Stability of Recursive Gaussian Filtering for Piecewise Linear Bilateral Filtering

Koichiro Watanabe, Yoshihiro Maeda, and Norishige Fukushima*
Nagoya Institute of Technology, Nagoya, Japan
*fukushima@nitech.ac.jp

***Abstract*—Constant-time bilateral filtering requires recursive Gaussian filtering for acceleration. An issue of this research is lower stability in the Gaussian filtering for constant-time bilateral filtering. The stability problem in constant-time bilateral filtering is more serious than the natural image filtering case. In this paper, we clarify where the degradation from the instability and which constant-time Gaussian filtering is stable for the approximation of bilateral filtering.**

***Index Terms*—constant-time bilateral filtering, recursive Gaussian filtering, edge-preserving filter, acceleration**

I. Introduction

Bilateral filtering (BF) [1] is edge-preserving filtering, and the filter is based on Gaussian convolutions of spatial and range domains. An issue of BF is high computational complexity; thus, many acceleration approaches are proposed. FFT based approach is a seminal work [2], and its down-sampled approach [3] further accelerates the performance. Separable filtering [4], [5] improves computational order from $O(r^2)$ to $O(r)$, where r is filtering kernel radii. Histogram based filtering [6] makes the order $O(1)$. Constant-time BF [7]–[12] more sophisticatedly approximates the $O(1)$ BF.

In the constant-time BF, the filter is decomposed into the product-sum of the outputs of the Gaussian filtering (GF). In this deformation, the range kernel is approximated by using a piecewise linear approximation, polynomial approximation or trigonometric function decomposition. GF can be implemented as radius-independent filtering by recursive representation [13]–[18]; thus, BF is also filtering radius-independent in the constant-time BF.

GF is classical filtering, although, it is a fundamental tool of image processing. Therefore, the acceleration of this filter has been continuously researched at present. The main difference from the conventional GF work is that filtering targets are not natural images. As a result, the stability in the constant-time BF decreases as compared with the case of natural images. Therefore, we cannot directly use the optimal GF implementation for acceleration, and we should focus the stability of the numerical computation. In the most of the previous paper of constant-time BF, most verifications are achieved in Matlab or double floating point precision; thus, this stability problem has not been outstanding. For further acceleration with single precision computing, the stability problem becomes more conspicuous. Also, narrow range kernel filtering of BF becomes more unstable. In such cases, filtering for weak noise images and depth map/optical flow refinements [19]–[21] are topical applications.

In this paper, we clarify where the stability problem occurs and which constant-time GF is stable in many implementations for the approximation of BF.

II. Related Works

A. Constant-time bilateral filtering

BF is defined by;

$$\tilde{f}(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\, w_r\big(f(\boldsymbol{p}), f(\boldsymbol{q})\big) f(\boldsymbol{q})}{\sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\, w_r\big(f(\boldsymbol{p}), f(\boldsymbol{q})\big)}, \tag{1}$$

where $\mathcal{S}$ and $\mathcal{R}$ are spatial and range domains of images. An input image is $f:\mathcal{S}\rightarrow\mathcal{R}$ and an output image is $\tilde{f}:\mathcal{S}\rightarrow\mathcal{R}$. $\mathcal{N}(\boldsymbol{p})$ are the set of neighboring pixels of $\boldsymbol{p}$. w_s and w_r are spatial and range kernels, respectively, and are defined by;

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

where $\|\cdot\|_2$ is L_2 norm, σ_s and σ_r are spatial and range scales, respectively.

In constant-time BF, the range kernel is approximated by variables separation. Substituting the approximate equation $w_r(a,b) \approx \sum_{k=0}^{K-1} \phi_k(a)\psi_k(b)$ for (1), the results are;

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

where K is the quantization levels of range kernel. Let be images g_k and h_k that satisfy the following conditions;

$$g_k(\boldsymbol{q}) = \psi_k(f(\boldsymbol{q})) f(\boldsymbol{q}), \quad h_k(\boldsymbol{q}) = \psi_k(f(\boldsymbol{q})). \tag{4}$$

Considering g_k and h_k as images, and $\sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\{\cdot\}$ as a GF convolution, the constant-time BF is expressed by the product-sum of the Gaussian convolution with normalization.

