Mesh Total Generalized Variation for Denoising

Zheng Liu, Yanlei Li, Weina Wang, Ligang Liu, and Renjie Chen†

Abstract—Recent studies have shown that the Total Generalized Variation (TGV) is highly effective in preserving sharp features as well as smooth transition variations for image processing tasks. However, currently there is no existing work that is suitable for applying TGV to 3D data, in particular, triangular meshes. In this paper, we develop a novel framework for discretizing second-order TGV on triangular meshes. Further, we propose a TGV-based variational method for the denoising of face normal fields on triangular meshes. The TGV regularizer in our method is composed of a first-order term and a second-order term, which are automatically balanced. The first-order term allows our TGV regularizer to locate and preserve sharp features, while the second-order term allows our regularizer to recognize and recover smoothly curved regions. To solve the optimization problem, we introduce an efficient iterative algorithm based on variable-splitting and augmented Lagrangian method. Extensive results and comparisons on synthetic and real scanning data validate that the proposed method outperforms the state-of-the-art visually and numerically.

Index Terms—Mesh denoising, total generalized variation, augmented Lagrangian method, total variation, normal filtering

1 Introduction

Mesh denoising is one of the most fundamental research topics in geometry processing. With the rapid development of 3D scanning devices and depth cameras, it has become increasingly popular and common to acquire and reconstruct meshes from the real world automatically [1]. However, the acquired meshes are inevitably contaminated by noise because of local measurement errors in the scanning process and computational errors in the reconstruction algorithm used. Noise not only degrades the quality of meshes, but also causes problems in downstream geometry processing applications [2]. Thus, mesh denoising has been a widely studied topic in recent years, whose main purpose is to remove noise while recovering geometric features as accurately as possible [3]. However, noise and geometric features are both of high frequency information, which makes it challenging to distinguish them from noisy input, especially in the presence of large noise.

To suppress noise while preserving geometric features, various mesh denoising methods have been investigated, including filtering-based methods [4], [5], [6], variational methods [7], [8], [9], [10], [11], [12], nonlocal-based methods [13], [14], [15], data-driven methods [3], [16], [17], [18], etc. Among them, variational methods have attracted much attention, as they can well preserve sharp features while suppressing noise significantly.

The variational method usually consists of a regularization term and a fidelity term. The total variation (TV) regularizer is known for its excellent edge-preserving capability in image processing, and it has been extended by Zhang et al. [8] to restore the face normal field for triangular meshes. However, as the TV regularizer uses the first-order operator, it tends to transform smooth transition variations into piecewise constant ones. Hence, the TV regularizer often suffers from staircase artifacts in smooth regions. These artifacts degrade the visual quality of the denoised result, which may induce false features that do not exist in smooth regions. To reduce the undesired artifacts from TV, several higher-order regularizer [19], [20] have been proposed. They can preserve geometric features and simultaneously prevent staircase artifacts. Unfortunately, when the noise level is high, these higher-order method may blur geometric features in varying degrees. To address these issues, it is natural to combine the first- and higher-order terms. For example, Zhong et al. [21] proposed a variational method, which combines a first- and a higher-order terms directly. Although their method performs better than TV, it still has some artifacts near sharp features and flattens fine details more or less. Thus, although some variational methods have been introduced, it is still challenging to find one regularization technique, which can effectively preserve sharp features in some parts of the surface while simultaneously recovering smooth regions in other parts.

Recently, the total generalized variation (TGV), proposed by Bredies et al. [22], has become one of the most popular regularization technique in image processing. TGV is composed of polynomials of arbitrary order, which can reconstruct piecewise polynomial functions with automatically balanced first- and higher-order variations rather than using fixed combination [23], [24]. TGV can be interpreted as combining smoothness from the first-order up to arbitrary order variations. It preserves sharp features via the first-order variations while effectively approximates smooth transition regions via the higher-order variations. As a result, it does not produce staircase artifacts. For most signal processing tasks, we believe that the second-order variant of TGV is sufficient. The reason is that most signals can be approximated with piecewise linear functions, and relatively it is more difficult to discretize higher-order TGV. Therefore, in this work, we focus on the second-order TGV, and we refer TGV in particular to its second-order version throughout the

- *Z. Liu and Y. Li are with National Engineering Research Center of Geographic Information System, School of Geography and Information Engineering, China University of Geosciences (Wuhan).*
- *W. Wang is with Department of Mathematics, Hangzhou Dianzi University.*
- *R. Chen and L. Liu are with School of Mathematical Sciences, University of Science and Technology of China.*

†*Corresponding author. E-mail:renjiec@ustc.edu.cn.*

paper.

It is non-trivial to extend typical methods for 2D image processing to 3D mesh data because of the inherent data irregularities in meshes. To the best of our knowledge, despite that TGV has achieved great success in image processing (e.g. image restoration [22], depth upsampling [23], speckle reduction [25], texture decomposition [24], image reconstruction [26], [27]), currently, there is no existing work that applies TGV to triangular meshes. In this paper, we develop a numerical framework to discretize TGV over triangular meshes. Based on this discretization, a vectorial TGV regularizer is proposed for face normal field. Then, we introduce an efficient and effective algorithm to solve the problem. The main contributions of this work include:

- We define necessary discrete operators and further use these operators to apply TGV to triangular meshes. To the best of our knowledge, this is the first numerical framework for discretizing TGV on triangular meshes.
- We present a normal filter using TGV-based regularization. Our method is able to preserve sharp features and recover smooth regions, while preventing unnatural artifacts. We solve the optimization problem using variable-splitting and augmented Lagrangian method.
- Qualitative and quantitative experiments on synthetic and scanned data show that our denoising method performs favorably against the state-of-the-art methods.

The rest of this paper is organized as follows. A brief survey of mesh denoising methods is presented in section 2. Section 3 recalls TGV in image processing. Section 4 presents the numerical framework for discretizing TGV and its vectorial version for meshes. In section 5, we propose a vectorial TGV based face normal filter, and we introduce an augmented Lagrangian method for solving the optimization problem. Section 6 shows the results of our TGV-based method and further compares it to the state-of-the-art methods visually and quantitatively. Finally, we conclude with remarks and discuss directions for future work in section 7.

2 Related work

Due to the abundance of mesh denoising methods in the literature, it is beyond our scope to review all existing work. In this section, we review four categories of research that are most relevant to this work.

Filter-based methods. The spatial filtering methods first compute filtering weights based on signal similarities, and then average the neighboring signals in each local region with the computed weights. Early spatial filtering methods directly adopt isotropic smoothing [28], [29] or anisotropic smoothing [30], [31], [32], [33], [34] to mesh vertices in order to remove noise. Although anisotropic methods are more robust against noise compared to the isotropic smoothing methods, they are still unable to preserve geometric features in the case of heavy noise.

Recently, it has become so widespread with normal filtering followed by vertex updating that it could arguably replace direct vertex position smoothing [4], [35]. Zheng et al. [5] applied the bilateral filter to face normal field. Although their method preserves geometric features, it may blur sharp features when the noise level increases. To address this problem, Zhang et al. [36] introduced a bilateral normal filter based on a well-designed guidance normal field. Later on, Zhang et al. [37] proposed a scale-aware normal filter using both static and dynamic guidance. Yadav et al. [38] proposed a normal filter based on tensor voting and binary optimization. Furthermore, Yadav et al. [39] developed a normal filter in the robust statistics framework that can preserve sharp features, but may smooth weak features and fine details. Arvanitis et al. [40] introduced a coarse-to-fine framework for the restoration of face normal field based on graph spectral processing. Zhao et al. [41] presented a feature-preserving normal filter, by first computing the guidance normal field using the graph-cut scheme, and then performing normal filtering using the guidance field.

Variational methods. For mesh denoising, variational methods aim to find and apply appropriate priors in order to formulate the denoising process as an optimization problem. Based on the prior that geometric features are sparse over the underlying surface, sparse regularizers are typically applied in the variational methods for the recovering of geometric features.

He and Schaefer [42] and Zhao et al. [43] applied the ℓ_0 minimization to triangular meshes based on the piecewise constant prior. These ℓ_0 minimization methods achieve impressive results in preserving sharp features, but inevitably flatten weak features because of their high sparsity requirement. Moreover, since the ℓ_0 minimization is a NP-problem, the computation is time consuming. Another popular sparse regularizer is the total variation (TV) minimization, which essentially imposes the first-order ℓ_1 quasi-norm. Zhang et al. [8] extended the TV regularizer for restoring the face normal field. A commonly known drawback of the TV regularizer is that it tends to produce staircase artifacts in smoothly curved regions. To address this problem, higher-order methods [19], [20] have been introduced, which can prevent producing staircase artifacts. Unfortunately, when noise level increases, these high-order methods tend to blur fine details and curve sharp features. Another technique [21] for reducing staircase artifacts is to combine a high-order term with the TV term. This straightforward combined technique reduces staircase artifacts to some extent, but unnatural artifacts may still appear around sharp features. Thus, it is still challenging to preserve sharp features while simultaneously recover smooth transition variations.

Nonlocal-based methods. Most of the above-mentioned variational methods are local (using local operators to formulate the problem). Based on the observation that pattern similarity may exist on the underlying surface, several researchers introduced nonlocal methods [13], [14], [15], [44]. These nonlocal methods first group similar patches together, and then perform a low-rank minimization on the patch group to recover pattern similarity of the underlying surface. These methods can effectively recover surfaces using the pattern similarity prior. However, due to the multi-patch collaborative mechanism, these nonlocal methods are computationally intensive, and tend to blur sharp features.

Data-driven methods. More recently, data-driven methods are receiving increased attention. Wang et al. [16] presented their pioneer work with cascaded normal regression (CNR) for face normal smoothing. Their method first learns non-linear regression functions that map filtered normal descriptors to those of the ground-truth counterparts, and

then applies the learned functions to filter normals. In order to better recover details, Wang et al. [3] and Wei et al. [17] proposed a two-step denoising framework (denoising followed by refinement). They first learn the mapping from noisy meshes to their ground-truth counterparts for smoothing face normals. Then, they recover details by learning the mapping from the filtered normals to the ground-truth. Later on, Li et al. [45] proposed a normal filtering neural network, called NormalF-Net, which consists of a denoising and refinement subnetwork. Li et al. [18] presented an end-to-end convolutional neural network, named DNF-Net, to predict filtered normals from the noisy mesh. The above mentioned data-driven methods can produce satisfactory denoising results using convolutional network. However, the performance of these methods depends on the completeness of the training data set. Moreover, the computation cost of the training process for them is usually high.

3 Background of Total Generalized Variation

Rudin, Osher, and Fatemi proposed the total variation (TV) in their seminal work [46], and it has since started the trend of applying variational methods for image processing. TV has been widely used as a regularizer for edge recovering, which are considered as the key features of images. For an image $u : \Omega \to \mathbb{R}$, TV of u is defined as follows:

$$\mathrm{TV}(u) = \int_\Omega |\nabla u|. \tag{1}$$

A commonly known drawback of TV is it tends to produce staircase artifacts in smooth transition regions of the images, as TV favors solutions that are piecewise constant. To address this problem, a more general variational method, called total generalized variation (TGV), was introduced by Bredies et al. [22]. In theory, TGV can be used to measure image characteristics up to a certain order of differentiation. As proved in [22], the first-order TGV is equivalent to TV. Thanks to its higher-order nature, TGV can eliminate staircase artifacts effectively. However, for TGV in an order that is too high, it becomes difficult to discretize and computationally expensive. Considering the trade-off between computational complexity and numerical accuracy, we focus on the second-order TGV in this work.

Given an image u, the second-order TGV of u is formulated as

$$\mathrm{TGV}(u) = \min_v \Big\{\alpha_1 \int_\Omega |\nabla u - v| + \alpha_0 \int_\Omega |\xi(v)|\Big\}, \tag{2}$$

where $\alpha_1, \alpha_0 \in \mathbb{R}^+$ are weights, and $\xi(v) = \frac{1}{2}(\nabla v + \nabla v^T)$ denotes the distributional symmetrized derivative. The 2-tensor v is converted into a vector by concatenating its columns for computational convenience. In the following, we give a more intuitive explanation for TGV. On one hand, in smooth transition regions of u, the second-order derivative $\nabla^2 u$ is small locally, and the optimum of (2) is obtained by choosing $\nabla u \approx v$ therein locally. On the other hand, in regions near edges, $\nabla^2 u$ is evidently larger than ∇u, hence the minimum of (2) tends to have $v \approx 0$ in these regions. However, this is only an intuitive assumption, the actual values of minimum v are located anywhere in the range of $[0, \nabla u]$. Under the help of edge-aware variable v, TGV automatically balances between first- and second-order variations, instead of having fixed combination of them. We refer the interested reader to [22], [26], [27] for further discussions about TGV.

For a $\mathfrak{N}$-channel image $\mathbf{u} : \Omega \to \mathbb{R}^{\mathfrak{N}}$, where $\mathbf{u} = (u_1, u_2, \ldots, u_{\mathfrak{N}})$, the vectorial TGV of $\mathbf{u}$ is formulated as

$$\mathbf{TGV}(\mathbf{u}) = \min_{\mathbf{v}} \Big\{\alpha_1 \int_\Omega \|\nabla \mathbf{u} - \mathbf{v}\| + \alpha_0 \int_\Omega \|\xi(\mathbf{v})\|\Big\}, \tag{3}$$

where $\|\nabla \mathbf{u} - \mathbf{v}\| = \big(\sum_{i=1}^{\mathfrak{N}} |\nabla u_i - v_i|^2\big)^{\frac{1}{2}}$, and $\|\xi(\mathbf{v})\| = \sum_{j=1}^{4} \|\xi_j(\mathbf{v})\| = \sum_{j=1}^{4} \big(\sum_{i=1}^{\mathfrak{N}} |\xi_j(v_i)|^2\big)^{\frac{1}{2}}$. As we can see, (3) can be applied to process multi-spectral images with the special case $\mathfrak{N} = 3$ for RGB images.

4 Discretization of Total Generalized Variation on triangular meshes

In this section, we first introduce some basic notation. Then, we elaborate on how to discretize TGV and its vectorial version over triangular meshes. Finally, we discuss the related work [8], [19] with our discretized TGV.

