

Real-Time $O(1)$ Bilateral Filtering*

Qingxiong Yang* Kar-Han Tan† Narendra Ahuja*

*University of Illinois at Urbana Champaign †HP Labs, Palo Alto

<http://vision.ai.uiuc.edu/~qyang6/>

Abstract

We propose a new bilateral filtering algorithm with computational complexity invariant to filter kernel size, so-called $O(1)$ or constant time in the literature. By showing that a bilateral filter can be decomposed into a number of constant time spatial filters, our method yields a new class of constant time bilateral filters that can have arbitrary spatial¹ and arbitrary range kernels. In contrast, the current available constant time algorithm requires the use of specific spatial or specific range kernels. Also, our algorithm lends itself to a parallel implementation leading to the first real-time $O(1)$ algorithm that we know of. Meanwhile, our algorithm yields higher quality results since we are effectively quantizing the range function instead of quantizing both the range function and the input image. Empirical experiments show that our algorithm not only gives higher PSNR, but is about $10\times$ faster than the state-of-the-art. It also has a small memory footprint, needed only 2% of the memory required by the state-of-the-art for obtaining the same quality as exact using 8-bit images. We also show that our algorithm can be easily extended for $O(1)$ median filtering. Our bilateral filtering algorithm was tested in a number of applications, including HD video conferencing, video abstraction, highlight removal, and multi-focus imaging.

1. Introduction

Originally introduced by Tomasi and Manduchi [2], bilateral filters are edge preserving operators that have found widespread use in many computer vision and graphics tasks like denoising [3, 4, 5, 6, 7], texture editing and relighting [8], tone management [9, 10], demosaicking [11], stylization [12], optical-flow estimation [13, 14] and stereo matching [15, 16].

Until recently, bilateral filters were too computationally intensive for real time applications. Several efficient numerical schemes [9, 17, 18, 19, 20] enable it to be computed at

interactive speed or even video rate using GPU (Graphics Processing Unit) implementation [21]. With the exception of [19], which approximates the bilateral by filtering subsampled copies of the image, these algorithms do not scale well since they become more expensive as the filtering window size grows, which limits their utility in high resolution real time applications. [19] actually becomes faster as the size increases due to greater subsampling, but the exact output is dependent on the phase of subsampling.

It was therefore a significant advance when Porikli [1] demonstrated that bilateral filters can be computed at constant time with respect to filter size for three types of bilateral filters. (1) Box spatial and arbitrary range kernels. Integral histogram is used to avoid the redundant operations and interactive speed is achieved by quantizing the input image using a small number of bins, thus trading memory footprint and image quality for speed. For a 8-bit grayscale image, assume 256 bins are used to compute and store the integral histogram, $256\times$ the size of the image memory are required. The memory could be reduced but will also change single integral histogram computation to be 256 times, which will be much slower. (2) Arbitrary spatial¹ and polynomial range kernels. A bilateral filter of this form can be interpreted as the weighted sum of the spatial filtered responses of the powers of the original image. No approximation is used in this method. (3) Arbitrary spatial¹ and Gaussian range kernels. Taylor series is used to approximate the Gaussian range function up to the four order derivatives. However, this method is a bad approximation for small Gaussian variances.

A new $O(1)$ bilateral filtering method extending Durand and Dorsey's piecewise-linear bilateral filtering method [9] is proposed in the paper. As in [9], we discretize the image intensities into a number of values, and compute a linear filter for each such value, the output of which is defined as Principle Bilateral Filtered Image Component (PBFIC) in this paper. The final output is then a linear interpolation between the two closest PBFICs. Instead of confining the kernels to be Gaussian spatial and Gaussian range and using Fast Fourier Transform (FFT) for Gaussian convolution which has cost $O(\log r)$ (r is the filter radius), we show that

*This work is supported by a grant from HP lab. Part of this work was done while Qingxiong Yang was an intern in HP Lab.

¹an IIR $O(1)$ solution needs to be available for the kernel.

