FAST EDGE PRESERVING 2D SMOOTHING FILTER USING INDICATOR FUNCTION

Ryo Abiko, Masaaki Ikehara

EEE Dept., Keio Univ., Yokohama, Kanagawa 223-8522, Japan
{abiko,ikehara}@tkhm.elec.keio.ac.jp

ABSTRACT

Edge-preserving smoothing filter smoothes the textures while it preserves the information of sharp edges. In image processing, this 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 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 implementation is quite fast. We demonstrate the usefulness of our new edge-preserving smoothing method for some applications.

Index Terms — edge-preserving smoothing, indicator function, denoising, contents matching

1. INTRODUCTION

Image filtering is used in many applications in computer vision and computer graphics. In particular, edge-preserving smoothing filter is used in fundamental process of many kinds of applications in image processing, including clip-art JPEG artifact removal [1, 2], detail manipulation [3,4], guided denoising [5,6], colorization [7,8], guided upsampling [8, 9], tone mapping [3, 4, 10], depth-of-field effect [11], haze removal [12], and stylization [11]. Edge-preserving smoothing filter smoothes the image but when sharp edge appears, it stops smoothing. It means that the texture region is smoothed but the sharp edge will be preserved. The most popular edge-preserving smoothing filter is bilateral filter [13]. This method computes only 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 smoothing and edge-preserving, various methods have been proposed: weighted least squares [3], edge-avoiding wavelets [14], domain transform [11], guided filter [5], L_0 gradient minimization [10] and L_0 gradient projection [2]. The local filtering method can process the image rapidly but the smoothing effect is not enough. On the other hand, non-local method like L_0 gradient projection [2] can smooth image sufficiently but the implementation time is long.

We propose a new method for performing edge-preserving smoothing filter to images. 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 sharp edges. That feature gives us flexibility of which type to use for smoothing such as moving average, gaussian smoothing filter, etc.

Our approach has two remarkable features. First, since our method uses 2D filter, the artifact which appears when the 1D filter [11] is applied does not appear. In addition, the output image is rotationally invariant and this will be an advantage when the edge-preserving smoothing filter 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 range and the simplicity of our algorithm. This result is remarked in section 4.

We demonstrate the application of our edge-preserving smoothing filter in section 5, including clip-art JPEG artifact removal.

2. SUPPORTING METHODS

2.1. Bilateral Filter (BF)

A bilateral filter [13] 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

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

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 normalizing parameter. 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

$$J_{xy} = \sum_{p,q\in\Omega} w_{xypq} I_{x+p,y+q}. \tag{2}$$

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 [15–17] but in most cases they need quantization, downsampling or approximation.

2.2. Domain Transform (DT)

Domain Transform [11] is a fast method for the edge-preserving image processing. To speed up the process, it applies domain transform to the image in advance. The smoothing amount in spatial and range dimension are determined by parameter σ_s and σ_r, and the transformed domain is given by

$$t_x = \sum_{i=1}^{x} |1 + \frac{\sigma_s}{\sigma_r}\big(I_i - I_{i-1}\big)|. \tag{3}$$

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:

$$J_x = (1/K_x) \sum_{p \in D(\Omega)} I_{x+p} H(t_x, t_{x+p})$$
$$H(t_x, t_{x+p}) = \begin{cases} 1 & \text{if } |t_x - t_{x+p}| \leq \sqrt{3}\sigma_s \\ 0 & \text{otherwise} \end{cases}. \quad (4)$$

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. It may cause artifacts along the direction which the 1D filter is applied and also the output image is not rotationally invariant.

3. PROPOSED METHOD

We define a new 2D filter kernel including smoothing kernel and indicator function kernel. Since the indicator function constrains the range of smoothing, it improves the edge-preserving effect and speeds up the implementation time. Before computing the indicator function kernel, we compute the reconstructed distance of pixels by integrating edge information. The indicator function is defined by judging if the reconstructed distance of the pixels is smaller than parameter σ_i.

