Computers & Graphics 106 (2022) 222–236

Contents lists available at ScienceDirect

Computers & Graphics

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

Technical Section

Feature-preserving Mumford–Shah mesh processing via nonsmooth nonconvex regularization[☆]

Chunxue Wang[a,b], Zheng Liu[c,*], Ligang Liu[d]

[a] Dunhuang Academy, Dunhuang, 736200, China
[b] Gansu Provincial Research Center for Conservation of Dunhang Cultural Heritage, Dunhuang, 736200, China
[c] School of Computer Science, China University of Geosciences, Wuhan, 430074, China
[d] School of Mathematical Sciences, University of Science and Technology of China, Hefei, 230026, China

ARTICLE INFO

Article history:
Received 23 February 2022
Received in revised form 9 June 2022
Accepted 10 June 2022
Available online 16 June 2022

Keywords:
Mumford–Shah functional
Nonsmooth nonconvex regularization
Discontinuity function
Alternating minimization scheme
Feature-preserving

ABSTRACT

Motivated by the success in image processing, the Mumford–Shah functional has attracted extensive attentions in geometry processing. Existing methods, mainly focusing on discretizations on the triangulated mesh, either over-smooth sharp features or are sensitive to noises or outliers. In this paper, we first introduce a nonsmooth nonconvex Mumford–Shah model for a feature-preserving filtering of face normal field to ameliorate the staircasing artifacts that appear in the original Mumford–Shah total variation (MSTV) and develop an alternating minimization scheme based on alternating direction method of multipliers to realize the proposed model. After restoring the face normal field, vertex updating is then employed by incorporating the oriented normal constraints and discontinuities to achieve a detail-preserving reconstruction of mesh geometry. Extensive experimental results demonstrate the effectiveness of the above shape optimization routine for various geometry processing applications such as mesh denoising, mesh inpainting and mesh segmentation.

1. Introduction

As a well-known mathematical tool, the Mumford–Shah functional (MS) has been proven successful in image processing [1,2] since it was first formally proposed for image segmentation [3]. The most distinguishing feature of minimizing this functional is that both a piecewise-smooth image and the set of discontinuities are obtained at the same time where the former is suitable for applications such as image restoration and the latter helps to develop a high-quality feature detection. This observation motivates us to go in for further studies of MS functional in geometry processing.

There are several challenges to construct a appropriate approximations of MS function on 3D mesh. Firstly, as meshes are irregular in connectivity and sampling, it is difficult to apply efficient numerical algorithms of regular grids to large unstructured meshes directly. Secondly, the MS formulation is strongly dependent on the discretization of differential operators, which determines where both piecewise-smooth function and discontinuity function are well-defined. Thirdly, it is not clear that what the piecewise-smooth function and discontinuities represent for an individual application of geometry processing.

Fortunately, recent advances have been made on above issues in geometric processing, especially in extending classic approximations of the MS functional such as Ambrosio–Tortorelli functional [4] (AT) and Mumford–Shah total variation functional (MSTV) [5] from image processing to geometry processing. An outstanding work is that Zhang et al. [6] successfully extended a convexified version of MS model for mesh segmentation, constraining the boundary between different segments to be as short as possible, but it is not suitable for our model since we are focusing on how to filter normal field with the MS functional. In addition, various discretization schemes on 3D signals are discussed. More specifically, Pokrass et al. [7] defined both membership function and phase field (discontinuities) on vertices. An alternative finite element discretization was proposed by Tong and Tai [8] and discrete exterior calculus discretization was designed in [9,10], which are two similar pointwise schemes to solve AT approximation on 3D triangulated mesh since the discontinuities of both schemes are defined on vertices. In order to void mismatch between the underlying geometric features and pointwise features, Liu et al. [11] solved MSTV approximation by defining discontinuity function on mesh edges, which yields better denoised results and locates geometric discontinuities more accurately. However, this method tends to suppress fine details due to the using of TV term in model.

In this paper, we propose a nonsmooth nonconvex version of MSTV approximation for normal filtering and formulate a set of

[☆] This article was recommended for publication by S. Hahmann.
[*] Corresponding author.
E-mail address: liu.zheng.jojo@gmail.com (Z. Liu).

https://doi.org/10.1016/j.cag.2022.06.006
0097-8493/© 2022 Elsevier Ltd. All rights reserved.

classical mesh processing problems. The main contributions of this work include:

- We present a normal filter using nonsmooth nonconvex Mumford–Shah regularization which is an extension of the work of Liu et al. [11], and develop an alternating minimization algorithm based on alternating direction method of multipliers (ADMM) which recovers high quality of face normals with neat features and locates discontinuities of the surface simultaneously.
- We propose a new method for vertex updating by incorporating the oriented normal constraints and the discontinuity function optimized from normal filtering, which deforms the geometry gradually to match the resulted mesh with target normals while preserving sharp edges.
- We demonstrate the superiority of our approach visually and numerically on several applications of mesh processing, including mesh denoising, mesh inpainting and mesh segmentation.

The rest of the paper is organized as follows. We review related work in Section 2 and define some basic function spaces and associated differential operators in Section 3. The nonsmooth nonconvex Mumford–Shah model for normal filtering and the vertex updating scheme are presented in Section 4, followed by an analysis and discussion in Section 5. A number of applications in geometry processing are illustrated in Section 6. Finally, the conclusion and future work are discussed in Section 7.

2. Related work

2.1. Mumford–Shah functional and its Γ-convergence approximations

In 1989, Mumford and Shah proposed a functional to approximate an input image in terms of a piecewise-smooth function [3]. For a scalar image $u : \Omega \to \mathbb{R}$ with its discontinuity set $C \subset \Omega \subset \mathbb{R}^2$, the Mumford–Shah functional is defined as:

$$E^{MS}(u, C) = \gamma \int_{\Omega - C} |\nabla u|^2 + \beta \int_C d\mathcal{H}^1 + \alpha \int_\Omega (u - f)^2, \qquad (1)$$

with f the input image on a two-dimensional planar domain Ω, γ, β and α are tuning parameters. The first term of Eq. (1) imposes the smoothness of u everywhere except on the discontinuity set C, the second term minimizes the total edge length of set C in terms of its one-dimensional Hausdorff measure $\mathcal{H}^1$ in $\mathbb{R}^2$ and the last term denotes the data fidelity term. This model is nonconvex and one of the major challenges is to develop efficient algorithms to find or approximate its minimizer.

Using the Γ-convergence framework, Ambrosio and Tortorelli approximated this functional by a sequence of elliptic functionals [4]. They proposed to replace the discontinuity set by a smooth auxiliary function v and Ambrosio–Tortorelli approximation of Eq. (1) is given by

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

where $0 < v < 1$ describes the discontinuity of edges: $v(x) \approx 0$ if $x \in C$, and $v(x) \approx 1$ otherwise, and ϵ is a small positive constant controlling the smoothness of v. And especially, Ambrosio and Tortorelli proved that when ϵ tends to zero in the $L^2(\Omega)$-topology (v_ϵ goes to 1), the minimizer $u = u_\epsilon$ of $E_\epsilon^{AT}(u, v)$ approximates a minimizer u of $E^{MS}(u, C)$.

MSTV approximation [5] was first suggested by Shah, replacing the L^2 norm $|\nabla u|^2$ by a total variation regularizer $|\nabla u|$ (L^1 norm), defined as

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

The Γ-convergence of MSTV approximation (3) was proved by Alicandro et al. [12]. Several generalizations of MS with Γ-convergence results were discussed in [13,14,2], and turned out the MSTV approximation (3) is more robust to recover much cleaner results. But it tends to suppress fine details, especially in the case of high noise density.

Inspired by the success of approximations of the original MS in image processing, they have been extended to process signals on mesh surfaces based on new discretization schemes [8–11]. Although they all minimized MS approximations on the normal vector field of voxel-based data [9] or triangular meshes [8,10,11], there is still a distinction among their discretizations of discontinuity function. More specifically, the discretizations in [8–10] are based on pointwise diffusion (defined on vertices), which fail to match the underlying geometric features with the discontinuity function while Liu et al. [11] calculated the discontinuity function on edges directly to locate geometric discontinuities.

In this paper, we will exploit a nonsmooth nonconvex version of MSTV based on the discretization of [11], helping to recover the weak edges that are smoothed out by the original MSTV.

2.2. Nonconvex regularizer and optimization

In general, variational models for signal and image processing mainly consist of two terms. The first term is a fidelity or data-fitting term, and the second one is a regularization term that recovers edges and smoothes the other homogeneous parts. It is well known that total variation model [15] is a classic convex regularizer ensuring the existence and uniqueness of a solution and performs well in preserving sharp features, but inevitably smoothes some weak features and fine details for their sparsity requirements. Since the pioneering work of Geman and Geman [16], different nonconvex regularizers [17,18] have been studied either in a statistical or variational framework and shown remarkable advantages over convex ones for restoring high-quality images with neat edges, especially the nonsmooth nonconvex ones. One can refer to [19] for a theoretical explanation for this phenomenon. Specifically, several studies [20–23] have utilized nonconvex higher-order regularizers and shown superiority over their corresponding convex ones for image restoration by preserving edges. To find or approximate the global optimal solution of nonconvex formulations, various efficient algorithms have been proposed, such as smoothing approximation methods [24–26], the graduated nonconvexity method [27,28], iteratively reweighted algorithms [29,30], half-quadratic minimization [31–33], proximal alternating minimization [34–36], primal–dual algorithm [37] and the alternating direction method of multipliers [38–42]. In this work, we adopt the iteratively convex majorization–minimization method [29,43], which has been successfully extended to solve nonsmooth nonconvex optimization problems of image processing problems with convergence analysis.

2.3. Variational methods in mesh processing

Variational methods have been developed to satisfy some prior knowledge by optimizing a given functional over a particular space or norm, and proven to be one of the most powerful techniques for solving many mesh processing tasks. The main variational methods in mesh processing include Poisson equation-type model [44], Lloyd's functional [45], L^0-based model [46,47], TV-based model [48,49], the curvature energy [50], filtering-based model [51,52], low-rank based model [53] and the MS functional [6,8–11]. This work focuses specifically on the applications of MS functional in the geometry processing.

The MS functional, representing an image as a piecewise-smooth function, is an important model for image processing in a variational framework. In the last 30 years, various variants of the MS functional have been proposed and led to numerous applications. Generally speaking, image segmentation [54,55] and denoising [54] are two main applications which are directly achieved from the variables being optimized. Inpainting [56] is modeled possible by removing the data fidelity term inside the missing area and inpainting the missing pixel under the optimization of MS regularizer. We believe the MS functional, as a classical tool, has the potential to produce more similar applications in geometry processing, and this paper explores a feature-preserving Mumford–Shah model via nonsmooth nonconvex regularizer for several applications.

3. Basic function spaces and operators

In this section, we introduce notations and define some basic function spaces and associated differential operators, which will yield an effective discretization scheme for our nonsmooth nonconvex MSTV over triangular mesh.

3.1. Notations

Suppose $\mathcal{M} \subset \mathbb{R}^3$, which is an oriented piecewise linear 2-manifold that consists of a set of vertices $\{p_i : i = 0, 1, \ldots, P-1\}$, edges $\{e_j : j = 0, 1, \ldots, E-1\}$ and oriented triangles (faces) $\{\tau_k : k = 0, 1, \ldots, T-1\}$. $p \prec e$ denotes that p is an endpoint of an edge e, $e \prec \tau$ denotes that e is an edge of a triangle τ, $p \prec \tau$ denotes that p is a vertex of a triangle τ.

Assume that all triangles are with counterclockwise orientation and all edges are with randomly chosen fixed orientations, the relative orientation of an edge e to a triangle τ can be introduced by $\mathrm{sgn}(e, \tau)$ in a similar way as follows. $\mathrm{sgn}(e, \tau) = 1$ denotes that the orientation of e is consistent with the orientation of τ, otherwise $\mathrm{sgn}(e, \tau) = -1$.

3.2. Function spaces and associated operators

In order to describe piecewise constant signal (e.g. face normal field) on a triangular mesh $\mathcal{M}$, we first define the piecewise constant function space $U = \mathbb{R}^T$, which is isomorphic to the piecewise constant function space on $\mathcal{M}$. For example, $u = (u_0, \ldots, u_{T-1}) \in U$ means that the value of u restricted on the triangle τ is u_τ, which is written as $u|_\tau$ sometimes. In order to further describe function v in Eq. (3), we define the edge function space $V = \mathbb{R}^E$ (sometimes denoted as $\mathcal{E}$), whose elements are defined at the edges of $\mathcal{M}$. For example, $v = (v_0, \ldots, v_{E-1}) \in V$ means that the value of v restricted on the edge e is v_e, which is written as $v|_e$ sometimes.

