PAPER

Fast Edge Preserving 2D Smoothing Filter Using Indicator Function

Ryo ABIKO[†a)], ***Nonmember and*** **Masaaki IKEHARA**[†b)], ***Fellow***

SUMMARY Edge-preserving smoothing filter smoothes the textures while preserving the information of sharp edges. In image processing, this kind of filter is used as a fundamental process of many applications. In this paper, we propose a new approach for edge-preserving smoothing filter. Our method uses 2D local filter to smooth images and we apply indicator function to restrict the range of filtered pixels for edge-preserving. To define the indicator function, we recalculate the distance between each pixel by using edge information. The nearby pixels in the new domain are used for smoothing. Since our method constrains the pixels used for filtering, its running time is quite fast. We demonstrate the usefulness of our new edge-preserving smoothing method for some applications.

key words: *edge-preserving smoothing, indicator function, denoising, contents matching*

1. Introduction

Due to the spread of smartphones and SNS, image processing became more familiar to us and image filtering is used in many applications in image processing and computer graphics. In particular, edge-preserving smoothing filter is used in a fundamental process of many kinds of applications in image processing, including clip-art JPEG artifact removal [1], [2], detail manipulation [3]–[5], guided denoising [6], [7], colorization [5], [8], [9], guided upsampling [9], [10], tone mapping [3], [4], [11], super resolution [12], depth-of-field effect [5], haze removal [13], and stylization [5].

Edge-preserving smoothing filter smoothes the image but when a sharp edge appears, it stops smoothing. It means that the texture region is smoothed but the sharp edge will be preserved. This operation is visualized in Fig. 1. In the field of image processing, to apply a different process to texture region and edge region is important to obtain a high quality image. Since the normal smoothing filter such as Gaussian filter or moving average filter smoothes both texture region and edge region, it is difficult to decompose image into texture region and edge region. On the other hand, since edge-preserving smoothing filter smooths only texture region, it is possible to separate image into texture region and global edge region by taking the difference between the input image and smoothed image. It is a reason why edge-preserving smoothing filter is used in many applications.

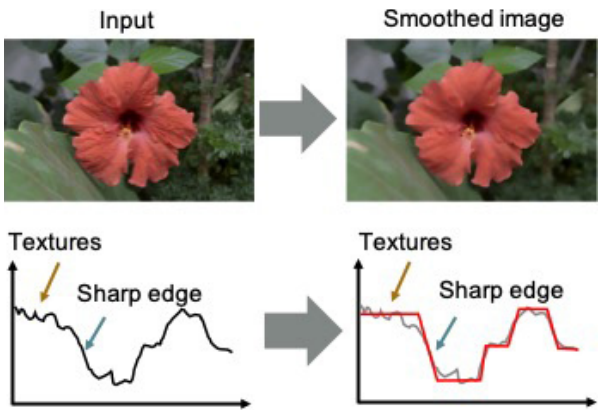


Fig. 1 The example of edge-preserving smoothing filter.

The most popular and classical edge-preserving smoothing filters are anisotropic diffusion [14] and bilateral filter [15]. Anisotropic diffusion applies heat conduction equation on the image and many iterative processes are required. On the other hand, bilateral filter computes L_2 norm in both spacial and range domain. Therefore, even if there are thin edges in image like in Fig. 3, bilateral filter might refer the pixels over the valley. The cross reference over sharp edges might deteriorate the performance of smoothing. To improve the quality of edge-preserving smoothing, various methods have been proposed: weighted least squares (WLS) [3], edge-avoiding wavelets (EAW) [16], domain transform (DT) [5], guided filter (GF) [6], fast global image smoothing (FGS) [9], tree filtering (TF) [17], L_0 gradient minimization (L_0GM) [11] and L_0 gradient projection (L_0GP) [2]. The local filtering method can process the image in short time but the edge-preserving effect is not enough. On the other hand, non-local method like L_0 gradient projection [2] can smooth image sufficiently but the running time is quite long.

To overcome the disadvantage of local filter, we propose a new method for performing edge-preserving smoothing to images in local filter. Our approach defines the indicator function which indicates the pixels used for smoothing. It is defined in 2D and we use edge information to gather the pixels which belong to the same region. Because the indicator function restricts the region of smoothing, our filter will not smooth across the sharp edges. That feature gives us flexibility of which type to use for smoothing such as Simple Moving Average (SMA), Gaussian smoothing filter, etc.

Manuscript received December 21, 2018.
Manuscript revised May 27, 2019.
Manuscript publicized July 22, 2019.

