Fast and Effective L_0 Gradient Minimization by Region Fusion

Rang M. H. Nguyen Michael S. Brown
School of Computing, National University of Singapore
nguyenho@comp.nus.edu.sg | brown@comp.nus.edu.sg

Abstract

L_0 gradient minimization can be applied to an input signal to control the number of non-zero gradients. This is useful in reducing small gradients generally associated with signal noise, while preserving important signal features. In computer vision, L_0 gradient minimization has found applications in image denoising, 3D mesh denoising, and image enhancement. Minimizing the L_0 norm, however, is an NP-hard problem because of its non-convex property. As a result, existing methods rely on approximation strategies to perform the minimization. In this paper, we present a new method to perform L_0 gradient minimization that is fast and effective. Our method uses a descent approach based on region fusion that converges faster than other methods while providing a better approximation of the optimal L_0 norm. In addition, our method can be applied to both 2D images and 3D mesh topologies. The effectiveness of our approach is demonstrated on a number of examples.

1. Introduction and Related Work

This paper focuses on L_0 gradient minimization applied to images and 3D meshes. L_0 gradient minimization is used to control the global number of non-zero gradients between neighbors in a graph. When done properly, L_0 gradient minimization effectively creates a series of piecewise constant functions whose transitions correspond to important changes in the original function. This can be considered as a feature-preserving filter that has many applications in computer vision, from image and mesh denoising, to image enhancement and segmentation.

There are a number of feature-preserving filters in the literature. These can be broadly categorized as local or global filters. The trade-off among these various methods lies in their overall effectiveness for the task at hand and time complexity. Well-known local filters include the anisotropic filter [1, 16], the bilateral filter [2, 6, 15, 21, 26, 27], the guided filter [10, 11, 28], and the geodesic filter [9, 4]. These filters were originally designed for use on images, but have been extended to other domains, including 3D meshes [8, 29]. Local filters are popular given their simplicity and effectiveness in smoothing noise while preserving edges, however, they do require the tuning of the filter's local support size. In addition, as discussed in [7], it can often be difficult to use local filters to achieve progressive coarsening.

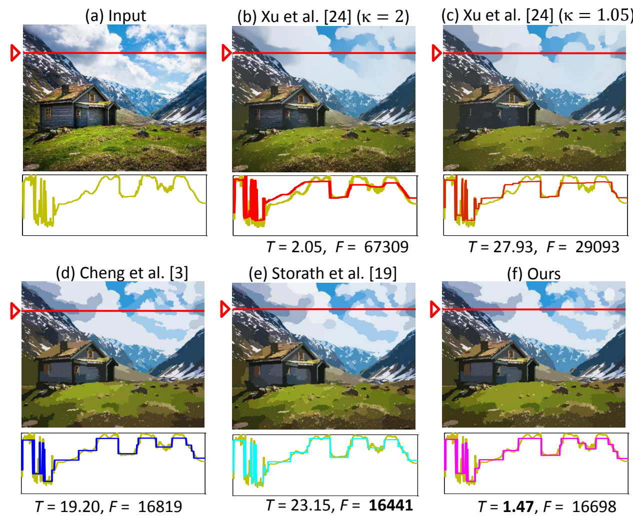

Figure 1. This figure shows the results of applying L_0 gradient minimization on an image by Xu et al. [24], Cheng et al. [3], Storath et al. [19], and our method. Parameters for each method was tuned to produce the best results (closest result to L_0). The plots under each image show a 1D scanline from the green channel. We also report the running time T (in seconds) and the final objective function value F (defined in Equation 2) which includes the L_0 regularization.

L_0 gradient minimization falls into the category of global filters that impose a constraint on the entire image. Most global filters are realized in an optimization framework. One of the best known global methods is the work by Rudin et al. [17] and subsequent variations (e.g. [5, 23, 25]) that introduced the total variation (TV) minimization which essentially imposes an L_1 gradient minimization. Another effective global filter is the weighted-least-square filter [7, 13] that solves an L_2 objective function defined over the entire image, using image affinity based on local image gradients.

Recently, the L_0 gradient minimization was used by Xu et al. [24] for the task of image smoothing. As previously mentioned, this approach aims to limit the number of gradient transitions in the output image. However, in terms of computational complexity, minimizing the L_0 norm is NP-hard [14]. As such, approximation strategies must be used. Xu et al. [24] split their optimization function into two subproblems by introducing auxiliary variables and solved them by an L_0-L_2 iteration framework. A parameter κ was used to balance the influence of the L_0 norm on the final filtered signal. A smaller κ weights the L_0 regularization more, but at the cost of a longer running time. Figure 1 shows an example, where F represents the final optimization energy and T is the time in seconds. Cheng et al. [3] extended this work to propose a better approximation algorithm based on a fused-coordinate descent framework (as shown in Figure 1-d). However, their method is also slow to converge, often needing $700-1000$ iterations. Storath et al. [19] proposed an optimization method based on dynamic programming and alternating direction method of multipliers. Their approach obtains a good approximation of the L_0, but requires a significantly large running time (as shown in Figure 1-e). Shen et al. [18] defined a different optimization function that changed the L_2 norm term in the optimization function into an L_1 norm. As with local filters, the L_0 gradient minimization has been extended beyond images to work with 3D meshes (e.g. He et al. [12], Cheng et al. [3]).

Contribution We present a method for L_0 gradient minimization that is fast and is able to approximate the L_0 norm effectively. Our method uses a descent strategy based on region fusing. We show that at each minimization step, the objective function will decrease monotonically. This allows our method to converge quickly. Moreover, our method can be applied to arbitrary graphs such as image and mesh topologies. We detail our algorithm and demonstrate its effectiveness on a number of experiments showing that it provides a good approximation of the L_0 norm and is significant faster than prior methods.

