Computers & Graphics 38 (2014) 150–157

Contents lists available at ScienceDirect

Computers & Graphics

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

CAD/Graphics 2013

Feature-preserving filtering with L_0 gradient minimization

Xuan Cheng [a], Ming Zeng [b], Xinguo Liu [a,*]

[a] State Key Lab of CAD&CG, Zhejiang University, China
[b] Software School of Xiamen University, China

ARTICLE INFO

Article history:
Received 5 August 2013
Received in revised form
4 October 2013
Accepted 19 October 2013
Available online 31 October 2013

Keywords:
Feature-preserving
L_0 norm
Gradient
Fused coordinate descent

ABSTRACT

Feature-preserving filtering is a fundamental tool in computer vision and graphics, which can smooth input signal while preserving its sharp features. Recently, a piecewise smooth model called L_0 gradient minimization, has been proposed for feature-preserving filtering. Through optimizing an energy function involving gradient sparsity prior, L_0 gradient minimization model has strong ability to keep sharp features. Meanwhile, due to the non-convex property of L_0 term, it is a challenge to solve the L_0 gradient minimization problem. The main contribution of this paper is a novel and efficient approximation algorithm for it. The energy function is optimized in a fused coordinate descent framework, where only one variable is optimized at a time, and the neighboring variables are fused together once their values are equal. We apply the L_0 gradient minimization in two applications: (i) edge-preserving image smoothing (ii) feature-preserving surface smoothing, and demonstrate its good performance.

1. Introduction

Many problems in computer vision and graphics require estimating some spatially varying quantity from noisy raw data. A main property of such quantity is piecewise smoothing: they should vary smoothly almost everywhere except at the sharp features that should be preserved. For this purpose, feature-preserving filtering was proposed, and became a fundamental tool in many applications.

Meanwhile, feature-preserving filtering is inherently challenging, because it is difficult to distinguish features from noise. There exist some different feature-preserving filtering methods, which follow the same energy minimization framework with a data term and a smooth term. The data term measures the disagreement between the filtered signal and the original signal, while the smooth term measures the extent to which the filtered signal is not piecewise smooth. The design of the data term is usually straight forward. For instance, the squared L_2 distance between the filtered signal and the original signal is often used. The choice of the smooth term is a critical issue.

A key observation is that, if a signal is piecewise smooth, most of the gradients tend to be small or even zero, and large gradients only appear at the sharp features. This gradient sparsity prior brings us a great help to design the smooth term. A representative work is total variation [1], where the smooth term is the L_1 norm of gradient.

Recently, Xu et al. [2] use L_0 term instead of L_1 term to directly measure the gradient sparsity in the context of image smoothing, and achieve some promising results. Compared with L_1 norm, L_0 norm can obtain more sparse solution. Some mathematical analysis [3] have shown that, under certain conditions, there is a formal equivalence between L_1 norm and L_0 norm. But this equivalence does not hold in our case. Gradient with L_1 norm or L_0 norm will lead to completely different solution, as validated by our experimental results.

Although L_0 gradient minimization is very suitable for feature-preserving filtering, it is indeed a non-convex problem. Xu et al. [2] give an approximation of this problem with a variational method. Through introducing a set of auxiliary variables related to gradients, the original non-convex problem is decomposed to a sequence of computationally tractable L_0–L_2 problems. The sparsity of auxiliary variables obtained by L_0 minimization is expected to be transferred to the gradients through L_2 minimization. However, since L_2 norm tends to severely penalize outliers and propagate the residual uniformly, the sparsity may be corrupted in transferring. Thus the gradients obtained by their method are not sparse enough.

In this paper, we propose a new approximation algorithm for L_0 gradient minimization problem. Our algorithm is based on a fused coordinate descent framework. It can obtain a solution with good gradient sparsity and sufficiently close to the original input. We apply the L_0 gradient minimization in edge-preserving image smoothing and feature-preserving surface smoothing. We compare our method with some existing feature-preserving filtering methods, which show that our method produces better results.

The rest of paper is organized as follows. Section 2 reviews some related work, Section 3 introduces our algorithm for L_0 gradient minimization, and Section 4 presents some applications in image smoothing and surface smoothing. Finally, we conclude our work in Section 5.

* Corresponding author. Tel.: +86 571 8820 6681x526; fax: +86 571 8820 6680
E-mail address: xgliu@cad.zju.edu.cn (X.Liu).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.10.025

2. Related work

Edge-preserving image smoothing: A good edge-preserving image smoothing method should not blur the edges that are vital for neural interpretation to make the best sense of the scene, while smooth the regions between such edges. Bilateral filtering [4] is a traditional method dealing with this problem, which extends the concept of Gaussian smoothing by re-weighting the filter coefficients with their corresponding relative pixel intensities. And several variants of the bilateral filtering have been proposed [5–7]. Anisotropic diffusion [8] achieves edge-preserving smoothing by involving an edge-stopping function to make smoothing take place only in the interior of regions without crossing edges. Farbman et al. [9] formulate this problem in a weighted least square framework, which is more flexible compared with local filtering such as bilateral filtering. Rudin et al. [1] propose total variation, which utilizes the gradient sparsity enforced by an L_1 penalty term to do edge-preserving smoothing. Xu et al. [2] use an L_0 penalty term to directly measure gradient sparsity, and their method has a stronger ability to preserve edges. There also exist some methods depending on local features [10,11]. Our method is a global estimation process.

Feature-preserving surface smoothing: To preserve the sharp features of surface, anisotropic methods are often used. Bajaj and Xu [12] extend the anisotropic diffusion [8] in image processing to 3D surface. Hildebrandt and Polthier [13] propose a prescribed mean curvature flow to preserve surface features. Bilateral filtering [4] is also an anisotropic smoothing method, and it has been extended to surface smoothing by Fleishman et al. [14] and Jones et al. [15]. Their methods average the vertex position in a neighborhood using a weight function of both spatial difference and vertex difference. There is a body of work that decouple normal and vertex, i.e. firstly filter surface normals and then update vertex positions from the filtered normal field. Compared with vertex updating, normal filtering has a greater impact on final result. Yagou et al. propose to use mean and median filters [16] for facet normals, and later use alpha-trimming filters [17]. Sun et al. [18] filter the facet normals within local neighborhood, weighted by the normal difference. Zheng et al. [19] improve on this method by applying bilateral facet normal filtering, considering both normal and spatial difference. Most recently, He and Schaefer [20] introduce an area-based edge operator, and adopt it in L_0 minimization using Xu et al.'s solver [2]. Our mesh smoothing method belong to normal-vertex decoupling scheme, and L_0 gradient minimization is applied in the facet normal filtering step.