[†]The authors are with the Department of Electronics and Electrical Engineering, Keio University, Yokohama-shi, 223–8522, Japan.
a) E-mail: abiko@tkhm.elec.keio.ac.jp
b) E-mail: ikehara@tkhm.elec.keio.ac.jp

DOI: 10.1587/transinf.2018EDP7437

Our approach has two remarkable features. First, since our method uses 2D filter, the artifact which appears when the 1D filter [5] is applied does not appear. In addition, the output image is rotationally invariant and this will be an advantage when our edge-preserving smoothing method is used for content matching in a database. Second, since the indicator function restricts the spatial range of pixels in the filtering, the computational cost is low. Our edge-preserving smoothing filter can smooth the image rapidly due to the constraint of the range and the simplicity of our algorithm. This result is remarked in Sect. 4. We demonstrate the applications of our edge-preserving smoothing filter in Sect. 5, including clip-art JPEG artifact removal and detail manipulation.

2. Supporting Methods

2.1 Bilateral Filter (BF)

A bilateral filter [15] is a most popular edge-preserving smoothing filter. It works by weight averaging the value of neighbor pixels in space and range dimension. If the spatial or range distance is long, the weighting factor will be small. In contrast, if the similarity of the pixels is high, the weighting factor will be large. Therefore, the bilateral filtering weight w_{xypq} is given by

$$w_{xypq} = \frac{1}{K_{xy}} \exp\Big(-\frac{p^2+q^2}{\sigma_s^2}\Big) \exp\Big(-\frac{(I_{xy}-I_{x+p,y+q})^2}{\sigma_r^2}\Big) \quad (1)$$

$$K_{xy} = \sum_{p,q\in\Omega} \exp\Big(-\frac{p^2+q^2}{\sigma_s^2}\Big) \exp\Big(-\frac{(I_{xy}-I_{x+p,y+q})^2}{\sigma_r^2}\Big) \quad (2)$$

where (x, y) are the coordinates of the target pixel, (p, q) are the offsets in the kernel, I_{xy} is the pixel value and K_{xy} is a normalization factor which makes the sum of the filter to 1. σ_s and σ_r are parameters to decide the sensitivity in the spacial and range dimension. The filtered image value J_{xy} is computed by

$$J_{xy} = \sum_{p,q\in\Omega} w_{xypq} I_{x+p,y+q}. \quad (3)$$

Since this method have to compute all weighting factor in the kernel, the computational cost is high. To speed up the process, many approaches have been proposed [18]–[20] but in most cases they need quantization, downsampling or approximation. To overcome these deterioration factor, several methods were also proposed [21], [22].

2.2 Domain Transform (DT)

Domain Transform (DT) [5] is a fast method for the edge-preserving image processing. To speed up the process, it applies 1D domain transform to the image in advance and process the image by 1D filter. The smoothing amount in spatial and range dimension are determined by the parameter σ_s and σ_r, and the 1D transformed domain is given by

$$t_x = \sum_{i=1}^{x} |1 + \frac{\sigma_s}{\sigma_r}(I_i - I_{i-1})|. \quad (4)$$

To apply filter in the transformed domain, three kinds of methods were proposed: Normalized Convolution, Interpolated Convolution and Recursive Filtering. In particular, Normalized Convolution smoothes image strongly since it simply averages the values of the nearby pixels in the transformed domain. The equation of Normalized Convolution is:

$$J_x = (1/K_x) \sum_{p\in D(\Omega)} I_{x+p} H(t_x, t_{x+p})$$

$$H(t_x, t_{x+p}) = \begin{cases} 1 & \text{if } |t_x - t_{x+p}| \le \sqrt{3}\sigma_s \\ 0 & \text{otherwise} \end{cases}. \quad (5)$$

Since this method uses 1D-domain transform, it can only apply 1D filter to the images. Therefore, it applies 1D filter to the input image (2D signal) for several times from the different direction but it may cause artifacts along the direction which the 1D filter is applied finally. Since the image is processed as a 1D signal, the output image is not rotationally invariant.

2.3 L_0 Gradient Projection (L_0GP)

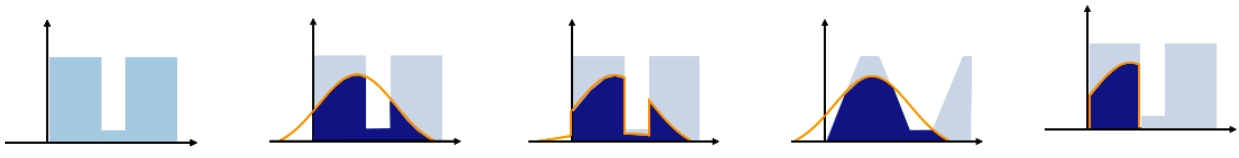
L_0 Gradient Projection (L_0GP) [2] minimizes the quadratic data-fidelity term subject to L_0 gradient term is less than user-given parameter α. L_0 gradient term shows the number of non-zero gradient of the output image and the quadratic data-fidelity term shows the L_2 norm of the input image and the output image. The amount of smoothing is controlled by α which determine the weight between L_0 gradient term and quadratic data-fidelity term. To perform the smoothing, a constrained optimization problem has to be solved and the formulation is shown as:

$$\text{Find } \boldsymbol{u}^\star \in \underset{\boldsymbol{u}\in R^{3N}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{u} - \bar{\boldsymbol{u}}\|^2 \text{ subject to } \mathrm{Grad}_{L_0}(\boldsymbol{u}) \le \alpha. \quad (6)$$

Since this method minimize the L_0 gradient by solving the constrained optimization problem, the image is strongly smoothed but the computational time is quite long.