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

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

where s_τ is the area of triangle τ.

For any $v^1, v^2, v \in V$, we define

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

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

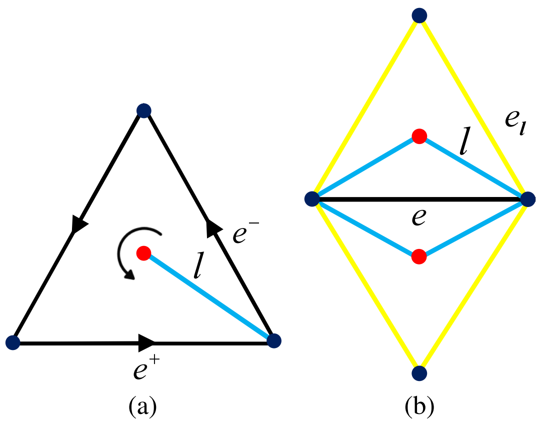

Fig. 1. Illustration on differential operators. (a) $[v]_l$ over the line l plotted in *cyan* in triangle τ with barycenter plotted in *red*; (b) $H(e)$ assembles all lines associated with the edge e (plotted in *cyan*) and the set of e_l associated with the lines contained in $H(e)$ refers to four edges (plotted in *yellow*).

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

$$\nabla_\mathcal{M} u|_e = \begin{cases} \sum\limits_{\tau, e \prec \tau} u_\tau \mathrm{sgn}(e, \tau), & e \nsubseteq \partial\mathcal{M} \\ 0, & e \subseteq \partial\mathcal{M} \end{cases}, \quad \forall e. \tag{6}$$

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

$$(\mathrm{div}_\mathcal{M} v)|_\tau = -\frac{1}{s_\tau} \sum_{\substack{e \prec \tau, \\ e \nsubseteq \partial\mathcal{M}}} v_e \mathrm{sgn}(e, \tau) \mathrm{len}(e), \quad \forall \tau. \tag{7}$$

The Laplace operator $\Delta_\mathcal{M} : U \to U$ is expressed as

$$(\Delta_\mathcal{M} u)|_\tau = -\frac{1}{s_\tau} \sum_{\substack{e \prec \tau, \tau \cap \tau_e = e, \\ e \nsubseteq \partial\mathcal{M}}} (u_\tau - u_{\tau_e}) \mathrm{len}(e), \quad \forall \tau. \tag{8}$$

To describe the diffusion of function defined at mesh edges, we will give the definitions of operators $\nabla_\mathcal{E}$, $\mathrm{div}_\mathcal{E}$ and $\Delta_\mathcal{E}$ on the edge function space V. 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 with two connected edges e^+ and e^-. e^+ enters the common vertex in the counterclockwise direction with $\mathrm{sgn}(e^+, l) = 1$, whereas e^- leaves the vertex in the counterclockwise direction with $\mathrm{sgn}(e^-, l) = -1$. All the aforementioned descriptions are illustrated in Fig. 1a. Then, for any $v \in V$, we define the jump of v over a line l as

$$[v]_l = v_{e^+} \mathrm{sgn}(e^+, l) + v_{e^-} \mathrm{sgn}(e^-, l), \tag{9}$$

The gradient operator $\nabla_\mathcal{E} : V \to W \subseteq \mathbb{R}^{3\times T}$ is given by

$$(\nabla_\mathcal{E} v)|_l = [v]_l, \quad \forall l. \tag{10}$$

The W space is equipped with the following inner product and norm:

$$(w^1, w^2)_W = \sum_l w^1|_l w^2|_l \mathrm{len}(l), \qquad \|w\|_W = \sqrt{(w, w)_W}, \tag{11}$$

for any $w^1, w^2, w \in W$, where $\mathrm{len}(l)$ is the length of line l. The divergence operator $\mathrm{div}_\mathcal{E} : W \to V$, which is the adjoint operator of $\nabla_\mathcal{E}$, can be derived using the inner products in V and W, as follows

$$(\mathrm{div}_\mathcal{E} w)|_e = -\frac{1}{\mathrm{len}(e)} \sum_{l \in H(e)} w_l \mathrm{sgn}(e, l) \mathrm{len}(l), \quad \forall e. \tag{12}$$

where $H(e)$ is the set of lines associated with the edge e, indicated as the *cyan* lines in Fig. 1b.

Based on the gradient operator (10) and divergence operator (12), the Laplace operator $\Delta_{\mathcal{E}} = \mathrm{div}_{\mathcal{E}}\nabla_{\mathcal{E}} : V \to V$ can be derived as

$$(\Delta_{\mathcal{E}} v)|_e = -\frac{1}{\mathrm{len}(e)} \sum_{l \in H(e)} (v_e - v_{e_l})\mathrm{len}(l), \quad \forall e. \tag{13}$$

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

We refer the readers to [48,11] for more details about the above operators.

To handle vectorial data in some applications, it is necessary to extend above descriptions to vectorial cases. Given the vectorial spaces $\mathbf{U} = \underbrace{U \times \cdots \times U}_{\mathfrak{M}}$, $\mathbf{V} = \underbrace{V \times \cdots \times V}_{\mathfrak{N}}$ and $\mathbf{W} = \underbrace{W \times \cdots \times W}_{\mathfrak{K}}$, the inner products and norms in $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ are defined as follows

$$(\mathbf{u}^1, \mathbf{u}^2)_{\mathbf{U}} = \sum_{1 \le i \le \mathfrak{M}} (u_i^1, u_i^2)_U, \quad \|\mathbf{u}\|_{\mathbf{U}} = \sqrt{(\mathbf{u}, \mathbf{u})_{\mathbf{U}}},$$
$$(\mathbf{v}^1, \mathbf{v}^2)_{\mathbf{V}} = \sum_{1 \le j \le \mathfrak{N}} (v_j^1, v_j^2)_V, \quad \|\mathbf{v}\|_{\mathbf{V}} = \sqrt{(\mathbf{v}, \mathbf{v})_{\mathbf{V}}},$$
$$(\mathbf{w}^1, \mathbf{w}^2)_{\mathbf{W}} = \sum_{1 \le k \le \mathfrak{K}} (w_i^1, w_i^2)_W, \quad \|\mathbf{w}\|_{\mathbf{W}} = \sqrt{(\mathbf{w}, \mathbf{w})_{\mathbf{W}}}.$$

4. Nonsmooth nonconvex Mumford–Shah model over surfaces

This section describes our model for mesh processing by utilizing the operators introduced in Section 3, which extends the model of Liu et al. [11] to a nonsmooth nonconvex version for normal filtering. We also develop an efficient optimization algorithm based on alternating direction method of multipliers (ADMM) to solve our model. Then, we present a vertex updating scheme to deform a mesh to the resulted normal field, recovering the underlying features with high mesh quality. For simplicity of descriptions and comparisons, we denoted two MS-related methods on mesh by MSAT [10] and MSTV [11].

4.1. Nonsmooth nonconvex MSTV normal estimation

For a given orientable triangular mesh, we denote raw face normal vectors as $\mathbf{N}^0 = [\mathbf{n}_0^0, \ldots, \mathbf{n}_{T-1}^0] \in \mathbb{R}^{3\times T}$, which are computed by

$$\mathbf{n}_i^0 = \frac{(\mathbf{v}_{i_1}^0 - \mathbf{v}_{i_2}^0) \times (\mathbf{v}_{i_3}^0 - \mathbf{v}_{i_2}^0)}{\|(\mathbf{v}_{i_1}^0 - \mathbf{v}_{i_2}^0) \times (\mathbf{v}_{i_3}^0 - \mathbf{v}_{i_2}^0)\|}, \tag{14}$$

where $\mathbf{v}_{i_1}^0, \mathbf{v}_{i_2}^0, \mathbf{v}_{i_3}^0 \in \mathbb{R}^3$ are its vertex positions enumerated according to the orientation. To recover the underlying face normal vectors with fine details, we introduce a nonsmooth nonconvex version of the vectorial MSTV approximation (3) with unit normal constraints (denoted as NMSTV), and consider its discretized version as follows:

$$\min_{\mathbf{N}\in\mathcal{C}_{\mathbf{N}}, v\in V} \Big\{ E(\mathbf{N}, v) = \gamma \sum_e v_e^2 \phi(\|(\nabla_{\mathcal{M}}\mathbf{N})|_e\|)\mathrm{len}(e) + \beta\Big(\epsilon\|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}\Big) + \alpha\|\mathbf{N}-\mathbf{N}^0\|_{\mathbf{U}}^2 \Big\}, \tag{15}$$

where $\mathcal{C}_{\mathbf{N}} = \{\mathbf{N} \in \mathbb{R}^{3\times T} : \|\mathbf{N}_\tau\| = 1, \forall \tau\}$, ϵ is small constant fixed by 0.001 in our implementation. ϕ is a nonsmooth nonconvex function such as

$$\phi_1(|t|) = \frac{|t|}{1+\rho|t|}, \quad \textit{or} \quad \phi_2(|t|) = \frac{1}{\rho}\log(1+\rho|t|), \tag{16}$$

which formulates nonconvex approximations to $|t|$ as $\rho \to 0$ (nonsmooth at zero).

Here, we briefly discuss the properties of ϕ. To protect edges from oversmoothing, the growth condition on ϕ should be imposed by $\lim_{t\to\infty}\phi(t) = c$ (c is a constant), so that the nonconvex term does not penalize the formulation of strong gradient of $\mathbf{N}$. In other words, its contribution is preferable to protect large details and sharp features or in the presence of high level of noise. On the other hand, $\lim_{t\to 0^+}\frac{\phi(t)}{t} = 1$ should be enforced to ensure that $\phi(t)$ has the similar behavior as the TV regularizer for meshes with weak edges or few discontinuous transitions, so that $\mathbf{N}$ can be better smoothed in homogeneous regions without staircasing effects. According to the work of in [57], ϕ_1 is well adapted for the reconstruction of piecewise-constant signals while ϕ_2 is more suitable to reconstruct piecewise-smooth signals. Since the normal vector field of 3D mesh is mostly piecewise-smooth, we choose the log function ϕ_2 as our nonconvex function. Because our work aims to show the superiority of nonsmooth nonconvex regularizers, a comparison of nonconvex regularizers is beyond the scope of this paper.