The remainder of paper is organized as follows. Section 2 overviews the L_0 gradient minimization framework and its application to different graph topologies. Section 3 describes our descent and fusion approach for approximating L_0 gradient minimization. Section 4 evaluates the performance of our method against existing L_0 minimization approaches. Section 5 shows the effectiveness of our approach on a number of applications. The work is concluded in Section 6.

2. L_0 Gradient Minimization

Here we describe L_0 gradient minimization for a general input signal. Let I be the original signal and S represent the filtered output. The gradient of the output S is denoted by ∇S. The objective function of the L_0 gradient minimization is formulated as follows:

$$F = \min_S ||S - I||^2 + \lambda||\nabla S||_0, \tag{1}$$

where $||\cdot||$ denotes the L_2 norm, $||\cdot||_0$ denotes L_0 norm, and λ is the parameter to control the level of sparseness in the final signal S. A larger λ produces a coarser result with less gradient.

Equation 1 can be rewritten as follows:

$$F = \min_S \sum_{i=1}^{M} \left[||S_i - I_i||^2 + \lambda/2 \sum_{j \in N_i} ||S_i - S_j||_0 \right], \tag{2}$$

where M is the length of the signal and N_i is the neighboring set of the i^{th} element. Here, λ is divided by 2 since the neighboring relationship between S_i and S_j is counted twice. The neighboring set N_i is defined for each case (e.g. 1D signal, 2D images, or 3D mesh models) as follows:

$$N_i = \begin{cases} \{i-1, i+1\} & \text{1D} \\ \{\text{four-connected pixels}\} & \text{2D} \\ \{\text{all neighbor faces of the } i^{th} \text{ face}\} & \text{3D} \end{cases}. \tag{3}$$

3. Our Region Fusion Minimization

Our goal is to minimize the objective function introduced in Equation 2. Our method uses a fusion technique that examines neighboring regions in the signal that have nearly similar values and decides whether to fuse them to have the same value. By combining two regions, we create a larger single region, but also remove the gradients between the regions. The work by Cheng et al. [3] used a somewhat similar method termed the fused-coordinate descent. Their work, however, used a more complicated optimization mechanism that separated the fusion step from the coordinate descent step in a way that did not guarantee the objective function to monotonically decrease during the fusion step (see accompanying supplemental material for more details). Their method also required a longer running time. In contrast, our approach combines the fusion and descent into a single step that guarantees a decrease of the objective function and allows our method to converge quickly. Our optimization approach is detailed in the following.

3.1. Optimization

As discussed in Section 1, the inclusion of the L_0 norm in the objective function makes it NP-hard. We approximate this objective function by considering each pair of neighboring elements in the graph one at a time, instead of the entire signal at once. Our descent optimization works as follows. We first assign the output signal S to be the same as the input signal I. Our algorithm then loops through all the signal elements. At each step, we consider two neighboring elements i and j. The amount these two elements contribute to the objective function F in Equation 2 can be expressed as follows:

$$f = \min_{S_i, S_j} ||S_i - I_i||^2 + ||S_j - I_j||^2 + \lambda ||S_i - S_j||_0. \quad (4)$$

Our goal here is to find the best S_i and S_j that minimize the sub-function f. To do this, we divide the problem into two cases: $S_i \neq S_j$, and $S_i = S_j$ to eliminate the L_0 term $||S_i - S_j||_0$ in Equation 4.

- **Case** $S_i \neq S_j$**:** The L_0 term $||S_i - S_j||_0$ is equal to 1 and the function f in Equation 4 becomes:

$$f = \min_{S_i, S_j} ||S_i - I_i||^2 + ||S_j - I_j||^2 + \lambda. \quad (5)$$

  In this case, we have a trivial solution as follows:

$$\begin{cases} S_i = I_i, \quad S_j = I_j \\ f = \lambda \end{cases}. \quad (6)$$

- **Case** $S_i = S_j$**:** The L_0 term $||S_i - S_j||_0$ is equal to 0 and the function f in Equation 4 becomes:

$$f = \min_{S_i} ||S_i - I_i||^2 + ||S_i - I_j||^2. \quad (7)$$

  Equation 7 is a quadratic equation that requires only one variable S_i to be solved. By using the first derivative, its solution can be easily obtained as follows: $S_i = (I_i + I_j)/2$. Therefore, the solution for this case is:

$$\begin{cases} S_i = S_j = (I_i + I_j)/2 \\ f = (I_i - I_j)^2/2 \end{cases}. \quad (8)$$

Combining these two cases together, we have the solution for Equation 4 as follows:

$$\{S_i, S_j\} = \begin{cases} \{A, A\} & \text{if } ||I_i - I_j||^2/2 \leq \lambda \\ \{I_i, I_j\} & \text{otherwise} \end{cases}, \quad (9)$$

where $A = (I_i + I_j)/2$.

We call Equation 9 the *fusion criterion*. Note that Equation 9 still holds true in the case of $I_i = I_j$. According to this fusion criterion, we will decide whether to fuse these two elements into one group or not.