B. Real-time $O(1)$ bilateral filtering

Real-time $O(1)$ bilateral filtering (RBF) [7] is a constant-time BF that separates range kernel w_r by a piecewise linear approximation. Equation (5) is substituted into the separated range kernels ϕ_k and ψ_k.

$$\phi_k(a) = \max\left(0, 1 - \frac{K-1}{R^{\max}}|a-\theta_k|\right), \quad \psi_k(b) = w_r(\theta_k, b), \tag{5}$$

This work was supported by KAKENHI JP17H01764.

where $R^{\max}$ is the Maximum value of range domain, θ_k is the quantized range value. Equation (6) is led by to be substituted for Eq. (4) and (5) into Eq. (3);

$$\tilde{f}(\boldsymbol{p}) \approx \sum_{k=0}^{K-1} \phi_k(f(\boldsymbol{p})) \frac{\sum_{\boldsymbol{q}\in\boldsymbol{\mathcal{N}}(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\, g_k(\boldsymbol{q})}{\sum_{\boldsymbol{q}\in\boldsymbol{\mathcal{N}}(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\, h_k(\boldsymbol{q})}. \tag{6}$$

Equation (6) shows that the output of RBF calculated by the piecewise linear interpolation of normalized images generated from the intermediate images g_k and h_k.

C. Constant time Gaussian filtering

Multi-dimensional GF can be separated to one-dimensional GF. Many constant-time GFs [13]–[18] are designed for the one-dimensional filtering and extend the 1D filtering for higher dimensions by using the separability. Therefore, we discuss the one-dimensional GFs.

GF is the weighted average filtering based on the Gaussian distribution. The simplest implementation is approximating a Gaussian convolution as the finite impulse response (FIR) filtering. The correct convolution requires infinite kernel radii, but FIR filtering cuts off finite radii, such as 3σ or 6σ. The principal problem with this filtering is the computational complexity depending on the spatial scale σ_s; thus, various methods are proposed to solve this problem. In this paper, we use the methods with infinite impulse response (IIR) filtering, and a method using a recursive representation of discrete cosine transform (DCT) based filtering.

IIR filtering utilizes feedbacks of the outputs to subsequent processing, and a few taps filter can calculate the influence of pixels at infinity. More terms filtering can approximate Gaussian convolution with high accuracy; however, this filtering requires high precision due to the high influence of numerical computing errors. Since IIR filtering cannot handle future time, the Gaussian convolution requires forward and backward passes. There are two ways to connect the forward and backward filtering, i.e., the series and parallel types. The series type filters utilize the resulting output of the forward filtering for backward-pass. The parallel type filters just add the outputs of the forward and backward filtering in the same way.

Per-pixel DCT can decompose the Gaussian convolution into a sum of low-frequency cosine terms. Recursive representation of the filter can make this filter $O(1)$ per pixel, such as box filtering [22]. Note that this filter is not IIR based filtering, but this filter approximates FIR based GF by stacking the filtering output of short time DCT.

In this paper, as constant-time GF, we use Vliet-Young-Verbeek (VYV) [13], [14], [23], Deriche [15], Alvarez-Mazorra (AM) [16], and spectral recursive GF (SR) [17], [18].

VYV is the series type of IIR filtering. The forward and backward filters are defined by Eq. (7) and (8).

$$v^{\mathrm{vyv}}(n) = b^{\mathrm{vyv}} x(n) - \sum_{m=1}^{M} a_m^{\mathrm{vyv}} v^{\mathrm{vyv}}(n-m), \tag{7}$$

$$y(n) = b^{\mathrm{vyv}} v^{\mathrm{vyv}}(n) - \sum_{m=1}^{M} a_m^{\mathrm{vyv}} y(n+m), \tag{8}$$

where $x(n)$ and $y(n)\,(n = 0, 1, \cdots, N-1)$ are input and output signals, a^{vyv} and b^{vyv} are the filter coefficients, M is the filter order and m is the index of the order, respectively. v^{vyv} is output signals of the forward filtering. VYV can apply the filter order $M = 3, 4, 5$ in this paper.

Deriche is the parallel type IIR filtering. The forward and backward filters are defined by Eq. (9) and (10).

$$v^{\mathrm{der}+}(n) = \sum_{m=0}^{M-1} b_m^{\mathrm{der}+} x(n-m) - \sum_{m=1}^{M} a_m^{\mathrm{der}} v^{\mathrm{der}+}(n-m), \tag{9}$$

$$v^{\mathrm{der}-}(n) = \sum_{m=1}^{M} b_m^{\mathrm{der}-} x(n+m) - \sum_{m=1}^{M} a_m^{\mathrm{der}} v^{\mathrm{der}-}(n+m), \tag{10}$$

where a^{der}, $b^{\mathrm{der}+}$, and $b^{\mathrm{der}-}$ are filter coefficients. $v^{\mathrm{der}+}$ and $v^{\mathrm{der}-}$ are output signals of the forward and backward filtering. The output signals y is summed up of the forward and backward passes;

$$y(n) = v^{\mathrm{der}+}(n) + v^{\mathrm{der}-}(n). \tag{11}$$

Deriche can apply the filter order $M = 2, 3, 4$ in this paper.

