200 FPS CONSTANT-TIME BILATERAL FILTER USING SVD AND TILING STRATEGY

Kenjiro Sugimoto[†*]*, Norishige Fukushima*[‡†]*, and Sei-ichiro Kamata*[†]

[†]Waseda University, Japan
[‡]Nagoya Institute of Technology, Japan

ABSTRACT

This paper presents a constant-time bilateral filter that supports arbitrary range kernel designed via singular value decomposition (SVD). Bilateral filter (BF) suffers from high computational complexity in real-time processing due to the time-variant kernel. Although various accelerations for BF have been proposed, most of them have not achieved both arbitrary range kernel and tight computational complexity simultaneously. The proposed method supports arbitrary range kernel but requires half computational complexity of most state-of-the-art methods. Moreover, we present two implementation techniques well matched to the SVD approach: range fitting and tiling strategy. Experiments show that, in the cases of major range kernels, the proposed method not only runs faster (200 FPS) but also achieves higher accuracy than the state-of-the-art methods.

***Index Terms*—** SVD, constant-time bilateral filtering, edge-preserving filtering, acceleration

1. INTRODUCTION

Bilateral filter (BF) [1, 2] is one of the major edge-preserving smoothing methods widely used in image processing. Various applications have utilized BF such as denoising [3], high dynamic range imaging [4], detail enhancement [5], deblurring [6, 7], stereo matching [8], haze removing [9], depth map refinement [10], optical flow estimation [11] and so on. BF smoothes an image using a composite kernel depending on pixel position (spatial kernel) and pixel intensity (range kernel). Because the kernel shape differs for each target pixel, the computational complexity is much higher than linear filters such as Gaussian filter (GF).

In order to overcome this difficulty, many accelerated algorithms have been proposed in the past. Most of them share the general framework that approximate BF by an appropriate combination of linear filters. The piece-wise linear approximation [4], an early approach for acceleration, decomposes BF into multiple of GFs implemented by FFT. Separable BF [12] makes the composite kernel separable to achieve the computational complexity $O(r)$ where r is filter window radius. Unfortunately, this approach causes low accuracy because the composite kernel is essentially non-separable. This approach was improved in accuracy in [13]. Another approach for acceleration is a combination of spatial downsampling [14, 15]. As a more algorithmic improvement, constant-time, or $O(1)$, BF is considered as state-of-the-art for this topic where constant-time means the per-pixel complexity does not depend on filter window radius r. The first method [16] utilized integral histogram [17]. Real-time $O(1)$ BF [18] further accelerates it using recursive GF [19, 20]. The raised cosine based approximation [21, 22, 23], compressive bilateral filtering [24, 25], Taylor decomposition based approximation [26, 27], and eigenvalue decomposition (EVD) based approach [28] well approximate BF than the piece-wise linear approximation.

Table 1: Characteristics of major accelerated BFs.

	Kernel type	Conv. sharing
Linear [18]	**Arbitrary**	N/A
Raised Cosine [21, 22, 23]	Gaussian	N/A
Polynomial [26, 27]	Differentiable	**Possible**
Compressive BF [24, 25]	Gaussian	**Possible**
EVD [28, 29]	**Arbitrary**	N/A
SVD (Proposed)	**Arbitrary**	**Possible**

Recently, a remarkable technique has been reported for reducing computational complexity. Deng [25] succeeded to reduce the number of GFs by half as compared with [24] by sharing the results of GFs for the numerator and denominator of BF. However, the range kernel is limited to Gaussian. Although the EVD method [28] supports arbitrary range kernel, the combination with convolution sharing has not been found yet. The polynomial approximations [26, 27] shares numerator/denominator convolutions but are limited to differentiable range kernel. Table 1 lists characteristics of each method.

