Computer-Aided Design 126 (2020) 102858

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Mesh Denoising via a Novel Mumford–Shah Framework

Zheng Liu [a], Weina Wang [b], Saishang Zhong [c,*], Bohong Zeng [d], Jinqin Liu [d], Weiming Wang [e]

[a] School of Geography and Information Engineering, National Engineering Research Center of Geographic Information System, China University of Geosciences, Wuhan 430074, China
[b] Department of Mathematics, Hangzhou Dianzi University, Hangzhou 310018, China
[c] School of Earth Resources, State Key Laboratory of Geological Processes and Mineral Resources, China University of Geosciences, Wuhan 430074, China
[d] School of Geography and Information Engineering, China University of Geosciences, Wuhan 430074, China
[e] School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China

ARTICLE INFO

Article history:
Received 26 March 2020
Accepted 16 April 2020

Keywords:
Mesh denoising
Mumford–Shah functional
Γ-convergence approximation
Feature preserving

ABSTRACT

In this paper, we introduce a Mumford–Shah framework to restore the face normal field on the triangulated surface. To effectively discretize Γ-convergence approximation of the Mumford–Shah model, we first define an edge function space and its associated differential operators. They are helpful for directly diffusing the discontinuity function over mesh edges instead of computing the approximated discontinuity function via pointwise diffusion in existing discretizations. Then, by using the operators in the proposed function space, two Mumford–Shah-based denoising methods are presented, which can produce denoised results with neat geometric features and locate geometric discontinuities exactly. Our Mumford–Shah framework overcomes the limitations of existing techniques that blur the discontinuity function, be less able to preserve geometric features, be sensitive to surface sampling, and require a postprocessing to form feature curves from located discontinuity vertices. Intensive experimental results on a variety of surfaces show the superiority of our denoising methods qualitatively and quantitatively.

1. Introduction

Mesh denoising is a typical inverse problem in geometry processing. By a given noisy input, the main goal of mesh denoising is to recover a noise-free mesh while preserving essential geometric features from the underlying surface. With the rapid development of scanner devices, more and more meshes can be easily acquired from the real world. Yet, the scanning process inevitably produces some level of noise due to local measurement errors. These noise not only degrade the quality of meshes, but also cause errors in downstream geometry processing applications [1]. Thus, the task of removing the noise from the corrupted mesh while preserving geometric features becomes increasingly important, especially in the case of high noise.

1.1. Related work

Over the last two decades, great efforts have been done on the mesh denoising problem. Although there are numerous methods in literature, it is beyond our scope to review all existing methods, and we only review several notable methods and those are most relevant to this paper.

Filter-based methods. Early isotropic filtering methods [2,3] are effective for removing the noise, but they do not consider underlying geometric features during the filtering process and often cause significant shape distortion. In order to tackle this problem, a variety of methods [4–7] based on anisotropic filtering were proposed, most of which can be seen as processes of shrinking weighted surface area of the mesh. Recently, the use of normal filtering followed by vertex updating [4,8–16] has become so widespread that it could arguably substitute for directly smoothing vertex positions. Although these normal filtering methods can preserve geometric features to some extent, they still have limitations. For example, bilateral normal filtering [9] is not well developed for preserving sharp features, especially in the presence of large noise, though it can effectively recover fine details and smooth regions. The guided normal filtering proposed by Zhang et al. [17] preserves sharp features well, but it sometimes lacks the robustness to mesh topology. The normal filtering method proposed in [13] can stop diffusion at sharp features and produce smooth transition regions. Nevertheless, due to the using of the robust error estimator, this method tends to oversmooth small scale features and fine details.

* Corresponding author.
E-mail address: cugsaishang@foxmail.com (S. Zhong).

https://doi.org/10.1016/j.cad.2020.102858
0010-4485/

Optimization-based methods. Recently, variational methods have received extensive concern. They formulate the denoising process as an optimization problem and seek for a desired solution satisfying the optimization goal. Zheng et al. [9] proposed a global bilateral weighting normal filtering model. Their method performs well on non-CAD meshes with fine details, but cannot do a good job on CAD meshes including sharp features, especially in the case of high noise. Total variation (TV) normal filtering [10] and ℓ_0 minimization [18,19] achieve impressive results for preserving sharp features but inevitably flatten some weak features and fine details for their sparsity requirements. In particular, this drawback is more severe for ℓ_0 minimization [18], which produces false edges in smoothly curved regions. To overcome these problems, high order methods [15,20] were proposed to restore the face normal field. Yet, the high order methods tend to blur sharp features, especially for dealing with large noise. Some low-rank based methods [21–23] were proposed to recover pattern similarity patches of the mesh. However, these methods still seem to have difficulties to effectively handle sharp features.

Data-driven methods. Several data-driven mesh denoising methods [22,24–26] have been proposed typically. Wang et al. [25] proposed a method based on the cascades normal regression, which can remove the noise without assumptions about geometric features of the underlying surface and noise patterns. Their method first learns non-linear regression functions mapping the filtered face normal descriptors to the face normals of the ground-truth input, and then applied the learned functions to compute filtered face normals. Wang et al. [26] and Wei et al. [22] presented data-driven methods by learning normal variations in two steps. In the first step, they learn mapping from the noisy input to its ground-truth counterpart and use neural networks to remove the noise, which might lost some fine details of the underlying surface. Then in the second step, they learn to recover the missing details. Although these data-driven methods perform well for scanned data, the performance of these methods depends on the completeness of the training data set.

Mumford–Shah framework. More recently, it has attracted great interest to study the Mumford–Shah (MS) model in geometry processing by adapting the idea from image processing. The MS model was originally proposed in [27], which has been proved very successful in dealing with image restoring [28–30] and segmentation [31–33]. Since the classical MS functional is non-convex, it is challenging to directly minimize it. Thus, numerical schemes to approximate the MS functional have been widely studied. Using Γ-convergence theory, Ambrosio and Tortorelli [34] have approximated the MS functional by two coupled elliptic functionals. This numerical scheme, called AT approximation, is one of the most successful approximations of the MS functional.

Inspired by the success of the AT approximation in image processing, it has been extended to 3D data in [14,35,36]. Specifically, Coeurjolly et al. [36] discretized the AT approximation for dealing with voxel-based data, while Tong and Tai [35] and Bonneel et al. [14] discretized it over triangulated surfaces. However, a fundamental limitation of the existing discretizations [14,35,36] is that they all compute the discontinuity function of the approximation based on pointwise diffusion, which tends to make the discontinuity function mismatch the underlying geometric features. This mismatch degrades the quality of denoised results, especially in the case of high noise. Moreover, because the existing discretizations compute the discontinuity function on vertices, an extra postprocessing is needed to connect located discontinuity vertices to form feature curves. Thus, developing an effective framework to discretize the MS functional over triangulated surfaces is still an open problem.

1.2. Contribution

In this paper, we discretize Γ-convergence approximations of the MS functional in a novel framework. In order to discretize the approximations, two function spaces are introduced over the mesh. One is the piecewise constant function space in which we smooth the face normal field, and the other is the edge function space in which we diffuse the discontinuity function. These two function spaces are tightly connected by the differential mapping between them. Compared to the existing discretizations [14,35,36], the proposed discretizations calculate the discontinuity function in a completely different manner, which yields better denoised results and more accurately located geometric discontinuities. Our main contributions are three-fold:

- Two coupled function spaces and associated operators are given out over triangulated surfaces. To the best of our knowledge, this is the first work to describe the edge function space and its operators for directly diffusing the function over mesh edges.
- Two Mumford–Shah-based models are formulated in the proposed function spaces, which are more able to produce high quality denoised results with neat features and at the same time locate discontinuities of the surface accurately.
- Two efficient algorithms based on alternating minimization are presented to solve the proposed Mumford–Shah regularizations, which can be easily implemented.

2. Background of Mumford–Shah functional and its Γ-convergence approximations

For the sake of completeness and readability, we give a brief review of the Mumford–Shah (MS) regularizing functional and its Γ-convergence variants. For a scalar image $u : \Omega \to \mathbb{R}$ with its discontinuity set $\mathrm{K} \subset \Omega$, the original MS model is to minimize the following MS functional:

$$\mathcal{MS}(u, \mathrm{K}) = \gamma \int_{\Omega \setminus \mathrm{K}} |\nabla u|^2 + \beta \int_{\mathrm{K}} dl + \alpha \int_{\Omega} (u - f)^2, \tag{1}$$

f denotes an observed image, γ, β, and α are tuning parameters. The first term of (1) smoothes u except on the discontinuity set K, the second term minimizes the length of the set K, and the last term helps the solution to harmonize well with the observed image f. Due to the coupling of the function u and the length set K, it is challenging to minimize the MS functional (1).

Using the Γ-convergence theory, Ambrosio and Tortorelli [34] approximated the MS functional (1) by elliptic functionals. The AT approximation of (1) is defined as follows

$$\mathrm{AT}(u, v) = \int_{\Omega} \gamma(v^2 |\nabla u|^2) + \beta(\epsilon |\nabla v|^2 + \frac{(v-1)^2}{4\epsilon}) + \alpha(u - f)^2, \tag{2}$$

where ϵ is a small positive constant. When $\epsilon \to 0$, $\mathrm{AT}(u, v)$ converges to $\mathcal{MS}(u, \mathrm{K})$. v is an introduced auxiliary variable to be used instead of the discontinuity set K. More specifically, v is close to 0 on the discontinuities and 1 otherwise. Thus, the values of $1 - v$ range between 0 and 1 and can be interpreted as the probability for the presence of the discontinuity. The minimizing of AT approximation (2) is to find an equilibrium between two competing minimizing processes of u and v.

Shah [37] suggested a modified version of the AT approximation (2), by replacing the quadratic term $|\nabla u|^2$ by a total variation (TV) term $|\nabla u|$, defined as

$$\mathrm{MSTV}(u, v) = \int_{\Omega} \gamma(v^2 |\nabla u|) + \beta(\epsilon |\nabla v|^2 + \frac{(v-1)^2}{4\epsilon}) + \alpha(u - f)^2. \tag{3}$$

The Γ-convergence of the Mumford–Shah total variation (MSTV) approximation (3) was proved in [38]. Due to the using of the TV term, the MSTV approximation has excellent edge-preserving property. Compared with the AT approximation (2), the MSTV approximation (3) is more robust against noise and can recover much clearer edges, but tends to suppress fine details, especially in the case of high noise [29,39].