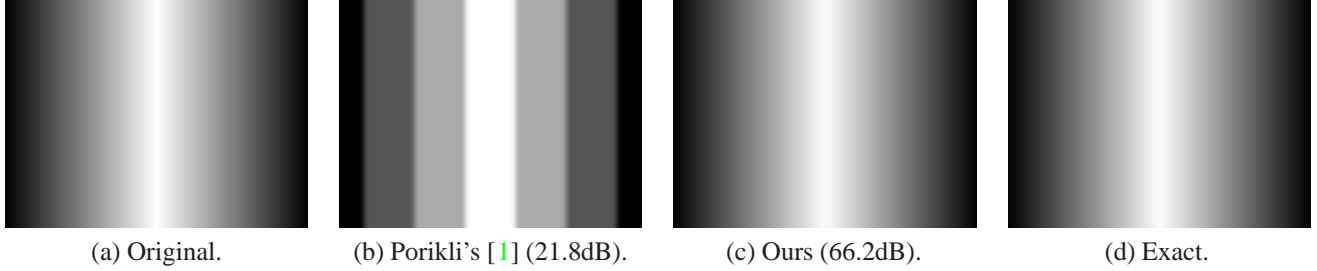


Figure 1. Robustness to quantization. (a) is a synthetic 8-bit grayscale image with resolution 512×384 , (b) and (c) are bilateral filtered results using Porikli’s method [1] and our method with the same parameter setting, respectively. Box spatial and Gaussian range kernels are adopted. The spatial filter size is 21×21 , and the range variance is $\sigma_R^2 = 0.006$. Only 4 bins/PBFICs are used to obtain (b) and (c). (d) is the filtered image without quantization (256 PBFICs). It’s the same as exact. Our method outperforms Porikli’s method because only coefficients of the range filter are quantized in our method, while in Porikli’s method, the image intensities are also quantized.

the discretization method can be directly extended to obtain $O(1)$ bilateral filtering with arbitrary spatial and arbitrary range kernels assuming that the exact or approximated spatial filter can be computed at constant time, *e.g.*, box filtering using integral image [22], Gaussian filtering using recursive filtering [23], and Polynomial filtering using a set of integral images and more as presented in [1]. We also extend the method for $O(1)$ median filtering using integral image.

The contribution of this paper is a new algorithm for constant time bilateral filtering with the following advantages over the state-of-the-art [1]:

1. Uniform framework for constant time bilateral filtering with *arbitrary spatial*¹ and *arbitrary range* kernels. In [1], only three types of $O(1)$ bilateral filtering is available.
2. Better Gaussian range function representation. Although the range function is quantized in our method, it is valid for both low and high variance Gaussian. The bilateral filtering method with arbitrary spatial and Gaussian range kernels presented in [1] uses Taylor series approximation, which is a bad approximation for low variance Gaussian. It is important to note that many applications require low range variance to preserve edges.
3. More accurate. Our method only quantizes the range function, while in [1], image intensities are also quantized, resulting in lower accuracy as shown in Figure 1.
4. Faster ($10\times$). Our method can be easily implemented in parallel. On the NVIDIA Geforce 8800 GTX GPU, we show that given the same output accuracy, our method can be about $10\times$ faster on average.
5. Lower memory requirement (2%) enabling processing of high resolution images/videos. For Box spatial bilateral filtering with 8-bit grayscale images, to obtain

the exact bilateral filtering results, our method only requires about $4\times$ the size of the image memory, while [1] requires $256\times$ the size of the image memory for computing and storing integral histogram (C implementation of our method is provided at the author’s homepage).

6. Extends the $O(1)$ framework for cross/joint bilateral filtering and median filtering.

The effectiveness of the proposed method is then experimentally verified for a variety of applications including natural video conferencing, interactive filtering, image/video abstraction, highlight removal, and multi-focus.

2. $O(1)$ Bilateral Filtering with Arbitrary Spatial and Arbitrary Range Kernels

A bilateral filter generally contains a spatial and a range filter kernel. Denote \mathbf{x} as a pixel in the image and \mathbf{y} as a pixel in the neighborhood $N(\mathbf{x})$ of \mathbf{x} , $I(\mathbf{x})$ and $I(\mathbf{y})$ as the corresponding range values of pixel \mathbf{x} and \mathbf{y} . The filtered range value of \mathbf{x} is

$$I^B(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in N(\mathbf{x})} (f_S(\mathbf{x}, \mathbf{y}) \cdot f_R(I(\mathbf{x}), I(\mathbf{y})) \cdot I(\mathbf{y}))}{\sum_{\mathbf{y} \in N(\mathbf{x})} (f_S(\mathbf{x}, \mathbf{y}) \cdot f_R(I(\mathbf{x}), I(\mathbf{y})))}, \quad (1)$$

