SIAM J. IMAGING SCIENCES
Vol. 15, No. 1, pp. 1–22



# A Novel Mesh Denoising Method Based on Relaxed Second-Order Total Generalized Variation*

Huayan Zhang†, Zhishuai He†, and Xiaochao Wang‡

**Abstract.** In the paper, we develop a relaxed second-order total generalized variation model on triangulated surfaces, which couples the first-order gradient operator and the weighted divergence operator. An iterative two-stage mesh denoising method is proposed with the relaxed model, which contains facet normal filtering based on the relaxed model and robust vertex updating. The nondifferentiable optimization problem is solved by an iterative algorithm based on variable splitting and augmented Lagrangian method. Our denoising method is discussed and compared to several state-of-the-art techniques in terms of reconstruction quality, quantitative comparison, and computational costs. Experiments indicate that our approach is comparable to state-of-the-art algorithms at reasonable costs. It can produce denoising results with more structures, alleviate the staircase effect (false edges), and prevent edge flips. The quantitative errors also verify that the newly proposed algorithm is robust numerically.

**Key words.** mesh denoising, differential operator, triangulated surface, relaxed second-order total generalized variation, augmented Lagrangian method

**AMS subject classifications.** 65K10, 68U05

**DOI.** 10.1137/21M1397945

## 1. Introduction.

Triangulated surfaces have been extensively used to represent 3D models in computer graphics and scientific computing due to their simplicity and the rapid development of 3D acquisition techniques. However, when generating triangulated meshes by digital scanning devices and some triangulation algorithms, they usually are polluted by noise due to inevitable measurement errors and algorithm errors. Therefore, it is a crucial fundamental problem to devise an effective algorithm to filter the noise of meshes and generate high-quality meshes before further applications. The main challenge of mesh denoising methods is to remove noise while preserving sharp features and details.

Recently, the famous total variation (TV) term using the gradient operator and $L_1$ norm has been shown very successful in edge-preserving image processing and mesh processing [28, 7, 6, 11, 48, 47]. Yet TV also has shortcomings, for instance, the notable staircase effect (the piecewise constant results). To overcome the drawbacks of TV, the following total

*Received by the editors February 10, 2021; accepted for publication (in revised form) August 25, 2021; published electronically January 4, 2022.
https://doi.org/10.1137/21M1397945
**Funding:** This work was supported by NSF of China grants 61802279 and 61602341 and NSF of Tianjin grants 18JCQNJC00100 and 17JCQNJC00600.

†School of Computer Science and Technology, Tiangong University, Tianjin, People's Republic of China (zhanghy307@163.com, hezhishuai20@163.com).
‡Corresponding author. School of Mathematical Sciences, Tiangong University, Tianjin, People's Republic of China (wangxiaochao18@163.com).

generalized variation (TGV) model (1.1) has been introduced [4]:

$$\text{(1.1)}\quad \mathrm{TGV}_{\alpha}^{k}(u)=\sup\left\{\int_{\Omega} u\mathrm{div}^{k}v\mathrm{d}x|v\in\mathcal{C}_{c}^{k}(\Omega,\mathrm{Sym}^{k}(\mathbb{R}^{d})),||\mathrm{div}^{l}v||_{\infty}\leq\alpha_{l},l=0,\ldots,k-1\right\},$$

where $\mathrm{Sym}^{k}(\mathbb{R}^{d})$ indicates the space of symmetric tensors of order $k$ with arguments in $\mathbb{R}^{d}$ and $\alpha_l$ are fixed positive parameters. Actually, when $k=1$, (1.1) equals TV. When $k=2$, (1.1) has the following form:

$$\text{(1.2)}\qquad \mathrm{TGV}_{\alpha}^{2}(u)=\min_{v}\alpha_{1}\int_{\Omega}|\nabla u-v|\mathrm{d}x+\alpha_{0}\int_{\Omega}|\mathcal{E}(v)|\mathrm{d}x,$$

where $\mathcal{E}(v)=\frac{1}{2}(\nabla v+\nabla v^{T})$ denotes the symmetrized derivative.

The two terms in (1.2) balance the first and second derivative of a function. The TGV model has achieved great success in image restoration [4, 21, 36, 20]. However, due to the irregularity of surfaces, to the best of our knowledge, the research on TGV with arbitrary $k$-order for the surface is limited [3].

Motivated by the good edge-preserving property of TV defined on meshes [48], we attempt to approximate the second-order ($k=2$) TGV (1.2) on triangulate surfaces, which utilizes the gradient operator and the weighted divergence operator. This approximation is not trivial. As stated in [48], the TV norm based on the gradient operator (see (2.1)) and $L_1$ norm can preserve the sharp features of meshes well. However, the gradient operator is not unique due to the edge normal $n_e$, which leads to the uncertainty of $\mathcal{E}(v)$ operator on meshes (see (2.4)). From the above observations, we find that the six freedoms of $\mathcal{E}(v)$ have a similar representation with the weighted divergence operator. The difference is the weight. Therefore, we consider relaxing the $\mathrm{TGV}_{\alpha}^{2}$ model on meshes by using the weighted divergence operator to approximate the $\mathcal{E}(v)$ operator; see Remark 2.1 for details. This relaxed model can resolve the uncertainty of $\mathcal{E}(v)$ and decrease the computational complexity. In addition, numerical experiments also show that the relaxed model can effectively preserve sharp features and overcome the staircase effects.

With the relaxed $\mathrm{TGV}_{\alpha}^{2}$ model, we introduce an iterative two-stage mesh denoising method. In the first stage, we propose a facet normal filtering method based on the relaxed TGV model. A robust vertex updating strategy is designed in the second stage, which can prevent edge flips and improve reconstruction quality. The augmented Lagrangian method (ALM) is used to solve the nondifferential optimization problem. Numerical experiments show that this newly developed model can remove noise efficiently while preserving sharp features and structures. In addition, the triangle's quality of mesh surfaces is also guaranteed with the robust vertex updating stage. Our method verifies the superiority of several existing mesh denoising approaches visually and quantitatively at reasonable costs.

**1.1. Related work.** A variety of mesh denoising algorithms have been developed in the last two decades. Fast speed and excellent results with high-quality triangles are vital factors to evaluate mesh denoising methods. In the following, we give an overview of the current state-of-the-art methods.

Early methods of [14], [34], and [12] use isotropic Laplace operator or mean curvature flow to filter noise, which are simple and fast. However, these methods tend to smooth the meshes and cannot preserve sharp features well.

To better preserve sharp features, several anisotropic diffusion and bilateral methods are explored [2], [10], [24], [33], [15], [19]. The first four methods filter meshes by discretizing diffusion operators into different forms. The latter two methods extend the bilateral filtering in image fields to mesh surfaces. This type of method proposes to filter vertex position directly. However, they sometimes smooth the sharp features.

The third type is a two-stage method based on normal filtering and vertex updating [30], [29], [50], [49], [46], [48], [45], [41], [5]. These two-stage methods first filter the face normals (i.e., triangle normals) and then update the vertex positions according to the filtered face normals. The first five methods filter normals by averaging neighborhood normals with different weights. They can preserve most of the sharp features. However, these methods fail to recover small-scale features. [48] extends the TV regularization to the mesh surface and uses it to filter normals. This method can preserve sharp features well while producing staircase effects for meshes with higher-level noise. [45] and [41] adopt the voting tensor and low rank strategies to filter normals, respectively. However, the two methods cannot guarantee the triangle quality for mesh with higher-level noise, and the computation cost of [41] is high. The method of [5] proposes to filter the normals by anisotropic diffusion process. After vertex updating, they implement a geometric aliasing correction to optimize the alignment of the edges to the mesh features, where promising results are obtained.

The fourth method is based on prefiltering, vertex classification, and applying different regularization to remove noise, which include [13], [40], [18], [26], [9], [23], [1], [16]. [13] uses consistent subneighborhoods to classify vertices and applies different denoising methods for different classes of vertices. The methods of [40], [23], and [16] contain prefiltering, feature detection, and $L_1$-based feature recovery. These methods can recover sharp features well. However, these methods sometimes produce wavy edges. [18] explores the $L_0$ minimization of a new defined edge operator to filter the meshes. [26] revisits the classical shock filter in image processing and extends it to mesh processing. [9] studies the property of gradient domain of meshes to devise mesh processing techniques. [1] introduces a coarse-to-fine graph spectral processing approach. In the coarse step, they use the Bayesian learning method to identify the noise level of the features and then iteratively smooth face normals and vertices in the fine step.

The last method is based on the learning techniques rising in recent years. See, e.g., [39], [38], and [22]. [39] uses cascaded normal regression (CNR) to learn the relation between noisy inputs and ground truth. This method can recover structures of meshes with less sharp features effectively. However, for meshes with sharp features, it produces wavy edges and smooths the fine details excessively. To overcome this drawback, [38] introduces a new data-driven method to further improve the recovery of sharp features. [22] presents a pioneering work named deep normal filtering network (DNF-Net). The DNF-Net utilizes the training data to enhance denoising performance and avoids manual input to extract features. This learning method can produce satisfying results to a certain extent. However, it sometimes fails in preserving sharp features.

**1.2. Our contribution and paper organization.** The contributions of the paper are summarized as follows:

- We give the definition of a relaxed second-order TGV model on triangulated surfaces, which combines the gradient operator and the weighted divergence operator.
- We introduce a feature preserving facet normal filtering method based on the relaxed model.
- A robust vertex position updating technique is explored, which can prevent edge flips and guarantee the triangle's quality of meshes.
- The ALM is presented to solve the nondifferentiable problem.
- Our approach is discussed and compared to several typical existing methods in various aspects, including parameter settings, iterative numbers, denoising effects, and computational efficiency. Numerical experiments show that our method outperforms existing methods at reasonable CPU costs.

The remaining of the paper is structured as follows. Section 2 gives some notations, operators, and the definition of relaxed second-order TGV model. In section 3, we present the iterative two-stage mesh denoising method. Section 4 introduces the ALM for solving the proposed nondifferential problem. In section 5, our denoising method is discussed and compared to typical existing methods in various aspects, including parameter settings, denoising effects, and computational efficiency. Conclusions and future work are given in section 6.