For $\mathfrak{N}$-channel images $\mathbf{u}, \mathbf{f} : \Omega \to \mathbb{R}^{\mathfrak{N}}$, where $\mathbf{u} = (u_1, u_2, \ldots, u_{\mathfrak{N}})$, the AT (2) and MSTV (3) approximations can be naturally extended to their vectorial versions as follows:

$$\mathbf{AT}(\mathbf{u}, v) = \int_{\Omega} \gamma(v^2\|\nabla\mathbf{u}\|^2)+\beta(\epsilon|\nabla v|^2+\frac{(v-1)^2}{4\epsilon})+\alpha\|\mathbf{u}-\mathbf{f}\|^2, \quad (4)$$

$$\mathbf{MSTV}(\mathbf{u}, v)=\int_{\Omega} \gamma(v^2\|\nabla\mathbf{u}\|)+\beta(\epsilon|\nabla v|^2+\frac{(v-1)^2}{4\epsilon})+\alpha\|\mathbf{u}-\mathbf{f}\|^2, \quad (5)$$

where $\|\nabla\mathbf{u}\| = \left(\sum_{i=1}^{\mathfrak{N}} |\nabla u_i|^2\right)^{\frac{1}{2}}$. Note that, in the above two vectorial approximations, the scalar function v is common for the $\mathfrak{N}$ channels.

3. Basic function spaces and operators

To effectively discretize Γ-convergence approximations of the Mumford–Shah functional, we define some basic function spaces and associated differential operators in this section.

3.1. Notations

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 $\{p_i : i = 0, 1, \ldots, \mathrm{P}-1\}$, $\{e_i : i = 0, 1, \ldots, \mathrm{E}-1\}$, and $\{\tau_i : i = 0, 1, \ldots, \mathrm{T}-1\}$, respectively. Here P, E, and T are the numbers of vertices, edges, and triangles of $\mathcal{M}$, respectively. If p is an endpoint of an edge e, then we write it as $p \prec e$. Similarly, $e \prec \tau$ denotes that e is an edge of a triangle τ; $p \prec \tau$ denotes that p is a vertex of a triangle τ.

We further introduce the relative orientation of an edge e to a triangle τ, which is denoted by $\mathrm{sgn}(e, \tau)$ as follows. First, we assume that all triangles are with counterclockwise orientation and all edges are with randomly chosen fixed orientations. For an edge $e \prec \tau$, if the orientation of e is consistent with the orientation of τ, then $\mathrm{sgn}(e, \tau) = 1$; otherwise $\mathrm{sgn}(e, \tau) = -1$.

3.2. Function spaces and associated operators

Given a triangulated surface $\mathcal{M}$, we now define two basic function spaces. The space $U = \mathbb{R}^{\mathrm{T}}$ is a set, whose elements are the values at the faces of $\mathcal{M}$, which is also called the piecewise constant function space in [10,15,40]. For instance, $u = (u_0, u_1, \ldots, u_{\mathrm{T}-1}) \in U$ means that the value of u restricted on the triangle τ is u_τ, which is written as $u|_\tau$ sometimes. The space $V = \mathbb{R}^{\mathrm{E}}$ is a set, whose elements are the values at mesh edges of $\mathcal{M}$. We also call the space V as the edge function space $\mathcal{E}$ sometimes. Likewise, the component of $v \in V$ is v_e, which is the value restricted on the edge e written as $v|_e$ sometimes.

We equip the spaces U and V with inner products and norms. For $u^1, u^2, u \in U$, we define

$$(u^1, u^2)_U = \sum_{\tau} u^1|_\tau u^2|_\tau s_\tau, \qquad \|u\|_U = \sqrt{(u, u)_U}, \quad (6)$$

where s_τ is the area of triangle τ. For $v^1, v^2, v \in V$, we have

$$(v^1, v^2)_V = \sum_{e} v^1|_e v^2|_e \mathrm{len}(e), \quad \|v\|_V = \sqrt{(v, v)_V}, \quad (7)$$

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

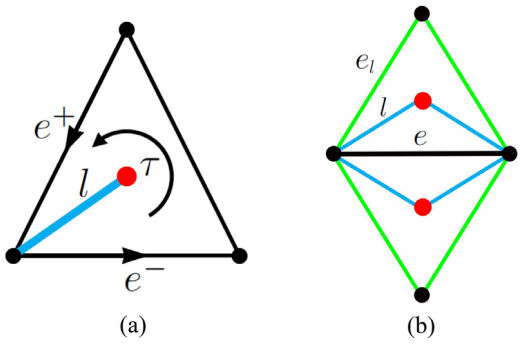

Fig. 1. (a) The illustration of $[v]_l$ over the line l plotted in *cyan* in triangle τ with the barycenter plotted in *red*. (b) The illustration of $H(e)$, which is the set of lines associated with the edge e. The lines contained in $H(e)$ are plotted in *cyan*. The illustration of e_l, which is the edge sharing the common vertex of e and l. The set of e_l associated with the lines contained in $H(e)$ refers to four edges, and the elements contained in the set are plotted in *green*.

The differential operators $\nabla_{\mathcal{M}}$, $\mathrm{div}_{\mathcal{M}}$, and $\Delta_{\mathcal{M}}$ on $\mathcal{M}$ are approximated by using first order finite differences. For $u \in U$, the gradient operator $\nabla_{\mathcal{M}} : U \to V$, is given by

$$(\nabla_{\mathcal{M}} u)|_e = \begin{cases} \sum_{\tau, e\prec\tau} u_\tau \mathrm{sgn}(e, \tau), & e \not\subset \partial\mathcal{M} \\ 0, & e \subset \partial\mathcal{M} \end{cases}, \quad \forall e. \quad (8)$$

For $v \in V$, the divergence operator $\mathrm{div}_{\mathcal{M}} : V \to U$, as the adjoint operator of $-\nabla_{\mathcal{M}}$, is expressed as

$$(\mathrm{div}_{\mathcal{M}} v)|_\tau = -\frac{1}{s_\tau} \sum_{\substack{e\prec\tau, \\ e\not\subset\partial\mathcal{M}}} v_e \mathrm{sgn}(e, \tau)\mathrm{len}(e), \quad \forall\tau. \quad (9)$$

The Laplace operator $\Delta_{\mathcal{M}} : U \to U$ has the following form:

$$(\Delta_{\mathcal{M}} u)|_\tau = -\frac{1}{s_\tau} \sum_{\substack{e\prec\tau, \tau\cap\tau_e=e, \\ e\not\subset\partial\mathcal{M}}} (u_\tau - u_{\tau_e})\mathrm{len}(e), \quad \forall\tau. \quad (10)$$

We refer the readers to [10,15,40] for more details about the above operators.

Then, we will give the definitions of operators $\nabla_{\mathcal{E}}$, $\mathrm{div}_{\mathcal{E}}$, and $\Delta_{\mathcal{E}}$ on the edge function space $\mathcal{E}$, which are first proposed to describe the diffusion of function defined at mesh edges. Again, these operators are approximated by first order finite differences.

Let l be a line connecting the barycenter and one vertex of the triangle τ. For $v \in V$, we define the jump of v over a line l as

$$[v]_l = v_{e^+}\mathrm{sgn}(e^+, l) + v_{e^-}\mathrm{sgn}(e^-, l), \quad (11)$$

where e^+ and e^- are two edges sharing the common vertex of l. e^+ enters the common vertex in the counterclockwise direction, whereas e^- leaves the vertex in the counterclockwise direction. Since all of the triangles are with counterclockwise orientation, we can directly set $\mathrm{sgn}(e^+, l) = 1$ and $\mathrm{sgn}(e^-, l) = -1$. All aforementioned descriptions are illustrated in Fig. 1a.

The gradient operator on the edge function space $\mathcal{E}$ is naturally defined as

$$\nabla_{\mathcal{E}} : V \to W, \quad (\nabla_{\mathcal{E}} v)|_l = [v]_l, \ \forall l, \ \text{for } v \in V, \quad (12)$$

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

$$(w^1, w^2)_W = \sum_{l} w^1|_l w^2|_l \mathrm{len}(l), \quad \|w\|_W = \sqrt{(w, w)_W}, \quad (13)$$

for $w^1, w^2, w \in W$, where $\mathrm{len}(l)$ is the length of line l.

The adjoint operator of $\nabla_{\mathcal{E}}$, namely $\mathrm{div}_{\mathcal{E}} : W \to V$, can be derived by using the above inner products in V and W. For $w \in W$, $\mathrm{div}_{\mathcal{E}} w$ is given by

$$(\mathrm{div}_{\mathcal{E}} w)|_e = -\frac{1}{\mathrm{len}(e)} \sum_{l \in H(e)} w_l \mathrm{sgn}(e,l)\mathrm{len}(l), \quad \forall e, \tag{14}$$

where $H(e)$ is the set of lines associated with the edge e; see Fig. 1b. The readers interested in the mathematical derivation of $\mathrm{div}_{\mathcal{E}}$ can find more information in Appendix.

Using the gradient operator (12) and its adjoint operator (14), the Laplace operator $\Delta_{\mathcal{E}} = \mathrm{div}_{\mathcal{E}} \nabla_{\mathcal{E}} : V \to V$, can be derived as:

$$\begin{aligned}(\Delta_{\mathcal{E}} v)|_e &= -\frac{1}{\mathrm{len}(e)} \sum_{l \in H(e)} \Big(v_e \mathrm{sgn}(e,l) + v_{e_l} \mathrm{sgn}(e_l, l) \Big) \mathrm{sgn}(e,l)\mathrm{len}(l) \\ &= -\frac{1}{\mathrm{len}(e)} \sum_{l \in H(e)} (v_e - v_{e_l})\mathrm{len}(l), \quad \forall e, \text{ for } v \in V,\end{aligned} \tag{15}$$

where e_l is the edge sharing the common vertex of e and l, indicated as the green line in Fig. 1b.

