Variational Mesh Denoising Using Total Variation and Piecewise Constant Function Space

Huayan Zhang, Chunlin Wu, Juyong Zhang, and Jiansong Deng

Abstract—Mesh surface denoising is a fundamental problem in geometry processing. The main challenge is to remove noise while preserving sharp features (such as edges and corners) and preventing generating false edges. We propose in this paper to combine total variation (TV) and piecewise constant function space for variational mesh denoising. We first give definitions of piecewise constant function spaces and associated operators. A variational mesh denoising method will then be presented by combining TV and piecewise constant function space. It is proved that, the solution of the variational problem (the key part of the method) is in some sense continuously dependent on its parameter, indicating that the solution is robust to small perturbations of this parameter. To solve the variational problem, we propose an efficient iterative algorithm (with an additional algorithmic parameter) based on variable splitting and augmented Lagrangian method, each step of which has closed form solution. Our denoising method is discussed and compared to several typical existing methods in various aspects. Experimental results show that our method outperforms all the compared methods for both CAD and non-CAD meshes at reasonable costs. It can preserve different levels of features well, and prevent generating false edges in most cases, even with the parameters evaluated by our estimation formulae.

Index Terms—Mesh denoising, piecewise constant function space, total variation, sharp feature

1 Introduction

Triangulated surfaces are widely used in computer graphics. Compared with other surfaces, such as parametric and implicit surfaces, mesh surfaces have their own merits. First, mesh surfaces are quite easy to obtain and convenient to operate. Second, they can approximate surfaces with arbitrary topology and geometry. Third, mesh surfaces are more ready to render on various types of hardwares.

Meshes are usually generated by digital scanning devices and triangulation methods with inevitable measurement errors and algorithm errors. Consequently, most of them are noisy. Therefore, a fundamental problem is to remove noise to obtain high-quality meshes before further processing. The main challenge in this problem is to reduce noise while preserving sharp features (such as edges and corners) and preventing generating false edges. Both noise and features are of high frequency, and they are hard to be distinguished. Most existing methods make use of the neighborhood information to define a weight to detect features. These methods blur sharp features, and even remove small-scale features. The very recent L_0 minimization method [1] adopts the concept of sparsity to remove noise from meshes. This method can preserve sharp features very well, but usually generates false edges at smooth regions on non-CAD meshes, no matter how we adjusted the parameters in our tests. In other words, typical existing mesh denoising methods either fail to preserve sharp features well, or generate false edges.

In this paper, we propose a variational mesh denoising method using piecewise constant function space and TV. TV has been shown very successful in edge-preserving image processing. We here extend it to mesh denoising. This extension is not trivial because we need to carefully choose a proper function space to realize it. So far TV has been incorporated with piecewise linear function spaces in computer graphics. In contrast, by considering that the gradient of the face normal field is sparser than that of the vertex normal field, here we define TV rigorously in piecewise constant function spaces for feature-preserving mesh denoising. Results show that this new combination produces much better denoising results than TV incorporated with piecewise linear function spaces; see Fig. 1. Our method can remove noise efficiently while preserving both sharp and small-scale features. It also prevents generating false edges in most cases by choosing appropriate algorithm parameters, e.g., the optimal parameters and the parameters evaluated by the given estimation formulae. Here TV shows much weaker staircase effect than it applied to image denoising. The possible reason is that, we use TV to penalize the normal field of the surface, while in image denoising it penalizes the image intensity.

- *H. Zhang, J. Zhang, and J. Deng are with the School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China. E-mail: zhy101@mail.ustc.edu.cn, {juyong, dengjs}@ustc.edu.cn.*
- *C. Wu is with the School of Mathematical Sciences, University of NanKai, Tianjin 300071, China. E-mail: chunlin97.wu@gmail.com.*

Manuscript received 6 Feb. 2014; revised 7 Nov. 2014; accepted 21 Jan. 2015. Date of publication 1 Feb. 2015; date of current version 29 May 2015. Recommended for acceptance by A. Sheffer.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2015.2398432

2 Related Work

2.1 Total Variation

Total variation regularization was first introduced in [2] for image denoising. Due to its good edge-preserving property, TV immediately received much attention. It has many extensions such as vectorial TV and high order TV; see, e.g., [3], [4], [5], [6], [7], [8], [9], [10]. TV and its extensions are widely used in image restoration and

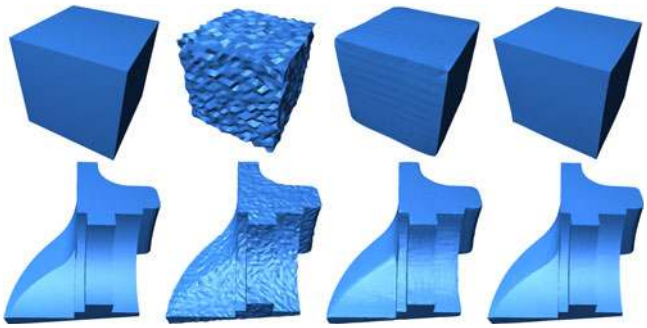


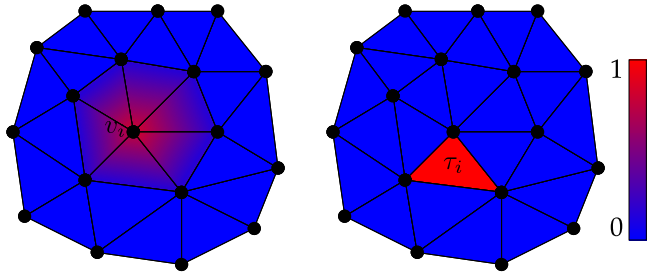
Fig. 1. Comparison between the proposed method (using piecewise constant function space) and the method in [10] (using piecewise linear function space) for mesh denoising. (a) Clean meshes. (b) Noisy meshes (Gaussian noise, standard deviation = 0.2 mean edge length for Cube; standard deviation = 0.1 mean edge length for Fandisk). (c) Denoising results by the method in [10]. (d) Denoising results by the proposed method.

segmentation problems. In image restoration, they are demonstrated very successful to preserve image features such as edges [2], [3], [4], [5], [6], [7], [8]. In image segmentation, TV and vectorial TV play an important role in convexifying variational image segmentation models [11], [12], overcoming the main difficulty of variational segmentation problems. Although TV related regularizers are non-differentiable and thus hard to solve by conventional optimization methods, there have been proposed very recently many efficient algorithms; see, e.g., [13], [14], [15], [16], [17], [18]. For more details, readers can refer to a very recent and interesting review [19].

2.2 Piecewise Linear Function Space and Piecewise Constant Function Space

In recent years, piecewise linear function space, as a basic finite element space in numerical PDE [20], has achieved great successes in many computer graphics applications like mesh smoothing, image smoothing and segmentation on meshes, derivation of discrete differential operators and texture generation; see, e.g., [21], [22], [23], [24], [25], [26], [10], [27]. In particular, piecewise linear function space combined with TV has been studied [10] for image restoration and segmentation on meshes. However, we found that, in feature-preserving mesh denoising, piecewise linear function space has some limitations, compared to piecewise constant function space which is related to piecewise constant finite element method in numerical PDE. For clarity, we would like to list the main differences between these spaces.

- First, the basis functions of these spaces are quite different. See Fig. 2. A basis function of piecewise linear function space used in [26] and [10] is a hat function, which is defined *vertex-by-vertex*. These basis can be used to interpolate data defined on mesh vertices, such as vertex color and normal fields. However, a basis function of piecewise constant function space is assigned value 1 inside a triangle and 0 otherwise. This is a *triangle-by-triangle* definition. These basis can be used to interpolate data defined on mesh triangles, such as triangle normal field.



(a) piecewise linear basis ϕ_i on vertex v_i (b) piecewise constant basis ϕ_i on triangle τ_i

Fig. 2. Basis of piecewise linear function space and piecewise constant function space.

- Second, the definitions of the gradient operators of these function spaces are quite different. See [10] and Section 3.
- Third, piecewise linear function spaces are suitable for processing data defined on mesh vertices, while piecewise constant function spaces are suitable for data defined on mesh triangles. When applied to normal-based mesh denoising, piecewise linear function space requires the input to be vertex normals, while the input of piecewise constant function space is face normals. The vertex normals are averaged from face normals. The gradient of this smoothed normal field is much less sparse than that of face normal field. Therefore, it is more appropriate to apply TV to face normal field.
- Finally, applying TV to these two spaces indeed has quite different mesh denoising effects. See Fig. 1. The method using piecewise linear function space in [10] cannot filter noise effectively and fails in preserving corners of the meshes.

2.3 Mesh Denoising

There has been much work focusing on mesh denoising recently. Excellent results and fast speed are key factors to evaluate mesh denoising methods. No method can be efficient for all meshes so far.

The classical Laplacian smoothing method [28] is simple and fast. However, it would cause mesh shrinkage while removing the noise. In order to solve this shrinkage problem, Taubin [29] proposed a signal processing approach to mesh denoising. Taubin's approach uses two filters. One is to smooth the mesh while the other is to prevent shrinkage. The method is unstable, if parameters are not chosen properly. Desbrun et al. [21] introduced the mean curvature flow for mesh fairing using implicit discretization. The method rescales the mesh to prevent shrinkage. These two approaches overcome the shrinkage effect of Laplacian smoothing, but introduce new mesh distortions. Also, these methods can be understood as isotropic filtering methods, which do not consider the surface features and the directions of the features.

To better keep sharp features, several new types of methods were proposed recently. The first type is based on nonlinear and anisotropic diffusion as proposed in [23], [22], [30], [31], [32]. The first four methods consider the sharp features such as curvature tensors of meshes. The method in [32] based on the non-local image denoising schemes proposes a powerful mesh denoising method, which obtains

the new vertex position by the nonlinear weighted mean of local neighborhood vertices. The method is quite simple. Yet, its computational cost is a little high.

The second type is based on bilateral filtering, which extends the bilateral filters in image processing. See, e.g., [33], [34]. Bilateral mesh filtering is a one-stage iterative approach. This type of methods use normals to estimate the weights in the filters. The weights cannot be estimated accurately due to the noise, even in the method using Gaussian mollification [34]. Consequently, these methods cannot always preserve sharp features.

