SOFT CLUSTERING GUIDED IMAGE SMOOTHING

Anonymous ICME submission

Paper ID: 1232

ABSTRACT

Image smoothing, which aims to remove unwanted textures and preserve desired structures, plays an important role in many multimedia and computer vision tasks. The key to image smoothing, despite different applications, is to distinguish the structures from the textures. This paper presents a novel image smoothing method, following the principle that, for a certain pixel, its neighbors in both space and intensity should contribute more on smoothing, while the distant ones be insulated for avoiding over-smoothing. Intuitively, clustering is a good candidate to achieve the goal. However, due to rich textures and clutters within images, simply performing the clustering on the input likely obtains inaccurate results, and thus leads to unsatisfied smoothing results. In addition, for our task, using traditional hard clustering techniques is at high risk of generating staircase artifacts. For addressing these issues, an algorithm is customized, which on the one hand adopts the soft clustering to more faithfully assign pixels, on the other hand iterates the soft clustering and smoothing, expecting to improve each other. Experiments on several challenging images are provided to show the efficacy of our method, and its superiority over other prevailing approaches.

***Index Terms*—** Image smoothing, texture removal, edge-preserving, soft clustering

1. INTRODUCTION

Edge-preserving image smoothing is of critical importance to a great deal of multimedia, computer vision and graphics tasks, which attempts to smooth away textures whilst retaining sharp edges in images. Considering many scene properties, *e.g.* depth, color, object category and matte, are correlated within smooth regions of an image while differing across discontinuities in the image, it is common to smooth a *rough* property map (also known as the filter input) without crossing strong edges. For example, in texture enhancement, the core mission is to separate structural edges and textures from an input image (under the circumstances, the input itself acts as the rough map), and then magnify the textures to accomplish the task; in image colorization [3], the rough map is the user-scribble map and the reference is the original gray-scale image; in image dehazing [4], the transmission map obtained by drak-channel-prior and the foggy image server as these two image; in joint upsampling [5, 6], the low-resolution output and the high-resolution image are rough map and reference image respectively. *Although for different applications, the goals are diverse, their performance unanimously depends on how well the structures to preserve are distinguished from the textures to eliminate.*

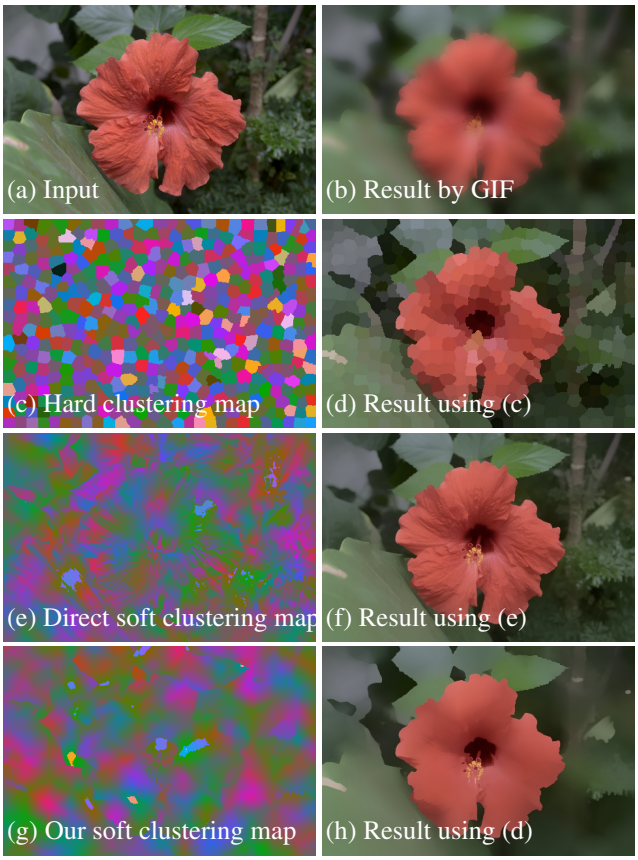


Fig. 1: (a) Input. (b) Result by GIF [1]. (c) and (d) Iteratively refined hard clustering (SLIC superpixel [2]) map and corresponding result. (e) and (f) Direct soft clustering map and corresponding result. (g) and (h) Iteratively refined soft clustering map and corresponding result.