To handle vectorial data, we extend the above concepts to vectorial cases. Three vectorial spaces $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are defined as:

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

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

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

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

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

for $\mathbf{u}^1, \mathbf{u}^2, \mathbf{u} \in \mathbf{U}$, $\mathbf{v}^1, \mathbf{v}^2, \mathbf{v} \in \mathbf{V}$, and $\mathbf{w}^1, \mathbf{w}^2, \mathbf{w} \in \mathbf{W}$. Thus, all aforementioned operators in this section can be computed channel by channel.

4. Mesh denoising using Mumford–Shah regularizations

In this section, we first propose two Mumford–Shah-based normal filtering by utilizing the operators introduced in Section 3. The vertex positions are reconstructed according to the filtered face normals. Then, we discuss the differences between the two proposed MS-based normal filtering. At last, we compare the proposed discretization of the AT approximation with the existing ones.

4.1. AT normal filtering

Given a noisy mesh, we denote its face normals as $\mathbf{N}^{in}$. Assume v to be a function to represent the discontinuity of the face normals $\mathbf{N}$, where $v_e \approx 1$ in the homogeneous regions of $\mathbf{N}$ and $v_e \approx 0$ on the jump set. To remove noise in $\mathbf{N}^{in}$ through vectorial AT approximation (4) with unit normal constraints, we propose the following variational model

$$\begin{aligned}\min_{\mathbf{N} \in C_{\mathbf{N}}, v \in V} \Big\{ &\mathbf{AT}(\mathbf{N}, v) = \gamma \|v(\nabla_{\mathcal{M}} \mathbf{N})\|_{\mathbf{V}}^2 \\ &+ \beta(\epsilon \|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}) + \alpha \left\|\mathbf{N} - \mathbf{N}^{in}\right\|_{\mathbf{U}}^2 \Big\},\end{aligned} \tag{16}$$

where $C_{\mathbf{N}} = \{\mathbf{N} \in \mathbb{R}^{3\times \mathrm{T}} : \|\mathbf{N}_\tau\| = 1, \forall \tau\}$. Although in theory $\epsilon \to 0$, in practice we fix it by 0.001. The minimization problem (16) can be split into two subproblems with respect to the variables $\mathbf{N}$ and v. Thus, the AT normal filtering model (16) is solved by alternatively minimizing the following two subproblems:

- The $\mathbf{N}$-subproblem: for fixed v

$$\min_{\mathbf{N} \in C_{\mathbf{N}}} \gamma \|v(\nabla_{\mathcal{M}} \mathbf{N})\|_{\mathbf{V}}^2 + \alpha \left\|\mathbf{N} - \mathbf{N}^{in}\right\|_{\mathbf{U}}^2; \tag{17}$$

- The v-subproblem: for fixed $\mathbf{N}$

$$\min_{v \in V} \gamma \|v(\nabla_{\mathcal{M}} \mathbf{N})\|_{\mathbf{V}}^2 + \beta(\epsilon \|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}). \tag{18}$$

The $\mathbf{N}$-subproblem (17) is a quadratic minimization with the unit normal constraints. Here we adopt an approximate strategy to solve the problem. We first ignore the unit normal constraints and solve a quadratic programming and then project the minimizer onto a unit sphere. The Euler–Lagrange equation of the problem (17) (without the unit normal constraints) is given as

$$\alpha \mathbf{N} - \gamma \Big(\mathrm{div}_{\mathcal{M}} (v^2 \nabla_{\mathcal{M}} \mathbf{N}) \Big) = \alpha \mathbf{N}^{in}. \tag{19}$$

Eq. (19) can be reformulated into a sparse and positive linear system, which is able to be solved using various numerical packages such as Eigen, Taucs, and Math Kernal Library (MKL).

The v-subproblem (18) is also a quadratic minimization. The Euler–Lagrange equation of it is

$$(2\gamma \|\nabla_{\mathcal{M}} \mathbf{N}\|^2 + \frac{\beta}{2\epsilon}) v - 2\beta\epsilon(\Delta_{\mathcal{E}} v) = \frac{\beta}{2\epsilon}, \tag{20}$$

which also can be reformulated into a sparse linear system.

The alternating minimization procedure for solving (16) is sketched in Algorithm 1. The iteration procedure terminates when one of the stopping criteria is satisfied.

Algorithm 1: Solving AT normal filtering model (16)

Initialization: $\mathbf{N}^{-1} = 0$, $v^{-1} = 0$, $k = 0$, $\varepsilon = 1e-6$;
repeat
  Solve $\mathbf{N}$-subproblem
    For fixed v^{k-1}, compute $\mathbf{N}^k$ from (19) ;
    Normalize $\mathbf{N}^k$;
  Solve v-sub problem
    For fixed $\mathbf{N}^k$, compute v^k from (20) ;
until $\|\mathbf{N}^k - \mathbf{N}^{k-1}\|_{\mathbf{U}} < \epsilon$ or $k \ge 30$;
return $\mathbf{N}^k$.

4.2. MSTV normal filtering

The normal filtering model based on the vectorial MSTV approximation (5) is proposed, as follows:

$$\begin{aligned}\min_{\mathbf{N} \in C_{\mathbf{N}}, v \in V} \Big\{ &\mathbf{MSTV}(\mathbf{N}, v) = \gamma \sum_e v_e^2 \|(\nabla_{\mathcal{M}} \mathbf{N})|_e\| \,\mathrm{len}(e) \\ &+ \beta(\epsilon \|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}) + \alpha \left\|\mathbf{N} - \mathbf{N}^{in}\right\|_{\mathbf{U}}^2 \Big\}.\end{aligned} \tag{21}$$

Due to the nondifferentiability and nonlinearity of the MSTV normal filtering model (21), it is challenging to directly solve the problem. Recently, variable splitting and augmented Lagrangian method (ALM) has achieved great successes in ℓ_1 related problems [10,15,41]. Here, we introduce an auxiliary variable and employ ALM to solve the minimization problem (21).

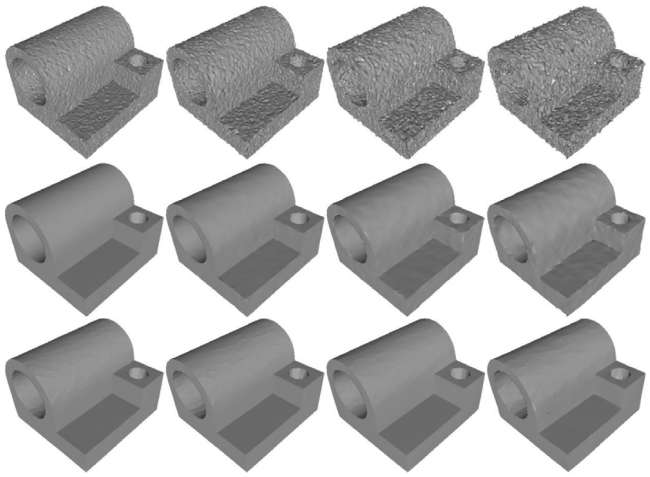

Fig. 2. Denoising results of joint corrupted by different levels of noise. The first row shows noisy meshes (corrupted with $\sigma = 0.2, 0.3, 0.4$, and $0.5\bar{l}_e$, where σ is standard deviation of Gaussian noise and $\bar{l}_e$ is mean edge length). The second row shows the corresponding results produced by AT, and the third row illustrates results generated by MSTV.

We first introduce a new variable $\mathbf{p} \in \mathbf{V}$ and reformulate the problem (21) as

$$\min_{\mathbf{N}\in C_{\mathbf{N}}, v\in V, \mathbf{p}\in\mathbf{V}} \left\{ \gamma \sum_e v_e^2 \|\mathbf{p}_e\| \operatorname{len}(e) + \beta(\epsilon \|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}) + \alpha \left\|\mathbf{N} - \mathbf{N}^{in}\right\|_{\mathbf{U}}^2 \right\} \quad \text{s.t.} \quad \mathbf{p} = \nabla_{\mathcal{M}}\mathbf{N}. \tag{22}$$

To solve (22), we define the augmented Lagrangian function

$$\mathcal{L}(\mathbf{N}, v, \mathbf{p}; \lambda_{\mathbf{p}}) = \gamma \sum_e v_e^2 \|\mathbf{p}_e\| \operatorname{len}(e) + \beta(\epsilon \|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}) + \alpha \left\|\mathbf{N} - \mathbf{N}^{in}\right\|_{\mathbf{U}}^2 + (\lambda_{\mathbf{p}}, \mathbf{p} - \nabla_{\mathcal{M}}\mathbf{N})_{\mathbf{V}} + \frac{r_{\mathbf{p}}}{2} \|\mathbf{p} - \nabla_{\mathcal{M}}\mathbf{N}\|_{\mathbf{V}}^2, \tag{23}$$

where $\lambda_{\mathbf{p}}$ is a Lagrange multiplier and $r_{\mathbf{p}}$ is a positive penalty coefficient. The primal variables update procedure can be separated into three sub problems:

- The **N**-subproblem: for fixed **p**

$$\min_{\mathbf{N}\in C_{\mathbf{N}}} \alpha \|\mathbf{N} - \mathbf{N}^{in}\|_{\mathbf{U}}^2 + \frac{r_{\mathbf{p}}}{2} \|\nabla_{\mathcal{M}}\mathbf{N} - (\mathbf{p} + \frac{\lambda_{\mathbf{p}}}{r_{\mathbf{p}}})\|_{\mathbf{V}}^2; \tag{24}$$

- The **p**-subproblem: for fixed **N** and v

$$\min_{\mathbf{p}\in\mathbf{V}} \gamma \sum_e v_e^2 \|\mathbf{p}_e\| \operatorname{len}(e) + \frac{r_{\mathbf{p}}}{2} \|\mathbf{p} - (\nabla_{\mathcal{M}}\mathbf{N} - \frac{\lambda_{\mathbf{p}}}{r_{\mathbf{p}}})\|_{\mathbf{V}}^2; \tag{25}$$

- The v-subproblem: for fixed **p**

$$\min_{v\in V} \gamma \sum_e v_e^2 \|\mathbf{p}_e\| \operatorname{len}(e) + \beta(\epsilon \|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}). \tag{26}$$

The **N**-subproblem (24) can be solved by the same approximate strategy for solving (17). The Euler–Lagrange equation of (24) is given as follows

$$r_{\mathbf{p}}(\Delta_{\mathcal{M}}\mathbf{N}) - 2\alpha\mathbf{N} = \operatorname{div}_{\mathcal{M}}(r_{\mathbf{p}}\mathbf{p} + \lambda_{\mathbf{p}}) - 2\alpha\mathbf{N}^{in}. \tag{27}$$

This equation can be reformulated into a sparse linear system, which can be solved by various numerical packages. Then, we directly project the solution onto the unit sphere.

