a reweighted term, and $\xi_\tau = \frac{1}{\sum_{\tau_j \in D_1(\tau)} w_{\tau\tau_j}}$ is a normalization factor. To handle smooth regions and avoid blurring sharp features in this Laplacian term, we define the weight $w_{\tau\tau_j}$ as a function about $\nabla$ of the face normal field given by

$$w_{\tau\tau_j} = s_{\tau_j} e^{-(\|(\nabla \mathrm{N})_e\|/\rho)^2}, \tag{2}$$

where $e$ is the edge shared by triangles $\tau$ and $\tau_j$, $s_{\tau_j}$ is the weight considering surface sampling rate, and $\rho$ is a user-specified threshold.

As we can see, both our weighting scheme (2) and the bilateral weighting scheme proposed by Zheng et al.[3] use the Gaussian function to measure the face normal difference of each local region. However, in order to handle considerable amount of noise, our weighting scheme dynamically updates the Laplacian weights, whereas the bilateral scheme[3] does not (the reason will be explained in Section 4.2).

## 4.2 | An iterative algorithm to solve the proposed normal filtering model

Due to the nondifferentiability of the TV term and the dynamic changing weights of the Laplacian term, it is challenging to solve the model (1). In this subsection, we propose an iterative algorithm with ALM to solve it. Recently, variable splitting and ALM are shown to be very successful to solve $\ell_1$-related minimization problems.[21,22] Thus, we introduce an auxiliary variable and use ALM to deal with the TV term. The Laplacian term can be handled by one iterative algorithm, in which the weights are dynamically updated from the current solution. In each iteration, it consists of two steps: solving the minimization problem (1) using ALM, and updating Laplacian weights by (2). Because the second step is obvious, we only introduce the implementation details of the first step.

In order to solve problem (1), we first introduce a new variable $\mathrm{p} \in \mathbb{R}^{3\times \mathrm{E}}$ and reformulate the problem as follows:

$$\begin{aligned}\min_{\mathrm{N},\mathrm{p}} \; & E_f(\mathrm{N}) + \frac{\alpha}{2}R_{tv}(\mathrm{p}) + \frac{\beta}{2}E_{wlap}(\mathrm{N}) + \psi(\mathrm{N}) \\ & \text{s.t., } \mathrm{p} = \nabla \mathrm{N},\end{aligned}$$

where

$$\psi(\mathrm{N}) = \begin{cases} 0, & \mathrm{N} \in C_{\mathrm{N}} \\ +\infty, & \mathrm{N} \notin C_{\mathrm{N}}. \end{cases}$$

For the above constrained optimization problem, we define the following augmented Lagrangian function:

$$\mathcal{L}(\mathrm{N},\mathrm{p};\lambda) = E_f(\mathrm{N}) + \frac{\alpha}{2}R_{tv}(\mathrm{p}) + \frac{\beta}{2}E_{wlap}(\mathrm{N}) + \psi(\mathrm{N}) + \sum_e l_e \lambda_e \cdot (\mathrm{p}_e - (\nabla\mathrm{N})_e) + \frac{r}{2}\sum_e l_e \|\mathrm{p}_e - (\nabla\mathrm{N})_e\|^2,$$

where $\lambda \in \mathbb{R}^{3\times \mathrm{E}}$ is a Lagrange multiplier, and $r$ is a penalty coefficient. This primal variables update procedure can be separated into two subproblems: the N subproblem and p subproblem.

For N subproblem, we fix the variable p and solve the following minimization problem:

$$\min_{\mathrm{N}} \; E_f(\mathrm{N}) + \frac{\beta}{2}E_{wlap}(\mathrm{N}) + \psi(\mathrm{N}) + \frac{r}{2}\sum_e l_e \left\| (\nabla\mathrm{N})_e - \left(\mathrm{p}_e + \frac{\lambda_e}{r}\right)\right\|^2. \tag{3}$$

This problem is a quadratic minimization if we ignore $\psi(N)$. Here, taking into account the computation efficiency, we first solve the following quadratic programming:

$$\min_{\mathrm{N}} \; E_f(\mathrm{N}) + \frac{\beta}{2}E_{wlap}(\mathrm{N}) + \frac{r}{2}\sum_e l_e \left\| (\nabla\mathrm{N})_e - \left(\mathrm{p}_e + \frac{\lambda_e}{r}\right)\right\|^2 \tag{4}$$

and then project the minimizer to an unit sphere. The quadratic problem (4) has the following first-order optimality condition:

$$A + \beta L^T S L = b, \tag{5}$$

where $A, L, S \in \mathbb{R}^{\mathrm{T}\times\mathrm{T}}$, $b \in \mathbb{R}^{\mathrm{T}\times 1}$, and entries of them are