AM is the series type IIR filtering. This filtering consists of scaling step and filtering steps. First, the scaling step is implemented in accordance with Eq. (12).

$$u_0^-(n) = \lambda x(n), \tag{12}$$

where λ is scale factor. Then, forward and backward filters are repeated M-times. The forward and backward filters are defined by Eq. (13) and (14);

$$u_m^+(n) = u_{m-1}^-(n) + \nu u_m^+(n-1), \tag{13}$$

$$u_m^-(n) = u_m^+(n) + \nu u_m^-(n+1), \tag{14}$$

where ν is a filter coefficient. u_m^+ and u_m^- are output signals of the m-th forward and backward filters. The output signals y equals u_M^-. The approximation accuracy of filtering is higher when the filter order M increases. If M is infinity, this filtering can approximate the Gaussian convolution correctly. In this paper, we use the filter order $M = 2, 3, 4, 5$.

SR is DCT-5 based filtering. The Gaussian convolution can be approximated by a few low frequency cosine terms. Let direct current components be D and alternating current components be A. D and A are defined by Eq. (15) and (16);

$$D(n) = \sum_{u=-R}^{R} x(n+u), \tag{15}$$

$$A_m(n) = \sum_{u=-R}^{R} \cos\left(\frac{2\pi}{2R+1} mu\right) x(n+u). \tag{16}$$

TABLE I: Computational time and accuracy of various Gaussian filters with various filtering orders.

Method	Single		Double	
	Time [ms]	PSNR [dB]	Time [ms]	PSNR [dB]
VYV ($M = 3$)	2.29	64.70	4.56	64.88
VYV ($M = 4$)	2.72	66.75	5.40	73.15
VYV ($M = 5$)	3.17	39.56	6.28	81.27
Deriche ($M = 2$)	2.47	40.19	4.69	40.19
Deriche ($M = 3$)	2.87	59.32	5.56	59.39
Deriche ($M = 4$)	3.33	81.46	6.43	75.53
AM ($M = 2$)	2.80	47.65	5.79	47.65
AM ($M = 3$)	4.08	50.67	8.44	50.68
AM ($M = 4$)	5.37	52.97	11.10	52.97
AM ($M = 5$)	6.65	54.81	13.84	54.81
AM ($M = 100$)	129.23	80.92	269.01	80.90
SR ($M = 2$)	0.83	77.86	2.89	77.87

The output signals y is the convolution of these components represented by Eq. (17);

$$y\left(n\right) = \beta \left(D\left(n\right) + \sum_{m=1}^{M} \alpha_m A_m \left(n\right) \right), \tag{17}$$

where α and β are the filter coefficients and R is a truncation radius of filtering. The shift property of DCT enables us to recursively compute D and A at an R-independent cost. We implemented this filter with the filter order $M = 2$ and the truncation radius $R = \lceil 3\sigma \rceil$.

TABLE I lists filtering computational time and accuracy of each method under the spatial scales $\sigma_s = 10$. Single and double represent the precision of floating point number that is used for filtering. The test image was a grayscale (768×512) image. The filters are vectorized by SSE and implemented by a single thread. The constitution of the test environment is Intel Core i5-7500 3.4 GHz (4 threads). The ideal output is the output of FIR based GF that is truncated with $\lceil 6\sigma_s \rceil$ and is implemented with double precision.

TABLE I shows that the implementation with single precision is suitable for acceleration than the double precision one. When the same filter order, the filter is faster in the order of SR, VYV, Deriche, and AM. In all filtering method, the filters are fast when the filter order is small, and the filters have high accuracy when it is large. Note that VYV ($M = 5$) implemented with single precision is a low accuracy due to the high influence of numerical calculation error.

III. UNSTABLE CONDITION OF CONSTANT-TIME BILATERAL FILTERING

GF includes the approximation error and the numerical calculation error. GF in constant-time BF is more easily influenced by the numerical calculation error than the natural image filtering case. The effect of the approximation error is slightly buried due to the approximation of range kernels of bilateral filter, which causes large errors. Therefore, constant-time BF generates dedicated errors in low precision computing, such as Fig.1. We discuss the cause of the noise in constant-time BF. In constant-time BF, we apply GF to intermediate images g_k, h_k defined by Eq. (4). These images are weighted by ψ_k; thus, the dynamic range of them is larger extensive than the natural image. In the processing of such a wide dynamic range, the small value is easily influenced by the errors. Also, normalization by the small range value further amplifies the errors. As a result, the constant-time BF easily cause by numerical errors. Figure 2 is an example of the profile curves of Gaussian filtered intermediate images g_k, h_k, and a normalized image $\bar{g_k}/\bar{h_k}$. $\bar{\cdot}$ indicates Gaussian filtering output. The horizontal axis is index columns, and the vertical axis is the range value. Noisy signals in the normalized image have small range values in the intermediate images. A small range scale σ_r generates an intermediate image with a wide dynamic range; thus, the smallness of the range scale is the cause of the noise. This noise is due to the smallness of range weight ψ_k. The range weight ψ_k becomes small when there is the large difference between the quantized range value θ_k and the range value of the input image. Increasing the quantization levels K reduces the gaps between the quantized range value θ_k and the range value of the input image. Therefore, we can depress the generation of noise in the output by increasing the number of intermediate images.