The third type is based on normal filtering and vertex updating, which is a type of two-stage methods. See [37], [38], [39], [40], [41], [42], and the third method in [35]. These two-stage methods first filter the face normals (i.e., triangle normals) and then update the vertex positions according to the filtered face normals. They can preserve most of sharp features. However, most of them filter face normals by averaging neighborhood face normals. As averaging weights cannot be computed accurately in relatively flat regions, these methods fail to recover small-scale features and will blur these features. We mention that this type of methods are also applied to image denoising in [36].

The fourth type is of one-stage methods, which include surface reconstruction and decimation by combining both vertex and normal regularization [43], [44] and [1]. The former two propose to minimize energies consisting of both vertex position error and normal error. They provide good surface reconstruction. However, the method in [43] cannot preserve sharp features well while the method in [44] is too slow. The latter proposes a new denoising model, which combines a new defined edge operator and L_0 minimization. The method can get good denoising results. However, it will produce severe staircase effect, especially for non-CAD meshes.

The final type of methods is based on vertex classification and applying different regularization to different vertices. See, e.g., [45] and [46]. The authors of [45] proposed to use consistent subneighborhoods to classify vertices. For different classes of vertices, they chose different denoising methods. This approach can preserve features very well. However, it is a little slow. The method of [46] is an iterative approach combing pre-filtering, feature detection, and L_1-based feature recovery. In each step, it first filters the noisy mesh to get a base mesh by a global Laplacian denoising scheme. An L_1-based optimization is then used to detect and recover sharp features. The method can recover sharp features well. However, it sometimes produces wavy edges and its computational cost is too high.

The mesh denoising method put forward in this paper is a two-stage approach based on face normal filtering and vertex updating, like [37], [38], [39]. Therein, the face normal filtering, as a key step of two-stage approaches, is done by a minimization problem combing total variation and piecewise constant function space.

2.4 Our Contribution and Paper Organization

The contributions of the paper are as follows:

- We give a systematic formulation of piecewise constant function spaces and associated differential operators on triangulated manifolds.
- We propose a new variational mesh denoising method based on these operators and TV. It is proved that the normal filtering variational model has continuous dependency property on its parameter.
- An efficient algorithm based on augmented Lagrangian method (ALM) is presented to solve the proposed variational problem.
- Our method is discussed and compared to typical existing methods in various aspects including parameter settings, denoising effects, and computational efficiency. Two formulae are given to set parameters automatically for meshes corrupted by Gaussian noise. Our method outperforms existing methods for both CAD and non-CAD meshes at reasonable CPU costs. Also, our algorithm is much more robust to small perturbations of parameters, compared to the recent L_0 minimization method.

The remainder of the paper is organized as follows. Section 3 gives the definitions of the operators and TV seminorms on piecewise constant function spaces. In Section 4 we present a variational mesh denoising method. The solution of the normal filtering variational model is shown continuously dependent on its parameter. We also propose an iterative algorithm to solve the model. In Section 5 our denoising method is discussed and compared to typical existing methods in various aspects including parameter settings, denoising effects, and computational efficiency. Conclusion and future work are given in Section 6.

3 Piecewise Constant Function Spaces and Operators

In this section, we introduce some notations followed by definitions of piecewise constant function spaces and associated differential operators on triangulated surfaces.

3.1 Notations

Assume $M \subset \mathbb{R}^3$ to be a compact triangulated surface of arbitrary topology with no degenerate triangles. The set of vertices, edges and triangles of M are denoted as $\{v_i : i = 0, 1, \ldots, \mathrm{V}-1\}$, $\{e_i : i = 0, 1, \ldots, \mathrm{E}-1\}$ and $\{\tau_i : i = 0, 1, \ldots, \mathrm{T}-1\}$, respectively. Here V, E and T are the numbers of vertices, edges and triangles, respectively. If v is an endpoint of an edge e, then we denote it as $v \prec e$. Similarly, that e is an edge of a triangle τ is denoted as $e \prec \tau$; that v is a vertex of a triangle τ is denoted as $v \prec \tau$. Let $D_1(i)$ be the 1-disk of the vertex v_i. $D_1(i)$ is the set of triangles containing v_i.

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

3.2 Piecewise Constant Function Spaces and Operators

We define the space $V_M = \mathbb{R}^{\mathrm{T}}$, which is isomorphic to the piecewise constant function space over M. For example, $u = (u_0, u_1, \ldots, u_{\mathrm{T}-1}) \in V_M$. It means that the value of u

restricted on the triangle τ is u_τ, which is written as $u|_\tau$ sometimes.

For any $u^1, u^2, u \in V_M$, we define the inner product and norm as follows:

$$(u^1, u^2)_{V_M} = \sum_\tau u^1|_\tau u^2|_\tau s_\tau, \quad \|u\|_{V_M} = \sqrt{(u,u)_{V_M}}, \tag{1}$$

where s_τ is the area of triangle τ.

For any $u \in V_M$, we define the jump of u over an edge e as

$$[u]_e = \begin{cases} \sum_{e \prec \tau} u|_\tau \mathrm{sgn}(e,\tau), & e \notin \partial M \\ 0, & e \subseteq \partial M. \end{cases} \tag{2}$$

Due to the piecewise constant function space, the gradient operator can be defined as

$$\nabla : u \to \nabla u, \quad \nabla u|_e = [u]_e, \ \forall e, \ \text{for } u \in V_M.$$

It can be regarded as the signed amplitude of the usual vector definition of the gradient. However, in real computation this simplified definition is enough. In the vector definition, one need choose a tangent space at an edge. This tangent space (at an edge) is ambiguous. A discussion on this can be found in the supplemental file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2015.2398432.

We then denote the range of ∇ by Q_M, i.e., $Q_M = \mathrm{Range}(\nabla)$. The Q_M space is equipped with the following inner product and norm:

$$(p^1, p^2)_{Q_M} = \sum_e p^1|_e p^2|_e l_e, \quad \|p\|_{Q_M} = \sqrt{(p,p)_{Q_M}} \tag{3}$$

for $p^1, p^2, p \in Q_M$, where l_e is the length of the edge e. The divergence operator, $\mathrm{div} : \ Q_M \to V_M$, as the adjoint operator of $-\nabla$, has the following form:

$$(\mathrm{div} p)|_\tau = \frac{-1}{s_\tau} \sum_{e \prec \tau, e \nsubseteq \partial M} l_e p|_e \mathrm{sgn}(e,\tau), \ \text{for } p \in Q_M. \tag{4}$$

Given $u \in V_M$, the total variation of u is thus

$$R_{\mathrm{tv}}(\nabla u) = (\mathrm{TV})(u) = \sum_e l_e |(\nabla u)|_e| = \sum_e l_e |[u]_e|. \tag{5}$$

Note that including the edge length l_e in the above formulae meets the perimeter formulae defined using total variation of the characteristic function. This is a rigorous definition of total variation.

In many applications we need to handle vectorial data. We now extend the above definitions to vectorial case. Two spaces $\mathbf{V}_M$, $\mathbf{Q}_M$ are defined as follows:

$$\mathbf{V}_M = \underbrace{V_M \times \cdots \times V_M}_{\mathfrak{N}}, \mathbf{Q}_M = \underbrace{Q_M \times \cdots \times Q_M}_{\mathfrak{N}}$$

for $\mathfrak{N}$-channel data.

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

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

$$(\mathbf{p}^1, \mathbf{p}^2)_{\mathbf{Q}_M} = \sum_{1 \le i \le \mathfrak{N}} (p_i^1, p_i^2)_{Q_M}, \quad \|\mathbf{p}\|_{\mathbf{Q}_M} = \sqrt{(\mathbf{p},\mathbf{p})_{\mathbf{Q}_M}}.$$

$\nabla \mathbf{u}$ and $\mathrm{div}\mathbf{p}$ are computed channel by channel. For $\mathbf{u} \in \mathbf{V}_M$, the vectorial total variation is

$$R_{\mathrm{vtv}}(\nabla \mathbf{u}) = (\mathrm{TV})(\mathbf{u}) = \sum_e \sqrt{\sum_{i=1}^{\mathfrak{N}} |[u_i]_e|^2} l_e. \tag{6}$$

These spaces and operators can be used in mesh denoising, facet-based segmentation and simplification. In this paper we focus on the application in mesh denoising.

4 Mesh Denoising Using Total Variation and Piecewise Constant Function Space

As mentioned before, the key difficulty in mesh denoising is, while removing the noise, to preserve surface features (such as sharp edges) without generating false features. Most existing methods are either based on vertex smoothing directly or based on face normal filtering using weights for feature detection. Since small-scale features are hard to detect accurately on noisy meshes, these methods fail to preserve small-scale features; see Figs. 8b, 8c, 8d, and 8e. The very recent L_0 minimization method can preserve sharp edges well, but usually generates false edges even with optimal parameters; see the bunny example in Fig. 15. The L_0 minimization seems to produce over-sparse solutions. By piecewise constant function space and TV (TV is in some sense an L_1 minimization and the solution is less sparse than L_0 minimization), here we propose a variational mesh denoising method to try to overcome these problems.

4.1 Variational Denoising Method

Similar to most approaches based on face normal filtering, our method also has two stages, i.e., face normal filtering followed by vertex updating. Therein face normal filtering is the key step.

Given a noisy mesh M^{in}, its face normals are also noisy, and we denote the noisy face normals as $\mathbf{N}^{in}$. In the first step of our method, we filter $\mathbf{N}^{in}$ through our variational model which is solved by augmented Lagrangian method. In the second step, we update the vertex positions from the filtered face normals.