Our overall approach is described in Algorithm 1. Elements in the signal (e.g. pixels in an image or faces on a mesh) are denoted as I_i. Group (connected regions) of elements with the same values will be denoted as G_i. The number of elements in each group is denote as w_i. The number elements that connect group i and j is denote as $c_{i,j}$. To initialize the algorithm, each group G_i contains exactly one element i. Therefore, the number of elements of each group w_i is equal to 1. We use Y_i to store the mean value of all elements in group G_i which is initialized to the original signal I_i. All neighboring groups N_i are initialized using Equation 3. We also define the number of initial connections between two neighboring groups as follows:

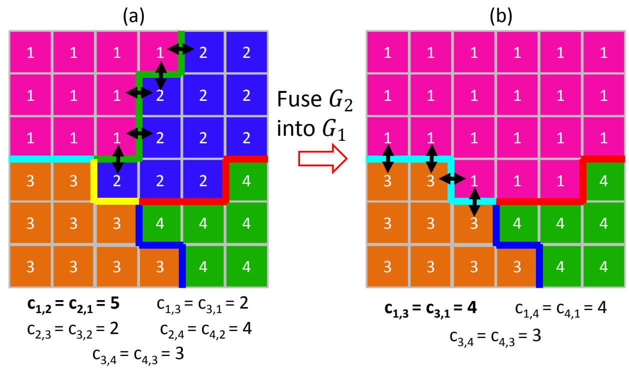

Figure 2. This figure shows an example of the connection numbers for a 2D image. (a) shows the initial configuration with four groups of pixels, while (b) shows the configuration after fusing group G_2 into G_1. The numbers below each image show the corresponding connection numbers for each pair of neighboring groups.

$$c_{i,j} = \begin{cases} 1 & \text{if } j \in N_i \\ 0 & \text{otherwise} \end{cases}. \quad (10)$$

The matrix that represents the connection numbers c is very large, but sparse, since each element has only a few neighbors. Therefore, we can use a sparse matrix representation storing only non-zero values to save memory. Note that the connection number is equivalent to the number of gradients between two neighboring groups. Figure 2-(a) shows an example of group neighbors and the connection numbers for a 2D image. All pixels that belong to the same group have the same numerical value, e.g. G_1 has 10 elements in Figure 2-(a). Groups G_1 and G_2 have five elements connections together (shown along the green line with double arrows), therefore, their connection numbers are $c_{1,2} = c_{2,1} = 5$. Note that these connections are counted twice, once for $c_{i,j}$ and $c_{j,i}$, and account for why λ is divided by two in Equation 2.

Our algorithm loops through all groups of a current filtered signal. For each group G_i, we consider its neighbors G_j. Like prior methods, we use an auxiliary parameter β ($0 \leq \beta \leq \lambda$) that increases for each iteration. Details to this parameter are provided in Section 3.2. Factoring in the auxiliary parameter, Equation 4 becomes as follows:

$$\min_{S_i, S_j} w_i ||S_i - Y_i||^2 + w_j ||S_j - Y_j||^2 + \beta c_{i,j} ||S_i - S_j||_0. \quad (11)$$

Recall that Y_i and Y_j represent the mean signal values for the groups G_i and G_j containing w_i and w_j elements respectively. The above equation can be solved in the exact same manner as described for Equation 4 as follows:

Algorithm 1 Region Fusion Minimization for L_0

Input: signal I of length M, the level of spareness λ

$G_i \leftarrow \{i\}, Y_i \leftarrow I_i, w_i \leftarrow 1$
Initialize N_i as Equation 3
Initialize $c_{i,j}$ as Equation 10
$\beta \leftarrow 0,\ \ iter \leftarrow 0,\ \ P \leftarrow M$
repeat
  $i \leftarrow 1$
  while $i \le P$ **do**
    for all $j \in N_i$ **do**
      if $w_i w_j ||Y_i - Y_j||^2 \le \beta c_{i,j}(w_i + w_j)$ **then**
        $G_i \leftarrow G_i \cup G_j$
        $Y_i \leftarrow (w_i Y_i + w_j Y_j)/(w_i + w_j)$
        $w_i \leftarrow w_i + w_j$
        Remove j in N_i and delete $c_{i,j}$
        for all $k \in N_j \setminus \{i\}$ **do**
          if $k \in N_i$ **then**
            $c_{i,k} \leftarrow c_{i,k} + c_{j,k}$
            $c_{k,i} \leftarrow c_{i,k} + c_{j,k}$
          else
            $N_i \leftarrow N_i \cup \{k\}$
            $N_k \leftarrow N_k \cup \{i\}$
            $c_{i,k} \leftarrow c_{j,k}$
            $c_{k,i} \leftarrow c_{j,k}$
          end if
          Remove j in N_k and delete $c_{k,j}$
        end for
        Delete G_j, N_j, w_j
        $P \leftarrow P - 1,\ i \leftarrow i + 1$
      end if
    end for
  end while
  $iter \leftarrow iter + 1$
  $\beta \leftarrow g(iter, K, \lambda)$ ▷ Defined in Equation 13
until $\beta > \lambda$

for $i = 1 \rightarrow P$ **do** ▷ Reconstruct the output signal
  for all $j \in G_i$ **do**
    $S_j \leftarrow Y_i$
  end for
end for

Output: filtered signal S of length M

$$\{S_i, S_j\} = \begin{cases} \{B, B\} & \text{if } w_i w_j ||Y_i - Y_j||^2 \le \beta c_{i,j}(w_i + w_j) \\ \{Y_i, Y_j\} & \text{otherwise} \end{cases} \tag{12}$$

where $B = (w_i Y_i + w_j Y_j)/(w_i + w_j)$ is the weighted average of the two groups G_i and G_j.

The criterion in Equation 12 is used to decide whether to fuse the group G_j into the group G_i or not. Note that changing the values of groups G_i and G_j can affect to the magnitudes of gradients with their other neighbors but it does not create any new non-zero gradients. Therefore, this will not affect the other terms in Equation 2. In addition, $\beta \le \lambda$, if β satisfies Equation 12, then λ does too. Therefore, the total objective function F in Equation 2 will decrease by $\lambda c_{i,j} - w_i w_j ||Y_i - Y_j||^2/(w_i + w_j)$ if we preform the fusion step, or will remain unchanged otherwise. As a result, our fusion-based method acts as a descent strategy to either lower the objective function or remain the same.