4.1 Notation

Let $\mathcal{M}$ be a compact triangulated surface of arbitrary topology with no degenerate triangles in $\mathbb{R}^3$. The set of vertices, edges, and triangles of $\mathcal{M}$ are denoted as $\{p_i : i = 1, \cdots, \mathrm{P}\}$, $\{e_i : i = 1, \cdots, \mathrm{E}\}$, and $\{\tau_i : i = 1, \cdots, \mathrm{T}\}$, respectively. Here P, E, and T are the numbers of vertices, edges, and triangles of $\mathcal{M}$. If p is an endpoint of an edge e, then we write $p \prec e$. Similarly, $e \prec \tau$ denotes that e is an edge of τ, and $p \prec \tau$ denotes that p is a vertex of τ.

We further define the relative orientation of an edge e w.r.t. a triangle τ, denoted by $\mathrm{sgn}(e, \tau)$, as follows. Assume all triangles are with counterclockwise orientation, while all edges are with randomly chosen orientations. For an edge $e \prec \tau$, if its orientation is consistent with the orientation of τ, then $\mathrm{sgn}(e, \tau) = 1$; otherwise $\mathrm{sgn}(e, \tau) = -1$.

4.2 Discretizing Total Generalized Variation

In order to describe piecewise constant data (e.g., face normal field) on a triangular mesh $\mathcal{M}$, we introduce the piecewise constant function space. Given mesh $\mathcal{M}$, we define the space $U = \mathbb{R}^{\mathrm{T}}$, which is isomorphic to the piecewise constant function space on $\mathcal{M}$. For example, $u = (u_1, \cdots, u_{\mathrm{T}}) \in U$, where the value of u restricted to triangle τ is u_τ, sometimes written as $u|_\tau$ for convenience. In order to further describe function v (see the definition of TGV (2)), we propose the edge function space $V = \mathbb{R}^{\mathrm{E}}$, whose elements are functions defined at the edges of $\mathcal{M}$. We sometimes also write v_e as $v|_e$, to denote the component of $v \in V$, restricted to edge e. Sometimes, we also refer to space V as the edge function space $\mathcal{E}$.

We equip space U and V with the standard Euclidean inner product and norm as follows. $\forall\, u^1, u^2, u \in U$, we have:

$$(u^1, u^2)_U = \sum_\tau u^1|_\tau u^2|_\tau \mathrm{area}(\tau), \quad \|u\|_U = \sqrt{(u, u)_U}, \tag{4}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3088118, IEEE Transactions on Visualization and Computer Graphics

where $\mathrm{area}(\tau)$ is the area of τ. $\forall\, v^1, v^2, v \in V$, we have:

$$(v^1, v^2)_V = \sum_e v^1|_e v^2|_e \mathrm{len}(e), \quad \|v\|_V = \sqrt{(v,v)_V}, \tag{5}$$

where $\mathrm{len}(e)$ is the length of e.

As discussed in [12], it is natural to define the first-order difference operator $\mathcal{D}_\mathcal{M} : U \to V$ on $\mathcal{M}$ as

$$(\mathcal{D}_\mathcal{M} u)|_e = \begin{cases} \sum\limits_{\tau, e\prec\tau} u_\tau \mathrm{sgn}(e,\tau), & e \not\subset \partial\mathcal{M} \\ 0, & e \subset \partial\mathcal{M} \end{cases}, \quad \forall e. \tag{6}$$

The adjoint operator of $\mathcal{D}_\mathcal{M}$, i.e., $\mathcal{D}^\star_\mathcal{M} : V \to U$, is given by

$$(\mathcal{D}^\star_\mathcal{M} v)|_\tau = -\frac{1}{\mathrm{area}(\tau)} \sum_{\substack{e\prec\tau, \\ e\not\subset\partial\mathcal{M}}} v_e \mathrm{sgn}(e,\tau)\mathrm{len}(e), \quad \forall \tau. \tag{7}$$

In the discrete case, for each triangle τ, there are three first-order differences over the edges along three different directions. Thus, we can approximate the gradient operator in triangle τ as,

$$\nabla u|_\tau = (\mathcal{D}_\mathcal{M} u|_{e_{1,\tau}}, \mathcal{D}_\mathcal{M} u|_{e_{2,\tau}}, \mathcal{D}_\mathcal{M} u|_{e_{3,\tau}}),$$

where $e_{i,\tau} \prec \tau, i = 1,2,3$. For convenience, we write the discrete gradient as $\nabla u = (\partial_1 u, \partial_2 u, \partial_3 u)$. It is natural to denote the second-order gradient in τ as,

$$\nabla^2 u|_\tau = \begin{pmatrix} \partial_1\partial_1 u & \partial_1\partial_2 u & \partial_1\partial_3 u \\ \partial_2\partial_1 u & \partial_2\partial_2 u & \partial_2\partial_3 u \\ \partial_3\partial_1 u & \partial_3\partial_2 u & \partial_3\partial_3 u \end{pmatrix}, \tag{8}$$

where the diagonal entries $\partial_i\partial_i u, i = \{1,2,3\}$ are the second-order directional derivatives in the same direction, while the off-diagonal entries $\partial_i\partial_j u, i \neq j$ are the second-order directional derivatives in two different directions.

To further discretize TGV, we need to define operators in the edge function space $\mathcal{E}$. For each triangle τ, $v|_\tau = (v_{e_{1,\tau}}, v_{e_{2,\tau}}, v_{e_{3,\tau}})$ denotes the values of v restricted to the three edges of τ. For brevity, the values of v in triangle τ can be written as $v|_\tau = (v_1, v_2, v_3)$. Then, the gradient (in three different directions) is given by

$$\nabla v|_\tau = \begin{pmatrix} \partial_1 v_1 & \partial_2 v_1 & \partial_3 v_1 \\ \partial_1 v_2 & \partial_2 v_2 & \partial_3 v_2 \\ \partial_1 v_3 & \partial_2 v_3 & \partial_3 v_3 \end{pmatrix}, \tag{9}$$

where $\partial_i v_j$ is the first-order derivative. Note that $\xi(v) = \frac{1}{2}(\nabla v + \nabla v^T)$, the symmetrized tensor gradient $\xi(v)$ in τ can be directly derived as,

$$\xi(v)|_\tau = \begin{pmatrix} \partial_1 v_1 & \frac{\partial_2 v_1 + \partial_1 v_2}{2} & \frac{\partial_3 v_1 + \partial_1 v_3}{2} \\ \frac{\partial_2 v_1 + \partial_1 v_2}{2} & \partial_2 v_2 & \frac{\partial_3 v_2 + \partial_2 v_3}{2} \\ \frac{\partial_3 v_1 + \partial_1 v_3}{2} & \frac{\partial_3 v_2 + \partial_2 v_3}{2} & \partial_3 v_3 \end{pmatrix}. \tag{10}$$

In the following, we discretize the symmetrized tensor gradient operator $\xi(\cdot)$, which is the core contribution of this paper. As mentioned in Section 2, in smooth transition regions, v has minimums whose values are close to ∇u. Intuitively, this means,

$$v \approx \nabla u \to \nabla v \approx \nabla^2 u \to \partial_i v_i \approx \partial_i\partial_i u, \partial_i v_j \approx \partial_i\partial_j u, \tag{11}$$

where $i, j = 1,2,3$, ignoring the order of i and j. As we can see from (9) and (10), we need two discretization forms of first-order derivatives w.r.t. v (one for $\partial_i v_i$ and the other for $\partial_i v_j$). From (11), we can easily see that these two discretizations also intuitively determine the second-order derivatives w.r.t. u.

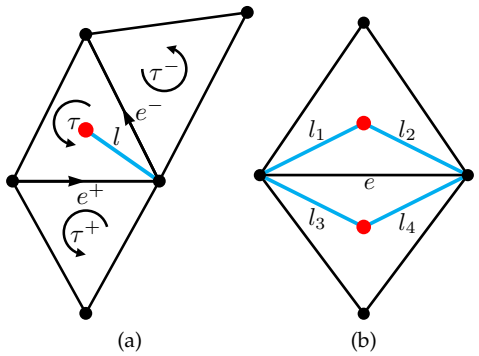


Fig. 1: (a) Illustration for the definition of the 1-form operator $\overline{\mathcal{D}}_\mathcal{E}$v. $[v]$ is the jump of v over line l plotted in cyan in triangle τ with its barycenter plotted in red. (b) Illustration for the definition of the adjoint operator $\overline{\mathcal{D}}^\star_\mathcal{E}\overline{w}$. $B_1(e)$ is a set containing four lines associated with edge e.

Let l be the line segment connecting the barycenter and a vertex of τ. Given $v \in V$, with the Neumann boundary condition, we define the 1-form jump of v over l as

$$[v]_l = \begin{cases} v_{e^+}\mathrm{sgn}(e^+,\tau) + v_{e^-}\mathrm{sgn}(e^-,\tau), & e^+ \text{ and } e^- \not\subset \partial\mathcal{M}, \\ 0, & e^+ \text{ or } e^- \subset \partial\mathcal{M}, \end{cases} \tag{12}$$

where e^+ and e^- are two edges sharing a common vertex of l. e^+ enters the common vertex in counterclockwise direction, whereas e^- leaves the common vertex in counterclockwise direction. The two triangles sharing edges e^+ and e^- are denoted as τ^+ and τ^- respectively. All the aforementioned descriptions are illustrated in Fig. 1a. Then, the discrete 1-form operator $\overline{\mathcal{D}}_\mathcal{E}$ is defined as

$$\overline{\mathcal{D}}_\mathcal{E} : V \to \overline{W}, \quad (\overline{\mathcal{D}}_\mathcal{E} v)|_l = [v]_l, \ \forall l, \text{ for } v \in V, \tag{13}$$

where $\overline{W} = \mathbb{R}^{3\times \mathrm{T}}$. The $\overline{W}$ space is equipped with the following inner product and norm:

$$(\overline{w}^1, \overline{w}^2)_{\overline{W}} = \sum_l \overline{w}^1|_l \overline{w}^2|_l \mathrm{len}(l), \quad \|\overline{w}\|_{\overline{W}} = \sqrt{(\overline{w},\overline{w})_{\overline{W}}}, \tag{14}$$

where $\overline{w}^1, \overline{w}^2, \overline{w} \in \overline{W}$, and $\mathrm{len}(l)$ is the length of line l. The adjoint operator of $\overline{\mathcal{D}}_\mathcal{E}$, that is $\overline{\mathcal{D}}^\star_\mathcal{E} : \overline{W} \to V$, can be derived using the inner products in V and $\overline{W}$. For $\overline{w} \in \overline{W}$, $\overline{\mathcal{D}}^\star_\mathcal{E}$ has the following form

$$(\overline{\mathcal{D}}^\star_\mathcal{E}\overline{w})|_e = -\frac{1}{\mathrm{len}(e)} \sum_{l \in B_1(e)} \overline{w}_l \mathrm{sgn}(e, \tau_l)\mathrm{len}(l), \quad \forall e, \tag{15}$$

where $B_1(e)$ is the set of lines associated with the edge e (see Fig. 1b) and τ_l is the triangle containing the line l. Details for the derivation of $\overline{\mathcal{D}}^\star_\mathcal{E}$ can be found in lemma 1 in Part 1 of the supplementary material.

Remark 1. We give an intuitive interpretation of the discrete 1-form operator $\overline{\mathcal{D}}_\mathcal{E}$. From (12) and (13), we can see that the 1-form operator w.r.t. v depicts the variation of v over the two adjacent edges. Moreover, this operator can also be seen as an analogue of the second-order operator

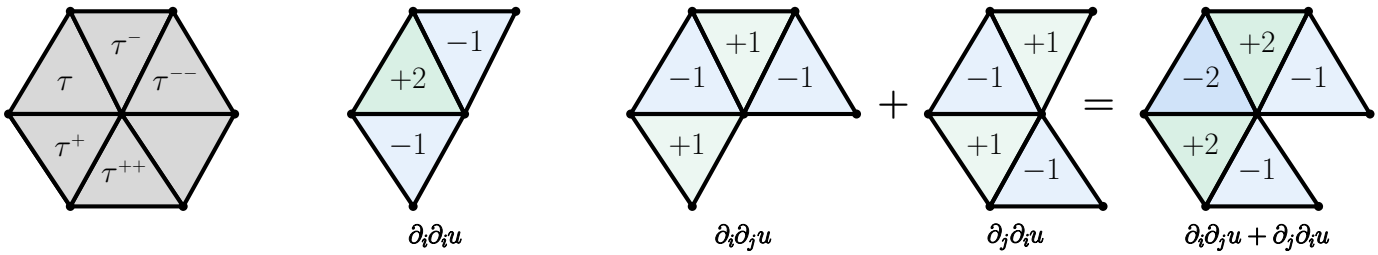


Fig. 2: Illustration of the second-order directional derivatives for a triangle (τ).

w.r.t. u (in the same direction), which depicts

$$\begin{aligned}(\overline{\mathcal{D}}_{\mathcal{E}}v)|_l =& v_{e^+}\mathrm{sgn}(e^+,\tau) + v_{e^-}\mathrm{sgn}(e^-,\tau)\\ \approx&(u_\tau \mathrm{sgn}(e^+,\tau) + u_{\tau^+}\mathrm{sgn}(e^+,\tau^+))\mathrm{sgn}(e^+,\tau)+\\ &(u_\tau \mathrm{sgn}(e^-,\tau) + u_{\tau^-}\mathrm{sgn}(e^-,\tau^-))\mathrm{sgn}(e^-,\tau)\\ =&\big(u_\tau - u_{\tau^+}\big) + \big(u_\tau - u_{\tau^-}\big)\\ =&2u_\tau - u_{\tau^+} - u_{\tau^-}.\end{aligned} \tag{16}$$

In summary, the 1-form operator $\overline{\mathcal{D}}_{\mathcal{E}}v$ can depict the first-order variations of v, which can be seen as an approximation of the first-order derivatives $\partial_i v_i$. Besides, this operator also can describe the second-order variations of u, which can be seen as an approximation of the second-order directional derivatives $\partial_i\partial_i u$, whose discretization on meshes is demonstrated in Fig. 2.