4.1.1 Normal Filtering

To filter the noisy face normals $\mathbf{N}^{in}$, we propose the following variational model:

$$\min_{\mathbf{N} \in C_{\mathbf{N}}} \left\{ R_{\mathrm{wvtv}}(\nabla \mathbf{N}) + \frac{\alpha}{2} \|\mathbf{N} - \mathbf{N}^{in}\|^2_{\mathbf{V}_M} \right\}, \tag{7}$$

where

$$R_{\mathrm{wvtv}}(\nabla \mathbf{N}) = \sum_e w_e \sqrt{\sum_{i=1}^{3} |[N_i]_e|^2} l_e,$$

$$C_{\mathbf{N}} = \{\mathbf{N} \in \mathbf{V}_M : \ \|\mathbf{N}_\tau\|_2 = 1, \ \ \forall \tau\},$$

with w_e as a weight to be detailed below. This constrained problem is equivalent to

$$\min_{\mathbf{N}\in\mathbf{V}_M} \Big\{ R_{\text{wvtv}}(\nabla \mathbf{N}) + \frac{\alpha}{2}\|\mathbf{N} - \mathbf{N}^{in}\|^2_{\mathbf{V}_M} + \chi(\mathbf{N}) \Big\}, \tag{8}$$

where

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

The first term in (8) is a weighted vectorial total variation of the face normals for regularization. The reason to use TV of the face normals as regularization is that the sharp features are sparse on a clean mesh. The gradient of the face normal field is a proper quantity to measure this sparsity. The weight w_e is defined as

$$w_e = \exp\big(-\|N_{e,1} - N_{e,2}\|^4\big), \tag{9}$$

where $N_{e,1}$ and $N_{e,2}$ are normals of the faces sharing a common edge e. This weight can improve preserving sharp features, since it penalizes these features less than smooth regions of the surface [47]. The third term is to constrain the normal vectors to be unit vectors. The second term of (8) is a data fidelity term, guaranteeing that the resulted normal $\mathbf{N}$ is not too far away from $\mathbf{N}^{in}$. α is a positive parameter balancing these two terms. If $\alpha = 0$, the solution is any constant vector field. If $\alpha = +\infty$, the solution is $\mathbf{N}^{in}$. For α, we have the following theorem (See the supplemental file for a proof, available online).

Theorem 1. *Let* $E_\alpha(\mathbf{N}) = R_{\text{wvtv}}(\nabla\mathbf{N}) + \frac{\alpha}{2}\|\mathbf{N} - \mathbf{N}^{in}\|^2_{\mathbf{V}_M} + \chi(\mathbf{N})$ *and* $\mathcal{N}_\alpha = \{\mathbf{N}_\alpha : E_\alpha(\mathbf{N}_\alpha) = \min_{\mathbf{N}} E_\alpha(\mathbf{N}) = E^*_\alpha\}$. *Then* $\forall \mathbf{N}_{\alpha+\delta\alpha} \in \mathcal{N}_{\alpha+\delta\alpha}$, *any cluster point of* $\mathbf{N}_{\alpha+\delta\alpha}$ *as* $\delta\alpha \to 0$ *is in* $\mathcal{N}_\alpha$.

This means that the solution of (8) is in some sense continuously dependent on α. When α changes little, each new solution will be around one of the old solutions (Note that the solution of (8) is usually not unique due to the non-convexity). Hence the denoising results can be expected not sensitive to small perturbations of α; see Fig. 16.

4.1.2 Vertex Updating

After optimizing the face normals through (8), we update the vertex positions using the iterative updating method proposed by Sun et al. [37]. The iteration number is fixed as 35 in our experiments. Since this is not our contribution, we leave the discussion on the influences of the iteration number to the supplemental file, available online.

4.2 Augmented Lagrangian Method for Solving the Normal Filtering Model (8)

Like the absolute value function, the first term in (8) is nondifferentiable. Consequently, it is difficult to solve by conventional methods. Recently, variable splitting and augmented Lagrangian method [48] have been shown very successful for such problems [18], [10]. It should be mentioned that many other related methods for solving this type of problems can be found in [18]. In the following, we apply ALM to our problem (8).

We first introduce a new variable $\mathbf{p} \in \mathbf{Q}_M$ and rewrite the problem (8) as

$$\begin{aligned} \min_{\mathbf{N}\in\mathbf{V}_M, \mathbf{p}\in\mathbf{Q}_M} & \Big\{ R_{\text{wvtv}}(\mathbf{p}) + \frac{\alpha}{2}\|\mathbf{N} - \mathbf{N}^{in}\|^2_{\mathbf{V}_M} + \chi(\mathbf{N}) \Big\} \\ & \text{s.t. } \mathbf{p} = \nabla\mathbf{N}. \end{aligned} \tag{10}$$

For this constrained optimization problem, we then introduce the following augmented Lagrangian function:

$$\begin{aligned} \mathcal{L}_{\text{wvtv}}(\mathbf{N}, \mathbf{p}; \lambda) =& R_{\text{wvtv}}(\mathbf{p}) + \frac{\alpha}{2}\|\mathbf{N} - \mathbf{N}^{in}\|^2_{\mathbf{V}_M} + \chi(\mathbf{N}) \\ & + (\lambda, \mathbf{p} - \nabla\mathbf{N})_{\mathbf{Q}_M} + \frac{r}{2}\|\mathbf{p} - \nabla\mathbf{N}\|^2_{\mathbf{Q}_M}, \end{aligned} \tag{11}$$

where λ is a Lagrange multiplier and r is a positive real number.

The augmented Lagrangian method is an iterative method; see [48], [18], [10]. In each iteration we need to solve a minimization problem for a fixed multiplier (For computational efficiency the weight w_e is specially treated; see the following remark). This minimization problem can be separated into two subproblems:

- The **N**-sub problem: for a given **p**,

$$\min_{\mathbf{N}\in\mathbf{V}_M} \frac{\alpha}{2}\|\mathbf{N} - \mathbf{N}^{in}\|^2_{\mathbf{V}_M} - (\lambda, \nabla\mathbf{N})_{\mathbf{Q}_M} + \frac{r}{2}\|\mathbf{p} - \nabla\mathbf{N}\|^2_{\mathbf{Q}_M} + \chi(\mathbf{N}). \tag{12}$$

- The **p**-sub problem: for a given **N**,

$$\min_{\mathbf{p}\in\mathbf{Q}_M} R_{\text{wvtv}}(\mathbf{p}) + (\lambda, \mathbf{p})_{\mathbf{Q}_M} + \frac{r}{2}\|\mathbf{p} - \nabla\mathbf{N}\|^2_{\mathbf{Q}_M}. \tag{13}$$

The **N**-sub problem (12) is a quadratic programming if we ignore $\chi(\mathbf{N})$. Therefore, we solve

$$\min_{\mathbf{N}\in\mathbf{V}_M} \frac{\alpha}{2}\|\mathbf{N} - \mathbf{N}^{in}\|^2_{\mathbf{V}_M} - (\lambda, \nabla\mathbf{N})_{\mathbf{Q}_M} + \frac{r}{2}\|\mathbf{p} - \nabla\mathbf{N}\|^2_{\mathbf{Q}_M}, \tag{14}$$

first and then normalize the solution. The solution of the above problem can be achieved by a sparse linear system. There are many existing libraries such as MKL, Taucs and Eigen for solving linear systems. We remark that this strategy can only approximate the solution of the original **N**-sub problem (12). However, due to the non-convexity of the (8) and the iterative algorithm, it seems no need to solve the **N**-sub problem exactly. Experiments show that our simple strategy does work.

The **p**-sub problem (13) is separable and can be reformulated as an edge-by-edge problem. That is, for an edge e, we have the following simplified form:

$$\min_{\mathbf{p}_e} w_e|\mathbf{p}_e| + (\lambda_e, \mathbf{p}_e) + \frac{r}{2}|\mathbf{p}_e - (\nabla\mathbf{N})_e|^2,$$

which has a closed form solution as

$$\mathbf{p}_e = \begin{cases} (1 - \frac{w_e}{r|\mathbf{w}_e|})\mathbf{w}_e, & |\mathbf{w}_e| > \frac{w_e}{r} \\ 0, & |\mathbf{w}_e| \le \frac{w_e}{r}, \end{cases} \tag{15}$$

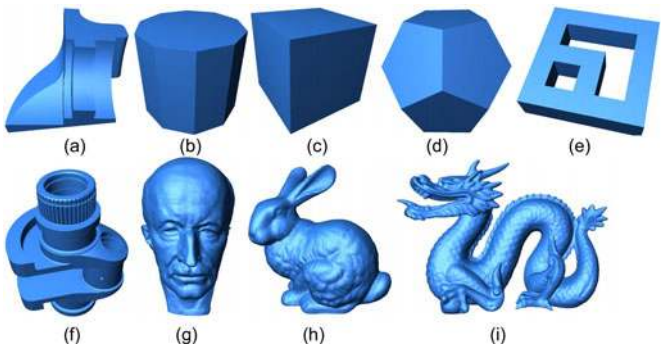


Fig. 3. The clean meshes tested in the paper. (a) Fandisk. (b) Prism. (c) Cube. (d) Dodecahedron. (e) Doubletorus. (f) Crank. (g) Head. (h) Bunny. (i) Dragon.

where

$$\mathbf{w} = \nabla \mathbf{N} - \frac{\lambda}{r}.$$

Remark. In each iteration of augmented Lagrangian method, w_e is computed by the normal $\mathbf{N}$ evaluated at previous step.

The overall algorithm is detailed in Algorithm 1.

Algorithm 1. ALM for mesh denoising

1. **Normal filtering**;
   Initialization: $\lambda^0 = 0, \mathbf{p}^{-1} = 0, \mathbf{N}^{-1} = 0,$
   $k = -1, K = 70, \epsilon = 1e-5;$
   Repeat
   (a) N-sub problem
   For fixed $(\lambda^k, \mathbf{p}^{k-1})$ solve
   three sparse linear systems by (14);
   normalize($\mathbf{N}^k$);
   (b) p-sub problem
   For fixed $(\lambda^k, \mathbf{N}^k)$ compute $\mathbf{p}^k$ by (15);
   Compute w_e through (9);
   (c) Update Lagrange multipliers;
   $\lambda^{k+1} = \lambda^k + r(\mathbf{p}^k - \nabla \mathbf{N}^k)$;
   Until($\|\mathbf{N}^k - \mathbf{N}^{k-1}\|^2_{\mathbf{v}_M} < \epsilon$ or $k \geq K$)
2. **Update the vertex position**;

5 Experiments and Discussion

In this section, we present numerical experiments on both synthetic and real data. The synthetic data are generated by clean meshes with additive Gaussian noise (zero-mean Gaussian functions with different deviations σ's multiplied by mean edge length of the input mesh) or taken from the test data of [39]. The clean meshes for synthetic data are listed in Fig. 3; See Table 1 for mesh sizes. The real data are downloaded from the internet. All these meshes are normalized before test. We implemented all the algorithms tested in this paper (except Wang et al. [46]) by Microsoft Visual Studio 2010. All the examples (except the results of Fig. 11 Column 3) are tested on a laptop with Intel Corei3 and 4 GB RAM and all models are rendered using flat shading. We will discuss our algorithm from various aspects, including influences of parameters and computational efficiency. In particular, we can find two formulae to automatically compute the parameters of our algorithm for Gaussian noise removal by fitting the parameter values in lots of experiments. Our method is also compared to previous techniques by visual effects and quantitative errors.