## 2. Relaxed second-order TGV.

In this section, we first give a brief introduction to some notations. In addition, for self-inclusion, we review several essential spaces and derivative operators on triangulated surfaces defined by [48] and then introduce the relaxed second-order TGV model on triangulated surfaces.

### 2.1. Notation.

Without loss of generality, we denote a triangulated surface with no degenerate triangles as $M \subset \mathbb{R}^3$. The set of vertices, edges and triangles of $M$ are denoted as $\{p_i : i = 0, 1, \ldots, \mathrm{P}-1\}$, $\{e_i : i = 0, 1, \ldots, \mathrm{E}-1\}$, and $\{\tau_i : i = 0, 1, \ldots, \mathrm{T}-1\}$. Here P, E, and T are the numbers of points, edges, and triangles, respectively. If $p$ is an endpoint of an edge $e$, then we denote it as $p \prec e$. Similarly, that $e$ is an edge of a triangle $\tau$ is denoted as $e \prec \tau$. Assume that all the triangles are with anticlockwise orientation and all edges are with fixed orientations which are randomly chosen. For an edge $e \prec \tau$, if the orientation of $e$ is consistent with the orientation of $\tau$, then $\mathrm{sgn}(e, \tau) = 1$; otherwise, $\mathrm{sgn}(e, \tau) = -1$.

### 2.2. Spaces and operators on $M$.

The piecewise constant function space over $M$ is denoted as $\mathbf{U} = \mathbb{R}^{\eta \times \mathrm{T}}$ over $M$. For any $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{\mathrm{T}-1}) \in \mathbf{U}$, the gradient operator is defined as

$$\nabla : \mathbf{u} \to \nabla\mathbf{u}, \quad \nabla\mathbf{u}|_e = (\nabla\mathbf{u})_e \vec{n}_e = [\mathbf{u}]_e \ \forall e,$$

where

$$[\mathbf{u}]_e = \begin{cases} (\sum\limits_{\tau \prec e} \mathbf{u}|_\tau \mathrm{sgn}(e, \tau))\vec{n}_e, & e \nsubseteq \partial M, \\ 0, & e \subseteq \partial M, \end{cases} \tag{2.1}$$

and $\vec{n}_e = (n_{ex}, n_{ey}, n_{ez})$ is the normal of edge $e$.

The range of $\nabla$ is then rewritten as $\mathbf{V}$, i.e., $\mathbf{V} = \mathrm{Range}(\nabla)$. For $\mathbf{v} \in \mathbf{V}$, as the divergence operator, $\mathrm{div} : \mathbf{V} \to \mathbf{U}$, is the adjoint operator of $-\nabla$, the form of the weighted divergence operator is shown as

$$(\mathrm{div}(w\mathbf{v}))|_\tau = \frac{-1}{s_\tau} \sum_{e \prec \tau, e \nsubseteq \partial M} w_e \mathbf{v}_e \mathrm{sgn}(e, \tau) l_e, \tag{2.2}$$

where $s_\tau$ and $l_e$ are the area of triangle $\tau$ and the length of edge $e$, respectively, and $w_e = e^{-\|(\nabla \mathbf{u})_e\|^2}$ is a Gauss kernel weight to alleviate the smoothing of sharp features.

For $\mathbf{u}^1, \mathbf{u}^2, \mathbf{u} \in \mathbf{U}$, $\mathbf{v}^1, \mathbf{v}^2, \mathbf{v} \in \mathbf{V}$, the inner products and norms in $\mathbf{U}$ and $\mathbf{V}$ are as follows:

$$(\mathbf{u}^1, \mathbf{u}^2)_{\mathbf{U}} = \sum_{1 \le i \le \eta} \sum_{\tau} u^1_{i,\tau} u^2_{i,\tau} s_\tau, \quad \|\mathbf{u}\|_{\mathbf{U}} = \sqrt{(\mathbf{u}, \mathbf{u})_{\mathbf{U}}},$$

$$(\mathbf{v}^1, \mathbf{v}^2)_{\mathbf{V}} = \sum_{1 \le i \le \eta} \sum_{e} v^1_{i,e} v^2_{i,e} l_e, \quad \|\mathbf{v}\|_{\mathbf{V}} = \sqrt{(\mathbf{v}, \mathbf{v})_{\mathbf{V}}}.$$

**2.3. Relaxed second-order TGV.** For $\mathbf{u} \in \mathbf{U}, \mathbf{v} \in \mathbf{V}$, we define the following relaxed second-order TGV model combining the gradient operator (2.1) and the weighted divergence operator (2.2):

$$\begin{aligned} H_{\mathrm{rtgv}}(\mathbf{u}) &= \min_{\mathbf{v}} \|\nabla \mathbf{u} - \mathbf{v}\|_1 + \alpha \|\mathrm{div}(w\mathbf{v})\|_1 \\ &= \min_{\mathbf{v}} \sum_e \sqrt{\sum_{k=1}^{\eta} |\nabla \mathbf{u}_k - \mathbf{v}_k|_e^2 l_e} + \alpha \sum_\tau \sum_{k=1}^{\eta} \sqrt{(\mathrm{div}(w\mathbf{v}_k))_\tau^2} s_\tau, \end{aligned} \tag{2.3}$$

where the first term and the second term represent the first-order and second-order differential information, respectively, and $\alpha$ is a positive parameter balancing between first-order and second-order derivatives. The relationship between our relaxed second-order TGV model (2.3) and the second-order TGV model (1.2) is shown in Remark 2.1.

*Remark* 2.1. Our relaxed TGV model (2.3) is motivated by the second-order ($k = 2$) TGV model (1.2) in image processing and the TV norm defined in [48] for mesh denoising. As stated in [48], the TV based on the gradient operator (2.1) (piecewise constant space) has the better edge-preserving property (which is a challenge of mesh denoising problem) than that of the TV norm based on the piecewise linear space in [43]. Therefore, for better sharp feature preserving, we need to choose the TV norm based on the piecewise constant space, not the piecewise linear space. Based on the gradient operator (2.1) as well as the divergence operator div being the adjoint operator of $-\nabla$, for $v \in V, \phi \in U$, we then have $(v, \nabla\phi)_V = -(\mathrm{div}(v), \phi)_U$ (here, for simplicity, we use scalar case to explain the relation). By the above formula, we can

give definition of $\mathcal{E}(v)$ restricted in triangle $\tau$ with the following form:

(2.4)
$$(\mathcal{E}(v))_\tau = \begin{pmatrix} v_{1x} & \frac{v_{1y}+v_{2x}}{2} & \frac{v_{1z}+v_{3x}}{2} \\ \frac{v_{1y}+v_{2x}}{2} & v_{2y} & \frac{v_{2z}+v_{3y}}{2} \\ \frac{v_{1z}+v_{3x}}{2} & \frac{v_{2z}+v_{3y}}{2} & v_{3z} \end{pmatrix}$$
$$= \begin{pmatrix} \frac{-1}{s_\tau}\sum_{e\prec\tau} v_e n_{ex}^2 \mathrm{sgn}(e,\tau) l_e & \frac{-1}{2s_\tau}\sum_{e\prec\tau} v_e n_{ex} n_{ey} \mathrm{sgn}(e,\tau) l_e & \frac{-1}{2s_\tau}\sum_{e\prec\tau} v_e n_{ex} n_{ez} \mathrm{sgn}(e,\tau) l_e \\ \frac{-1}{2s_\tau}\sum_{e\prec\tau} v_e n_{ey} n_{ez} \mathrm{sgn}(e,\tau) l_e & \frac{-1}{s_\tau}\sum_{e\prec\tau} v_e n_{ey}^2 \mathrm{sgn}(e,\tau) l_e & \frac{-1}{2s_\tau}\sum_{e\prec\tau} v_e n_{ey} n_{ez} \mathrm{sgn}(e,\tau) l_e \\ \frac{-1}{2s_\tau}\sum_{e\prec\tau} v_e n_{ez} n_{ex} \mathrm{sgn}(e,\tau) l_e & \frac{-1}{2s_\tau}\sum_{e\prec\tau} v_e n_{ez} n_{ey} \mathrm{sgn}(e,\tau) l_e & \frac{-1}{s_\tau}\sum_{e\prec\tau} v_e n_{ez}^2 \mathrm{sgn}(e,\tau) l_e \end{pmatrix}.$$

However, the gradient operator (2.1) is not unique due to the nonuniqueness of normal $\vec{n}_e$. There is some uncertainty to define the higher-order $\mathcal{E}(v)$ strictly by the gradient operator (2.1). By studying formula (2.4) carefully, we find that $(\mathcal{E}(v))_\tau$ is determined by six freedoms $(v_{1x}, v_{1y}+v_{2x}, v_{1z}+v_{3x}, v_{2y}, v_{2z}+v_{3y}, v_{3z})$, which have the similar representation with the weighted divergence (2.2). The difference is the weight $|n_{ei}n_{ej}| \le 1, (i,j = x,y,z)$ and $w_e$. If the weight is devised appropriately, the $(\mathcal{E}(v))_\tau$ can be simplified and approximated by the weighted divergence. Therefore, we consider relaxing $(\mathcal{E}(v))_\tau$ to the weighted divergence (2.2) by setting weight $w_e$ as the Gauss kernel. Strictly, this relaxation may affect the accuracy of the solution compared to the $(\mathcal{E}(v))_\tau$, as $n_{ei}n_{ej}$ and $w_e$ are not exactly consistent. However, the solution is acceptable for application. Numerical experiments also show that the relaxed model is effective in avoiding staircase and preserving sharp features. Furthermore, the relaxation can resolve the uncertainty and reduce computational complexity of $(\mathcal{E}(v))_\tau$ from six freedoms to one freedom.

The relaxed model (2.3) can be used for triangulated surfaces or point clouds processing. In the paper, we focus on the application in mesh denoising with $\eta = 3$.