This paper presents an accelerated $O(1)$ BF for the arbitrary range kernel that provides convolution sharing. We overcome the difficulty using singular value decomposition (SVD). Our contributions are as follows: 1) Our SVD method reduces the number of convolutions by half by sharing the results of GFs in the numerator and denominator of BF, 2) Our method supports arbitrary range kernels, and 3) Our method enhances both computational complexity and approximate ac-

[*]This work was supported by JSPS KAKENHI JP17H01764, 18K18076.
[†]This work was supported by JSPS KAKENHI JP17H01764, 18K19813.

To appear in *Proc. IEEE International Conference on Image Processing (ICIP), 2019.*

curacy by using image tiling strategy and range fitting.

2. $O(1)$ BILATERAL FILTERING

Let us discuss a D-dimensional R-tone grayscale image $f : \mathcal{S} \mapsto \mathcal{R}$ where $\mathcal{S} \subset \mathbb{Z}^D$ denotes the domain of pixel positions and $\mathcal{R} = \{0, \ldots, R-1\} \subset \mathbb{Z}$ denotes dynamic range (generally, $D = 2$ and $R = 256$). Using a pixel position $\boldsymbol{p} \in \mathcal{S}$ and its intensity $f_{\boldsymbol{p}} \in \mathcal{R}$, BF [1] is defined by

$$\hat{f}_{\boldsymbol{p}} = \frac{\sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q}) w_r(f_{\boldsymbol{p}}, f_{\boldsymbol{q}}) f_{\boldsymbol{q}}}{\sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q}) w_r(f_{\boldsymbol{p}}, f_{\boldsymbol{q}})}, \tag{1}$$

where $w_s : \mathcal{S}\times\mathcal{S} \mapsto \mathbb{R}$ is spatial kernel and $w_r : \mathcal{R}\times\mathcal{R} \mapsto \mathbb{R}$ is range kernel. The most common choice is Gaussian:

$$w_s(\boldsymbol{p},\boldsymbol{q}) = e^{-\frac{\|\boldsymbol{q}-\boldsymbol{p}\|_2^2}{2\sigma_s^2}}, \qquad w_r(a,b) = e^{-\frac{(b-a)^2}{2\sigma_r^2}} \tag{2}$$

where $\sigma_s \in \mathbb{R}_+$ is spatial scale and $\sigma_r \in \mathbb{R}_+$ is range scale.

In $O(1)$ BF [24, 25, 28, 29], the range kernel is generally approximated by separable form $w_r(a,b) \approx \sum_{k=0}^{K-1} \phi_k(a)\psi_k(b)$. By plugging it to (1),

$$\hat{f}_{\boldsymbol{p}} \approx \frac{\sum_{k=0}^{K-1} \phi_k(f_{\boldsymbol{p}}) \sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q})\{\psi_k(f_{\boldsymbol{q}}) f_{\boldsymbol{q}}\}}{\sum_{k=0}^{K-1} \phi_k(f_{\boldsymbol{p}}) \sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q})\{\psi_k(f_{\boldsymbol{q}})\}}. \tag{3}$$

In this equation, $\{\cdot\}$ can be regarded as intermediate images and $\sum_{\boldsymbol{q}\in\mathcal{S}}$ indicates their convolution. This separable approximation results in BF decomposed into a product sum of $2K$ convolutions. By implementing the convolutions as $O(1)$ filters including [30, 31, 32], BF (3) can run in $O(1)$ time per pixel. In this framework, our purpose is to achieve higher approximate accuracy using the smaller number of convolutions, or intermediate images. In existing methods, as $\phi_k(\cdot)$ and $\psi_k(\cdot)$ in (3), trigonometric functions [24, 25] for the Gaussian range kernel, and eigenvectors for arbitrary kernels [28, 29] have been used.

As a remarkable approach, [25] succeeded to reduce the number of convolutions by half by sharing the results of convolutions in the numerator and denominator of (3). However, this method does not ensure to minimize least-squares error.