5.1 Parameters

To our knowledge, most denoising algorithms have at least two parameters, which need to be manually tuned except [1]. Our algorithm also has two parameters: α and r. Since the main computational cost of our algorithm is the step 1(a) of Algorithm 1, we keep α and r unchanged during the iteration and hence the coefficient matrix is fixed. To study the influence of the parameters, we did the experiments on almost all synthetic mesh surfaces used in mesh denoising papers. The conclusions on parameter tests are summarized as follows.

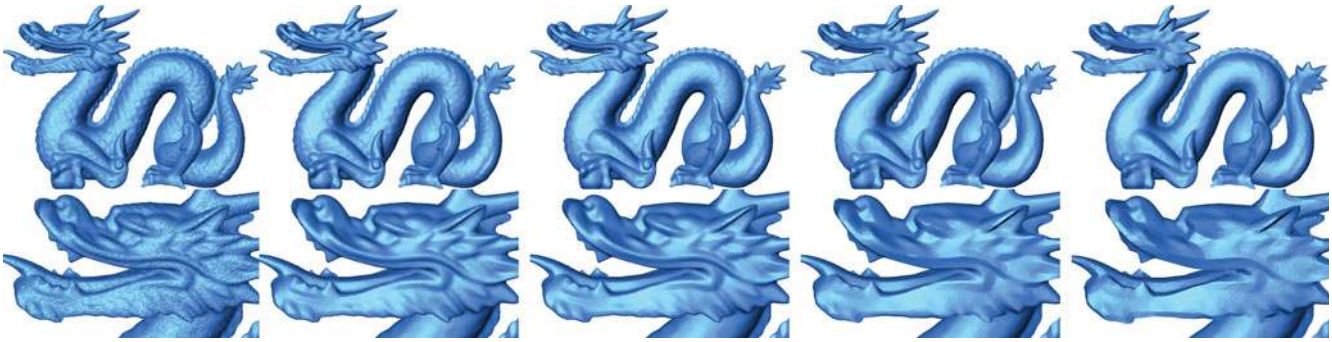
First, α controls the denoising effect and smoothness of the result surface. The smaller the α is, the smoother the denoising result is. If α is too large, the algorithm fails to remove noise. If α is too small, the mesh will become oversmoothed and some features will be lost. Fig. 4 shows an example of the results of different α with fixed r. We can discover that the details of the dragon mesh disappear when α decreases. For each noisy mesh, there exist a range of α for our algorithm giving good denoising results, which is an interval around the optimal value (giving visually optimal denoising result).

Second, r also controls the denoising effect. The larger r is, the more staircase effect (with false edges) the result has. If r is too large, the surface will be smoothed. If r is too small, there are some noise left on the surface. Fig. 5 presents some denoising results of different r with fixed α. Again, for each noisy mesh, there exist a range of r for our algorithm giving good denoising results, which is an interval around the optimal value (giving visually optimal denoising result).

Third, the parameters for our algorithm giving visually optimal denoising result depend on the input noisy mesh. These optimal parameters usually need to be tuned manually. Here we provide two estimation formulae to automatically compute them, which give quite promising results for meshes corrupted by Gaussian noise and for real mesh data. The formulae are obtained by summarizing lots of

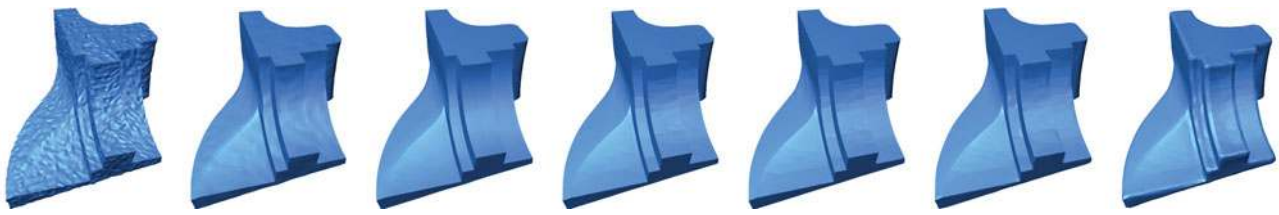
TABLE 1
The Number of Vertices and Triangles of Meshes in Fig. 3

Mesh	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
V	6,475	4,562	2,402	4,610	2,628	50,012	30,942	34,817	422,631
T	12,946	9,120	4,800	9,216	5,376	100,056	61,880	69,630	845,478



(a) Noisy (b) $\alpha = 5000$ (c) $\alpha = 3000$ (d) $\alpha = 1000$ (e) $\alpha = 500$

Fig. 4. Denoising results for different α with r computed by Eq. (17). Gaussian noise, standard deviation $= 0.1$ mean edge length.



(a) Noisy (b) $r = 0.1$ (c) $r = 1$ (d) $r = 5$ (e) $r = 20$ (f) $r = 100$ (g) $r = 300$

Fig. 5. Denoising results for different r with α computed by Eq. (16). Gaussian noise, standard deviation $= 0.1$ mean edge length.

experimental data. We tested almost all synthetic mesh surfaces used in mesh denoising papers with noise level $[0.1, 0.3]$ and found optimal values for α and r. By "optimal value" here, we mean that the proposed algorithm with these parameter values gave visually best denoising results. According to our observation, these parameter values depend on $\frac{\mathrm{E}}{\mathrm{tvEnergy}}$, where E is the number of edges and tvEnergy is the TV of the noisy mesh defined in (6). These values are used to construct fitting functions of $\frac{\mathrm{E}}{\mathrm{tvEnergy}}$. Taking into account that CAD meshes and non-CAD meshes usually have different features, we consider to use spline functions to fit the optimal parameter values. Almost all noisy CAD mesh models tested here have $\frac{\mathrm{E}}{\mathrm{tvEnergy}} \leq 350$ while other models have $\frac{\mathrm{E}}{\mathrm{tvEnergy}} \geq 350$. We choose to use piecewise continuous polynomials of degree 3 as the fitting function with 350 as the common end point of the argument of the two polynomials. Fig. 6 shows the corresponding fitting curves by 32 models with different levels of Gaussian noise. The concrete formulae for computing α and r are presented in (16) and (17). We used these formulae to determine the parameter values in the examples shown in this paper except those especially mentioned. As one can see, these parameter values give quite promising denoising results.

$$\alpha = \begin{cases} 2.4821014 \times 10^{-5}x^3 - 0.00791059836x^2 \\ \quad +1.42844258x - 4, & x \leq 350 \\ -5.81273391 \times 10^{-7}x^3 + 0.00226508x^2 \\ \quad -0.2783605x + 436, & x \geq 350 \end{cases} \tag{16}$$

$$r = \begin{cases} -5.09 \times 10^{-7}x^3 - 2.85422456 \times 10^{-4}x^2 \\ \quad -0.0553458x - 6.1, & x \leq 350 \\ -2.51922 \times 10^{-10}x^3 + 1.3952 \times 10^{-6}x^2 \\ \quad -0.0011917x + 0.3408, & x \geq 350. \end{cases} \tag{17}$$

Finally, the denoising result seems in some sense continuously dependent on the parameters. On one hand, we have theoretically proved in Theorem 1 the continuous dependency of the solution on α. On the other hand, according to

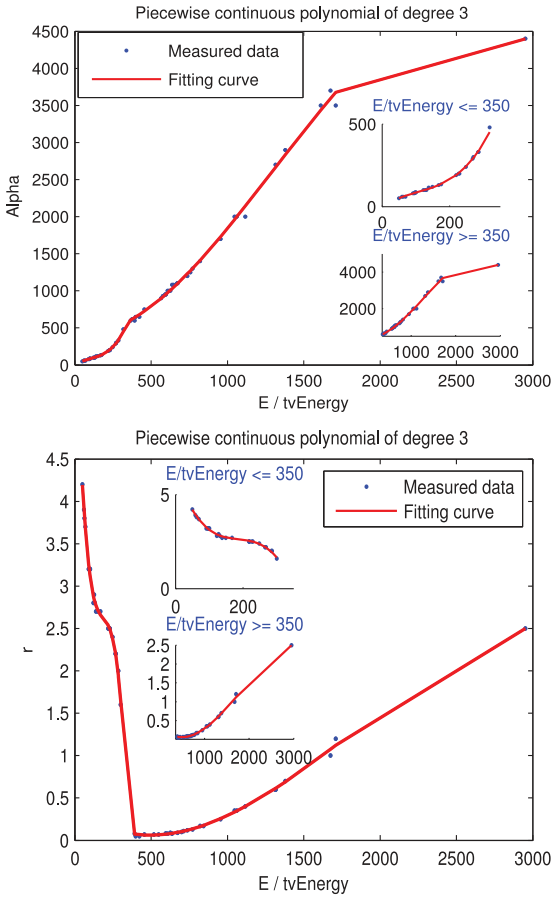


Fig. 6. Fitting parameter α and r using cubic spline.