where f_S and f_R are the spatial and range filter kernels, respectively. If the range function is computed based on another image D where the range values of pixel \mathbf{x} and \mathbf{y} are $D(\mathbf{x})$ and $D(\mathbf{y})$, the spatial filtered range value of pixel \mathbf{x} for image I is

$$I_D^B(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in N(\mathbf{x})} (f_S(\mathbf{x}, \mathbf{y}) \cdot f_R(D(\mathbf{x}), D(\mathbf{y})) \cdot I(\mathbf{y}))}{\sum_{\mathbf{y} \in N(\mathbf{x})} (f_S(\mathbf{x}, \mathbf{y}) \cdot f_R(D(\mathbf{x}), D(\mathbf{y})))}, \quad (2)$$

and the resulting filter is called a cross (or joint) bilateral filter [10, 24], which enforces the texture of filtered image I_D^B to be similar to image D .

2.1. Decomposing a Bilateral Filter into Spatial Filters

We review Durand and Dorsey’s piecewise-linear bilateral filtering method in this section and show that it can be directly extended for $O(1)$ bilateral filtering with arbitrary spatial and arbitrary range kernels.

In practice, the pixel intensity for an image $I(\mathbf{x})$ is discrete with $I(\mathbf{x}) \in \{0, \dots, N-1\}$, where N is the total number of grayscale values. Letting $I(\mathbf{x}) = k$, Equation (1) can be expressed as

$$I^B(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in N(\mathbf{x})} (f_S(\mathbf{x}, \mathbf{y}) \cdot f_R(k, I(\mathbf{y})) \cdot I(\mathbf{y}))}{\sum_{\mathbf{y} \in N(\mathbf{x})} (f_S(\mathbf{x}, \mathbf{y}) \cdot f_R(k, I(\mathbf{y})))}. \quad (3)$$

For every pixel \mathbf{y} and every intensity value $k \in \{0, \dots, N-1\}$, define

$$W_k(\mathbf{y}) = f_R(k, I(\mathbf{y})) \quad (4)$$

and

$$J_k(\mathbf{y}) = W_k(\mathbf{y}) \cdot I(\mathbf{y}). \quad (5)$$

Bilateral filtering can then be decomposed into N sets of linear filter responses

$$J_k^B(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in N(\mathbf{x})} f_S(\mathbf{x}, \mathbf{y}) J_k(\mathbf{y})}{\sum_{\mathbf{y} \in N(\mathbf{x})} f_S(\mathbf{x}, \mathbf{y}) W_k(\mathbf{y})} \quad (6)$$

so that

$$I^B(\mathbf{x}) = J_{I(\mathbf{x})}^B(\mathbf{x}), \quad (7)$$

where J_k^B is defined as Principle Bilateral Filtered Image Component (PBFIC) in this paper. In practice, assume only \hat{N} out of N PBFIC ($k \in \{L_0, \dots, L_{\hat{N}-1}\}$) are used, and the intensity of pixel \mathbf{x} is $I(\mathbf{x}) \in [L_k, L_{k+1}]$, the bilateral filtering value $I^B(\mathbf{x})$ can then be linearly interpolated from $J_k^B(\mathbf{x})$ and $J_{k+1}^B(\mathbf{x})$ as follows:

$$I^B(\mathbf{x}) = (L_{k+1} - I(\mathbf{x})) J_k^B(\mathbf{x}) + (I(\mathbf{x}) - L_k) J_{k+1}^B(\mathbf{x}). \quad (8)$$

Note that, the range filter f_R is not constrained and any desired filter function can be chosen, but approximation can be poor if \hat{N} is extremely small for some range filters, *e.g.*, Gaussian filter.

According to Equation (4) and (5), the noise due to quantization only affect the range function W_k , and the pixel intensity values of the input image ($I(\mathbf{y})$ in Equation 5) will be preserved. However, both are quantized in the $O(1)$ bilateral filtering method presented in [1], thus less precise. The main computation of the method is $\hat{N} \times 2$ spatial filtering processing according to Eqn. (6). Additionally, in our method, image pixels are processed independently, allowing for parallel implementation. These are the two main reasons why our method outperforms the current state-of-the-art [1] for both accuracy and speed. The main storage

required is three memory buffers with the same size as the input image for images J_k^B , J_{k+1}^B and W_k . Note that J_k^B and J_{k+1}^B share the same memory buffer. Box/Gaussian spatial filtering also require an additional memory buffer with the same size as the input image. Hence, the total memory buffer is about $4 \times$ the size of the image memory. However, [1] requires a set of \hat{N} image buffers to store the integral histogram during aggregation. Otherwise, the program will compute the integral histogram \hat{N} times at the cost of speed.