The **p**-subproblem (25) is easy to solve because it can be spatially decomposed, where the minimization problem with respect to each edge is performed individually. Thus, for each edge e, we only need to solve the following problem

$$\min_{\mathbf{p}_e} \gamma v_e^2 \|\mathbf{p}_e\| + \frac{r_{\mathbf{p}}}{2} \|\mathbf{p}_e - \left((\nabla_{\mathcal{M}}\mathbf{N})|_e - \frac{\lambda_{\mathbf{p}_e}}{r_{\mathbf{p}}}\right)\|^2,$$

which has a closed form solution

$$\mathbf{p}_e = \begin{cases} (1 - \frac{\gamma v_e^2}{r_{\mathbf{p}}\|\Psi_e\|})\Psi_e, & \|\Psi_e\| > \frac{\gamma v_e^2}{r_{\mathbf{p}}}, \\ 0, & \|\Psi_e\| \le \frac{\gamma v_e^2}{r_{\mathbf{p}}}, \end{cases} \tag{28}$$

where $\Psi = \nabla_{\mathcal{M}}\mathbf{N} - \frac{\lambda_{\mathbf{p}}}{r_{\mathbf{p}}}$.

The v-subproblem (26) is also a quadratic programming. The Euler–Lagrange equation of it can be written as

$$(2\gamma\|\mathbf{p}\| + \frac{\beta}{2\epsilon})v - 2\beta\epsilon(\Delta_{\mathcal{E}} v) = \frac{\beta}{2\epsilon}. \tag{29}$$

Similarly, Eq. (29) can be reformulated into a sparse linear system, which can be solved directly.

The entire procedure for solving (21) is outlined in Algorithm 2. The algorithm iteratively and alternatively solves the above three subproblems and updates the Lagrange multiplier.

Algorithm 2: Solving MSTV normal filtering model (21)

Initialization:
$\lambda_{\mathbf{p}}^0 = 0, \mathbf{N}^{-1} = 0, \mathbf{p}^{-1} = 0, v^{-1} = 0, k = 0, \varepsilon = 1e-6;$
repeat
  Solve **N**-sub problem
    For fixed $(\lambda_{\mathbf{p}}^k, \mathbf{p}^{k-1})$, compute $\mathbf{N}^k$ from (27) ;
    Normalize $\mathbf{N}^k$;
  Solve **p**-sub problem
    For fixed $(\lambda_{\mathbf{p}}^k, \mathbf{N}^k, v^{k-1})$, compute $\mathbf{p}^k$ from (28) ;
  Solve v-sub problem
    For fixed $\mathbf{p}^k$, compute v^k from (29) ;
  Update Lagrange multiplier
    $\lambda_{\mathbf{p}}^{k+1} = \lambda_{\mathbf{p}}^k + r_{\mathbf{p}}(\mathbf{p}^k - (\nabla_{\mathcal{M}}\mathbf{N})^k)$;
until $\|\mathbf{N}^k - \mathbf{N}^{k-1}\|_{\mathbf{U}} < \varepsilon$ or $k \ge 30$;
return $\mathbf{N}^k$.

4.3. Vertex updating scheme

After restoring the face normal field by the proposed normal filtering, vertex positions should be reconstructed to match the filtered face normals. As mentioned in previous works [13,16], the traditional vertex updating scheme proposed by Sun et al. [8] has the trianglewise orientation ambiguity problem. To address this problem, we reconstruct the denoised mesh using the vertex updating scheme proposed by Zhang et al. [16], which can prevent orientation ambiguity. The iteration number is empirically fixed as 30 in our experiments for producing well results. Since the vertex updating scheme is not our main contribution, we refer interested readers to the work [16] for further information.

4.4. AT versus MSTV normal filtering

It is necessary to discuss differences between AT (16) and MSTV (21) normal filtering, abbreviated as AT and MSTV.

Fig. 2 shows a comparison of these two methods against different levels of noise. As can be seen, when the noise level is moderate, both AT and MSTV can effectively remove noise and simultaneously preserve sharp features; see the first column of Fig. 2. Since MSTV uses ℓ_1 norm for its sparsity requirement, it tends to produce staircase effects over smoothly curved regions

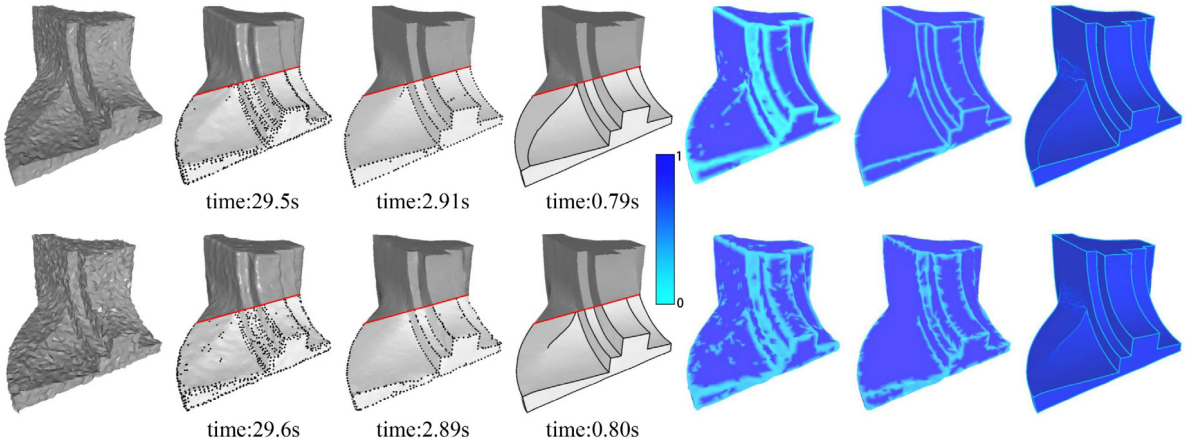

Fig. 3. Results yielded by discretizations in [35] and [14], and the proposed discretization. The first column shows Fandisk corrupted with $\sigma = 0.1\bar{l}_e$ and $\sigma = 0.2\bar{l}_e$. From left to right: input noisy surfaces, denoising (feature extraction) results produced by discretizations in [35] and [14] and the proposed discretization, and the corresponding discontinuity functions produced by these three discretizations.

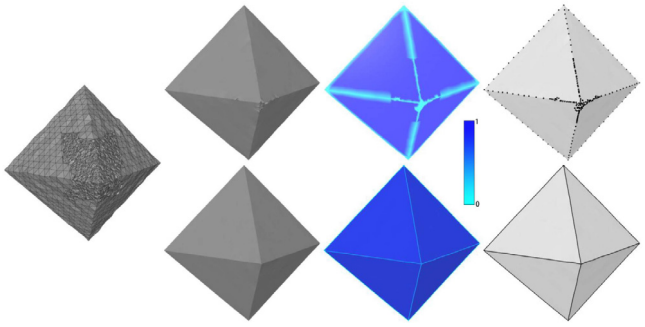

Fig. 4. Results yielded by discretization in [14] and the proposed discretization for irregular sampling mesh (corrupted with $\sigma = 0.2\bar{l}_e$). From left to right: input noisy surface, denoising results produced by discretization in [14] and the proposed discretization, the corresponding discontinuity functions, and the corresponding feature extraction results.

in all the noise levels. In contrast, AT generates better results in smooth regions. However, when the noise level increases, MSTV produces better results than those of AT; see the second and third columns of Fig. 2. In the case of high noise, MSTV still can effectively remove noise and preserve sharp features, whereas AT cannot; see the last column of Fig. 2.

In summary, MSTV is more robust against the noise and can produce cleaner sharp features, but enforces the denoised results toward the piecewise constant limit due to its ℓ_1 constraint on the gradient information. Although AT favors piecewise smooth solutions and gives desired results in the case of moderate noise, it blurs sharp features in some extent and leaves some bumps on the surface when the noise level is high. More examples about the discussion of AT and MSTV can be found in Section 5.

4.5. Comparisons to previous discretizations

To further illustrate the efficiency of our discretization, we compare it with those proposed by Tong and Tai [35] and Bonneel et al. [14]. To make the paper self-contained, we give a brief review of the discretizations in [35] and [14].

Tong and Tai [35] discretized the AT approximation via linear finite element method (FEM). Their discretization computes the normal field and discontinuity function vertex by vertex, which is suitable to deal with vertex-based problems. However, vertex normals are averaged from face normals and thus vertex normal filtering is less effective for mesh denoising. Moreover, the calculation of the discretization in [35] is complex and time-consuming, because trigonometric functions exist in calculation. Last but not least, an extra postprocessing is needed to connect located discontinuity vertices to form feature curves, which is not an easy task; see the feature curve extraction part in [35].

The discretization in [14] defined the signal on mesh faces, and defined the discontinuity function on mesh vertices. To simultaneously regularize them, Bonneel et al. [14] adopt a smoothing strategy for the discontinuity function (averaging the discontinuity values on two adjacent vertices, and assigning the average value onto the edge connecting these two vertices). On one hand, there is not a tight connecting between their regularizing processes of the signal and the discontinuity function. On the other hand, their smoothing strategy tends to blur the discontinuity function inevitably. Again, a postprocessing is needed to form feature curves.