Note that the two unknowns $\mathbf{N}$ and v are coupled in Eq. (15), our idea is to use the variable splitting technique to reformulate the unconstrained model into the following constrained problem as

$$\min_{\substack{\mathbf{N}\in\mathcal{C}_{\mathbf{N}}, v\in V,\\ \mathbf{p}\in\mathbf{V}}} \Big\{ \gamma \sum_e v_e^2 \phi(\|\mathbf{p}_e\|)\mathrm{len}(e) + \beta\Big(\epsilon\|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}\Big) + \alpha\|\mathbf{N}-\mathbf{N}^0\|_{\mathbf{U}}^2 \Big\}, \quad \text{s.t.} \quad \mathbf{p} = \nabla_{\mathcal{M}}\mathbf{N}. \tag{17}$$

We adopt the iteratively reweighted l_1 algorithm (IRLA) [29] to model (17), and a convex minimization problem to update $(\mathbf{N}, v, \mathbf{p})$ is given by

$$\min_{\substack{\mathbf{N}\in\mathcal{C}_{\mathbf{N}}, v\in V,\\ \mathbf{p}\in\mathbf{V}}} \Big\{ \gamma \sum_e v_e^2 \tilde{\mathbf{p}}_e^k \|\mathbf{p}_e\|\mathrm{len}(e) + \beta\Big(\epsilon\|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}\Big) + \alpha\|\mathbf{N}-\mathbf{N}^0\|_{\mathbf{U}}^2 \Big\}, \quad \text{s.t.} \quad \mathbf{p} = \nabla_{\mathcal{M}}\mathbf{N}. \tag{18}$$

with $\tilde{\mathbf{p}}^k = \frac{1}{\rho\|\mathbf{p}^k\|_{\mathbf{V}}+1}$. Problem (18) is a l_1 related convex optimization and can be efficiently solved by ADMM, which yields

$$\begin{cases} \mathbf{N}^{k+1} = \arg\min_{\mathbf{N}\in\mathcal{C}_{\mathbf{N}}} \alpha\|\mathbf{N}-\mathbf{N}^0\|_{\mathbf{U}}^2 + \frac{r_{\mathbf{p}}}{2}\|\nabla_{\mathcal{M}}\mathbf{N} - \mathbf{p}^k - \frac{\boldsymbol{\lambda}_{\mathbf{p}}^k}{r_{\mathbf{p}}}\|_{\mathbf{U}}^2, \\ \mathbf{p}^{k+1} = \arg\min_{\mathbf{p}\in\mathbf{V}} \gamma \sum_e (v_e^k)^2 \tilde{\mathbf{p}}_e^k \|\mathbf{p}_e\|\mathrm{len}(e) + \frac{r_{\mathbf{p}}}{2}\|\mathbf{p} - (\nabla_{\mathcal{M}}\mathbf{N}^{k+1} - \frac{\boldsymbol{\lambda}_{\mathbf{p}}^k}{r_{\mathbf{p}}})\|_{\mathbf{V}}^2, \\ v^{k+1} = \arg\min_{v\in V} \gamma \sum_e v_e^2 \tilde{\mathbf{p}}_e^k \|\mathbf{p}_e^{k+1}\|\mathrm{len}(e) + \beta\Big(\epsilon\|\nabla_{\mathcal{E}} v\|_W^2 + \frac{\|v-1\|_V^2}{4\epsilon}\Big), \\ \boldsymbol{\lambda}_{\mathbf{p}}^{k+1} = \boldsymbol{\lambda}_{\mathbf{p}}^k - \nu r_{\mathbf{p}}(\mathbf{p}^{k+1} - \nabla_{\mathcal{M}}\mathbf{N}^{k+1}). \end{cases} \tag{19}$$

where the penalty parameter $r_{\mathbf{p}} > 0$ is a fixed constant, and the relaxation $\nu \in (0, (\sqrt{5}+1)/2]$ is required for the convergence of the ADMM algorithm.

The **N**-subproblem in problem (19) is a quadratic minimization with unit normal constraints. Here we employ an approximation strategy to solve the problem. Specifically, we first solve a quadratic programming for $\mathbf{N}^{k+1}$ and then project the minimizer onto a unit sphere to satisfy the unit normal constraints. The Euler–Lagrange equation is given as follows

$$r_{\mathbf{p}}(\Delta_{\mathcal{M}}\mathbf{N}) - 2\alpha\mathbf{N} = \mathrm{div}_{\mathcal{M}}(r_{\mathbf{p}}\mathbf{p}^k + \boldsymbol{\lambda}_{\mathbf{p}}^k) - 2\alpha\mathbf{N}^0, \tag{20}$$

This equation can be reformulated into a sparse linear system and solved by efficient sparse linear solvers, such as Eigen, Taucs, and Math Kernal Library (MKL).

The **p**-subproblem in problem (19) is solved separately on each edge since the minimization problem can be spatially decoupled respect to edge. Hence, for each $\mathbf{p}_e$, the following simplified problem is needed to be solved

$$\min_{\mathbf{p}_e} \gamma(v_e^k)^2\tilde{\mathbf{p}}_e^k\|\mathbf{p}_e\| + \frac{r_{\mathbf{p}}}{2}\|\mathbf{p}_e - \big((\nabla_{\mathcal{M}}\mathbf{N}^{k+1})|_e - \frac{\boldsymbol{\lambda}_{\mathbf{p}_e}^k}{r_{\mathbf{p}}}\big)\|_{\mathbf{V}}^2, \tag{21}$$

which has a closed form solution

$$\mathbf{p}_e = \begin{cases} (1 - \dfrac{\gamma(v_e^k)^2\tilde{\mathbf{p}}_e^k}{r_{\mathbf{p}}\|\boldsymbol{\psi}_e^{k+1}\|_{\mathbf{V}}})\boldsymbol{\psi}_e^{k+1}, & \|\boldsymbol{\psi}_e^{k+1}\|_{\mathbf{V}} > \dfrac{\gamma(v_e^k)^2\tilde{\mathbf{p}}_e^k}{r_{\mathbf{p}}} \\ 0, & \|\boldsymbol{\psi}_e^{k+1}\|_{\mathbf{V}} \le \dfrac{\gamma(v_e^k)^2\tilde{\mathbf{p}}_e^k}{r_{\mathbf{p}}} \end{cases}, \tag{22}$$

where $\boldsymbol{\psi}^{k+1} = \nabla_{\mathcal{M}}\mathbf{N}^{k+1} - \frac{\boldsymbol{\lambda}_{\mathbf{p}}^k}{r_{\mathbf{p}}}$.

The v-subproblem in problem (19) is also a quadratic programming and the corresponding Euler–Lagrange equation is given as follows

$$(2\gamma\tilde{\mathbf{p}}^k\|\mathbf{p}^{k+1}\| + \frac{\beta}{2\epsilon})v - 2\beta\epsilon(\Delta_{\mathcal{E}}v) = \frac{\beta}{2\epsilon}. \tag{23}$$

Plugging the Laplace operator in Eq. (13), we can rewrite the above equation as a sparse linear system, which again can be solved by linear solvers.

Finally, the iterative algorithm to solve our NMSTV model (15) is summarized in Algorithm 1.

Algorithm 1 The IRLA with ADMM algorithm for NMSTV normal estimation

```
Initializations: Set $k = 0$, $(v^0, \mathbf{p}^0; \boldsymbol{\lambda}_{\mathbf{p}}^0) = \mathbf{0}$, $\tilde{\mathbf{p}}^0 = 1$ and
parameters $\alpha, \beta, \gamma, \epsilon, \rho, r_{\mathbf{p}}, \nu$ and $\varepsilon_1$;
Repeat
    (1) For fixed $(\mathbf{p}^k; \boldsymbol{\lambda}_{\mathbf{p}}^k)$, compute $\mathbf{N}^{k+1}$ according to Eq. (20);
    (2) For fixed $(\mathbf{N}^{k+1}, v^k; \boldsymbol{\lambda}_{\mathbf{p}}^k)$, compute $\mathbf{p}^{k+1}$ according to Eq. (22);
    (3) For fixed $\mathbf{p}^{k+1}$, compute $v^{k+1}$ according to Eq. (23);
    (4) Update Lagrange multiplier $\boldsymbol{\lambda}_{\mathbf{p}}^{k+1}$ according to Eq. (19);
Until $\|\mathbf{N}^{k+1} - \mathbf{N}^k\|_{\mathbf{U}} < \varepsilon_1$ or $k \ge 30$;
Return $\mathbf{N}^{k+1}$.
```

Remark 1. Here, we analyze the effect of our nonconvex regularizers on feature recovery of normal estimation, superior to that of the original MSTV [11]. For the **p**-subproblem in Eq. (21), since $\tilde{\mathbf{p}}_e^k < 1$, the weight function $\tilde{\mathbf{p}}_e^k(v_e^k)^2$ in our NMSTV is less than $(v_e^k)^2$ in MSTV [11]. This illustrates that our model smoothes edges less where $\nabla_{\mathcal{M}}\mathbf{N} > 0$, which produces high-quality face normals with sharp features just as shown in Fig. 2. For each testing method, we compare the denoised (feature extraction) results and the corresponding discontinuity functions via color coding. From the figure, we can observe that MSTV almost obtains the same result as ours when weak noise is corrupted while the nonconvex one performs better than others with the increased noise. Please refer to more numerical examples in Section 6.

4.2. Vertex updating

After estimating the face normal field by the proposed NMSTV model, we need to deform the mesh by moving its vertices such that the face normal vectors match the prescribed normal field. To matching normals, existing methods usually minimize an energy of enforcing orthogonality between the new edge vectors and the target face normals [58], which are efficient but lack control over inversions because of the triangle-wise orientation ambiguity. To address this issue, we take both the constraint-based optimization and the discontinuity function v of normal filtering into account, enforcing the oriented normals as soft constraints and encoding a fairness regularization with the discontinuities.

Minimizing energy. Let $\mathbf{V}^0 = [\mathbf{v}_0^0, \ldots, \mathbf{v}_{P-1}^0] \in \mathbb{R}^{3\times P}$ be the original vertex positions, and $\hat{\mathbf{N}} = [\hat{\mathbf{n}}_0, \ldots, \hat{\mathbf{n}}_{T-1}] \in \mathbb{R}^{3\times T}$ the target face normal field with a collection of feasible sets $\mathcal{C} = [\mathcal{C}_0, \ldots, \mathcal{C}_T]$. The new vertex positions $\mathbf{V} = [\mathbf{v}_0, \ldots, \mathbf{v}_{P-1}]$ are achieved by minimizing the following energy

$$\min_{\mathbf{V},\mathbf{P}\in\mathcal{C}} \Big\{\omega_1\sum_{i=0}^{T-1}\|\mathbf{M}\mathbf{V}_{f_i} - \mathbf{P}_i\|_F^2 + \omega_2\sum_e v_e^2\|\mathbf{v}_{i_1} + \mathbf{v}_{i_2} - \mathbf{v}_{i_3} - \mathbf{v}_{i_4}\|_F^2 + \omega_3\|\mathbf{V} - \mathbf{V}^0\|_F^2\Big\}, \tag{24}$$

where $\mathbf{V}_{f_i} \in \mathbb{R}^{3\times 3}$ stores the oriented vertex positions of face f_i in its rows, the matrix $\mathbf{M} = \frac{1}{3}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$ produces the mean-centered constraint vertex positions for a face, and $\mathbf{P}_i \in \mathbb{R}^{3\times 3}$ represents the closest projection of $\mathbf{M}\mathbf{V}_{f_i}$ onto the corresponding feasible set $\mathcal{C}_i$. $\tau_1 = (\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \mathbf{v}_{i_3})$ and $\tau_2 = (\mathbf{v}_{i_1}, \mathbf{v}_{i_4}, \mathbf{v}_{i_2})$ are neighbor triangles associated with the edge $e = (\mathbf{v}_{i_1}, \mathbf{v}_{i_2})$.

Here the first term penalizes the violation of oriented normal constraint for each face through a distance measure, and aims to find a direct solution or least-squares one in the feasible sets $\mathcal{C}$. The use of mean-centering matrix **M** adopts the translation-invariance of the oriented normal constraint to allow for faster convergence of the solver [59,60]. The second term utilizes the discontinuity function v to guide neighboring triangles to share geometric normals as similar as possible, smoothing the deformed edge where $v_e \approx 1$ while keeping sharp edges where $v_e \approx 0$. For each edge e, the last term prevents vertices to depart too much from their original positions $\mathbf{V}^0$. ω_1, ω_2 and ω_3 are user-specified positive parameters, and the setting of ω_2 and ω_3 are discussed in Section 5.2 while ω_1 is fixed to 1 by default throughout all our experiments.

Solving for positions. Following the approach of [59,61], we solve the optimization problem (24) via an alternating minimization scheme of **V** and **P**.