3. PROPOSED METHOD

3.1. Range Kernel Decomposition via SVD

We propose $O(1)$ BF that supports convolution sharing for arbitrary range kernel approximated in a least-squares manner. Inspired from [25], we reformulate (1) as

$$\hat{f}_{\boldsymbol{p}} - f_{\boldsymbol{p}} = \frac{\sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q}) w_r(f_{\boldsymbol{p}}, f_{\boldsymbol{q}})(f_{\boldsymbol{q}} - f_{\boldsymbol{p}})}{\sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q}) w_r(f_{\boldsymbol{p}}, f_{\boldsymbol{q}})}. \tag{4}$$

By newly defining $\tilde{w}_r(a,b) = w_r(a,b)(b-a)$, we also decompose it into $\tilde{w}_r(a,b) \approx \sum_{k=0}^{K-1} \tilde{\phi}_k(a)\tilde{\psi}_k(b)$ in the numerator. Substituting them for (4), BF is rewritten as

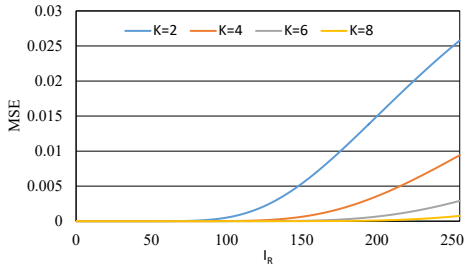

Fig. 1: MSE of kernel approximation via SVD.

$$\hat{f}_{\boldsymbol{p}} - f_{\boldsymbol{p}} \approx \frac{\sum_{k=0}^{K-1} \tilde{\phi}_k(f_{\boldsymbol{p}}) \sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q})\{\tilde{\psi}_k(f_{\boldsymbol{q}})\}}{\sum_{k=0}^{K-1} \phi_k(f_{\boldsymbol{p}}) \sum_{\boldsymbol{q}\in\mathcal{S}} w_s(\boldsymbol{p},\boldsymbol{q})\{\psi_k(f_{\boldsymbol{q}})\}}. \tag{5}$$

If $\psi_k(a,b) = \tilde{\psi}_k(a,b)$, we can share the convolution results of the numerator and denominator, which means the number of the convolutions is reduced from $2K$ to K.

We can find such a kernel decomposition via SVD. First, we introduce $\boldsymbol{W}, \tilde{\boldsymbol{W}} \in \mathbb{R}^{R\times R}$ and $\boldsymbol{W}[a,b] = w_r(a,b)$, $\tilde{\boldsymbol{W}}[a,b] = \tilde{w}_r(a,b)$ where $[\cdot]$ denotes element accessing operator. By applying SVD to the vertically-connected matrix $\boldsymbol{X} = [\boldsymbol{W}^\top, \tilde{\boldsymbol{W}}^\top]^\top \in \mathbb{R}^{2R\times R}$, they can be approximated by the top-K components. Using the singular vectors, each element of $\boldsymbol{X}$ can be represented as the separable form

$$\boldsymbol{X}[a,b] \approx \sum_{k=0}^{K-1} \sigma_k \boldsymbol{u}_k[a] \boldsymbol{v}_k^\top[b], \tag{6}$$

where σ_k ($\sigma_0 \geq \ldots \geq \sigma_{K-1}$) is the k-th singular value, and $\boldsymbol{u}_k \in \mathbb{R}^{2R}$, $\boldsymbol{v}_k \in \mathbb{R}^R$ are the left/right singular vectors. If we set to $\psi_k(b) = \tilde{\psi}_k(b) = \sigma_k \boldsymbol{v}_k[b]$, the intermediate images are shared. Similary, we obtain $\phi_k(a) = \boldsymbol{u}_k[a]$ and $\tilde{\phi}_k(a) = \boldsymbol{u}_k[a+R]$. Thus, we can perform (5) by K convolutions.