If these groups are fused, all elements in G_j are joined into G_i, then the mean value Y_i and the element number w_i are updated. Next, all the neighbors of G_j are inserted into N_i and the corresponding connection numbers are updated. After each fusion step, all the information of the fused group G_j is deleted and the number of remaining groups is reduced by one. Figure 2-(b) shows an example of how the connection numbers are updated during the fusion step. Here, group G_2 is fused into G_1. The connection numbers $c_{1,3}, c_{3,1}$ (between G_1 and G_3) and $c_{1,4}, c_{4,1}$ (between G_1 and G_4) are updated. All connection numbers related to G_2 are deleted.

Our algorithm is repeated until the auxiliary parameter β reaches the regular parameter λ. At that time, the remaining groups contain all elements in the signal. Finally, the output value for each element S_j will be assigned by the mean value of the group that it belongs to.

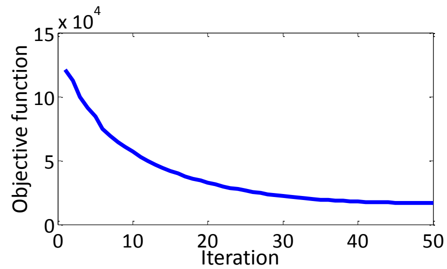

Figure 3. This plot shows the value of the objective function in Equation 2 for each iteration. The input signal is the image shown in Figure 1. The objective function monotonically decreases at each iteration.

Figure 3 shows the value of the objective function, F, in Equation 2 with each iteration. For this example, the input signal is the image shown in Figure 1. The objective function monotonically decreases with each iteration, converging after approximately 50 iterations.

3.2. Auxiliary Parameter β

Like the other prior works [3, 24], we use an auxiliary parameter β which gradually increases from 0 to λ at each iteration. This parameter is used to make pairs of neighboring groups that have small differences in their mean values fuse together. We experimented with three different strategies to increase β: linear, non-linear, and multiplicative, de-

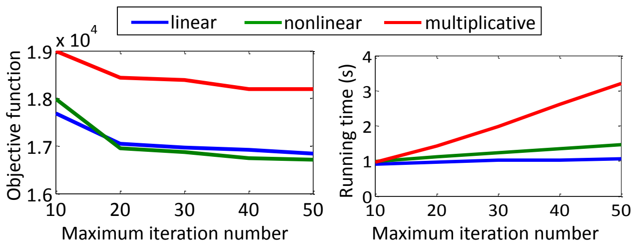

Figure 4. This figure shows the results with different values of the maximum iteration number (e.g. $10, 20, 30, 40, 50$) on three strategies for increasing β, i.e. linear, nonlinear, and multiplicative. The left shows the values for objective function, F, while the corresponding running-time are shown in the right.

fined as follows:

$$g(iter, K, \lambda) = \begin{cases} (iter/K)\lambda & \text{linear} \\ (iter/K)^{\gamma}\lambda & \text{nonlinear} \\ \alpha^{(iter-K)}\lambda & \text{multiplicative} \end{cases}. \quad (13)$$

Figure 4 shows the results with different values for the maximum iteration number K (e.g. $10, 20, 30, 40, 50$). The left plot shows the values for objective function while the corresponding run-time is shown in the right plot. Here, we use $\gamma = 2.2$ for the nonlinear increase function and $\alpha = 1.5$ for multiplication strategy. As shown in Figure 4, the multiplication strategy is the worst in terms of the objective function value and running time. The linear strategy is faster than the nonlinear approach, but has a slightly larger value for the objective function. Based on this, we choose the nonlinear strategy for our L_0 gradient minimization, since it gives us the best approximation to the L_0 norm.

4. Experiments

In this section, we compare our method with the three other L_0 gradient minimizations proposed by Xu et al. [24], Cheng et al. [3], and Storath et al. [19]. The comparison examines the final objective function value F defined in Equation 2 and running-time T. All the experiments are run on a dual core 3.10 GHz PC with 16.0 GB RAM. Our method is implemented in C++. For processing an image sized approximately 600×400, our approach takes roughly 1 second. The other methods [3, 19, 24] are implemented in C++, Java, and Matlab by their authors. Xu et al.'s method [24] has a parameter κ to control the amount of L_0 minimization. A smaller κ gives results with better L_0 minimization, but requires more iterations. In our experiments, we report Xu et al.'s results for both $\kappa = 2$ and $\kappa = 1.05$.

Figure 5 shows the results of each method. As can be seen, the objective function values, F, obtained from Xu et al.'s method are notably large. This is because of L_2 step in their approximation that smooths the signal. As a result, there are still small gradients left in the output signal (shown in the plot in second row). Our results are very close to Storath et al. [19] that are the best in terms of minimizing the objective function, however, our running-time is significantly faster.

5. Applications

In this section, we show our approach applied to several applications involving images and 3D meshes. In particular, we demonstrate: image denoising, content-based color quantization, clip art compression artifact removal, and 3D mesh denoising. *Additional results for each application are also included in the supplemental material.*

5.1. Image Denoising

The L_0 gradient minimization can approximate the input signal by a series of piecewise constant functions. Therefore, it can be used to denoise the 2D images that have sparse colors with sharp edges.

In this experiment, we compare our method against two methods: the total variation method proposed by Dahl et al. [5], and L_0 gradient proposed by Storath et al. [19]. Figure 6 shows two examples of image denoising. Storath et al. [19] obtain the best results in terms of signal-to-noise-ratio (SNR). Our quantitative results are very close to Storath et al.'s results, but again, our running-time is significantly faster.