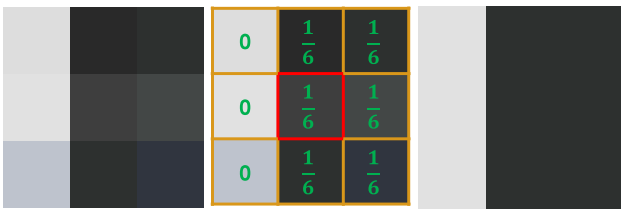
3. Proposed Method

We define a new 2D filter composed of smoothing term and indicator term. Since the indicator function constrains the range of smoothing, it improves the edge-preserving effect and speeds up the running time. The example of our filter weights are shown in Fig. 2. Before computing the indicator term, we compute the reconstructed distance of pixels by integrating the edge information. The indicator function is defined by judging if the reconstructed distance of the pixels is smaller than the user-given parameter σ.



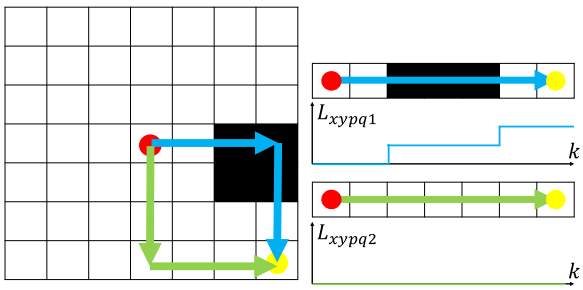
(a) Input signal. (b) Gaussian filter. (c) Bilateral filter [15]. (d) DT [5] + Gaussian filter. (e) Proposed + Gaussian filter.

Fig. 3 Visualization of the filter weights and the convolution when a sharp edge exists in the image.



(a) Input image. (b) Filter weights. (c) Smoothed image.

Fig. 2 The example of our filtering weights.



(a) The edge (colored in black) is in the filter kernel and L_{xypq1} and L_{xypq2} are visualized. (b) Edge integration.

Fig. 4 The visualized image of computing reconstructed domain.

3.1 Indicator Function

When the edge-preserving smoothing is performed, it is important to smooth the pixels which belong to the same region. However, classic smoothing filter (e.g. moving average, Gaussian smoothing filter or bilateral filter [15]) might smooth across different regions because they only refer to the space or intensity information. This characteristic deteriorates the performance of smoothing. Therefore, we define H_{xypq} as an indicator function. It indicates the pixels which are in the same region. From the above, our filtering weight w_{xypq} and indicator term H_{xypq} are expressed as:

$$w_{xypq} = \frac{1}{K_{xy}} S_{xypq} H_{xypq} \tag{7}$$

$$H_{xypq} = \begin{cases} 1 & \text{if } r_{xypq} \leq \sigma \\ 0 & \text{otherwise} \end{cases}, \tag{8}$$

where S_{xypq} is a smoothing term and r_{xypq} is a distance of two pixels in the reconstructed domain, which is explained in Sect. 3.2. K_{xy} is a normalization factor which makes the sum of the filter to 1. σ is a user-given parameter which determines the size of region and it also determines the amount of smoothing. When we use Simple Moving Average (SMA) for the smoothing term, the maximum smoothing effect is obtained. In that case, our filtering weight can be expressed as

$$w^{\mathrm{SMA}}_{xypq} = \frac{1}{K_{xy_{\mathrm{SMA}}}} H_{xypq}. \tag{9}$$

Since the computational cost is low, we mainly use (9) for the experiments. Our final smoothed image J^{EPFIF}_{xyc} is obtained by applying filter to RGB color space as following equation.

$$J^{\mathrm{EPFIF}}_{xyc} = \sum_{p,q\in\Omega} w_{xypq} I_{x+p,y+q,c}. \tag{10}$$

3.2 Reconstructed Domain

Our indicator function judges whether pixels are in the same region or not by the information of edges. When a sharp edge exists between two pixels, the pixels should be classified into different regions. Thus, we recalculate the distance of each pixels by integrating the intensity of edge information. We call this new domain as Reconstructed domain. In reconstructed domain, the distance of two pixels should be larger if more edges exist between them. The similar approach was proposed in DT [5] but this method was only available in 1D signal. Since there are no solution for 2D signals to define the accurate domain transform [5], we define the reconstructed domain by integrating the information of edges in a limited situation.