$$\begin{aligned} A_{ij} &= \begin{cases} 2s_{\tau_i} + r\sum\limits_{e\prec\tau_i} l_e, & i = j \\ -r l_e, & i \neq j; \end{cases} \\ L_{ij} &= \begin{cases} 1, & i = j \\ -\xi_{\tau_i} w_{\tau_i\tau_j}, & i \neq j; \end{cases} \quad S_{ij} = \begin{cases} s_{\tau_i}, & i = j \\ 0, & i \neq j; \end{cases} \\ b_i &= 2s_{\tau_i}\mathrm{N}^{in} + r\sum_{e\prec\tau_i} l_e\left(\mathrm{p}_e + \frac{\lambda_e}{r}\right), \end{aligned}$$

where $\tau_j \in D_1(\tau_i)$ and $e = \tau_i \cap \tau_j$. Thus, (5) can be efficiently solved using various well-developed numerical libraries such as Eigen, Taucs, and Math Kernel Library (MKL)

Next, we solve the p subproblem by fixing the variable N and reformulate it to

$$\min_{\mathrm{p}} \; R_{tv}(\mathrm{p}) + \frac{r}{\alpha}\sum_e l_e \left\| \mathrm{p}_e - (\nabla\mathrm{N})_e + \frac{\lambda_e}{r}\right\|^2. \tag{6}$$

Because p can be decoupled and solved individually, the problem (6) is solved edge by edge. Thus, for each $p_e$, we need to solve

$$\min_{p_e} \|p_e\| + \frac{r}{\alpha}\left\|p_e - (\nabla N)_e + \frac{\lambda_e}{r}\right\|^2,$$

which has a closed-form solution

$$p_e = \max\left(0, 1 - \frac{\alpha}{2r\|\Psi\|}\right)\Psi, \tag{7}$$

where $\Psi = (\nabla N)_e - \frac{\lambda_e}{r}$.

In summary, the procedure of solving the model (1) is outlined in Algorithm 1, where $E_d$ is the difference of normal field between two iterations defined as

$$E_d = \sum_{\tau} s_\tau \left\|N_\tau^k - N_\tau^{k-1}\right\|^2.$$

As can be seen, we use an iteration strategy to solve the model (1). We solve the model with fixed weights exactly in each iteration and then update the weights according to the current solution for next iteration.

We should point out that if we keep weights of the Laplacian term unchanged, the minimization problem(1) needs to be solved only one time. This can be considered as an inexact version of Algorithm 1, which is used in the work of Zheng et al.[3] If the noise level is low, this strategy works well. However, when the noise level increases, our experiments demonstrate that this inexact strategy is hard to obtain satisfactory results, even with fine-tuning parameters. The reason may be as follows. Because both the noise and sharp features belong to high-frequency signal, the Laplacian weights(2) directly estimated from the noisy mesh cannot distinguish them clearly. Thus, some noise will leave in flat regions, and some sharp features will be blurred. In contrast, our method with iteratively updating the weights is able to effectively improve the denoised result. See the zoomed-in view of Figure 2.

**Algorithm 1:** Iterative Algorithm for Solving Normal Filtering Model (1)

```
Initialization: $N^{-1}=0, \epsilon=1e-5, c=-1,$
                $C=30, K=70;$
repeat
    Initialization: $N^{-1,c}=0, p^{-1}=0,$
                    $\lambda^0=0, k=-1;$
    repeat
        Solve N-subproblem
            Compute $N^{k,c}$ from (4);
            Normalize $N^{k,c}$;
        Solve p-subproblem
            Compute $p^k$ from (7);
        Update Lagrange multiplier
            $\lambda^{k+1}=\lambda^k+r(p^k-\nabla N^{k,c})$;
    until $E_d < \epsilon$ or $k \geq K$;
    Update Laplacian weights by (2);
    $N^c = N^{k,c}$;
until $E_d < \epsilon$ or $c \geq C$;
```

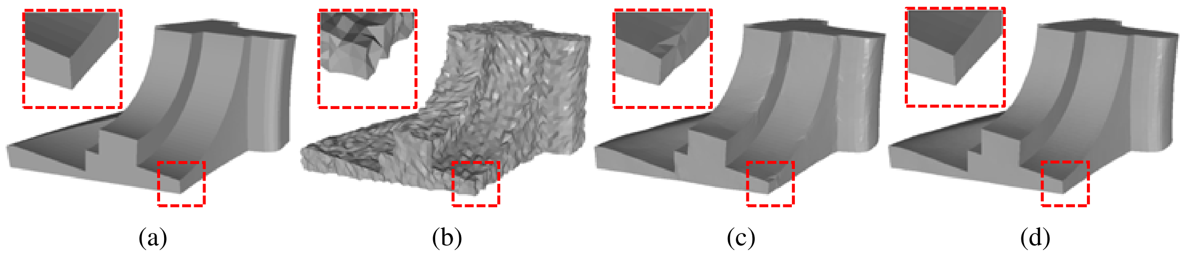

**FIGURE 2** Denoising results of Fandisk (corrupted by Gaussian noise, $\sigma = 0.2$). From left to right: (a) clean mesh, (b) noisy mesh, (c) result of solving (1) without iteration, and (d) result produced by Algorithm 1 with iteration