1.1. Related Work

Many image smoothing approaches have been developed over the past decades. Among others, linear translation-invariant

filters using explicitly designed kernels, *i.e.* the Gaussian and Laplacian kernels [7], are arguably the most efficient ones. But these spatial invariant kernels cannot effectively distinguish the structures and textures, and thus result in poor results or even fail in practical scenarios.

Certainly, it is beneficial if having some guidance information. In this paper, we call the smoothers that require guidance information the guided smoothers (or guided filters). In addition, we name the smoothers using a single input itself as guidance, as the self-guided smoother. As a classical self-guided smoother, the bilateral filter (BF) [8] is able to maintain strong edges while removing weak textures, via handling a target pixel through averaging its neighbors. The importance of the neighbors is measured by the Gaussian of both spatial and intensity distances. But BF often suffers from gradient reversal artifacts.

To mitigate this kind of artifact, the gradient has been employed as guidance. It is intuitive to do so as the pixels with larger gradient magnitudes have higher possibilities of being on the structural edges, and vice versa. A representative work in this category is the weighted least squares [9], which shows the superiority over the total variation minimization method [10]. To further boost the performance, Xu *et al.* employed the L_0 regularizer, expecting to form hard weights, to enhance the sparsity of gradients [11]. Rolling guidance filter, recently proposed by Zhang *et al.* [12], alternatively recalls the strong edges from the previously smoothed result, inspired by the scale-space theory. Though RGF is efficient, its ability is limited by its inaccurate edge localization.

If another image, instead of the input, can provide useful information, the structural details in the reference image can be regarded as a prior and transferred to the target output during filtering. It is natural to adopt such information to act as guidance. We call this kind of methods the joint guided smoothers (or joint guided filters). The joint bilateral filter [13] computes the weights from the reference rather than the filtered image. Even though, the problem of gradient reverse artifacts still remains. He *et al.* proposed an approach, called guided image filter (GIF) [1]. GIF is locally a linear transform of the guidance image, which has proven the flexibility and the effectiveness in coping with a bunch of tasks, such as HDR compression, image matting, and dehazing. But GIF exposes its shortcoming of producing halo artifacts near edges.

1.2. Contribution

As aforementioned, image smoothing heavily depends on how well the structures are distinguished from the textures. Figure 1 (a) gives an example natural image. This work tries to, for a certain pixel, find the neighbors in both space and intensity, to contribute on smoothing, in a clustering manner. But, applying traditional hard clustering (SLIC [2] for example) on the image introduces heavy staircase artifacts into results, as shown in Fig. 1 (c) and (d). To more faithfully assign neighbors, we employ a soft clustering scheme. However, due to rich textures and clutters within the image, directly performing the soft clustering on the input leads to inaccurate clustering results, like in Fig. 1 (e). The smoothing result using such a low-quality clustering map is barely satisfied, as shown in Fig. 1 (f). To fix this problem, we iteratively alternate the clustering and smoothing, and can obtain significant better results as shown in Fig. 1 (g) and (h). In experiments, we will compare with other state-of-the-art methods on a number of challenging images to reveal the advances of our design.

2. OUR METHOD

This section first revisits and analyzes the guided image filter (GIF) [1]. Based on the analysis, we then explain our method in details to see how and why it can improve GIF on the task of image smoothing.

2.1. Guided Filter Revisit

Given a reference image $\mathbf{R} \in \mathbb{R}^{m\times n}$, where m and n represent the height and width of the image, respectively. The key assumption of the guided image filter is a local linear model between $\mathbf{R}$ and the filter output $\mathbf{Y} \in \mathbb{R}^{m\times n}$. For the sake of clarity, we first define the neighboring matrix $\mathbf{S} \in \mathbb{R}^{K\times K}$ ($K = m \times n$) as follows:

$$\mathbf{S}_{ij} = \begin{cases} 1 & , \quad j \in \omega_i \\ 0 & , \quad \text{otherwise} \end{cases}, \tag{1}$$

where i, j are pixel indices and ω_i is a local window centered at i. In fact, $\mathbf{S}$ is a highly sparse matrix as, in the ith row of $\mathbf{S}$, only the entries that indexed by pixel $j \in \omega_i$ are 1. Having $\mathbf{S}$ defined, the local linearity can be formulated as:

$$\mathbf{S}_k\mathbf{y} = a_k \cdot \mathrm{Diag}(\mathbf{S}_k) \cdot \mathbf{r} + b_k, \forall k \in \{1, 2, \cdots, K\}, \tag{2}$$

where a_k and b_k are linear coefficients assumed to be constant in local window ω_k. In addition, $\mathbf{y}$ and $\mathbf{r}$ are vectorized versions of $\mathbf{Y}$ and $\mathbf{R}$, respectively. $\mathbf{S}_k$ is the kth row of $\mathbf{S}$, $\mathrm{Diag}(\cdot)$ is a math operator that returns a square diagonal matrix with the elements of a given vector. To obtain a_k and b_k, GIF minimizes the difference between $\mathbf{y}$ and the filter input $\mathbf{x}$ by optimizing the following cost function:

$$E(a_k, b_k) = ||a_k \cdot \mathrm{Diag}(\mathbf{S}_k) \cdot \mathbf{r} + b_k - \mathrm{Diag}(\mathbf{S}_k)\mathbf{x}||^2 + \epsilon a_k^2, \tag{3}$$

where ϵa_k^2 is to prevent a_k from over-large values. As can be seen from Eq. (3), it is a classic Weighted Least Square (WLS) problem, the solution of which can be obtained by:

$$a_k = \frac{(\mathbf{S}_k^2)^\top(\mathbf{r} \circ \mathbf{x}) - \bar{\mathbf{r}}_k\bar{\mathbf{x}}_k}{\sigma_k^2}, \tag{4}$$

$$b_k = \bar{\mathbf{x}}_k - a_k\bar{\mathbf{r}}_k, \tag{5}$$

where $\mathbf{S}_k^2$ stands for element-wise square of $\mathbf{S}_k$, and the operator $\circ$ designates the Hadamard product, $\bar{\mathbf{r}}_k = (\mathbf{S}_k^2)^\top \mathbf{r}$, $\bar{\mathbf{x}}_k = (\mathbf{S}_k^2)^\top \mathbf{x}$, $\sigma_k^2 = (\mathbf{S}_k^2)^\top (\mathbf{r}^2 - \bar{\mathbf{r}}^2)$. By solving Eq. (3) for each $k \in \{1, 2, \cdots, K\}$, we obtain stacked vectors $\mathbf{a} \in \mathbb{R}^{K\times 1}$ of a_k and $\mathbf{b} \in \mathbb{R}^{K\times 1}$ of b_k for each ω_k. The coefficients for each pixel are calculated by averaging all the windows containing a specified pixel as follows:

$$\begin{array}{rcl} \hat{\mathbf{a}} & = & \mathbf{S}^\top \mathbf{a} / \mathbf{S}^\top \cdot \mathbf{1} \\ \hat{\mathbf{b}} & = & \mathbf{S}^\top \mathbf{b} / \mathbf{S}^\top \cdot \mathbf{1} \end{array}, \tag{6}$$

where $\mathbf{1}$ is an all-one vector with compatible size. The filter output of GIF turns out to be $\mathbf{y} = \hat{\mathbf{a}} \circ \mathbf{r} + \hat{\mathbf{b}}$.

Actually, GIF assumes that a single linear model between the guidance image and filter output in a local window. Consider the case where $\mathbf{r} \equiv \mathbf{x}$. In this case, $a_k = \sigma_k^2/(\sigma_k^2 + \epsilon)$ in Eq. (4). Suppose pixel i lies in a strong edge in $\mathbf{R}$ and the gradient is expected to be maintained in the output, *i.e.* $\nabla \mathbf{y}_i \approx \nabla \mathbf{r}_i$. When a local window ω_k contains pixel i as well as the whole edge. This means $\mathbf{r}_k$ has a high variance, *i.e.* we have $\sigma_k^2 \gg \epsilon$. Consequently, $a_k \approx 1$ and $\nabla \mathbf{y}_i = a_k \nabla \mathbf{r}_i \approx \nabla \mathbf{r}_i$. On the contrary , when another local window ω_u also contains pixel i but only a small portion of the edge, and appears to be mostly flat, which means that $\mathbf{r}_u$ is of low variance, then we have $\sigma_u^2 \ll \epsilon$. So $a_u \approx 0$ and $\nabla \mathbf{y}_i = a_u \nabla \mathbf{r}_i \approx 0$. The final $\hat{\mathbf{a}}_i$ is obtained by averaging a_k and a_u, which will decrease the original gradient $\nabla \mathbf{r}_i$. By setting a small window can reduce the gradient decrease to some extent, but weak textures will also be preserved. In summary, due to the averaging process in Eq. (6), the gradient magnitude in output corresponding to strong edges in guidance is weakened. We show a simple 1D illustration in Fig. 2.