3.2. Range Fitting

The default dynamic range in an image is usually $[0 : 255]$, but the actual range depends on the image content. We do not use out of the actual range in the bilateral filter; thus, we can reduce the matrix size of $\boldsymbol{W}$ by $I_{\min}$ and $I_{\max}$, where the minimum and maximal intensity in the input image. We call the process, *range fitting*. After the range fitting, the least square error in the approximated range kernel becomes small. Based on the Frobenius norm, the mean least square error between the ideal range kernel and the approximated one is defined as:

$$e(K, I_R) = \frac{\sum_{a=I_{\min}}^{I_{\max}} \sum_{b=I_{\min}}^{I_{\max}} \{w_r(a,b) - \sum_{k=0}^{K-1} \sigma_k \boldsymbol{u}_k[a]\boldsymbol{v}_k[b]\}^2}{(I_R+1)\times(I_R+1)}. \tag{7}$$

Figure 1 shows the error for with respect to $I_R = I_{\max} - I_{\min}$ and K. When I_R is small, the error of the kernel is small; thus, we approximate the bilateral filtering well. The performance of the range filtering is further improved by tiling, which is introduced in the next section.

The SVD computation is overhead costs; thus, we utilize look-up tables (LUTs) to save the computational time. For computing the range kernel based on the SVD, we need the singular value and vectors σ_k, $\boldsymbol{u}_k$. The LUT size of the sigular values is I_R. The size of vectors is $K \times I_R$ To keep the resulting values, the size of the LUT is $2 \times K \times I_R$. The singular values and vectors are changed based on σ_r, K, and I_R. Therefore, we should pre-compute the values for each parameter for the LUT.

3.3. Efficient Computational Scheduling

This section discusses the efficient computational scheduling of $O(1)$ BF. In general, the whole process of (5) consists of four subprocesses: decomposition D, convolution C, product sum P and normalization N. After precomputation such as SVD, we first generate some intermediate images from an input image as

$$D(\boldsymbol{p}, k) = \psi_k(f_{\boldsymbol{p}}) = \tilde{\psi}_k(f_{\boldsymbol{p}}). \tag{8}$$

All the intermediate images are then convolved with spatial kernel w_s as

$$C(\boldsymbol{p}, k) = \sum_{\boldsymbol{q} \in \mathcal{S}} w_s(\boldsymbol{p}, \boldsymbol{q}) D(\boldsymbol{p}, k) \tag{9}$$

If it is Gaussian spatial kernel, one can use an efficient $O(1)$ GF such as [30, 31, 32] instead of naïve Gaussian convolution, which results in $O(1)$ BF. After computing two product-sums from these results, we normalize the results for each pixel as

$$N(\boldsymbol{p}) = \frac{P_{\mathrm{N}}(\boldsymbol{p})}{P_{\mathrm{D}}(\boldsymbol{p})} = \frac{\sum_{k=0}^{K-1} \tilde{\phi}_k(f_{\boldsymbol{p}})\, C(\boldsymbol{p}, k)}{\sum_{k=0}^{K-1} \phi_k(f_{\boldsymbol{p}})\, C(\boldsymbol{p}, k)}. \tag{10}$$

Note that the intermediate image $C(\boldsymbol{p}, k)$ is shared both in numerator and denominator. These subprocesses should be efficiently operated in real-time processing.

We design the whole process by utilizing parallel processing on the CPU. Here, we assume 2D image filtering, i.e., $\boldsymbol{p} = [x, y]^\top$. In most cases, it is adequate to parallelize the outermost loop for efficient parallel computation in image processing. The tendency, however, is changed by the length of image processing pipeline and the kinds of image processing.

Since the D, C, and P steps contain triple loops (x, y, k) and the N step has double loops (x, y), parallelizing the k-loop seems the most effective. The important point is that the number of elements must be sufficiently more massive than the number of CPU cores for parallelizing load balance. Since the number of k is 4–20 in general, the number is not sufficient for multi-core CPUs. In this case, parallelizing the second outer loop with respect to y could be more effective.