*Remark* 1. Algorithm 1 is insensitive to the initialization of parameters. In our experiments, the numbers of iterations $C$ and $K$ are in the range of $[10, 30]$ and $[30, 70]$, respectively. We empirically set the threshold $\epsilon = 1e - 5$.

## 5 | ROBUST VERTEX UPDATING

After filtering the face normals by Algorithm 1, mesh vertex positions should be updated to match the filtered face normals. One famous vertex updating model is to minimize the following quadratic energy:

$$E(\mathrm{V}) = \sum_{\tau} \sum_{(v_i, v_j) \in \tau} s_\tau (\mathrm{N}_\tau \cdot (v_i - v_j))^2, \tag{8}$$

where $N_\tau$ is the filtered normal of triangle $\tau$. This model has a beautiful implementation by Sun et al.[2] When the noise level is low, the method of Sun et al.[2] achieves good results. However, when the noise level increases, the recovered vertex positions deviate far from those of the clean mesh. In this situation, method of Sun et al.[2] suffers producing frequent foldovers. The reason is that the model (8) neglects the orientations of face normals. It only penalizes the nonorthogonality and cannot distinguish $-\mathrm{N}_\tau$ and $\mathrm{N}_\tau$.

To overcome this ambiguity problem, we propose a new vertex updating model defined as

$$E_v = \sum_{\tau = (v_i, v_j, v_k)} \left( \mathrm{N}_\tau - \frac{(v_j - v_i) \times (v_k - v_i)}{\|(v_j - v_i) \times (v_k - v_i)\|} \right)^2, \tag{9}$$

where $(v_i, v_j, v_k)$ are vertices of $\tau$ in anticlockwise order. This model can effectively solve the ambiguity problem, because it not only considers the orthogonality between the face and its corresponding normal direction but also takes the orientation of the face into account.

We can reformulate the partial derivatives of (9) with respect to $v_i$ as follows:

$$\frac{\partial E_v}{\partial v_i} = \sum_{\tau \prec M_1(v_i)} \left[ \mathrm{N}_\tau - (\mathrm{N}_\tau \cdot \mathcal{N}_\tau) \mathcal{N}_\tau \right] \times (v_j - v_k), \tag{10}$$

where $\mathcal{N}_\tau$ is the updating normal of $\tau$ according to updated $v$ (derivation process of formula in Equation 10 is given in the Appendix). With the gradient information calculated from (10) and the initial vertex positions, we adopt the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm[23] to resolve the model (9). In each iteration, BFGS uses the energy and gradient evaluated at current and previous iterations.

As we can see in Figure 3, the results produced by Sun et al.[2] suffer from severe foldovers (highlighted in red), whereas our method produces much better results without foldovers (both methods use filtered normals produced by Algorithm 1

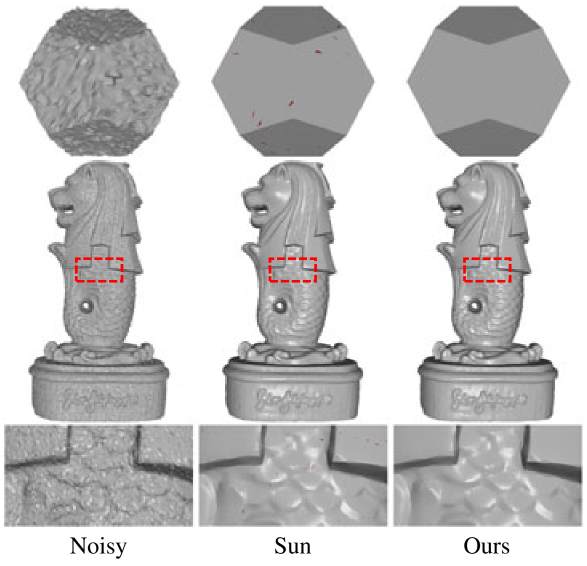

**FIGURE 3** Comparisons of vertex updating methods. The first column shows noisy meshes (corrupted by Gaussian noise, $\sigma = 0.3$). The second and third columns are results generated by Sun et al.[2] and ours (9). The foldovers are highlighted in red

**TABLE 1** The performance of two vertex updating methods

| | Foldovers / Time (seconds) | |
|---|---|---|
| **Meshes($\|V\|$, $\|F\|$)** | **Sun et al.[2]** | **Ours** |
| Twelve(4.6K, 9.2K) | 116 / 0.16 | 0 / 0.43 |
| Merlion(283K, 566K) | 8,236 / 3.79 | 0 / 14.22 |

as input). This shows that, compared with the method of Sun et al.,[2] our method is more robust to the noise level and can produce better mesh quality.