## 3. Mesh denoising based on relaxed second-order TGV model (2.3).

As mentioned in the previous sections, the critical difficulty in mesh denoising is, while removing the noise, to preserve surface features (such as sharp edges) without generating false features. Most existing approaches either produce staircase effects or overly smooth the fine details. Here, we propose an iterative two-stage mesh denoising method to overcome these problems. Our approach involves two stages, i.e., face normal filtering followed by vertex updating.

### 3.1. Normal filtering.

Assuming $\mathbf{N}^{in} \in \mathbf{U}$ is the set of triangle normals of an observed noisy mesh, which is degraded from the normals $\mathbf{N}$ of a clean mesh, we propose to filter $\mathbf{N}^{in}$ by the minimizing following convex optimization problem based on (2.3):

$$\min_{\mathbf{N}\in\mathbf{U}} H_{\mathrm{rtgv}}(\mathbf{N}) + \frac{\beta}{2}||\mathbf{N}-\mathbf{N}^{in}||_{\mathbf{U}}^2, \tag{3.1}$$

where $H_{\mathrm{rtgv}}$ is defined in (2.3) and $\beta$ is a positive parameter.

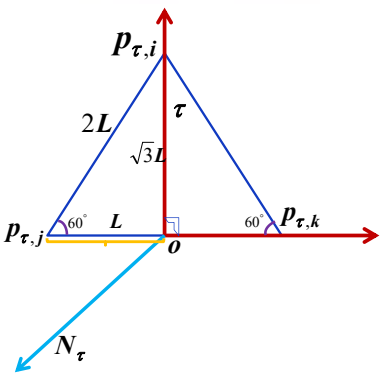


**Figure 1.** *Interpretation of* $\mathbf{g}(\mathbf{p},\mathrm{N})_\tau$ *in* (3.2).

**3.2. Vertex updating.** After optimizing the face normals through (3.1), we update the noisy vertex positions $\mathbf{p}^{in}$ by the following convex optimization model:

$$\min_{\mathbf{p}} \sum_{\tau} |\mathbf{g}(\mathbf{p},\mathbf{N})_\tau|^2 + \beta_0 \sum_{\tau} |\mathbf{h}(\mathbf{p},\mathbf{N})_\tau|^2 + \beta_1 \|\mathbf{p}-\mathbf{p}^{in}\|_2^2, \tag{3.2}$$

where $\beta_0, \beta_1$ are positive parameters and for a triangle $\tau = (\mathbf{p}_{\tau,i}, \mathbf{p}_{\tau,j}, \mathbf{p}_{\tau,k})$, $\mathbf{g}_\tau, \mathbf{h}_\tau$ are defined as follows:

$$\mathbf{g}(\mathbf{p},\mathbf{N})_\tau = \begin{cases} \sqrt{3}(\mathbf{p}_{\tau,j}-\mathbf{p}_{\tau,i})\times\mathbf{N}_\tau + 2\mathbf{p}_{\tau,k} - (\mathbf{p}_{\tau,j}+\mathbf{p}_{\tau,i})), \\ \sqrt{3}(\mathbf{p}_{\tau,k}-\mathbf{p}_{\tau,j})\times\mathbf{N}_\tau + 2\mathbf{p}_{\tau,i} - (\mathbf{p}_{\tau,j}+\mathbf{p}_{\tau,k})), \\ \sqrt{3}(\mathbf{p}_{\tau,i}-\mathbf{p}_{\tau,k})\times\mathbf{N}_\tau + 2\mathbf{p}_{\tau,j} - (\mathbf{p}_{\tau,i}+\mathbf{p}_{\tau,k})), \end{cases}$$

$$\mathbf{h}(\mathbf{p},\mathbf{N})_\tau = \begin{cases} (\mathbf{p}_{\tau,j}-\mathbf{p}_{\tau,i})\cdot\mathbf{N}_\tau, \\ (\mathbf{p}_{\tau,k}-\mathbf{p}_{\tau,j})\cdot\mathbf{N}_\tau, \\ (\mathbf{p}_{\tau,i}-\mathbf{p}_{\tau,k})\cdot\mathbf{N}_\tau. \end{cases}$$

The first term of (3.2) restrains the outer product between the edge of a triangle $\tau$ and normal $\mathbf{N}_\tau$ equal to the midline vector pointing to the edge, which aims at preventing edge flip and constraining the triangles of a mesh as equilateral as possible; see Remark 3.1 for details. The second term restricts the normal of a triangle perpendicular to its three edges.

*Remark* 3.1. Assuming a regular triangle $\tau = (\mathbf{p}_{\tau,i}, \mathbf{p}_{\tau,j}, \mathbf{p}_{\tau,k})$ in Figure 1 with the length of edge being $2L$, we then have the local coordinate system $(\mathbf{N}_\tau, \frac{\mathbf{p}_{\tau,k}-o}{|\mathbf{p}_{\tau,k}-o|}, \frac{\mathbf{p}_{\tau,i}-o}{|\mathbf{p}_{\tau,i}-o|})$ of a triangle $\tau$. According to the outer product formula, we then get

$$\begin{aligned} &\mathbf{N}_\tau \times \frac{\mathbf{p}_{\tau,k}-o}{|\mathbf{p}_{\tau,k}-o|} = \frac{\mathbf{p}_{\tau,i}-o}{|\mathbf{p}_{\tau,i}-o|} \Rightarrow \mathbf{N}_\tau \times \frac{\mathbf{p}_{\tau,k}-o}{L} = \frac{\mathbf{p}_{\tau,i}-o}{\sqrt{3}L} \\ &\Rightarrow \sqrt{3}\mathbf{N}_\tau \times (\mathbf{p}_{\tau,k}-o) = \mathbf{p}_{\tau,i}-o \\ &\Rightarrow \sqrt{3}\mathbf{N}_\tau \times \frac{(\mathbf{p}_{\tau,k}-\mathbf{p}_{\tau,j})}{2} = \mathbf{p}_{\tau,i} - \frac{(\mathbf{p}_{\tau,k}+\mathbf{p}_{\tau,j})}{2} \\ &\Rightarrow \sqrt{3}\mathbf{N}_\tau \times (\mathbf{p}_{\tau,k}-\mathbf{p}_{\tau,j}) = 2\mathbf{p}_{\tau,i} - (\mathbf{p}_{\tau,k}+\mathbf{p}_{\tau,j}) \\ &\Rightarrow -\sqrt{3}(\mathbf{p}_{\tau,k}-\mathbf{p}_{\tau,j})\times\mathbf{N}_\tau = 2\mathbf{p}_{\tau,i} - (\mathbf{p}_{\tau,k}+\mathbf{p}_{\tau,j}). \end{aligned} \tag{3.3}$$

**3.3. Boosting technique.** Moreover, to improve our mesh denoising model involving (3.1) and (3.2) for dealing with higher-level noise, we adopt the following boosting technique, which has been used extensively in image processing [35, 31, 25, 8, 32, 27, 37]. It generally means a procedure calling a processing algorithm iteratively,

$$[\hat{\mathbf{N}}_{k+1}, \hat{\mathbf{p}}_{k+1}] = R(\hat{\mathbf{N}}_k, \hat{\mathbf{p}}_k, \mu), \tag{3.4}$$

where $R()$ denotes the restoration method in (3.1) and (3.2).

Specifically, according to (3.4), we are proposing the following iterative procedure.

**Algorithm 3.1** The boosting iterative procedure

1. **Initialization**:
   $\hat{\mathbf{N}}_0 = \mathbf{N}^{in}$, $\hat{\mathbf{p}}_0 = \mathbf{p}^{in}, \mu = 1.3, K = 3$;
2. **For** $k = 0, 1, \ldots, K$
   2.1) **Compute** $\hat{\mathbf{N}}_{k+1}$ by

$$\hat{\mathbf{N}}_{k+1} = \arg\min_{\mathbf{N}\in\mathbf{U}} H_{\mathrm{rtgv}}(\mathbf{N}) + \frac{\beta}{2}||\mathbf{N} - \hat{\mathbf{N}}_k||^2_{\mathbf{U}}. \tag{3.5}$$

   2.2) **Compute** $\hat{\mathbf{p}}_{k+1}$ by

$$\begin{aligned}\hat{\mathbf{p}}_{k+1} = \arg\min_{\mathbf{p}} &\sum_{\tau} |\mathbf{g}(\mathbf{p}, \hat{\mathbf{N}}_{k+1})_\tau|^2 + \beta_0 \sum_{\tau} |\mathbf{h}(\mathbf{p}, \hat{\mathbf{N}}_{k+1})_\tau|^2 \\ &+ \beta_1 \|\mathbf{p} - \hat{\mathbf{p}}_k\|_2^2.\end{aligned} \tag{3.6}$$

   2.3) **Update** $\beta, \beta_1$:

$$\beta = \mu\beta, \qquad \beta_1 = \mu\beta_1.$$

As the relaxed model in (3.1) is nondifferentiable, it is difficult to solve by conventional methods. Recently, variable splitting and the ALM [17] have been proven to be very efficient for such problems [42, 44, 43, 48]. Moreover, the vertex updating problem (3.6) is a convex quadratic programming, and can be directly solved by various numerical packages, such as Eigen, Taucs, MKL. In the following, we present the ALM for solving (3.5).