5.2. Content-Based Color Quantization

Content-based color quantization is used to reduce the number of colors in an image. This is useful for tasks such as image segmentation or image retrieval since it reduces the color complexity of an imaged scene.

Figure 7 shows an example of content-based color quantization. Figure 7-(a) shows the original input image that over $100,000$ different colors (number of colors denote as P). Figure 7-(b)-(e) show our results with different values for the level of spareness λ. Our method reduces the number of colors in image but still keep the overall image structure. Figure 8 shows some examples of content-based color quantization on images. As can be seen, our approach can obtain results with the least number of remaining colors in the fastest time.

5.3. Clip Art Compression Artifact Removal

Using JPEG compression on clip art images often create artifacts as shown in Figure 9-(a). Most of clip art images have sparse colors with sharp edges and artifacts are most noticeable near the sharp edges. Prior works [3, 22, 24] showed that using local filters can reduce these artifacts, but they also tend to blur the edges. The L_0 gradient minimization is well-suited to remove these artifacts.

In this experiment, we compare our method with Wang et al. [22], Xu et al. [24], and Cheng et al. [3]. The work by

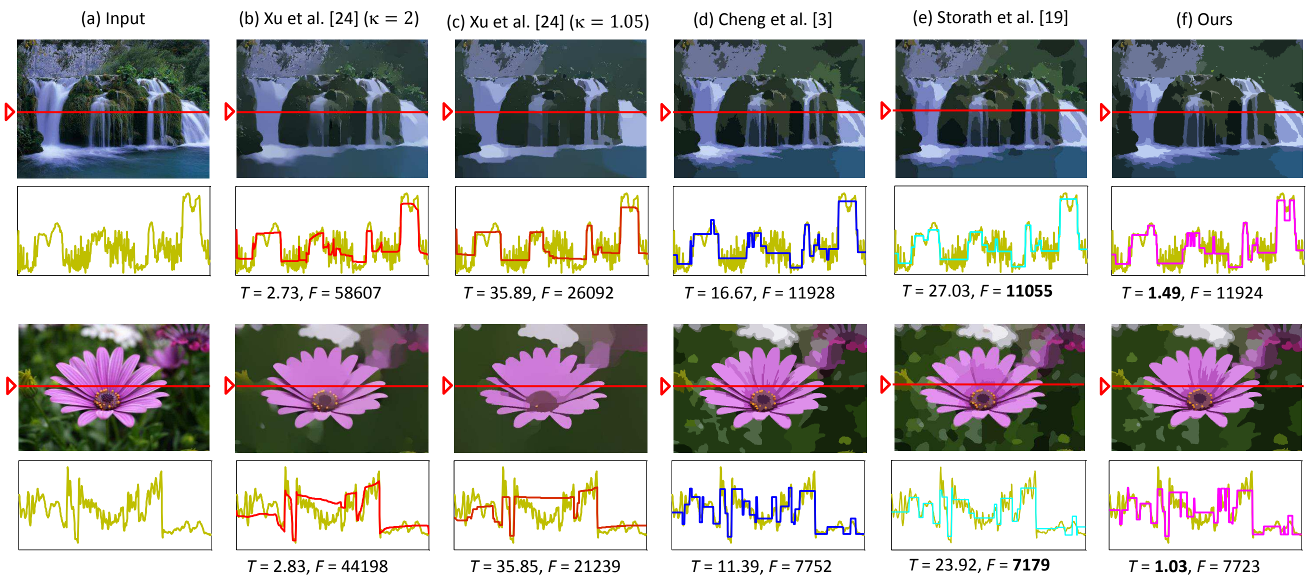

Figure 5. This figure shows the details of the experiment that was already shown in Fig. 1 in Sec. 1. One more result for Xu et al.'s method with $\kappa = 2$ is added in here. (a) Input images. (b)-(c) Results of Xu et al. [24] with different $\kappa = 2$, and $\kappa = 1.05$ ($\lambda = 0.05$). (d) Results of Cheng et al. [3] ($\lambda = 0.2$). (e) Results of Storath et al. [19] ($\lambda = 0.2$). (f) Our results ($\lambda = 0.2$). The plot in the second row shows a 1D scanline from the green channel. We also report the running time (in seconds) and the objective function value defined in Equation 2.