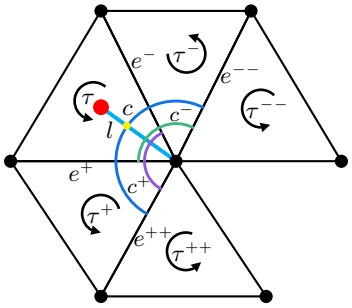


Fig. 3: Illustration of the definition of 2-form operator $\widetilde{\mathcal{D}}_{\mathcal{E}}v$. $[[v]]$ is the 2-form jump over curve c plotted in blue, which passes through four edges $(e^{--}, e^-, e^+, e^{++})$ and attaches itself to line l. Auxiliary curve c^- passing through (e^{--}, e^+) is plotted in green, and curve c^+ passing through (e^-, e^{++}) is plotted in purple.

Let c be a curve passing through four edges $(e^{--}, e^-, e^+, e^{++})$ and associates itself to the line l of τ. For $v \in V$, we define the 2-form jump of v over c as

$$\begin{aligned}[[v]]_c =&[[v]]_{c^-} + [[v]]_{c^+}\\ =&\Big(v_{e^{--}}\mathrm{sgn}(e^{--},\tau^-) + v_{e^+}\mathrm{sgn}(e^+,\tau^+)\Big)+\\ &\Big(v_{e^-}\mathrm{sgn}(e^-,\tau^-) + v_{e^{++}}\mathrm{sgn}(e^{++},\tau^+)\Big).\end{aligned} \tag{17}$$

With Neumann boundary condition, if any of e^{--}, e^-, e^+, e^{++} is on the boundary $\partial\mathcal{M}$, we directly set $[[v]]_c = 0$. The triangle that shares edges e^{--} with τ^- is denoted as τ^{--}, while τ^{++} denotes the triangle that shares edge e^+ with τ^+. The two auxiliary curves passing through (e^{--}, e^+) and (e^-, e^{++}) are denoted as c^- and c^+, respectively. We demonstrate all aforementioned descriptions in Fig. 3. Then, we define the 2-form operator $\widetilde{\mathcal{D}}_{\mathcal{E}}$ as

$$\widetilde{\mathcal{D}}_{\mathcal{E}} : V \to \widetilde{W},\ \ (\widetilde{\mathcal{D}}_{\mathcal{E}}v)|_c = [[v]]_c,\ \forall c,\ \text{for } v \in V, \tag{18}$$

where $\widetilde{W} = \mathbb{R}^{3\times \mathrm{T}}$. The $\widetilde{W}$ space is equipped with the following inner product and norm:

$$(\widetilde{w}^1, \widetilde{w}^2)_{\widetilde{W}} = \sum_c \widetilde{w}^1|_c \widetilde{w}^2|_c \mathrm{len}(c),\ \ \|\widetilde{w}\|_{\widetilde{W}} = \sqrt{(\widetilde{w},\widetilde{w})_{\widetilde{W}}}, \tag{19}$$

where $\widetilde{w}^1, \widetilde{w}^2, \widetilde{w} \in \widetilde{W}$, and $\mathrm{len}(c) = \frac{1}{4}\big(\mathrm{len}(l^-) + 2\mathrm{len}(l) + \mathrm{len}(l^+)\big)$ is an approximation of the length of c. l^+ is the line segment contained in triangle τ^+ and shares the vertex with l, while l^- is the segment contained in τ^-.

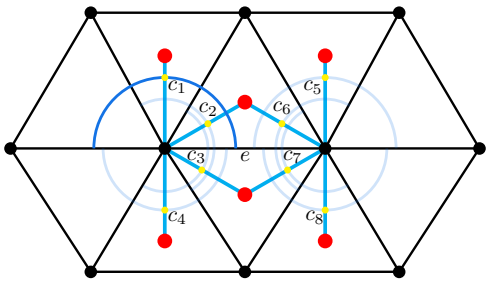


Fig. 4: Illustration of the definition of adjoint operator $\widetilde{\mathcal{D}}^\star_{\mathcal{E}}\widetilde{w}$. $B_2(e)$ is the set of curves associated with edge e, which refers to eight curves. The attached lines of curves in $B_2(e)$ are also shown.

Similarly, the adjoint operator of $\widetilde{\mathcal{D}}_{\mathcal{E}}$, that is $\widetilde{\mathcal{D}}^\star_{\mathcal{E}} : \widetilde{W} \to V$, is given by

$$(\widetilde{\mathcal{D}}^\star_{\mathcal{E}}\widetilde{w})|_e = -\frac{1}{\mathrm{len}(e)}\sum_{c\in B_2(e)} \widetilde{w}_c \mathrm{sgn}(e,\tau_c)\mathrm{len}(c),\ \ \forall e, \tag{20}$$

where $B_2(e)$ is the set of curves associated with edge e (see Fig. 4), and τ_c is the triangle satisfying conditions $e \prec \tau_c$ and $\tau_c \in \{\tau^+, \tau^-\}$. The derivation of $\widetilde{\mathcal{D}}^\star_{\mathcal{E}}$ can be found in lemma 2 in Part 1 of the supplementary material.

Remark 2. We give an intuitive interpretation of the 2-form operator $\widetilde{\mathcal{D}}_{\mathcal{E}}$. From (17) and (18), we can see that the 2-form operator w.r.t. v describes the sum of variations of v across edges (e^{--}, e^+) and across edges (e^-, e^{++}). This operator can also be seen as an analogue of the second-order operator w.r.t. u (in different directions), which can

be expressed as

$$\begin{aligned}(\widetilde{\mathcal{D}}_{\mathcal{E}}v)|_c =&\Big(v_{e^{--}}\mathrm{sgn}(e^{--},\tau^-)+v_{e^{+}}\mathrm{sgn}(e^{+},\tau^+)\Big)+\\ &\Big(v_{e^{-}}\mathrm{sgn}(e^{-},\tau^-)+v_{e^{++}}\mathrm{sgn}(e^{++},\tau^+)\Big)\\ \approx&(u_{\tau+}+u_{\tau-}-u_{\tau}-u_{\tau--})+\\ &(u_{\tau+}+u_{\tau-}-u_{\tau}-u_{\tau++}).\end{aligned} \tag{21}$$

Intuitively, (21) can be used to depict both first-order derivatives $\partial_i v_j + \partial_j v_i$ and second-order directional derivatives $\partial_i\partial_j u + \partial_j\partial_i u$. We illustrate the discretization of second-order directional derivatives $\partial_i\partial_j u + \partial_j\partial_i u$ in Fig. 2.

Note that with the 1- and 2-form operators (13) and (18), the discrete symmetrized gradient operator $\xi : V \to W$ can be directly approximated. Space $W = \mathbb{R}^{6\times \mathrm{T}}$ is a composition of the spaces $\overline{W}$ and $\widetilde{W}$, and it is equipped with the following inner product and norm:

$$\begin{aligned}(w^1,w^2)_W &= (\overline{w}^1,\overline{w}^2)_{\overline{W}} + (\widetilde{w}^1,\widetilde{w}^2)_{\widetilde{W}},\\ \|w\|_W &= \|\overline{w}\|_{\overline{W}} + \|\widetilde{w}\|_{\widetilde{W}},\end{aligned}$$

with $w^1, w^2, w \in W$, $\overline{w}^1, \overline{w}^2, \overline{w} \in \overline{W}$, and $\widetilde{w}^1, \widetilde{w}^2, \widetilde{w} \in \widetilde{W}$. Then, the second-order term of TGV can be expressed as

$$\begin{aligned}\|\xi(v)\|_W &= \sum_{\tau}\Big(\sum_i \|\partial_i v_i\| + \sum_{i,j}\|\partial_i v_j + \partial_j v_i\|\Big)\\ &= \|\overline{\mathcal{D}}_{\mathcal{E}}v\|_{\overline{W}} + \|\widetilde{\mathcal{D}}_{\mathcal{E}}v\|_{\widetilde{W}},\end{aligned} \tag{22}$$

where $i, j = 1, 2, 3$ and $i \neq j$. Given $u \in U$, with the above definition, we now formulate the discretized TGV as

$$\mathrm{TGV}(u) = \min_{v\in V}\Big\{\alpha_1\|\mathcal{D}_{\mathcal{M}}u - v\|_V + \alpha_0\|\xi(v)\|_W\Big\}, \tag{23}$$

which defines the TGV semi-norm over meshes.

We can extend the TGV semi-norm (23) to the vectorial case. To consider vectorial data, three vectorial spaces $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are defined as

$$\mathbf{U} = \underbrace{U\times\cdots\times U}_{\mathfrak{N}}, \mathbf{V} = \underbrace{V\times\cdots\times V}_{\mathfrak{N}}, \mathbf{W} = \underbrace{W\times\cdots\times W}_{\mathfrak{N}},$$

for $\mathfrak{N}$-channel data. The inner products and norms in $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are defined as follows:

$$(\mathbf{u}^1,\mathbf{u}^2)_{\mathbf{U}} = \sum_{1\le i\le\mathfrak{N}}(u_i^1,u_i^2)_U, \quad \|\mathbf{u}\|_{\mathbf{U}} = \sqrt{(\mathbf{u},\mathbf{u})_{\mathbf{U}}},$$

$$(\mathbf{v}^1,\mathbf{v}^2)_{\mathbf{V}} = \sum_{1\le i\le\mathfrak{N}}(v_i^1,v_i^2)_V, \quad \|\mathbf{v}\|_{\mathbf{V}} = \sqrt{(\mathbf{v},\mathbf{v})_{\mathbf{V}}},$$

$$(\mathbf{w}^1,\mathbf{w}^2)_{\mathbf{W}} = \sum_{1\le i\le\mathfrak{N}}(w_i^1,w_i^2)_W, \quad \|\mathbf{w}\|_{\mathbf{W}} = \sqrt{(\mathbf{w},\mathbf{w})_{\mathbf{W}}},$$

with $\mathbf{u}^1,\mathbf{u}^2,\mathbf{u} \in \mathbf{U}$, $\mathbf{v}^1,\mathbf{v}^2,\mathbf{v} \in \mathbf{V}$, and $\mathbf{w}^1,\mathbf{w}^2,\mathbf{w} \in \mathbf{W}$. Thus, all the aforementioned discrete operators can be evaluated channel by channel, and the vectorial TGV semi-norm is then defined as

$$\mathbf{TGV}(\mathbf{u}) = \min_{\mathbf{v}\in\mathbf{V}}\Big\{\alpha_1\|\mathcal{D}_{\mathcal{M}}\mathbf{u} - \mathbf{v}\|_{\mathbf{V}} + \alpha_0\|\xi(\mathbf{v})\|_{\mathbf{W}}\Big\}. \tag{24}$$

4.3 Differences between TV, HO and TGV

There are some existing works highly related to our discretized TGV. It is necessary to discuss the differences between our discretized TGV, total variation (TV) in [8], and high-order variation (HO) in [19]. Given a signal $u \in V$, Zhang et al. [8] defined the discretized TV as

$$\mathrm{TV}(u) = \|\mathcal{D}_{\mathcal{M}}u\|_V, \tag{25}$$

which describes the first-order variations over mesh edges. TV (25) works exceptionally well in preserving sharp features, but produces staircase artifacts in smooth regions.

To overcome the staircase artifacts of the TV regularizer, Liu et al. [19] introduced a second-order difference operator to discretize second-order derivatives, and propose second-order variation as follows:

$$\mathrm{HO}(u) = \sum_l \|2u_\tau - u_{\tau^+} - u_{\tau^-}\|\mathrm{len}(l). \tag{26}$$

HO (26) recovers smooth regions well, but blurs sharp features in case of large noise.

Next, we discuss the differences between the second-order term (22) of TGV and HO (26). From (16), we can see $\mathrm{HO}(u) \approx \|\overline{\mathcal{D}}_{\mathcal{E}}v\|_{\overline{W}}$. Therefore, HO (26) can be seen as an analogue of the second-order derivative in the same direction ($\partial_i\partial_i u$). In other words, the HO regularizer minimizes only the second-order variations in the same direction. In contrast, the minimization of the second-order term of TGV attempts to simultaneously minimize the second-order variations in the same direction ($\partial_i\partial_i u$) as well as those in different directions ($\partial_i\partial_j u + \partial_j\partial_i u$).

As mentioned earlier, TV is more effective than HO in preserving sharp features, while HO handles smooth regions better than TV. Until now, it has been challenging to have one regularizer simultaneously preserve sharp features in some parts of the mesh and recover smooth regions in some other parts. To address this problem, we propose the discretized TGV, which automatically balances the first- and second-order terms via the auxiliary variable v. In consequence, it combines the advantages from both TV and HO, and manages to overcome their weakness. See Section 6 for more detailed comparisons.

5 Mesh Denoising using Vectorial TGV

In this section, we first propose a vectorial TGV based normal filter, and then design an algorithm to effectively solve the optimization problem. After that, we reconstruct vertex positions based on the optimized face normals.

5.1 Vectorial TGV based Normal Filter

Given a noisy mesh, we denote its face normal field as $\mathbf{N}^{in}$. In order to remove noise in $\mathbf{N}^{in}$ using vectorial TGV (24), we formulate our normal filter as the following problem,

$$\begin{aligned}\min_{\mathbf{N},\mathbf{v}}\Big\{&\frac{\beta}{2}\|\mathbf{N}-\mathbf{N}^{in}\|_{\mathbf{U}}^2 + \alpha_1\sum_e w_e\|(\mathcal{D}_{\mathcal{M}}\mathbf{N}-\mathbf{v})|_e\|\mathrm{len}(e)\\ &+\alpha_0\|\xi(\mathbf{v})\|_{\mathbf{W}}\Big\}, \ \text{s.t. } \|\mathbf{N}_\tau\| = 1, \forall\tau,\end{aligned} \tag{27}$$

where $\|\xi(\mathbf{v})\|_{\mathbf{W}} = \|\overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v}\|_{\overline{\mathbf{W}}} + \|\widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v}\|_{\widetilde{\mathbf{W}}}$. Note that $\mathbf{N} \in \mathbf{U}$ and $\mathbf{U}$ denotes 3-channel U. Weight w_e is given by

$$w_e = \exp\Big(-\frac{\|\mathbf{N}_{e,1}-\mathbf{N}_{e,2}\|^2}{2{\sigma_e}^2}\Big), \tag{28}$$

where $\mathbf{N}_{e,1}$ and $\mathbf{N}_{e,2}$ are normals of the triangles sharing edge e, and σ_e is a user-specified parameter. w_e is expected to be large when $\|\mathbf{N}_{e,1} - \mathbf{N}_{e,2}\|$, the modulus of the first-order normal difference defined on e, is small and vice versa. Thus, it results in large weights for smooth regions, and small weights for sharp features, and therefore allows the proposed filter (27) to smooth non-features regions while preserving sharp features.