Firstly, we fix **V** and update **P**. We separate this problem to a set of subproblems $\{\mathbf{P}_i\}$ with respect to each face f_i and solve in parallel. Each subproblem searches for an solution that projects $\mathbf{M}\mathbf{V}_{f_i}$ onto the plane orthogonal to $\hat{\mathbf{n}}_i$. Obviously, the closest projection $\hat{\mathbf{P}}_i$ can be computed as $\mathbf{M}\mathbf{V}_{f_i}(\mathbf{I}_3 - \hat{\mathbf{n}}_i\hat{\mathbf{n}}_i^{\mathrm{T}})$. Let $\mathbf{n}_i$ be the oriented unit normal for the current vertex positions $\mathbf{M}\mathbf{V}_{f_i}$, we achieved the possible solutions for $\mathbf{P}_i$ according to the relation between $\mathbf{n}_i$ and $\hat{\mathbf{n}}_i$

$$\mathbf{P}_i = \begin{cases} \hat{\mathbf{P}}_i, & \mathbf{n}_i\cdot\hat{\mathbf{n}}_i \ge 0 \\ \hat{\mathbf{P}}_i(\mathbf{h}\mathbf{h}^{\mathrm{T}}), & \mathbf{n}_i\cdot\hat{\mathbf{n}}_i < 0 \end{cases} \tag{25}$$

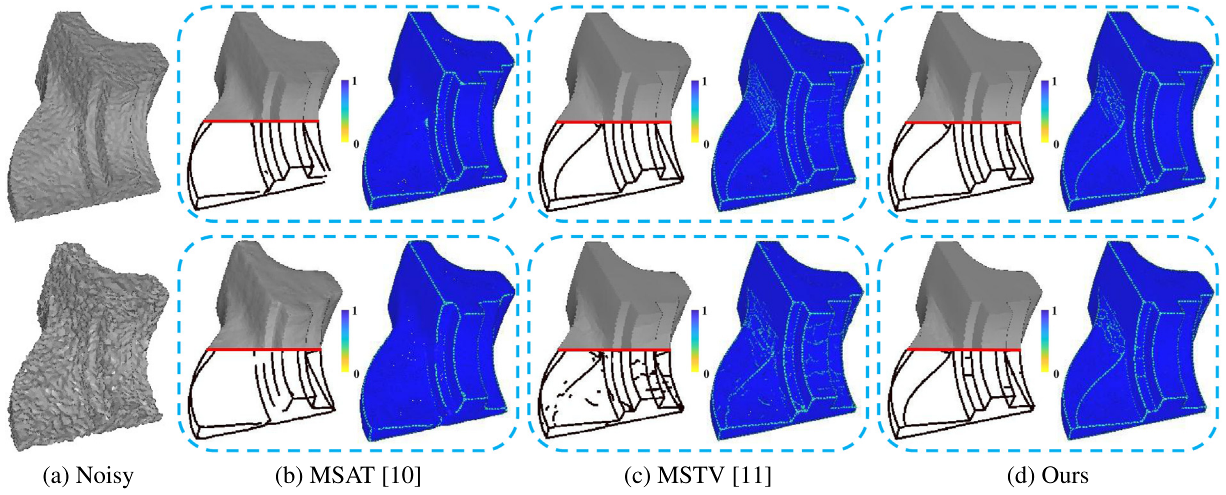

Fig. 2. Feature-preserving comparison of different Mumford–Shah models on different levels of noise. (a) shows Fandisk corrupted with $\sigma = 0.1\bar{l}_e$ and $\sigma = 0.2\bar{l}_e$, each column from (b) to (d) shows denoised (feature extraction) results and corresponding discontinuity functions produced by MSAT [10], MSTV [11] and our proposed model, where $v(x) = 0$ indicates the discontinuous areas (features) and $v(x) = 1$ indicates the smooth areas.

where $\mathbf{h}$ is the right-singular vector of $\hat{\mathbf{P}}_i$ corresponding to its largest singular value.

Next, we fix $\mathbf{P}$ and update $\mathbf{V}$. This is equivalent to

$$\min_{\mathbf{V}} \omega_1 \|\mathbf{KV} - \mathbf{P}\|_F^2 + \omega_2 \|\mathbf{HV}\|_F^2 + \omega_3 \|\mathbf{V} - \mathbf{V}^0\|_F^2, \tag{26}$$

where the sparse matrix $\mathbf{K} \in \mathbb{R}^{3T\times P}$ collects the mean-centering matrix coefficients for each face and the sparse matrix $\mathbf{H}$ combines the discontinuity function v and related vertices for each edge. This amounts to solve the following normal equations

$$(\omega_1 \mathbf{K}^{\mathrm{T}}\mathbf{K} + \omega_2 \mathbf{H}^{\mathrm{T}}\mathbf{H} + \omega_3 \mathbf{I}_P)\mathbf{V} = \omega_1 \mathbf{K}^{\mathrm{T}}\mathbf{P} + \omega_3 \mathbf{V}^0. \tag{27}$$

where $\mathbf{I}_P$ is the $P \times P$ identity matrix. Since the left-hand matrix is fixed during all iterations, we can pre-factor it using sparse Cholesky factorization once to allow for efficient solving in each iteration.

The above alternating minimization is repeated until convergence and sketched in Algorithm 2.

Algorithm 2 The alternating minimization for vertex updating

```
Initializations: Set $k = 0$, parameters $\omega_1$, $\omega_2$, $\omega_3$ and $\varepsilon_2$;
Repeat
    (1) For fixed $\mathbf{V}^k$, compute $\mathbf{P}^{k+1}$ according to Eq. (25);
    (2) For fixed $\mathbf{P}^{k+1}$, compute $\mathbf{V}^{k+1}$ according to Eq. (27);
Until $||\mathbf{V}^{k+1} - \mathbf{V}^k|| < \varepsilon_2$ or $k \geq 30$;
Return $\mathbf{V}^{k+1}$.
```

Our vertex updating method can efficiently compute a new mesh that is consistent with the target face normals, while being close to the original mesh shape with rich geometric details. Especially, a fairness term is designed by considering the discontinuity function and favors more regular near-Delaunay meshes, even prevents the reconstructed vertices from overlapping. Fig. 3 compares our approach with the vertex updating method proposed in [10,60]. For each method, we visualize the aspect ratio of each triangle $\kappa_\tau = 1 - \frac{\theta_{\min}}{\pi/3}$ ($\theta_{\min}$ is the minimum angle of triangle τ) via color coding, evaluate the mean square angular error (MSAE) between face normals of the resulting mesh and the original mesh, the vertex deviation $E_{v,h}$ of the denoised result from the underlying clean surface and the number of inverted triangles. The resulting mesh using our method with a fairness term (shown in the last column of Fig. 3) is noticeably closer to the original mesh from the comparisons of MSAE and $E_{v,h}$ in the brackets. Moreover, our method achieves the best uniformity of mesh without overlap, even for the input mesh corrupted by of high level of impulsive noise with random directions.

5. Analysis

5.1. Implementation details

The proposed method is implemented in C++ on a windows 10 platform with an Intel Corei7 at 3.5 GHz and 16 GB RAM, using Eigen for all linear algebra operations and OpenMP for parallelization. Since the coefficient ρ determines the degree of non-convexity of regularizers, we use weaker non-convexity ($\rho = 0.1$) to achieve a smoother normal field for non-CAD models and use stronger non-convexity ($\rho = 0.9$) to keep more sharp edges for CAD models. For the parameters in Algorithm 1, the default setting is as follows: $\epsilon = 0.001$, $\nu = 1.618$, $r_{\mathbf{p}} = 1$, $\varepsilon_1 = e^{-6}$ and $k = 30$. For the parameters in Algorithm 2, we fix $\omega_1 = 1$, $\varepsilon_2 = e^{-6}$ and $k = 30$. The setting of other parameters is discussed in Section 5.2 in detail. In addition, we present a full explanation of how parameters are in a special setting for a specific application in Section 6.

5.2. Parameters setting

Our normal estimation is influenced by three parameters: α, β and γ, which balance fidelity, filtering and discontinuity terms of Eq. (15). Specifically, α plays an important role in preventing the solution deviating from far from the input. To produce satisfactory results, on one hand, α is suggested with smaller values for CAD meshes and with larger values for non-CAD and scanned meshes. On the other hand, α is first set in a suggested range with lower values for higher level of noise before β and γ are adjusted. Fig. 4 shows the effect of α on denoising results of a non-CAD mesh.

$1/\beta$ represents the length of located discontinuity edges, the smaller value produces longer discontinuities and thus less smooth denoising results, as shown in Fig. 5. This fully explains why filtering and feature detection are complementary in mesh denoising. Extensive experimental results show that there exists a range ([0.005, 0.5]) for β that can produce satisfactory results. γ controls the impact of nonsmooth nonconvex TV regularization

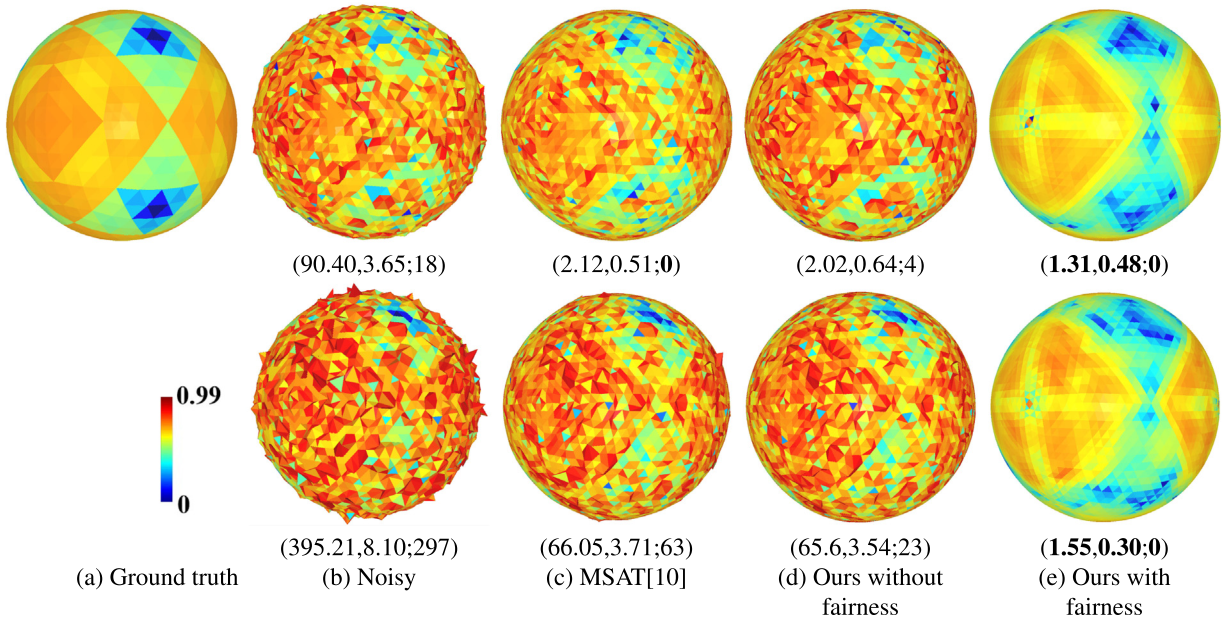

Fig. 3. Illustration on the property of our fairness term. (a) the original model; (b) noisy models corrupted by two different levels of impulsive noise with random direction; (c) denoised results achieved by MSAT [10] where discontinuity function v on edge e in vertex updating was set as the mean value of two vertices of this edge; (d) denoised results achieved by our NMSTV normal estimation and vertex updating without fairness; (e) denoised results achieved by our NMSTV normal estimation and vertex updating with a fairness term solved by Algorithm 2, where v_e is solved from the normal filtering. The color encodes $\kappa_\tau = 1 - \frac{\theta_{\min}}{\pi/3}$ characterizing the uniformity of meshes, where $\theta_{\min}$ is the minimum angle of triangle τ. The numbers in bracket below the figure are MSAE ($\times 10^{-3}$), $E_{v,h}$ ($\times 10^{-2}$) and number of inverted triangles respectively.

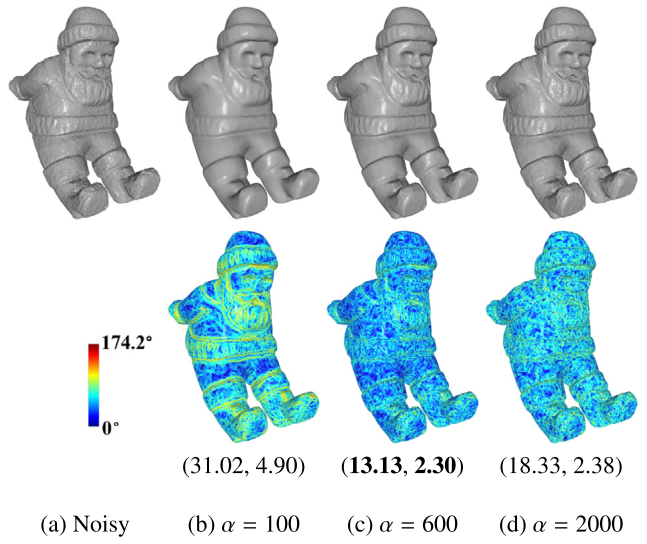

Fig. 4. Impact of parameter α on denoising results. The first column shows the input noisy mesh (corrupted by impulsive noise with $\sigma = 0.1\bar{l}_e$), each column from the second to the fourth shows a denoised mesh achieved with different α and the corresponding error maps using the mean angular difference between the face normals of a denoised mesh and its ground truth mesh. The numbers in bracket below the figure are MSAE ($\times 10^{-3}$) and $E_{v,h}$ ($\times 10^{-2}$). The smaller value of α usually yields an over-smooth result while the larger one always fails to remove noise effectively.