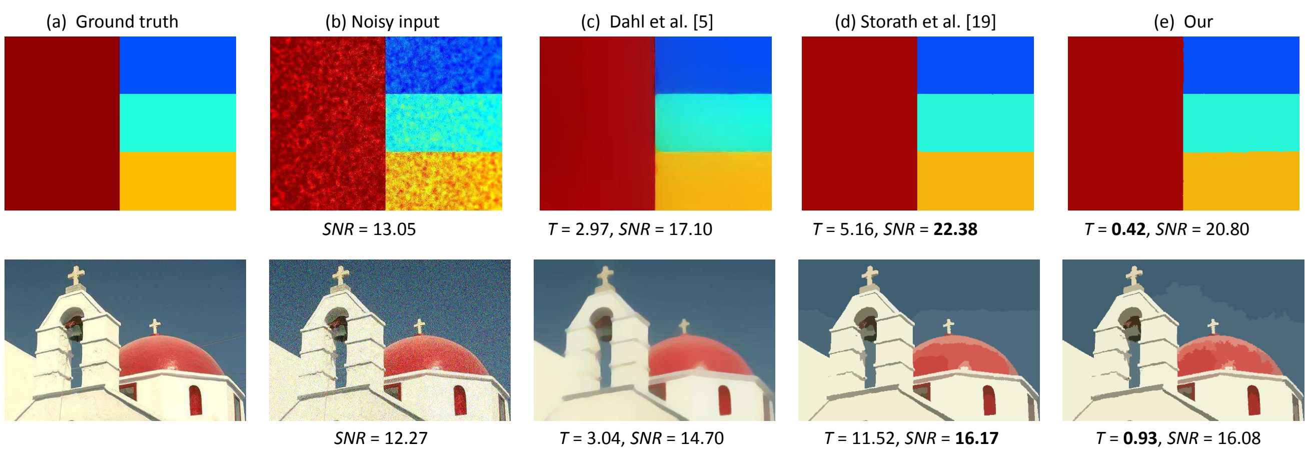

Figure 6. This figure shows two examples of image denosing. (a) Ground truth images. (b) Noisy input images. (c) Results of total variation method proposed by Dahl et al. [5]. (d) Results of Storath et al. [19]. (e) Our results. We also report the running time (in seconds) and the signal-to-noise-ratio SNR (in dB).

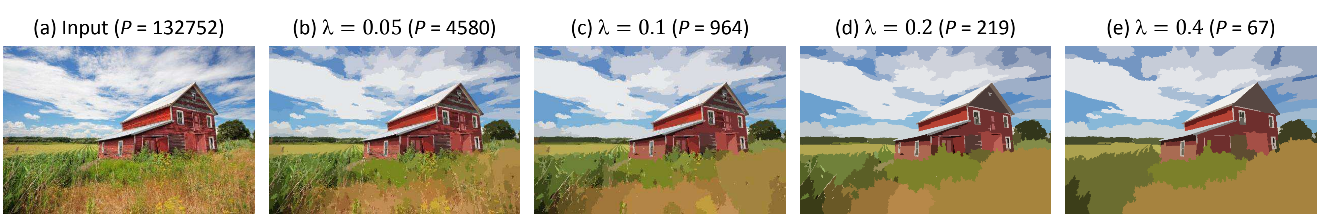

Figure 7. This figure shows an example of content-based color quantization. (a) The original input image with 132752 different colors. (b)-(e) Our results with different values for λ. The numbers reported in parenthesis are the remaining colors P.

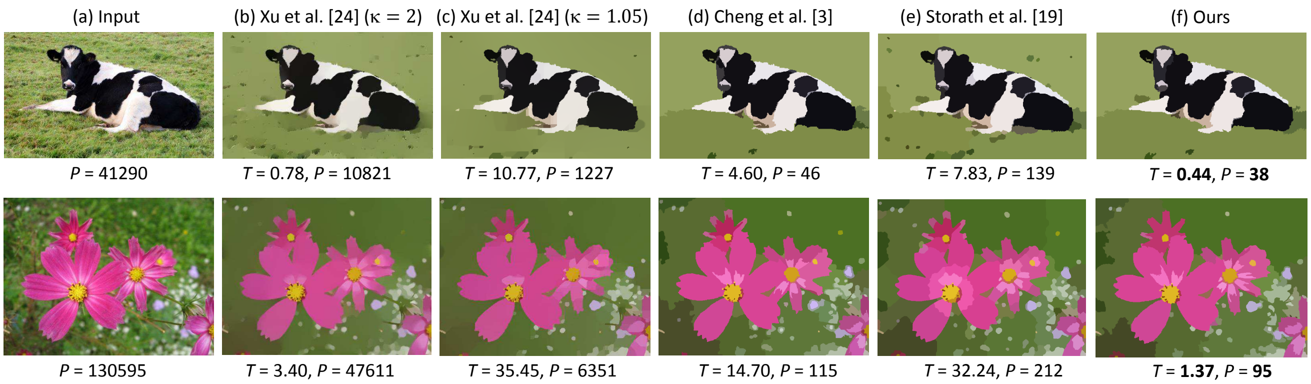

Figure 8. This figure shows several examples of content-based color quantization on images. This is useful for computer vision tasks such as image segmentation or image retrieval. (a) Input images. (b)-(c) Results of Xu et al.'s method with different $\kappa = 2$ and $\kappa = 1.05$. (d) Result of Cheng et al.'s method. (e) Results of Storath et al.'s method. (f) Our results. We also report the running time T (in seconds) and the number of remaining colors P.

Wang et al. [22] is explicitly designed for artifact reduction for JPEG compressed clip art. Figure 9 shows two examples of artifact reduction. Our method obtains comparable results to prior methods but with the fastest running time.

5.4. Mesh Denoising

Our approach can also be applied to 3D mesh topologies. We modified Algorithm 1 to use the two-step framework proposed by Sun et al. [20]. In step one, we use our L_0 gradient minimization to filter the noisy face normals. In step two, the vertex coordinates are reconstructed from the filtered face normals using the iterative updating vertex framework (see [20] for more details).

The nature of the L_0 gradient minimization will approximate the input signal by a series of piecewise constant functions. Therefore, it can be used to denoise the 3D meshes that have sharp transitions between their faces. In this experiment, we test our approach on several sharp-edge meshes and compare our results with the results of three methods: Sun et al. [20], He et al. [12], and Cheng et al. [3]. Sun et al.'s method is a local filtering method based on the weighted averaging of neighboring face normals. He et al.'s method is an extension of Xu et al.'s method [24] to work on 3D meshes. Figure 10 shows two examples of denoising on 3D mesh models. Here, the noisy 3D models are synthesized from the ground truth ones. The results show that our approach produces the best results and with the fastest running time.