2.2. $O(1)$ Spatial Filtering

As shown in Equation (6), bilateral filter with arbitrary spatial and range kernels can be decomposed into two sets of spatial filters on $J_k(\mathbf{y})$ and $W_k(\mathbf{y})$, respectively. The computation complexity thus depends on complexity of spatial filtering. Enabling constant time spatial filtering results in constant time bilateral filtering with arbitrary range functions.

One of the most popular spatial filter is box filter which can be easily computed in constant time using integral image [22] or summed area table [25]. Another popular spatial filter is Gaussian filter which if implemented in the Fourier domain is constant in the filter size. However, the discrete FFT and its inverse have cost $O(\log r)$, where r is spatial filter size. Hence, to achieve higher speed, we used Deriche’s recursive method [23] to approximate Gaussian filtering which is able to run in constant time and the results are visually very similar to the exact. Polynomial filtering can also be computed in constant time $O(1)$ using a set of integral images [1]. More $O(1)$ spatial filters are presented in [1].

2.3. $O(1)$ Cross/Joint Bilateral Filtering and Median Filtering

In this section, we show that the decomposition method used for bilateral filtering can be easily extended for cross/joint bilateral filtering and median filtering. Changing $I(\mathbf{y})$ in Equation (4) to $D(\mathbf{y})$ and changing $I(\mathbf{x})$ in Equation (7) and (8) to $D(\mathbf{x})$ enables $O(1)$ cross/joint bilateral filtering, which enforces the texture of filtered image to be similar to image D .

For median filtering, assume

$$\text{sign}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

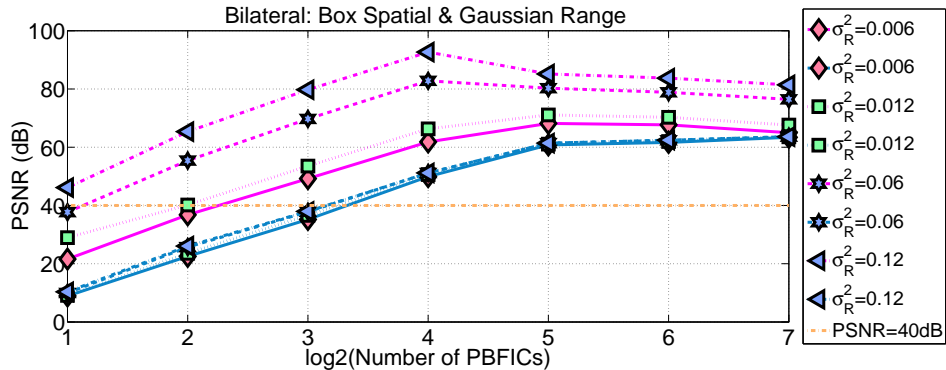
and the intensity value is also in $[0, \dots, N-1]$, the median filtered value $I^M(\mathbf{x})$ of a pixel \mathbf{x} can be expressed as:

$$M_k(\mathbf{x}) = \left| \sum_{\mathbf{y} \in N(\mathbf{x})} \text{sign}(k - I(\mathbf{y})) \right| = \left| \sum_{\mathbf{y} \in N(\mathbf{x})} Q_k(\mathbf{y}) \right|, \quad (9)$$

$$I^M(\mathbf{x}) = \underset{k \in \{0, 1, \dots, N-1\}}{\text{argmin}} M_k(\mathbf{x}), \quad (10)$$



(a) Original image.



(b) PSNR accuracy.

Figure 2. Effect of quantization on image quality. (a). Original image (480×640). (b). PSNR accuracy with respect to the number of PBFICs (bins) used. Colors indicate different methods. The pink and blue curves are the performance of our method and Porikli’s method [1], respectively. Bilateral filtering with Box spatial and Gaussian range kernels is used. The spatial filter size is 21×21 . As visible, our method outperforms Porikli’s method [1] for both small and large range kernel variances. Some of the corresponding filtered images are provided in Figure 3 for visual comparison.



(a) Porikli’s method.



(b) Our method.



(c) Exact.



(d) Porikli’s method.



(e) Our method.



(f) Exact.