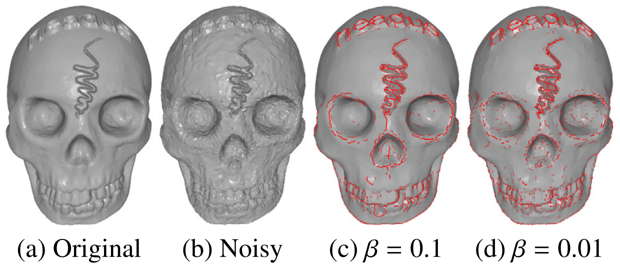

Fig. 5. Impact of parameter β on denoising results. From left to right: original mesh, noisy mesh (corrupted by Gaussian noise with $\sigma = 0.1\bar{l}_e$) and denoising results with $\beta = 0.1$ and $\beta = 0.01$, respectively. The smaller value of β produces longer features and more discontinuities.

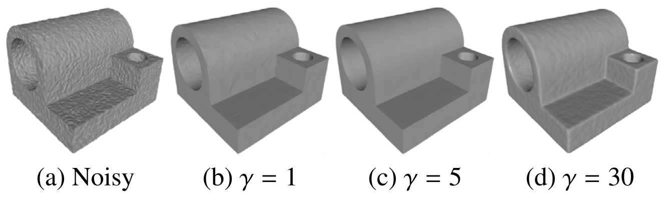

Fig. 6. Impact of parameter γ on denoising results. From left to right: noisy mesh (corrupted by Gaussian noise with $\sigma = 0.1\bar{l}_e$) and denoising results with $\gamma = 1$, $\gamma = 5$ and $\gamma = 30$, respectively. The smaller value of γ cannot remove noise effectively, while the larger one always smoothes feature edges.

in our NMSTV model, which increases with the increase of noise level. A smaller γ cannot remove noise effectively while a larger γ will oversmooth sharp features, as shown in Fig. 6. Similar to β, there exists a range ([0.5, 10]) for γ that leads to promising results.

In vertex updating optimization, there are two parameters ω_2 and ω_3 to balance the fairness and fidelity term. ω_2 is introduced to prevent a large number of flipped triangles when matching prescribed normal field, ω_3 penalizes the deviation between the new vertex positions and the input ones. For producing satisfactory results, ω_2 is suggested with smaller values for meshes of a slight noise and with larger values for meshes of severe noise. In contrast, ω_3 is suggested with larger values for meshes of a slight noise and with smaller values for meshes of severe noise. In our implementation, they are both set in the range of [0.001, 0.1].

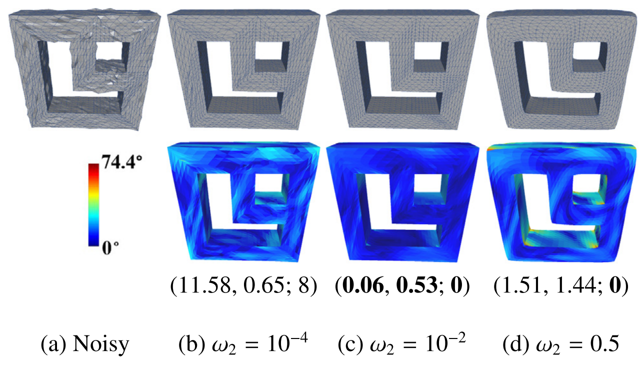

Fig. 7. Impact of parameter ω_2 on denoising results. The first column shows the input noisy mesh (corrupted by impulsive noise with $\sigma = 0.2\bar{l}_e$), each column from the second to the fourth shows a denoised mesh achieved with different ω_2 and the corresponding error maps using the mean angular difference between the face normals of a denoised mesh and its ground truth mesh. The numbers in bracket below the figure are MSAE ($\times 10^{-3}$), $E_{v,h}$ ($\times 10^{-2}$) and number of inverted triangles respectively. The smaller value of ω_2 usually results in poor mesh quality even overlaps, while the larger one always smoothes and shrinks the results with high distortions.

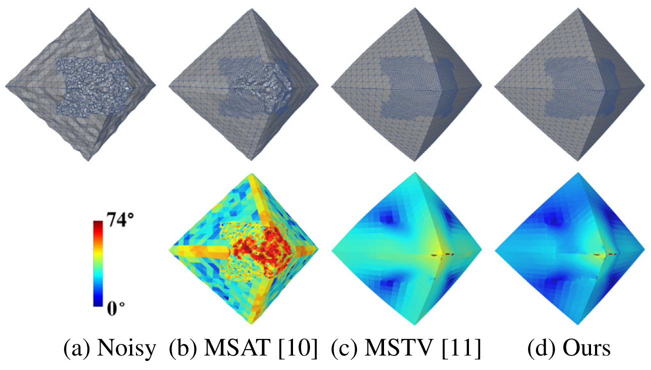

Fig. 8. Results of different Mumford–Shah models on irregular sampling mesh (corrupted by Gaussian noise with $\sigma = 0.3\bar{l}_e$). The first column shows the input noisy surface, each column from the second to the fourth shows a denoised mesh and the corresponding error maps using the mean angular difference between the face normals of a denoised mesh and its ground truth mesh.

Fixed $\omega_3 = 0.01$, we show the effect of varying ω_2 in Fig. 7. For comparison, we visualize the error map for the normals with MSAE. From the figure we can observe that the smaller value of ω_2 usually results in poor mesh quality even overlaps, while the larger one always smoothes and shrinks the results with high distortions.

5.3. Robust to irregular sampling

To test the sensitivity of our method to irregular sampling, Pyramid model with irregular sampling is compared in Fig. 8. For each testing method, we compare the denoised results and visualize the error map for the normals with MSAE. As we can see, although the noisy meshes are of varying density distributions, our results still show compelling quality, especially in the feature and irregular sampling regions.

5.4. Robust to different levels of noise

To test the sensitivity of our method to different levels of noise, Fig. 9 shows a comparison on the feature extraction with two Mumford–Shah models including MSAT [10] and MSTV [11], where the first row is with moderate noise and the second with extreme one. Results show that our method is more robust to noise on feature extraction, making our method suitable for the denoising application.

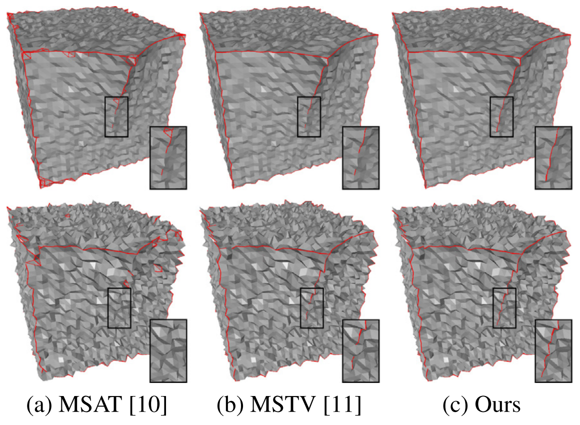

Fig. 9. Feature extraction on meshes corrupted by different levels of noise. First row: feature lines on meshes with a moderate noise level from MSAT [10], MSTV [11] and NMSTV, respectively. Second row: feature lines on meshes with a extreme noise level from MSAT [10], MSTV [11] and NMSTV, respectively. Our method are more robust to extract features of extreme noise levels.

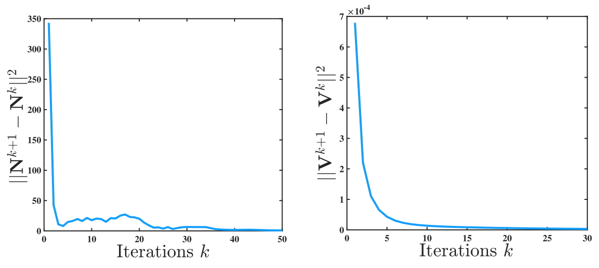

Fig. 10. Illustration on the convergence of Algorithms 1 and 2. These plots illustrate that, as the iteration number k increases, the minimizing solution $\mathbf{N}^k$ and $\mathbf{V}^k$ both converge to their fixed points within a few iterations.

5.5. Convergence analysis

In our normal estimation, an inner alternating minimization scheme is applied to decrease the energy monotonically and a partial convergence of the IRLA in problem (18) for solving the constrained problem (17) can be obtained. In other words, the sequence $\{\mathbf{N}^k, \mathbf{p}^k, v^k\}$ generated by problem (18) is bounded and has at least one accumulation point. That is, there exists a converging subsequence to an accumulation point. However, the nonconvex log function in problem (17) is not a sum of a convex function; thus, we cannot even assure the local convergence (see [29] for more details). In vertex updating, an iterative two-step minimization strategy is employed and guaranteed to converge monotonically to a local minimum, even though this minimum is not necessarily reached in a finite number of steps. The convergence rate depends on the conditions of the problem and the projection functions involved (see [59] for more details).

In Fig. 10, we display plots of $\|\mathbf{N}^{k+1} - \mathbf{N}^k\|^2$ and $\|\mathbf{V}^{k+1} - \mathbf{V}^k\|^2$ with respect to iterations of k in Algorithms 1 and 2 (take the model in Fig. 8 for an example). We can observe that, from two curves, the minimizing solution $\mathbf{N}^k$ and $\mathbf{V}^k$ both converge to their fixed points within a few iterations. Although the energy $\|\mathbf{N}^{k+1} - \mathbf{N}^k\|^2$ in Algorithm 1 increases abruptly at 18th iteration, it eventually decreases. Therefore, this figure indicates that our proposed algorithms asymptotically solve the original normal estimation and vertex updating problem.

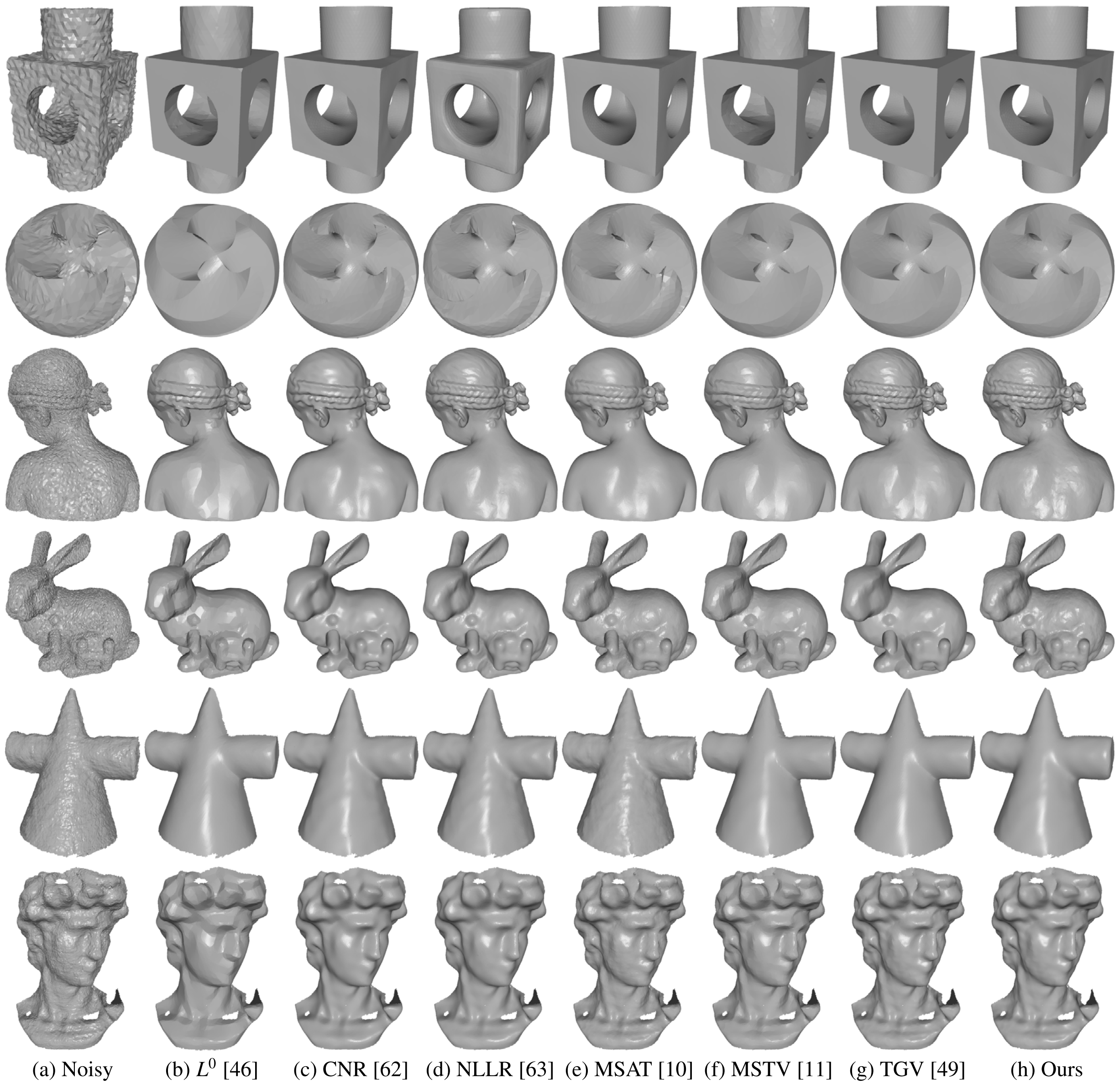

Fig. 11. Denoising results and comparisons. The first two rows illustrate denoising on CAD meshes, the middle two rows illustrate denoising on non-CAD meshes and the last two rows illustrate denoising on scanning data. Visual comparisons show the superior performance of our method in simultaneously preserving features and recovering smooth regions.