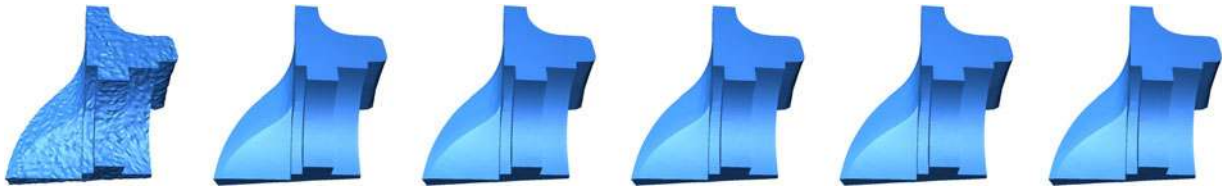


Fig. 7. Influence of small perturbations of parameter r by estimation formulae.

our tests, the denoising result of our algorithm changes little when the parameters have small perturbations. See Figs. 7 and 16. This is important, because continuous dependency is helpful for controlling the parameter tuning procedure. In contrast, the very recent L_0 minimization method [1] seems not having this property and its result is quite sensitive to parameter perturbations; see Fig. 16.

5.2 Examples and Comparisons to Other Algorithms

In this section, we present some examples and compare our algorithm with other typical existing algorithms both visually and quantitatively. In particular, we compare our method to fuzzy vector median filter (FVM) [38], Sun et al. [37], Local algorithm in Zheng et al. [39], Global algorithm in Zheng et al. [39], L_0 minimization [1], Bilateral filtering method [33] and Wang et al. [46]. These methods are sometimes abbreviated as Ours, FVM [38], Sun [37], ZhengLa [39], ZhengGa [39], L_0 [1], Bilateral [33] and Wang [46] in the following. All these methods have parameters, which were set by the following criterion except those in Figs. 12 and 15. For our method, the parameters were computed through Eqs. (16) and (17). For L_0[1], we obtained the parameters by the default formulae given in [1]. The results of Wang[46] were provided by its authors. For the other five methods, we tuned the parameters to get visually best denoising results. All the parameters in Figs. 12 and 15 were tuned carefully to get visually best results, except Fig. 12h.

In Fig. 8 we show and compare the denoising results for models containing sharp features including small-scale features. As we can see from Fig. 8, all six algorithms can preserve most sharp edges and corners well. However, the first four algorithms fail in keeping the shallow edge, while our method and L_0 minimization method [1] can preserve it very well; see the zoomed view in Fig. 8. Furthermore, from (f) in Fig. 8, an important thing is that L_0 minimization [1] produces false edges in smooth regions. This phenomenon can also be found in Figs. 10 and 13. The reason may be as follows. The first four methods use weights to detect features. As far as we know, both features and noise belong to high frequency parts. These weights cannot distinguish noise from small-scale features, and thus their methods will treat small-scale features as noise and blur them finally. The method in [1] incorporates the sparse L_0 norm into mesh denoising and our method employs total variation. Both methods have good feature preserving property. However, the high sparsity requirement of L_0 norm in [1] is more prone to produce false edges than total variation.

In Fig. 9 we show denoising results for mesh models with only sharp edges and higher-level noise. According to our tests, the first four algorithms can yield good results for this type of meshes when the noise level is low. Yet, when the noise level is slightly higher, they fail to keep sharp features well. The L_0 minimization method [1] produces better results than the first four algorithms, but worse results than our method. Both can generate even better results if the parameters are tuned carefully, which will be compared in Section 5.2.1.

Fig. 10 shows results of meshes with few sharp features. Results indicate that our method can also deal with this

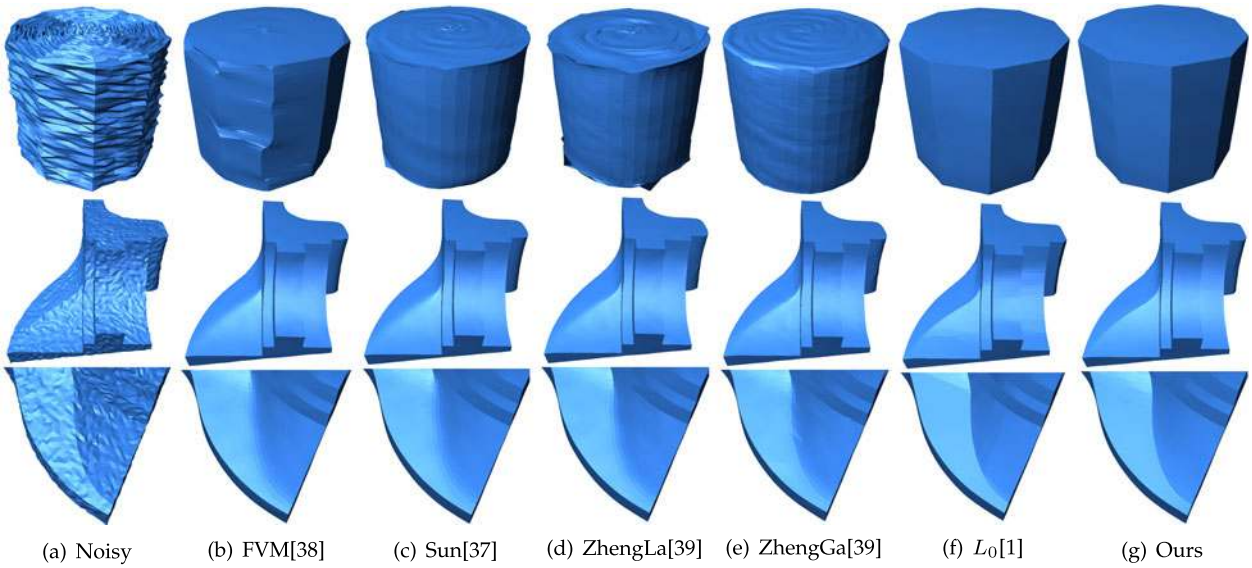
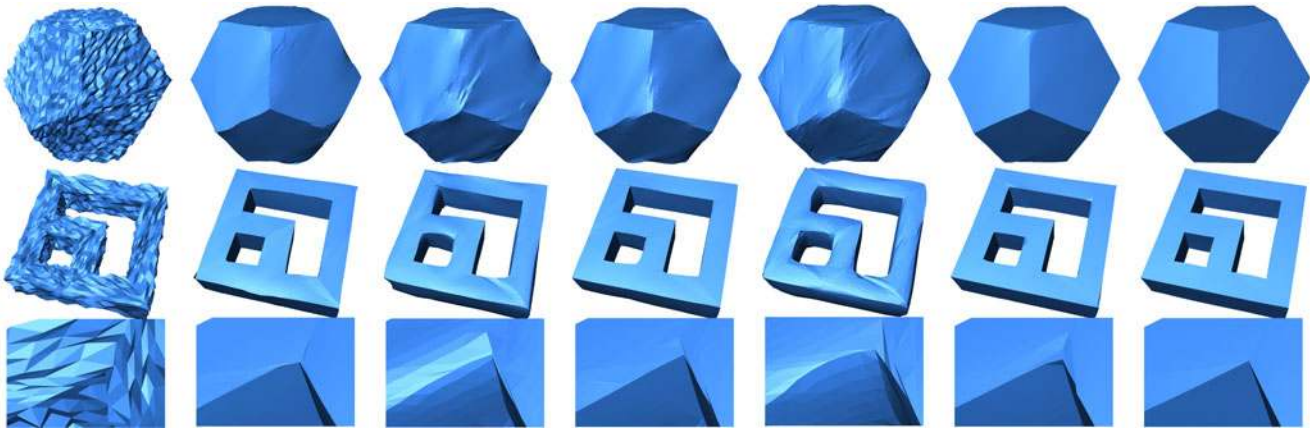


Fig. 8. Denoising results of Prism (Gaussian noise, standard deviation = 0.2 mean edge length) and Fandisk (Gaussian noise, standard deviation = 0.1 mean edge length). The third row shows zoomed view of a shallow edge on Fandisk.



(a) Noisy (b) FVM[38] (c) Sun[37] (d) ZhengLa[39] (e) ZhengGa[39] (f) L_0[1] (g) Ours

Fig. 9. Denoising results of Dodecahedron and Doubletorus (Gaussian noise, standard deviation = 0.3 mean edge length). The third row shows zoomed view of Doubletorus.

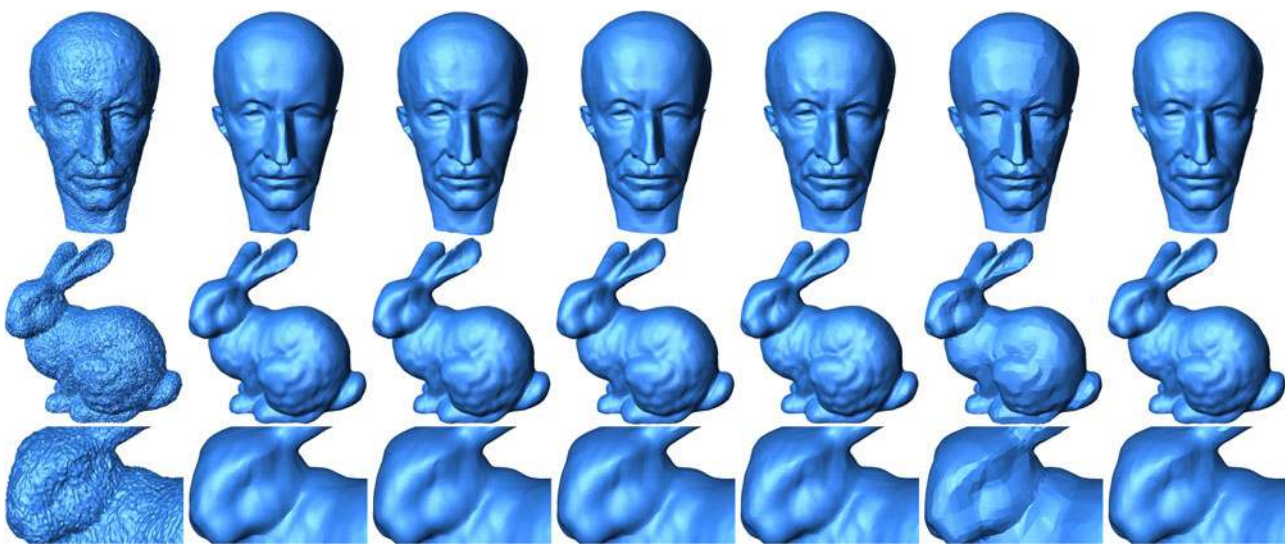
kind of models well. However, the L_0 minimization method in [1] causes staircase effect which can also be found through Fig. 13. One can also find that the first four methods produce over smooth effects by the zoomed view in Figs. 10 and 13.