The C step is an exception because we use recursive processing for effective convolution [30, 31, 32]. The recursive filter has a dependency on the processing pixel order of x and y. In this case, we should parallelize the k-loop. Considering memory cache and processing pipeline, parallelizing the k-loop in the step before convolution seems better depending on the processing unit. In the remaining loops, y loop parallelization is effective. As described in computing steps, the $O(1)$ BF has multiple *fork-join* parallel processes and the complex image processing pipeline. Therefore, it has low parallelization efficiency and low cache efficiency.

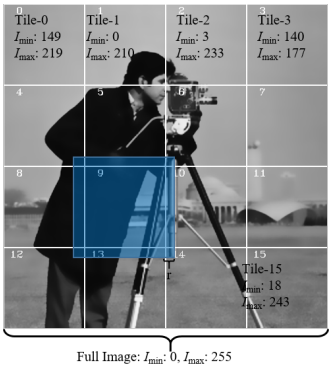

Fig. 2: Illustration of 4×4 tiling. The subimages are expected to have lower contrast (narrower range) than the full image.

We overcome this difficulty by image tiling strategy, which is a major loop optimization technique for compilers. Tiling strategy enables the x and y loops to be split to contain loaded data in processing unit cache to improve cache efficiency. Besides, by processing each tile in parallel, the parallel ability is also improved. Note that, as the blue region in Fig. 2 shows, it is required for supporting parallel processing to care about additional $2r$ pixels for each tile in the convolution step.

More importantly, the tiling strategy improves the performance of range fitting. As Fig. 2 indicates, each subimage tends to represent local texture such as flat or edge parts. In natural images, a local region shows lower contrast, i.e., narrow intensity range. Specifically, $I_R = I_{\max} - I_{\min}$ for each subimage is expected to be smaller than the full image. This mechanism is matched to our SVD approach: narrower range contributes to lower computation cost.

4. EXPERIMENTAL RESULTS

We evaluated the performance of our method for approximation accuracy and computational time. All the codes were implemented in C++, developed on Visual Studio 2017 parallelized by OpenMP and vectorized by AVX. Approximate accuracy is quantified as Peak Signal-to-Noise Ratio (PSNR) between the ideal result (original BF) and filter output.

In the first experiment, we justified approximation accuracy and computational performance of BF with Gaussian range kernel. We compared the proposed method

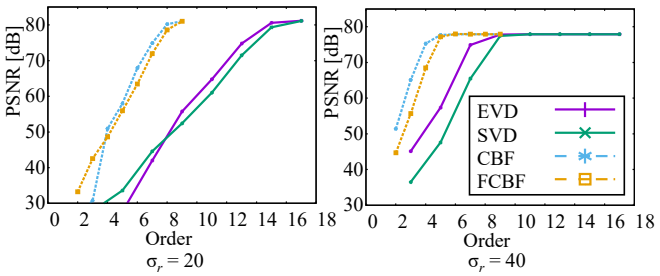

Fig. 3: Order w.r.t. PSNR on various methods ($\sigma_s = 5$).

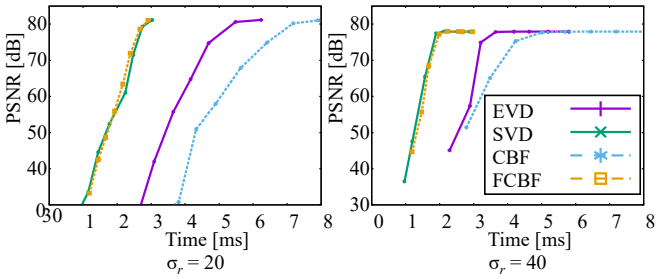

Fig. 4: Computational time [ms] w.r.t. PSNR [dB] on various methods ($\sigma_s = 5$).

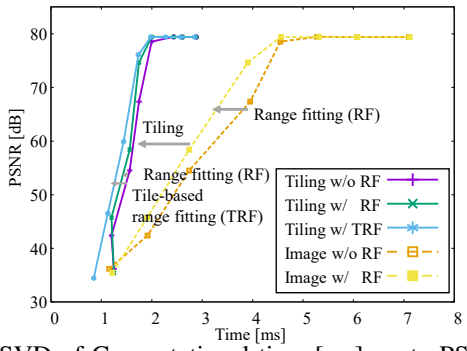

Fig. 5: SVD of Computational time [ms] w.r.t. PSNR [dB] with/without tiling and with/without range fitting ($\sigma_s = 5$).