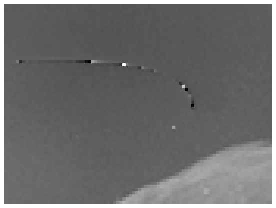
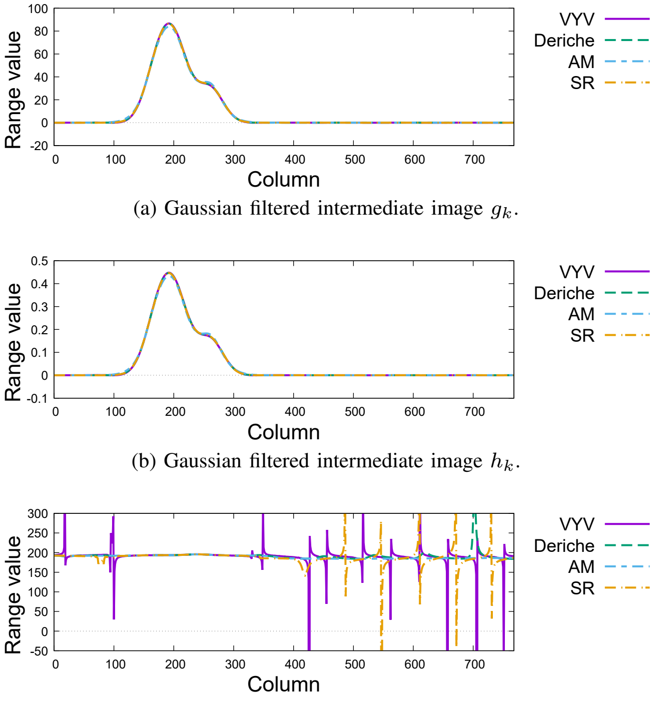


Fig. 1: Example of the dedicated errors in constant-time BF in single precision.

IV. EXPERIMENTAL RESULTS

We examined the constant-time GF from two aspects. In the first experiment, we examined which methods of constant-time GF has high stability in the constant-time BF. In the second experiment, we examined which methods of constant-time GF is efficient in accuracy and speed for the constant-time BF.

In the first experiment, we used RBF implemented with the constant-time GF listed in TABLE I. The code is written in C++ (Visual Studio 2015), and OpenMP is used for parallelization for the each intermediate image. The test image was grayscale Kodak test images, whose resolution is 768×512. We compared the output of RBF based on each GF with the correct image. The correct image is the output of the naïve FIR implementation of BF with double precision 3σ. We use Intel Core i5-7500 3.4 GHz (4 threads). The noise may not be detected by only PSNR metrics as shown in Fig. 3. Therefore, we also evaluated the noise by using the max error in the whole image pixels. Figure 4 shows the relationship between the number of intermediate image pairs and the accuracy of the output image. The left side plots indicate PSNR and the plots

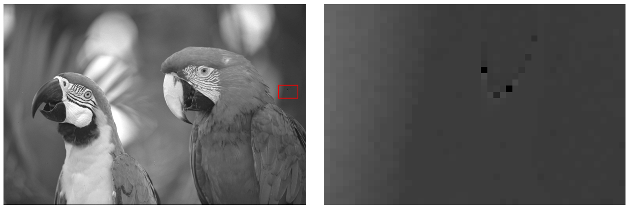


(a) Gaussian filtered intermediate image g_k.

(b) Gaussian filtered intermediate image h_k.

(c) Normalized image generated from g_k, h_k.

Fig. 2: The profile curves of Gaussian filtered intermediate images g_k, h_k, and a normalized image $(\bar{g_k}/\bar{h_k})$.