**4. ALM for solving (3.5).** By introducing two auxiliary variables $\mathbf{q} \in \mathbf{V}$ and $\mathbf{z} \in \mathbf{U}$, we reformulate (3.5) to be the following equality constrained problem:

$$\begin{aligned}\min_{\substack{\mathbf{N}\in\mathbf{U},\mathbf{v}\in\mathbf{V},\\ \mathbf{q}\in\mathbf{V},\mathbf{z}\in\mathbf{U}}} &||\mathbf{q}||_1 + \alpha||\mathbf{z}||_1 + \frac{\beta}{2}||\mathbf{N} - \hat{\mathbf{N}}_k||^2_{\mathbf{U}} \\ \text{s.t.} \quad &\mathbf{q} = \nabla\mathbf{N} - \mathbf{v}, \mathbf{z} = \mathrm{div}(w\mathbf{v}).\end{aligned} \tag{4.1}$$

To solve (4.1), we define the following augmented Lagrangian functional:

$$\begin{aligned}\mathcal{L}(\mathbf{N},\mathbf{v},\mathbf{q},\mathbf{z};\lambda_{\mathbf{q}},\lambda_{\mathbf{z}}) = &||\mathbf{q}||_1+\alpha||\mathbf{z}||_1+(\lambda_{\mathbf{z}},\mathbf{z}-\mathrm{div}(w\mathbf{v}))_{\mathbf{U}}+\frac{r_{\mathbf{z}}}{2}\|\mathbf{z}-\mathrm{div}(w\mathbf{v})\|_{\mathbf{U}}^2\\ &+(\lambda_{\mathbf{q}},\mathbf{q}-(\nabla\mathbf{N}-\mathbf{v}))_{\mathbf{V}}+\frac{r_{\mathbf{q}}}{2}\|\mathbf{q}-(\nabla\mathbf{N}-\mathbf{v})\|_{\mathbf{V}}^2\\ &+\frac{\beta}{2}||\mathbf{N}-\hat{\mathbf{N}}_k||_{\mathbf{U}}^2.\end{aligned} \tag{4.2}$$

It has been demonstrated that the solution of (4.1) is equivalent to the following saddle-point problem:

$$\max_{\substack{\lambda_{\mathbf{q}}\in\mathbf{V},\\ \lambda_{\mathbf{z}}\in\mathbf{U}}}\ \min_{\substack{\mathbf{N}\in\mathbf{U},\mathbf{v}\in\mathbf{V},\\ \mathbf{q}\in\mathbf{V},\mathbf{z}\in\mathbf{U}}} \mathcal{L}(\mathbf{N},\mathbf{v},\mathbf{q},\mathbf{z};\lambda_{\mathbf{q}},\lambda_{\mathbf{z}}). \tag{4.3}$$

We then iteratively solve the saddle-point problem by separating (4.2) into the following four subproblems:

- The $\mathbf{N}$-sub problem: For given $\mathbf{v},\mathbf{q},\mathbf{z}$,

$$\min_{\mathbf{N}\in\mathbf{U}}(\lambda_{\mathbf{q}},-\nabla\mathbf{N})_{\mathbf{V}}+\frac{r_{\mathbf{q}}}{2}\|\mathbf{q}-(\nabla\mathbf{N}-\mathbf{v})\|_{\mathbf{V}}^2+\frac{\beta}{2}||\mathbf{N}-\hat{\mathbf{N}}_k||_{\mathbf{U}}^2. \tag{4.4}$$

  This quadratic problem can be directly solved by various numerical packages, such as MKL, Taucs, and Eigen.
- The $\mathbf{v}$-sub problem: For given $\mathbf{N},\mathbf{q},\mathbf{z}$,

$$\min_{\mathbf{v}\in\mathbf{V}}(\lambda_{\mathbf{z}},-\mathrm{div}(w\mathbf{v}))_{\mathbf{U}}+\frac{r_{\mathbf{z}}}{2}\|\mathbf{z}-\mathrm{div}(w\mathbf{v})\|_{\mathbf{U}}^2+(\lambda_{\mathbf{q}},\mathbf{v})_{\mathbf{V}}+\frac{r_{\mathbf{q}}}{2}\|\mathbf{q}-(\nabla\mathbf{N}-\mathbf{v})\|_{\mathbf{V}}^2. \tag{4.5}$$

  This quadratic problem can also be directly solved by various numerical packages, such as the $\mathbf{N}$-sub problem.
- The $\mathbf{q}$-sub problem: For given $\mathbf{N},\mathbf{v}$,

$$\min_{\mathbf{q}\in\mathbf{V}}||\mathbf{q}||_1+(\lambda_{\mathbf{q}},\mathbf{q})_{\mathbf{V}}+\frac{r_{\mathbf{q}}}{2}\|\mathbf{q}-(\nabla\mathbf{N}-\mathbf{v})\|_{\mathbf{V}}^2, \tag{4.6}$$

  which has the following closed-form solution [43]:

$$\mathbf{q}_e=\begin{cases}(1-\frac{1}{r_{\mathbf{q}}|\mathbf{c}_e|})\mathbf{c}_e, & |\mathbf{c}_e|>\frac{1}{r_{\mathbf{q}}},\\ 0, & |\mathbf{c}_e|\le\frac{1}{r_{\mathbf{q}}},\end{cases} \tag{4.7}$$

  where

$$\mathbf{c}_e=\left(\nabla\mathbf{N}-\mathbf{v}-\frac{\lambda_{\mathbf{q}}}{r_{\mathbf{q}}}\right)|_e.$$

- The $\mathbf{z}$-sub problem: For given $\mathbf{v}$,

$$\min_{\mathbf{z}\in\mathbf{U}}\alpha||\mathbf{z}||_1+(\lambda_{\mathbf{z}},\mathbf{z})_{\mathbf{U}}+\frac{r_{\mathbf{z}}}{2}\|\mathbf{z}-\mathrm{div}(w\mathbf{v})\|_{\mathbf{U}}^2 \tag{4.8}$$

similarly, which also has the following closed-form solution:

$$\mathbf{z}_\tau = \begin{cases} (1 - \frac{\alpha}{r_{\mathbf{z}}|\mathbf{r}_\tau|})\mathbf{r}_\tau, & |\mathbf{r}_\tau| > \frac{\alpha}{r_{\mathbf{z}}}, \\ 0, & |\mathbf{r}_\tau| \le \frac{\alpha}{r_{\mathbf{z}}}, \end{cases} \tag{4.9}$$

where

$$\mathbf{r}_\tau = \left(\mathrm{div}(w\mathbf{v}) - \frac{\lambda_{\mathbf{z}}}{r_{\mathbf{z}}}\right)|_\tau.$$

The overall algorithm for solving (4.3) is listed in Algorithm 4.1.

**Algorithm 4.1** ALM for solving (4.3)

1. **Initialization**:
   1.1 $\lambda_{\mathbf{q}}^{-1} = 0, \lambda_{\mathbf{z}}^{-1} = 0, \mathbf{v}^{-1} = 0, \mathbf{q}^{-1} = 0, \mathbf{z}^{-1} = 0, l = 0, \varepsilon = 10^{-5}$;
2. **Repeat**
   2.1 For fixed $\mathbf{v}^{l-1}, \mathbf{q}^{l-1}$, $\mathbf{z}^{l-1}$, computing $\mathbf{N}^l$ by (4.4);
   2.2 For fixed $\mathbf{N}^l, \mathbf{q}^{l-1}$, $\mathbf{z}^{l-1}$, computing $\mathbf{v}^l$ by (4.5);
   2.3 For fixed $\mathbf{N}^l$,$\mathbf{v}^l$, computing $\mathbf{q}^l$ by (4.6);
   2.4 For fixed $\mathbf{v}^l$, computing $\mathbf{z}^l$ by (4.8);
   2.4 Update Lagrange multiplier $\lambda_{\mathbf{q}}^l, \lambda_{\mathbf{z}}^l$:

$$\lambda_{\mathbf{q}}^l = \lambda_{\mathbf{q}}^{l-1} + r_{\mathbf{q}}(\mathbf{q}^l - (\nabla\mathbf{N}^l - \mathbf{v}^l)), \lambda_{\mathbf{z}}^l = \lambda_{\mathbf{z}}^{l-1} + r_{\mathbf{z}}(\mathbf{z}^l - \mathrm{div}(w\mathbf{v})) \tag{4.10}$$

3. **Until**$(\|\mathbf{N}^l - \mathbf{N}^{l-1}\|_{\mathrm{U}} < \varepsilon)$.

**5. Experimental examples and discussions.** In this section, we present numerical experiments with our mesh denoising method based on the relaxed TGV model. The algorithm is implemented by Microsoft Visual Studio 2010. All the examples are tested on a laptop with Intel Corei7 and 16 GB RAM. Most meshes in the paper are downloaded from website[1] and are rendered using flat shading.

To demonstrate the effectiveness of our method, we discuss our algorithm from several aspects, such as parameters, iterative number, numerically error, computational costs and comparisons, and several state-of-the-art methods, including [18], [48], [49], [39], [46], [22] (These methods are abbreviated as $L_0$ [18], TV [48], GNF [49], CNR [39], RoFi [46], DNF [22], and our method is denoted as "our.") The first two approaches [18, 48] are realized by the authors. The codes of [46] and [49] are provided by the authors. All these methods have parameters. For fairness, the parameters for these four methods are set by the statements in their paper, which are optimized to the smallest error values as well as the visually best denoising results. For parameters in our method, we set them according to the analysis in section 5.1. In addition, the results of [39] and [22] are provided by the authors.

**5.1. Parameters and iterative numbers of Algorithm 4.1.** Algorithm 4.1 consumes extremely high computational costs for the full convergence, which is unrealistic for applications.

[1] https://wang-ps.github.io/denoising.html.