Figure 3. Performance under extreme quantization. Bilateral filtering with box spatial and Gaussian range kernels is used. (a), (b) and (c) are filtered images with Gaussian range variance $\sigma_R^2 = 0.012$ (8-bit grayscale images: $\sigma_R^2 = 28 \times 28$). (a) and (b) are obtained using 4 bins (PBFICs), and (c) is the exact result (256 PBFICs). Apparently, Porikli’s methods is invalid for low bins. (d), (e) and (f) are filtered images with $\sigma_R^2 = 0.12$ (8-bit grayscale images: 88×88). (d) and (e) are obtained using 2 bins (PBFICs), and (f) is the exact result. (d) shows that Porikli’s method is invalid, in which the input image is represented using a 2-bin integral histogram (either 0 or 255). However, our result as shown in (e) is still visually very similar to the exact.

which shows that median filtering can be separated into two steps: (1) Box filtering is applied to a set of N images Q_k computed based on current intensity possibility k and the original image I . The absolute value of the box filtering results is computed as images M_k . (2) For each pixel, the intensity hypothesis $k \in \{0, 1, \dots, N - 1\}$ corresponding to the minimum pixel values of the images M_k is selected as correct. The box filtering in the first step depends on the filter size, but as shown in Section 2.2, it can be computed in constant time.

3. Experimental Results

This section reports experimental results supporting our claims that our method advanced the state-of-the-art. Same as [1], we use the peak signal-to-noise ratio (PSNR) to eval-

uate numerical accuracy. For two intensity images $I, J \in [0, 1]$, this ratio is defined as $10 \log_{10}((h \cdot w) / \sum_{\mathbf{x}} |I(\mathbf{x}) - J(\mathbf{x})|^2)$, where h and w are the height and width of image I and J , and \mathbf{x} is one of the pixels. It is assumed [19] the PSNR values above 40dB often corresponds to almost invisible differences.

Using Figure 2(a) as input image, the PSNR values for bilateral filtering with Box spatial and Gaussian range ($f_R(I(\mathbf{x}), I(\mathbf{y})) = \exp(-\frac{(I(\mathbf{x}) - I(\mathbf{y}))^2}{2\sigma_R^2}) / \sqrt{2\pi\sigma_R^2}$) kernels are presented in Figure 2(b) with respected to the number of PBFICs (bins) used. Figure 2 shows that our method is more accurate than Porikli’s method [1] using both small and large range kernel variances. Also note that for our method, larger range variance results in much higher accuracy with lower number of PBFICs, as the range function (Equation (4)) is more flat, and quantization introduces

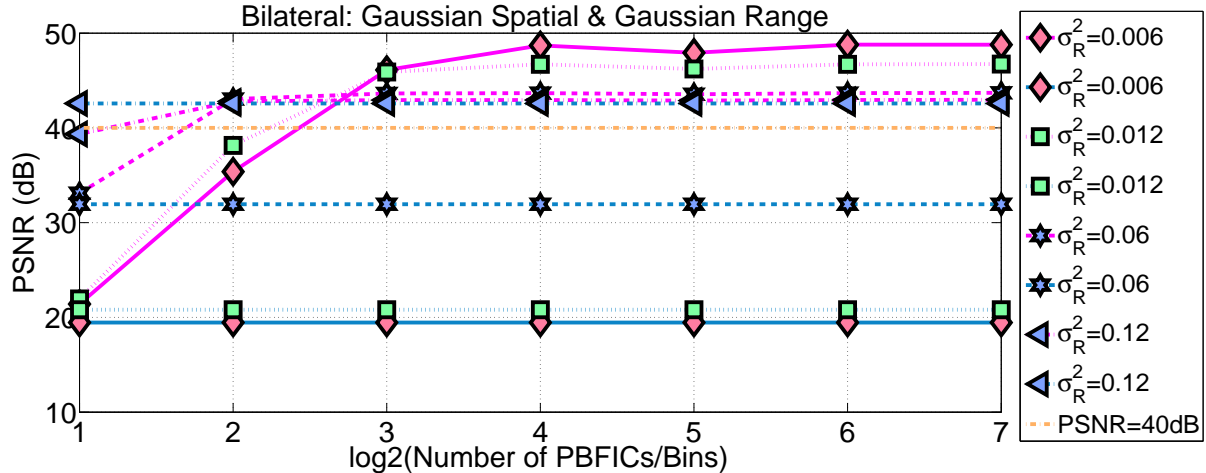


Figure 4. PSNR accuracy with respect to the number of PBFICs (bins) used. Colors indicate different methods. The pink and blue curves are the performance of our method and Porikli’s method [1], respectively. Bilateral filtering with Gaussian spatial and Gaussian range kernels is used. The spatial filter size is the same as the image size (480×640). For Porikli’s method, Taylor expansion up to third order derivatives are used to approximate the Gaussian range function. It is valid only when σ_R is large. Our method, on the other hand, works fine for both large and small σ_R . The corresponding filtered images are provided in Figure 5 for visual comparison.