In contrast, we discretize the AT approximation via two tightly coupled function spaces: one is the piecewise constant function space, in which the face normal field is smoothed; the other is the edge function space, in which the calculation of the discontinuity function is surprisingly simple yet effective. We list the advantages of our discretization as follows:

1. Our discretization calculates the discontinuity function at mesh edges directly. It can produce better denoised results, and make located discontinuity edges more exactly distribute on geometric features of the underlying surface. See Fig. 3 for example. Besides, our discretization can directly extract feature curves from located discontinuity edges, whereas the discretizations in [35] and [14] cannot.
2. Our discretization has simpler formulation and higher efficiency in practice. Compared with the discretization in [35] based on linear FEM and that in [14] based on DEC, our discretization is simple in calculating by using the operators in the proposed function spaces. As can be seen in Fig. 3, the CPU costs of our discretization are dramatically decreased, compared to those of the discretizations in [35] and [14].
3. Our discretization is robust against non-uniform surface sampling. The proposed operators used in our discretization are strictly-defined by finite element representation in numerical PDE. In contrast, the discretization in [14] is calculated irrespective of geometric measure of the underlying surface that makes it sensitive to surface sampling. As we can see in Fig. 4, our discretization outperforms that in [14], especially in irregular sampling regions.

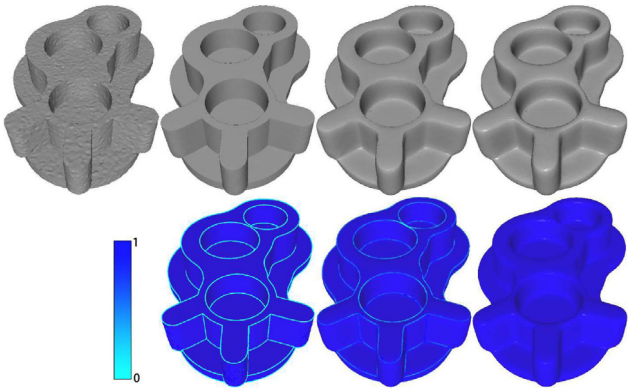

Fig. 5. Denoising results and the corresponding discontinuity functions for different β. From left to right: noisy mesh (corrupted with $\sigma = 0.1\bar{l}_e$) and results with different β. The top row shows noisy mesh and denoising results, while the bottom row illustrates the corresponding discontinuity map v.

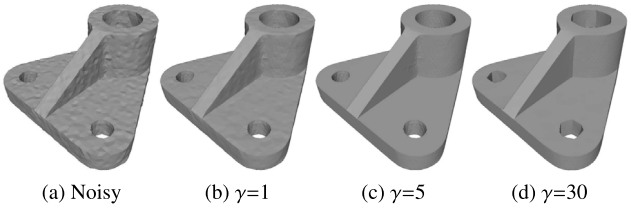

Fig. 6. Denoising results for different γ. From left to right: noisy mesh (corrupted with $\sigma = 0.1\bar{l}_e$) and results with different γ.

5. Experimental results and comparisons

We evaluate our mesh denoising methods on a variety of surfaces including CAD, non-CAD, and real scanned meshes captured by the laser scanner and Kinect sensors. The tested meshes are corrupted by either synthetic or raw noise. The synthetic noise is added in random directions, and is generated by a zero-mean Gaussian function with standard deviation (σ) proportional to the mean edge length ($\bar{l}_e$). The mesh sizes of the tested surfaces in this section are listed in Table 1. We compare our mesh denoising methods, abbreviated as AT and MSTV, with the state-of-the-art methods including TV normal filtering [10], ℓ_0 minimization [18], robust and high fidelity mesh denoising [13], and bilateral normal filtering [9], abbreviated as TV, ℓ_0, RHM, and BF, respectively. We have implemented all the methods tested in the paper (except Yadav et al. [13]) by using C++. All the examples are run on a PC with an Intel i7 dual core 2.6 GHz processor and 8 GB RAM; all the meshes are rendered in a flat-shading model for showing faceting effect.

5.1. Parameters tuning

As we know, most mesh denoising methods have parameters, which need to be manually tuned. Both our AT (16) and MSTV (21) normal filtering models have three parameters: α, β, and γ. These parameters have different roles, and need to be tuned to generate satisfactory results. Because the parameter influences on these two models are similar, we just discuss the parameters of AT normal filtering.

α is introduced to prevent the solution deviating far from the input. For producing satisfactory results, α is suggested with smaller values for CAD meshes and with greater values for non-CAD and scanned meshes. When tuning parameters, we first set α in a suggested range with lower values for higher level of noise, and then tune the other two parameters.

β controls the penalization of total length of located discontinuity edges. Fig. 5 shows results with different β. As we can see, with the increasing of β, the length of discontinuity edges is decreasing. As mentioned before, the processes of denoising and locating discontinuities are complementary. Thus, the decreasing of the length of discontinuity edges will influence denoising results (blurring sharp features); see Figs. 5c and 5d for example.

γ influences the smoothness of the denoised result, which increases with the noise level. If γ is too small, noise cannot be effectively removed; see Fig. 6b. On the contrary, too large γ will flatten smooth regions and generate false edges; see Fig. 6d.

5.2. Qualitative comparisons

We visually compare our methods AT and MSTV with the state-of-the-art methods including TV, ℓ_0, BF, and RHM, respectively. The parameters in all the tested methods are elaborately tuned for generating visually best results.

Denoise CAD surfaces. In Fig. 7, we demonstrate and compare results for a CAD mesh containing sharp features and smoothly curved regions. The mesh is corrupted by moderate noise. It can be seen that, except BF, all the other methods can recover sharp features at this noise level. TV, ℓ_0, and MSTV preserve sharp features well, but suffer from unnatural staircase effects in smoothly curved regions; see Figs. 7b, 7c, and 7g. This annoying visual artifact is more serious for ℓ_0 for its high sparsity requirement. As we can see, ℓ_0 even produces some false edges in smooth regions. In contrast, both RHM and our method AT preserve sharp features and simultaneously recover smooth regions well. However, due

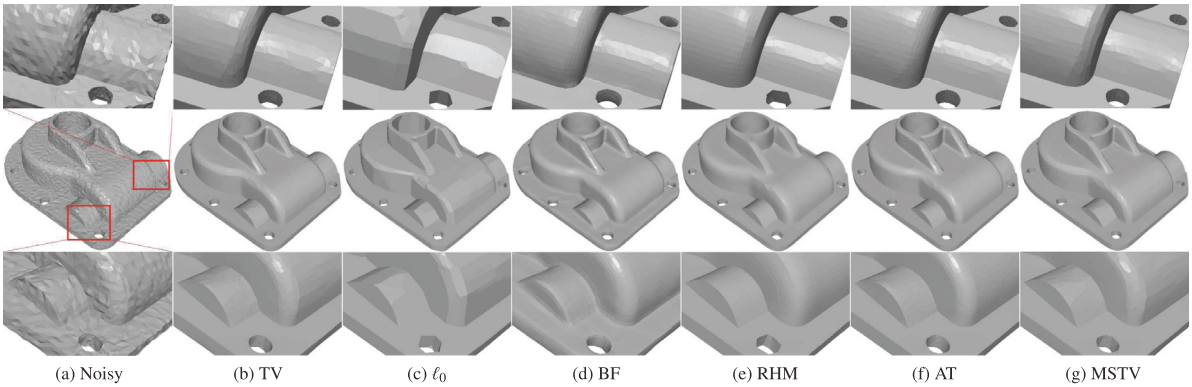

Fig. 7. Denoising results of Casting, corrupted with $\sigma = 0.15\bar{l}_e$. From left to right: noisy mesh, denoising results produced by TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV, respectively. The top and bottom rows are zoomed-in views.

Table 1
Mesh sizes of surfaces tested in Section 5.

Model	Casting	Fandisk	Vase	Dodecahedron	Prism	Bunny-iH	Gargoyle	Angel	Embossment	Pyramid	Big-Girl	Block	Child
$\|V\|$	18.4 K	6.5 K	3.8 K	4.6 K	4.5 K	35.2 K	25.0 K	24.6 K	66.0 K	35.3 K	46.1 K	8.7 K	49.3 K
$\|F\|$	36.9 K	12.9 K	7.7 K	9.1 K	9.1 K	70.5 K	50.0 K	48.1 K	129.1 K	69.6 K	91.1 K	17.5 K	98.6 K

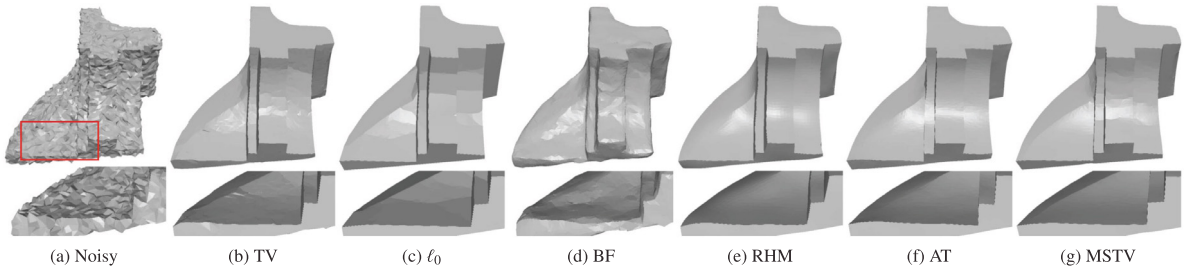

Fig. 8. Denoising results of Fandisk, corrupted with $\sigma = 0.5\bar{l}_e$. From left to right: noisy mesh, denoising results produced by TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV, respectively. The second row shows zoomed-in views.