Considering the case of 2D signal, there are many routes for integrating edge information and each of them may have different values. In order to reduce the computational cost, only two routes are computed in our method. The example of two routes L_{xypq1} and L_{xypq2} are shown in Fig. 4 (a). When (p, q) are both positive, L_{xypq1} and L_{xypq2} are given by following expressions:

$$\begin{aligned} \boldsymbol{N}_{pq} &= \{N_0, N_1, \cdots, N_p, N_{p+1}, \cdots, N_k\} \\ &= \{I_{x,y}, I_{x+1,y}, \cdots, I_{x+p,y}, I_{x+p,y+1}, \cdots, I_{x+p,y+q}\} \end{aligned}$$

$$L_{xypq1} = \sum_{c} \sum_{l=1}^{k} |N_l - N_{l-1}|$$

$$\boldsymbol{M}_{pq} = \{M_0, M_1, \cdots, M_q, M_{q+1}, \cdots, M_k\}$$

(a) Input image (b) $\sigma = 0.25$ (c) $\sigma = 0.50$ (d) $\sigma = 0.75$ (e) $\sigma = 1.0$

Fig. 5 The comparison of the results when various σ is selected. Equation (9) is used for the process and the filter size is 9×9. Three iterations are performed.

$$= \{I_{x,y}, I_{x,y+1}, \cdots, I_{x,y+q}, I_{x+1,y+q}, \cdots, I_{x+p,y+q}\}$$

$$L_{xypq2} = \sum_{c} \sum_{l=1}^{k} |M_l - M_{l-1}|. \tag{11}$$

$\boldsymbol{N}$ and $\boldsymbol{M}$ are the sorted pixels on the each route and $k = |p| + |q|$ is the number of pixels in each route. When (p, q) are not positive, the sign of (11) must be appropriately inverted. The information on color channel is integrated when we calculate L_{xypq1} and L_{xypq2}. This process is visualized in Fig. 4 (b). DT use the information of space domain in order to define the accurate domain transform but in (11), the space information is not integrated since it is not important for our method.

To calculate (11) in every kernel will cost high computational resource. Thus, our method calculates 1D edge integrated values in advance. The integration in (11) can be calculated easily by using this value. The 1D edge integrated values L^X_{xy} and L^Y_{xy} are expressed as:

$$\begin{aligned} L^X_{xy} &= \sum_{c} \sum_{i=1}^{x} |I_{iy} - I_{i-1,y}| \\ L^Y_{xy} &= \sum_{c} \sum_{i=1}^{y} |I_{xi} - I_{x,i-1}|. \end{aligned} \tag{12}$$

Applying (12) to (11), the two routes are given by:

$$\begin{aligned} L_{xypq1} &= |L^X_{x+p,y} - L^X_{xy}| + |L^Y_{x+p,y+q} - L^Y_{x+p,y}| \\ L_{xypq2} &= |L^Y_{x,y+q} - L^Y_{xy}| + |L^X_{x+p,y+q} - L^X_{x,y+q}|. \end{aligned} \tag{13}$$

Because classifying as many pixels as possible in the same region is important for smoothing, we define the smaller value of L_{xypq1} and L_{xypq2} as the distance of two pixels in the reconstructed domain. Therefore, the distance r_{xypq} of the two pixels in the reconstructed domain is defined by (14).

$$r_{xypq} = \min(L_{xypq1}, L_{xypq2}) \tag{14}$$

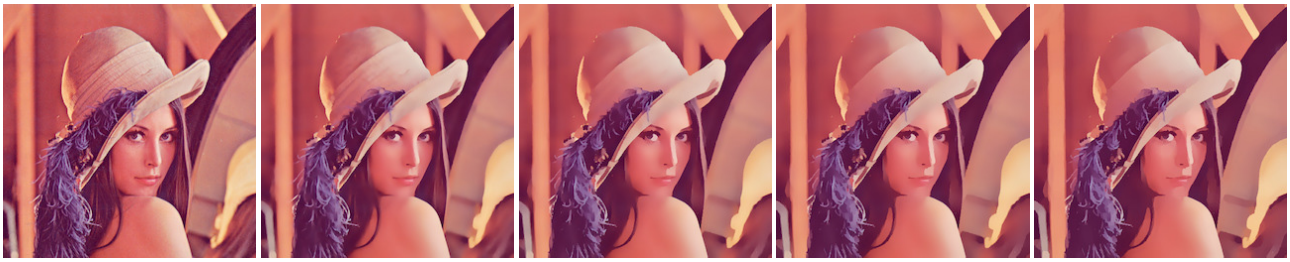
We revealed experimentally that using two routes for calculating distance in reconstructed domain shows a good trade-off between edge-preserving quality and the computational cost.

Figure 3 shows the 1D signal and the weighting factor of some smoothing methods. The methods which can be classified as a local filter are shown in the figure. We can see that conventional methods smooth over the sharp edge but our method does not refer to the information over the edge. It is because our method constrains the region of smoothing by the indicator term and it leads to efficient edge-preserving smoothing.