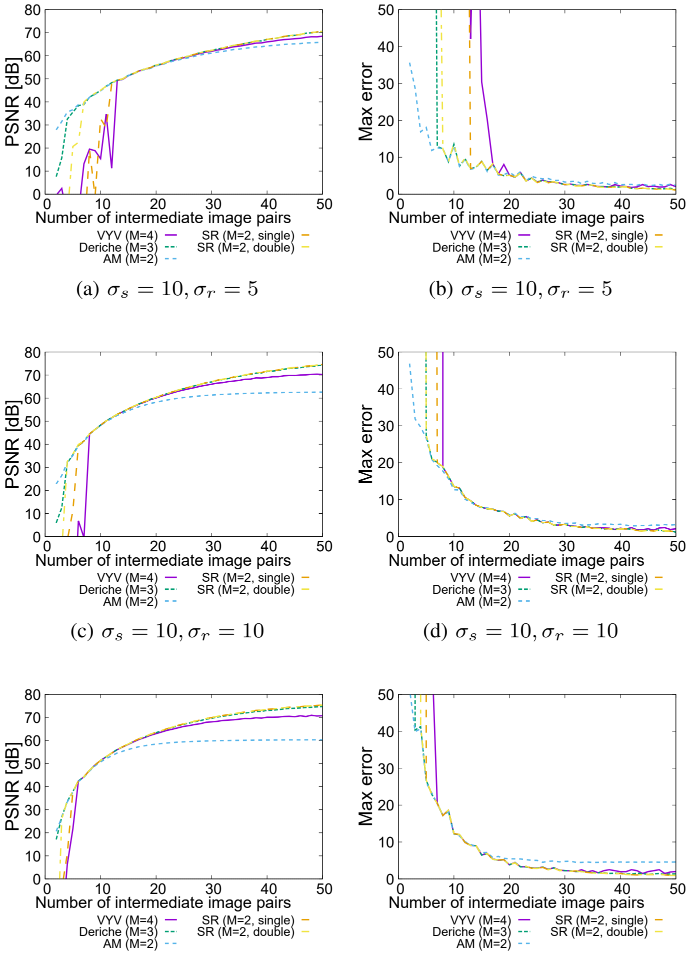


(a) Output image

(b) Zoomed noisy region

Fig. 3: Filtering output with noise caused by unstability. PSNR (54.60 dB).

of the other side indicate the max error. Filter methods are VYV ($M = 4$), Deriche ($M = 3$), AM ($M = 2$), SR ($M = 2$, single), and SR ($M = 2$, double). SR ($M = 2$, single) and SR($M = 2$, double) are the results of SR implemented with single and double precision. The results show that increasing the number of intermediate image pairs improves stability. Also, the results show that constant-time BF easily generates the noise when the range scale σ_r is small. The number of the intermediate images indispensable for suppressing noise is less in the order of AM ($M = 2$), Deriche ($M = 3$), SR ($M = 2$, double), SR ($M = 2$, single), and VYV ($M = 4$). This result shows that the constant-time GF used in constant-time BF has high stability in this order. On the other hand, the filtering time is short in reverse order except for VYV. Therefore, it is essential to consider the trade-off between the time and stability.



(a) $\sigma_s = 10, \sigma_r = 5$

(b) $\sigma_s = 10, \sigma_r = 5$

(c) $\sigma_s = 10, \sigma_r = 10$

(d) $\sigma_s = 10, \sigma_r = 10$

(e) $\sigma_s = 10, \sigma_r = 15$

(f) $\sigma_s = 10, \sigma_r = 15$

Fig. 4: Accuracy of filtering output w.r.t. the number of pairs of intermediate images.

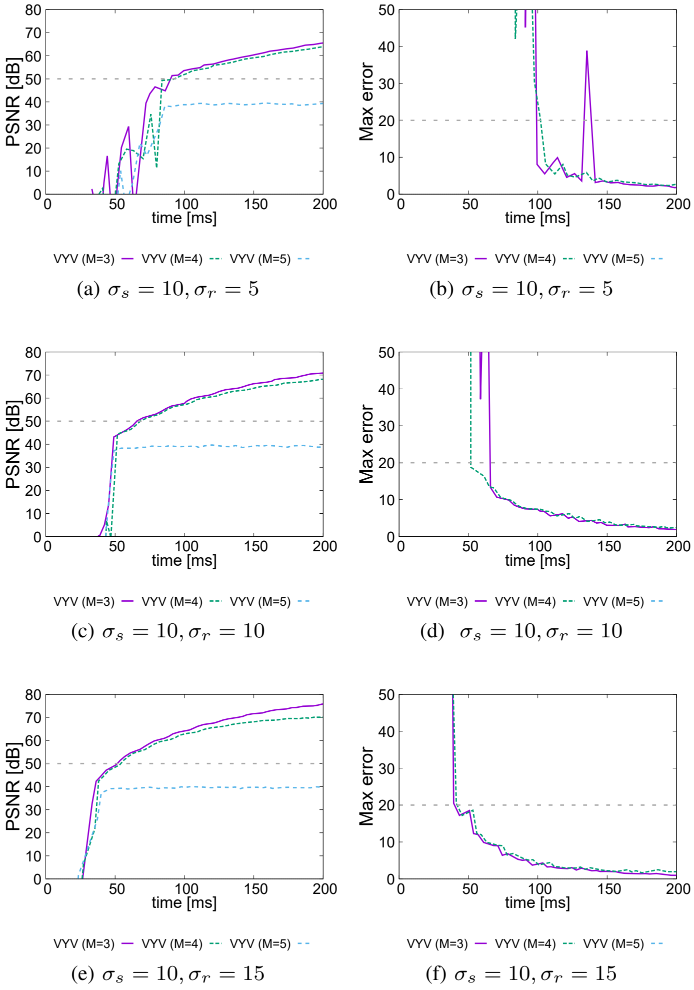
In the second experiment, we examined which methods of constant-time GF is efficient for the constant-time BF. Figures 5, 6, and 7 show the relationship between filtering time and accuracy of RBF. We drew a line with PSNR 50 dB and error value 20 as a line showing a sufficient accuracy.

