CONSTANT TIME BILATERAL FILTERING FOR COLOR IMAGES

Ying-An Lai, Wei-Chih Tu, and Shao-Yi Chien

Graduate Institute of Electronics Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan

ABSTRACT

The bilateral filter is a commonly used technique in image processing. However, being nonlinear, it is computationally expensive. The situation gets worse while the filter radius grows up. Several works have been proposed to accelerate the computation. Nevertheless, most techniques are tailored for gray-scale image bilateral filtering or confined to specific kernel functions. In this paper, we propose a constant time bilateral filtering for color images. Specifically, we extend an existing constant time bilateral filtering technique, which has been demonstrated the state-of-the-art for gray-scale images. We generalize the original approach as a three-stage algorithm. Based on the generalization, we explain the drawbacks of its naïve extension for color images and propose a solution that adapts to the image content. Experimental results demonstrate the effectiveness of our solution.

Index Terms— Constant time algorithm, bilateral filtering, adaptive sampling, color image processing

1. INTRODUCTION

The bilateral filter [1] has long been an important technique in image processing. It can smooth images while carefully preserve edges. The general form can be expressed as:

$$I^F(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Omega(\mathbf{p})} g_s(\mathbf{p},\mathbf{q}) g_r(I(\mathbf{p}),I(\mathbf{q}))I(\mathbf{q})}{\sum_{\mathbf{q}\in\Omega(\mathbf{p})} g_s(\mathbf{p},\mathbf{q}) g_r(I(\mathbf{p}),I(\mathbf{q}))}, \quad (1)$$

where I and I^F denote the input and filtered image respectively. The filter response is a weighted sum of pixels in the neighborhood $\Omega(\mathbf{p})$ of center pixel at site $\mathbf{p}$. The filter weight contains spatial and range kernel functions g_s and g_r. Typically, both kernel functions are low-pass filters, such that spatial kernel assigns lower weights to farther pixels and range kernel assigns lower weights to pixels with larger appearance difference. Note that the choice of range kernel is essential to the filter effect of bilateral filter. One common choice for the range kernel is the Gaussian function $g_r(x,y) = exp(-\frac{(x-y)^2}{2\sigma_r^2})$.

The bilateral filter for color images can also be expressed as Eq. (1), where $I(\mathbf{p})$ becomes a color vector and the range kernel takes the color difference of two pixels into account. For example, if Gaussian function is used, then the range kernel becomes

$$g_r(I(\mathbf{p}),I(\mathbf{q})) = exp(-\frac{\sum_{c\in r,g,b}(I_c(\mathbf{p})-I_c(\mathbf{q}))^2}{2\sigma_r^2}). \quad (2)$$

Being nonlinear, direct computation of the bilateral filter is slow. Let r be the filter radius, the complexity of direct computation is $O(r^2)$, which means the computational burden is getting higher when the filter radius grows up. The issue is important that the image resolution is getting higher so the corresponding filter radius is getting larger as well. Several works have been proposed to accelerate the bilateral filter either by highly accurate approximation or by confining specific kernel functions that are cheaper to evaluate. The key idea behind is that we don't need exact computation of the bilateral filter. Empirically [2], when the numerical accuracy is high enough (PSNR $>$ 40 dB), the approximated bilateral filter is still edge-preserving. Pham and Vliet [3] proposed a separable implementation of bilateral filter. The complexity is $O(r)$ but it suffers from the visual streaking artifacts. Weiss [4] showed that if the spatial kernel is a box filter then the bilateral filter can be evaluated from local histograms. Porikli [2] explored the redundancy of local histograms and proposed a per-pixel constant time $O(1)$ solution using integral histogram. The algorithm can be further accelerated by quantizing the histogram such that the number of bins is reduced. However, quantized histograms can not produce accurate results when the range parameter σ_r is small [5]. Paris et al. [6] proposed to implement the bilateral filter in a high dimensional data structure called bilateral grid. An isotropic filter on the grid corresponds to an edge-aware filter in image domain. They found that when σ_r is large, the filtered image gets more blurry. Based on this, they showed that faster filtering can be conducted in a downscaled grid. However, the technique can only benefit larger σ_r. As a result, the quantized histogram or the bilateral grid can not handle small σ_r well. Yang et al. [5] proposed another $O(1)$ implementation by decomposing the bilateral filter into a set of principle bilateral filtered image components (PBFIC). The PBFIC method can accelerate the bilateral filter with arbitrary spatial and range kernel functions. Moreover, the memory require-