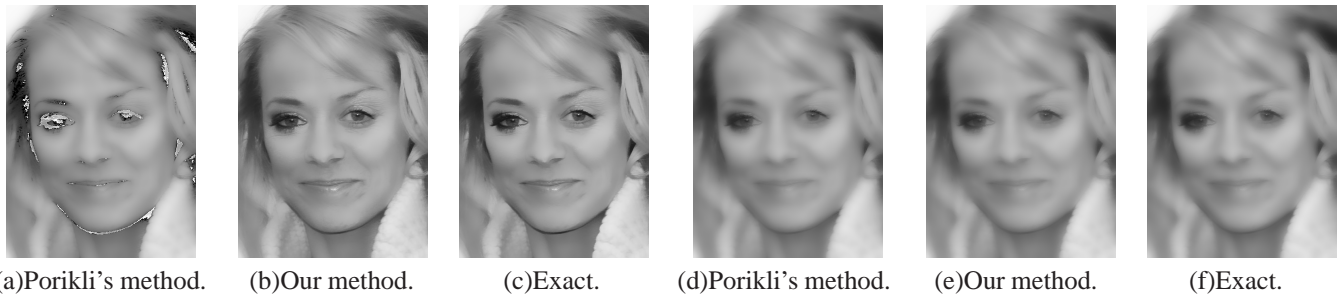


Figure 5. Bilateral filtering with Gaussian spatial and Gaussian range kernels. (a), (b) and (c) are filtered image with $\sigma_R^2 = 0.012$ (8-bit grayscale images: 28×28). (a) is obtained using Porikli’s method (recursive Gaussian spatial and Taylor expansion approximated Gaussian range), (b) is our result (recursive Gaussian spatial and exact Gaussian range) using 4 PBFICs, and (c) is the exact result (exact Gaussian spatial and exact Gaussian range). Apparently, Porikli’s method is invalid for small range variances. (d), (e) and (f) are filtered image with $\sigma_R^2 = 0.12$ (8-bit grayscale images: 88×88). (d) is obtained using Porikli’s method, (e) is our result using 4 PBFICs, and (f) is the exact result. (d) and (e) are visually very similar to the exact filter responses as presented in (f), which shows that both methods are good for large range variances. However, most of the applications based on bilateral filtering require low range variance for edge preserving.

less noise. However, Porikli’s method also quantizes the original image. The improvement is thus much smaller. The filtered images with $\sigma_R^2 = 0.012$ and $\sigma_R^2 = 0.12$ using Porikli’s method and our method are provided in Figure 3. As visible, our results are visually very similar to the exact even using very small number of PBFICs. To achieve acceptable PSNR value ($> 40\text{dB}$) for variance $\sigma_R^2 \in [0.006, 0.12]$, our method generally requires 2 to 8 PBFICs, and the running time is about 3.7 ms to 15 ms for 1MB image. To achieve acceptable quality, Porikli’s method requires 16-bin integral histogram, and the total running time for constructing the integral histogram and

computing the response for any given spatial filter size is about 75 ms, thus our method is about $10 \times$ faster on average. Also, our method is less memory consuming. Only twice the memory of the original image is required by our method regardless of the number of PBFICs used. However, to obtain the same quality as exact, Porikli’s method computes integral histogram using a total of 256 bins for 8-bit grayscale image. Hence, $256 \times$ the image memory is required. The heavy memory consuming can be avoided by repeatedly computing the integral histogram for every possible intensity value but at the cost of speed.

Figure 4 presents the PSNR values of bilat-

eral filtering with Gaussian spatial ($f_S(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x}, \mathbf{y}\|^2}{2\sigma_S^2})/\sqrt{2\pi\sigma_S^2}$) and Gaussian range kernels for the original image presented in Figure 2(a). Range variances ranging from 0.006 to 0.12 are tested. Obviously, Porikli’s method (blue lines) fails for small range variances due to Gaussian approximation using Taylor series expansion. However, our method (pink curves) is valid for both small and large range variances. The corresponding filtered images are provided in Figure 5. For a typical 1 MB image, Porikli’s method runs at about 0.35 second. Our GPU implementation runs at about 30 frames per second using 8 PBFICs (Computation complexity of Recursive Gaussian filtering is about twice the box filtering), which is above the acceptable threshold (> 40 dB) as shown in Figure 4. Hence, our method is about $10\times$ faster than Porikli’s method. The memory requirement is similar for both methods since neither of them depends on the number of PBFICs (bins) used.