The vectorial TGV (24) (applied to the face normal field) can produce satisfactory denoising results in most cases. However, it may oversmooth sharp features for some meshes with large noise. Thus, we propose the dynamic weighting (w_e) in our vectorial TGV based normal filter (27). These weights are updated in each iteration, which enhance the sparsity of the original vectorial TGV (24) for improved sharp feature reconstruction. Essentially, these dynamically adjusted weights penalize smooth regions more than sharp features, which allows the lower-than-ℓ_1-sparsity effect to be achieved [47].

5.2 Numerical Optimization

Due to the vectorial ℓ_1 semi-norm involved, problem (27) has a non-differentiable objective which makes it difficult to solve. Here, we use variable-splitting and augmented Lagrange method (ALM) to solve (27), which has achieved great success in solving ℓ_1 related problems [48], [49].

By introducing new variables $\mathbf{P}$, $\overline{\mathbf{Q}}$, and $\widetilde{\mathbf{Q}}$, we reformulate (27) as a constrained optimization problem,

$$\begin{aligned}\min_{\mathbf{N},\mathbf{v},\mathbf{P},\overline{\mathbf{Q}},\widetilde{\mathbf{Q}}} \Big\{ &\frac{\beta}{2}\left\|\mathbf{N}-\mathbf{N}^{in}\right\|_{\mathbf{U}}^2 + \alpha_1 \sum_e w_e \|\mathbf{P}_e\| \mathrm{len}(e) \\ &+ \alpha_0 \|\overline{\mathbf{Q}}\|_{\overline{\mathbf{W}}} + \alpha_0 \|\widetilde{\mathbf{Q}}\|_{\widetilde{\mathbf{W}}} + \Psi(\mathbf{N}) \Big\}, \\ \text{s.t. } &\mathbf{P} = \mathcal{D}_{\mathcal{M}}\mathbf{N} - \mathbf{v},\ \overline{\mathbf{Q}} = \overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v},\ \widetilde{\mathbf{Q}} = \widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v},\end{aligned}$$

where

$$\Psi(\mathbf{N}) = \begin{cases} 0, & \text{if } \|\mathbf{N}_\tau\| = 1,\ \forall \tau, \\ +\infty, & \text{otherwise.} \end{cases}$$

To solve the above constrained optimization problem, we introduce the augmented Lagrangian function as follows,

$$\begin{aligned}&\mathcal{L}(\mathbf{N},\mathbf{v},\mathbf{P},\overline{\mathbf{Q}},\widetilde{\mathbf{Q}};\lambda_{\mathbf{P}},\lambda_{\overline{\mathbf{Q}}},\lambda_{\widetilde{\mathbf{Q}}}) = \frac{\beta}{2}\left\|\mathbf{N}-\mathbf{N}^{in}\right\|_{\mathbf{U}}^2 \\ &+ \alpha_1 \sum_e w_e \|\mathbf{P}_e\| \mathrm{len}(e) + \alpha_0 \|\overline{\mathbf{Q}}\|_{\overline{\mathbf{W}}} + \alpha_0 \|\widetilde{\mathbf{Q}}\|_{\widetilde{\mathbf{W}}} + \Psi(\mathbf{N}) \\ &+ (\lambda_{\mathbf{P}}, \mathbf{P} - (\mathcal{D}_{\mathcal{M}}\mathbf{N} - \mathbf{v}))_{\mathbf{V}} + \frac{r_1}{2}\left\|\mathbf{P} - (\mathcal{D}_{\mathcal{M}}\mathbf{N} - \mathbf{v})\right\|_{\mathbf{V}}^2 \\ &+ (\lambda_{\overline{\mathbf{Q}}}, \overline{\mathbf{Q}} - \overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v})_{\overline{\mathbf{W}}} + \frac{r_0}{2}\left\|\overline{\mathbf{Q}} - \overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v}\right\|_{\overline{\mathbf{W}}}^2 \\ &+ (\lambda_{\widetilde{\mathbf{Q}}}, \widetilde{\mathbf{Q}} - \widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v})_{\widetilde{\mathbf{W}}} + \frac{r_0}{2}\left\|\widetilde{\mathbf{Q}} - \widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v}\right\|_{\widetilde{\mathbf{W}}}^2,\end{aligned}$$

where $\lambda_{\mathbf{P}}$, $\lambda_{\overline{\mathbf{Q}}}$, and $\lambda_{\widetilde{\mathbf{Q}}}$ are Lagrange multipliers, r_1 and r_0 are positive penalty weights.

Then we apply the variable-splitting technique and iteratively update the different set of variables in alternation with the following five subproblems:

- The $\mathbf{N}$-subproblem:

$$\begin{aligned}\min_{\mathbf{N}} &\frac{\beta}{2}\|\mathbf{N}-\mathbf{N}^{in}\|_{\mathbf{U}}^2 + \Psi(\mathbf{N}) \\ &+ \frac{r_1}{2}\|\mathcal{D}_{\mathcal{M}}\mathbf{N} - \mathbf{v} - (\mathbf{P} + \frac{\lambda_{\mathbf{P}}}{r_1})\|_{\mathbf{V}}^2;\end{aligned} \tag{29}$$

- The $\mathbf{v}$-subproblem:

$$\begin{aligned}\min_{\mathbf{v}} &\frac{r_0}{2}\|\overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v} - (\overline{\mathbf{Q}} + \frac{\lambda_{\overline{\mathbf{Q}}}}{r_0})\|_{\overline{\mathbf{W}}}^2 + \frac{r_0}{2}\|\widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v} - (\widetilde{\mathbf{Q}} + \frac{\lambda_{\widetilde{\mathbf{Q}}}}{r_0})\|_{\widetilde{\mathbf{W}}}^2 \\ &+ \frac{r_1}{2}\|\mathcal{D}_{\mathcal{M}}\mathbf{N} - \mathbf{v} - (\mathbf{P} + \frac{\lambda_{\mathbf{P}}}{r_1})\|_{\mathbf{V}}^2;\end{aligned} \tag{30}$$

- The $\mathbf{P}$-subproblem:

$$\min_{\mathbf{P}} \alpha_1 \sum_e w_e \|\mathbf{P}_e\| \mathrm{len}(e) + \frac{r_1}{2}\|\mathbf{P} - (\mathcal{D}_{\mathcal{M}}\mathbf{N} - \mathbf{v} - \frac{\lambda_{\mathbf{P}}}{r_1})\|_{\mathbf{V}}^2; \tag{31}$$

- The $\overline{\mathbf{Q}}$-subproblem:

$$\min_{\overline{\mathbf{Q}}} \alpha_0 \|\overline{\mathbf{Q}}\|_{\overline{\mathbf{W}}} + \frac{r_0}{2}\|\overline{\mathbf{Q}} - (\overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v} - \frac{\lambda_{\overline{\mathbf{Q}}}}{r_0})\|_{\overline{\mathbf{W}}}^2; \tag{32}$$

- The $\widetilde{\mathbf{Q}}$-subproblem:

$$\min_{\widetilde{\mathbf{Q}}} \alpha_0 \left\|\widetilde{\mathbf{Q}}\right\|_{\widetilde{\mathbf{W}}} + \frac{r_0}{2}\|\widetilde{\mathbf{Q}} - (\widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v} - \frac{\lambda_{\widetilde{\mathbf{Q}}}}{r_0})\|_{\widetilde{\mathbf{W}}}^2. \tag{33}$$

The $\mathbf{N}$-subproblem (29) is a quadratic optimization problem with the unit normal constraints. Here, we adopt an approximation strategy to solve this problem. We first ignore the unit normal constraints and solve the quadratic program, and then project the minimizer onto the unit sphere. Specifically, we check the first-order optimality condition of (29), and obtain the following Euler-Lagrange equation

$$\beta\mathbf{N} - r_1\mathcal{D}_{\mathcal{M}}^{\star}\mathcal{D}_{\mathcal{M}}\mathbf{N} = \beta\mathbf{N}^{in} - \mathcal{D}_{\mathcal{M}}^{\star}\big(\lambda_{\mathbf{P}} + r_1(\mathbf{P} + \mathbf{v})\big). \tag{34}$$

Plugging in the first-order operator (6) and its adjoint operator (7), we can rewrite the above equation as a sparse and positive semidefinite linear system, which can be solved by efficient sparse linear solvers, such as TAUCS and Intel Math Kernel Library (MKL).

The $\mathbf{v}$-subproblem (30) is also a quadratic program, whose Euler-Lagrange equation is given as

$$\begin{aligned}r_1\mathbf{v} - r_0\overline{\mathcal{D}}_{\mathcal{E}}^{\star}\overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v} - r_0\widetilde{\mathcal{D}}_{\mathcal{E}}^{\star}\widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v} = &-\lambda_{\mathbf{P}} - r_1(\mathbf{P} - \mathcal{D}_{\mathcal{M}}\mathbf{N}) \\ &- \overline{\mathcal{D}}_{\mathcal{E}}^{\star}(\lambda_{\overline{Q}} + r_0\overline{Q}) - \widetilde{\mathcal{D}}_{\mathcal{E}}^{\star}(\lambda_{\widetilde{Q}} + r_0\widetilde{Q}).\end{aligned} \tag{35}$$

Plugging the 1- and 2-form operators (13) and (18) and their adjoint operators (15) and (20), we can rewrite the above equation as a sparse linear system, which again can be solved by linear solvers.

The $\mathbf{P}$-subproblem (31) is solved directly as it can be spatially decoupled, where the minimization problem for each edge is solved separately. For each $\mathbf{P}_e$, we have the following simplified problem:

$$\min_{\mathbf{P}_e} \alpha_1 w_e \|\mathbf{P}_e\| + \frac{r_1}{2}\|\mathbf{P}_e - \big((\mathcal{D}_{\mathcal{M}}\mathbf{N})|_e - \mathbf{v}_e - \frac{\lambda_{\mathbf{P}_e}}{r_1}\big)\|^2,$$

which has a closed form solution:

$$\mathbf{P}_e = \mathrm{Shrink}(\alpha_1 w_e,\ r_1,\ (\mathcal{D}_{\mathcal{M}}\mathbf{N})|_e - \mathbf{v}_e - \frac{\lambda_{\mathbf{P}_e}}{r_1}), \tag{36}$$

with the soft shrinkage operator defined as:

$$\mathrm{Shrink}(x, y, z) = \max(0, 1 - \frac{x}{y\|z\|})z.$$

The $\overline{\mathbf{Q}}$-subproblem (32) is solved for each line indepen-

dently. For each $\overline{\mathbf{Q}}_l$, we solve the following problem:

$$\min_{\overline{\mathbf{Q}}_l} \alpha_0 \|\overline{\mathbf{Q}}_l\| + \frac{r_0}{2}\|\overline{\mathbf{Q}}_l - ((\overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v})|_l - \frac{\lambda_{\overline{\mathbf{Q}}_l}}{r_0})\|^2,$$

whose closed form solution is:

$$\overline{\mathbf{Q}}_l = \text{Shrink}(\alpha_0,\ r_0,\ (\overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v})|_l - \frac{\lambda_{\overline{\mathbf{Q}}_l}}{r_0}). \tag{37}$$

Similarly, the $\widetilde{\mathbf{Q}}$-subproblem (33) can be separated into the following problem for each curve $\widetilde{\mathbf{Q}}_c$:

$$\min_{\widetilde{\mathbf{Q}}_c} \alpha_0 \|\widetilde{\mathbf{Q}}_c\| + \frac{r_0}{2}\|\widetilde{\mathbf{Q}}_c - ((\widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v})|_c - \frac{\lambda_{\widetilde{\mathbf{Q}}_c}}{r_0})\|^2,$$

which has a closed form solution:

$$\widetilde{\mathbf{Q}}_c = \text{Shrink}(\alpha_0,\ r_0,\ (\widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v})|_c - \frac{\lambda_{\widetilde{\mathbf{Q}}_c}}{r_0}). \tag{38}$$

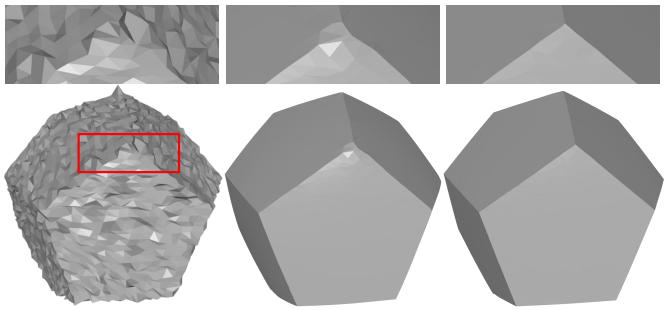


Fig. 5: Denoising results of Dodecahedron (corrupted with $\sigma = 0.3\bar{l}_e$, where σ is standard deviation of Gaussian noise and $\bar{l}_e$ is mean edge length). From left to right: noisy mesh, denoising results produced by vectorial TGV normal filter (27) without and with dynamic weights, respectively.

In summary, the full procedure for the TGV normal filter (27) is sketched in Algorithm 1. Based on variable-splitting and ALM, this algorithm iteratively solves the above five subproblems and updates the Lagrange multipliers. The algorithm terminates when one of the stopping criteria is met. As mentioned in section 5.1, the dynamic weights w_e play a key role in recovering sharp features. As Fig. 5 shows, without these dynamic weights, some sharp features are blurred in the denoised result.

5.3 Vertex Updating Scheme

After smoothing the normal field via the proposed TGV-based normal filter, vertex positions should be updated to match the filtered normals. To avoid the triangle orientation ambiguity problem in the traditional vertex updating scheme [4], we reconstruct the mesh using the vertex updating scheme proposed by Zhang et al. [37]. We empirically fix the iteration number as 30 in our experiments, which allows producing satisfactory results. We refer the interested reader to the work [37] for more details.