6. Applications

In this section, we demonstrate the effectiveness of our approach on several geometry processing applications, including denoising, inpainting and segmentation. Meanwhile, we compare the proposed method with various state-of-the-art methods in each application. For fair comparisons, the results for comparison are all from the source codes or already-trained models obtained from the original authors and the parameters are set to be optimal according to the original papers.

6.1. Denoising

Given an input noisy mesh, our method smoothes out noise while preserving sharp features. For this application, we solve for the nonsmooth nonconvex MS functional to estimate the normal vector field, and then deform mesh vertices to match the estimated normals. To demonstrate the performance of our proposed approach on noise removal, we compare with six popular methods. Among these competitors, L^0 [46] and TGV [49] are two models of variational framework, MSAT [10] and MSTV [11] are Mumford–Shah methods, NLLR [62] is nonlocal-based method, and CNR [63] is a data-driven method. Fig. 11 shows a comparison both on synthetic shapes perturbed using a Gaussian noise and realistic noisy LiDAR scanning data. First, L^0 always oversmoothes small-scale features while over-sharpens medium-scale features, transforming smooth regions into piecewise constant ones (see Fig. 11b); CNR is effective in preserving medium-scale features (see Block model in Fig. 11c) but it may smooth small-scale features and fine details (see Child model in Fig. 11c); In contrast, NLLR recovers some small-scale features in a better matter (see Bunny model in Fig. 11d) while over-smoothes large-scale features (see Block model in Fig. 11d); MSAT usually preserves sharp features well, but produces some false edges in smooth regions even overlaps due to its smoothing strategy for the discontinuity function (see Fig. 11e); MSTV produces clearer feature-preserving results but still induces slight staircase effects appeared in smooth regions (see Fig. 11f); the results

Table 1
Quantitative evaluation of the results in Fig. 11. For each result, we compared MSAE ($\times 10^{-3}$), $E_{v,h}$ ($\times 10^{-2}$), number of inverted triangles and the execution time (in seconds) respectively.

Mesh	L0 [46]	CNR [63]	NLLR [62]	MSAT [10]	MSTV [11]	TGV [49]	Ours
Block	4.68, 1.12, 1, 3.01	**2.57**, **0.86**, **0**, 0.72	67.31, 2.30, **0**, 2.75	8.42, 1.74, **0**, 16.30	5.18, 1.21, **0**, 0.98	3.24, 1.15, **0**, 5.98	2.55, **0.84**, **0**, 1.10
Sharp-sphere	32.39, 1.16, 46, 2.66	64.88, 1.29, 178, **0.79**	175.31, 1.69, 403, 2.32	76.43, 1.92, 97, 20.14	45.11, 1.63, 67, 0.96	27.80, 1.27, 67, 4.80	**20.02**, **0.73**, **36**, 1.08
Child	42.54, 0.58, 7, 24.11	31.62, 0.52, 15, **3.75**	**21.57**, 0.42, 6, 10.59	44.93, 0.89, 30, 112.11	41.39, 0.82, 11, 5.88	24.20,0.58, 5, 29.54	24.81, **0.38**, **3**, 7.32
Bunny-hi	24.08, 0.71, 4, 16.14	12.95, 0.66, **2**, 2.43	**16.01**, 0.78, 5, 6.696	18.77, 0.79, **2**, 72.84	19.26, 0.80, 3, 3.91	24.44, 1.32, 3, 21.13	16.68, **0.54**, **2**, **2.20**
Cone	58.31, 3.71, /, 8.28	67.76, **2.78**, /, **1.81**	63.40, 2.99, /, 54.32	66.94, 2.90, /, 57.93	63.87, 3.35, /, 3.43	59.79, 3.57, /, 43.13	**57.50**, 3.39, /, 3.84
David	93.22, 2.35, /, 15.43	94.17, **1.94**, /, **3.78**	100.54, 2.20, /, 171.35	90.83, 2.63, /, 95.24	100.36, 2.23, /, 5.79	89.42, 2.29, /, 24.27	**88.36**, 2.27, /, 6.88

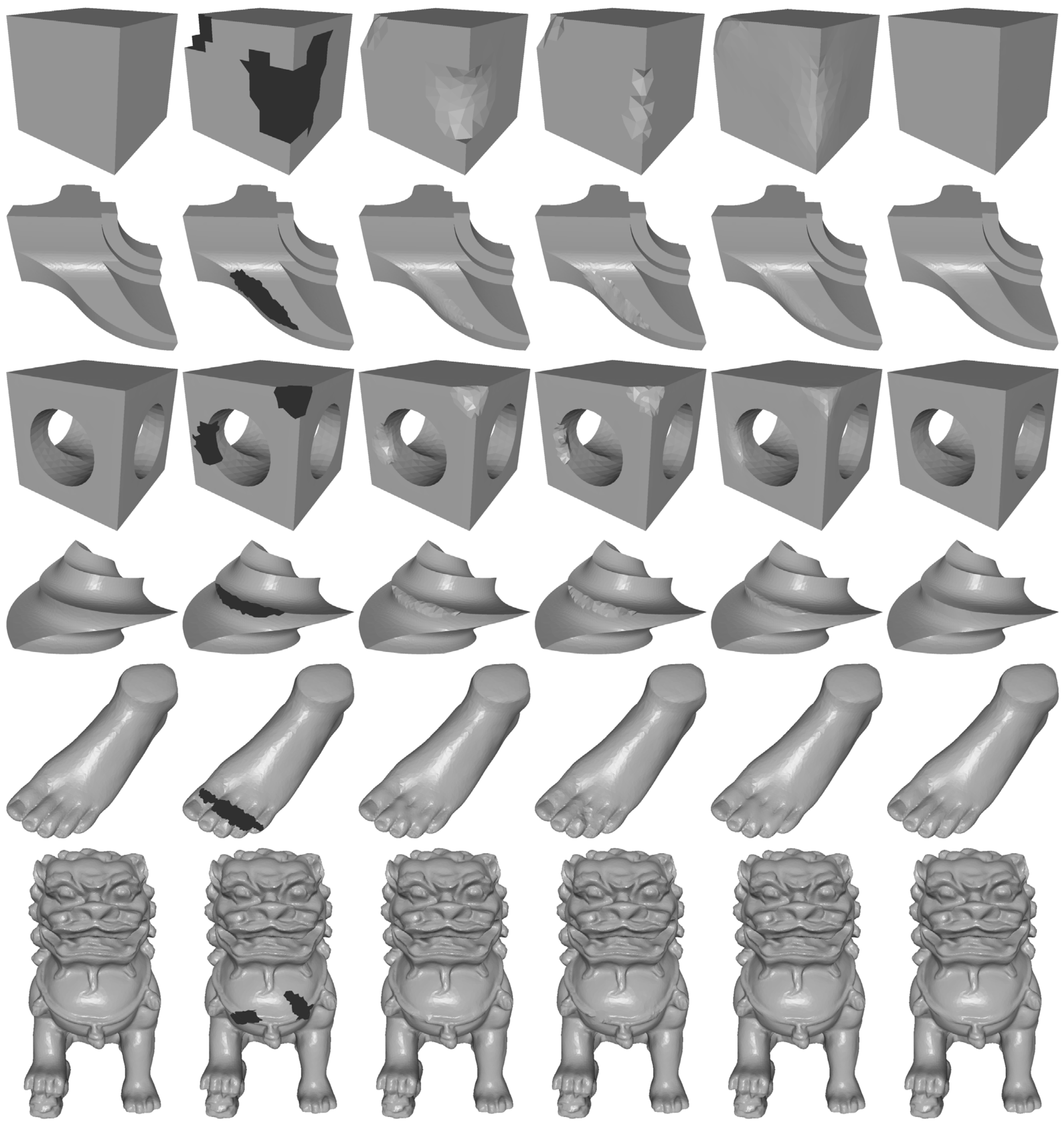

Fig. 12. Inpainting results and comparisons. The original models (a) are edited by eliminating several triangles, thus creating models with hole (b). Inpainting results from different algorithms are shown in (c)–(f). Compared with the original models, our results are more robust to repair the hole area while recovering sharp features and fine details even varying sizes and numbers of holes are considered.

produced by TGV and our method NMSTV look more natural and detail-preserving than those produced by the other methods (see Fig. 11g and h).

To further evaluate the quality of the denoising results, we adopt MSAE to measure mean square angular error between face normals of the denoised mesh and the ground truth, and use the vertex-based Hausdorff distance $E_{v,h}$ to measure the position error between the denoised mesh and the ground truth. These two error metrics are widely suggested in previous works [11,48, 49,64,65]. Table 1 lists MSAE, $E_{v,h}$, number of inverted triangles and the execution time (in seconds) of examples in Fig. 11 from each method. Since the ground truth of each scanning data has self intersections, the statistics of number of inverted triangles for them are omitted. From Table 1, our method achieves the least amount of error with the least overlaps on all types of the tested meshes (CAD, non-CAD, and scanned meshes) in most cases. In addition, the execution time of our method is comparable to that of other methods. As we can see from Table 1, CNR is the fastest method, thanks to its pre-trained neural networks. MSAT is the slowest for synthetic meshes, while NLLR is the slowest for scanned data. MSTV and our NMSTV are slower than CNR, but significantly faster than L^0 and TGV.

Overall, in most cases, our method produces much better results in terms of visual quality and error metrics at reasonable running time, which sufficiently shows the merit of the nonsmooth nonconvex regularization and efficient vertex updating.

Table 2
The max, mean and root mean square (RMS) of Hausdorff distances (10^{-3}) between original shapes and corresponding results from different inpainting methods.