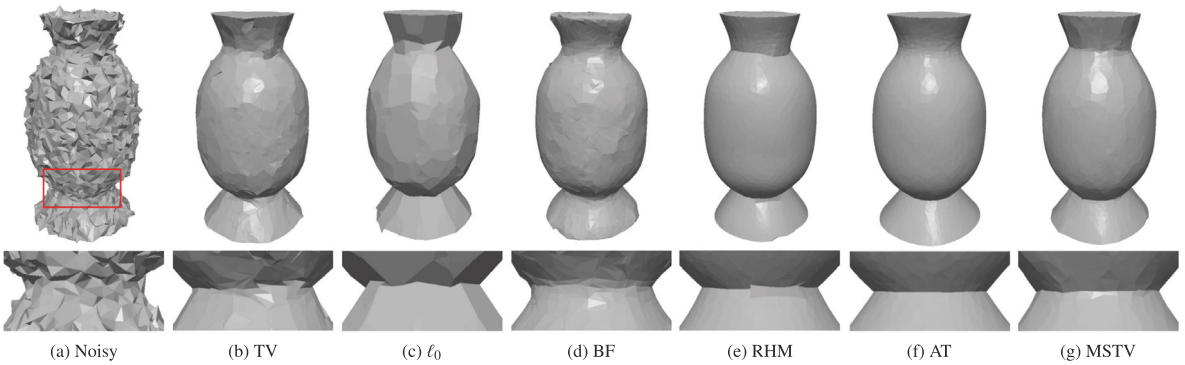

Fig. 9. Denoising results of Vase, corrupted with $\sigma = 0.6\bar{l}_e$. From left to right: noisy mesh, denoising results produced by TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV, respectively. The second row shows zoomed-in views.

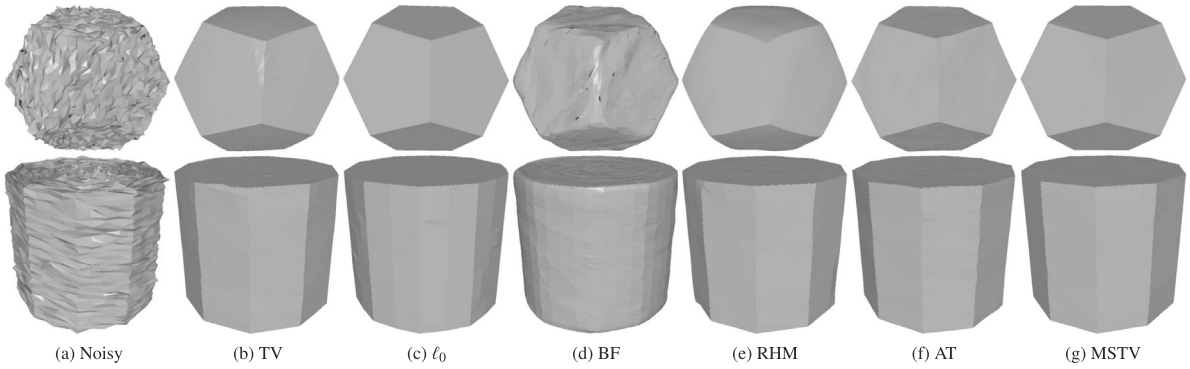

Fig. 10. Denoising results of Dodecahedron and Prism, corrupted with $\sigma = 0.5\bar{l}_e$ and $\sigma = 0.3\bar{l}_e$. From left to right: noisy meshes, denoising results produced by TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV, respectively.

to the using of robust error norm, RHM oversmoothes small-scale features and sharpens some curved features; see the zoomed-in view of Fig. 7e. Thus, in the case of moderate noise, AT outperforms the other methods in dealing with the surface containing sharp features and curved surface characteristics.

Fig. 8 shows comparisons of a CAD mesh corrupted by considerable amount of noise. We note that, although MSTV induces slight staircase effects, it is superior in recovering geometric features with respect to the compared methods. Moreover, both AT and MSTV produce clearer feature-preserving results over those of the state-of-the-art methods. Compared to TV, MSTV reduces the staircase effects appeared in smooth regions, and is more robust for preserving features; see the artifacts of the result produced by TV in Fig. 8b. ℓ_0 preserves sharp features well, but flattens smooth regions and induces false edges inevitably. Although RHM does a good job on smooth regions, it oversmoothes the weak feature of Fandisk; see Fig. 8e. Thus, when the noise level is increasing, our method AT and MSTV can yield promising feature-preserving results.

Denoise piecewise smooth and constant surfaces. Fig. 9 shows that AT outperforms the other methods evidently for the surface with piecewise smooth priors. In contrast, TV, ℓ_0, and MSTV surfer from the staircase effects in varying degrees, especially for ℓ_0. We also note that, compared to TV, our method MSTV has less staircase effects in smooth regions, and produces more faithful feature-preserving result; see Figs. 9b and 9g.

In Fig. 10, we show results for meshes containing only flat regions and sharp features. As can be seen, TV, RHM, and AT

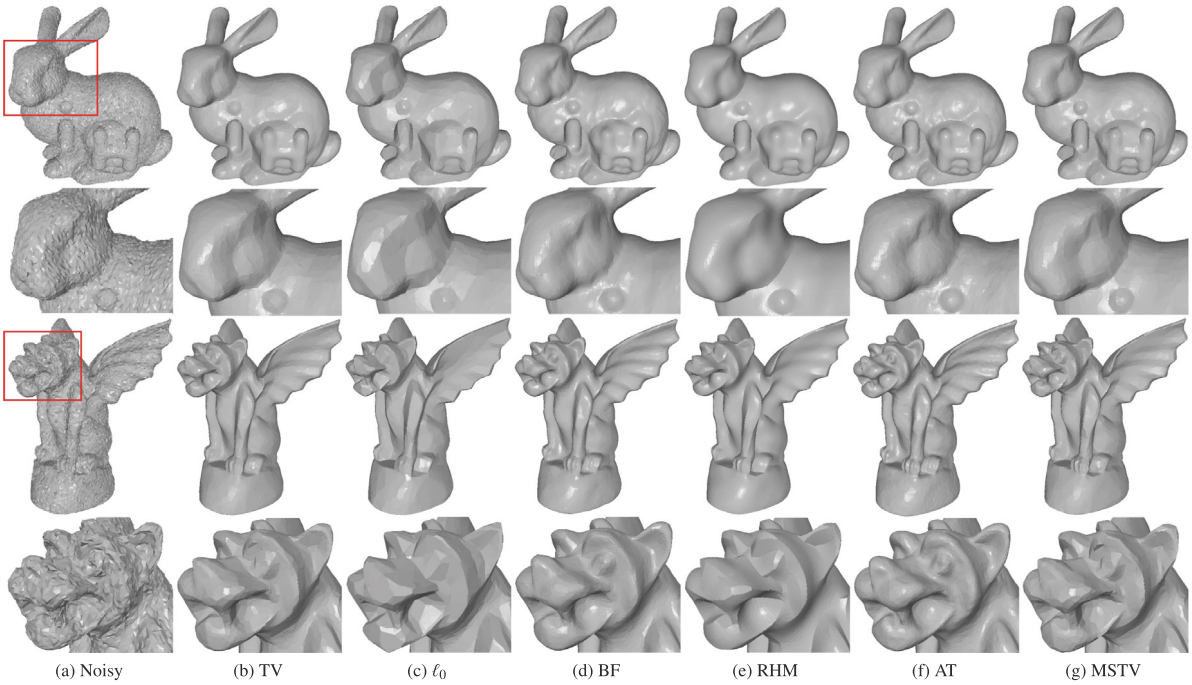

Fig. 11. Denoising results of Bunny-iH and Gargoyle, corrupted with $\sigma = 0.3\bar{l}_e$. From left to right: noisy meshes, denoising results produced by TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV, respectively. The even rows are zoomed-in views.

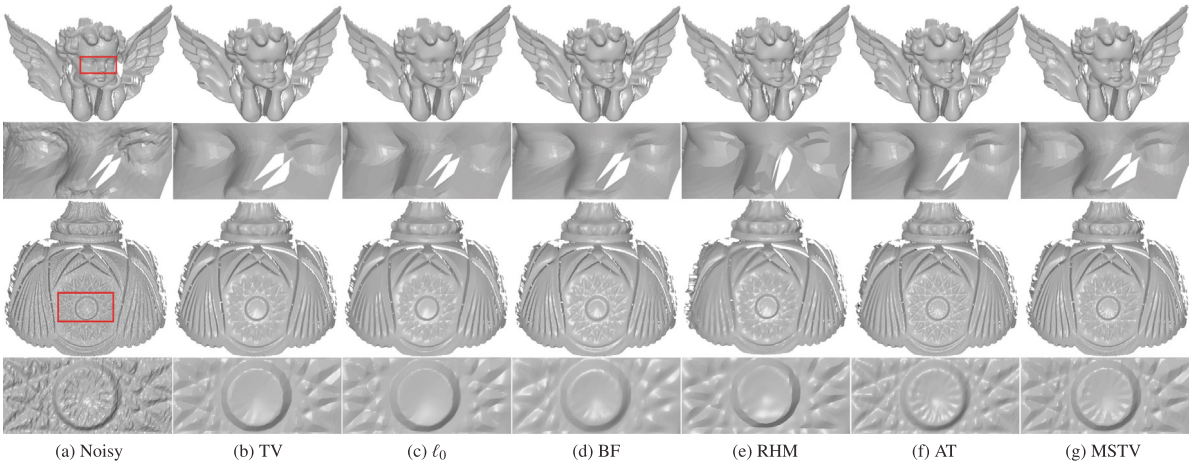

Fig. 12. Denoising results of real scanned meshes acquired by laser scanners. From left to right: noisy meshes, denoising results produced by TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV, respectively. The even rows are zoomed-in views.

produce wavy edges more or less, especially for RHM. Although ℓ_0 preserves sharp features well, it tends to introduce spurious edges over the surface; see the second row of Fig. 10c. In contrast, our method MSTV leads to results be more faithful to the underlying surface with piecewise constant priors.

Denoise non-CAD surfaces. Fig. 11 gives comparisons on non-CAD meshes with different levels of features. As can be seen, all the methods can remove the noise effectively. Nevertheless, TV, ℓ_0, and MSTV flatten some details at small scales, and ℓ_0 makes this situation even worse; see the zoomed-in views of Figs. 11b, 11c, and 11g. RHM oversmoothes small-scale features evidently and tends to enhance smoothly curved features; see the head region of Gargoyle in Fig. 11e. In contrast, BF and our method AT perform better than the other methods. Thanks to the discontinuity function v of our method, AT can preserve geometric features at various sizes, while BF may lose some finer details; see the eye region of Gargoyle in Figs. 11d and 11f.