3.3 Parameters Analysis

Since we define the smoothing term and the indicator function separately, our filter can smooth images flexibly. When σ is large enough, the indicator function always take the value of 1. Thus, our filter acts like conventional smoothing methods: simple moving average filter (SMA), Gaussian filter, and the bilateral filter. In contrast, when σ is set to an appropriate value, the filter acts as an edge-preserving smoothing filter. In this case, the method chosen for the smoothing term determines the amount of smoothing. We mainly use (9) as our filter since it can obtain the best smoothing effect. The difference occurred by the choice of σ is shown in Fig. 5. When σ is set to a small value, the smoothing effect is limited and some texture remains. In contrast, when σ is set to a large value, our filter smoothes over edges. The suitable parameter of σ is depended on the applications.

Since indicator function takes the sum of the pixel differences, the further pixels tend to be classified to a different



(a) Input (b) 1 iteration. (c) 2 iterations. (d) 3 iterations. (e) 15 iterations.

Fig. 7 The comparison between the number of iterations. Equation (9) is used for the process, $\sigma = 0.45$ and the filter size is 9×9. The result of 3 iterations is quite similar to 15 iterations, during the results of 1 and 2 iterations do not have enough smoothing effect.

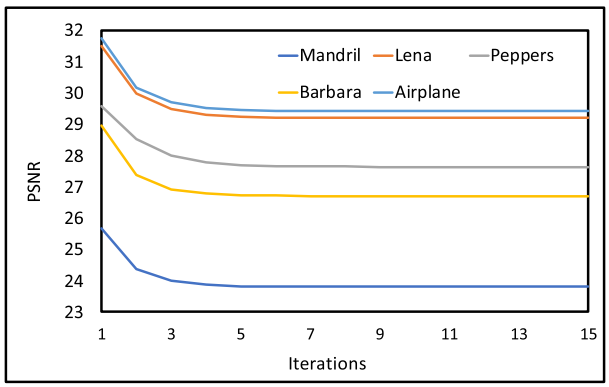


Fig. 6 The PSNR between the original image and the smoothed image in each iteration. $\sigma = 0.50$, the filter size is 9×9.

region. In order to overcome this problem, we apply our filter to the image for several times. When the number of iteration increases, more smoothed image is outputted but when we use the same value of σ during the iterations, too much smoothing effect is obtained. Experimentally we decide to reduce the value of σ by half through iterations.

$$\sigma_{it} = (0.5^{(t-1)})\sigma \tag{15}$$

where σ_{it} is the parameter used in t-th iteration. The result image of t-th iteration is used for the input of $(t+1)$-th iteration. In most cases, when the number of iterations gets larger than 3, the change of the PSNR [23] between the non-smoothed image and the smoothed image will be small enough (Fig. 6). Therefore, we decide to perform 3 iterations for our method. The visual results with different numbers of iterations are shown in Fig. 7.

4. Experimental Results

In this section, we show the smoothed image processed by our method and the comparison methods. The comparison methods are Domain Transform (DT) [5], L_0 Gradient Projection (L_0GP) [2] and bilateral filter (BF) [15]. The edge-preserving smoothing result to 5 images are shown in Fig. 8.

Analyzing the results of DT and L_0GP in Fig. 8, some artifacts are observed in vertical and horizontal direction when preserving diagonal edges. In contrast, the artifact does not appear in our method. It is because our method applies 2D indicator function and it has flexibility in preserving complex edges. In addition, we can see that DT and BF cannot preserve thin edges but our method can preserve them. The usefulness of our indicator function is shown by comparing the results in Fig. 8 (d) and Fig. 8 (f). Figure 8 (d) is the output of bilateral filter and Fig. 8 (f) is the output of proposed method which uses bilateral filter as the smoothing kernel. Bilateral filter smoothes over edges as shown in Fig. 8 (d) but Fig. 8 (f) shows the effective smoothing result which does not smoothes over edges and preserves thin edges.