Models	CGAL [66]	MeshFix [68]	MSAT [10]	Ours
Cube	(50.453, 0.704, 4.521)	(32.705, 0.197, 2.148)	(24.819, 2.358, 3.871)	(**4.527**, **1.558**, **2.062**)
Fandisk	(17.560, **0.113**, 1.036)	(26.662, 0.249, 1.944)	(15.775, 0.345, 1.211)	(**9.020**, 0.534, **1.209**)
Ccylinder	(26.814, 0.070, 0.914)	(26.495, 0.062, 0.909)	(19.505, 0.317, 0.800)	(**2.968**, 0.238, **0.485**)
Octa-flower	(9.396, 0.018, 0.313)	(13.078, **0.029**, 0.484)	(8.346, 0.129, 0.431)	(**5.896**, **0.275**, 0.502)
Foot	(4.175, 0.049, 0.333)	(11.487, 0.102, 0.685)	(6.152, 0.225, 0.646)	(**2.962**, **0.042**, **0.208**)
Chinese-lion	(5.694, 0.005, 0.074)	(8.400, 0.008, 0.144)	(5.480, **0.028**, 0.141)	(**5.213**, 0.033, **0.013**)

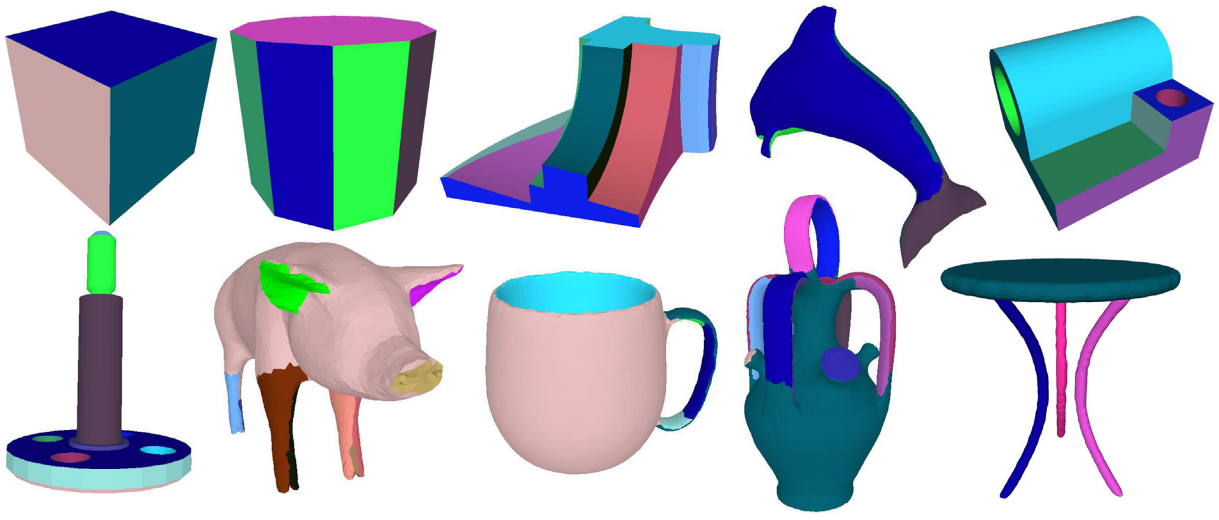
Fig. 13. Our segmentation results on both CAD and non-CAD models. Our method produces segmentations with boundary-preserving and piecewise smooth parts.

6.2. Inpainting

Inpainting aims to fill in the missing areas of a mesh. To perform mesh inpainting, one should first extract the hole boundaries and triangulate the missing areas. In this work, we employ the hole filling algorithm [66] implemented in CGAL [67] to recover the topology of the missing areas. Following [10], we set the data attachment terms α and ω_3 of normal estimation and vertex updating to a large value outside of the missing areas to prevent known vertices from moving, and ω_3 to a very small value inside. To better recover longer features of the inpainted areas, we set β to one tenth of its original value inside the inpainted areas, producing less smooth inpainting results. Finally, we perform the NMSTV normal estimation in Section 4.1 and vertex updating in Section 4.2 for a feature-preserved inpainting result.

We compare our method with three inpainting methods, including CGAL filling [66], MeshFix [68] and MSAT [10]. With a better detail-preserving normal estimation and efficient vertex updating, missing parts can be well reconstructed in a feature-sensitive manner. Visual examples of inpainting results are provided in Fig. 12, which contain different scales of features on CAD and non-CAD surfaces. One can observe that competitors can recover correct topology information without degenerate elements, but they fail to reconstruct the underlying shapes especially for the region of the sharp features and fine details, as shown in Fig. 12b-e. Our results are exhibited in Fig. 12f, from which we can observe that our method is more robust to repair the hole areas while recovering sharp features and fine details even varying sizes and numbers of holes are considered (see the top five examples). For the features with weak edges, such as the last result in Fig. 12f, our method and MSAT still show their superiority on recovering fine details compared with the competitors although the small-scale features are blurred to an extent.

To further evaluate the inpainting deviation from the ground truth, we use the max, mean and root mean square of vertex-based Hausdorff distance to measure the position error between the inpainted mesh and the ground truth, which is given in Table 2. As can be seen, for CAD and non-CAD surfaces, our method achieves the least errors (max, mean and root mean square) on most examples.

Overall, our approach can well preserve local discontinuities, especially sharp edges, with the optimized discontinuity function v and recognize the intrinsic structures of the original shapes while be coherent with the whole remaining shape.

6.3. Segmentation

Mesh segmentation is a fundamental but challenging problem in geometric modeling and computer graphics, and widely used in texture mapping [75], 3D morphing [76,77], simplification [45], shape retrieval [78], skeleton extraction [79] and so on. In this work, we segment meshes into piecewise smooth parts based on geometric information rather than semantic information. In contrast to semantic segmentation benchmarks [80], there is no ground truth available for non-semantic segmentation, which makes numerical evaluation difficult.

To perform this segmentation, we make use of the discontinuity function v on edge function space obtained from our NMSTV solver. Following [10], we first normalize v in the range of [0, 1] and solve a minimum multicut problem in the triangle adjacency graph using the Kernighan Lin method [72]. For each edge e, we define the splitting probability of two adjacent triangles belonging to different segments as the value $1 - v_e$ and force this probability to 0.1% when adjacent triangles have their (regularized) normals further than 15° apart. Note that, our input probability is defined on edge space directly and favors more accurate and smooth segmentation boundaries, which is different

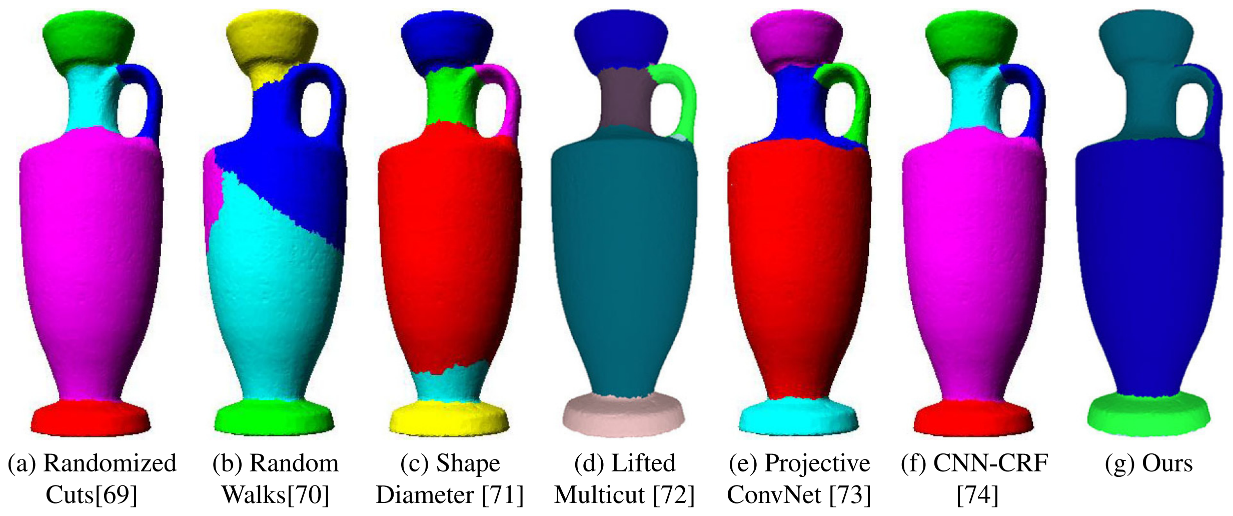

Fig. 14. Comparison of semantic segmentations (a–f) produced by [69–74] with our geometric segmentation (g). Meaningful semantics such as the base, the neck, or the handle of this jar, are produced by semantic segmentations, while our geometric segmentation extracts piecewise smooth segments with clear boundaries emphasizing the underlying geometric information.

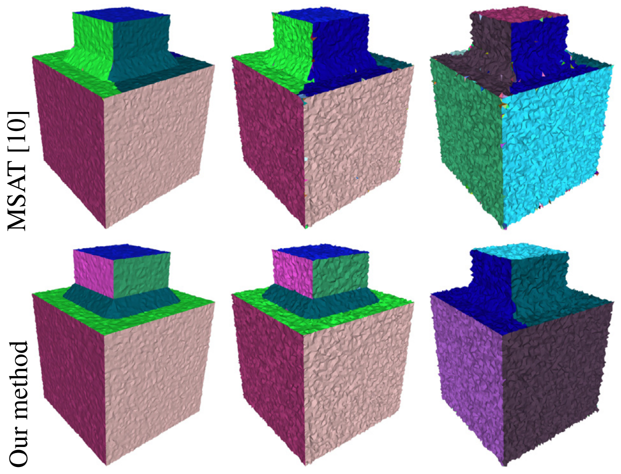

Fig. 15. Comparison of segmentations on meshes with different levels of noise (corrupted by Gaussian noise with $\sigma = 0.1\bar{l}_e$, $\sigma = 0.2\bar{l}_e$ and $\sigma = 0.3\bar{l}_e$, respectively). Our method better recovers discontinuities consistent with the geometric characteristics and achieves smoother segmented parts even to extreme noise levels.

from the one in [10]. Fig. 13 shows our segmentation results on both CAD and non-CAD models. One can observe that our method can recover piecewise smooth segments with clear boundaries.

To further demonstrate the differences between our geometric segmentation and state-of-the-art semantic segmentation methods, we show an example in Fig. 14. Our method produces geometrically consistent segmentations with piecewise smooth parts by minimizing a nonsmooth and nonconvex MSTV energy, while semantic segmentation methods always output semantically meaningful parts from heuristics or learning from large manually segmented Benchmarks [80].

Since the discontinuity function v plays an important role in mesh segmentation, we compare our discontinuities to that of MSAT [10] where v is discretized as a primal 0-form defined on each vertex of the mesh, using meshes with varying amount of noise shown in Fig. 15. In this experiment, we obtain MSAT's splitting probability on edge e by averaging the value $1 - v_e$ of both vertices of the edge, and set probabilities of both two methods to 0.1% when the normal angle between adjacent triangles in optimized mesh is large than 15°. As can be observed that, our recovered discontinuities keep a good consistency with the geometric characteristics, achieving an almost similar value (close to 1) in the smooth region while different values across the sharp edges. Thus, our method ensures piece-wise constant discontinuities to give a good guidance of producing better geometric segmentations and is more robust to noise.

7. Conclusion

In this paper, we have proposed a Mumford–Shah tool for mesh processing via nonsmooth nonconvex regularization, which consists of a feature-preserving normal estimation and efficient vertex updating. Moreover, the nonsmooth nonconvex Mumford–Shah model is able to produce a detail-preserving face normal field to eliminate the staircasing artifacts commonly appeared in the results of TV-based models, and the vertex updating is more robust to noises guided by a feature field. Efficient algorithms have been developed for both optimizations and produced state-of-the-art results for various geometry processing problems such as mesh denoising, mesh inpainting and mesh segmentation.

Some future works are left open. Since our method has satisfied the unit length constraints for face normal vectors by simple projection, we cannot ensure the integrability of face normals. Meanwhile, we do not explicitly prevent the self intersection in our vertex updating. How to incorporate global constraints into the solver to avoid the above limitations simultaneously could be investigated in the future. Although the proposed MS framework has been only considered on the filtering of face normals, it is general enough to be applied for other scenarios such as texture colors, shape operators and other geometric representations such as point clouds and implicit surfaces.

CRediT authorship contribution statement

Chunxue Wang: Methodology, Software, Validation, Data curation, Writing – original draft. **Zheng Liu:** Conceptualization, Formal analysis, Writing – reviewing and editing. **Ligang Liu:** Supervision, Writing – reviewing and editing.

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

We would like to thank authors of competitors for providing their codes or already-trained models. We also thank anonymous reviewers for their constructive suggestions for improving the manuscript. This work was supported by the Youth Science and Technology Foundation of Gansu (20JR5RA050) and the Open Project of Gansu Provincial Research Center for Conservation of Dunhang Cultural Heritage (GDW2021YB05).