In Fig. 11, we show comparisons between bilateral filtering method [33], Wang et al. [46] and our method. Both Bilateral [33] and Wang [46] directly smooth the surface vertices (in local or global manners) incorporating vertex normals. The more thing done in Wang [46] is to separate the feature points and specially treat them. Results show that Bilateral [33] cannot preserve sharp feature well. The special treatment of feature points in Wang [46] helps keeping sharp features. However, it generates wavy edges and over smooth small-scale features due to the inaccuracy of detection. In contrast, our method keeps both sharp and small-scale features very well, and does not generate any wavy edges.

Fig. 12 presents results and comparisons for filtering impulsive noise. Therein, Figs. 12b, 12c, 12d, 12e, 12f and 12g are results of algorithms with carefully tuned parameters, while (12h) is the result of our method with estimated parameters by Eqs. (16), (17). As can be seen, our algorithm with optimal parameters still outperforms others. Our result with estimated parameters shows a little staircase effect, because of that the estimation formulae were obtained by fitting results for removing Gaussian noise.

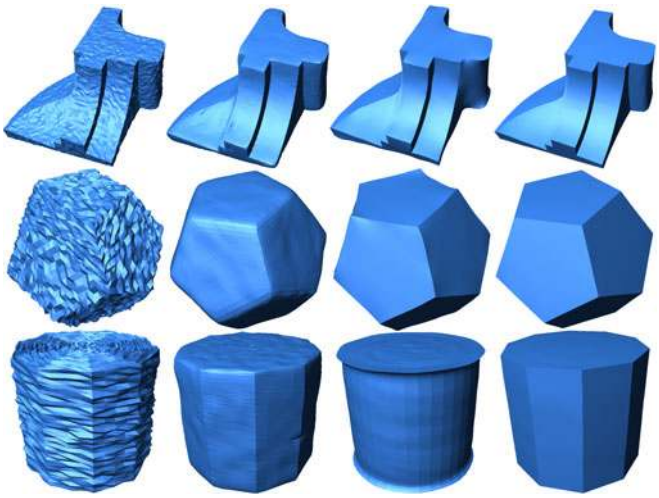
We also present Fig. 13 for a comparison between our method with other five methods applied to a real scanned model. The zoomed view of Fig. 13 shows that our algorithm can preserve more features than other methods; see the hand in the zoomed view. Fig. 14 shows our results for three more real scanned meshes. Our method yields very promising results.

The above comparisons visually show that our method has best denoising results among all the methods. In the following, we quantitatively compare our method to other approaches by numerical errors. For quantitative comparisons, we can only use synthetic data. The following three metrics will be used to measure the distance between the denoising results and the clean mesh.



(a) Noisy (b) FVM[38] (c) Sun[37] (d) ZhengLa[39] (e) ZhengGa[39] (f) L_0[1] (g) Ours

Fig. 10. Denoising results of Head (Gaussian noise, standard deviation = 0.1 mean edge length) and Bunny (Gaussian noise, standard deviation = 0.2 mean edge length). The third row shows zoomed view of Bunny.



(a) Noisy (b) Bilateral[33] (c) Wang[46] (d) Ours

Fig. 11. Comparison between Bilateral [33], Wang [46] and our method. Gaussian noise, from top to bottom, the standard deviation is 0.1,0.3,0.2 mean edge length, respectively.

- Mean square angular error (MSAE):

$$\mathrm{MSAE} = E[\angle(\mathbf{n}', \mathbf{n})], \tag{18}$$

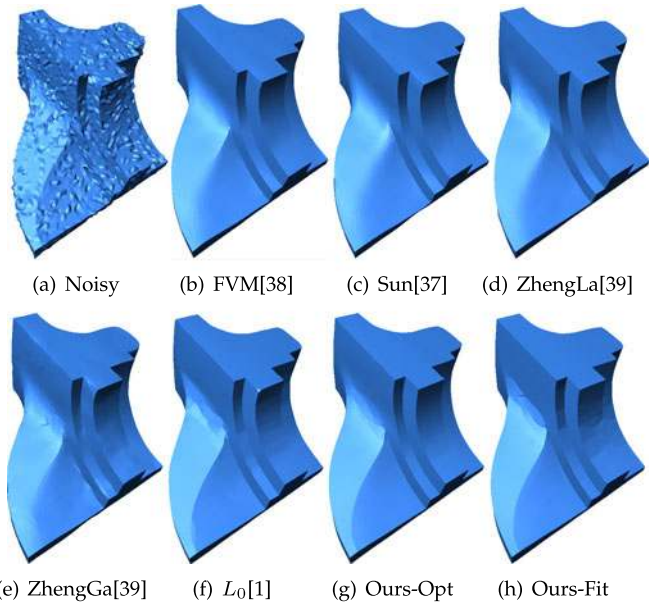
where E is the expectation operator and $\angle(\mathbf{n}', \mathbf{n})$ is the angle between $\mathbf{n}'$ (the normal of the denoised result) and $\mathbf{n}$ (the normal of the clean mesh).

- The L_2 vertex-based mesh-to-mesh error:

$$E_{v,2} = \sqrt{\frac{1}{3\sum_\tau s_\tau}\sum_{i=0}^{\mathrm{V}-1}\left(\sum_{\tau\in D_1(i)} s_\tau\right)\mathrm{dist}(v_i', M)^2}, \tag{19}$$

where $\mathrm{dist}(v_i', M)$ is the distance between the filtered new vertex v_i' and a triangle of the origin mesh M which is closest to v_i'.

- The L_∞ vertex-based mesh-to-mesh error (vertex-based Hausdorff distance):



(a) Noisy (b) FVM[38] (c) Sun[37] (d) ZhengLa[39]

(e) ZhengGa[39] (f) L_0[1] (g) Ours-Opt (h) Ours-Fit

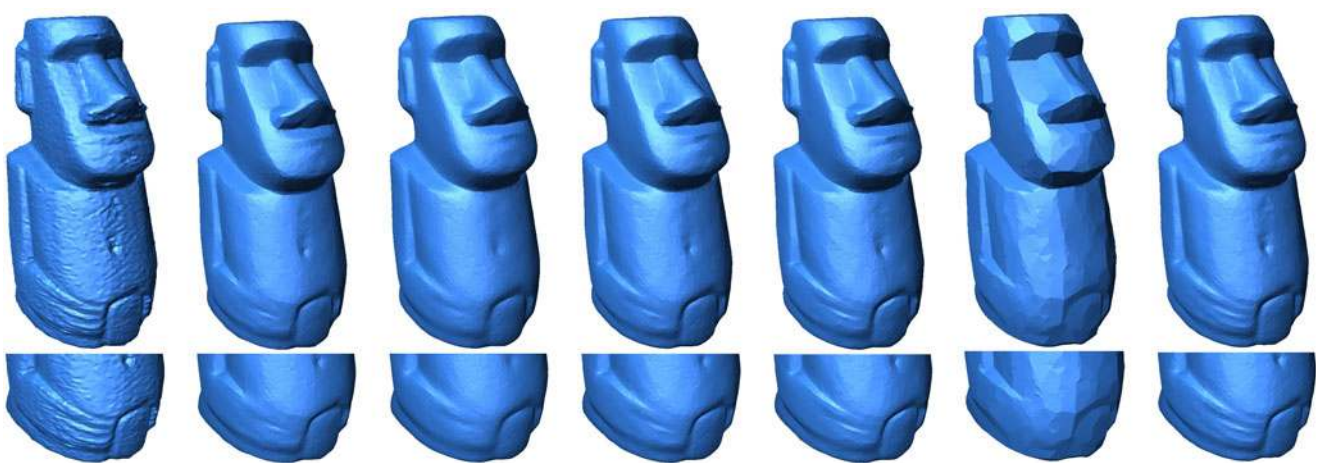
Fig. 12. Denoising results of Fandisk with impulsive noise. 30 percent vertices were corrupted, standard deviation = 0.3 mean edge length. Here "Ours-Fit" and "Ours-Opt" mean our results with fitting parameters (computed by Eqns. (16)(17)) and optimal parameters, respectively.

$$E_{v,\infty} = \max_i \mathrm{dist}(v_i', M) \tag{20}$$

where $\mathrm{dist}(v_i', M)$ is defined as above.

These error metrics were computed for the examples shown in Figs. 8, 9 and 10, which are listed in Tables 2, 3, 4. As can be seen, the denoising results by our method have least errors in most cases, especially in the sense of L_∞ vertex-based mesh-to-mesh error (vertex-based Hausdorff distance).

Qualitative (visual) and quantitative comparisons show that our denoising method outperforms typical existing methods. It is also observed that, the L_0[1] works better for CAD meshes than non-CAD meshes, while their other methods work better for non-CAD meshes than CAD meshes. For both types of meshes, our method outperforms all the methods compared in this paper.



(a) Real scanned (b) FVM[38] (c) Sun[37] (d) ZhengLa[39] (e) ZhengGa[39] (f) L_0[1] (g) Ours

Fig. 13. Denoising results of a real scanned model. The second row shows zoomed view.

Fig. 14. Our results for three real scanned meshes.