The performances of the two methods are listed in Table 1. As can be seen, the method of Sun et al.[2] is faster than ours, especially for large meshes.

## 6 | EXPERIMENTS AND COMPARISONS

We evaluate our mesh denoising method on both synthetic data and real scans. The synthetic data are obtained by adding Gaussian noise in vertex normal directions of clean meshes. All methods tested in this paper have been implemented by C++; all meshes are rendered in a flat-shading model to show faceting effect.

### 6.1 | Parameters tuning

Our model (1) has two parameters, $\alpha$ and $\beta$, used to balance the fidelity, TV, and Laplacian terms.

$\alpha$ affects sparsity of the model introduced by TV term. Figure 4 demonstrates results of different $\alpha$ with fixed $\beta$. As can be seen, if $\alpha$ is zero, sharp features of the result should be blurred, illustrated in Figure 4b, and if $\alpha$ is too large, smooth regions will be oversharpened, showed in Figure 4e. Moreover, there exists a range of $\alpha$ for our model (1) producing visually well results; see Figure 4c,d.

$\beta$ has influence on smoothness of the result. Figure 5 shows the results of different $\beta$ with fixed $\alpha$. As we can see, if $\beta$ is zero, the result suffers the staircase effect, showed in Figure 5b; and too large $\beta$ should oversmooth sharp features demonstrated in Figure 5e. Again, there exists a range of $\beta$ for our model producing satisfactory results, given in Figure 5c,d.

Due to the values of $\alpha$ and $\beta$ that are highly related to the mesh type and noise level, it is challenging to automatically determine them. Fortunately, there are several tuning strategies: First, the value of $\alpha$ should be bigger than $\beta$ when handling computer-aided design (CAD) meshes, and vice versa when dealing with non-CAD meshes; second, for almost all examples, $\alpha$ and $\beta$ are in the range of $[0, 1]$ and $[10, 1000]$, respectively.

### 6.2 | Experiments on synthetic data

We compare our denoising method denoted as ICNF to TV method,[4] $\ell_0$ minimization,[5] the bilateral normal filter,[3] and the method proposed by Sun et al.,[4] abbreviated as TV, $\ell_0$, ZhengL\ZhengG, and SunNF, respectively.

Figure 6 shows the results on CAD meshes consisting of both sharp features and smooth regions. As can be seen, ICNF, TV, and $\ell_0$ can preserve the sharp features. However, both TV and $\ell_0$ suffer from the undesired staircase effect in smoothly curved regions, and this phenomenon is serious for $\ell_0$; see the zoomed-in view in Figure 6c,d. On the contrary,

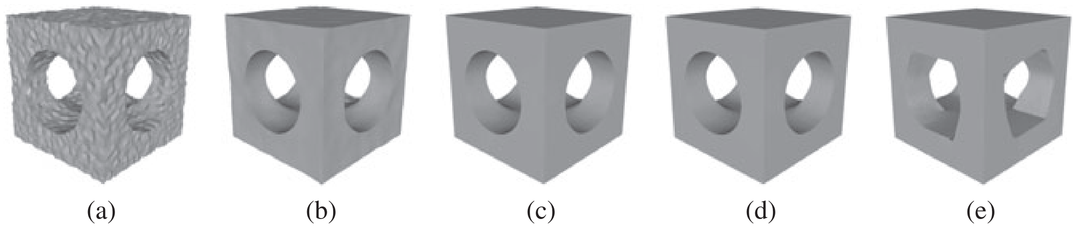

**FIGURE 4** Denoising results for $\alpha$ with fixed $\beta$. From left to right: input noisy mesh (corrupted by Gaussian noise, $\sigma = 0.15$) and results with different $\alpha$. (a) noisy, (b) $\alpha = 0$, (c) $\alpha = 0.04$, (d) $\alpha = 0.1$, and (e) $\alpha = 1.0$