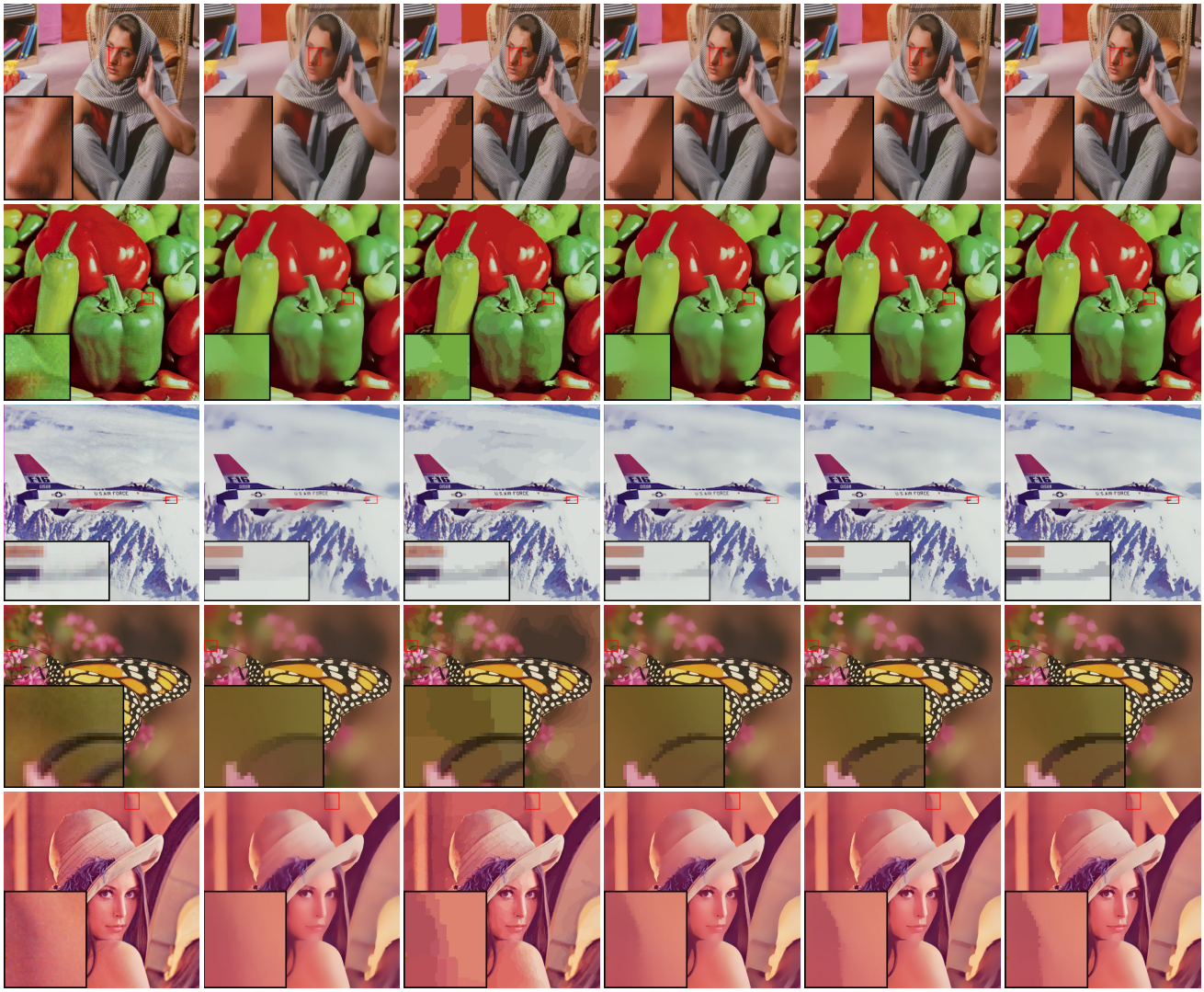
Our method can process 1M pixel RGB image in 0.25 seconds on quad core CPU@3.7GHz with our C++ implementation. Equation (9) is used in the computation, filter size is set to 9 and the number of iterations are 3. When we do not use Eq. (12) and Eq. (13) for the speeding up, the running time will be 3.2 times slower. Since our method is classified to local filter, parallel coding is effective for decreasing the execution time. The comparison of the execution time is shown in Table 1.

5. Applications

Since edge-preserving smoothing filter smooths only texture region, it is mainly used to separate the texture from the image. This feature is used in many applications. For example, detail manipulation is performed by adding enhanced texture information to the smoothed image. Guided denoising, colorization, guided upsampling, tone mapping, depth-of-field effect, haze removal, stylization and clip-art JPEG artifact removal are also available as applications.

The example of clip-art JPEG artifact removal is shown in Fig. 9. Since edge-preserving smoothing filter preserves sharp edges, it can be used for the preprocess method of image segmentation.

The comparison of segmentation result between preprocessed image and normal image is shown in Fig. 10. SLIC [26] is used for the segmentation. We can see that the segmented result after smoothed by our method can track the edge more effectively. The result of edge/boundary detection evaluation on 100 test images in BSDS300 is shown in Fig. 11. We can see that the F-measure got better when



(a) Input image (b) DT (c) L_0GP (d) BF (e) Proposed method. Smoothed by SMA. (f) Proposed method. Smoothed by BF.

Fig. 8 Image smoothing results comparison with DT [5] ($\sigma_s = 40$, $\sigma_r = 0.4$), L_0GP [2] ($\alpha = 0.25N$) and BF [15] ($\sigma_s = 20$, $\sigma_r = 0.07$). Figure 8 (e) and Fig. 8 (f) are the result of proposed method. σ is set to 0.35 in both Fig. 8 (e) and Fig. 8 (f). We use SMA as the smoothing term (defined in Eq. (9)) in Fig. 8 (e) while BF ($\sigma_r = 0.07$ $\sigma_s = 20$) in Fig. 8 (f). The filter size of Proposed method are set to 9×9. 3 iterations are performed in all methods excepting L_0GP. The same parameters are used to process each image.