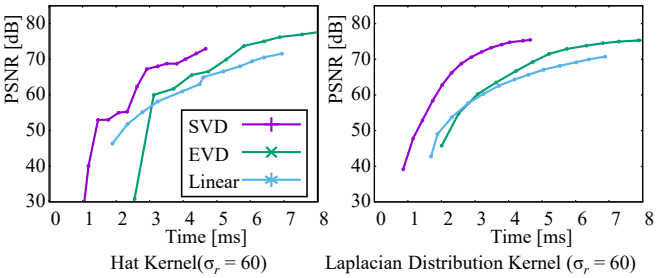

Fig. 6: SVD of Computational time [ms] w.r.t. PSNR [dB] on the hat and Laplacian distribution kernel cases ($\sigma_s = 5$).

with Compressive BF (CBF) [24], its acceleration method called Fast Compressive BF (FCBF) [25] and the EVD-based method [28]. Note that CBF and EVD do not share convolutions of numerator/denominator; by contrast, FCBD and SVD (ours) share convolutions of them. Figure 3 shows the order of CBF/FCBF and EVD/SVD for the approximation accuracy. Under the same order, CBF/EVD achieved higher performance than the shared method. When K and σ_r are small, which is a hard condition for approximating BF, the shared methods FCBF/SVD outperforms the others. This is because FCBF/SVD subtract edges from the source image, but EVD/CBF add edges for blurred image. Resulting images for small σ_r are similar to source images in visibility. Hence, the shared methods have superior performance in this condition. Note that the relation of the order and the number of convolutions is different for each method, such as CBF: $4K + 1$, FCBF: $2K$, EVD: $2K$ and SVD: K.

In order to consider the actual computational performance, we replot the horizontal axis as computational time. Figure 4 plots computational time for the PSNR. SVD (our method) is faster with almost the same accuracy than FCBF. Note that all the methods are parallelized and optimized by 4×4 tiling strategy.

In the second experiment, we confirmed the effectiveness of tiling strategy and range fitting. Figure 5 shows the results of SVD with/without tiling and with/without range fitting. Note that tiling with range fitting (Tiling w/ RF) is the same curve of SVD in Fig. 4. The tiling accelerates $O(1)$ BF significantly. Range fitting improves the performance for both with/without tiling cases. Also, the tile-based range fitting further improves performance. The precomputing overhead for computing I_R took only 0.03 [ms].

In the third experiment, we verified that our method was able to approximate various range kernels. We tested hat kernel (or the Bartlett window, triangular window), defined by

$$w_r(a, b) := \max(1 - \frac{|a - b|}{\sigma}, 0). \tag{11}$$

The double exponential (or Laplace) distribution kernel is defined by

$$w_r(a, b) := e^{-\frac{|a-b|}{\sigma}}. \tag{12}$$

Note that this is not Laplacian filtering for edge detection. Both kernels are non-differentiable and, obviously, not Gaussian. We compared the proposed method with the linear interpolation approach [18] and EVD, which support arbitrary range kernels. Figure 6 reveals performance in the cases of the Bartlett window and the Laplace distribution kernel. The proposed method outperformed both state-of-the-art methods.

5. CONCLUSION

This paper proposed $O(1)$ BF designed via SVD that supports arbitrary range kernel and reduces the number of convolutions by half. Moreover, image tiling strategy and range fitting improve the performance drastically. Our method showed higher performance in both accuracy and computational time. The proposed method achieves sufficient approximation accuracy within 5 ms for arbitrary range kernel approximation in 512×512 size image; thus, the frame rate of our method is over 200 frame per second.

6. REFERENCES

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 1998.