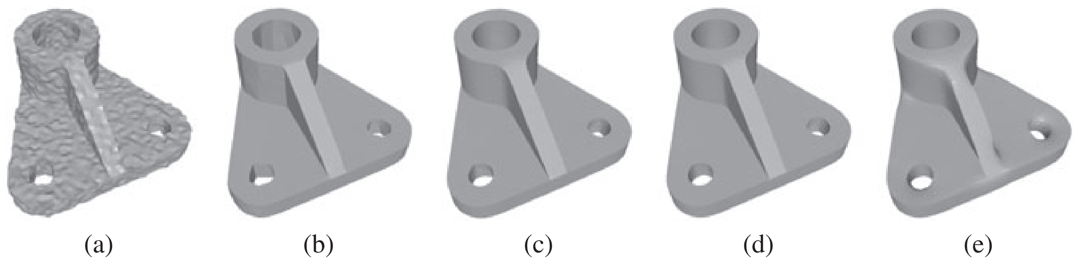

**FIGURE 5** Denoising results for $\beta$ with fixed $\alpha$. From left to right: input noisy mesh (corrupted by Gaussian noise, $\sigma = 0.15$) and results with different $\beta$. (a) noisy, (b) $\beta = 0$, (c) $\beta = 120$, (d) $\beta = 600$, and (e) $\beta = 1,400$

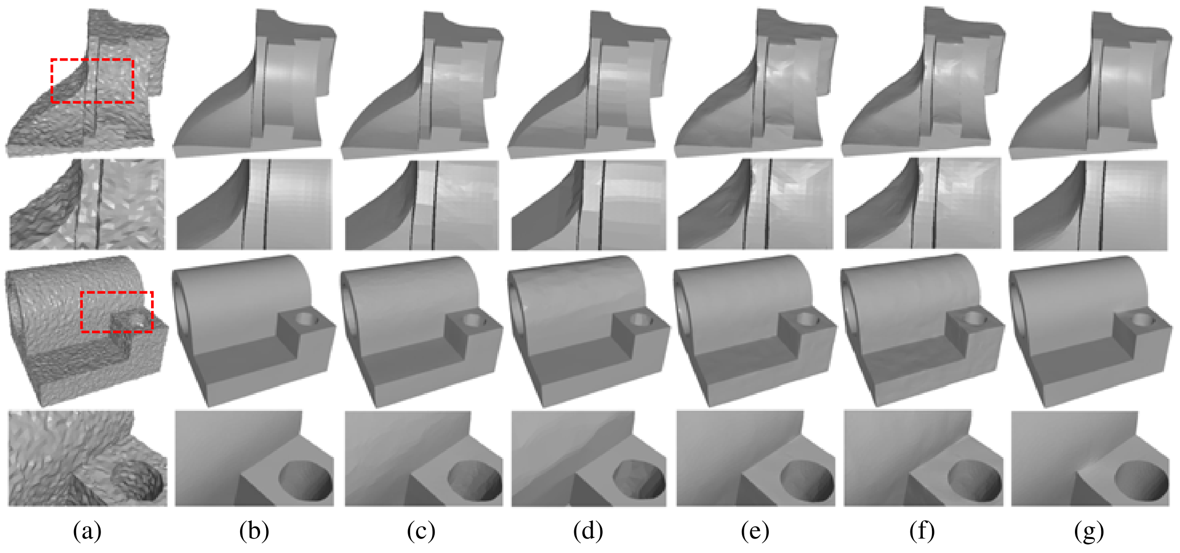

**FIGURE 6** Comparison of denoising methods for CAD meshes (corrupted by Gaussian noise, $\sigma = 0.15$). The second and fourth rows are the zoomed-in view. (a) noisy, (b) ICNF, (c) TV, (d) $\ell_0$, (e) ZhengL, (f) ZhengG, (g) SunNF