Figure 5 shows VYV ($M = 5$) is a low accuracy due to the influence of the numerical calculation error. Also, Fig. 5b shows VYV ($M = 3$) is unstable when the small range scale σ_r. The filter order 4 is the most stable parameter for VYV.

Figures 6a and 6b show Deriche ($M = 2$) and Deriche ($M = 4$) are unstable when the small range scale σ_r. The filter order $M = 3$ is the most stable parameter for Deriche.

Figure 7 shows AM is stable on all filter orders. Considering the time and accuracy, the filter order $M = 2$ is the most efficient parameter for AM.

Figure 8 shows the relationships between the accuracy and time of each filter method; specifically, filter methods are VYV ($M = 4$), Deriche ($M = 3$), AM ($M = 2$), SR ($M = 2$, single), and SR ($M = 2$, double). When small range value, the methods are stable with fast computation in the order AM



(a) $\sigma_s = 10, \sigma_r = 5$

(b) $\sigma_s = 10, \sigma_r = 5$

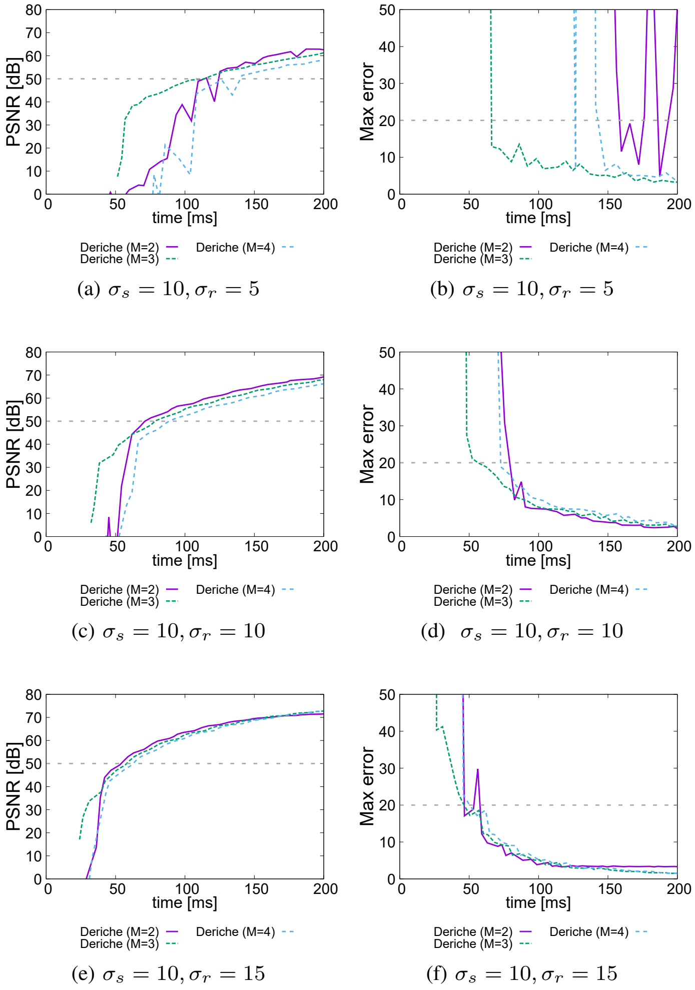
(c) $\sigma_s = 10, \sigma_r = 10$

(d) $\sigma_s = 10, \sigma_r = 10$

(e) $\sigma_s = 10, \sigma_r = 15$

(f) $\sigma_s = 10, \sigma_r = 15$

Fig. 5: Accuracy and filtering time (VYV).