[2] M. Elad, "On the origin of the bilateral filter and ways to improve it," *IEEE Transactions on image processing*, vol. 11, no. 10, pp. 1141–1151, 2002.

[3] M. Zhang and B. K. Gunturk, "Multiresolution bilateral filtering for image denoising," *IEEE Transactions on image processing*, vol. 17, no. 12, pp. 2324–2333, 2008.

[4] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 257–266, 2002.

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

[6] S. Dai, M. Han, Y. Wu, and Y. Gong, "Bilateral back-projection for single image super resolution.," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2007, pp. 1039–1042.

[7] S. Cho and S. Lee, "Fast motion deblurring," *ACM Transactions on graphics*, vol. 28, no. 5, pp. 145, 2009.

[8] T. Matsuo, S. Fujita, N. Fukushima, and Y. Ishibashi, "Efficient edge-awareness propagation via single-map filtering for edge-preserving stereo matching," in *Proc. Three-Dimensional Image Processing, Measurement (3DIPM), and Applications*, 2015.

[9] N. Fukushima, K. Sugimoto, and S. Kamata, "Guided image filtering with arbitrary window function," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[10] T. Matsuo, N. Fukushima, and Y. Ishibashi, "Weighted joint bilateral filter with slope depth compensation filter for depth map refinement," in *Proc. The International Conference on Computer Vision Theory and Applications (VISAPP)*, 2013, pp. 300–309.

[11] S. Fujita, T. Matsuo, N. Fukushima, and Y. Ishibashi, "Cost volume refinement filter for post filtering of visual corresponding," in *Proc. Image Processing: Algorithms and Systems XIII*, 2015.

[12] T. Q. Pham and L. J. V. Vliet, "Separable bilateral filtering for fast video preprocessing," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[13] N. Fukushima, S. Fujita, and Y. Ishibashi, "Switching dual kernels for separable edge-preserving filtering," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[14] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," in *Proc. European conference on computer vision (ECCV)*, 2006.

[15] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[16] F. Porikli, "Constant time o(1) bilateral filtering," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[17] F. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[18] Q. Yang, K. H. Tan, and N. Ahuja, "Real-time o(1) bilateral filtering," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[19] R. Deriche, "Recursively implementating the gaussian and its derivatives," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 1992, pp. 263–267.

[20] Rachid Deriche, "Recursively implementating the gaussian and its derivatives," *Research Report RR-1893, INRIA*, p. 24, 1993.

[21] K. N. Chaudhury, "Constant-time filtering using shiftable kernels," *IEEE Signal Processing Letters*, vol. 18, no. 11, pp. 651–654, 2011.

[22] 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, 2011.

[23] K. N. Chaudhury, "Acceleration of the shiftable o(1) algorithm for bilateral filtering and nonlocal means," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1291–1300, 2013.

[24] K. Sugimoto and S. Kamata, "Compressive bilateral filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3357–3369, 2015.

[25] G. Deng, "Fast compressive bilateral filter," *Electronics Letters*, vol. 53, no. 3, pp. 150–152, 2017.

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

[27] K. N. Chaudhury and S. D. Dabhade, "Fast and provably accurate bilateral filtering," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2519–2528, 2016.

[28] K. Sugimoto, T. Breckon, and S. Kamata, "Constant-time bilateral filter using spectral decomposition," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2016.

[29] G. Papari, N. Idowu, and T. Varslot, "Fast bilateral filtering for denoising large 3d images," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 251–261, 2017.

[30] K. Sugimoto and S. Kamata, "Fast gaussian filter with second-order shift property of dct-5," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2013.

[31] K. Sugimoto and S. Kamata, "Efficient constant-time gaussian filtering with sliding dct/dst-5 and dual-domain error minimization," *ITE Transactions on Media Technology and Applications*, vol. 3, no. 1, pp. 12–21, 2015.

[32] K. Sugimoto, S. Kyochi, and S. Kamata, "Universal approach for dct-based constant-time gaussian filter with moment preservation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.