This magnitude decrease lies in the linear assumption between guidance and output in local patches. Suppose a patch contains very bright pixels and very dark pixels separated by an edge, the relationship between input and output may be very different on either side of that edge. A linear model on the "high variance" patch cannot capture this difference. Thus, to preserve the gradient magnitude, we can fit a separated linear model to each intensity range present in a patch. A similar approach is used by Chen [5] to map input images to the output of a large range of non-linear operators.

2.2. Soft Clustering Guided Image Smoothing

Compared to GIF, where each row of $\mathbf{S}$ actually is a cluster of all pixels' indices in a local window, we can rebuild $\mathbf{S}$ to group pixels of different intensity ranges to different rows and solve the corresponding a_k and b_k. The main functionality of $\mathbf{S}$ is to aggregate pixels that have both similar spatial positions and intensity ranges. To this end, a straightforward idea is applying hard clustering techniques. However, the hard manner will inevitably introduce artifacts around the boundaries between clusters. Figure 1 (c) and (d) reveal the artifacts, which employs the SLIC superpixel [2] as the clustering tool.

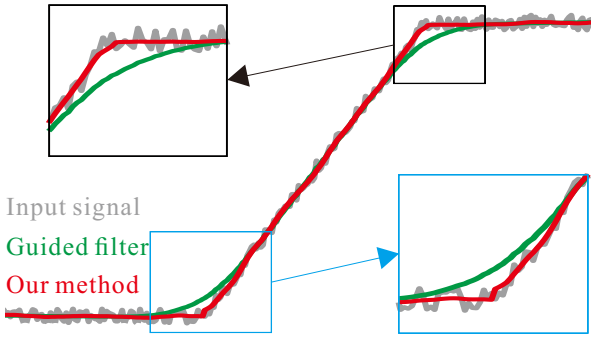


Fig. 2: 1D illustration of the decrease of the gradient using GIF and our method. The gradient of an edge is averaged by the surrounding smooth local patch.

In the sequel, we alternatively resort to soft clustering to mitigate this issue. As a soft clustering algorithm, permutohedral lattice [14] is first proposed to accelerate bilateral filtering. Filtering with permutohedral lattice works by "splatting" a value at each pixel onto a set of high-dimensional vertices, performing a blur in the space of vertices, and "slicing" out values at each pixel to get a filtered set of values. Typically, the value at pixel i is a 5-dimensional vector $\mathbf{U}_i = (x_i, y_i, r_i, g_i, b_i)$, where (x_i, y_i) is the spatial coordinate and (r_i, g_i, b_i) is the channel-wise intensity. In fact, the vertices in high-dimension is an weighted aggregation of pixels with similar intensities and positions. The vertices in permutohedral lattice are arranged as tetrahedral and the weights are calculated using barycentric interpolation. Suppose the whole 5-dimensional space is tessellated into K^p vertices $V = \{v_1, \cdots, v_{K^p}\}$ using permutohedral lattice representation, and $\mathbf{U}_j$ falls into the tetrahedral constructed by vertices set $T_j = \{v_{i_1^j}, \cdots, v_{i_6^j}\} \subset V$, where $i^j_{1,\cdots,6} \in \{1, 2, \cdots, K^p\}$ are indices of coordinate of 6 vertices of the tetrahedral in 5-dimensional space, we use $b_1^j, \cdots, b_6^j$ to denote the barycentric coordinates of $\mathbf{U}_j$, *i.e.* $\mathbf{U}_j = \sum_{z=1}^6 b_z^j v_{i_z^j}$ and $\sum_{z=1}^6 b_z^j = 1$. Then j^{th} column of the neighboring matrix based on PL method are constructed by putting b_z^j to $\#i_z^j$ row, *i.e.* $\mathbf{S}_{i_z^j j} = b_z^j$. According to [15], $\mathbf{S}$ is the so called *splatting matrix* and the transpose of it $\mathbf{S}^\top$ is *slicing matrix*. In each column of $\mathbf{S}$, there are 6 non-zero entries which correspond to the weights of vertices in a 5-dimensional tetrahedral. As sparse matrices, the main functionality of $\mathbf{S}$ is to aggregate pixels that have both similar spatial positions and range intensities. Each row of $\mathbf{S}$ corresponds a vertex in bilateral space and only the pixels that belong to the vertex are non-zero entries.