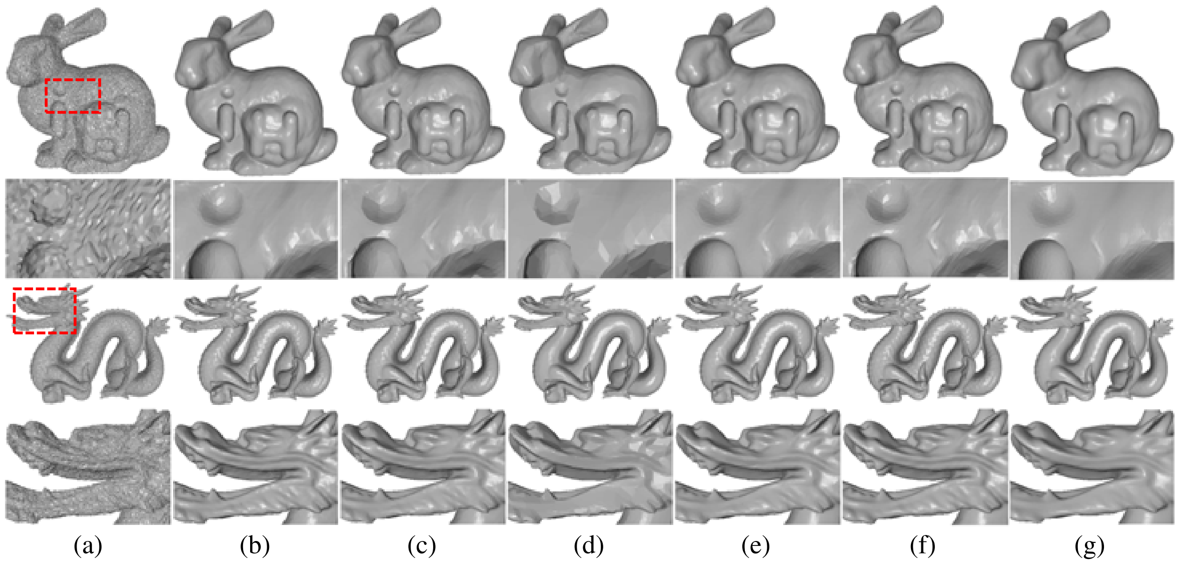

**FIGURE 7** Comparison of denoising methods on non-CAD meshes (corrupted by Gaussian noise, $\sigma = 0.2$). The second and fourth rows are the zoomed-in view. (a) noisy, (b) ICNF, (c) TV, (d) $\ell_0$, (e) ZhengL, (f) ZhengG, (g) SunNF

ZhengL\ZhengG and SunNF recover smooth regions well but blur some sharp features; see the results in Figure 6e–g. Thus, the compared four methods either preserve sharp features or recover smooth regions well, whereas ICNF can effectively deal with both. For CAD meshes, visual comparisons in Figure 6 show that ICNF is noticeably better than the other four methods.

Figure 7 demonstrates the results on non-CAD meshes. As we can see, TV and $\ell_0$ tend to oversharp fine details, especially $\ell_0$ method, whereas SunNF blurs some fine details; see the zoomed-in view in Figure 7c,d,g. In contrast, both ICNF and ZhengL\ZhengG produce visually satisfactory results. However, from the metric errors introduced in next the subsection, we find that the errors of ICNF are always lower than ZhengL\ZhengG. Thus, for non-CAD meshes, ICNF also yields better results than the other four methods.

## 6.3 | Quantitative comparisons

We employ the mean square angular error to measure the error produced by normal filtering method, and we use the $L_2$ vertex-based error ($E_{v,2}$) to measure the error generated by the vertex updating scheme. The evaluation results about these two metrics of the examples, shown in Figures 6 and 7, are listed in Table. 2. As we can see, mean square angular error by ICNF is obviously smaller than all the other methods. For CAD meshes, ICNF outperforms the other methods in $E_{v,2}$. For non-CAD meshes, although the results by ZhengG are better than others in $E_{v,2}$, in those cases, results by ICNF are close to the best ones.

The CPU costs of all the methods are also summarized in Table. 2. As can be seen, SunNF is the fastest one, whereas $\ell_0$ is the slowest. ICNF is faster than $\ell_0$, and slower than the others.

## 6.4 | Experiments on real scans

We testify the validity of ICNF on real scans; see Figure 8. One can see that ICNF also can yield satisfactory results for the real scans.

**TABLE 2** Quantitative comparisons for all the five denoising methods

| | Mean square angular error($\times 10^{-3}$), $E_{v,2}$($\times 10^{-3}$); Time (seconds) | | | | | |
|---|---|---|---|---|---|---|
| **Meshes($\|V\|$, $\|F\|$)** | **ICNF** | **TV** | **$\ell_0$** | **ZhengL** | **ZhengG** | **SunNF** |
| Block(8.7K, 17.6K) | **4.23**, **1.69**; 5.94 | 4.70, 1.93; 0.67 | 5.39, 2.09; 6.24 | 10.2, 2.81; 0.21 | 9.78, 2.84; 1.3 | 6.66, 2.07; 0.42 |
| Fandisk(6.5K, 12.9K) | **1.67**, **2.04**; 3.15 | 2.12, 2.13; 0.44 | 3.24, 2.22; 3.47 | 4.94, 2.24; 0.11 | 4.09, 2.06; 0.65 | 3.92, 2.66; 0.3 |
| Joint(20.9K, 41.8K) | **1.22**, **1.47**; 18.3 | 2.08, 1.53; 1.78 | 4.43, 2.02; 23.6 | 2.94, 1.97; 0.49 | 3.39, 2.64; 7.92 | 2.07, 2.46; 1.2 |
| Bunny(34.9K, 69.5K) | **16.3**, 1.54; 28.4 | 18.3, 1.6; 3.85 | 22.4, 2.49; 34.6 | 17.7, 1.53; 0.77 | 16.8, **1.37**; 8.62 | 24.2, 2.28; 1.96 |
| Dragon(161K, 323K) | **16.7**, **0.48**; 71.3 | 23.8, 0.6; 27.3 | 35, 0.76; 187.5 | 29.1, 0.71; 4.72 | 17, 0.58; 5.56 | 25, 1.14; 7.94 |