3. L_0 gradient minimization

Let g be the input signal and f be its filtered result. The gradients of f are denoted by ∇f. The energy function of L_0 gradient minimization is defined as follows:

$$\min_f |f-g|^2+\lambda|\nabla f|_0 \tag{1}$$

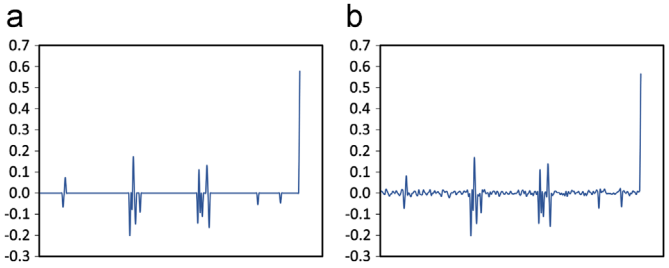

Fig. 1. The final value of variables δ and ∇f obtained by L_0–L_2 iteration algorithm.

where the data term is the squared L_2 distance between f and g, the smooth term is the L_0 norm of ∇f, and λ is a non-negative parameter controlling the weight of the smooth term. The L_0 norm of a vector is the number of non-zero value, which directly measures the sparsity. However, this term is difficult to optimize due to its non-convex and non-derivative nature. Traditional optimization techniques such as gradient descent are no longer applicable.

In Section 3.1, we briefly review L_0–L_2 iteration algorithm proposed by Xu et al. [2] for Eq. (1). In Section 3.2, we introduce our algorithm. Finally, we show the experiments and comparisons with some existing feature-preserving methods in Section 3.3.

3.1. L_0–L_2 iteration algorithm

Through introducing a set of auxiliary variables δ, the original minimization problem (Eq. (1)) then becomes

$$\min_{f,\delta} |f-g|^2+\beta|\nabla f-\delta|^2+\lambda|\delta|_0 \tag{2}$$

where β is a parameter to control the similarity between the auxiliary variables δ and their corresponding gradients ∇f.

Eq. (2) can be solved with an alternating optimization. First, δ is optimized with f fixed

$$\min_{\delta} |\nabla f-\delta|^2+\frac{\lambda}{\beta}|\delta|_0 \tag{3}$$

This equation can be spatially decomposed to a set of single variable function minimization. And each element δ_i has the following closed form:

$$\delta_i=\begin{cases}0 & \text{if } \nabla f_i<\sqrt{\lambda/\beta}\\ \nabla f_i & \text{otherwise}\end{cases} \tag{4}$$

Then, f is optimized with δ fixed

$$\min_{f} |f-g|^2+\beta|\nabla f-\delta|^2 \tag{5}$$

The equation is quadratic and a global minimum can be easily found by gradient descent.

L_0 minimization (Eq. (3)) and L_2 minimization (Eq. (5)) alternate until convergence is reached. After L_0 minimization, δ has a very high degree of sparsity. Next, L_2 minimization attempts to force ∇f to match δ. Indeed, the L_2 term tends to severely penalize outliers and propagate the residual in the energy function uniformly. This property of the L_2 term will lead to failure of sparsity transfer from δ to ∇f in Eq. (5).

To substantiate our claim, we input a one-dimensional noisy signal ranging from 0 to 1, and set λ to 0.001, β to 5. After iterations of L_0–L_2 minimization until convergence, the auxiliary variables δ are shown in Fig. 1(a) and the gradients of the filtered signal ∇f are shown in Fig. 1(b). It is clear that ∇f do not share the good sparsity with δ. In Xu et al.'s experiments [2], β is set as a large fixed value 1e5 to enhance the sparsity, which means most energy of the function is spent on matching ∇f with δ. In this paper, we provide another idea for solving Eq. (1) and we will give details in the next subsection.

3.2. Our algorithm

Our approximation algorithm here is basically built on coordinate descent [21]. Coordinate descent minimizes the energy function by solving a sequence of scalar minimization subproblems cyclically. Each subproblem performs line search along one coordinate direction with the others fixed. Coordinate descent is efficient in the situation where subproblems can be solved quickly.

However, according to the proposition of Bertsekas [22], every minimum of successive coordinate minimization of a continuously

differentiable function is a stationary point for the whole minimization. Since our energy function is highly non-continuous, the coordinate descent can get stuck.

A possible way to avoid getting stuck is to move a group of variables together rather than a single variable. After a pass of coordinate descent for all variables, we check whether each variable f_i has equal value with its neighbors $\{f_j\}$. If so, this variable and its equal neighbors are fused to a single variable, which will be moved together in the next iteration of coordinate descent. In this way, we enforce the constraint $|f_i - f_j|_0 = 0$ implicitly.

We consider the case that signal is one-dimensional, and it can be extended to high dimension naturally. The main steps of the algorithm is summarized in Algorithm 1. It repeats coordinate descent step (Lines 2–5) and fusion step (Lines 6–16), with an increase in parameter λ' (Line 17) until a target value λ is reached (Line 18). The original expression (Eq. (1)) can be further expanded for 1D signal:

$$\min_f \sum_{i=1}^{N} (f_i - g_i)^2 + \lambda \sum_{i=1}^{N-1} |f_{i+1} - f_i|_0 \tag{6}$$

At each iteration, Eq. (6) is solved with additional constraints: neighboring variables that have equal value obtained from previous iterations, should be optimized together. So Eq. (6) has a modified form:

$$\min_f \sum_{i=1}^{M} w_i (f_i - h_i)^2 + \lambda' \sum_{i=1}^{M-1} |f_{i+1} - f_i|_0 \tag{7}$$

Here, f_i becomes a fused variable, which represents w_i number of neighboring variables in the original problem. h_i is the target that f_i should fit to. M is the number of all fused variables at current iteration.