Denoise scanning surfaces. Fig. 12 shows denoising results for the real scanning data acquired by laser scanners. Similar results can be observed as those of the non-CAD meshes. Although all the tested methods can remove noise and recover geometric features to some extent, our methods AT and MSTV preserve fine details better than the other methods; see the zoomed-in views of Fig. 12. Again, although MSTV preserves geometric features well, it suffers a little staircase effects. We also testify effectiveness of our methods on scanning data acquired by Kinect sensors. These scanned meshes have been provided by Wang et al. [25]. As we can see in Fig. 13, except BF and RHM which leave some bumps in the results, all the other methods can remove the noise. ℓ_0 may sharpen curved features and flatten some smooth regions;

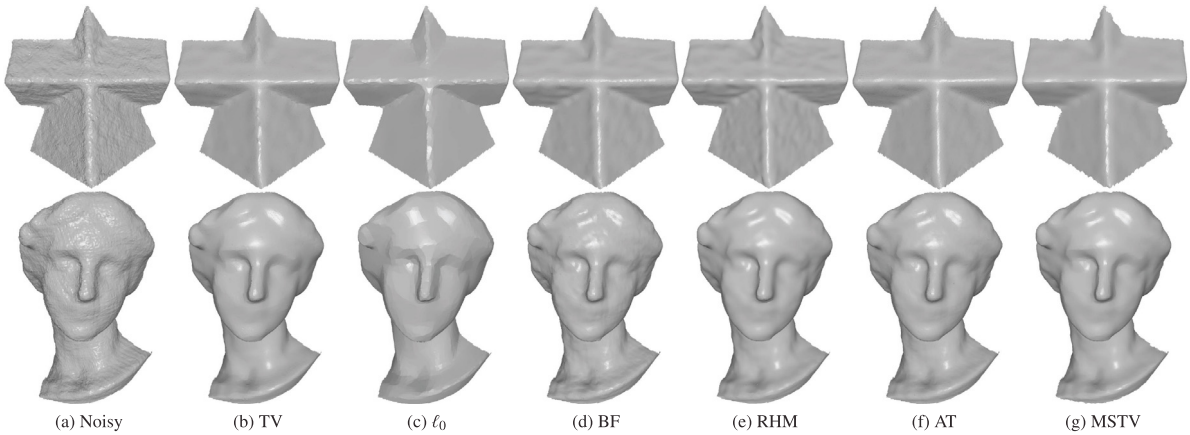

Fig. 13. Denoising results of Pyramid and Big-Girl captured by Kinect sensors. From left to right: noisy meshes, denoising results produced by TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV, respectively.

Table 2
Quantitative comparisons for the tested denoising methods including TV [10], ℓ_0 [18], BF [9], RHM [13], and our methods AT and MSTV.

Methods	MASE ($\times 10^{-3}$), $E_{v,2}$($\times 10^{-3}$); Time (in Seconds)								
	Casting	Fandisk	Vase	Dodecahedron	Prism	Bunny-iH	Gargoyle	Pyramid	Big-Girl
TV	8.37,1.51; 1.48	19.3,3.90; 0.43	27.5,6.20; 0.26	15.6,4.62; 0.29	1.63,3.04; 0.29	21.6,1.78; 3.10	37.3,2.68; 2.29	63.9,14.5; 1.78	44.7,5.67; 1.79
ℓ_0	15.6,2.51; 15.9	46.1,5.54; 26.0	49.1,14.7; 13.3	10.5,2.34; 46.1	1.06,2.42; 8.48	55.9,2.83; 39.9	48.1,3.69; 31.2	63.4,**11.7**; 38.8	44.1,5.85; 57.0
BF	11.1,1.29; 0.75	58.0,8.78; 0.20	34.9,9.06; 0.10	49.4,5.76; 0.14	23.7,5.56; 0.11	18.6,1.62; 1.51	21.9,2.20; 0.83	58.4,15.4; 1.02	47.9,5.80; 1.46
RHM	13.8,1.45; 12.6	21.3,3.79; 3.70	20.3,5.69; 2.14	16.6,8.32; 2.83	10.5,7.21; 5.33	34.3,2.12; 23.4	45.8,2.54; 21.6	54.8,13.4; 18.7	49.7,5.91; 25.1
AT	**5.53**,**0.99**; 2.32	16.7,3.27; 0.91	**14.5**,5.89; 0.63	12.6,3.64; 0.71	1.61,5.31; 0.60	**17.1**,**1.21**; 6.31	**18.3**,**1.71**; 4.67	58.2,15.7; 7.64	42.1,5.65; 8.12
MSTV	8.07,1.50; 2.75	**15.6**,**2.74**; 1.09	18.6,**5.62**; 0.71	**9.26**,**2.06**; 0.80	**0.84**,**2.13**; 0.74	20.5,1.65; 7.14	33.2,2.21; 5.63	**54.1**,16.3; 9.23	**41.5**,**5.58**; 11.0

see Fig. 13c. AT, MSTV, and TV can produce appealing results. However, from numerical metrics (which will be introduced in Section 5.3), we find that numerical errors of MSTV are always lower than those of TV and AT. This shows that MSTV preserves features better than TV and AT for scanning data.

The above qualitative comparisons show that, our methods AT and MSTV produce better denoised results and recover more faithful geometric features, especially in the case of high noise.

5.3. Quantitative comparisons

In order to show more objective comparisons, we further assess the performances of the tested methods quantitatively. Specifically, we use MSAE to measure the mean square angular error between face normals of the denoised result and those of the corresponding clean surface, and use $E_{v,2}$ to measure the vertex deviation of the denoised result from the underlying clean surface. These two error metrics are widely suggested in previous works [9,10,13,15,20]. The evaluation results on these two metrics are listed in Table 2.

As shown in Table 2, both AT and MSTV consistently outperform the compared methods in terms of MSAE values. Specifically, for CAD meshes, AT outperforms the other ones when the noise level is moderate, whereas MSTV achieves the lowest MSAE value in the case of high noise. AT performs well on the surface with piecewise smooth priors, while MSTV does a good job on that with piecewise constant priors. AT produces the lowest MSAE values for non-CAD meshes with fine details, while MSTV performs favorably against the other methods for scanning data. Moreover, in most cases, one of our methods has least errors in the sense of $E_{v,2}$. This fact shows that the results produced by our methods are more faithful to the underlying surfaces, compared to those produced by the state-of-the-art methods. Accordingly, the quantitative evaluations are consistent with the visual comparisons, which show the superiority of our methods AT and MSTV.

Table 3
Quantitative comparisons for the methods including non-local based method [21], learning-based method [25], and our methods AT and MSTV.

Meshes	MASE ($\times 10^{-3}$), $E_{v,2}$($\times 10^{-3}$); Time (in Seconds)			
	Non-local	Learning	AT	MSTV
Block	123, 5.83; 8.16	81.9, 1.82; 0.94	**5.39**, **1.73**; 1.11	6.78, 2.14; 1.28
Child	51.6, **0.82**; 20.4	64.1, 1.08.; 3.74	**22.8**, 0.85; 7.06	25.0, 0.88; 8.13

The CPU costs of all the tested methods are also recorded in Table 2. As can be seen, BF is the fastest method, while ℓ_0 is the slowest one. Although RHM is some faster than ℓ_0, it is still more computationally intensive than the other methods. Our methods are a little slower than BF and TV, but much faster than ℓ_0 and RHM. MSTV is more computationally intensive than AT, because MSTV needs to solve the ℓ_1 related problem. As a result, in most cases, our methods produce better denoising results at reasonable CPU costs.

5.4. Comparisons to non-local and learning-based methods

To further demonstrate the robustness and effectiveness of our Mumford–Shah framework, we compare our methods AT and MSTV with the non-local based method [21] and the learning-based method [25] visually and quantitatively. As we can see in Fig. 14, for the CAD mesh, the non-local based method oversmoothes sharp features, and the learning-based method induces slight artifacts in sharp features. In contrast, our methods AT and MSTV preserve sharp features well. For the non-CAD mesh with different levels of features, the non-local based method and AT produce visually better results. The learning-based method smoothes weak features, and MSTV flattens some smooth features. As can be seen in Table 3, for both CAD and non-CAD meshes, our methods outperform the compared two methods in terms of MSAE values. The non-local based method has the lowest $E_{v,2}$ value for the non-CAD mesh. We also record the CPU costs of

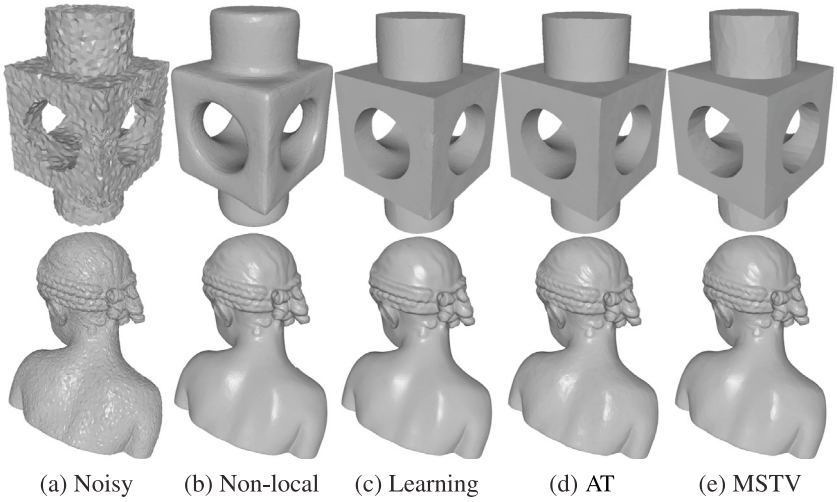

Fig. 14. Denoising results of Block and Child, corrupted with $\sigma = 0.3\bar{l}_e$ and $\sigma = 0.2\bar{l}_e$. From left to right: noisy meshes, denoising results produced by non-local based method [21], learning-based method [25], and our methods AT and MSTV, respectively.

these four methods in Table 3. As we can see, the non-local based method is the slowest method, while the learning-based method is the fastest one.