(a) $\sigma_s = 10, \sigma_r = 5$

(b) $\sigma_s = 10, \sigma_r = 5$

(c) $\sigma_s = 10, \sigma_r = 10$

(d) $\sigma_s = 10, \sigma_r = 10$

(e) $\sigma_s = 10, \sigma_r = 15$

(f) $\sigma_s = 10, \sigma_r = 15$

Fig. 6: Accuracy and filtering time (Deriche).

($M = 2$), SR ($M = 2$, double), Deriche ($M = 3$), SR ($M = 2$, single), and VYV ($M = 4$) . If a low accuracy is acceptable, AM ($M = 2$) or SR ($M = 2$, double) is efficient. In the environment that is impossible to implement with double precision, such as consumer GPU computation, Deriche ($M = 3$) is also suitable. For the other case, SR ($M = 2$, single) is the most efficient of these filters.

V. Conclusion

In this paper, we showed the cause of the noise in the constant-time BF. The main reason is that errors of GF targeting the wide dynamic range image and is the amplification of the errors by normalization generates characteristic noises. The constant-time BF has low stability when the range scale σ_r is small, and we can stabilize this filter by increasing the number of the intermediate images. The constant-time GF used in the constant-time BF has high stability in the order of AM, Deriche, SR, and VYV. On the other hand, the filtering time is in reverse order; therefore, a trade-off between time and stability is essential. If a low accuracy is not a problem, AM ($M = 2$) or SR ($M = 2$, double) implementation is efficient in the smaller cases. If the double precision cannot be used, Deriche ($M = 3$) is also suitable. In the other case, SR ($M = 2$, single) is the most efficient of these filters.

References

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

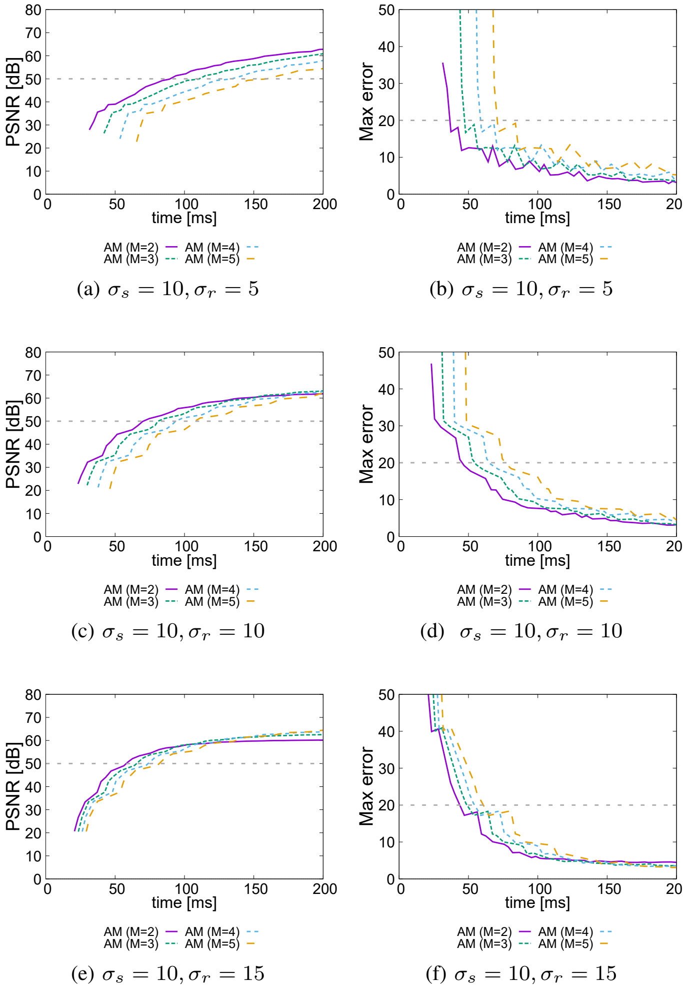
[2] F. Durand and J. Dorsey, “Fast bilateral filtering for the display of high-dynamic-range images,” *ACM Trans. on Graphics*, vol. 21, no. 3, pp. 257–266, 2002.

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

[4] 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.