Using the direct soft clustering result to guide the smoothing is inadequate to generate satisfactory results, as shown in Fig. 1 (e) and (f), because the clustering accuracy is affected by rich textures and clutters. To fix this problem, we propose to iteratively perform soft clustering and smoothing to obtain

significantly better results, as shown in Fig. 1 (g) and (h).

Analysis on S. The neighboring matrix created by GIF does not take intensity difference between pixels in guidance when averaging linear coefficients using Eq. (6). In fact, by interpreting $\mathbf{S}^\top$ as a similarity matrix of different pixels' a_k and b_k, Eq. (6) is also a filtering process. An isotropic filtering, either by box filtering or by gaussian filter proposed in [16], will unsharpen the coefficients of edges. By assuming a separated linear model to different intensity ranges, we actually add a structural prior in the "filtering" process presented by Eq. (6). We show the structural information coded in $\mathbf{S}$ in Fig. 1 (e) and (g), where each row of the neighboring matrix is assigned a random color and each color in this visualization corresponds to a coefficient pair need to solve. It is clearly that the coefficients is edge-aware in our algorithm. The price of obtaining edge-aware coefficients is changing the linear filtering in Eq. (6) to a non-linear one. As mentioned in [16], Eq. (6) can be implemented by a simple box filter or gaussian filer. Instead, in our algorithm, this process is dependant on the construction of $\mathbf{S}$.

Relationship to Bilateral Guided Filtering. In [5], the authors also proposed to fit a separated linear model to different intensity ranges for mapping the input color to the output of an operator. It enforces both spatial and intensity constraints on the linear coefficients in bilateral grid. It solves a global minimization to compute the coefficients for each pixel. Instead, we explicitly model the relationship between the coefficients in different "vertices" in $\mathbf{S}$.

3. EXPERIMENTS

As an important feature, our algorithm can preserve the distinguished structure while removing weak textures when smoothing. In the first experiment, we show some smoothing results of our algorithm as well as the state-of-the-art ones in Fig. 3 to demonstrate the effectiveness of our algorithm. To make the comparison fair, we adopt *smoothing level* [19], which is the normalized difference between filter input and output and can be formally defined as $\|\mathbf{x}-\mathbf{y}\|_2 / \|\mathbf{y}\|_2$. We tune the parameter(s) for each method to reach a similar smoothing level (~ 0.1). In Fig. 3(a) and (b), we show that our algorithm can preserve the details marked in green box while smoothing out the other non-significant textures. We observe GIF [1] can not preserve the edges well when intending to smooth out large texture regions (background in Fig. 3(a) and body of donkey in Fig. 3(b)). To achieve the same smoothing level as our algorithm, no algorithm can preserve the small structures in green rectangles well. Note that, in Fig. 3(b), although L0 [11] maintains the hair in donkey, the weak textures in the body also preserved. Most other algorithms can not preserve this thin structure well when smoothing.

In Fig. 3(c), the input image is a heavily degraded image due to the JPEG compression artifacts (quality factor is 5). Our algorithm can be used to smooth out these blocky artifacts while preserving notable small structures which are also boxed in green rectangles. RGF [12] preserve the small structures, but there is an edge-shifting effect (edge is not localized correctly) in the result. Though the smooth level of muGIF [19] is higher than us, the blocks is not smoothed out.

Fig. 4: Illustration of the iterative soft clustering and smoothing process. Small structures boxed by blue rectangle are not smoothes during iteration while the texture in green is progressively smoothed out during iteration. Iteration number and smoothing level are labeled on bottom.