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 [13]) might smooth across different regions because it only refer to the space or intensity information. This characteristic deteriorates the performance of smoothing. Therefore, we define H_{xypq} as an indicator function and it determines which pixels belong to the same region in the image. From the above, our filtering weight w_{xypq} and H_{xypq} are expressed as:

$$w_{xypq} = \frac{1}{K_{xy}} S_{xypq} H_{xypq} \quad (5)$$
$$H_{xypq} = \begin{cases} 1 & \text{if } r_{xypq} \leq \sigma_i \\ 0 & \text{otherwise} \end{cases}, \quad (6)$$

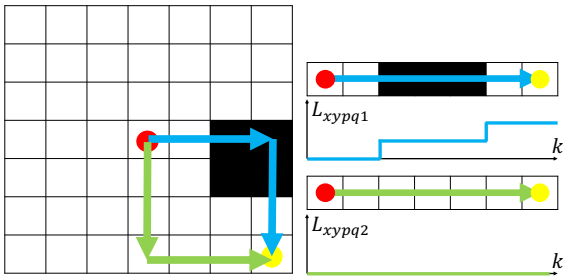
where S_{xypq} is a smoothing kernel and r_{xypq} is a distance of two pixels in the reconstructed domain, which is explained in Section 3.2. σ_i is a user-given parameter which determines the region of smoothing. When we use simple moving average for the smoothing kernel, the maximum smoothing effect is obtained. In that case, our filtering weight can be expressed as

$$w^{SMA}_{xypq} = \frac{1}{K_{xy_{\max}}} H_{xypq}. \quad (7)$$

Since the computational cost is low, we mainly use (7) for the experiments.

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 [11] but this method was only available in 1D signal. Since there are no solution for 2D signals to define the accurate domain transform [11], we define the reconstructed domain by integrating the information of edges in a limited situation.



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

(b) Edge integration.

Fig. 1. The visualized image of computing reconstructed domain.

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. 1(a). When (p, q) are both positive, are given by following expression:

$$\begin{aligned} \boldsymbol{N}_{pq} &= \{N_0, N_1, \cdots, N_p, N_{p+1}, \cdots, N_k\} \\ &= \{I_{x,y}, I_{x+1,y}, \cdots, I_{x+p,y}, I_{x+p,y+1}, \cdots, I_{x+p,y+q}\} \\ L_{xypq1} &= \sum_{l=1}^{k} |N_l - N_{l-1}| \\ \boldsymbol{M}_{pq} &= \{M_0, M_1, \cdots, M_q, M_{q+1}, \cdots, M_k\} \\ &= \{I_{x,y}, I_{x,y+1}, \cdots, I_{x,y+q}, I_{x+1,y+q}, \cdots, I_{x+p,y+q}\} \\ L_{xypq2} &= \sum_{l=1}^{k} |M_l - M_{l-1}|. \end{aligned} \quad (8)$$

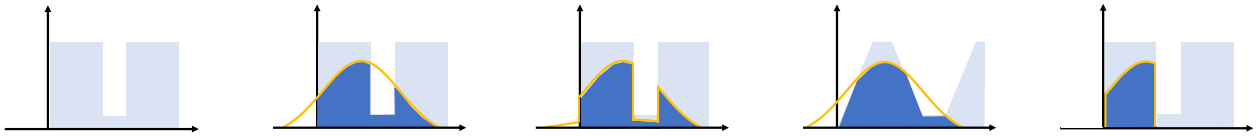
$\boldsymbol{N}$ and $\boldsymbol{M}$ are the sorted pixels on the each route and $k = |p| + |q|$ is the number of pixels in the each route. When (p, q) are not positive, the sign of (8) must be appropriately inverted. This process is visualized in Fig. 1(b). DT use the information of space domain in order to define the accurate domain transform but in (8), the space information is not integrated since it is not important for our method.