References

[1] Bar L, Sochen N, Kiryati N. Semi-blind image restoration via Mumford-Shah regularization. IEEE Trans Image Process 2006;15(2):483–93.
[2] Ben-Ari R, Sochen N. Stereo matching with Mumford-Shah regularization and occlusion handling. IEEE Trans Pattern Anal Mach Intell 2010;32(11):2071–84.
[3] Mumford DB, Shah J. Optimal approximations by piecewise smooth functions and associated variational problems. Comm Pure Appl Math 1989.
[4] Ambrosio L, Tortorelli VM. Approximation of functional depending on jumps by elliptic functional via t-convergence. Comm Pure Appl Math 1990;43(8):999–1036.
[5] Shah J. A common framework for curve evolution, segmentation and anisotropic diffusion. In: Proceedings of IEEE computer society conference on computer vision and pattern recognition. IEEE; 1996, p. 136–42.
[6] Zhang J, Zheng J, Wu C, Cai J. Variational mesh decomposition. ACM Trans Graph 2012;31(3):1–14.
[7] Pokrass J, Bronstein AM, Bronstein MM. A correspondence-less approach to matching of deformable shapes. In: Proceedings of the international conference on scale space and variational methods in computer vision. Springer; 2011, p. 592–603.
[8] Tong W, Tai X. A variational approach for detecting feature lines on meshes. J Comput Math 2016;34(1):87.
[9] Coeurjolly D, Foare M, Gueth P, Lachaud J-O. Piecewise smooth reconstruction of normal vector field on digital data. Comput Graph Forum 2016;35(7):157–67.
[10] Bonneel N, Coeurjolly D, Gueth P, Lachaud J-O. Mumford-Shah mesh processing using the Ambrosio-Tortorelli functional. Comput Graph Forum 2018;37(7):75–85.
[11] Liu Z, Wang W, Zhong S, Zeng B, Liu J, Wang W. Mesh denoising via a novel Mumford–Shah framework. Comput Aided Des 2020;126:102858.
[12] Alicandro R, Braides A, Shah JM. Free-discontinuity problems via functionals involving the L_1-norm of the gradient and their approximations. Interfaces Free Bound 1999;1(1):17–37.
[13] Brook A, Kimmel R, Sochen NA. Variational restoration and edge detection for color images. J Math Imaging Vision 2003;18(3):247–68.
[14] 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.
[15] Rudin LI, Osher S, Fatemi E. Nonlinear total variation based noise removal algorithms. Physica D 1992;60(1–4):259–68.
[16] Geman S, Geman D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans Pattern Anal Mach Intell 1984;(6):721–41.
[17] Geman D, Reynolds G. Constrained restoration and the recovery of discontinuities. IEEE Trans Pattern Anal Mach Intell 1992;14(3):367–83.
[18] Geman D, Yang C. Nonlinear image recovery with half-quadratic regularization. IEEE Trans Image Process 1995;4(7):932–46.
[19] Nikolova M. Analysis of the recovery of edges in images and signals by minimizing nonconvex regularized least-squares. Multiscale Model Simul 2005;4(3):960–91.
[20] Oh S, Woo H, Yun S, Kang M. Non-convex hybrid total variation for image denoising. J Vis Commun Image Represent 2013;24(3):332–44.
[21] Kang M, Kang M, Jung M. Nonconvex higher-order regularization based Rician noise removal with spatially adaptive parameters. J Vis Commun Image Represent 2015;32:180–93.
[22] Kang M, Kang M, Jung M. Total generalized variation based denoising models for ultrasound images. SIAM J Sci Comput 2017;72(1):172–97.
[23] Wang C, Xu L, Liu L. Structure-texture image decomposition via nonconvex total generalized variation and convolutional sparse coding. Vis Comput 2022;1–16.
[24] Chen X, Zhou W. Smoothing nonlinear conjugate gradient method for image restoration using nonsmooth nonconvex minimization. SIAM J Imaging Sci 2010;3(4):765–90.
[25] Bian W, Chen X. Linearly constrained non-Lipschitz optimization for image restoration. SIAM J Imaging Sci 2015;8(4):2294–322.
[26] Hintermüller M, Wu T. Nonconvex TV^q-models in image restoration: Analysis and a trust-region regularization–based superlinearly convergent solver. SIAM J Imaging Sci 2013;6(3):1385–415.
[27] Nikolova M, Ng MK, Tam C-P. Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. IEEE Trans Image Process 2010;19(12):3073–88.
[28] Nikolova M, Ng MK, Zhang S, Ching W-K. Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization. SIAM J Imaging Sci 2008;1(1):2–25.
[29] Ochs P, Dosovitskiy A, Brox T, Pock T. An iterated L_1 algorithm for non-smooth non-convex optimization in computer vision. In: Proceedings of conference on computer vision and pattern recognition. 2013. p. 1759–66.
[30] Ochs P, Dosovitskiy A, Brox T, Pock T. On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. SIAM J Imaging Sci 2015;8(1):331–72.
[31] Robini MC, Zhu Y. Generic half-quadratic optimization for image reconstruction. SIAM J Imaging Sci 2015;8(3):1752–97.
[32] Bergmann R, Chan RH, Hielscher R, Persch J, Steidl G. Restoration of manifold-valued images by half-quadratic minimization. 2015, arXiv preprint arXiv:1505.07029.
[33] Robini MC, Yang F, Zhu Y. Inexact half-quadratic optimization for linear inverse problems. SIAM J Imaging Sci 2018;11(2):1078–133.
[34] Kaloga Y, Foare M, Pustelnik N, Jensen P. Discrete Mumford-Shah on graph for mixing matrix estimation. IEEE Signal Process Lett 2019;26(9):1275–9.
[35] Foare M, Pustelnik N, Condat L. Semi-linearized proximal alternating minimization for a discrete Mumford–Shah model. IEEE Trans Image Process 2019;29:2176–89.
[36] Le HTV, Foare M, Pustelnik N. Proximal based strategies for solving discrete Mumford-Shah with Ambrosio-Tortorelli penalization on edges. IEEE Signal Process Lett 2022;29:952–6.
[37] Strekalovskiy E, Cremers D. Real-time minimization of the piecewise smooth Mumford-Shah functional. In: European conference on computer vision. Springer; 2014, p. 127–41.
[38] Storath M, Weinmann A, Demaret L. Jump-sparse and sparse recovery using Potts functionals. IEEE Trans Signal Process 2014;62(14):3654–66.
[39] Jung M, Kang M. Efficient nonsmooth nonconvex optimization for image restoration and segmentation. SIAM J Sci Comput 2015;62(2):336–70.
[40] Yang L, Pong TK, Chen X. Alternating direction method of multipliers for a class of nonconvex and nonsmooth problems with applications to background/foreground extraction. SIAM J Imaging Sci 2017;10(1):74–110.
[41] Storath M, Weinmann A, Frikel J, Unser M. Joint image reconstruction and segmentation using the Potts model. Inverse Problems 2015;31(2):025003.
[42] Hohm K, Storath M, Weinmann A. An algorithmic framework for Mumford–Shah regularization of inverse problems in imaging. Inverse Problems 2015;31(11):115011.
[43] Weinmann A, Storath M. Iterative Potts and Blake–Zisserman minimization for the recovery of functions with discontinuities from indirect measurements. Proc R Soc Lond Ser A Math Phys Eng Sci 2015;471(2176):20140638.
[44] Yu Y, Zhou K, Xu D, Shi X, Bao H, Guo B, et al. Mesh editing with Poisson-based gradient field manipulation. ACM Trans Graph 2004;644–51.
[45] Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. ACM Trans Graph 2004;905–14.
[46] He L, Schaefer S. Mesh denoising via L_0 minimization. ACM Trans Graph 2013;32(4):1–8.
[47] 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.
[48] 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.
[49] Liu Z, Li Y, Wang W, Liu L, Chen R. Mesh total generalized variation for denoising. IEEE Trans Vis Comput Graphics 2021.
[50] Zorin D. Curvature-based energy for simulation and variational modeling. In: Proceedings of international conference on shape modeling and applications. IEEE; 2005, p. 196–204.
[51] Lu X, Chen W, Schaefer S. Robust mesh denoising via vertex pre-filtering and l1-median normal filtering. Comput Aided Geom Design 2017;54:49–60.
[52] Lu X, Deng Z, Chen W. A robust scheme for feature-preserving mesh denoising. IEEE Trans Vis Comput Graphics 2016;22(3):1181–94.
[53] Lu X, Schaefer S, Luo J, Ma L, He Y. Low rank matrix approximation for 3D geometry filtering. IEEE Trans Vis Comput Graphics 2020.
[54] Tsai A, Yezzi A, Willsky AS. Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification. IEEE Trans Image Process 2001;10(8):1169–86.
[55] 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.
[56] Esedoglu S, Shen J. Digital inpainting based on the Mumford–Shah–Euler image model. European J Appl Math 2002;13(4):353–70.
[57] Vese L, Chan TF. Reduced non-convex functional approximations for image restoration & segmentation. Los Angeles: Department of Mathematics, University of California; 1997.

[58] Sun X, Rosin PL, Martin R, Langbein F. Fast and effective feature-preserving mesh denoising. IEEE Trans Vis Comput Graphics 2007;13(5):925–38.
[59] Bouaziz S, Deuss M, Schwartzburg Y, Weise T, Pauly M. Shape-up: Shaping discrete geometry with projections. Comput Graph Forum 2012;31(5):1657–67.
[60] Zhang J, Deng B, Hong Y, Peng Y, Qin W, Liu L. Static/dynamic filtering for mesh geometry. IEEE Trans Vis Comput Graphics 2018;25(4):1774–87.
[61] Liu Z, Xiao X, Zhong S, Wang W, Li Y, Zhang L, et al. A feature-preserving framework for point cloud denoising. Comput-Aided Des Solid Phys Model 2020;127:102857.
[62] Li X, Zhu L, Fu C-W, Heng P-A. Non-local low-rank normal filtering for mesh denoising. Comput Graph Forum 2018;37(7):155–66.
[63] Wang P-S, Liu Y, Tong X. Mesh denoising via cascaded normal regression. ACM Trans Graph 2016;35(6):232:1–232:12..
[64] Zheng Y, Fu H, Au OK-C, Tai C-L. Bilateral normal filtering for mesh denoising. IEEE Trans Vis Comput Graphics 2010;17(10):1521–30.
[65] Yadav SK, Reitebuch U, Polthier K. Robust and high fidelity mesh denoising. IEEE Trans Vis Comput Graphics 2018;25(6):2304–10.
[66] Liepa P. Filling holes in meshes. In: Proceedings of eurographics/ACM SIGGRAPH symposium on geometry processing. 2003. p. 200–5.
[67] Fogel E, Teillaud M. The computational geometry algorithms library CGAL. ACM Commun Comput Algebra 2014;47(3/4):85–7.
[68] Attene M. A lightweight approach to repairing digitized polygon meshes. Vis Comput 2010;26(11):1393–406.
[69] Golovinskiy A, Funkhouser T. Randomized cuts for 3D mesh analysis. In: Proceedings of ACM SIGGRAPH Asia. 2008. p. 1–12.
[70] Lai Y-K, Hu S-M, Martin RR, Rosin PL. Fast mesh segmentation using random walks. In: Proceedings of the 2008 ACM symposium on solid and physical modeling. 2008. p. 183–91.
[71] Shapira L, Shamir A, Cohen-Or D. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis Comput 2008;24(4):249–59.
[72] Keuper M, Levinkov E, Bonneel N, Lavoué G, Brox T, Andres B. Efficient decomposition of image and mesh graphs by lifted multicuts. In: Proceedings of IEEE international conference on computer vision. 2015. p. 1751–9.
[73] Kalogerakis E, Averkiou M, Maji S, Chaudhuri S. 3D shape segmentation with projective convolutional networks. In: Proceedings of conference on computer vision and pattern recognition. 2017. p. 3779–88.
[74] Abouqora Y, Herouane O, Moumoun L, Gadi T. A hybrid CNN-CRF inference models for 3D mesh segmentation. In: Proceedings of IEEE congress on information science and technology. 2021. p. 296–301.
[75] Lévy B, Petitjean S, Ray N, Maillot J. Least squares conformal maps for automatic texture atlas generation. ACM Trans Graph 2002;21(3):362–71.
[76] Gregory A, Lin MC, Manocha D, Livingston MA, et al. Interactive surface decomposition for polyhedral morphing. Vis Comput 1999;15(9):453–70.
[77] Zöckler M, Stalling D, Hege H-C. Fast and intuitive generation of geometric shape transitions. Vis Comput 2000;16(5):241–53.
[78] Zuckerberger E, Tal A, Shlafman S. Polyhedral surface decomposition with applications. Comput Graph 2002;26(5):733–43.
[79] Katz S, Tal A. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans Graph 2003;22(3):954–61.
[80] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation. ACM Trans Graph 2009;28(3):1–12.