ment is much smaller. As reported in [5], 4 to 8 components are sufficient for gray-scale images, while histogram-based method typically needs more than 64 bins to achieve pleasant results.

Nevertheless, most of above approaches are tailored for gray-scale images. The bilateral grid becomes a 5D grid for color images. It is costly to maintain. On the other hand, histogram-based methods are difficult to scale up since the number of bins grows exponentially. Even the PBFIC method becomes computationally expensive as the number of components grows up. For example, take 4 to 8 components for each channel, then there will be 4^3 to 8^3 components to evaluate. In this paper, we propose an extension of the PBFIC method for color images. Specifically, we generalize the PBFIC method as a three-stage algorithm. The three stages are sampling, filtering and interpolation. We notice that numerical accuracy is highly related to the sampling and interpolation strategies. As a result, we can improve the quality with better sampling and interpolation methods. As each stage is constant time for each pixel, the overall algorithm is also constant time for each pixel.

The rest of the paper is organized as follows. Section 2 reviews the PBFIC method. In section 3, we describe our generalization and based on the it we discuss the problem of naïve extension for color images. The proposed method is also described here. Section 4 shows the experimental results. Finally, we conclude this paper in section 5.

2. REVIEW OF THE PBFIC METHOD

For better understanding, we briefly review the principle bilateral filtered image component (PBFIC) method [5] here. Eq. (1) can be regarded as per-pixel division of two filter responses, J and J_w.

$$J(\mathbf{p}) = \sum_{\mathbf{q}\in\Omega(\mathbf{p})} g_s(\mathbf{p},\mathbf{q}) g_r(I(\mathbf{p}), I(\mathbf{q})) I(\mathbf{q}) \tag{3}$$

$$J_w(\mathbf{p}) = \sum_{\mathbf{q}\in\Omega(\mathbf{p})} g_s(\mathbf{p},\mathbf{q}) g_r(I(\mathbf{p}), I(\mathbf{q}))) \tag{4}$$

$$I^F(\mathbf{p}) = \frac{J(\mathbf{p})}{J_w(\mathbf{p})}. \tag{5}$$

J is the unnormalized bilateral filtered image and J_w is the normalization term. One can regard J_w as another unnormalized bilateral filtered image of an all-one image using the same weights. The PBFIC is defined as the bilateral filter response computed with fixed center pixel value k. Then Eq. (3) becomes

$$J^k(\mathbf{p}) = \sum_{\mathbf{q}\in\Omega(\mathbf{p})} g_s(\mathbf{p},\mathbf{q}) g_r(k, I(\mathbf{q})) I(\mathbf{q}). \tag{6}$$

The advantage of this design is that computing J^k is equivalent to spatial filtering of image $g_r(k, I)I$. Note that $g_r(k, I)I$ can be computed independently for each pixel. As the spatial filtering is $O(1)$, then computing J^k is also $O(1)$. J^k_w can be computed in the same way, resulting the overall complexity $O(1)$. If we compute J^k for every possible $k \in [0, 255]$, then the exact bilateral filter response becomes

$$I^F(\mathbf{p}) = \frac{J^{I(\mathbf{p})}(\mathbf{p})}{J_w^{I(\mathbf{p})}(\mathbf{p})}. \tag{7}$$

As described in [5], we can approximate the bilateral filter by only computing a subset of possible intensities. For those pixels $I(\mathbf{p})$ exist in the subset, their filter responses are still exact. If $I(\mathbf{p})$ is skipped, let k_1 and k_2 be the two nearest intensities in the subset, assume $I(\mathbf{p}) = \alpha k_1 + (1-\alpha)k_2$, then the final result can be computed as

$$I^F(\mathbf{p}) = \alpha \frac{J^{k_1}(\mathbf{p})}{J_w^{k_1}(\mathbf{p})} + (1-\alpha)\frac{J^{k_2}(\mathbf{p})}{J_w^{k_2}(\mathbf{p})}. \tag{8}$$

Empirically, 4 to 8 intensities are sufficient (PSNR $>$ 40 dB) for gray-scale images. Using fewer components also largely reduces the processing time.