Coordinate descent step: We break the problem Eq. (7) into a serials of subproblems and each has the form

$$\min_{f_i} E(f_i) = w_i (f_i - h_i)^2 + \lambda' |f_i - f_{i-1}|_0 + \lambda' |f_{i+1} - f_i|_0 \tag{8}$$

where f_{i-1} and f_{i+1} are assumed to be known from the previous iteration. This simple minimization problem (Line 4) has the following closed form:

$$f_i = S(w_i, \lambda', h_i, f_{i-1}, f_{i+1}) = \begin{cases} f_{i-1} & \text{if } f_{i-1} < f_{i+1},\ w_i (f_{i-1} - h_i)^2 < \lambda' \\ f_{i+1} & \text{if } f_{i-1} \geq f_{i+1},\ w_i (f_{i+1} - h_i)^2 < \lambda' \\ h_i & \text{otherwise.} \end{cases} \tag{9}$$

Note that Eq. (9) is a thresholding operator, f_i will either be its neighbor or h_i. This means that we are likely to find strictly equal neighboring variables in the fusion step.

Fusion step: After the subproblems are optimized independently, we check whether each neighboring pair (f_i, f_{i+1}) have equal value (Line 9). If so, we fuse them together, i.e. the neighboring variable f_{i+1} is replaced by f_i. In Line 10, target h_i is updated by the average value of f_i's and f_{i+1}'s original target. In Line 11, w_i adds the weight w_{i+1}. Since f_{i+1} being fused, the number of variables is reduced by 1 in Line 12.

Parameter λ can be adjusted to control the level of smoothness, and is given by users. α is usually set as one-thousandth of the magnitude of λ. λ' is automatically increased by α in iterations starting from zero.

Algorithm 1. L_0 gradient minimization for 1D signal.

Input: signal g of length N, parameters λ and α.
Initialize: $\lambda' \leftarrow 0$, $f \leftarrow g$, $h \leftarrow g$, $w \leftarrow [1]_N$, $M \leftarrow N$.
1: **repeat**
2: // coordinate descent step
3: **for** $i = 1 : M$ **do**
4: $f_i = S(w_i, \lambda', h_i, f_{i-1}, f_{i+1})$.
5: **end for**
6: // fusion step
7: $i \leftarrow 1$.
8: **while** $i < M$ **do**
9: **if** $f_i = f_{i+1}$ **then**
10: $h_i \leftarrow (w_i h_i + w_{i+1} h_{i+1})/(w_i + w_{i+1})$.
11: $w_i \leftarrow w_i + w_{i+1}$.
12: $M \leftarrow M - 1$.
13: Delete the $(i+1)$th element of f, h and w, and reduce all indices greater than $i+1$ by 1.
14: **end if**
15: $i \leftarrow i + 1$.
16: **end while**
17: $\lambda' \leftarrow \lambda' + \alpha$.
18: **until** $\lambda' > \lambda$
Output: f is restored from the fused variables to get $\hat{f}$.

3.3. Experiments

We experiment with a 1D signal that have 230 elements ranging from 0 to 1. λ is set to $5e-2$, and α is set to $5e-5$. To show the convergence of our algorithm, we plot the energy curve as the green solid line in Fig. 2. We put the variables f updated at each iteration into the original function (Eq. (6)), and compute its energy. Our algorithm indeed converges after about 700 iterations. What is more, we also show the energy of function minimized with coordinate descent only. At the beginning, both curves decrease at nearly same rate. Then the coordinate descent gets stuck at somewhere, while the fusion step helps our algorithm get around it. Finally, our algorithm converges in a lower place.

We make a comparison with some existing feature-preserving filtering methods, dealing with the same noisy signal. The experimental results are shown in Fig. 3. While bilateral filtering [4], weighted least square [9] and L_2 gradient minimization are effective in smoothing small changes of signal, their ability to keep sharpness is rather limited. The sharpness in the smoothed result first appears in total variation [1], but total variation will keep the smoothed signal far away from the original overall while bring more gradient sparsity. The reason is that, although L_1 norm can enforce small gradients to zero, it also decrease large gradients to some extent. Xu et al.'s method [2] also involve L_0 gradient minimization for feature-preserving filtering. We cannot compare its function energy directly with our algorithm, because Eq. (6) is relaxed by a set of auxiliary variables in their method. Our result that includes five

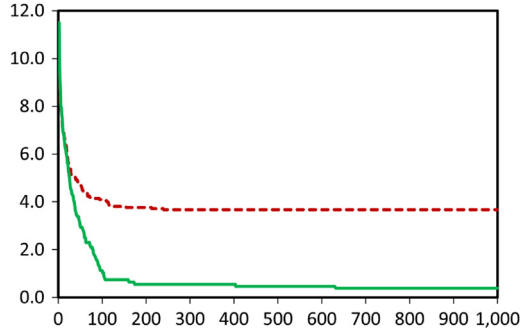

Fig. 2. Energy curves with (solid, green) and without (dashed, red) fusion step. *X*-axis: iteration numbers and *Y*-axis: energy of function Eq. (6). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

segments, has more gradient sparsity than theirs. Moreover, our method needs setting only one parameter λ, while they have to tune more parameters to control the optimization. A further comparison in image smoothing will be shown in the following section.

4. Applications

As a fundamental tool, there are many applications of feature-preserving filtering with L_0 gradient minimization. In this section we present two applications: edge-preserving image smoothing and feature-preserving surface smoothing.

4.1. Edge-preserving image smoothing

Edge-preserving image smoothing is an image editing method that preserves the large scale variations in intensity, while removes the small scale details in the image. We formulate the edge-preserving image smoothing using the L_0 gradient minimization as follows:

$$\min_X \sum_{i=1}^{h}\sum_{j=1}^{w}(X_{i,j}-I_{i,j})^2+\lambda\sum_{i=1}^{h-1}\sum_{j=1}^{w}|X_{i+1,j}-X_{i,j}|_0 +\lambda\sum_{i=1}^{h}\sum_{j=1}^{w-1}|X_{i,j+1}-X_{i,j}|_0 \quad (10)$$

where I is the input image of size $h\times w$, and X is smoothed result. $X_{i,j}=[X_{i,j,r},X_{i,j,g},X_{i,j,b}]$ represents the *rgb* value of pixel (i,j). The L_0 norm counts the number of color changes between the neighboring pixels along vertical and horizontal direction. This formulation can globally control the number of non-zero gradients while constrain image similarity to achieve edge-preserving and color patch flattened image smoothing.