Fig. 5: Example application on smoothing face while preserving small structures on eyelash.

As described in Sec. 2.2, the iterative process of soft clustering and smoothing is important to our algorithm. To demonstrate this, we show the intermediate smoothing results of Fig. 1(a) in Fig. 4. In the green rectangle region in Fig 4, there contains only weak textures while in the blue rectangle region, the stamen of the flower of a small thin structure. Our algorithm can iteratively smooth green rectangle region while preserving the structure in blue rectangle region. As the iteration going on, more and more weak textures are smoothed out while the distinguished structures are preserved.

We apply our algorithm on retouching face photos, in which we need to smooth skin while preserving small structures like eyelash, eyebrow *etc.*. In Fig. 5 we show an example of this application. The freckles on the face of women is removed while the eyelash is preserved.

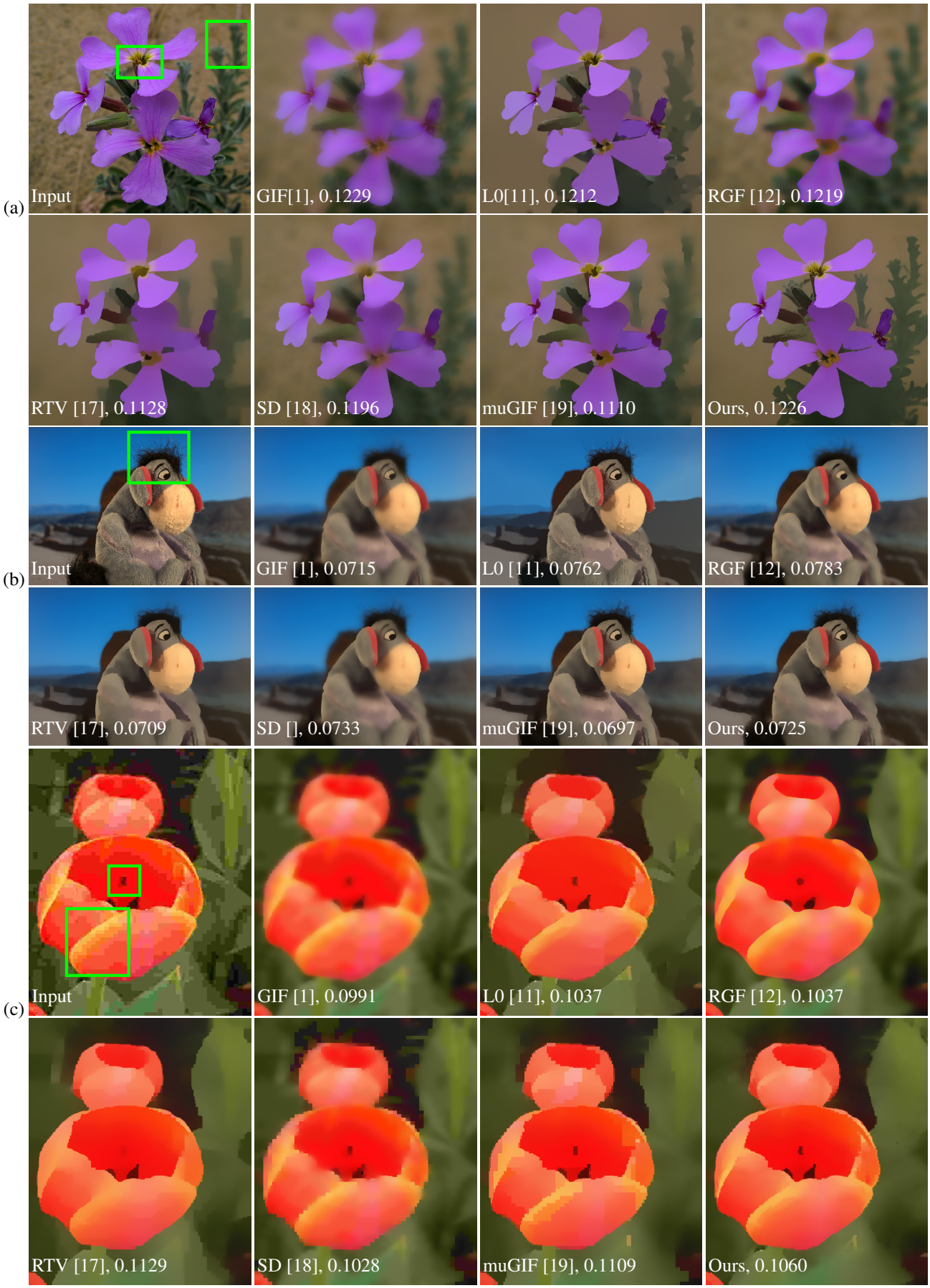
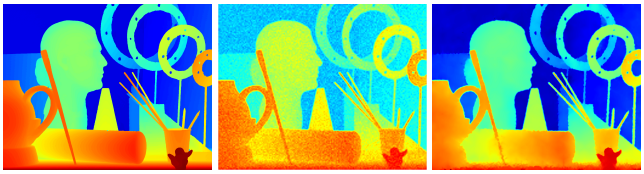