TABLE 2
Normal Error Comparison ($\times 10^{-3}$)

Mesh	FVM [38]	Sun [37]	Zheng La [39]	Zheng Ga [39]	L_0[1]	Ours
Fig. 8 Row 1	21	27	26	26	**6.45**	6.8
Fig. 8 Row 2	5.9	3.9	2.14	2.1	2.78	**1.5**
Fig. 9 Row 1	52	55	50	44	**1.36**	5.7
Fig. 9 Row 2	100	77	80	90	14.5	**13.5**
Fig. 10 Row 1	16.6	**9.5**	13.9	**9.5**	19	11
Fig. 10 Row 2	25	21	22	18	25	**17.8**

5.2.1 Our Method vs L_0 Minimization Method in [1]

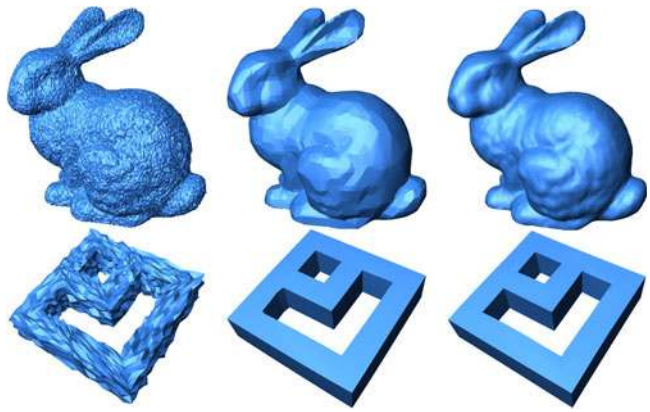
It is necessary to explain further more on the differences between our method and L_0 minimization method in [1]. Both our method and L_0[1] are variational methods. Our method minimizes TV which is, in some sense, an L_1 minimization, while the method in [1] uses L_0 minimization. Experiments have demonstrated that the two norms produce different results. The L_0 minimization method in [1] works better for CAD models, especially polyhedron surfaces (Fig. 8 Row 1 and Fig. 9). For non-CAD models, it causes staircase effects as shown in Figs. 10 and 13. In contrast, the TV regularization used in this paper gives more satisfying results for all types of meshes. It prevents generating false edges for Gaussian noise elimination by choosing appropriate algorithm parameters, e.g., the parameters evaluated by the given estimation formulae. In other words, the L_0 minimization produces results whose gradients of normal fields are sparser than TV minimization. So far our method and L_0 minimization method in [1] are compared using suggested parameters, i.e., the formulae in [1] and Eqs. (16) and (17), respectively. We now give an example for both methods using optimal parameters respectively. See Fig. 15. It is shown that both methods have excellent denoising results for polyhedron surfaces (the second row). For the bunny surface, the L_0 minimization method cannot produce satisfying results no matter how to adjust the parameters (the first row).

TABLE 3
L_2 Vertex-Based Error Comparison ($\times 10^{-4}$)

Mesh	FVM [38]	Sun [37]	Zheng La [39]	Zheng Ga [39]	L_0[1]	Ours
Fig. 8 Row 1	21.44	42.82	44.7	40.57	**7.82**	9.3
Fig. 8 Row 2	6.489	6.86	5.23	5.25	6.57	**1.55**
Fig. 9 Row 1	17.12	16.83	15.42	21.7	8.38	**7.76**
Fig. 9 Row 2	19.35	19.33	12.3	30.1	9.95	**9.87**
Fig. 10 Row 1	16.9	5.9	7.25	5.23	5.9	**4.94**
Fig. 10 Row 2	12.7	12.1	9.8	8.7	8.54	**8.2**

TABLE 4
L_∞ Vertex-Based Error Comparison ($\times 10^{-3}$)

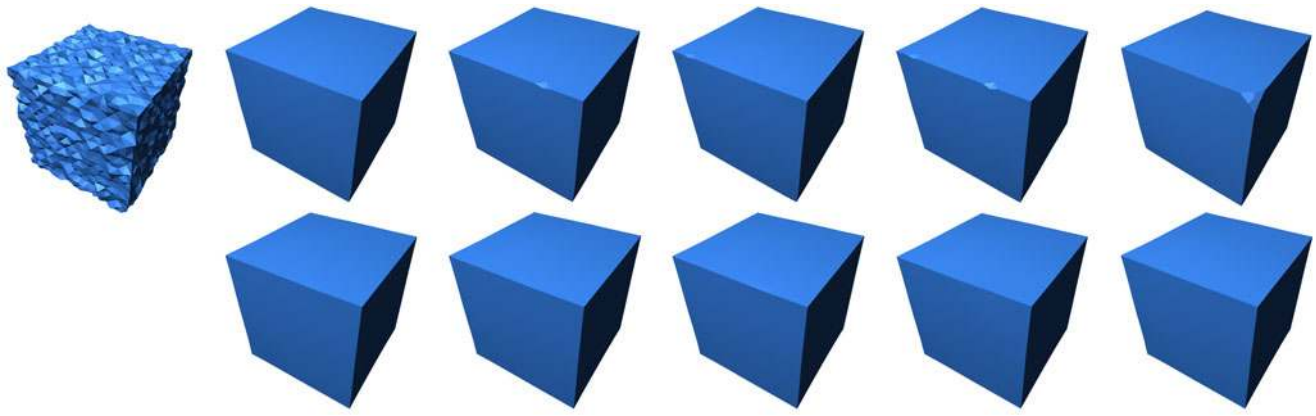
Mesh	FVM [38]	Sun [37]	Zheng La [39]	Zheng Ga [39]	L_0[1]	Ours
Fig. 8 Row 1	10.83	11.43	17.89	11.97	5.84	**2.20**
Fig. 8 Row 2	1.73	1.69	**0.97**	1.01	1.85	1.03
Fig. 9 Row 1	9.9	10.36	10.23	10.95	5	**2.58**
Fig. 9 Row 2	8.22	11.39	7.77	14.7	8.81	**3.01**
Fig. 10 Row 1	17.85	2.14	2.54	2.43	3.14	**1.72**
Fig. 10 Row 2	6.76	6.24	2.58	2.83	2.71	**2.40**



(a) Noisy (b) L_0[1] (c) Ours

Fig. 15. Comparison between L_0 minimization [1] and our method with optimal parameters respectively.

We now explain another difference between our method and L_0 minimization method. Our method uses TV which is a continuous function, while the L_0 term used in [1] is a discontinuous function. This would affect the sensitivity of denoising results on the parameters, i.e., the λ in Eq. (4) in [1] and α in Eq. (8) of our method (it can be theoretically understood as $\frac{1}{\alpha}$ for the TV term). In Fig. 16 we show how the denoising results are affected by small perturbations of parameters. The noisy cube is shown in column 16a. From column 16b to 16f, the first row shows the denoising results of L_0 minimization method; while the second row shows the results of our method. The results in column 16b are computed with parameters suggested by the parameter formulae, indicating good denoising effect of both methods. In columns Figs. 16c, 16d, 16e and 16f, we show the denoising results obtained with slightly perturbed parameters. This example demonstrates that our method is much more robust to parameter perturbations than the L_0[1]. This can also be verified from Theorem 1.



(a) Noisy (b) Default λ, α (c) $0.99\lambda, 0.99\alpha$ (d) $1.01\lambda, 1.01\alpha$ (e) $0.9\lambda, 0.9\alpha$ (f) $1.1\lambda, 1.1\alpha$

Fig. 16. Comparison between L_0 minimization in [1] and our method by small perturbations of default parameters by estimation formulae. The first row shows L_0 minimization [1] results. The second row shows our results.

Finally, our method is much faster than the L_0 minimization method [1]. Our objective function is much simpler and has very efficient solvers. See Table 5 for a comparison on CPU costs.

5.3 Computational Cost

We now discuss the computational costs. Overall, our algorithm is quite simple. The main step requiring CPU and memory costs is solving the $\mathbf{N}-$sub problem. Fortunately, the coefficient matrix keeps unchanged in our iterative algorithm. It can be pre-factored and thus the whole algorithm is quite efficient. However, for large meshes, pre-factoring a matrix consumes too much memory and thus iteration methods such as preconditioned bi-conjugate gradient (PBCG) are used for solving the $\mathbf{N}-$sub problem. In our implementation and test platform, we can use the pre-factoring method for meshes with vertex number up to 500 K. All the examples in this paper were computed by the former method except the lion surface in Fig. 14, which has about 650 K vertices. In our tests, we found that the parameter r affects the algorithm efficiency. When r is small, our algorithm is slow. The level of noise has also impact on the speed of our algorithm. The more noise are, the slower the algorithm is.

The CPU costs of these algorithms (except Bilateral [33] and Wang [46] in Fig. 11) are summarized in Table 5. As can be seen, Sun et al. method [37] is the fastest algorithm, while the L_0 minimization method is the slowest. The other 4 algorithms have similar CPU costs. Our method can obtain better denoising results than all the other methods at reasonable CPU costs.

TABLE 5
Time Comparison (in Seconds)

Mesh	FVM [38]	Sun [37]	Zheng La [39]	Zheng Ga [39]	L_0[1]	Ours
Fig. 8 Row 1	7	**0.17**	0.7	0.3	19	0.3
Fig. 8 Row 2	0.52	**0.15**	0.3	0.7	27	0.34
Fig. 9 Row 1	3	**0.13**	0.5	0.5	21	0.5
Fig. 9 Row 2	4	**0.1**	0.5	0.23	11	0.2
Fig. 10 Row 1	2.5	**1**	1.5	5	150	6
Fig. 10 Row 2	3	**1.2**	2.3	5.6	180	7

5.4 Limitations

Our algorithm has been demonstrated outperforming typical existing methods, but it still has some limitations. Like many algorithms, we cannot prove the convergence at present. Besides, our algorithm cannot deal with noisy meshes with complex features and extreme triangulation; see Fig. 17 for a such example. Actually for this case, no methods have been found producing good results. Our result seems better than other two results.

6 Conclusion and Future Work

In this paper, we proposed a variational method for feature-preserving mesh denoising using piecewise constant function space and total variation. For the proposed variational problem, which is the key part of the method, it was proven that the solution is in some sense continuously dependent on its parameter, indicating that the solution is robust to small perturbations of this parameter. Augmented Lagrangian method was then applied to solve this non-differentiable optimization problem, yielding an efficient iterative algorithm with closed-form solutions to subproblems. We finally discussed and compared our method to several typical previous methods from various aspects, including parameter setting, denoising effects, and computational efficiency.

Experimental results show that our method outperforms all the compared methods for both CAD and non-CAD meshes at reasonable costs. It can preserve different levels of features well and prevent generating false edges in most cases, even with the parameters evaluated by the parameter estimation formulae. The staircase effect of TV almost dis-

(a) Noisy (b) ZhengLa[39] (c) L_0[1] (d) Ours

Fig. 17. Denoising results for Crank with complex features (Gaussian noise, standard deviation = 0.2 mean edge length).

appears due to appropriate parameter settings (e.g., our parameter estimation formulae) and possibly that it is applied to penalize the normal field of the surface (instead of the image intensity in image denoising). The previous methods either fail to preserve sharp features especially small-scale features, or generate false edges for non-CAD models, no matter how we adjusted their parameters.