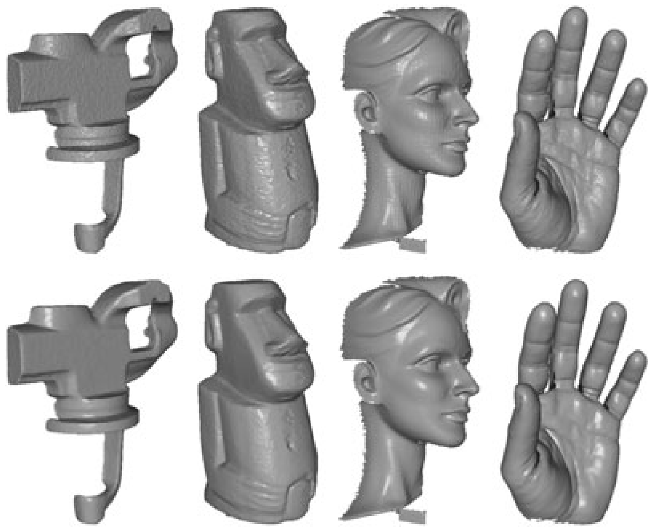

**FIGURE 8** Denoising results of real scans

## 7 | CONCLUSION

In this paper, we first present a variational model to restore the noisy normal vector field. It combines TV and anisotropic Laplacian regularization. Our model can preserve sharp features and also do a good job on smooth regions. A novel vertex updating scheme is also introduced to overcome the orientation ambiguity problem. We compare our denoising method with several state-of-the-art methods on different types of meshes visually and quantitatively. The experimental results demonstrate the robustness and efficiency of ours. We believe that our denoising method can be applied to more general meshes compared with the selected state-of-the-art denoising methods.

**Limitation and future work.** Our mesh denoising method is computationally expensive for large meshes. A future work is to accelerate our method and decrease CPU costs.

## ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China (Nos. 61702467,41671400).

## ORCID

*Saishang Zhong* http://orcid.org/0000-0003-4832-5296
*Zheng Liu* http://orcid.org/0000-0001-6713-6680

## REFERENCES

1. Wei M, Yu J, Pang W-M, et al. Bi-normal filtering for mesh denoising. IEEE Trans Vis Comput Graph. 2015;21(1):43–55.
2. Sun X, Rosin P, Martin R, Langbein F. Fast and effective feature-preserving mesh denoising. IEEE Trans Vis Comput Graph. 2007;13(5):925–938.
3. Zheng Y, Fu H, Au OK-C, Tai C-L. Bilateral normal filtering for mesh denoising. IEEE Trans Vis Comput Graph. 2011;17(10):1521–1530.
4. Zhang H, Wu C, Zhang J, Deng J. Variational mesh denoising using total variation and piecewise constant function space. IEEE Trans Vis Comput Graph. 2015;21(7):873–886.
5. He L, Schaefer S. Mesh denoising via $\ell_0$ minimization. ACM Trans Graph. 2013;32(4):1–8.
6. Chan T, Marquina A, Mulet P. High-order total variation-based image restoration. SIAM J Sci Comput. 2000;22(2):503–516.
7. Taubin G. A signal processing approach to fair surface design. Paper presented at: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques; 1995 Aug 6–11; Los Angeles, CA. New York, NY: ACM; 1995. p. 351–358.
8. Desbrun M, Meyer M, Schröder P, Barr AH. Implicit fairing of irregular meshes using diffusion and curvature flow. Paper presented at: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques; 1999 Aug 8–13; Los Angeles, CA. New York, NY: ACM Press/Addison-Wesley Publishing Co; 1999. p. 317–324.
9. Taubin G. Linear anisotropic mesh filters. Yorktown Heights, NY: IBM Thomas J. Watson Research Center; 2001. IBM research report. RC22213 (W0110-051).
10. Tasdizen T, Whitaker R, Burchard P, Osher S. Geometric surface smoothing via anisotropic diffusion of normals. Paper presented at: Proceedings of the Conference on Visualization; 2002 Oct 27–Nov 1; Boston, MA. Washington, DC: IEEE Computer Society; 2002. p. 125–132.
11. Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising. ACM Trans Graph. 2003;22(3):950–953.
12. Hildebrandt K, Polthier K. Anisotropic filtering of non-linear surface features. Comput Graph Forum. 2004;23(3):391–400.
13. Wang CCL. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. IEEE Trans Vis Comput Graph. 2006;12(4):629–639.
14. Desbrun M, Meyer M, Schroder P, Schroder AH, Usc C. Anisotropic feature-preserving denoising of height fields and bivariate data. Paper presented at: Proceedings of the Graphics Interface 2000 Conference; 2000 May 15–17; Montreal, Canada. p. 145–152.
15. Bajaj CL, Xu G. Anisotropic diffusion of surfaces and functions on surfaces. ACM Trans Graph. 2003;22(1):4–32.
16. Hildebrandt K, Polthier K. Constraint-based fairing of surface meshes. Paper presented at: Eurographics Symposium on Geometry Processing; 2007 Jul 4–6; Barcelona, Spain. Aire-la-Ville, Switzerland: Eurographics Association. p. 203–212.
17. Fan H, Yu Y, Peng Q. Robust feature-preserving mesh denoising based on consistent subneighborhoods. IEEE Trans Vis Comput Graph. 2010;16(2):312–324.
18. Wang R, Yang Z, Liu L, Deng J, Chen F. Decoupling noise and features via weighted $\ell_1$-analysis compressed sensing. ACM Trans Graph. 2014;33(2):1–12.
19. Zhang W, Deng B, Zhang J, Bouaziz S, Liu L. Guided mesh normal filtering. Comput Graph Forum. 2015;34(7):23–34.
20. Lu X, Chen W, Schaefer S. Robust mesh denoising via vertex pre-filtering and $\ell_1$-median normal filtering. Comput Aided Geom Des. 2017;54:49–60.