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

[6] F. Porikli, “Constant time o(1) bilateral filtering,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

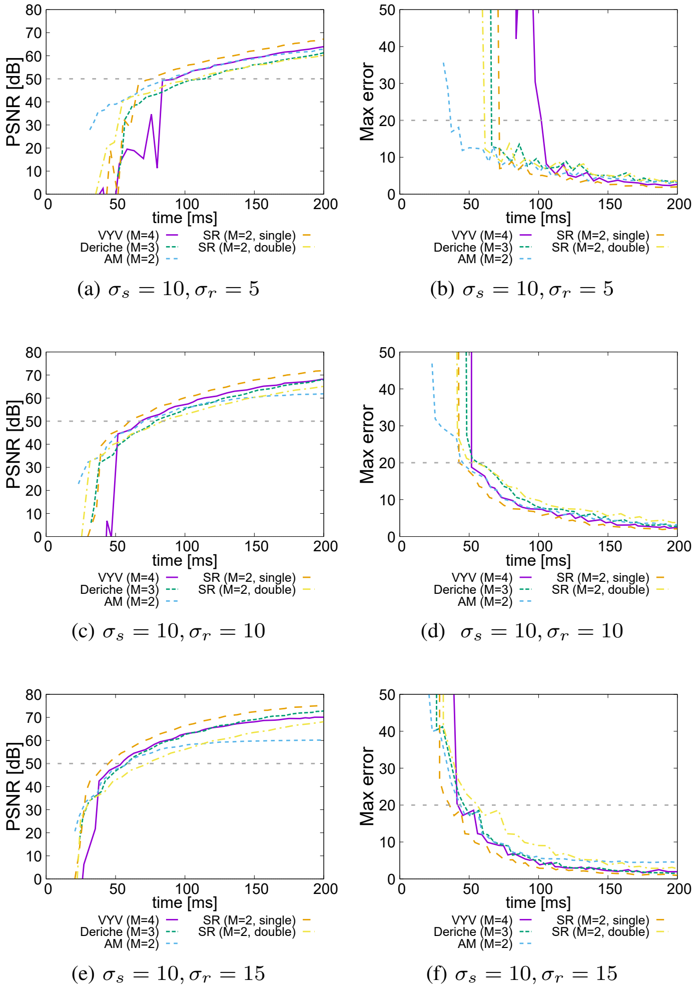


(a) $\sigma_s = 10, \sigma_r = 5$ (b) $\sigma_s = 10, \sigma_r = 5$

(c) $\sigma_s = 10, \sigma_r = 10$ (d) $\sigma_s = 10, \sigma_r = 10$

(e) $\sigma_s = 10, \sigma_r = 15$ (f) $\sigma_s = 10, \sigma_r = 15$

Fig. 7: Accuracy and filtering time (AM).



(a) $\sigma_s = 10, \sigma_r = 5$ (b) $\sigma_s = 10, \sigma_r = 5$

(c) $\sigma_s = 10, \sigma_r = 10$ (d) $\sigma_s = 10, \sigma_r = 10$

(e) $\sigma_s = 10, \sigma_r = 15$ (f) $\sigma_s = 10, \sigma_r = 15$

Fig. 8: Accuracy and filtering time (compare each method).

[7] Q. Yang, K. H. Tan, and N. Ahuja, "Real-time o(1) bilateral filtering," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 557–564.

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

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

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

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

[12] K. Sugimoto, N. Fukushima, and S. Kamata, "Fast bilateral filter for multichannel images via soft-assignment coding," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016.

[13] L. J. van Vliet, I. T. Young, and P. W. Verbeek, "Recursive implementation of the gaussian filter," in *Proc. Signal Processing*, vol. 44, no. 2, 1995, pp. 139 – 151.

[14] ——, "Recursive gaussian derivative filters," in *Proc. Fourteenth International Conference on Pattern Recognition*, vol. 1, 1998, pp. 509–514 vol.1.

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

[16] L. Alvarez and L. Mazorra, "Signal and image restoration using shock filters and anisotropic diffusion," *SIAM Journal on Numerical Analysis*, vol. 31, no. 2, pp. 590–605, 1994.

[17] K. Sugimoto and S. i. Kamata, "Fast gaussian filter with second-order shift property of dct-5," in *Proc. IEEE International Conference on Image Processing*, 2013, pp. 514–518.

[18] K. Sugimoto and S.-I. 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.

[19] T. Matsuo, N. Fukushima, and Y. Ishibashi, "Weighted joint bilateral filter with slope depth compensation filter for depth map refinement," in *Proc. The 8th International Conference on Computer Vision Theory and Applications (VISAPP)*, 2013, pp. 300–309.

[20] 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.

[21] 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.

[22] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. ACM SIGGRAPH*, 1984, pp. 207–212.

[23] B. Triggs and M. Sdika, "Boundary conditions for young-van vliet recursive filtering," in *Proc. IEEE Transactions on Signal Processing*, vol. 54, no. 6, 2006, pp. 2365–2367.