6 Experiments and Discussions

We test the proposed denoising method on a variety of meshes including CAD, non-CAD, and scanned data. The tested meshes are corrupted by either synthetic or raw noise. The synthetic noise is generated by a zero-mean Gaussian function with mean edge length ($\bar{l}_e$) as the standard deviation (σ). We present visual and numerical comparisons between the proposed method (TGV) and the state-of-the-art, including the total variation filter (TV) [8], the high-order filter (HO) [19], ℓ_0 minimization (L0) [42], the bilateral filter (BF) [5], the non-local low-rank filter (NLLR) [13], and the cascaded filter (CNR) [16]. We implemented TV, HO, L0, and BF according to the literature in C++. For NLLR, we execute the code kindly provided by the authors of [13] to produce the results. For CNR, we directly use the trained neural networks kindly provided by the authors [16] to generate the results. We carefully tune the parameters of each competing methods so that satisfactory results are produced. All the methods are performed on a laptop with an Intel i7 dual core 2.6 GHz processor and 16 GB RAM. All the meshes are rendered in flat-shading to emphasize faceting effect. To promote reproducibility, we release our executable program and data in the GitHub page [1].

Algorithm 1: ALM for TGV normal filtering (27)

Initialization: $\mathbf{N}^{-1} = \mathbf{v}^{-1} = \mathbf{P}^{-1} = \overline{\mathbf{Q}}^{-1} = \widetilde{\mathbf{Q}}^{-1} = 0, \lambda_{\mathbf{P}}^0 = \lambda_{\overline{\mathbf{Q}}}^0 = \lambda_{\widetilde{\mathbf{Q}}}^0 = 0, k = 0;$

repeat

1. fix $(\mathbf{v}^{k-1}, \mathbf{P}^{k-1}, \lambda_{\mathbf{P}}^k)$, solve $\mathbf{N}^k$ by (34); normalize $\mathbf{N}^k$;
2. fix $(\mathbf{N}^k, \mathbf{P}^{k-1}, \overline{\mathbf{Q}}^{k-1}, \widetilde{\mathbf{Q}}^{k-1}, \lambda_{\mathbf{P}}^k, \lambda_{\overline{\mathbf{Q}}}^k, \lambda_{\widetilde{\mathbf{Q}}}^k)$, solve $\mathbf{v}^k$ by (35);
3. fix $(\mathbf{N}^k, \mathbf{v}^k, \lambda_{\mathbf{P}}^k)$, solve $\mathbf{P}^k$ by (36);
4. fix $(\mathbf{v}^k, \lambda_{\overline{\mathbf{Q}}}^k)$, solve $\overline{\mathbf{Q}}^k$ by (37);
5. fix $(\mathbf{v}^k, \lambda_{\widetilde{\mathbf{Q}}}^k)$, solve $\widetilde{\mathbf{Q}}^k$ by (38);
6. update Lagrange multipliers
   $\lambda_{\mathbf{P}}^{k+1} = \lambda_{\mathbf{P}}^k + r_1(\mathbf{P}^k - (\mathcal{D}_{\mathcal{M}}\mathbf{N}^k - \mathbf{v}^k));$
   $\lambda_{\overline{\mathbf{Q}}}^{k+1} = \lambda_{\overline{\mathbf{Q}}}^k + r_0(\overline{\mathbf{Q}}^k - \overline{\mathcal{D}}_{\mathcal{E}}\mathbf{v}^k);$
   $\lambda_{\widetilde{\mathbf{Q}}}^{k+1} = \lambda_{\widetilde{\mathbf{Q}}}^k + r_0(\widetilde{\mathbf{Q}}^k - \widetilde{\mathcal{D}}_{\mathcal{E}}\mathbf{v}^k);$
7. update weights w_e using (28) ;
8. Increment k: $k = k + 1;$

until $\|\mathbf{N}^k - \mathbf{N}^{k-1}\|_{\mathbf{U}}^2 < 1e-10$ or $k \geq 100;$

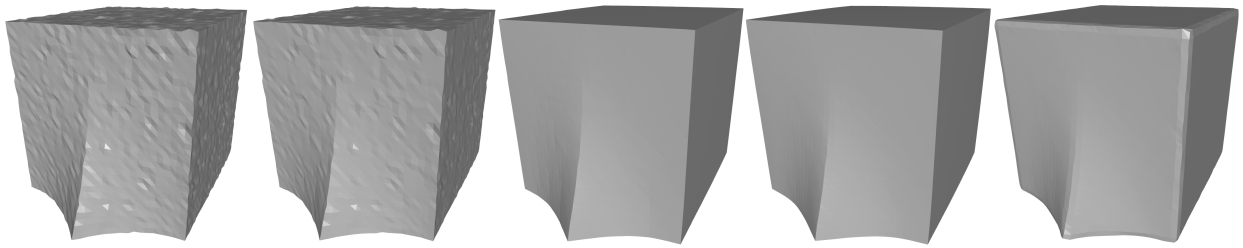
return $\mathbf{N}^k$.

6.1 Parameters Setting

Our TGV filter (27) has three parameters, i.e., α_1, α_0, and β, which balance the first-, second-order, and fidelity terms of (27). When the parameters are chosen properly, on one hand, in smooth regions we have $\mathbf{v} \approx \mathcal{D}_{\mathcal{M}}\mathbf{N}$, which results in the first-order term being close to vanishing. Thus, the minimization of (27) is mainly controlled by the second-order term in these smooth regions; see the corresponding regions in Figs. 8b and 8c. On the other hand, in regions near sharp features, we have $\mathbf{v} \approx 0$, which causes the second-order term to be close to vanish. Thus, the minimization mainly depends on the first-order term in these regions; see the corresponding regions in Figs. 8b and 8c.

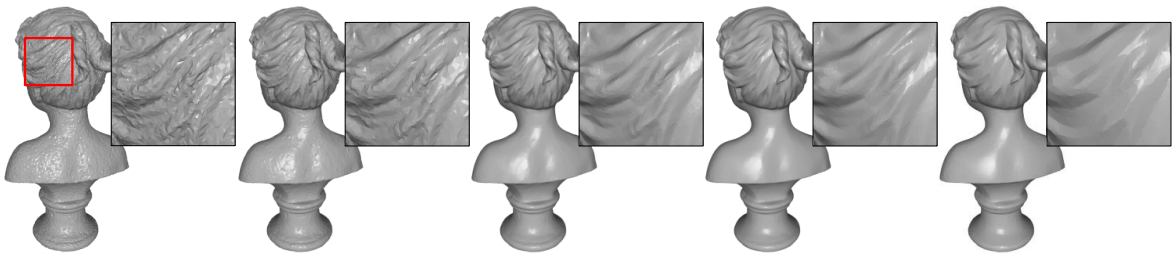
1. https://github.com/LabZhengLiu/MeshTGV

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3088118, IEEE Transactions on Visualization and Computer Graphics



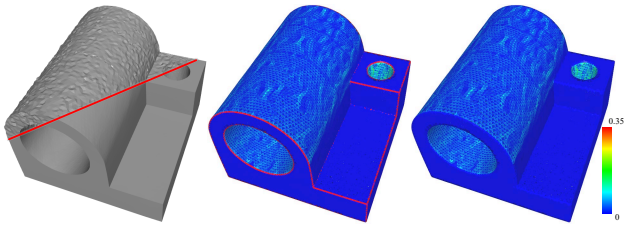
(a) Noisy (b) $\alpha_1 = 0.05$ (c) $\alpha_1 = 0.4$ (d) $\alpha_1 = 0.7$ (e) $\alpha_1 = 2.0$

Fig. 6: Denoising results for varying α_1 with fixed α_0 and β. From left to right: noisy mesh (corrupted with $\sigma = 0.1\bar{l}_e$), and results with increasing α_1.



(a) Noisy (b) $\alpha_0 = 0.01$ (c) $\alpha_0 = 0.03$ (d) $\alpha_0 = 0.05$ (e) $\alpha_0 = 1.0$

Fig. 7: Denoising results for varying α_0 with fixed α_1 and β. From left to right: noisy mesh (corrupted with $\sigma = 0.15\bar{l}_e$), and results with increasing α_0.



(a) Input (Result) (b) $\mathcal{D}_{\mathcal{M}}\mathbf{N}$ (c) $\mathbf{v}$

Fig. 8: (a) Noisy input (Denoising result). (b) $\mathcal{D}_{\mathcal{M}}\mathbf{N}$ of the result visualized using color coding. (c) The visualization of $\mathbf{v}$ of the result.

Parameter α_1 controls the impact of the first-order term in (27). For each noisy mesh, these exist a range ([0.5, 3.0]) for α_1, that leads to promising results. This indicates that our method is insensitive to the perturbation of α_1; see Figs. 6c and 6d. If α_1 is too small, the first-order term gets practically ignored causing $\mathbf{v} \approx 0$ over the whole mesh, which in turn causes the second-order term being close to vanish. Therefore, the TGV regularizer fails, leading to residual noise in the result; see Fig. 6b. If α_1 is too large, in regions near sharp features, the first-order term tends to have $\mathbf{v} \approx \mathcal{D}_{\mathcal{M}}\mathbf{N}$, and the minimization of (27) will be controlled by the second-order term in these regions, which may smooth sharp features; see Fig. 6e.

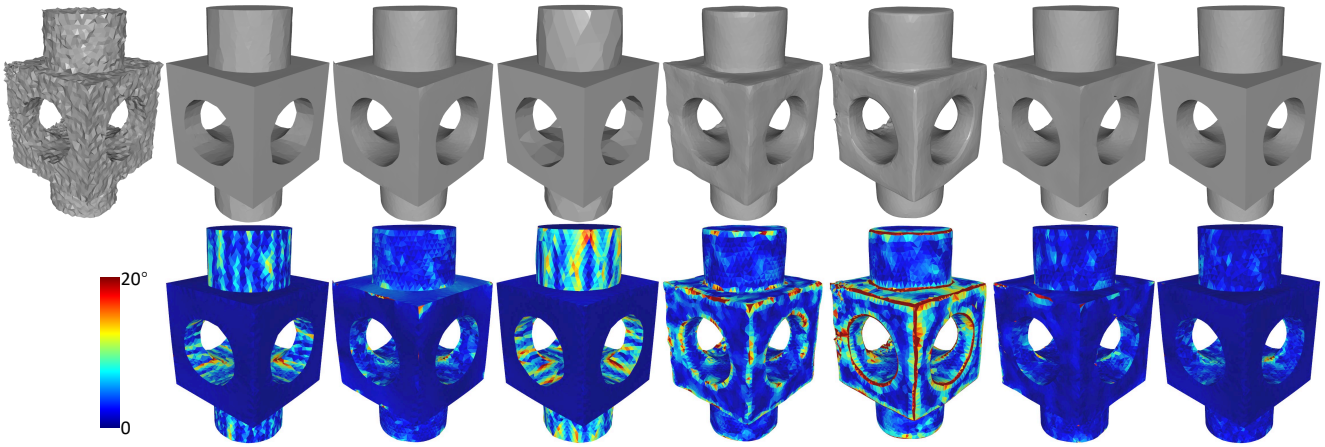
Parameter α_0 influences the effect of the second-order term in (27). Similar to α_1, for each noisy mesh, there exist a range ([0.05, 1]) for α_0 that can produce satisfactory results; see Figs. 7c and 7d. Underweighting the second-order term leads to residual noise in the result; see Fig. 7b. In contrast, overweighting the second-order term will penalize smooth regions as well as fine features and therefore oversmooth the details; see Fig. 7e.

Parameter β controls the degree of denoising procedure. It is empirically fixed as 100 for CAD and scanned surfaces usually, and is fixed as 1000 for non-CAD surfaces.

6.2 Qualitative Performance

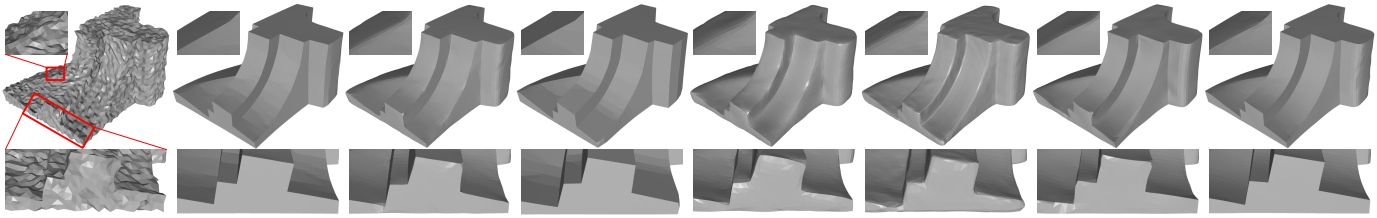
Denoise CAD surfaces. In Fig. 9, we present the denoising results on a CAD surface containing both sharp features and smooth regions. It can be seen that, except for BF and NLLR, all the other testing methods preserve sharp features to some extent. As both geometric features and noise belong to high frequency information, BF and NLLR cannot distinguish them, especially for sharp features, and as a result, some features are treated as noise and get blurred; see Figs. 9e and 9f. CNR, the learning-based method, performs well, however, it induces artifacts near sharp features; see Fig. 9g. We observe that, sparse optimization based methods, including TV, HO, TGV, and L0, preserve sharp features more accurately. However, due to its higher sparsity requirement, L0 flattens some smooth regions and induces false features in smooth regions sometimes, as Fig. 9d shows. In contrast, TGV is free from these artifacts in smooth regions, which makes it an significant improvement over TV; see Figs. 9b and 9h. Compared to HO, TGV recovers sharp features and flat regions more accurately; see Fig. 9c. For each testing method, we also visualize the error map for the normals, where the error is defined as the angular difference between the filtered normals and the ground truth. The normals

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3088118, IEEE Transactions on Visualization and Computer Graphics



(a) Noisy (b) TV (c) HO (d) L0 (e) BF (f) NLLR (g) CNR (h) Ours

Fig. 9: Comparison of denoising results of Block, corrupted with $\sigma = 0.35\bar{l}_e$. The second row visualizes the corresponding error maps, using the angular difference between face normals of denoised meshes and ground truth meshes.



(a) Noisy (b) TV (c) HO (d) L0 (e) BF (f) NLLR (g) CNR (h) Ours

Fig. 10: Comparison of denoising results of Fandisk, corrupted with $\sigma = 0.25\bar{l}_e$.

produced by our method are noticeably closer to the ground truth; see the second row of Fig. 9.