6. Conclusion

In this paper, we introduce Mumford–Shah-based regularization models for mesh denoising. A novel numerical method is proposed to discretize MS-based functionals over triangulated surfaces. The new discretizations directly diffuse the discontinuity function on mesh edges and thus avoid the postprocessing to form feature curves. Compared to existing discretizations, the proposed discretizations have simpler formulation, and are more robust for preserving and locating geometric features especially in the high noise case. Intensive experimental results on a variety of surfaces show the robustness and effectiveness of our mesh denoising methods.

Some future works are left open. The proposed MS framework can be extended to more applications over triangulated surfaces, such as feature detection, segmentation, completion, etc. Moreover, it would be interesting and challenging to extend this work on point clouds.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank anonymous reviewers for their constructive suggestions for improving the manuscript. Zheng Liu's research is supported by National Natural Science Foundation of China (No. 61702467). Weina Wang's research is supported by Zhejiang Provincial Natural Science Foundation of China (No. LQ20A010007). Weiming Wang's research is supported by National Natural Science Foundation of China (Nos. 61702079, 61976040).

Appendix

Lemma 1. *The adjoint operator of* $\mathrm{div}_{\mathcal{E}} : W \to V, w \mapsto \mathrm{div}_{\mathcal{E}} w$ *has the following form:*

$$(\mathrm{div}_{\mathcal{E}} w)|_e = -\frac{1}{\mathrm{len}(e)} \sum_{l \in H(e)} w_l \mathrm{sgn}(e,l)\mathrm{len}(l), \quad \forall e.$$

Proof. As the definition of the adjoint operator, we have

$$\langle \nabla_{\mathcal{E}} v, w\rangle_W = \langle v, -\mathrm{div}_{\mathcal{E}} w\rangle_V. \tag{30}$$

By using the inner products (13) and (7) in W and V, (30) can be reformulated as

$$\sum_l (\nabla_{\mathcal{E}} v)|_l w_l \mathrm{len}(l) = \sum_e v_e(-\mathrm{div}_{\mathcal{E}} w)|_e \mathrm{len}(e). \tag{31}$$

By using the definition of the gradient operator (12), the left-hand side of (31) is actually

$$\begin{aligned}\sum_l (\nabla_{\mathcal{E}} v)|_l w_l \mathrm{len}(l) &= \sum_l [v]_l w_l \mathrm{len}(l)\\ &= \sum_l (v_e \mathrm{sgn}(e,l)) w_l \mathrm{len}(l)\\ &= \sum_e v_e \sum_{l\in H(e)} \mathrm{sgn}(e,l) w_l \mathrm{len}(l)\end{aligned}$$

Therefore, we have

$$\sum_e v_e \sum_{l\in H(e)} \mathrm{sgn}(e,l) w_l \mathrm{len}(l) = \sum_e v_e(-\mathrm{div}_{\mathcal{E}} w)|_e \mathrm{len}(e).$$

Then, the assertion follows immediately. □

References

[1] Wang J, Zhang X, Yu Z. A cascaded approach for feature-preserving surface mesh denoising. Comput-Aided Des 2012;44(7):597–610.
[2] Taubin G. A signal processing approach to fair surface design. In: Proc. 22nd Annu. Conf. Comput. Graph. Interactive Tech. 1995, p. 351–58.
[3] Desbrun M, Meyer M, Schröder P, Barr A-H. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. 26th Annu. Conf. Comput. Graph. Interactive Tech. 1999, p. 317–24.

[4] Yagou H, Ohtake Y, Belyaev A. Mesh smoothing via mean and median filtering applied to face normals. In: Proc. Geometric Modeling Process. 2002, p. 124–31.
[5] Tasdizen T, Whitaker R, Burchard P, Osher S. Geometric surface smoothing via anisotropic diffusion of normals. In: Proc. Conf. visualization. 2002, p. 125–32.
[6] Bajaj C, Xu G. Anisotropic diffusion of surfaces and functions on surfaces. ACM Trans Graph 2003;22(1):4–32.
[7] Wang C. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. IEEE Trans Vis Comput Graphics 2006;12(4):629–39.
[8] Sun X, Rosin P, Martin R, Langbein F. Fast and effective feature-preserving mesh denoising. IEEE Trans Vis Comput Graphics 2007;13(5):925–38.
[9] Zheng Y, Fu H, Au KC, Tai CL. Bilateral normal filtering for mesh denoising. IEEE Trans Vis Comput Graphics 2011;17(10):1521–30.
[10] Zhang H, Wu C, Zhang J, Deng J. Variational mesh denoising using total variation and piecewise constant function space. IEEE Trans Vis Comput Graphics 2015;21(7):873–86.
[11] Lu X, Chen W, Schaefer S. Robust mesh denoising via vertex pre-filtering and L_1-median normal filtering. Comput Aided Geom Design 2019;114:133–42.
[12] Wei M, Liang L, Pang WM, Wang J, Li W, Wu H. Tensor voting guided mesh denoising. IEEE Trans Autom Sci Eng 2017;14(2):931–45.
[13] Yadav S, Reitebuch U, Polthier K. Robust and high fidelity mesh denoising. IEEE Trans Vis Comput Graphics 2019;25(6):2304–10.
[14] Bonneel N, Coeurjolly D, Gueth P, Lachaud J-O. Mumford-Shah mesh processing using the Ambrosio-Tortorelli functional. Comput Graph Forum (Proc Pac Graph) 2018;37(7):75–85.
[15] Liu Z, Lai R, Zhang H, Wu C. Triangulated surface denoising using high order regularization with dynamic weights. SIAM J Sci Comput 2019;41(1):1–26.
[16] Zhang J, Deng B, Hong Y, Peng Y, Qin W, Liu L. Static/dynamic filtering for mesh geometry. IEEE Trans Vis Comput Graphics 2019;25(4):1774–87.
[17] Zhang W, Deng B, Zhang J, Bouaziz S, Liu L. Guided mesh normal filtering. Comput Graph Forum (Proc Pac Graph) 2015;34(7):23–34.
[18] He L, Schaefer S. Mesh denoising via L_0 minimization. ACM Trans Graph 2013;32(4):1–8.
[19] Zhao Y, Qin H, Zeng X, Xu J, Dong J. Robust and effective mesh denoising using L_0 sparse regularization. Comput-Aided Des 2018;101:82–97.
[20] Liu Z, Zhong S, Xie Z, Wang W. A novel anisotropic second order regularization for mesh denoising. Comput Aided Geom Des (Proc Geom Model Process) 2019;71:190–201.
[21] Li X, Zhu L, Fu C-W, Heng P-A. Non-local low-rank normal filtering for mesh denoising. Comput Graph Forum (Proc Pac Graph) 2018;37(7):155–66.
[22] Wei M, Guo X, Huang J, Wang F, Xie H, Kwan R, Qin J. Mesh defiltering via cascaded geometry recovery. Comput Graph Forum 2019;38(7):591–605.
[23] Chen H, Huang J, Remil O, Xie H, Qin J, Guo Y, Wei M, Wang J. Structure-guided shape-preserving mesh texture smoothing via joint low-rank matrix recovery. Comput-Aided Des (Proc Solid Phys Model) 2019;115:122–34.
[24] Diebel JR, Thrun S, Brünig M. A Bayesian method for probable surface reconstruction and decimation. ACM Trans Graph 2006;25(1):39–59.
[25] Wang P-S, Liu Y, Tong X. Mesh denoising via Cascaded normal regression. ACM Trans Graph 2016;35(6):232:1–232:12.
[26] Wang J, Huang J, Wang FL, Wei M, Xie H, Qin J. Data-driven geometry-recovering mesh denoising. Comput-Aided Des (Proc Solid Phys Model) 2019;114:133–42.
[27] Mumford D, Shah J. Optimal approximations by piecewise smooth functions and associated variational problems. Comm. Pure and Applied Math. 1989;42(5):577–684.
[28] Brook A, Kimmel R, Sochen N. Variational restoration and edge detection for color images. J Math Imaging Vision 2003;18(3):247–68.
[29] Bar L, Brook A, Sochen N, Kiryati N. Deblurring of color images corrupted by impulsive noise. IEEE Trans Image Process 2007;16(4):1101–11.
[30] Jung M, Bresson X, Chan TF, Vese LA. Nonlocal Mumford-Shah regularizers for color image restoration. IEEE Trans Image Process 2011;20(6):1583–98.
[31] Chambolle A. Image segmentation by variational methods: Mumford and Shah functional and the discrete approximations. SIAM J Appl Math 1995;55(3):827–63.
[32] Vese LA, Chan TF. A multiphase level set framework for image segmentation using the Mumford and Shah model. Int J Comput Vis 2002;50(3):271–93.
[33] Chan TF, Esedoglu S, Nikolova M. Algorithms for finding global minimizers of image segmentation and denoising models. SIAM J Appl Math 2006;66(5):1632–48.
[34] Ambrosio L, Tortorelli VM. Approximation of functional depending on jumps by elliptic functional via Γ-convergence. Comm Pure Appl Math 1990;43(8):999–1036.
[35] Tong W, Tai XC. A variational approach for detecting feature lines on meshes. J Comput Math 2016;34(1):87–112.
[36] Coeurjolly D, Foare M, Gueth P, Lachaud J-O. Piecewise smooth reconstruction of normal vector field on digital data. Comput Graph Forum (Proc Pac Graph) 2016;35(7):157–67.
[37] Shah J. A common framework for curve evolution, segmentation and anisotropic diffusion. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. 1996, p. 136–42.
[38] Alicandro R, Braides A, Shah J. Free-discontinuity problems via functionals involving the L_1-norm of the gradient and their approximations. Interfaces Free Bound 1999;1(1):17–37.
[39] Rami B-A, Nir S. Stereo matching with Mumford-Shah regularization and occlusion handling. IEEE Trans Pattern Anal Mach Intell 2010;32(11):2071–84.
[40] Liu Z, Zhang H, Wu C. On geodesic curvature flow with level set formulation over triangulated surfaces. J Sci Comput 2017;70(2):631–61.
[41] Wu C, Tai XC. Augmented lagrangian method, dual methods, and split bregman iteration for ROF, vectorial TV, and high order models. SIAM J Imaging Sci 2010;3(3):300–39.