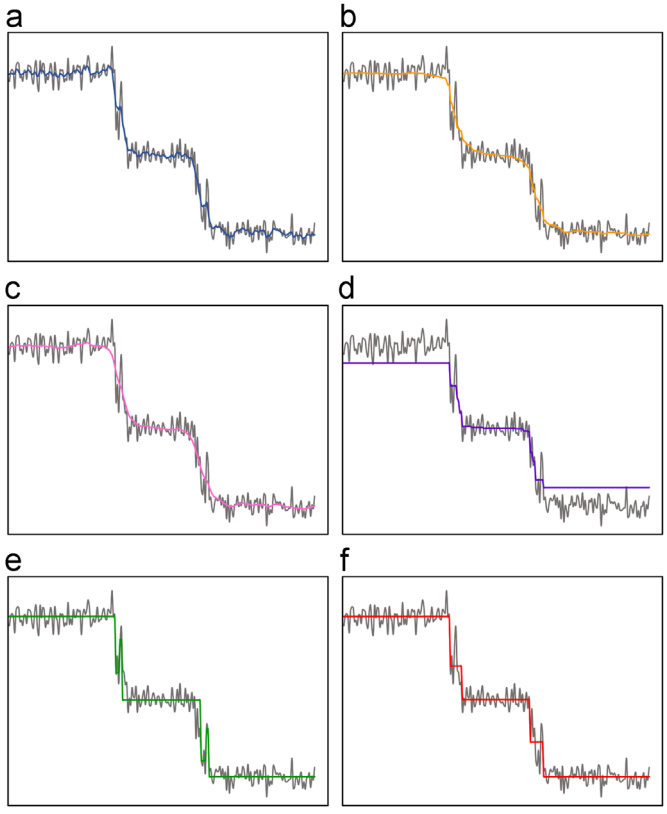

Fig. 3. The gray curve represents the noisy signal for smoothing, and the colorful curve represents the smoothed result by each method. (a) Bilateral filtering. (b) Weighted least square optimization. (c) Use L_2 norm to penalize gradient. (d) Total variation. (e) Xu et al.'s method [2]. (f) Our result. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Algorithm 2. Edge-preserving image smoothing.

Input: image I of size $h\times w$, parameters λ and α.
Initialize: $\lambda'\leftarrow 0$, $M\leftarrow h\times w$. Partition I into a set groups $\{G_p\}_M$, $G_p\leftarrow(i,j)$, $Y_p\leftarrow I_{i,j}$, $N_p\leftarrow 1$.

repeat
  // coordinate descent step
  for $i=1:M$ **do**
    solve Eq. (11) for X_p.
  end for

  // fusion step
  $p\leftarrow 1$.
  while $p<M$ **do**
    for all neighboring group G_q of G_p **do**
      if $X_q=X_p$ in all three channels **then**
        $G_p\leftarrow G_p\cup G_q$.
        $Y_p\leftarrow(N_pY_p+N_qY_q)/(N_p+N_q)$.
        $N_p\leftarrow N_p+N_q$.
        $M\leftarrow M-1$.
        Delete G_q, Y_q and N_q.
      end if
    end for
    $p\leftarrow p+1$.
  end while

  Find neighbors of every group and calculate the corresponding boundary length.
  $\lambda'\leftarrow\lambda'+\alpha$.
until $\lambda'>\lambda$
Output: Put every group's smoothed value $\{X_p\}$ to the output image $\{X_{i,j}\}$ with the pixels' indices.

We illustrate our algorithm for edge-preserving image smoothing in Algorithm 2, which is the extension of Algorithm 1 from one-dimensional signal to two-dimensional image. At each iteration, the fused variables are in fact the partition of image I, and we use a set of groups $\{G_p\}$ to represent them. Pixels in a group G_p have the same smoothed *rgb* value X_p. N_p is the number of pixels in G_p, Y_p is the average *rgb* value of pixels of G_p in the original image. K_p denotes the

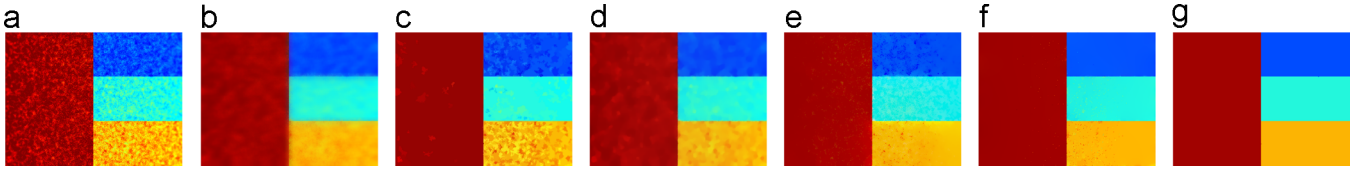

Fig. 4. (a) Noisy color image created by Farbman et al. [9]. (b) Bilateral filtering ($\sigma_s=12,\sigma_r=0.5$). (c) Mean-shift filtering ($h_s=10,h_r=8$). (d) Total variation ($\lambda=3$). (e) Weighted least square optimization ($\lambda=2,\alpha=3$). (f) Xu et al.'s method [2] ($\lambda=0.3,\kappa=2$). (e) Our result ($\lambda=0.45$). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

set of neighboring groups of G_p. For every two neighboring groups G_p and G_q, we use $l_{p,q}$ to denote their boundary length.

At the coordinate descent step, we solve the following problem:

$$\min_{X_p} N_p(X_p - Y_p)^2 + \lambda' \sum_{G_q \in K_p} l_{p,q}|X_p - X_q|_0 \quad (11)$$

In both case of 1D signal and 2D image, the elements inside a fused variable have equal value. L_0 norm of the gradients of these elements are zero, which have no contribution to the energy of function. Nonzero L_0 norm may exist in the boundary between two neighboring fused variables. For 1D signal, all the boundary lengths are 1, but for 2D image the boundary lengths vary. Hence, we set boundary length as a weight for L_0 norm in Eq. (11). Through examining the function value for $\{X_q|G_q \in K_p\}$ and Y_p, Eq. (11) can be solved simply. The next fusion step (Lines 6–19) shares the same spirit with Algorithm 1. After each iteration of coordinate descent and fusion, the neighbors of every group may change. In Line 20, we re-calculate the adjacency list for every group and update the boundary lengths.