In Fig. 10, we compare the results for a CAD surface including sharp features and shallow edge. Again, BF and NLLR blur sharp features in varying degrees, while L0 flattens smooth regions and produces false features; see Figs. 10e, 10f, and 10d. As TV applies only to the first-order information, it suffers from staircase artifacts in smoothly curved regions; see Fig. 10b. HO recovers smooth regions more accurately than TV. However, as HO only uses high-order information, it bends straight-line edges and blurs shallow features; see the zoomed-in view of Fig. 10c. Hence TV and HO are effective in preserving either sharp features or smooth regions, but not both. In contrast, TGV combines the advantages from both methods and accurately recovers both sharp features and smooth regions; see Fig. 10h. Visual comparisons for this example show the superior performance of TGV in simultaneously preserving features and recovering smooth regions.

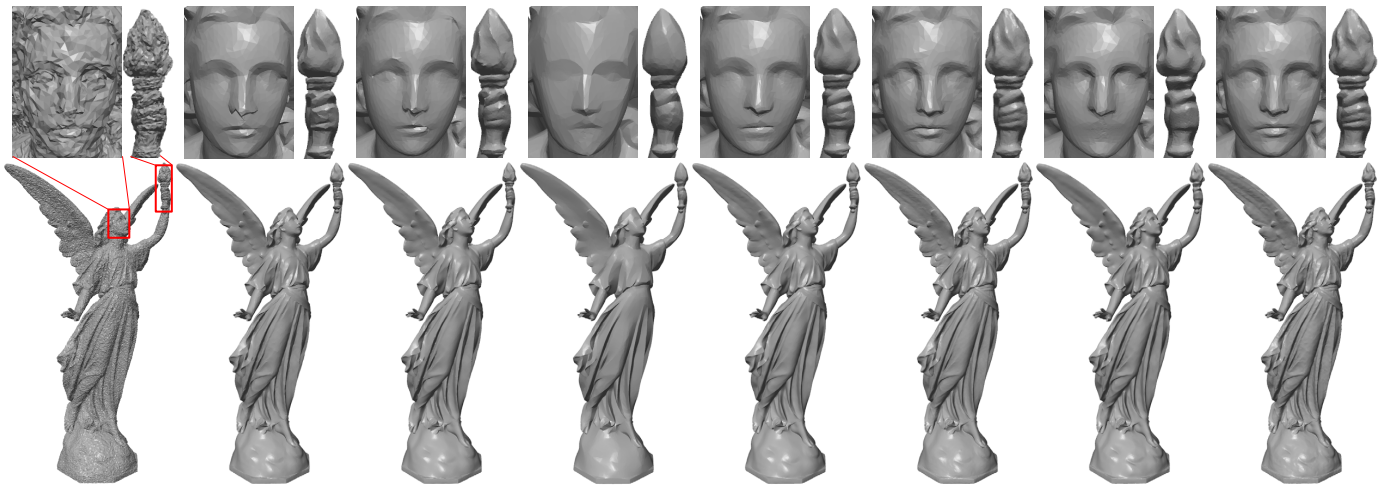
Denoise non-CAD surfaces. Fig. 11 shows a comparison on a non-CAD surface with rich geometric features. As expected, TV over-smoothes some small-scale features, and exhibits slight staircase artifacts. L0 makes this situation even worse by transforming smooth regions into piecewise constant ones while over sharpening medium-scale features; see Fig. 11d. HO and CNR are effective in preserving medium-scale features; see the torch in Figs. 11c and 11g. But they may smooth small-scale features and fine details; see the hand regions in Figs. 11c and 11g. In contrast, NLLR and our method TGV produce visually compelling results. Furthermore, from Table 1, we can see our method achieves higher numerical accuracy than NLLR. Therefore, our method produces appealing results with geometric features recovered better than all competing methods.

Fig. 12 compares the results on a non-CAD surface containing multi-scale features. As expected, comparing to the other methods, NLLR, CNR, and our method TGV recover different levels of features in a better manner. NLLR may retain some extra noise in the result, while CNR slightly blurs small-scale features. In contrast, TGV produces visually the best result with most geometric features well preserved.

Overall, for non-CAD meshes, our method TGV generates satisfactory results with features recovered better, and at the same time it prevents introducing additional artifacts (e.g., staircase artifacts, over-smoothing, over-sharpening effects, extra noise).

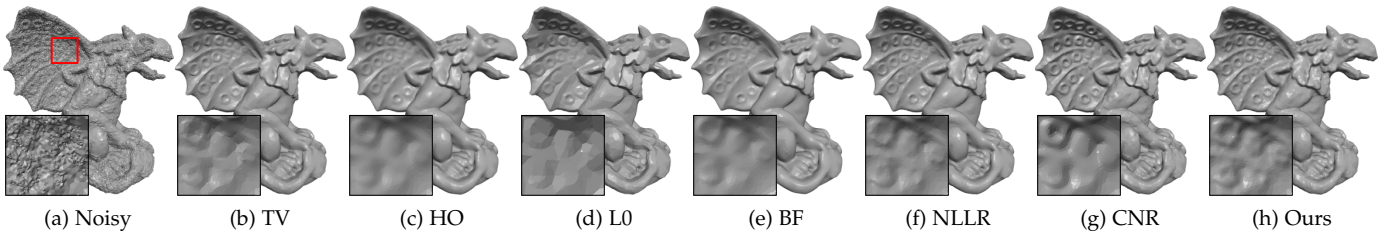
Denoise scanning data. We also compare the different methods on scanned data, where the noise pattern is unknown. Fig. 13 shows the results for data acquired by a laser scanner. First, TV, L0, and CNR over-smooth fine details while sharpen some features, which makes the results look less natural (see the zoomed-in views of Figs. 13b, 13d, and 13g). In contrast, HO and BF blur details to varying degrees. In this example, the results produced by NLLR and our method TGV look more natural and compelling than those produced by the other methods; see Figs. 13f and 13h. Our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3088118, IEEE Transactions on Visualization and Computer Graphics



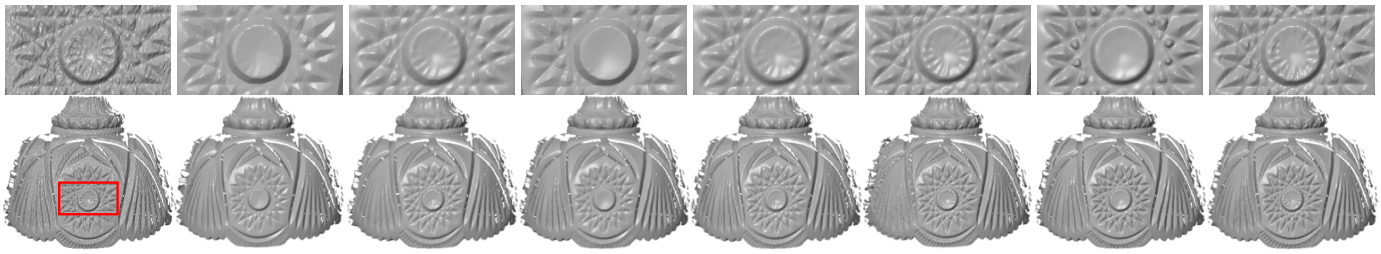
(a) Noisy (b) TV (c) HO (d) L0 (e) BF (f) NLLR (g) CNR (h) Ours

Fig. 11: Comparison of denoising results of Lucy, corrupted with $\sigma = 0.2\bar{l}_e$.



(a) Noisy (b) TV (c) HO (d) L0 (e) BF (f) NLLR (g) CNR (h) Ours

Fig. 12: Comparison of denoising results for Gargoyle, corrupted with $\sigma = 0.25\bar{l}_e$.



(a) Noisy (b) TV (c) HO (d) L0 (e) BF (f) NLLR (g) CNR (h) Ours

Fig. 13: Comparison of denoising results for scanned data acquired by a laser scanner.

result is free of visible artifacts and almost does not lose any features from the underlying surface.

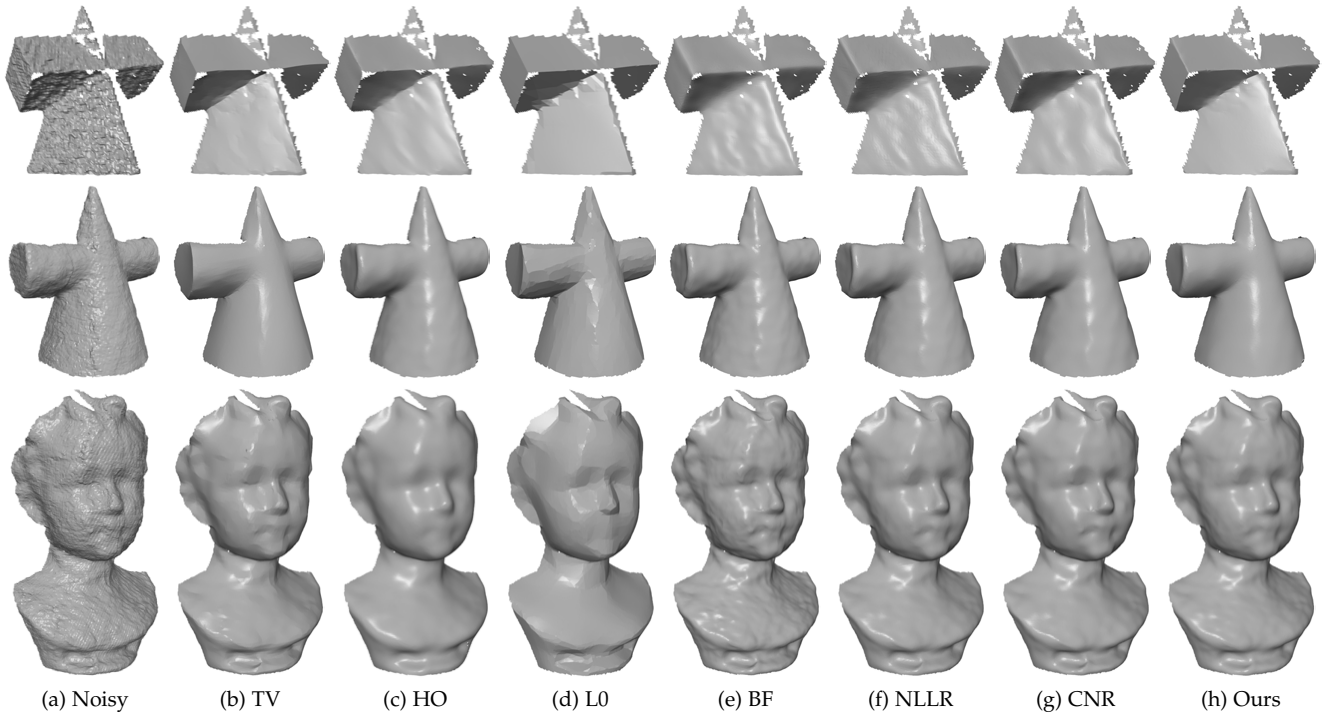
In Fig. 14, we examine the performance of our method on meshes acquired by the Kinect sensor. These scanned meshes are provided by Wang et al. [16]. As can be seen, all the competing methods remove noise effectively, except for BF which cannot distinguish features and noise clearly. TV and L0 produce staircase artifacts in smooth regions and sharpen curved features; see Figs. 14b and 14d. This phenomenon is more severe for L0. Although HO does a good job in smooth regions, it slightly blurs small-scale features. NLLR and CNR yield visually excellent results, although they induce some small bumps in smooth regions. In contrast, our method outperforms the other methods in preserving features and recovering smooth regions, while preventing visible artifacts.

Overall, in all the meshes being tested, our results present visually cleaner geometric features without noticeable artifacts. Hence, our method has succeeded in simultaneously recovering smoothly curved regions and preserving sharp features, even in the presence of heavy noise.

6.3 Quantitative Evaluation

To quantitatively evaluate the quality of the denoising results, we adopt the mean angular difference, abbreviated as θ, as an error metric. This error metric is widely used in recent work [16], [13], [18]. It measures the mean angular difference (θ) of normals between the clean mesh and the denoised result. For fair comparison, we compute θ after the filtering step for each testing method (except L0). Table

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3088118, IEEE Transactions on Visualization and Computer Graphics



(a) Noisy (b) TV (c) HO (d) L0 (e) BF (f) NLLR (g) CNR (h) Ours

Fig. 14: Comparison of denoising results for scanned data acquired by Kinect.

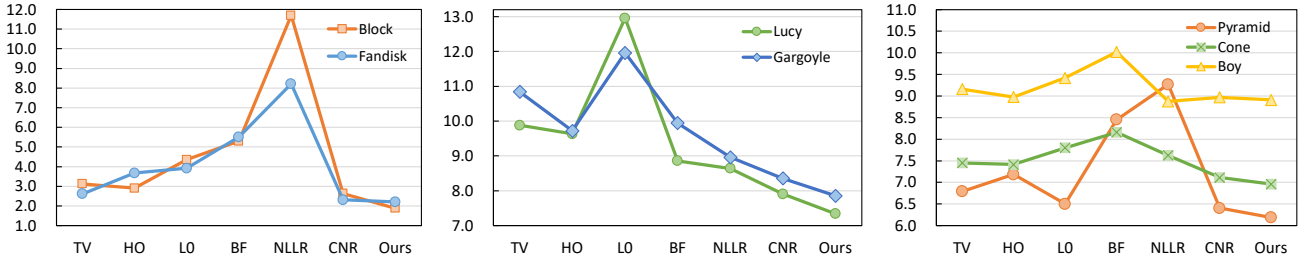


Fig. 15: Error plots of mean angular difference (θ) of the results in Figs. 9, 10, 11, 12, and 14, for all competing methods.

1 lists the error metric θ for all competing methods, and Fig. 15 further shows θ as polyline plots. The mesh sizes of the tested surfaces in Table 1 are listed in Table 2. As we can see from Table 1, our method TGV produces highly competitive results. More specifically, for CAD surfaces, our method outperforms the other methods in comparison in the sense that the θ values are significantly smaller; see the first column of Fig. 15. That is consistent with the visual comparisons in Figs. 9 and 10. For non-CAD surfaces, it is not surprising that our method gives θ values lower than the competing methods, indicating that our results are more faithful to the ground truth. Visually, results from our method and NLLR look almost identical in Fig. 11. For scanned data, NLLR exhibits slightly better performance than our method in the Boy example, even though the results look almost identical. However, in the other two examples (Cone and Pyramid), our method shows better performance in terms of θ values; see the third column of Fig. 15.