21. Osher S, Burger M, Goldfarb D, Xu J, Yin W. An iterative regularization method for total variation-based image restoration. SIAM J Multiscale Model Simul. 2005;4(2):460–489.
22. Wu C, Tai XC. Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models. SIAM J Imaging Sci. 2010;3(3):300–339.
23. Dennis JE, Moré JJ. Quasi-newton methods, motivation and theory. SIAM Rev. 1977;19(1):46–89.

**Saishang Zhong PhD candidate.** saishang@cug.edu.cn. His main research interests include computer graphics, image processing and so on.

**Zhong Xie PhD, Professor.** xiezhong@cug.edu.cn. His main research interests include spatial analysis modeling and application, deep learning and so on.

**Weina Wang PhD candidate.** wnwang@mail.ustc.edu.cn. Her main research interests include image and video processing.

**Zheng Liu PhD, Lecturer.** liu.zheng.jojo@gmail.com. His main research interests include digital geometry processing, computer graphics, image and video processing.

**Ligang Liu PhD, Professor.** lgliu@ustc.edu.cn. His main research interests include digital geometry processing, computer graphics, image processing and so on.

**How to cite this article:** Zhong S, Xie Z, Wang W, Liu Z, Liu L. Mesh denoising via total variation and weighted Laplacian regularizations. *Comput Anim Virtual Worlds*. 2018;29:e1827. https://doi.org/10.1002/cav.1827

## APPENDIX

Denoting $D_1 = (v_j - v_i) \times (v_k - v_i), D_2 = (v_k - v_j) \times (v_i - v_j), and\, D_3 = (v_i - v_k) \times (v_j - v_k)$, we can write $\mathcal{N}_\tau = \frac{D_1}{2s_\tau} = \frac{D_2}{2s_\tau} = \frac{D_3}{2s_\tau}$. By using the above facts, we take the derivative of Equation (9) with respect to $v_i$ and then obtain

$$\begin{aligned}
\frac{\partial E_v}{\partial v_i} &= 2 \sum_{\tau \prec M_1(v_i)} \left(\mathrm{N}_\tau - \mathcal{N}_\tau\right) \frac{\partial \mathcal{N}_\tau}{\partial v_i} \\
&= 2 \sum_{\tau \prec M_1(v_i)} \left(\mathrm{N}_\tau - \mathcal{N}_\tau\right) \left[\left(\frac{\partial \mathcal{N}_\tau}{\partial D_1}\frac{\partial D_1}{\partial v_i}\right) + \left(\frac{\partial \mathcal{N}_\tau}{\partial D_2}\frac{\partial D_2}{\partial v_i}\right)\right. \\
&\quad \left. + \left(\frac{\partial \mathcal{N}_\tau}{\partial D_3}\frac{\partial D_3}{\partial v_i}\right)\right] \\
&= 6 \sum_{\tau \prec M_1(v_i)} \frac{1}{2s_\tau}[\mathrm{N}_\tau - (\mathrm{N}_\tau \cdot \mathcal{N}_\tau)\mathcal{N}_\tau] \times (v_j - v_k).
\end{aligned}$$

Because the BFGS algorithm can ignore the weighting parameters involved in the above equation, for simplicity, we can further rewrite it as

$$\frac{\partial E_v}{\partial v_i} = \sum_{\tau \prec M_1(v_i)} [\mathrm{N}_\tau - (\mathrm{N}_\tau \cdot \mathcal{N}_\tau)\mathcal{N}_\tau] \times (v_j - v_k).$$