We compare our method with some existing edge-preserving image smoothing methods, including bilateral filtering [4], mean-shift filtering [23], total variation [1], weighted least square optimization [9] and the method of Xu at al. [2]. We have hand tuned parameters carefully for these methods to achieve the best performance. Fig. 4(a) shows a famous noisy color image created by Farbman et al. [9]. It is apparent that our method performs the best to remove high frequency noise while preserve major edges. Note that in the similar experiments [2,9], the authors use a noisy gray image as input, the color image (Fig. 4(a)) is only for visualization. Fig. 5 shows another comparison in a natural image. From Fig. 5(g)–(l), the smoothed result of our method enjoys gradient sparsity compared with the others. In Fig. 6, some edges are blurred by the method of Xu et al. [2], while our method preserves most significant edges. It usually takes 3 s to process a 500×500 color image on an Intel Core i3 CPU@2.93G with our C++ implementation. In fact, our edge-preserving image smoothing method does not belong to the class of image denoising, but it is particularly effective for removal of cartoon compression artifact and structure extraction from textured image.

Cartoon image: Current popular image standard such as JEPG will bring the annoying visual artifact, which is especially apparent in cartoon compressed at low bit rates. Our method is advantageous for cartoon compression artifact removal due to its strong ability to enhance edges. Similar to the experiments in [2], we restore many cartoon images with different degrees of compression artifacts, and make a quantitative comparison with a set of methods, including bilateral filtering [4], total variation [1], Xu et al.'s method [2] and Wang et al.'s method [24]. 50 cartoon images are compressed by the standard JPEG with quality values ranging from 10 to 90. After applying different methods to these images, we use the peak signal-noise ratio (PSNR) and the structural similarity (SSIM) [25] to measure the restoration ability of different methods. The statistics are shown in Fig. 8. For both PSNR and SSIM comparisons, our algorithm performs better than the other methods overall. We show two low quality compressed images and the restoration results in Fig. 7.

Textured image: Texture exists in many natural scenes and human-created art pieces. Graffiti and drawings can be commonly seen on brick walls and railroad boxcars. Mosaic, a art of creating images with an assemblage of small pieces of materials such as stone and colored glass, is widespread in the early human history. It is a challenging task to extract significant structure while remove meaningless textures from these images, especially without any texture prior. We show that our edge-preserving image smoothing with L_0 gradient minimization can handle this problem. The method of Xu et al. [2] that also involve L_0 gradient minimization but with another approximation algorithm, fails in this case. Later, Xu et al. proposed a special method [26] dealing with textured image. We demonstrate that both natural image and textured image can be edge-preserving smoothed in the same formulation using L_0 gradient minimization, as shown in Fig. 9.

Fig. 6. (a) A input natural image. (b) Result of Xu et al.'s method [2]. (c) Result of our method. (d)–(f) are the gradient maps of (a)–(c).

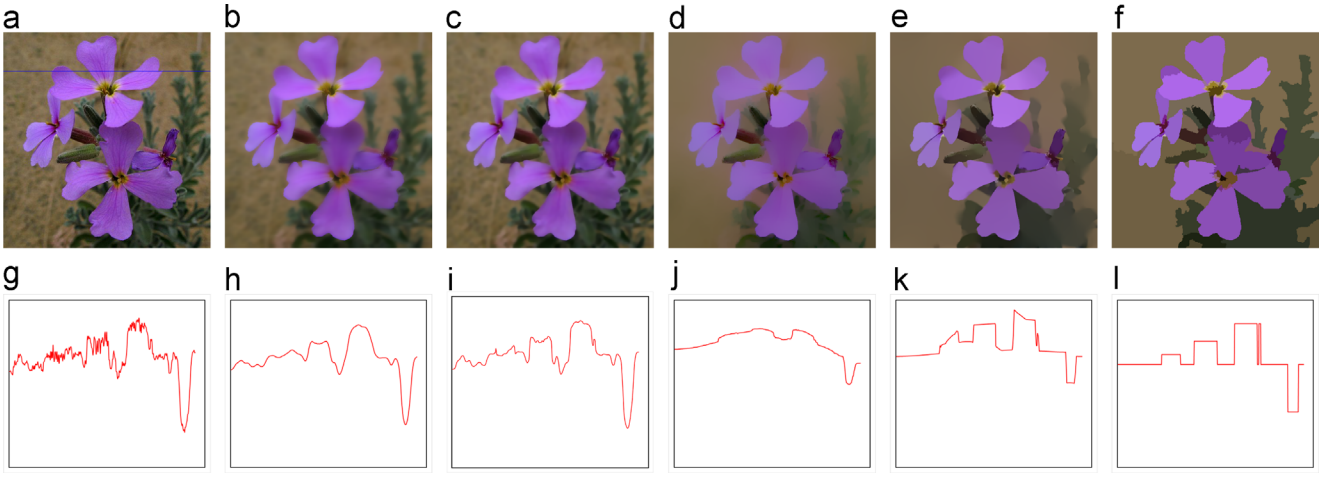

Fig. 5. (a) Input color image. (b) Bilateral filtering ($\sigma_s = 4, \sigma_r = 0.2$). (c) Total variation ($\lambda = 5$). (d) Weighted least square optimization ($\lambda = 0.8, \alpha = 2$). (e) Xu et al.'s method [2] ($\lambda = 0.03, \kappa = 2$). (f) Our result ($\lambda = 0.3$). (g)–(l) The 90th row of corresponding image (a)–(f) in red channel. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