3. PROPOSED METHOD

3.1. The three-stage generalization

We reformulate the process of PBFIC method into three stages. They are *sampling, filtering and interpolation*. In the single channel version reviewed in previous section, the sampling step is to determine which intensities are used as the subset. The filtering step is the spatial filtering of image $g_r(k, I)I$ in Eq.(6). And the interpolation is bilinear interpolation. The generalization helps us understand the ways to extend the PBFIC to filter color images and improve it. We notice that the image quality is highly related to the sampling and interpolation strategies. Obviously, more samples result in higher accuracy while it takes more time to compute. Taking the same number of samples while different sampling strategies also results in different quality. Take an extreme example. Assume only two components are used. The first case is sampling at the possible dynamic range, so the sampled intensities are 0 and 255. The second strategy is that intensities are sampled as I_{min} and I_{max} according to the image histogram. In some cases, $|I_{max} - I_{min}|$ can be far less than 255, so the filtered image quality of latter sampling will outperform the former one.

3.2. The problem of naïve extension for color images

The naïve extension of the PBFIC method for color images is referred to as using N intensities for each channel such that in total N^3 color combinations are used as the principle color values for filtering. Though the resulting algorithm still has $O(1)$ complexity, it is a huge constant owing to the cubic

Fig. 1: Test images. From left to right: *Dragon*, *Housecorner*, *Tulip*, *Flower*, *Dog*, *Polin*, *Greekdome*, *Swamp*

factor N^3. To reduce computation, N should be small. However, small N means fewer representative colors and thus the approximation quality is low. Moreover, the color distribution of a natural image [7] is usually sparse in the color space, so the sampled color might be far from the image colors. As a result, the ineffective sampling causes large interpolation errors.

3.3. Adaptive sampling in color space

The sampling of principle colors should take the image content into account. Furthermore, the sampled colors should distributed evenly in color space to avoid large interpolation errors. We propose a strategy based on Poisson disk sampling [8, 9]. Poisson disk sampling generates samples that are stochastic, evenly spaced, but no closer to each other than a given disk radius r_s. We conduct sampling along the color distribution. Specifically, we randomly pick a sample from image colors so the sampled color must be close enough to the distribution. We accept the new sample if the distance from it to every previously sampled color is larger than the disk diameter $2r_s$. When there is no space for a new sample, r_s is reduced. The procedure continues until we get enough number of samples. Note that the number of samples is no longer limited to the cubic number N^3. This advantage makes our algorithm more flexible.

3.4. $O(1)$ spatial filtering

The spatial filter used in the framework needs to be a constant time $O(1)$ approach such that the final algorithm has also $O(1)$ complexity. Box filter can be implemented in $O(1)$ by integral image [10]. Gaussian filter can be $O(1)$ using IIR implementation [11]. We tested the framework with these two spatial filters. More $O(1)$ spatial filters can be found in [2].

3.5. Interpolation

The interpolation in single channel version is simply bilinear interpolation. The weights α and $(1-\alpha)$ are used to balance two filtered components. Similarly, we propose to find the nearest colors in the sample set and use their filtered components to interpolate the final result. In three dimensional color space, at least four points are needed to form a volume to enclose a target point. As a result, we find the four nearest sampled colors and combine their filtered components by assigning weights according to their distance to the target color. Let the four nearest colors be $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_3$ and $\mathbf{k}_4$. Then the final result can be obtained by

$$I^F(\mathbf{p}) = \frac{\sum_i \omega_i \frac{J^{\mathbf{k}_i}(\mathbf{p})}{J_w^{\mathbf{k}_i}(\mathbf{p})}}{\sum_i \omega_i}, \tag{9}$$

where ω_i is a function of distance d_i from a sampled color $\mathbf{k}_i$ to the target color. We have tested linear weights $1/d_i$ and exponential weights $exp(-\frac{d_i}{2\sigma_d})$. In general, the exponential weight performs better, so we use it in our experimental result.