To calculate (8) in every kernel will cost high computational resource. Thus, our method calculates 1D edge integrated values in advance. The integration in (8) can be calculated easily by using this value. The 1D edge integrated values L_{xy1} and L_{xy2} are expressed as:

$$\begin{aligned} L_{xy1} &= \sum_{i=1}^{x} |I_{iy} - I_{i-1,y}| \\ L_{xy2} &= \sum_{i=1}^{y} |I_{xi} - I_{x,i-1}|. \end{aligned} \quad (9)$$

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

Fig. 2. The comparison between the number of iterations. Eq. (7) is used for the process, $\sigma_i = 0.45$ 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.



(a) Input signal. (b) Gaussian filter. (c) Bilateral filter [13]. (d) DT [11]+ gaussian filter. (e) Proposed + gaussian filter.

Fig. 3. Visualization of the filter weight (yellow lines) and the convolution area (blue regions) when a sharp edge exists in the image.

Applying (9) to (8), the two routes are given by:

$$\begin{aligned} L_{xypq1} &= |L_{x+p,y1} - L_{xy1}| + |L_{x+p,y+q,2} - L_{x+p,y2}| \\ L_{xypq2} &= |L_{x,y+q,2} - L_{xy2}| + |L_{x+p,y+q,1} - L_{x,y+q,1}|. \end{aligned} \tag{10}$$

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 as

$$r_{xypq} = \min(L_{xypq1}, L_{xypq2}). \tag{11}$$

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

Fig. 3 shows the 1D signal and the weighting factor of the conventional and proposed filter kernel. We can see that our proposed filter constrain the region of smoothing and it leads to the efficient edge-preserving smoothing.

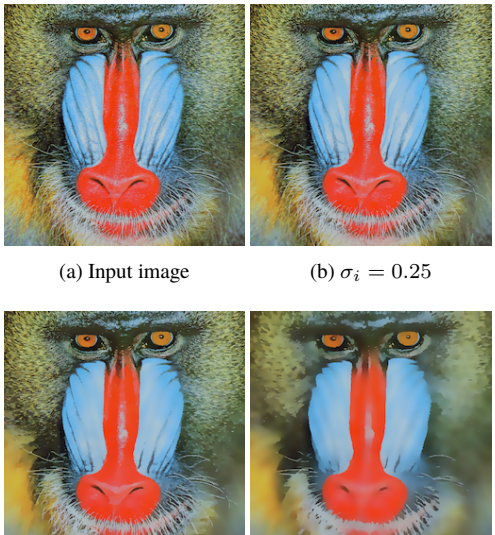
3.3. Parameters analysis

Since we define the smoothing term and the indicator function separately, our filter can smooth images flexibly. When σ_i is large enough, the indicator function always take the value of 1. Thus, our filter acts like conventional smoothing methods: simple moving average filter, gaussian filter, and the bilateral filter. In contrast, when σ_i is set to an appropriate value, the filter acts as an edge-preserving filter. In this case, the method chosen for the smoothing kernel determines the amount of smoothing. We mainly use (7) as our filter since it can obtain the best smoothing effect. The difference occurred by the choice of σ_i is shown in Fig. 4. When σ_i is set to a small value, the smoothing effect is limited and some texture remains. In contrast, when σ_i is set to a large value, our filter smoothes image over edges. The suitable parameter of σ_i depends on the applications.

Since indicator function takes the sum of the pixel differences, the further pixels tend to be classified to a different 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_i during the iterations, too much smoothing effect is obtained. Experimentally we decide to reduce the value of σ_i by half through iterations.

$$\sigma_{it} = (0.5^{(t-1)})\sigma_i \tag{12}$$

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 PSNR [18] between the results of subsequent iterations get larger than 50.