Table 1 The comparison of the computational time for 512×512 RGB image. The comparison methods are Bilateral Filter ($\sigma_s = 20$, $\sigma_r = 0.07$) [15], L_0 Gradient Minimization ($\lambda = 0.0015$) [11], L_0 Gradient Projection ($\alpha = 0.08N$, $\gamma = 3$, $\eta = 0.95$) [2], Domain Transform ($\sigma_s = 40$, $\sigma_r = 0.4$) [5], Fast Global Image Smoothing ($\lambda = 50$, $\sigma = 0.1$) [9] and our proposed method (filter size: 9×9, $\sigma = 0.5$).

	BF	L_0GM	L_0GP	DT	FGS	Ours
MATLAB	1.52 s	1.46 s	102 s	2.89 s	-	1.70 s
C++	-	-	-	0.04 s	0.15 s	0.07 s

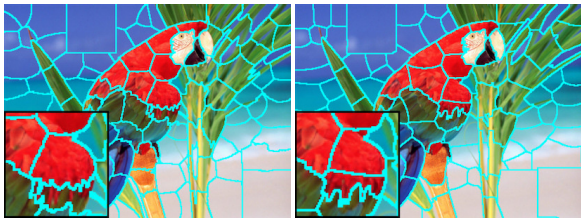
we applied our method before applying SLIC to the images.

Since we define the indicator function in 2D space, our method is rotationally invariant. The L_0 error rate are shown in Table 2. The L_0 error rate of image J^1 and J^2 can be expressed as:

(a) JPEG compressed image. (QF = 30) PSNR = 30.93 (b) Output image filtered with our method ($\sigma = 0.40$). PSNR = 33.44

Fig. 9 The result of denoising applied on JPEG image.

$$E = \frac{1}{N} \sum_{x,y} \delta(J^1_{xy} - J^2_{xy})$$



(a) Segmented image without preprocess.

(b) Segmented image preprocessed by our method.

Fig. 10 The comparison of image segmentation result.

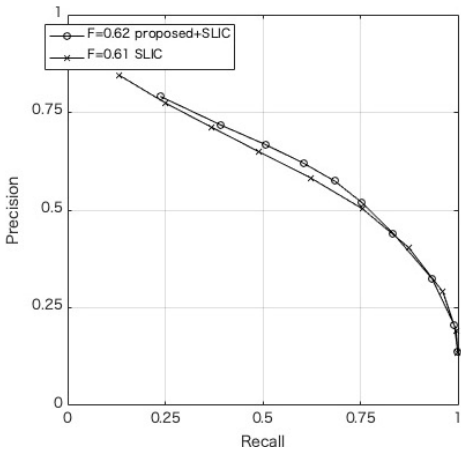


Fig. 11 The result of edge/boundary detection evaluation. The parameter is set to $\sigma = 0.45$ and the filter size is 9×9. The higher is the better.

Table 2 The comparison of L_0 error rate between the output images processed by 90 degrees rotated input and normal input. The result is expressed in percent. The comparison methods are L_0 Gradient Minimization ($\lambda = 0.0015$) [11], L_0 Gradient Projection ($\alpha = 0.08N$, $\gamma = 3$, $\eta = 0.95$) [2], Domain Transform ($\sigma_s = 40$, $\sigma_r = 0.4$) [5], Fast Global Image Smoothing ($\lambda = 50$, $\sigma = 0.1$) [9] and our proposed method (filter size: 9×9, $\sigma = 0.5$).

Dataset name	L_0GM	L_0GP	DT	FGS	Ours
BSDS300 [24]	0.29	0.92	0.25	0.25	**0.00**
COCO [25]	0.31	0.42	0.27	0.26	**0.00**
Average	0.30	0.67	0.26	0.26	**0.00**

$$\delta(a) = \begin{cases} 1 & a \neq 0 \\ 0 & a = 0 \end{cases}. \tag{16}$$

where N is the pixel number and x, y are the coordinates. When two images are completely same, the error rate will be 0. The result in Table 2 shows that our method is rotationally invariant. This feature is available in contents matching in a database system.

6. Conclusion

In this paper, we propose a new approach for the edge-preserving smoothing. We define the 2D indicator function to restrict the pixels which are used for smoothing. This function is defined by comparing the value of the integration of edge information and it is easily computed by using 1D integration. Because the 1D edge integration can be computed beforehand, the computational cost is low and also parallel implementation is available. Due to defining the indicator function in 2D space, the output of our filter is rotationally invariant. This feature is desirable in case of using our filter in content matching.