There are some problems in future work. First, although the parameter estimation formulae provided here work quite well for meshes with Gaussian noise, approximation formulae for other kinds of noise are unknown. Second, algorithms for filtering high level noise remain to be designed. Finally, mesh segmentation and simplification using our piecewise constant space framework will be reported subsequently with more details.

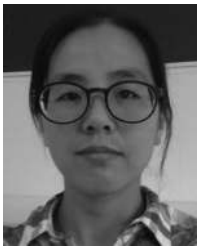
ACKNOWLEDGMENTS

This work was supported by 973 Program 2011CB302400, the NSF of China (Nos. 11301289, 61303148, and 11371341). We also thank the authors of Wang [46] for providing us their denoising results. Chunlin Wu is the corresponding author.

REFERENCES

[1] L. He and S. Schaefer, "Mesh denoising via l_0 minimization," *ACM Trans. Graph.*, vol. 32, no. 4, p. 64, 2013.
[2] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.
[3] G. Sapiro and D. L. Ringach, "Anisotropic diffusion of multivalued images with applications to color filtering," *IEEE Trans. Image Process.*, vol. 5, no. 11, pp. 1582–1586, Nov. 1996.
[4] P. Blomgren and T. Chan, "Color TV: Total variation methods for restoration of vector-valued images," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 304–309, Mar. 1998.
[5] A. Chambolle and P. Lions, "Image recovery via total variation minimization and related problems," *Numerische Mathematik*, vol. 76, no. 2, pp. 167–188, 1997.
[6] O. Scherzer, "Denoising with higher order derivatives of bounded variation and an application to parameter estimation," *Computing*, vol. 60, no. 1, pp. 1–27, 1998.
[7] T. Chan, A. Marquina, and P. Mulet, "High-order total variation-based image restoration," *SIAM J. Sci Comput.*, vol. 22, no. 2, pp. 503–516, 2000.
[8] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM J. Imaging Sci.*, vol. 3, no. 3, pp. 492–526, 2010.
[9] M. Elsey and S. Esedoglu, "Analogue of the total variation denoising model in the context of geometry processing," *Multiscale Model. Simul.*, vol. 7, no. 4, pp. 1549–1573, 2009.
[10] C. Wu, J. Zhang, Y. Duan, and X. C. Tai, "Augmented Lagrangian method for total variation based image restoration and segmentation over triangulated surfaces," *J. Sci. Comput.*, vol. 50, no. 1, pp. 145–166, 2012.
[11] T. Pock, D. Cremers, H. Bischof, and A. Chambolle, "An algorithm for minimizing the Mumford-Shah functional," in *Proc. IEEE 12th Conf. Comput. Vis.*, 2009, pp. 1133–1140.
[12] E. S. Brown, T. F. Chan, and X. Bresson, "Convex formulation and exact global solutions for multi-phase piecewise constant MumfordShah image segmentation," UCLA, Applied Mathematics, Los Angeles, CA, CAM-report-09-66, Jul. 2009, http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA518796
[13] T. Chan, G. Golub, and P. Mulet, "A nonlinear primal-dual method for total variation-based image restoration," *SIAM J. Sci. Comput.*, vol. 20, no. 6, pp. 1964–1977, 1999.
[14] A. Chambolle, "An algorithm for total variation minimization and applications," *J. Math. Imaging Vis.*, vol. 20, nos. 1/2, pp. 89–97, 2004.
[15] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imaging Sci.*, vol. 1, no. 3, pp. 248–272, 2008.
[16] T. Goldstein and S. Osher, "The split Bregman method for l_1-regularized problems," *SIAM J. Imaging Sci.*, vol. 2, no. 2, pp. 323–343, 2009.
[17] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[18] C. Wu and X. Tai, "Augmented Lagrangian method, dual methods, and split Bregman iteration for rof, vectorial TV, and high order models." *SIAM J. Imaging Sci.*, vol. 3, no. 3, pp. 300–339, 2010.
[19] A. Chambolle, V. Caselles, M. Novaga, T. Pock, and D. Cremers, "An introduction to total variation for image analysis," in *Theoretical Foundations and Numerical Methods for Sparse Recovery*, vol. 9. Berlin, Germany: Walter de Gruyter, 2010.
[20] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*. New York, NY, USA: Elsevier, 1978.
[21] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Techn.*, 1999, pp. 317–324.
[22] U. Clarenz, U. Diewald, and M. Rumpf, "Anisotropic geometric diffusion in surface processing," in *Proc. Vis.*, 2000, pp. 397–405.
[23] C. Bajaj and G. Xu, "Anisotropic diffusion on surfaces and functions on surfaces," *ACM Trans. Graph.*, vol. 22, no. 1, pp. 4–32, 2003.
[24] A. N. Hirani, "Discrete exterior calculus," Ph.D. dissertation, Dept. Control Dynamical Syst., California Inst. Technol., Pasadena, CA, USA, 2003.
[25] C. Wu, J. Deng, and F. Chen, "Diffusion equations over arbitrary triangulated surfaces for filtering and texture applications," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 3, pp. 666–679, May/Jun. 2008.
[26] C. Wu, J. Deng, F. Chen, and X. Tai, "Scale-space analysis of discrete filtering over arbitrary triangulated surfaces," *SIAM J. Imaging Sci.*, vol. 2, no. 2, pp. 670–709, 2009.
[27] J. Zhang, J. Zheng, C. Wu, and J. Cai, "Variational mesh decomposition," *ACM Trans. Graph.*, vol. 31, no. 3, p. 21, 2012.
[28] D. A. Field, "Laplacian smoothing and Delaunay triangulations," *Commun. Appl. Numer. Methods*, vol. 4, no. 6, pp. 709–712, 1988.
[29] G. Taubin, "A signal processing approach to fair surface design," in *Proc. Annu. Conf. Comput. Graph. Interactive Techn.*, 1995, pp. 351–358.
[30] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher, "Geometric surface smoothing via anisotropic diffusion of normals," in *Proc. Vis.*, 2002, pp. 125–132.
[31] K. Hildebrandt and K. Polthier, "Anisotropic filtering of non-linear surface features," *Comput. Graph. Forum*, vol. 23, no. 3, pp. 391–400, 2004.
[32] S. Yoshizawa, A. Belyaev, and H. Seidel, "Smoothing by example: Mesh denoising by averaging with similarity-based weights," in *Proc. IEEE Conf. Shape Model. Appl.*, 2006, p. 9.
[33] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 950–953, 2003.
[34] T. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature preserving mesh smoothing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 943–949, 2003.
[35] Y. Ohtake, A. Belyaev, and I. Bogaevski, "Mesh regularization and adaptive smoothing," *Comput.-Aided Design*, vol. 33, no. 11, pp. 789–800, 2001.
[36] M. Lysaker, S. Osher, and X. Tai, "Noise removal using smoothed normals and surface fitting," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1345–1357, Oct. 2004.
[37] X. Sun, P. Rosin, R. Martin, and F. Langbein, "Fast and effective feature-preserving mesh denoising," *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 5, pp. 925–938, Sept./Oct. 2007.
[38] Y. Shen and K. Barner, "Fuzzy vector median-based surface smoothing," *IEEE Trans. Vis. Comput. Graphics*, vol. 10, no. 3, pp. 252–265, May/Jun. 2004.
[39] Y. Zheng, H. Fu, O.-C. Au, and C. Tai, "Bilateral normal filtering for mesh denoising," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 10, pp. 1521–1530, Oct. 2011.
[40] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh smoothing via mean and median filtering applied to face normals," in *Proc. Geometric Model. Process.*, 2002, pp. 124–131.

[41] K. Lee and W. Wang, "Feature-preserving mesh denoising via bilateral normal filtering," in *Proc. Comput. Aided Design Comput. Graph.*, 2005, pp. 275–280.
[42] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding," in *Proc. Comput. Graph. Int.*, 2003, pp. 28–34.
[43] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi, "Efficiently combing positions and normals for precise 3d geometry," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 536–543, 2005.
[44] J. Diebel, S. Thrun, and M. Brunig, "A Bayesian method for probable surface reconstruction and decimation," *ACM Trans. Graph.*, vol. 25, no. 1, pp. 39–59, 2006.
[45] H. Fan, Y. Yu, and Q. Peng, "Robust feature-preserving mesh denoisng based on consistent subneighborhoods," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 2, pp. 312–324, Mar./Apr. 2010.
[46] R. Wang, Z. Yang, L. Liu, J. Deng, and F. Chen, "Decoupling noises and features via weighted-analysis compressed sensing," *ACM Trans. Graph.*, vol. 33, no. 2, p. 18, 2014.
[47] H. Avron, A. Sharf, and D. Cohen-Or, "ℓ_1-sparse reconstruction of sharp point set surfaces," *ACM Trans. Graph.*, vol. 29, no. 5, p. 135, 2010.
[48] R. Glowinski and P. L. Tallec, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, vol. 9. Philadelphia, PA, USA: SIAM, 1989.



Huayan Zhang is a graduate student in mathematical sciences at the University of Science and Technology of China. Her research interests are in computer graphics and architectural geometry.

Chunlin Wu received the PhD degree from the University of Science and Technology of China, Hefei, Peoples Republic of China, in 2006. Currently, he is an associate professor in the School of Mathematics, Nankai University, China. His research interests are in computer graphics and image processing.

Juyong Zhang received the BS degree from the University of Science and Technology of China in 2006, and the PhD degree from Nanyang Technological University, Singapore. He is currently an associate professor in the School of Mathematical Sciences of University of Science and Technology of China. His research interests include computer graphics, geometry processing, and image processing.

Jiansong Deng received the PhD degree from the University of Science and Technology of China, Hefei, Peoples Republic of China, in 1998. Currently, he is a professor in the School of Mathematical Sciences, University of Science and Technology of China. His research interests include computer-aided geometric design and computer graphics.