4. EXPERIMENTAL RESULTS

The bilateral filters with box spatial Gaussian range kernel (BsGr) and with Gaussian spatial Gaussian range kernel (GsGr) are tested. Test images are shown in Fig. 1. Both numerical accuracy and processing time are evaluated.

4.1. Numerical accuracy evaluation

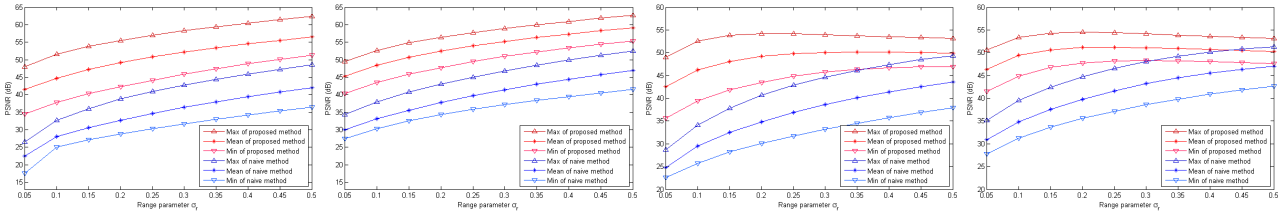
As discussed in section 3.2, smaller number of samples is favored to reduce the overall computation. To compare with the naïve extension , the number of samples is set to a cubic number. We show the result of 3^3 and 4^3 color samples in Fig. 2. The brute force results are used as groundtruth to measure PSNR. For each parameter setting, we plot the maximum, mean and minimum PSNR among all test images.

As one can see, our method can generate superior results that guarantee PSNR over 40 dB, while the naïve sampling approach fails to handle low sample counts due to large interpolation errors. Fig. 3 shows example results of different methods with 27 and 64 samples. We have also tested 2^3 samples. However, in average, 8 samples are not enough to produce pleasant image quality.

Note that our adaptive sampling approach is not limited to the cubic number of samples. We have tested several sample counts and found that we can take fewer samples yet still produce decent filtered image quality. In general, 20 principle colors are sufficient as demonstrated in Fig. 4 and Fig. 5.

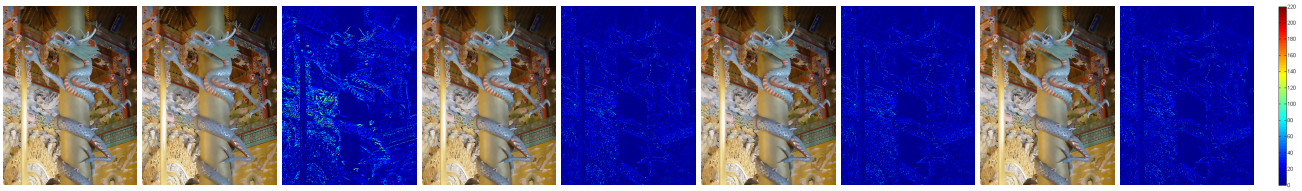
4.2. Runtime evaluation

We evaluate the processing time on a 2.1 GHz Intel Core i7-4600U CPU. The box filter and the Gaussian filter takes 7.8 ms and 28.7 ms to process 1M pixels. Let the number of samples be N, then there will be $4N$ set of filtering, N for J_w^k



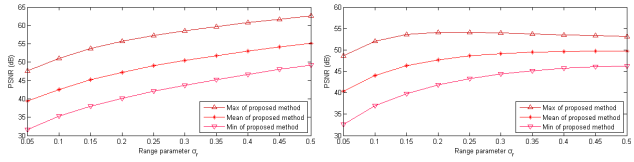
(a) BsGr with 27 samples (b) BsGr with 64 samples (c) GsGr with 27 samples (d) GsGr with 64 samples

Fig. 2: Numerical accuracy evaluation with 3^3 and 4^3 sampled colors. The filter radius is fixed to 15 pixels and the range parameter σ_r is varying (shown in normalized intensity scale.)