To further evaluate the vertex deviation from the ground truth, we use the vertex-based Hausdorff distance [8], [14] to measure the position error between the denoised mesh and the ground truth. The results are listed in Table. 1. As can be seen, for CAD and non-CAD surfaces, our method TGV outperforms the other methods in most cases. For scanned data, CNR shows better performance.

Overall, the quantitative comparisons show that our method is more effective in recovering shape details, including sharp features, multi-scale features, and smooth regions, from the noisy input, leading to the least amount of error in most cases, in comparison to the other methods. It is worth noting that our method performs favorably on all types of the tested meshes (CAD, non-CAD, and scanned meshes) rather than having a peak performance on specific types.

Computational time. We list the execution time of each method in Table 1. As we can see, CNR is the fastest method, thanks to its pre-trained neural networks. BF is slower than CNR, but significantly faster than the other methods. L0 is the slowest for synthetic meshes, while NLLR is the slowest for scanned data. Our method takes more execution time than TV and HO. Furthermore, we adopt the conjugate gradient (CG) method to iteratively solve our two linear systems (one for **N**-subproblem and the other for **v**-subproblem), and found that the runtime can be reduced by decreasing the number of iterations, trading off

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3088118, IEEE Transactions on Visualization and Computer Graphics

TABLE 1: Quantitative evaluation of the results in Figs. 9, 10, 11, 12, and 14. For each result, we list mean angular difference θ (in degrees), vertex-based mesh-to-mesh error E_v $(\times 10^{-2})$ and the execution time (in seconds).

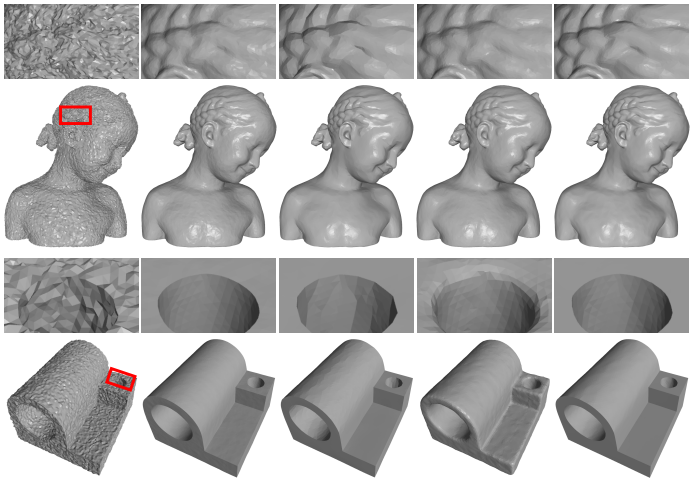
Mesh	TV	HO	L0	BF	NLLR	CNR	TGV
Block	3.12, 2.01; 1.44	2.90, 1.70; 2.03	4.35, 2.15; 13.3	5.30, 1.80; 1.07	11.7, 2.44; 10.4	2.63, **0.86**; 0.71	**1.88**, 0.95; 6.28
Fandisk	2.62, 2.39; 0.88	3.67, 1.67; 1.88	3.92, 2.17; 6.73	5.51, 1.62; 0.61	8.21, 1.57; 3.64	2.31, 1.47; 0.55	**2.20**, **1.20**; 3.08
Lucy	9.88, 0.38; 30.9	9.63, 0.61; 60.1	13.0, 0.72; 87.6	8.86, 0.50; 8.80	8.64, 0.36; 67.5	7.91, 0.30; 10.5	**7.34**, **0.26**; 66.6
Gargoyle	10.8, 0.59; 13.7	9.72, 0.67; 20.3	12.0, 0.63; 55.5	9.94, 2.22; 4.70	8.96, 1.58; 30.5	8.36, 0.77; 6.51	**7.86**, **0.54**; 49.8
Pyramid	6.79, 4.69; 1.28	7.18, 4.23; 1.93	6.50, 3.47; 8.03	8.45, 4.45; 0.77	9.27, 3.41; 104.2	6.40, **3.38**; 0.76	**6.19**, 4.52; 4.14
Cone	7.45, 3.98; 4.48	7.41, 3.33; 7.91	7.80, 3.45; 47.7	8.16, 2.96; 3.59	7.63, 3.04; 312.8	7.11, **2.78**; 1.82	**6.96**, 3.57; 30.1
Boy	9.16, 6.08; 10.2	8.98, **5.66**; 17.3	9.42, 6.04; 88.5	10.0, 6.19; 12.3	**8.88**, 5.97; 409.1	8.97, 6.07; 4.93	8.91, 5.94; 55.8

TABLE 2: Mesh sizes for the surfaces in Table 1

Mesh	Block	Fandisk	Lucy	Gargoyle	Pyramid	Cone	Boy
$\|V\|$	8.8K	6.5K	149.3K	85.6K	6.6K	31.2K	76.9K
$\|F\|$	17.6K	12.9K	298.5K	171.1K	12.6K	61.3K	152.2K

accuracy. Furthermore, since the coefficient matrices of the two linear systems stay fixed during the iteration, they can be pre-factorized and thus the runtime of the full algorithm is still acceptable.

Overall, our method produces much better results in terms of visual quality and error metrics in most cases, although it seems to be computationally more intensive, hence using modern GPUs and multi-core CPUs to speed up our method is one further direction.



(a) Noisy (b) AT (c) MSTV (d) HLO (e) Ours

Fig. 16: Comparison of denosing results of Child and Joint, corrupted with $\sigma = 0.2\bar{l}_e$.

TABLE 3: Quantitative evaluation of the results in Fig. 16 for AT [12], MSTV [12], HLO [34], and our method TGV. For each result, we list mean angular difference θ (in degrees), vertex-based mesh-to-mesh error E_v $(\times 10^{-2})$ and the execution time (in seconds).

Mesh	AT	MSTV	HLO	TGV
Child	7.61, 0.62; 3.80	7.22, 0.55; 7.06	7.93, 0.69; 1.31	**6.29**, **0.48**; 25.9
Joint	1.89, 0.79; 2.84	1.99, 0.81; 2.75	6.06, 2.15; 0.34	**1.45**, **0.70**; 12.4

6.4 TGV vs. AT, MSTV, and HLO

To further demonstrate the effectiveness of TGV, we compare it to the Mumford-Shah methods (AT and MSTV) in [12] and the Laplacian diffusion method (HLO) in [34]. As Fig. 16 shows, for non-CAD meshes containing different levels of features, AT and TGV produce visually better results. MSTV suffers from staircase artifacts, while HLO tends to smooth weak features. For CAD meshes, all methods, except HLO, recovers sharp features in the tested noise level. However, AT causes some bumping in flat regions, while MSTV suffers from staircase artifacts in smooth regions. In contrast, TGV produces visually more compelling results which are free of noticeable artifacts. As we can see in Table 3, for both CAD and non-CAD meshes, the TGV results show the lowest error values, indicating that TGV outperforms the other three competing methods (AT, MSTV and HLO) numerically. Table 3 also lists the CPU execution time for the four methods. We can see that HLO is the fastest method, while TGV is the slowest.

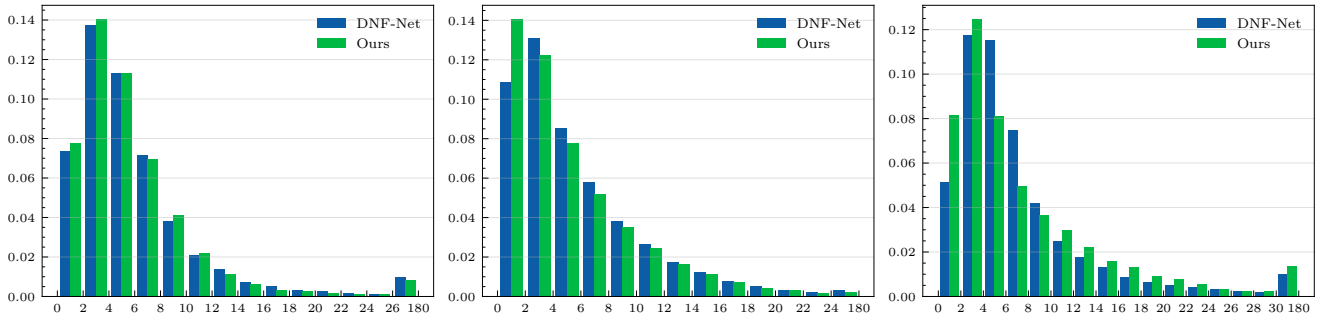
6.5 TGV vs. DNF-Net

Li et al. [18] recently proposed an end-to-end deep normal filtering network, named DNF-Net, which has received wide attention. In Fig. 18, we compare TGV with DNF-Net on three meshes (Sharpsphere, Carter, and Cone04). To further visualize the mean angular difference (θ) distribution for the tested meshes, we show the histogram of θ in Fig. 17. For the mesh containing sharp features and smooth regions (Sharpsphere), our method clearly outperforms DNF-Net in terms of visual quality and the error metric θ; see the top row in Fig. 18. For CAD mesh (Carter), both methods produce excellent feature-preserving results. Nevertheless, the θ value of our result is lower than that of DNF-Net; see the middle row in Fig. 18. For scanned mesh (Cone04), the DNF-Net result has θ value lower than ours, however it contains bumps in smooth regions. In contrast, our result does not show such artifact, hence we believe visually, our result is better; see the last row in Fig. 18. Moreover, as Fig. 17 shows, TGV consistently produces high quality results that contain more θ in the range of $[0^\circ, 2^\circ]$. Therefore, we argue that TGV performs favorably against DNF-Net.

6.6 Discussions

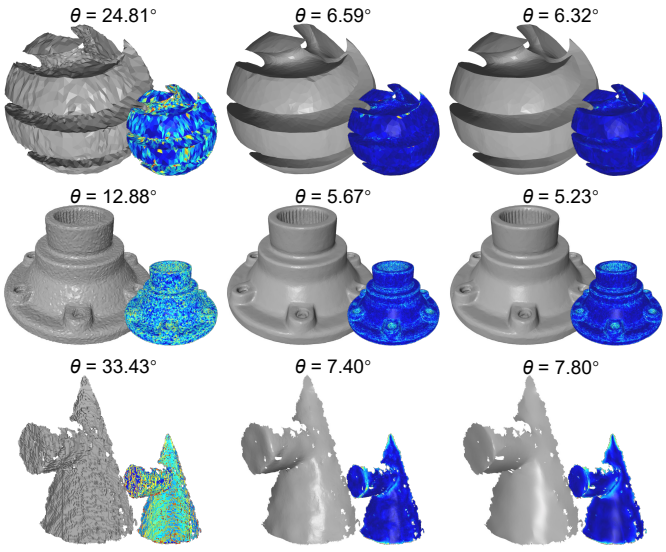
In the following, we discuss the performance of our method in various aspects, including efficacy for irregular sampling, robustness against different levels of noise, and robustness on sampling density (mesh resolution).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3088118, IEEE Transactions on Visualization and Computer Graphics



(a) Sharpsphere (b) Carter (c) Cone04

Fig. 17: Histograms of mean angular difference (θ) of the results in Fig. 18. The horizontal axis denotes θ (in degrees), while the vertical axis denotes the ratio of faces falling in the fixed range of θ.



(a) Noisy (b) DNF-Net (c) Ours

Fig. 18: Comparison between DNF-Net [18] and our method TGV. From left to right: input noisy meshes, denoising results produced by DNF-Net and ours. The corresponding error maps, using the mean angular difference between the face normals of denoised meshes and ground truth meshes, are also demonstrated.

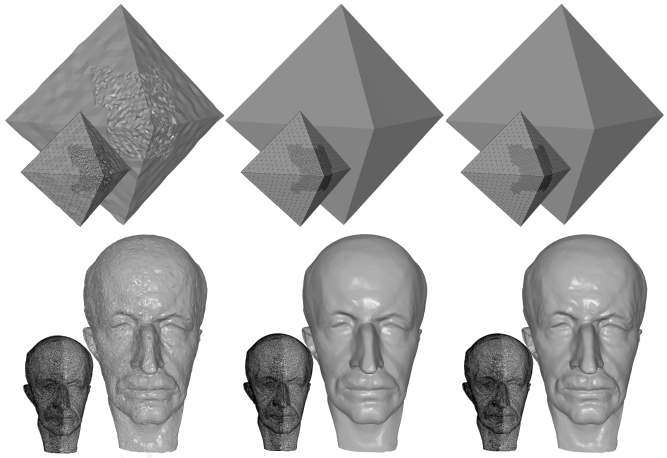
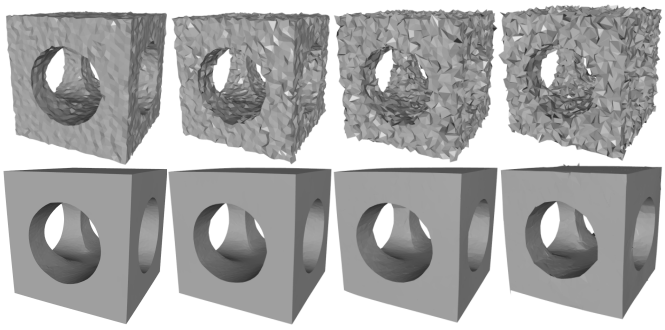


Fig. 19: Denoising results for noisy input with non-uniform sampling. From left to right: input noisy meshes, denoising results, and the corresponding clean meshes.



(a) $\sigma = 0.2\bar{l}_e$ (b) $0.4\bar{l}_e$ (c) $0.6\bar{l}_e$ (d) $0.75\bar{l}_e$

Fig. 20: Denoising results of Part, which is corrupted by different levels of noise. The top row shows the mesh after being corrupted with increasing level of noise, while the bottom row shows the corresponding denoising results.

Irregular sampling. As we have rigorously defined the discrete operators used in our TGV normal filter (27), our method is robust against non-uniform sampling. We demonstrate the robustness of our method against irregular sampling in Fig. 19. As we can see, although the noisy meshes are of varying density distributions, the obtained results still show compelling quality.