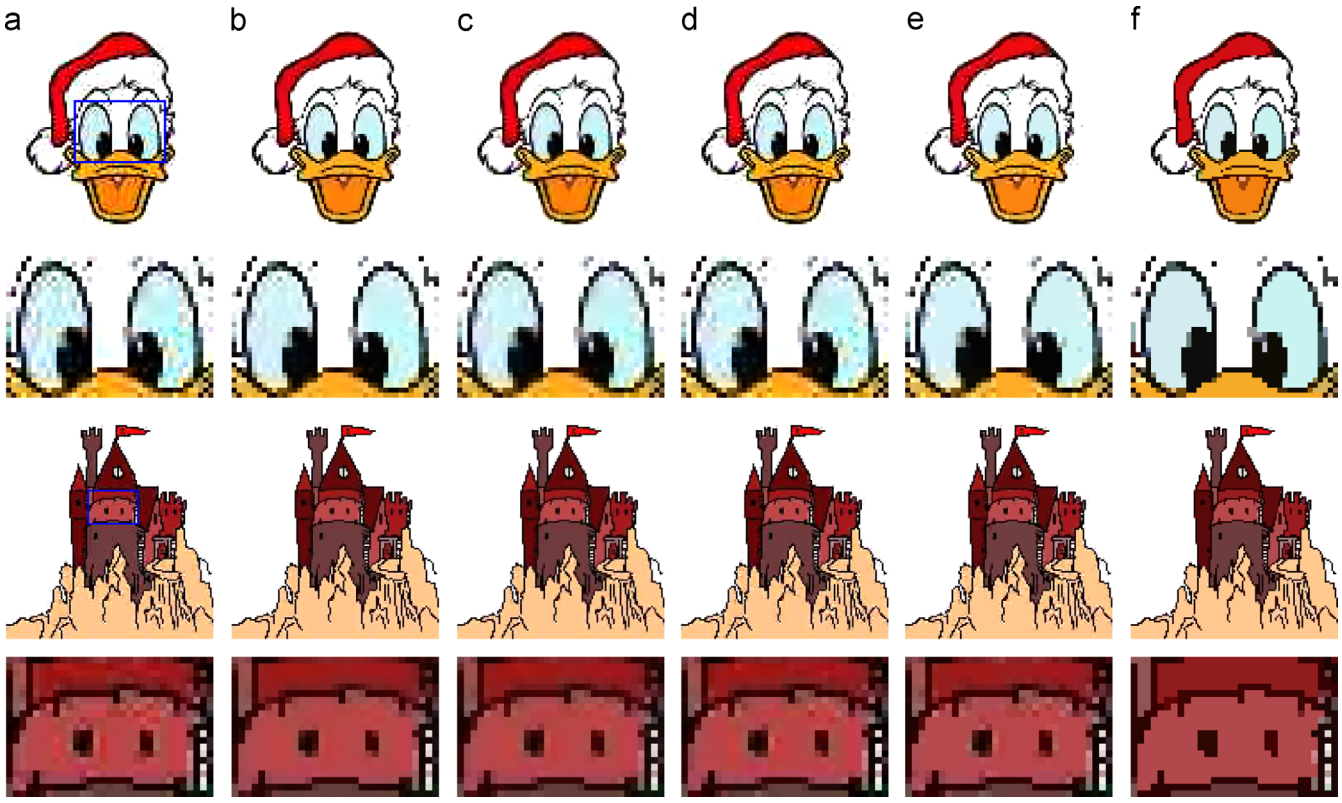

Fig. 7. Cartoon JPEG artifact removal. (a) JPEG compressed images with low quality 20. (b)–(f) Results of bilateral filtering, total variation, Wang et al.'s method [24], Xu et al.'s method [2] and our method. Images in 2nd and 4th row are the close-ups of images in 1st and 3rd row.

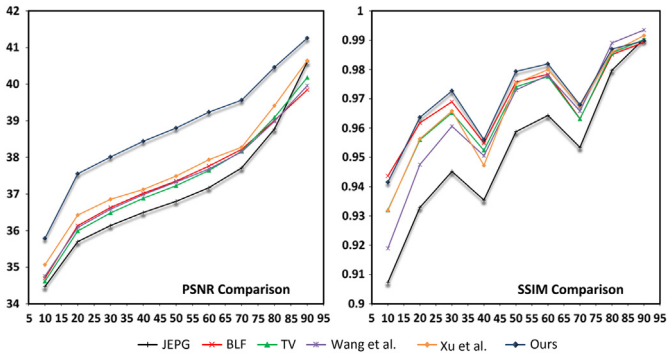

Fig. 8. Quantitative evaluation of JPEG artifact removal. *X*-axis: JPEG qualtiy. Left *Y*-axis: PSNR values. Right *Y*-axis: SSIM values.

4.2. *Feature-preserving surface smoothing*

3D reconstruction of real scene is a hot spot in computer vision and graphics. Current scanning devices are capable of capturing dense point sets to model detailed 3D shapes. However, the acquired raw data are inevitably noisy. To deal with the inherent noise, we apply L_0 gradient minimization to smooth surface.

As many previous work, we decouple the normal and the vertex. We first smooth the facet normals, which encode the first order information of surface. Based on the piecewise smoothed normal field, we then evolve the mesh to match the new normal field.

Piecewise smoothing of the facet normals can be formulated as L_0 gradient minimization:

$$\min_X \sum_{i \in F} (X_i - N_i)^2 + \lambda \sum_{j \in E(i)} |X_i - X_j|_0 \tag{12}$$

where F is the face set, $E(i)$ is the face neighbors set of face f_i, and N_i is the original facet normal of f_i. Piecewise smoothing is achieved by controlling the number of nonzero normal gradients in a global optimization process: nonzero gradients are expected to be concentrated near sharp features, while zero gradients are mainly located at the area with small scale variations in normal

Fig. 9. Our method shows good performance in edge-preserving smoothing of textured images, while Xu et al.'s method [2] fails. (a) Input images. (b) Results of Xu et al.'s method [2]. (c) Results of our method.

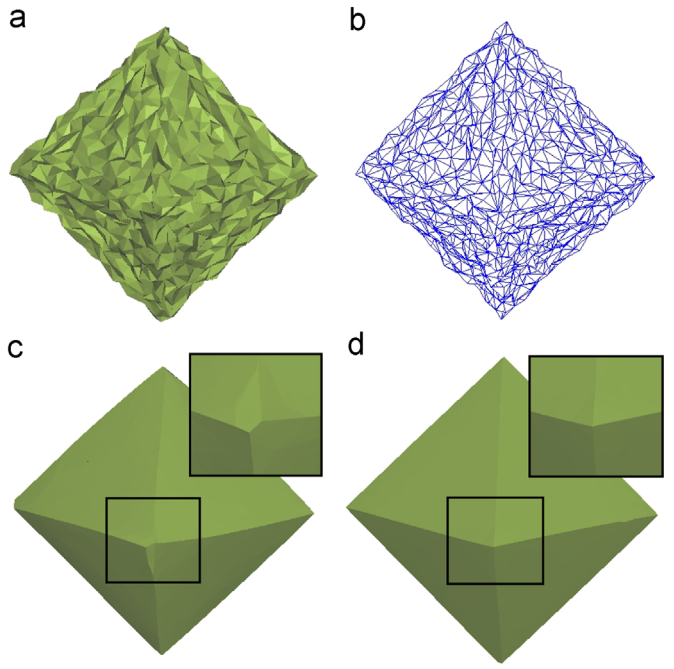

Fig. 10. Comparison between our method and He et al.'s method [20] in a noisy octahedron model with 1538 vertices. (a) Input mesh. (b) Wire frame model of (a). (c) Result of He and Schaefer's method [20]. (d) Result of our method.

field. In fact, the solver for Eq. (12) is similar to Algorithm 2, but the facet normals instead of pixel's *rgb* vector are optimized.