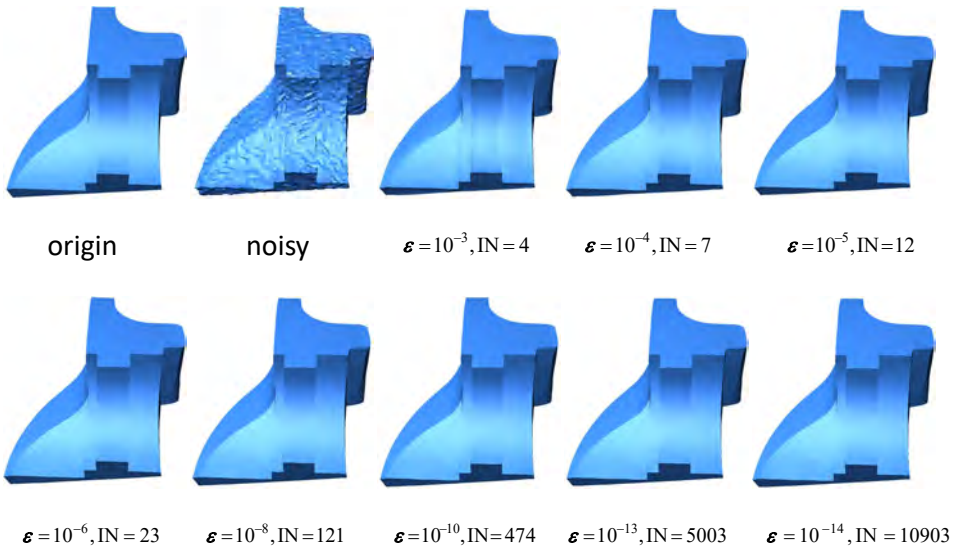


**Figure 2.** *Denoising results and iterative numbers* (IN) *for different stop conditions $\varepsilon$ with fixed parameters.*

Figure 2 shows denoising results and iterative steps of Algorithm 4.1 with different stopping criterion $\varepsilon$. As observed, the denoising results are nearly the same with $\varepsilon \leq 10^{-4}$. But the corresponding iterative numbers are increasing dramatically, especially for $\varepsilon \leq 10^{-14}$. In addition, as the algorithm is affected by noise levels, the higher the noise is, the more iterative numbers are needed. It is unfair to set a fixed iterative step. Therefore, all the results in the paper are produced with early stopping criterion $\|\mathbf{N}^l - \mathbf{N}^{l-1}\|_{\mathbf{U}} \leq \varepsilon = 10^{-5}$ of Algorithm 4.1, and the following parameters are also set under this stopping criterion. As our method contains the boosting technique Algorithm 3.1, $\varepsilon = 10^{-5}$ is enough for both low-level and high-level noisy meshes.

Our Algorithm 4.1 contains six parameters $\beta, \alpha, r_{\mathbf{q}}, r_{\mathbf{z}}, \beta_0, \beta_1$. $\beta$ and $\alpha$ are model parameters from the normal filtering optimization problem (3.1), and $r_{\mathbf{q}}$ and $r_{\mathbf{z}}$ are algorithm parameters from the ALM. In addition, $\beta_0$ and $\beta_1$ are parameters from the vertex updating optimization problem (3.2). To study the influence of the parameters, we did the experiments on almost all synthetic mesh surfaces used in this paper. According to our experiments, we have the following conclusions on these parameters.

First, $\beta$ controls the denoising effect. The smaller the $\beta$ is, the smoother the result is. If $\beta$ is too large, the algorithm fails to remove the noise. If $\beta$ is too small, the results will be overly smoothed, and some fine features will be lost. The second row of Figure 3 shows the results as well as iterative steps for different $\beta$ with the other five fixed parameters. As noted, when $\beta$ decreases, the shallow sharp feature of the Fandisk model is smoothed, and the iterative steps of Algorithm 4.1 are increasing. In addition, the parameter $\beta$ changed with the data size and degeneration level. It is difficult to compute it via formula automatically. Therefore, we need to tune this parameter manually. However, according to our experiments, our algorithm is not sensitive to this parameter. For instance, for the noisy mesh in Figure 3, the denoising results with $\beta \in [250, 450]$ are all satisfying.

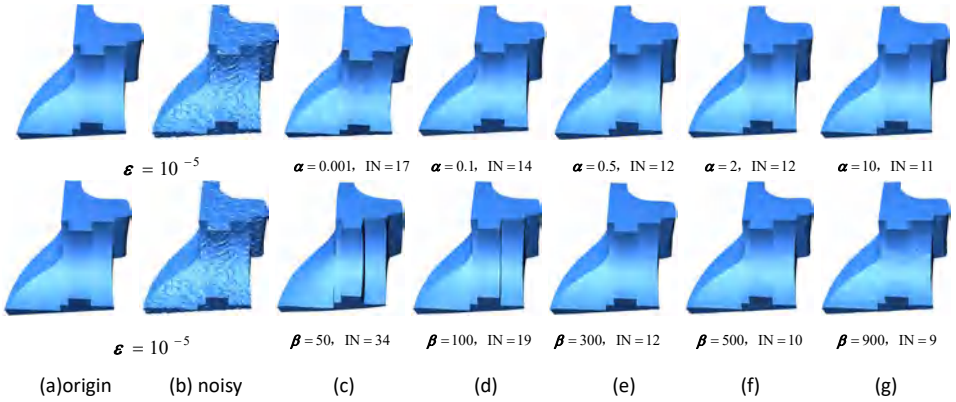


**Figure 3.** *Denoising results and iterative numbers* (IN) *for different* $\alpha, \beta$ *with fixed other parameters.*

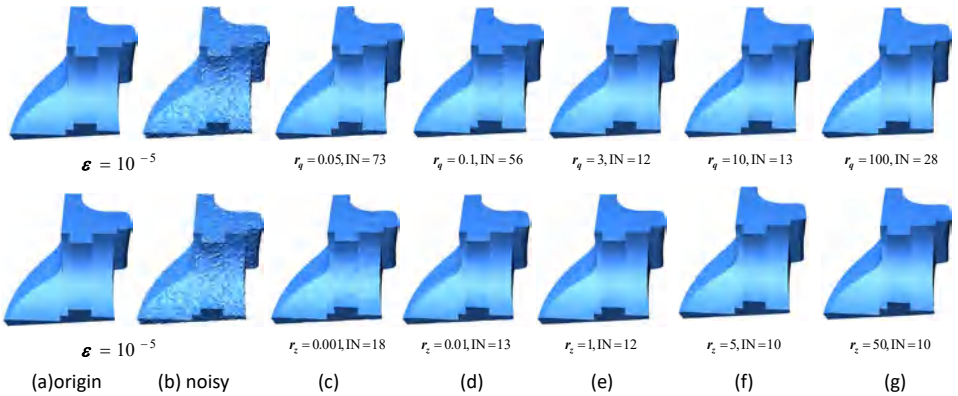


**Figure 4.** *Denoising results and iterative numbers* (IN) *for different* $r_{\mathbf{q}}, r_{\mathbf{z}}$ *with fixed other parameters.*

Second $\alpha$ affects the sharp features. The smaller the $\alpha$ is, the smoother the result is. When $\alpha$ is too small, there is some noise left on the surface. The first row of Figure 3 presents denoising results and iterative numbers for different $\alpha$ with fixed other parameters. Through a larger number of experiments, we have tested a reasonable range for $\alpha$ giving good denoising results. For meshes with many sharp features, $\alpha \in [2, 5]$; conversely, for meshes with few sharp features, $\alpha \in [0.5, 1]$.

Third, $r_{\mathbf{q}}$ and $r_{\mathbf{z}}$ are two parameters from the ALM algorithm. As our Algorithm 4.1 stops under the early stopping criterion $\|\mathbf{N}^l - \mathbf{N}^{l-1}\|_{\mathbf{U}} \leq \varepsilon = 10^{-5}$, both of them will affect the restored results and the convergence of the algorithm. When $r_{\mathbf{q}}$ is too large, some features will be destroyed. When $r_{\mathbf{q}}$ is too small, the algorithm cannot restore the data, and the iterative steps are increasing; see the results and iterative steps for different $r_{\mathbf{q}}$ with other fixed parameters in the first row of Figure 4. Moreover, $r_{\mathbf{z}}$ has an effect on preserving the sharp features. When $r_{\mathbf{z}}$ is too small, the algorithm destroys the sharp features, and the iterative steps are increasing; see the results and iterative numbers in the second row of Figure 4. Conveniently, we have tested a reasonable range for $r_{\mathbf{q}} \in [1, 5]$ and fixed $r_{\mathbf{z}}$ by $r_{\mathbf{z}} = 1$.

Finally, $\beta_0$ and $\beta_1$ are parameters from the vertex updating. Both of them have less impact on our algorithm and can be fixed by $\beta_0 = 5000$ and $\beta_1 = 1.5\beta$.

In the following subsection, we present some examples and compare our algorithm with several state-of-the-art methods both visually and quantitatively. Particularly, the sizes of triangular meshes used in the paper are shown in Table 2.

**5.2. Comparisons to other algorithms.** In Figure 5, we show denoising results for three meshes with many sharp edges by our algorithm and four other existing approaches. It can be noted that all of these methods can remove the noise and preserve most of the features. However, [18] and [48] adopt sparsity of the $L_0/L_1$ norm to filter the meshes, which can preserve the sharp edges well. Yet both methods exhibit the notable staircase effect (see the results in the third and fourth columns marked by (c) and (d)). [39] and [46] fail in preserving some sharp features and generate wavy edges (see the results in the columns marked by (e) and (f)). By comparison, for meshes of this type, our algorithm performs well. It overcomes the staircase effects and leads to better denoising results (see the results in the column marked by (g)). The zoomed views of the Joint model in the last row further indicate that our algorithm can recover more features than other methods. In addition, for the results in Figure 5, we present the quantitative average angle values $\theta$ (5.1) and visualize the triangle normal error between filtered mesh and ground-truth mesh by color maps (see the error visualizations in the second, fourth, and sixth rows of Figure 5, where the blue color means the lower errors and the red color represents the larger errors). As observed, the first four approaches cannot deal with meshes with mixed smooth parts and sharp feature parts well. Most of the $\theta$ values of our results are smaller than those of the other four methods, and the normal error visualizations further verify that our method can restore more information.

Figure 6 considers meshes with few sharp features. Results suggest that our method can deal with meshes of this kind well. However, [18] and [48] produce false edges (staircase effects) severely; see the columns marked by (c) and (d), respectively (the zoomed views of the Eroc mesh in the third row and the Carter mesh in the fifth row). [39] and [46] could not manage to retain some details and overly smooth out the details. For example, they fail in restoring the letters of the mesh in the first row, the eyes of the mesh in the second row (see the zoomed views of the Eroc model in the third row), and the narrow cylindrical area of mesh in the fourth row (see the zoomed view of the Carter model in the fifth row). Moreover, for most meshes, our method gives better quantitative results in terms of $\theta$ values.