References

[1] L. Zhao, H. Bai, A. Wang, and Y. Zhao, "Iterative range-domain weighted filter for structural preserving image smoothing and de-noising," Multimedia Tools and Applications, vol.78, no.1, pp.47–74, 2019.
[2] S. Ono, "L_0 gradient projection," IEEE Trans. Image Process., vol.26, no.4, pp.1554–1564, April 2017.
[3] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," ACM SIGGRAPH 2008 Papers, SIGGRAPH '08, New York, NY, USA, pp.67:1–67:10, ACM, 2008.
[4] K. Subr, C. Soler, and F. Durand, "Edge-preserving multiscale image decomposition based on local extrema," ACM SIGGRAPH Asia 2009 Papers, SIGGRAPH Asia '09, New York, NY, USA, pp.147:1–147:9, ACM, 2009.
[5] E.S.L. Gastal and M.M. Oliveira, "Domain transform for edge-aware image and video processing," ACM Transactions on Graphics (ToG), vol.30, no.4, ACM, 2011.
[6] K. He, J. Sun, and X. Tang, "Guided image filtering," IEEE Trans. Pattern Anal. Mach. Intell., vol.35, no.6, pp.1397–1409, June 2013.
[7] G. Deng, "Guided wavelet shrinkage for edge-aware smoothing," IEEE Trans. Image Process., vol.26, no.2, pp.900–914, Feb. 2017.
[8] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," ACM Transactions on Graphics (ToG), vol.23, no.3, pp.689–694, ACM, 2004.
[9] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M.N. Do, "Fast global image smoothing based on weighted least squares," IEEE Trans. Image Process., vol.23, no.12, pp.5638–5653, Dec. 2014.
[10] K.-H. Lo, Y.-C.F. Wang, and K.-L. Hua, "Edge-preserving depth map upsampling by joint trilateral filter," IEEE Trans. Cybern., vol.48, no.1, pp.371–384, Jan. 2018.
[11] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," Proceedings of the 2011 SIGGRAPH Asia Conference, SA '11, New York, NY, USA, pp.174:1–174:12, ACM, 2011.
[12] S. Huang, J. Sun, Y. Yang, Y. Fang, P. Lin, and Y. Que, "Robust single-image super-resolution based on adaptive edge-preserving smoothing regularization," IEEE Trans. Image Process., vol.27, no.6, pp.2650–2663, June 2018.
[13] D. Park, D.K. Han, and H. Ko, "Single image haze removal with wls-based edge-preserving smoothing filter," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp.2469–2473, May 2013.
[14] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," IEEE Trans. Pattern Anal. Mach. Intell., vol.12, no.7, pp.629–639, 1990.
[15] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," Sixth International Conference on Computer Vision (IEEE Cat. no.98CH36271), pp.839–846, Jan. 1998.
[16] R. Fattal, "Edge-avoiding wavelets and their applications," ACM Trans. Graph., vol.28, no.3, pp.1–10, 2009.
[17] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang, "Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree," IEEE Trans. Image Process., vol.23, no.2, pp.555–569, 2014.
[18] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," International Journal of Computer Vision, vol.81, no.1, pp.24–52, Jan. 2009.
[19] K.N. Chaudhury, D. Sage, and M. Unser, "Fast $O(1)$ bilateral filtering using trigonometric range kernels," IEEE Trans. Image Process.,

vol.20, no.12, pp.3376–3382, Dec. 2011.
[20] K.N. Chaudhury, "Fast and accurate bilateral filtering using gauss-polynomial decomposition," 2015 IEEE International Conference on Image Processing (ICIP), pp.2005–2009, Sept. 2015.
[21] K. Sugimoto and S.-I. Kamata, "Compressive bilateral filtering," IEEE Trans. Image Process., vol.24, no.11, pp.3357–3369, 2015.
[22] Y. Maeda, N. Fukushima, and H. Matsuo, "Effective implementation of edge-preserving filtering on cpu microarchitectures," Applied Sciences, vol.8, no.10, 2018.
[23] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," 2010 20th International Conference on Pattern Recognition, pp.2366–2369, Aug. 2010.
[24] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," Proc. 8th Int'l Conf. Computer Vision, pp.416–423, July 2001.
[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick, "Microsoft coco: Common objects in context," Computer Vision – ECCV 2014, ed. D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Cham, pp.740–755, Springer International Publishing, 2014.
[26] 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. Pattern Anal. Mach. Intell., vol.34, no.11, pp.2274–2282, 2012.

Ryo Abiko received the B.E. degrees in electrical engineering from Keio University, Yokohama, Japan, in 2018. He is currently an M.E. student at Keio University, Yokohama, Japan, under the supervision of Prof. Masaaki Ikehara. His research interests are in the field of image filtering, denoising and image processing using deep learning.

Masaaki Ikehara received the B.E., M.E. and Dr.Eng. degrees in electrical engineering from Keio University, Yokohama, Japan, in 1984, 1986, and 1989, respectively. He was Appointed Lecturer at Nagasaki University, Nagasaki, Japan, from 1989 to 1992. In 1992, he joined the Faculty of Engineering, Keio University. From 1996 to 1998, he was a visiting researcher at the University of Wisconsin, Madison, and Boston University, Boston, MA. He is currently a Full Professor with the Department of Electronics and Electrical Engineering, Keio University. His research interests are in the areas of multi-rate signal processing, wavelet image coding, and filter design problems.