Then, we reconstruct the vertex position to match the piecewise smoothed facet normals using the iterative vertex updating method proposed by Sun et al. [18]:

$$\hat{V}_i = V_i + \frac{1}{|FV(i)|} \sum_{k \in FV(i)} \hat{N}_k (\hat{N}_k \cdot (C_k - V_i)) \tag{13}$$

where V_i is the position of vertex i, $\hat{V}_i$ is the updated value in each iteration, $FV(i)$ is the set of faces sharing a common vertex i, $|FV(i)|$ is the number of faces in set $FV(i)$, $\hat{N}_k$ is the filtered normal of face f_k obtained from Eq. (12) and C_k is center of f_k. We usually perform 10 or 20 iterations of vertex updating in our experiments. To process a model that has 10 k vertices, the whole procedure of our method takes about 3 min with our C++ implementation.

To demonstrate the effectiveness of our method, we make some comparisons with several popular feature-preserving surface smoothing methods, namely, bilateral vertex filtering [14], unilateral normal filtering [18], bilateral normal filtering [19] and area-based edge filtering [20]. For each model, we create an input noisy mesh corrupted by Gaussian noise in random directions. We have tuned parameters carefully for these methods and choose a optimal set of parameters for each model. Experimental results are shown in Figs. 10–13.

Comparison to other normal-vertex decoupling methods: The close works to ours are unilateral normal filtering [18] and

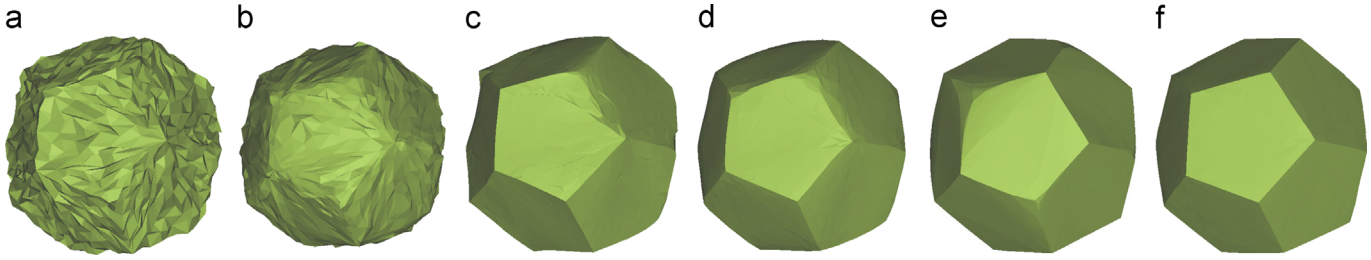

Fig. 11. Comparison in a noisy dodecahedron model with 1730 vertices. (a) Input mesh corrupted by Gaussian noise in random directions. (b) Bilateral vertex filtering [14]. (c) Unilateral normal filtering [18]. (d) Bilateral normal filtering [19]. (e) Area-based edge filtering [20]. (f) Result of our method.

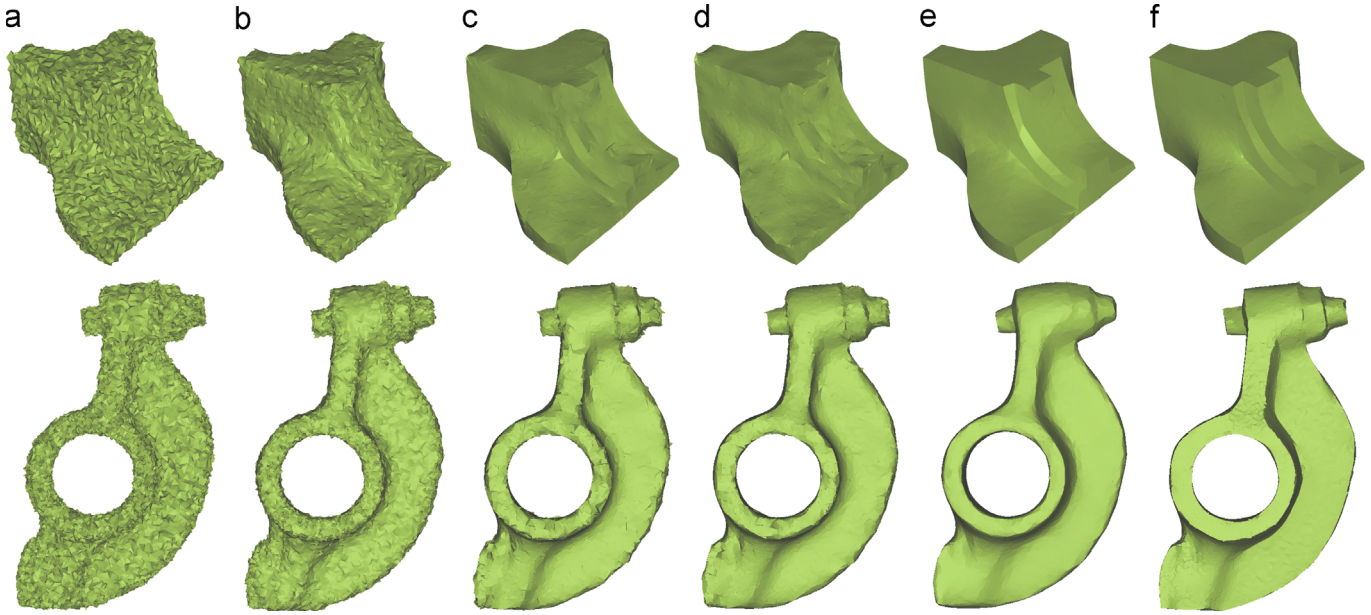

Fig. 12. Comparison in two noisy CAD-like models (above: 6475 vertices, below: 20,776 vertices). (a) Input mesh corrupted by Gaussian noise in random directions. (b) Bilateral vertex filtering [14]. (c) Unilateral normal filtering [18]. (d) Bilateral normal filtering [19]. (e) Area-based edge filtering [20]. (f) Result of our method.