Figure 7 depicts denoising results for meshes with noise produced by the Kinect scanner. Such a noise pattern differs from that of the Gaussian noise. As shown, our method is able to produce denoised results that are smoother and closer to the ground truths. Nevertheless, the other methods tend to retain noise in their results, exhibit staircase effects, or fail to smooth flat surfaces. In addition, the results by our method achieve the lower $\theta$ values, and the normal error visualizations again verify the advantage of our approach.

Moreover, we present comparisons with the guided mesh normal filtering method GNF [49] and the latest deep-learning method DNF [22] in Figure 8. Results show that all three methods can filter the noise and preserve most of the features. While our method performs better in recovering sharp structures and details, the deep-learning method [22] cannot preserve sharp features well and overly smooths the features (see the position marked by the red frame


**Figure 5.** *Denoising results and the normal error visualizations of Block (first row, Gaussian noise:* 0.3 *mean edge length), Sharp_Sphere (third row, Gaussian noise:* 0.2 *mean edge length), and Joint (fifth row, Gaussian noise:* 0.3 *mean edge length).* (a) *Ground truth,* (b) *noisy mesh,* (c) $L_0$ [18], (d) TV [48], (e) CNR [39], (f) RoFi [46], (g) *our.*

in Figure 8). The guided mesh normal filtering method [49] cannot deal with mesh with mixed sharp features and smooth parts well; see the mesh in the fifth row of Figure 8. The quantitative $\theta$ values and the error visualizations in Figure 8 further show the superiority of our method.

To show the validity of our vertex updating strategy (3.2), for the cube mesh with higher-level noise in Figure 9, we present the comparisons of denoising results and the corresponding reconstruction quality. As noted, $L_0$ [18] and RoFi [46] fail in preserving some sharp edges

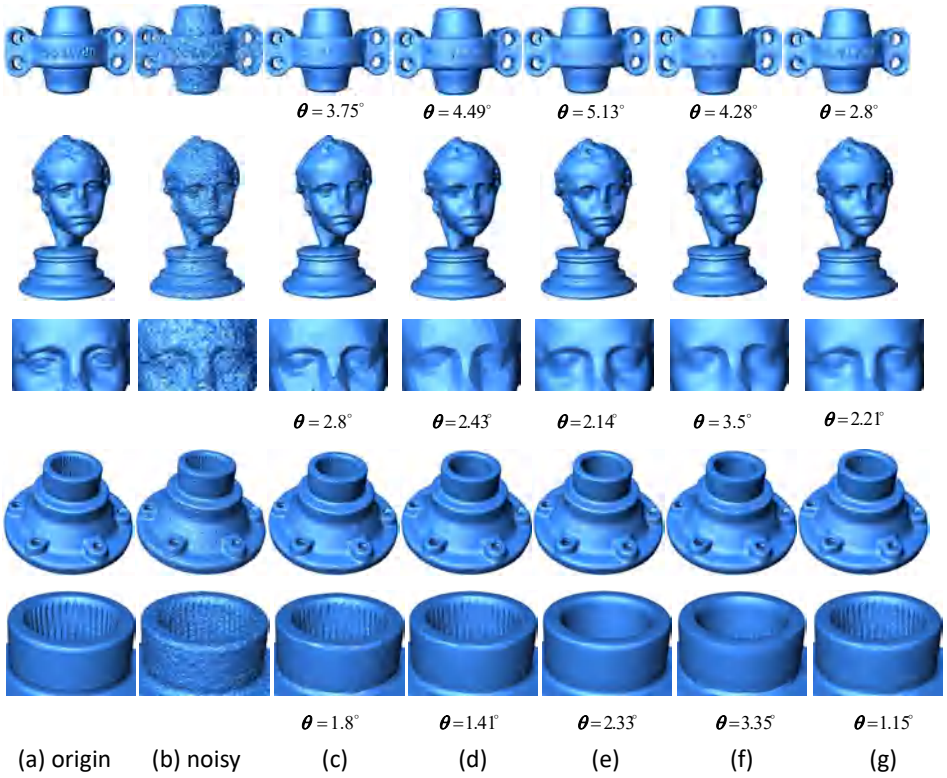


**Figure 6.** *Denoising results of Grayloc (first row, Gaussian noise:* 0.2 *mean edge length), Eros (second row, Gaussian noise:* 0.2 *mean edge length), and the Carter (fourth row, Gaussian noise:* 0.1 *mean edge length).* (a) *Ground truth,* (b) *noisy mesh,* (c) $L_0$ [18], (d) TV [48], (e) CNR [39], (f) RoFi [46], (g) *our.*

and corners. TV [48] generates flipped triangles severely. On the contrary, our method not only preserves the sharp features but also prevents edge flips. Moreover, from the outlines of one plane of the denoising cube mesh in the second row of Figure 9, we can observe that the triangle quality of our result is better than that of other three approaches. Finally, to further show the effectiveness of our method, we present Figure 10 to show results for three real scanned meshes. Our method yields very promising results.

In all, the above comparisons visually show that our denoising method can generate filtered results and avoid staircase effects, with more structures as well as high-quality triangles when compared with several typical techniques. In the following subsection, we quantitatively compare our method to other approaches by numerical errors.

**5.3. Quantitative comparison.** Similar to the metrics used in [48], the following two metrics are used to measure the distance between the denoising results and the ground-truth meshes:

- MSAE (mean square angular error),

$$\theta = \text{average}(\angle(\mathbf{n}', \mathbf{n})), \tag{5.1}$$

where average means summing up all angles and averaging and $\angle(\mathbf{n}', \mathbf{n})$ is the angle

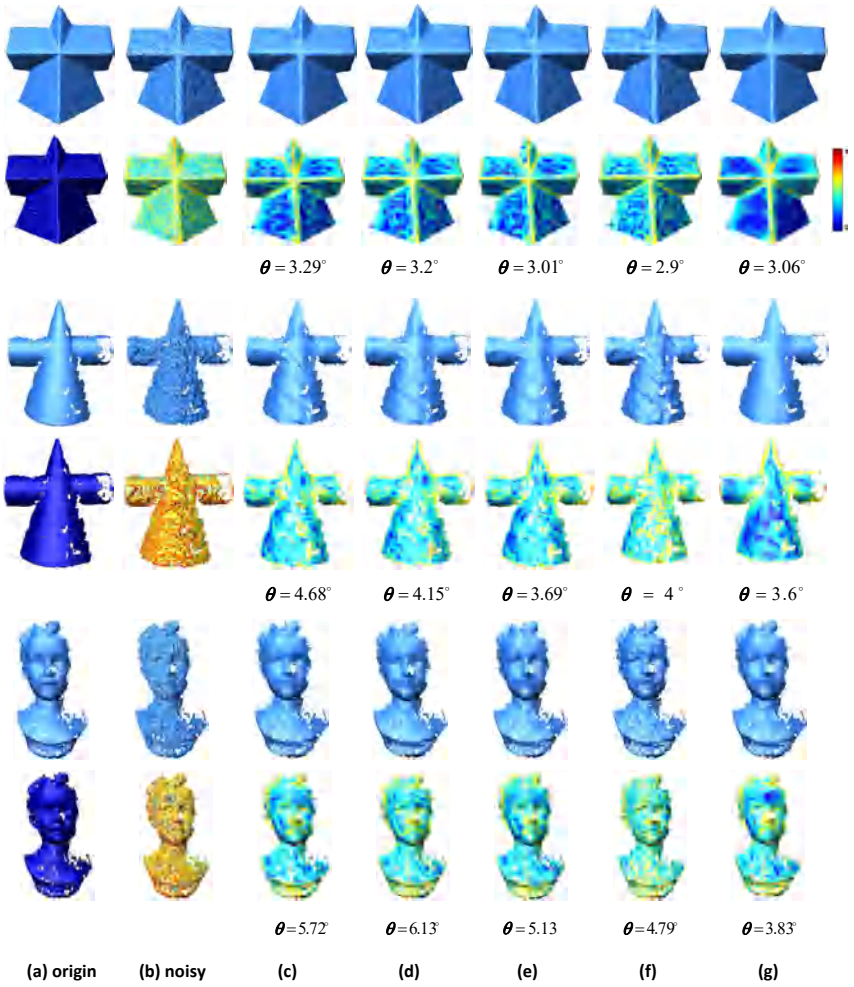


**Figure 7.** *Denoising results and normal error visualizations of Pyramid (first row), Cone (third row), and Boy (fifth row) with Kinect noise.* (a) *Ground truth,* (b) *noisy mesh,* (c) $L_0$ [18], (d) TV [48], (e) CNR [39], (f) RoFi [46], (g) *our.*

between $\mathbf{n}'$ (the triangle normal of the denoised result) and $\mathbf{n}$ (the triangle normal of the ground-truth mesh).

- The $L_2$ vertex-based mesh-to-mesh error,

$$E_{\mathbf{p},2} = \sqrt{\frac{1}{3\sum_{\tau} s_\tau} \sum_{i=0}^{\mathrm{P}-1} \Big( \sum_{\tau \in D_1(i)} s_\tau \Big) \mathrm{dist}(\mathbf{p}_i', M)^2}, \tag{5.2}$$

where $\mathrm{dist}(\mathbf{p}_i', M)$ is the distance between the filtered new vertex $\mathbf{p}_i'$ and a triangle of the origin mesh $M$ which is closest to $p_i'$ and $D_1(i)$ is the set of triangles containing $\mathbf{p}_i$.