Fig. 3: Visual comparison, with the difference value defined as $\|\mathbf{x}-\mathbf{y}\|_2 / \|\mathbf{y}\|_2$ labeled on bottom.



(a) Groundtruth (b) Bicubic as input (c) Result

Fig. 6: Depth recovery using using our algorithm.

We have also test our algorithm on depth recovery. Given a low resolution noisy depth map as input, we use the self-guided manner to recovery the high resolution one. Figure 6 shows the result, where we use an noisy depth map of down sample factor 8 to recover the original one. We first upsample the map using bicubic interpolation and then smooth it with itself as the guidance. As a baseline, MAD of bicubic interpolation is 4.47 and our algorithm achieves 2.10. which is comparable with the state-of-the-art one.

4. CONCLUSION

In this work, we have proposed a novel image smoothing method. The key principle is that the neighbors in both space and intensity of a pixel should contribute more on smoothing, while the distant ones should be insulated for preventing from over-smoothing. Intuitively, clustering is a good candidate to achieve the goal. We have empirically proven that, because of rich textures and clutters within images, simply performing the clustering on the input cannot generate satisfied results. In addition, using traditional hard clustering techniques is at high risk of generating staircase artifacts. To mitigate these issues, an algorithm has been customized, which on the one hand adopts the soft clustering to more faithfully assign pixels, on the other hand iterates the soft clustering and smoothing. Experimental results have been provided in comparison with the state-of-the-art alternatives to demonstrate the advantages of our method.

5. REFERENCES

[1] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. PAMI*, vol. 35, no. 6, pp. 1397–1409, 2013.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC: Superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. PAMI*, vol. 34, no. 11, pp. 2274–2281, 2012.

[3] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph*, vol. 23, no. 3, pp. 689–694, 2004.

[4] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," in *CVPR*, 2009.

[5] J. Chen, A. Adams, N. Wadhwa, and S. Hasinoff, "Bilateral guided upsampling," *ACM Trans. Graph*, vol. 35, no. 6, 2016.

[6] J. Kopf, M. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graph*, vol. 26, no. 3, 2007.

[7] R. Gonzalez and R. Woods, *Digital Image Processing*, Prentice Hall, 2002.

[8] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 103–111, 2007.

[9] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 67, 2008.

[10] L. Rudin, S. Osher, and E. Ftemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[11] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing using l_0 gradient minimization," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 174–173, 2011.

[12] Q. Zhang, X. Shen, L. Xu, and J. Jia, "Rolling guidance filter," in *ECCV*, 2014, pp. 815–830.

[13] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama, "Digital photography with flash and non-flash image pairs," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 664–672, 2004.

[14] A. Adams, J. Baek, and M. A. Davis, "Fast high-dimensinal filtering using the permutohedral lattice," in *Eurographics*, 2010.

[15] J. Barron and B. Poole, "The fast bilateral solver," in *ECCV*, 2016.

[16] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397 – 1409, 2013.

[17] L. Xu, Q. Yan, Y. Xia, and J. Jia, "Structure extraction from texture via relative total variation," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 139–137, 2012.

[18] B. Ham, M. Cho, and J. Ponce, "Robust guided image filtering using nonconvex potentials," *IEEE Trans. PAMI*, 2017.

[19] X. Guo, Y. Li, and J. Ma, "Mutually guided image filtering," in *ACM MM*, 2017.