Finally, experimental results on cross/joint bilateral filtering and median filtering are presented in Figure 6 and Figure 7.

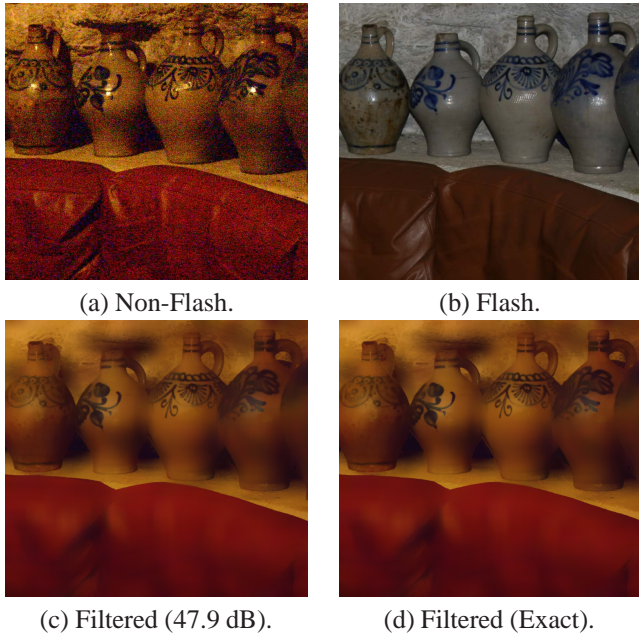


Figure 6. Joint Bilateral filtering with box spatial (filter size: 51×51) and Gaussian range ($\sigma_R^2 = 0.006$) kernels. (a) is non-flash image (resolution: 774×706) and (b) is the flash image used to guide the smoothing. (c) and (d) are the filtered images using 8 and 256 PBFICs, respectively.

4. Applications

In this section we demonstrate the usefulness of the constant time bilateral filtering operation for a variety of appli-



Figure 7. $O(1)$ median filtering. (a) is the original image and (b) is our median filtering result which is the same as exact.

cations.

4.1. Natural Video Conferencing

In modern video conference system, *e.g.*, Halo [26], the existing product line uses very high quality SD cameras that deliver natural, life-like images of meeting participants. With HD cameras and displays, even as they deliver additional sharpness, many unwanted details like wrinkles may also be amplified, resulting in images that may not be as pleasing as the SD images in today’s product line. The constant time bilateral filtering method proposed provide a way to retain the salient features in HD images while removing unwanted details and noise. Figure 8 shows the result of applying our $O(1)$ bilateral filter to a portrait. The filtered result in Figure 8 (b) clearly preserved salient features while removing wrinkles. Linearly blending the original image with the filtered result produces Figure 8 (c), which is natural and realistic. The amount of blending can also be controlled in real-time to deliver the most desirable output. Video demo is presented in the supplemental materials.

4.2. Interactive Filtering

As shown in Figure 8 (b), full automatic/global bilateral filtering removes unwanted details, *e.g.*, wrinkles. Unfortunately, some interesting details, *e.g.*, hair, are lost. A human-guided local bilateral filtering method is then developed. User is provided with a virtual brush. Filtering is applied locally to the regions where the brush passes by. The edge-preserving property guarantees a natural looking after local filtering and the real-time speed enables human-compute interaction. Intermediate results are presented in Figure 9 (a) and (b), and the final result in Figure 9 (c). Video demo is presented in the supplemental materials.

4.3. Other applications

Bilateral filtering and cross/joint bilateral filtering can also be used for image/video abstraction [12], highlight



Figure 8. Natural video conferencing. (a) is original image (512×683), (b) is filtered image and (c) is the blend of original and filtered versions. Note: the reader is urged to view these images at full size on a video display, for details may be lost in hard copy.