Stress test. A stress test of our method with increasing level of noise is presented in Fig. 20. As can be seen, when the noise level is moderate, our method can remove noise effectively, while preserving sharp features and simultaneously recover smooth regions. Moreover, our method can preserve sharp features even for the mesh under the highest level of noise; see Fig. 20c. However, when the noise level increases to be larger than the feature size, our method fails to produce satisfactory results; see Fig. 20d.

Sampling density. A robustness test of our method for varying mesh resolution is shown in Fig. 21. When the mesh resolution decreases, the θ value of our results does not change significantly for CAD mesh (Part) or scanned

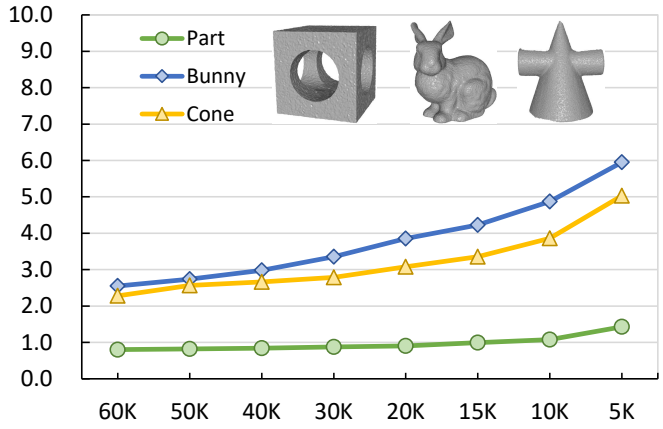


Fig. 21: Error plot of mean angular difference (θ) for three meshes (Part, Bunny, Cone) under different resolutions. All the meshes are corrupted by Gaussian noise with $\sigma = 0.1\bar{l}_e$. The horizontal axis denotes the number of faces in the meshes, while the vertical axis denotes the θ value.

data (Cone). For non-CAD mesh (Bunny), there is a slight jump of θ in the case of low-resolution, yet, overall it is still reasonable. Thus, our method is robust against mesh resolution.

7 Conclusion

In this work, we present a numerical framework to discretize TGV for triangular meshes. A normal filter based on vectorial TGV is proposed to smooth normal fields on meshes. The optimization problem for the proposed filter is efficiently solved by variable-splitting and the augmented Lagrangian method. Then, vertex positions are updated to match the filtered normal field. We carefully evaluate our method in various aspects and compare it to the state-of-the-art methods. Extensive experimental results show that our method has significant advantages in preserving sharp features, recovering smooth transition regions, as well as preventing various artifacts (e.g., staircase artifacts, over-smoothing or over-sharpening effects, and extra noise). In summary, our method is highly effective for denoising CAD and man-made surfaces that contain sharp features and smooth transition regions.

There are many interesting directions for future research. The proposed discretized TGV operator can be applied to other geometry processing problems, such as mesh segmentation, reconstruction, simplification, feature detection, etc. Furthermore, we plan to investigate the possibility to extend our method to point clouds.

Acknowledgments

This work was supported by NSF of China (Nos. 62072422, 12001144, 62025207, 62076227, and 61702467), National Key R&D Program of China (No. 2020YFC1523102), NSF of Anhui Province, China (No. 2008085MF195), Youth Science and Technology Foundation of Gansu (No. 20JR5RA050), NSF of Zhejiang Province, China (No. LQ20A010007), and Zhejiang Lab (No. 2019NB0AB03).

References

[1] Z. Liu, X. Xiao, S. Zhong, W. Wang, Y. Li, L. Zhang, and Z. Xie, "A feature-preserving framework for point cloud denoising," *Comput-Aided Des. (Solid and Physical Modeling)*, vol. 127, p. 102857, 2020.

[2] J. Wang, X. Zhang, and Z. Yu, "A cascaded approach for feature-preserving surface mesh denoising," *Comput-Aided Des.*, vol. 44, no. 7, pp. 597–610, 2012.

[3] J. Wang, J. Huang, F. L. Wang, M. Wei, H. Xie, and J. Qin, "Data-driven geometry-recovering mesh denoising," *Comput-Aided Des. (Solid and Physical Modeling)*, vol. 114, pp. 133–142, 2019.

[4] X. Sun, P. Rosin, R. Martin, and F. Langbein, "Fast and effective feature-preserving mesh denoising," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 5, pp. 925–938, 2007.

[5] Y. Zheng, H. Fu, K. C. Au, and C. L. Tai, "Bilateral normal filtering for mesh denoising," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 10, pp. 1521–1530, 2011.

[6] M. Wei, J. Yu, W.-M. Pang, J. Wang, J. Qin, L. Liu, and P.-A. Heng, "Bi-normal filtering for mesh denoising," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 1, pp. 43–55, 2014.

[7] R. Lai, X.-C. Tai, and T. F. Chan, "A ridge and corner preserving model for surface restoration," *SIAM J. Sci. Comput.*, vol. 35, no. 2, pp. 675–695, 2013.

[8] H. Zhang, C. Wu, J. Zhang, and J. Deng, "Variational mesh denoising using total variation and piecewise constant function space," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 7, pp. 873–86, 2015.

[9] X. Wu, J. Zheng, Y. Cai, and C.-W. Fu, "Mesh denoising using extended rof model with L_1 fidelity," *Comput. Graph. Forum (Pacific Graphics)*, vol. 34, no. 7, pp. 35–45, 2015.

[10] X. Lu, Z. Deng, and W. Chen, "A robust scheme for feature-preserving mesh denoising," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 3, pp. 1181–1194, 2015.

[11] X. Lu, W. Chen, and S. Schaefer, "Robust mesh denoising via vertex pre-filtering and L_1-median normal filtering," *Comput. Aided Geom. Des.*, vol. 114, pp. 133–142, 2019.

[12] Z. Liu, W. Wang, S. Zhong, B. Zeng, J. Liu, and W. Wang, "Mesh denoising via a novel Mumford-Shah framework," *Comput-Aided Des. (Solid and Physical Modeling)*, vol. 126, p. 102858, 2020.

[13] X. Li, L. Zhu, C.-W. Fu, and P.-A. Heng, "Non-local low-rank normal filtering for mesh denoising," *Comput. Graph. Forum (Pacific Graphics)*, vol. 37, no. 7, pp. 155–166, 2018.

[14] M. Wei, H. Jin, X. Xie, L. Liu, and Q. Jing, "Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 10, pp. 2910–2926, 2019.

[15] H. Chen, J. Huang, O. Remil, H. Xie, J. Qin, Y. Guo, M. Wei, and J. Wang, "Structure-guided shape-preserving mesh texture smoothing via joint low-rank matrix recovery," *Comput-Aided Des. (Solid and Physical Modeling)*, vol. 115, pp. 122–134, 2019.

[16] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 232:1–232:12, 2016.

[17] M. Wei, X. Guo, J. Huang, F. Wang, H. Xie, R. Kwan, and J. Qin, "Mesh defiltering via cascaded geometry recovery," *Comput. Graph. Forum*, vol. 38, no. 7, pp. 591–605, 2019.

[18] X. Li, R. Li, L. Zhu, C. Fu, and P. Heng, "DNF-Net: a deep normal filtering network for mesh denoising," *IEEE Trans. Vis. Comput. Graph.*, 2020.

[19] Z. Liu, S. Zhong, Z. Xie, and W. Wang, "A novel anisotropic second order regularization for mesh denoising," *Comput. Aided Geom. Des. (Geomtric Modeling and Processing)*, vol. 71, pp. 190–201, 2019.

[20] Z. Liu, R. Lai, H. Zhang, and C. Wu, "Triangulated surface denoising using high order regularization with dynamic weights," *SIAM J. Sci. Comput.*, vol. 41, no. 1, pp. 1–26, 2019.

[21] S. Zhong, Z. Xie, J. Liu, and Z. Liu, "Robust mesh denoising via triple sparsity," *Sensors*, vol. 19, p. 1001, 2019.

[22] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM J. Imaging Sci.*, vol. 3, no. 3, pp. 492–526, 2010.

[23] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof, "Image guided depth upsampling using anisotropic total generalized variation," in *IEEE International Conference on Computer Vision*, 2013, pp. 993–1000.

[24] M. Jung and M. Kang, "Simultaneous cartoon and texture image restoration with higher-order regularization," *SIAM J. Imaging Sci.*, vol. 8, no. 1, pp. 721–756, 2015.

[25] W. Feng, H. Lei, and Y. Gao, "Speckle reduction via higher order total variation approach," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1831–1843, 2014.

[26] F. Knoll, K. Bredies, T. Pock, and R. Stollberger, "Second order total generalized variation (TGV) for MRI," *Magn. Reson. Med.*, vol. 65, no. 2, pp. 480–491, 2011.

[27] S. Niu, Y. Gao, Z. Bian, J. Huang, W. Chen, G. Yu, Z. Liang, and J. Ma, "Sparse-view x-ray CT reconstruction via total generalized variation regularization," *Phys. Med. Biol.*, vol. 59, no. 12, pp. 2997–3017, 2014.

[28] G. Taubin, "A signal processing approach to fair surface design," in *Proc. 22nd Annu. Conf. Comput. Graph. Interactive Tech.*, 1995, pp. 351–358.

[29] M. Desbrun, M. Meyer, P. Schröder, and A.-H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Tech.*, 1999, pp. 317–324.

[30] C. Bajaj and G. Xu, "Anisotropic diffusion of surfaces and functions on surfaces," *ACM Trans. Graph.*, vol. 22, no. 1, pp. 4–32, 2003.

[31] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *ACM Trans. Graph. (SIGGRAPH)*, pp. 950–953, 2003.

[32] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," *ACM Trans. Graph. (SIGGRAPH)*, pp. 943–949, 2003.

[33] C. Wang, "Bilateral recovering of sharp edges on feature-insensitive sampled meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 4, pp. 629–639, 2006.

[34] W. Pan, X. Lu, Y. Gong, W. Tang, J. Liu, Y. He, and G. Qiu, "HLO: Half-kernel Laplacian operator for surface smoothing," *Comput-Aided Des.*, vol. 121, p. 102807, 2020.

[35] M. Wei, L. Liang, W. M. Pang, J. Wang, W. Li, and H. Wu, "Tensor voting guided mesh denoising," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 931–945, 2017.

[36] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, and L. Liu, "Guided mesh normal filtering," *Comput. Graph. Forum (Pacific Graphics)*, vol. 34, no. 7, pp. 23–34, 2015.

[37] J. Zhang, B. Deng, Y. Hong, Y. Peng, W. Qin, and L. Liu, "Static/dynamic filtering for mesh geometry," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 4, pp. 1774–1787, 2019.

[38] S. Yadav, U. Reitebuch, and K. Polthier, "Mesh denoising based on normal voting tensor and binary optimization," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 8, pp. 2366–2379, 2018.

[39] ——, "Robust and high fidelity mesh denoising," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 6, pp. 2304–2310, 2019.

[40] G. Arvanitis, A. S. Lalos, K. Moustakas, and N. Fakotakis, "Feature preserving mesh denoising based on graph spectral processing," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 3, pp. 1513–1527, 2018.

[41] W. Zhao, X. Liu, S. Wang, X. Fan, and D. Zhao, "Graph-based feature-preserving mesh normal filtering," *IEEE Trans. Vis. Comput. Graph.*, 2019.

[42] L. He and S. Schaefer, "Mesh denoising via L_0 minimization," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–8, 2013.

[43] Y. Zhao, H. Qin, X. Zeng, J. Xu, and J. Dong, "Robust and effective mesh denoising using L_0 sparse regularization," *Comput-Aided Des.*, vol. 101, pp. 82–97, 2018.

[44] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3d geometry filtering," *IEEE Trans. Vis. Comput. Graph.*, 2020.

[45] Z. Li, Y. Zhang, Y. Feng, X. Xie, Q. Wang, M. Wei, and P.-A. Heng, "NormalF-Net: Normal filtering neural network for feature-preserving mesh denoising," *Comput-Aided Des. (Solid and Physical Modeling)*, vol. 127, p. 102861, 2020.

[46] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[47] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "ℓ_1-sparse reconstruction of sharp point set surfaces," *ACM Trans. Graph.*, vol. 29, no. 5, pp. 135:1–12, 2010.

[48] 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.*, vol. 3, no. 3, pp. 300–339, 2010.

[49] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

Zheng Liu is currently an associate professor in China University of Geosciences (Wuhan). He received PhD from Central China Normal University in 2012. From 2013 to 2014, he held a post-doctoral position with School of Mathematical Sciences, University of Science and Technology of China. His research interests include geometry processing, computer graphics, 3D computer vision and deep learning.

Yanlei Li is currently a M.S. candidate in China University of Geosciences (Wuhan). He received B.S. degree from China University of Geosciences (Wuhan), in 2019. His research interests include geometry processing and computer graphics.

Weina Wang is currently a lecturer in the department of Mathematics, Hangzhou Dianzi University, China. She received the Ph.D. degree from the University of Science and Technology of China, in 2018. Her research interests include numerical optimization and image processing.

Ligang Liu is a professor at the University of Science and Technology of China. He received his PhD from Zhejiang University in 2001. He once worked at Microsoft Research Asia, Zhejiang University, and visited Harvard University. His research interests include computer graphics and geometry processing. He serves as the associated editors for journals including IEEE TVCG, IEEE CG&A, CAGD, C&G, The Visual Computer, etc. He served as the conference co-chair of GMP 2017 and the program co-chairs of conferences including Chinagraph 2020, SIAM GD 2019, GMP 2018, CAD/Graphics 2017, CVM 2016, SGP 2015, and SPM 2014. He serves as the steering committee member of GMP and the secretary of Asia-graphics Association.

Renjie Chen is a professor at the University of Science and Technology of China (USTC). He holds a PhD degree from Zhejiang University, China. Before joining USTC, he was a postdoctoral fellow at the Technion–Israel Institute of Technology, a postdoctoral research associate at the University of North Carolina at Chapel Hill, a key researcher in the BeingThere Center in Nanyang Technological University, Singapore, and a senior researcher heading a research group working on 3D geometry and images at the Max Planck Institute for Informatics (MPII) in Saarbrucken, Germany. His research interests includes computer graphics, geometry modeling, computational geometry and glasses-free 3D display.