6. Concluding Remarks

We have described a fusion-based descent method for L_0 gradient minimization. The nature of L_0 gradient minimization will approximate the input signal by a series of piecewise constant functions. Therefore, applying L_0 minimization to smooth signals will create artifacts that may be undesirable. For example, Figure 11 shows the an example for the task of image smoothing. The L_0-L_2 iterative strategy proposed by Xu et al. [24] is arguably more appropriate than our approach for this task. Similarly, for the mesh models with smooth surfaces, our method creates a sharp mesh instead of a smooth one. He et al.'s method [12] based on Xu et al.'s strategy is more suitable when smooth regions are present (see Figure 12). However, for tasks for which L_0 gradient minimization is needed, the proposed method in this paper offers a fast and effective approach that can be applied to both 2D images and 3D mesh topologies.

Figure 11. This figure shows the drawback of strong L_0 gradient minimization for images. For the task of image smoothing, the method by Xu et al. [24] may be more appropriate, as our method will introduce more noticeable color quantization into the result.

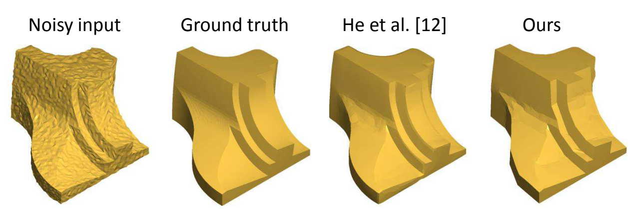

Figure 12. This figure shows the drawback of strong L_0 gradient minimization for 3D meshes. If the original input has smooth surfaces, these will be quantized to sharp edges, which could be undesirable for certain tasks.

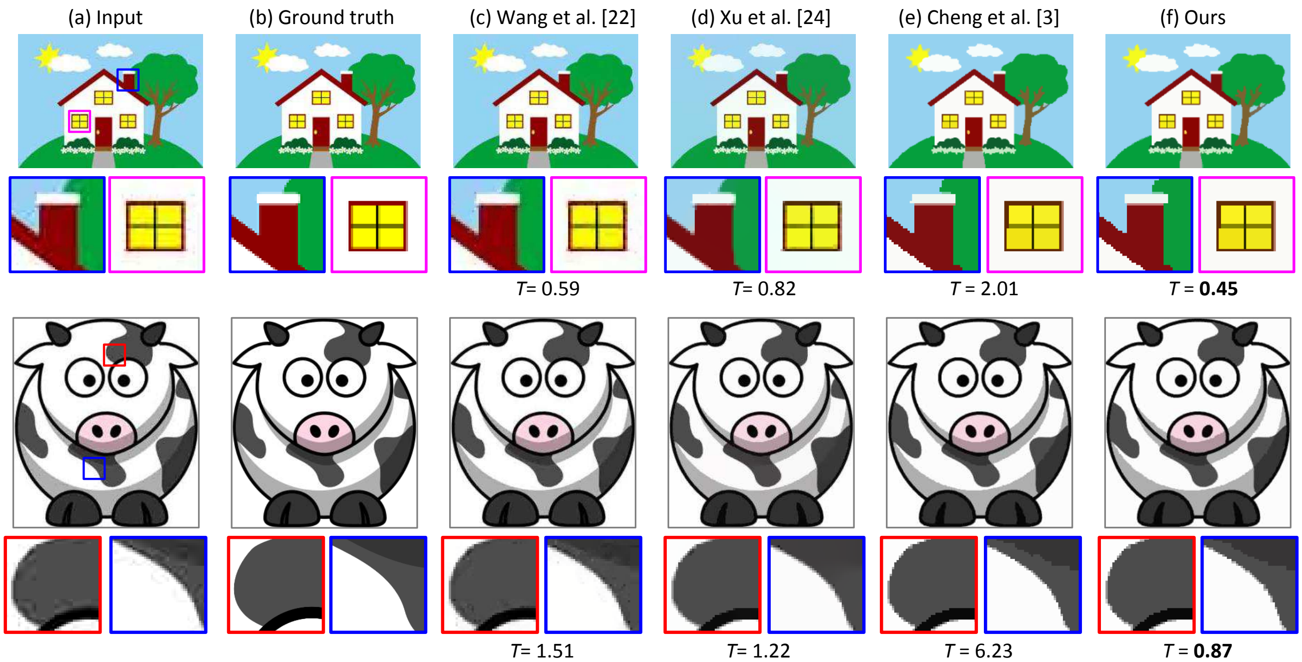

Figure 9. This figure shows two examples of JPEG compression artifact removal on clip art images. The first and third rows show the complete images, while the second and fourth rows shows the corresponding close-ups in these images. (a) Input JPEG compression artifact images. (b) The ground truth images without compression. (c) Results of Wang et al. [22]. (d) Results of Xu et al. [24] ($\lambda = 0.025$, $\kappa = 1.5$ for the first image, and $\lambda = 0.01$, $\kappa = 1.5$ for the second image). (e) Results of Cheng et al. [3] ($\lambda = 0.26$ for the first image, and $\lambda = 0.08$ for the second image). (f) Our results ($\lambda = 0.26$ for the first image, and $\lambda = 0.08$ for the second image).