(a) Input image (b) $\sigma_i = 0.25$

(c) $\sigma_i = 0.50$ (d) $\sigma_i = 1.0$

Fig. 4. The comparison of the results when various σ_i is selected. Equation (7) is used for the process and the filter size is 9×9. Three iterations are performed.

Table 1: 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 computational time for 512 × 512 RGB image is also shown in the table. The comparison methods are L_0 Gradient Minimization ($\lambda = 0.0015$) [10], L_0 Gradient Projection ($\alpha = 0.08N$,$\gamma = 3$,$\eta = 0.95$) [2], Domain Transform ($\sigma_s = 40, \sigma_r = 0.4$) [11], Fast Global Image Smoothing ($\lambda = 50, \sigma = 0.1$) [8] and our proposed method (filter size:9×9, $\sigma_i = 0.5$).

Dataset name		L_0GM	L_0GP	DT	FGS	Proposed
BSDS300 [19]		0.29	0.92	0.25	0.25	**0.00**
COCO [20]		0.31	0.42	0.27	0.26	**0.00**
Average		0.30	0.67	0.26	0.26	**0.00**
time	MATLAB	1.46s	102s	2.89s		1.70 s
	C++			0.04s	0.15s	0.07 s

Therefore, we decide to perform 3 iterations for our method. The results with different numbers of iterations are shown in Fig. 2.

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) [11], L_0 Gradient Projection (L_0GP) [2] and bilateral filter [13].

Analyzing the results of DT and L_0GP in Fig. 5, the thin gray edge is smoothed . In contrast, it is preserved in our result. It is because our 2D indicator function strictly restricts the region of smoothing. We can see the effect of our indicator function by comparing the results shown in Fig. 5(e) and Fig. 5(f). Fig. 5(e) is the output of bilateral filter and Fig. 5(f) is the output of proposed method which uses bilateral filter as the smoothing kernel. Bilateral filter smoothes over edges as shown in Fig. 5(e) but Fig. 5(f) shows the effective smoothing result which does not smoothes over edges.

Our method can process 1M pixel RGB image in 0.65 seconds on dual core CPU@2.2GHz with our C++ implementation. Eq. (7) is used the computation, filter size is set to 9 and the number of iterations are 3. Since our method is classified to local filter, parallel coding is effective for decreasing the execution time.

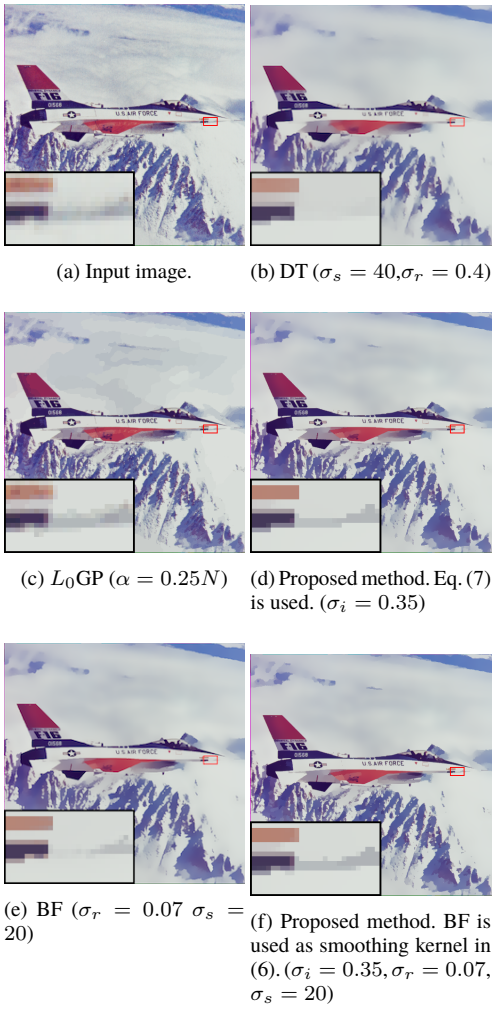
5. APPLICATIONS

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. 6.