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Fig. 3: Comparison of *Dragon* using BsGr bilateral filter, $\sigma_r = 0.15$. (a) Groundtruth (b) Naïve , 27 samples (d) Proposed, 27 samples (f) Naïve , 64 samples (h) Proposed, 64 samples. (c)(e)(g)(i) are the difference maps of (a) and (b)(d)(f)(h), respectively.

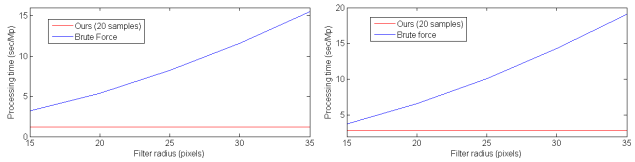


(a) BsGr (b) GsGr

Fig. 4: Our method is accurate even using only 20 samples.

(a) (b) (c)

Fig. 5: GsGr bilateral filter, $\sigma_r = 0.15$ (a) Original image (b) Groundtruth (c) Proposed method with 20 samples



(a) (b)

Fig. 6: Processing time of (a) box spatial kernel and (b) Gaussian spatial kernel using 20 samples.

and $3N$ for J^k (3 channels). The overall processing time can be understood by $T_{sampling}+4N\times T_{filtering}+T_{interpolation}$.

$T_{sampling}$ and $T_{interpolation}$ both increase with N. Using 20 samples to process 1M pixels, it takes 1.1 ms and 23.8 ms, respectively. And it takes 1.2 sec and 2.8 sec in total for box spatial and Gaussian spatial kernels. Fig. 6 compares the processing time with brute force computation with varying filter radius.

5. CONCLUSION

In this paper, we generalize the PBFIC method as a three-stage algorithm and propose an extension for color images. Thanks to the adaptive sampling strategy, the proposed method is able to achieve high accuracy with low sample counts compared to naïve approach. Using lower sample counts also benefits the processing time. Our future work is to implement this framework on GPU for potential real-time color image applications.

6. REFERENCES

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV*, Jan 1998, pp. 839–846.

[2] F. Porikli, "Constant time o(1) bilateral filtering," in *CVPR*, June 2008, pp. 1–8.

[3] T.Q. Pham and L.J. van Vliet, "Separable bilateral filtering for fast video preprocessing," in *ICME*, July 2005, pp. 4 pp.–.

[4] B. Weiss, "Fast median and bilateral filtering," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 519–526, July 2006.

[5] Q. Yang, K.-H Tan, and N. Ahuja, "Real-time o(1) bilateral filtering," in *CVPR*, June 2009, pp. 557–564.

[6] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *IJCV*, vol. 81, no. 1, pp. 24–52, Jan. 2009.

[7] I. Omer and M. Werman, "Color lines: image specific color representation," in *CVPR*, June 2004, vol. 2, pp. II–946–II–953 Vol.2.

[8] M. McCool and E. Fiume, "Hierarchical poisson disk sampling distributions," in *Proceedings of the Conference on Graphics Interface*, San Francisco, CA, USA, 1992, pp. 94–105, Morgan Kaufmann Publishers Inc.

[9] R. Bridson, "Fast poisson disk sampling in arbitrary dimensions," in *ACM SIGGRAPH 2007 Sketches*, New York, NY, USA.

[10] P. Viola and M.J. Jones, "Robust real-time face detection," *ICCV*, vol. 57, no. 2, pp. 137–154, 2004.

[11] L.J. van Vliet, I.T. Young, and P.W. Verbeek, "Recursive gaussian derivative filters," in *ICPR*. IEEE, 1998, vol. 1, pp. 509–514.