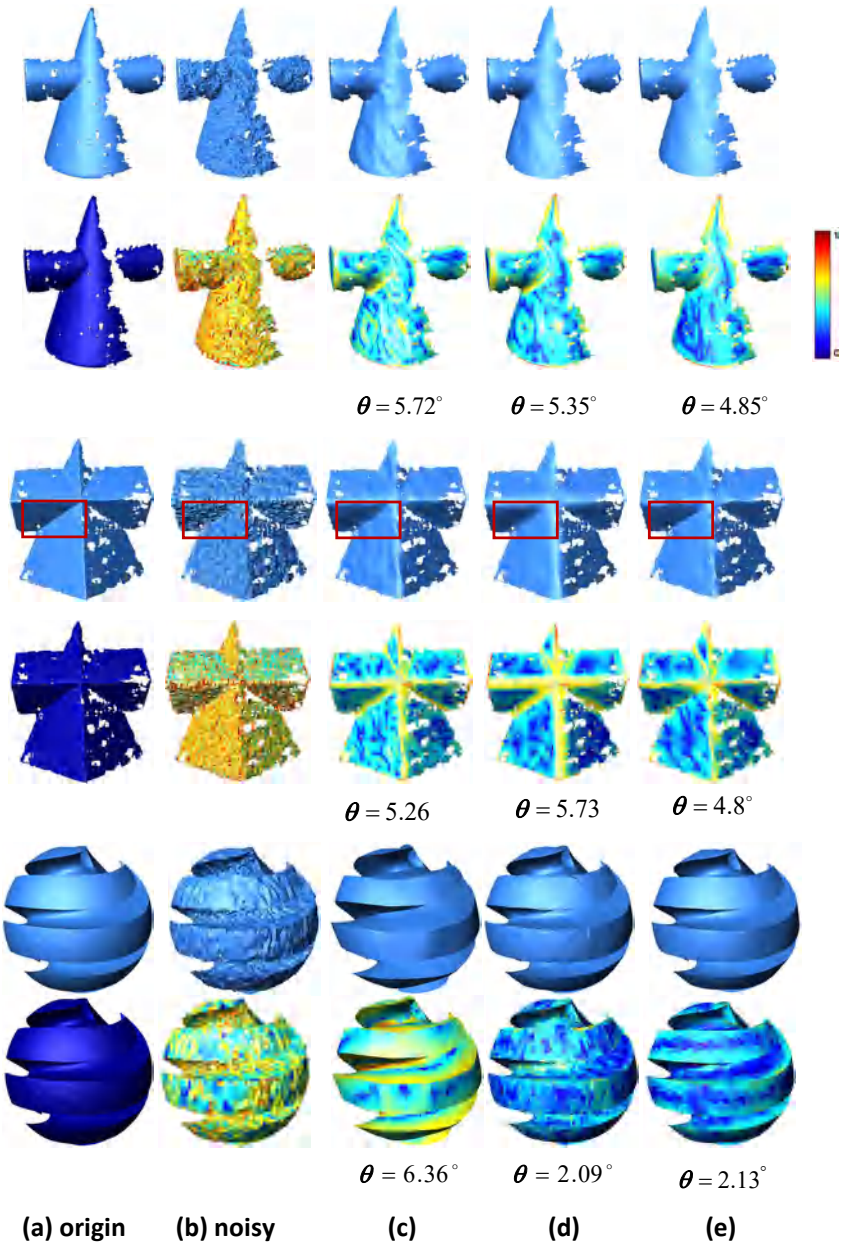


**Figure 8.** *Comparison of denoising results and normal error visualizations.* (a) *Ground truth,* (b) *noisy mesh,* (c) GNF [49], (d) DNF [22], (e) *our.*

The first metric is shown in Figures 5–8. We also compute the second metric for Figures 5–7 and present the errors in Table 1. As shown, the denoising results by our method achieve the least angular errors and vertex errors in most cases.

In all, both qualitative (visual) and quantitative comparisons show that our denoising method outperforms all the compared methods. It is also observed that the $L_0$ [18] and TV

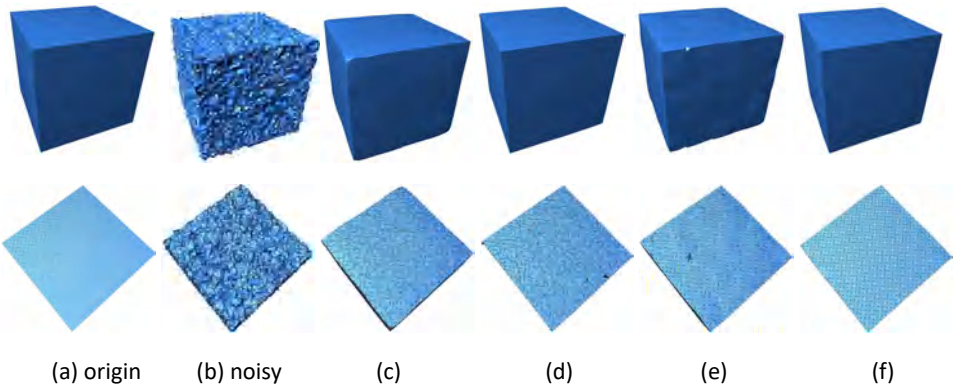


**Figure 9.** *Comparison of triangle quality of restored meshes.* (a) *Ground truth,* (b) *noisy mesh,* (c) $L_0$ [18], (d) TV [48], (e) RoFi [46], (f) *our.*

**Figure 10.** *Denoising results for real scanned models by our method.*

[48] work better for meshes with many sharp features than smooth meshes, while CNR [39] and RoFi [46] work better for meshes with less sharp features. For both types of meshes, our method achieves better results than that of all the methods compared in this paper.

**5.4. Computational costs.** We now discuss the computational costs. Overall, our algorithm is quite simple. The main step requiring CPU costs is solving the quadratic programming problems (4.4) and (4.5) (especially for (4.5)). In our tests, we note that the noise level and mesh size affect the algorithm efficiency. The more severe the noise level is, the slower the algorithm is. The larger the mesh size is, the slower the algorithms is. Table 2 presents the

**Table 1**
*$L_2$ vertex-based error comparison.*

| Model | $L_0$ [18] | TV [48] | CNR [39] | RoFi [46] | Our |
|---|---|---|---|---|---|
| Block | 0.0019 | 0.0015 | 0.00156 | 0.0019 | **0.00145** |
| Sharp_Sphere | 0.0026 | 0.0033 | **0.0017** | 0.0018 | 0.0019 |
| Joint | 0.0021 | **0.0012** | 0.0015 | 0.0015 | 0.0014 |
| Grayloc | 0.0046 | 0.0048 | 0.0069 | 0.0062 | **0.0043** |
| Eros | 0.0014 | 0.0027 | **0.0009** | 0.0012 | 0.0013 |
| Carter | 0.015 | 0.023 | 0.0098 | 0.004 | **0.0027** |
| Pyramid | 0.006 | 0.003 | 0.0031 | 0.0081 | **0.0028** |
| Cone | 0.003 | 0.0037 | 0.0029 | 0.005 | **0.0025** |
| Boy | 0.0086 | **0.0053** | 0.0062 | 0.01 | 0.0067 |

**Table 2**
*Time comparison (in seconds).*

| Model | P | T | $L_0$ [18] | TV [48] | RoFi [46] | Our |
|---|---|---|---|---|---|---|
| Block | 8771 | 17550 | 33 | 0.4 | 6.9 | 3.5 |
| Sharp_Sphere | 10443 | 20882 | 39 | 0.8 | 9.3 | 8 |
| Joint | 20902 | 41808 | 91 | 1.6 | 24.5 | 17.2 |
| Grayloc | 34274 | 685800 | 131 | 7.3 | 28.6 | 20 |
| Eros | 50002 | 100000 | 213 | 11.3 | 56.3 | 23 |
| Carter | 49988 | 100000 | 226 | 2.6 | 60 | 27 |
| Pyramid | 35261 | 69611 | 134 | 1.4 | 25.9 | 15 |
| Cone | 12407 | 23479 | 40 | 2 | 7.9 | 12.8 |
| Boy | 28019 | 53190 | 89 | 5 | 19.2 | 33.8 |

computational time for the results in Figures 5, 6, and 8 by $L_0$ [18], TV [48], RoFi [46], and our method (as the results of CNR [39] are downloaded from the home-page of the paper, we do not have the code and cannot get the runtime of their algorithm). From Table 2, we can see that the algorithm of TV [48] is the fastest and that of $L_0$ [18] is the slowest. As our relaxed TGV model needs to solve two quadratic programming problems and two subproblems with closed-form solutions, it is slower than the TV model. In all, our method can obtain better denoising results than all the other methods at reasonable CPU costs.

**5.5. Limitations.** Our mesh denoising method based on the relaxed TGV model has been demonstrated effectively in alleviating staircase effects and reconstructing results with more structures and details. However, it still has some limitations. For example, the parameter $\beta$ cannot be computed automatically. In addition, when the noise is too high, our method cannot produce satisfying results. Figure 11 presents the denoising results of a mesh with extremely high noise. None of the three methods can deal with it well, and more prior knowledge is needed.

**6. Conclusion.** In this paper, we consider relaxing the well-known second-order TGV model on triangulated surfaces. Our relaxed model combines the good properties of the gradient operator and the weighted divergence operator, which has the approximated representation with the elements of the $\mathcal{E}(v)$ operator in TGV. An iterative two-stage mesh denoising

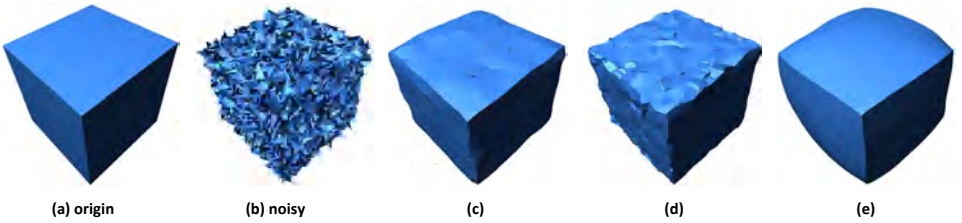


**Figure 11.** *Denoising results of mesh with higher noise.* (a) *Ground truth,* (b) *noisy mesh (Gaussian noise:* 0.9 *mean edge length),* (c) GNF [49], (d) RoFi [46], (e) *our.*

method using this relaxed TGV model is then introduced. At the first stage, we propose a facet normal filtering model based on the relaxed TGV model. A robust vertex position updating technique is designed at the second stage, which can prevent edge flips and guarantee the triangle quality of meshes. In what follows, the ALM is then employed to solve the nondifferentiable optimization problem, which produces an efficient iterative algorithm. We finally discuss and compare our approach with several typical previous methods from various aspects, including parameter setting, denoising effects, and computational efficiency. Experimental results confirm that our method outperforms all the compared methods at reasonable costs. It can overcome the staircase effects (false edges) and preserve different levels of features well. In addition, the vertex updating strategy can guarantee the mesh quality.