Since we define the indicator function in 2D space, our method is rotationally invariant. The L_0 error rate between the output images processed by 90 degrees rotated input and normal input are shown in Table 1. The execution time is also shown in Table 1. When two images are completely same, the error rate will be 0. The result in Table 1 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.



(a) Input image. (b) DT ($\sigma_s = 40$,$\sigma_r = 0.4$)

(c) L_0GP ($\alpha = 0.25N$) (d) Proposed method. Eq. (7) is used. ($\sigma_i = 0.35$)

(e) BF ($\sigma_r = 0.07$ $\sigma_s = 20$) (f) Proposed method. BF is used as smoothing kernel in (6). ($\sigma_i = 0.35, \sigma_r = 0.07$, $\sigma_s = 20$)

Fig. 5. Image smoothing results comparison with DT [11], L_0GP [2] and BF [13]. The filter size of Proposed method are set to 9×9. 3 iterations are performed in all methods instead of L_0GP.

(a) JPEG compressed image. (b) Output image filtered with our method ($\sigma_i = 0.40$).

Fig. 6. The result of denoising applied on JPEG image.

7. REFERENCES

[1] Lijun Zhao, Huihui Bai, Anhong Wang, and Yao Zhao, "Iterative range-domain weighted filter for structural preserving image smoothing and de-noising," *Multimedia Tools and Applications*, pp. 1–28, 2017.

[2] S. Ono, "l_0 gradient projection," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1554–1564, April 2017.

[3] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," in *ACM SIGGRAPH 2008 Papers*, New York, NY, USA, 2008, SIGGRAPH '08, pp. 67:1–67:10, ACM.

[4] Kartic Subr, Cyril Soler, and Frédo Durand, "Edge-preserving multiscale image decomposition based on local extrema," in *ACM SIGGRAPH Asia 2009 Papers*, New York, NY, USA, 2009, SIGGRAPH Asia '09, pp. 147:1–147:9, ACM.

[5] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, June 2013.

[6] G. Deng, "Guided wavelet shrinkage for edge-aware smoothing," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 900–914, Feb 2017.

[7] Anat Levin, Dani Lischinski, and Yair Weiss, "Colorization using optimization," in *ACM Transactions on Graphics (ToG)*. ACM, 2004, vol. 23, pp. 689–694.

[8] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do, "Fast global image smoothing based on weighted least squares," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5638–5653, Dec 2014.

[9] K. H. Lo, Y. C. F. Wang, and K. L. Hua, "Edge-preserving depth map upsampling by joint trilateral filter," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 371–384, Jan 2018.

[10] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia, "Image smoothing via l_0 gradient minimization," in *Proceedings of the 2011 SIGGRAPH Asia Conference*, New York, NY, USA, 2011, SA '11, pp. 174:1–174:12, ACM.

[11] Eduardo SL Gastal and Manuel M Oliveira, "Domain transform for edge-aware image and video processing," in *ACM Transactions on Graphics (ToG)*. ACM, 2011, vol. 30, p. 69.

[12] D. Park, D. K. Han, and H. Ko, "Single image haze removal with wls-based edge-preserving smoothing filter," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 2469–2473.

[13] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan 1998, pp. 839–846.

[14] Raanan Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–10, 2009.

[15] Sylvain Paris and Frédo 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.

[16] K. N. Chaudhury, D. Sage, and M. Unser, "Fast $o(1)$ bilateral filtering using trigonometric range kernels," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3376–3382, Dec 2011.

[17] K. N. Chaudhury, "Fast and accurate bilateral filtering using gauss-polynomial decomposition," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sept 2015, pp. 2005–2009.

[18] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 2366–2369.

[19] 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," in *Proc. 8th Int'l Conf. Computer Vision*, July 2001, vol. 2, pp. 416–423.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Eds., Cham, 2014, pp. 740–755, Springer International Publishing.