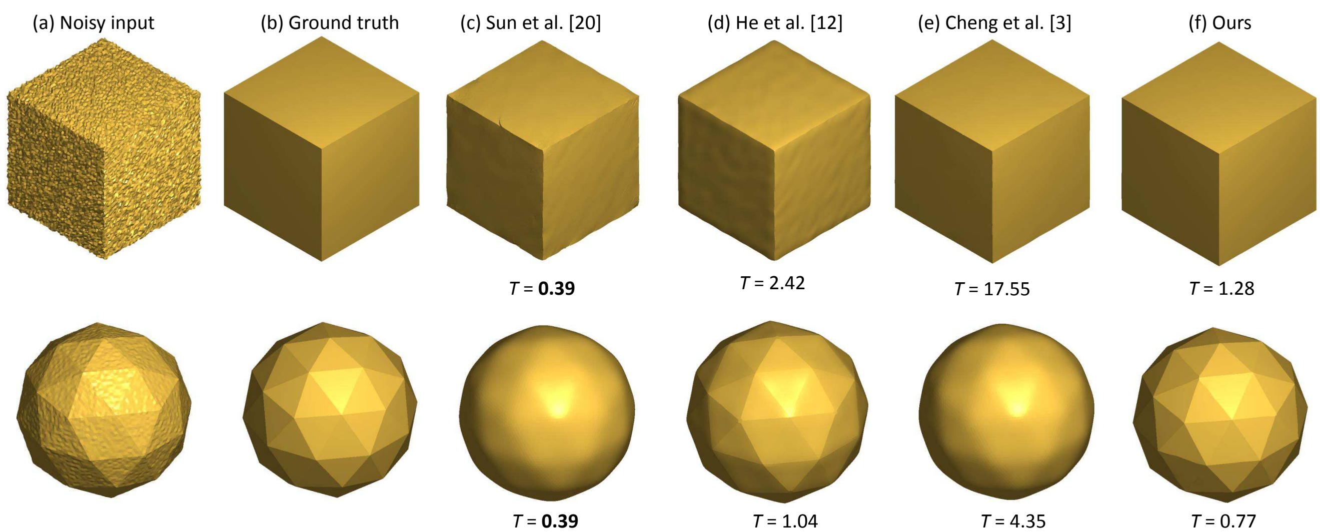

Figure 10. This figure shows two examples of denoising 3D mesh models. The first model contains 24578 vertices and 49152 faces, while the second one contains 10242 vertices and 20480 faces. (a) Noisy 3D mesh models. (b) The ground truth meshes. (c) Results of Sun et al. [20] ($\tau = 0.55$ for the first model, and $\tau = 0.4$ for the second model). (d) Results of Cheng et al. [3] ($\lambda = 0.5$ for the first model, and $\lambda = 0.04$ for the second model). (e) Our results ($\lambda = 0.5$ for the first model, and $\lambda = 0.04$ for the second model). We also report the running-time T (in seconds) for each method.

References

[1] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger. Robust anisotropic diffusion. *Image Processing, IEEE Transactions on*, 7(3):421–432, 1998.

[2] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. volume 26, page 103. ACM, 2007.

[3] X. Cheng, M. Zeng, and X. Liu. Feature-preserving filtering with L_0 gradient minimization. *Computers & Graphics*, 38:150–157, 2014.

[4] A. Criminisi, T. Sharp, C. Rother, and P. Pérez. Geodesic image and video editing. *ACM Transactions on Graphics*, 29(5):134, 2010.

[5] J. Dahl, P. C. Hansen, S. H. Jensen, and T. L. Jensen. Algorithms and software for total variation image reconstruction via first-order methods. *Numerical Algorithms*, 53(1):67–92, 2010.

[6] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3):257–266, 2002.

[7] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. volume 27, page 67. ACM, 2008.

[8] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. volume 22, pages 950–953. ACM, 2003.

[9] E. S. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. volume 30, page 69. ACM, 2011.

[10] K. He, J. Sun, and X. Tang. Guided image filtering. In *ECCV*, pages 1–14. Springer, 2010.

[11] K. He, J. Sun, and X. Tang. Guided image filtering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(6):1397–1409, 2013.

[12] L. He and S. Schaefer. Mesh denoising via L_0 minimization. *ACM Transactions on Graphics*, 32(4):64, 2013.

[13] R. L. Lagendijk, J. Biemond, and D. E. Boekee. Regularized iterative image restoration with ringing reduction. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(12):1874–1888, 1988.

[14] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[15] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*, pages 568–580. Springer, 2006.

[16] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[17] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[18] C.-T. Shen, F.-J. Chang, Y.-P. Hung, and S.-C. Pei. Edge-preserving image decomposition using l1 fidelity with l0 gradient. In *SIGGRAPH Asia 2012 Technical Briefs*, page 6. ACM, 2012.

[19] M. Storath, A. Weinmann, and L. Demaret. Jump-sparse and sparse recovery using potts functionals. *Signal Processing, IEEE Transactions on*, 62(14):3654–3666, 2014.

[20] X. Sun, P. L. Rosin, R. R. Martin, and F. C. Langbein. Fast and effective feature-preserving mesh denoising. *Visualization and Computer Graphics, IEEE Transactions on*, 13(5):925–938, 2007.

[21] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[22] G. Wang, T.-T. Wong, and P.-A. Heng. Deringing cartoons by image analogies. *ACM Transactions on Graphics*, 25(4):1360–1379, 2006.

[23] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[24] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L_0 gradient minimization. volume 30, page 174. ACM, 2011.

[25] J. Yang, W. Yin, Y. Zhang, and Y. Wang. A fast algorithm for edge-preserving variational multichannel image restoration. *SIAM Journal on Imaging Sciences*, 2(2):569–592, 2009.

[26] Q. Yang. Recursive bilateral filtering. In *ECCV*, pages 399–413. Springer, 2012.

[27] Q. Yang, K.-H. Tan, and N. Ahuja. Real-time O (1) bilateral filtering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 557–564. IEEE, 2009.

[28] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, pages 815–830. Springer, 2014.

[29] Y. Zheng, H. Fu, O.-C. Au, and C.-L. Tai. Bilateral normal filtering for mesh denoising. *Visualization and Computer Graphics, IEEE Transactions on*, 17(10):1521–1530, 2011.