There are a few problems for further investigation. For instance, point cloud based on the relaxed second-order TGV model and its application will be reported in details subsequently.

**Acknowledgments.** We would like to thank the authors of CNR [39] and DNF [22] for providing their results. We also thank the authors of RoFi [46] and GNF [49] for sharing their codes.

## REFERENCES

[1] G. Arvanitis, A. Lalos, K. Moustakas, and N. Fakotakis, *Feature preserving mesh denoising based on graph spectral processing*, IEEE Trans. Vis. Comput. Graphics, 25 (2019), pp. 1513–1527.
[2] C. Bajaj and G. Xu, *Anisotropic diffusion of surfaces and functions on surfaces*, ACM Trans. Graphics, 22 (2003), pp. 4–32.
[3] K. Bredies, M. Holler, M. Storath, and A. Weinmann, *Total generalized variation for manifold-valued data*, SIAM J. Imaging Sci., 11 (2018), pp. 1785–1848.
[4] K. Bredies, K. Kunisch, and T. Pock, *Total generalized variation*, SIAM J. Imaging Sci., 3 (2010), pp. 492–526.
[5] M. Centin and A. Signoroni, *Mesh denoising with (geo)metric fidelity*, IEEE Trans. Vis. Comput. Graphics, 24 (2018), pp. 2380–2396.
[6] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, *An introduction to total variation for image analysis*, Theoret. Found. Numer. Methods Sparse Recovery, 9 (2010), pp. 263–340.
[7] T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.
[8] M. R. Charest and P. Milanfar, *On iterative regularization and its application*, IEEE Trans. Circuits Syst. Video Technol., 18 (2008), pp. 406–411.
[9] M. Chuang, S. Rusinkiewicz, and M. Kazhdan, *Gradient-domain processing of meshes*, J. Comput. Graph. Tech., 5 (2016), pp. 44–54.

[10] U. Clarenz, U. Diewald, and M. Rumpf, *Anisotropic geometric diffusion in surface processing*, in Proceedings of IEEE Visualization, 2000, pp. 397–405.
[11] L. Condat, *Discrete total variation: New definition and minimization*, SIAM J. Imaging Sci., 10 (2017), pp. 1258–1290.
[12] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, *Implicit fairing of irregular meshes using diffusion and curvature flow*, in SIGGRAPH, 1999, pp. 317–324.
[13] H. Fan, Y. Yu, and Q. Peng, *Robust feature-preserving mesh denoising based on consistent subneighborhoods*, IEEE Trans. Vis. Comput. Graphics, 16 (2010), pp. 312–324.
[14] D. A. Field, *Laplacian smoothing and Delaunay triangulations*, Commun. Appl. Numer. Methods, 4 (1988), pp. 709–712.
[15] S. Fleishman, I. Drori, and D. Cohen-Or, *Bilateral mesh denoising*, ACM Trans. Graphics, 22 (2003), pp. 950–953.
[16] M. Fumero, M. Möller, and E. Rodolà, *Nonlinear spectral geometry processing via the* TV *transform*, ACM Trans. Graphics, 39 (2020), pp. 1–16.
[17] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, Vol. 9, SIAM, Philadelphia, 1989.
[18] L. He and S. Schaefe, *Mesh denoising via* $L_0$ *minimization*, ACM Trans. Graphics, 32 (2013), pp. 1–8.
[19] T. R. Jones, F. Durand, and M. Desbrun, *Non-iterative, feature preserving mesh smoothing*, ACM Trans. Graphics, 22 (2003), pp. 943–949.
[20] M. Kang, M. Kang, and M. Jung, *Total generalized variation based denoising models for ultrasound images*, J. Sci. Comput., 72 (2017), pp. 172–197.
[21] F. Knoll, K. Bredies, T. Pock, and R. Stollberger, *Second order total generalized variation* (TGV) *for* MRI, Magn. Resonance Med., 65 (2011), pp. 480–491.
[22] X. Z. Li, R. H. Li, L. Zhu, C. W. Fu, and P. A. Heng, *DNF-Net: A deep normal filtering network for mesh denoising*, IEEE Trans. Vis. Comput. Graphics, 27 (2021), pp. 4060–4072.
[23] X. Lu, W. Chen, and S. Schaefer, *Robust mesh denoising via vertex pre-filtering and* $l_1$*-median normal filtering*, Comput. Aided Geom. Design, 54 (2017), pp. 49–60.
[24] M. Meyer, M. Desbrun, P. Schroder, and A. Barr, *Discrete differential-geometry operator for triangulated* 2*-manifolds*, in Visualization and Mathematics III, Springer, New York, 2003, pp. 35–57.
[25] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Model. Simul., 4 (2005), pp. 460–489.
[26] F. Prada and M. Kazhdan, *Unconditionally stable shock filters for image and geometry processing*, Comput. Graphics Forum, 34 (2015), pp. 201–210.
[27] Y. Romano and M. Elad, *Boosting of image denoising algorithms*, SIAM J. Imaging Sci., 8 (2015), pp. 1187–1219.
[28] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.
[29] Y. Shen and K. Barner, *Fuzzy vector median-based surface smoothing*, IEEE Trans. Vis. Comput. Graphics, 10 (2004), pp. 252–265.
[30] X. Sun, P. Rosin, R. Martin, and F. Langbein, *Fast and effective feature-preserving mesh denoising*, IEEE Trans. Vis. Comput. Graphics, 13 (2007), pp. 925–938.
[31] E. Tadmor, S. Nezzar, and L. Vese, *A multiscale image representation using hierarchical (*BV*,* $L^2$*) decompositions*, Multiscale Model. Simul., 2 (2004), pp. 554–579.
[32] H. Talebi, X. Zhu, and P. Milanfar, *How to SAIF-ly boost denoising performance*, IEEE Trans. Image Process., 22 (2013), pp. 1470–1485.
[33] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher, *Geometric surface smoothing via anisotropic diffusion of normals*, in Proceedings of IEEE Visualization, 2002, pp. 125–132.
[34] G. Taubin, *A signal processing approach to fair surface design*, in SIGGRAPH, 1995, pp. 351–358.
[35] J. W. Tukey, *Exploratory data analysis*, (1977).
[36] T. Valkonen, K. Bredies, and F. Knoll, *Total generalized variation in diffusion tensor imaging*, SIAM J. Imaging Sci., 6 (2013), pp. 487–525.
[37] S. Wali, Z. Liu, C. Wu, and H. Chang, *A boosting procedure for variational-based image restoration*, Numer. Math. Theory Methods Appl., 11 (2017), pp. 49–73.
[38] J. Wang, J. Huang, F. L. Wang, M. Wei, and J. Qin, *Data-driven geometry-recovering mesh denoising*, Comput. Aided Design, 114 (2019), pp. 133–142.

[39] P. Wang, Y. Liu, and X. Tong, *Mesh denoising via cascaded normal regression*, ACM Trans. Graphics, 35 (2016), pp. 1–12.
[40] R. Wang, Z. Yang, L. Liu, J. Deng, and F. Chen, *Decoupling noises and features via weighted $L_1$-analysis compressed sensing*, ACM Trans. Graphics, 33 (2014), pp. 1–12.
[41] M. Wei, J. Huang, X. Xie, L. Liu, J. Wang, and J. Qin, *Mesh denoising guided by patch normal cofiltering via kernel low-rank recovery*, IEEE Trans. Vis. Comput. Graphics, 25 (2019), pp. 2910–2926.
[42] C. Wu and X.-C. Tai, *Augmented Lagrangian method, dual methods, and split Bregman iteration for* ROF*, vectorial* TV*, and high order models*, SIAM J. Imaging Sci., 3 (2010), pp. 300–339.
[43] C. Wu, J. Zhang, Y. Duan, and X.-C. Tai, *Augmented Lagrangian method for total variation based image restoration and segmentation over triangulated surfaces*, J. Sci. Comput., 50 (2012), pp. 145–166.
[44] C. Wu, J. Zhang, and X.-C. Tai, *Augmented Lagrangian method for total variation restoration with non-quadratic fidelity*, Inverse Probl. Imaging, 5 (2011), pp. 237–261.
[45] S. K. Yadav, U. Reitebuch, and K. Polthier, *Mesh denoising based on normal voting tensor and binary optimization*, IEEE Trans. Vis. Comput. Graphics, 24 (2018), pp. 2366–2379.
[46] S. K. Yadav, U. Reitebuch, and K. Polthier, *Robust and high fidelity mesh denoising*, IEEE Trans. Vis. Comput. Graphics, 25 (2019), pp. 2304–2310.
[47] H. Zhang and C. Wang, *Total variation diffusion and its application in shape decomposition*, Comput. Graphics, 90 (2020), pp. 95–107.
[48] H. Zhang, C. Wu, J. Zhang, and J. Deng, *Variational mesh denoising using total variation and piecewise constant function space*, IEEE Trans. Vis. Comput. Graphics, 21 (2015), pp. 873–886.
[49] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, and L. Liu, *Guided mesh normal filtering*, Comput. Graphics Forum, 34 (2015), pp. 23–34.
[50] Y. Zheng, H. Fu, O.-C. Au, and C. Tai, *Bilateral normal filtering for mesh denoising*, IEEE Trans. Vis. Comput. Graphics, 17 (2011), pp. 1521–1530.