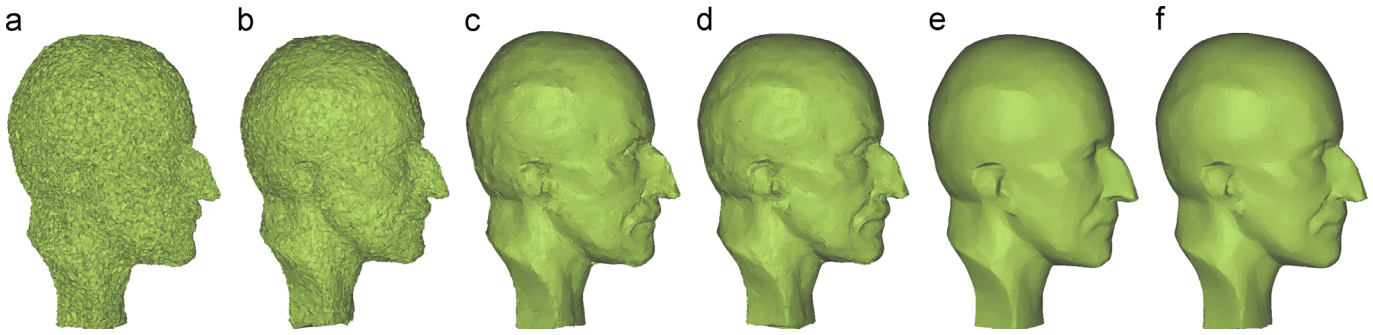

Fig. 13. Comparison in a noisy human face model with 30,942 vertices. (a) Input mesh corrupted by Gaussian noise in random directions. (b) Bilateral vertex filtering [14]. (c) Unilateral normal filtering [18]. (d) Bilateral normal filtering [19]. (e) Area-based edge filtering [20]. (f) Result of our method.

bilateral normal filtering [19]. The main difference between our work and theirs is the normal filtering step, where we perform L_0 gradient minimization. From the experimental results, L_0 gradient minimization shows stronger ability in removing noise.

Comparison to area-based edge filtering: This method is proposed by He and Schaefer [20] which also involve L_0 minimization. The authors develop a new differential operator called area-based edge operator, and adopt it in L_0 minimization. Then Xu et al.'s solver [2] is applied to solve the minimization problem. One issue is that, if there are not enough edges that lie at sharp features (it often happens in noisy model with small number of vertices), their method may fail to obtain good results (e.g. Figs. 10 and 11).

5. Conclusions

L_0 gradient minimization is a effective feature-preserving filtering method, but it is hard to optimize since L_0 norm is non-convex. We propose a new approximation algorithm for L_0 gradient minimization, and adopt it in two feature-preserving filtering tasks, image smoothing and surface smoothing.

We believe that feature-preserving filtering with L_0 gradient minimization may prove useful for many problems in computer vision and graphics. We plan to find more applications for our method as a future work.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially supported by NSFC (No. 61379068).

References

[1] Rudin L, Osher S, Fatemi E. Nonlinear total variation based noise removal algorithms. Physica D 1992;60:259–68.
[2] Xu L, Lu C, Xu Y, Jia J. Image smoothing via L_0 gradient minimization. ACM Trans Graph 2011;30(6).
[3] Candès EJ, Romberg JK, Tao T. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans Inf Theory 2006;52(2):489–509.
[4] Tomasi C, Manduchi R. Bilateral filtering for gray and color images. In: ICCV; 1998. p. 839–46.
[5] Durand F, Dorsey J. Fast bilateral filtering for the display of high-dynamic-range images. ACM Trans Graph 2002;21(3):257–66.
[6] Paris S, Durand F. A fast approximation of the bilateral filter using a signal processing approach. In: ECCV; 2006. p. 568–80.
[7] Chen J, Paris S, Durand F. Real-time edge-aware image processing with the bilateral grid. ACM Trans Graph 2007;26(3).
[8] Perona P, Malik J. Scale-space and edge detection using anisotropic diffusion. IEEE Trans Pattern Anal Mach Intell 1990;12(7):629–39.
[9] Farbman Z, Fattal R, Lischinski D, Szeliski R. Edge-preserving decompositions for multi-scale tone and detail manipulation. ACM Trans Graph 2008;27(3).
[10] Subr K, Soler C, Durand F. Edge-preserving multiscale image decomposition based on local extrema. ACM Trans Graph 2009;28(5).
[11] Kass M, Solomon J. Smoothed local histogram filters. ACM Trans Graph 2010;29(4).
[12] Bajaj CL, Xu G. Anisotropic diffusion of surfaces and functions on surfaces. ACM Trans Graph 2003;22(1):4–32.
[13] Hildebrandt K, Polthier K. Anisotropic filtering of non-linear surface features. Comput Graph Forum 2004;23(3):391–400.
[14] Fleishman S, Drori I, Cohen-Or D. Bilateral mesh denoising. ACM Trans Graph 2003;22(3):950–3.
[15] Jones TR, Durand F, Desbrun M. Non-iterative, feature-preserving mesh smoothing. ACM Trans Graph 2003;22(3):943–9.
[16] Yagou H, Ohtake Y, Belyaev AG. Mesh smoothing via mean and median filtering applied to face normals. In: GMP; 2002. p. 124–31.
[17] Yagou H, Ohtake Y, Belyaev AG. Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding. In: Computer graphics international; 2003. p. 28–33.
[18] Sun X, Rosin PL, Martin RR, Langbein FC. Fast and effective feature-preserving mesh denoising. IEEE Trans Vis Comput Graph 2007;13(5):925–38.
[19] Zheng Y, Fu H, Au OK-C, Tai C-L. Bilateral normal filtering for mesh denoising. IEEE Trans Vis Comput Graph 2011;17(10):1521–30.
[20] He L, Schaefer S. Mesh denoising via L_0 minimization. ACM Trans Graph 2013;32(4).
[21] Friedman J, Hastie T, Hofling H, Tibshirani R. Pathwise coordinate optimization. Ann Appl Stat 2007;1(2):302–32.
[22] Bertsekas D. Nonlinear programming. Athena Scientific.
[23] Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis. IEEE Trans Pattern Anal Mach Intell 2002;24(5):603–19.
[24] Wang G, Wong T-T, Heng P-A. Deringing cartoons by image analogies. ACM Trans Graph 2006;25(4):1360–79.
[25] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 2004;13(4):600–12.
[26] Xu L, Yan Q, Xia Y, Jia J. Structure extraction from texture via relative total variation. ACM Trans Graph 2012;31(6).