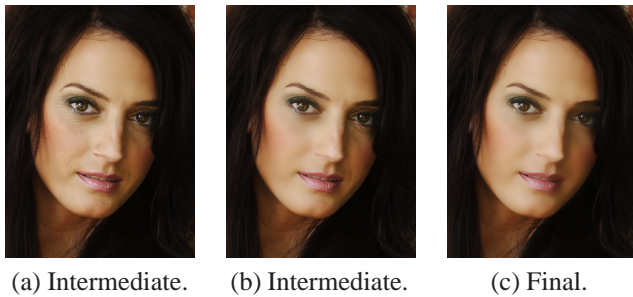


Figure 9. Local bilateral filtering. (a) and (b) are intermediate filtering results. Filtering is only applied to forehead in (a), and also the left cheek in (b). (c) is the final result with filtering applied to the whole face. Note that hair details lost in Figure 8 (b) are preserved, and the unwanted wrinkles are completely removed.

removal and multi-focus. Experimental results using our $O(1)$ bilateral filtering method are presented in Figure 10, 11 and 12, respectively.

5. Conclusion

A uniform framework enabling real time $O(1)$ bilateral filtering with *arbitrary spatial and arbitrary range* kernels and parallel implementation is presented in the paper. Experimental results show that our method outperforms the state-of-the-art [1] for accuracy, speed and memory consuming. For bilateral filter with arbitrary spatial and Gaussian range kernels, our method works for both small and large range variances, while the method presented in [1] is invalid for small variances due to Taylor series approximation. Our framework can be easily extended for $O(1)$ cross/joint bilateral filtering and median filtering. Experimental results on a variety of applications verify the effectiveness of the proposed method.

In the future, we are planning to implement the proposed method with NVIDIA Geforce 9800GX2 GPU which has twice the texture fill rate as the 8800GTX GPU used in the



Figure 10. Image/Video abstraction. Note: the reader is urged to view these images at full size on a video display, for details may be lost in hard copy.

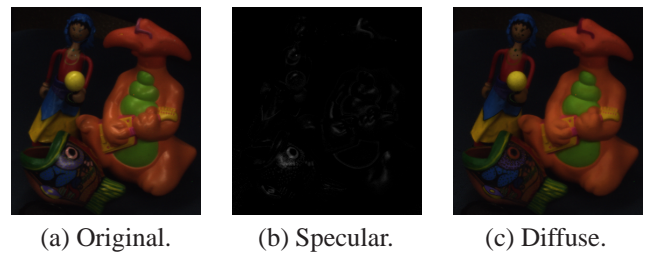


Figure 11. Highlight removal. (a) Original image. (b) Separated specular component. (c) Separated diffuse component.

paper, and has the potential to double the speed reported. Meanwhile, [9] demonstrated that it is safe to use a down-sampled version of the image except for the final interpolation (Eqn. 8). There is not any visible artifact up to down-sampling factor of 10 to 25. We are planning to implement this method which has the potential to improve the speed more than $10\times$.

References

- [1] Porikli, F.: Constant time $o(1)$ bilateral filtering. In: CVPR. (2008) 1, 2, 3, 4, 5, 7
- [2] Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: ICCV. (1998) 839–846 1
- [3] Buades, A., Coll, B., Morel, J.M.: A review of image denoising algorithms, with a new one. In: Multiscale Modeling and Simulation. Volume 4. (2005) 490–530 1



Figure 12. Multi-focus. (a), (b) and (c) are multi-focus images, (d) is the fusion result using the proposed $O(1)$ bilateral filtering method.

- [4] Yang, Q., Yang, R., Davis, J., Nistér, D.: Spatial-depth super resolution for range images. In: CVPR. (2007) **1**
- [5] Wong, W.C.K., Chung, A.C.S., Yu, S.C.H.: Trilateral filtering for biomedical images. In: International Symposium on Biomedical Imaging. (2004) **1**
- [6] Bennett, E.P., McMillan, L.: Video enhancement using per-pixel virtual exposures. In: Siggraph. Volume 24. (2005) 845–852 **1**
- [7] Bennett, E.P., Mason, J.L., McMillan, L.: Multispectral bilateral video fusion. In: Transactions on Image Processin. Volume 16. (2007) 1185–1194 **1**
- [8] Oh, B.M., Chen, M., Dorsey, J., Durand, F.: Image-based modeling and photo editing. In: Siggraph. (2001) **1**
- [9] Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. In: Siggraph. Volume 21. (2002) **1, 7**
- [10] Petschnigg, G., Agrawala, M., Hoppe, H., Szeliski, R., Cohen, M., Toyama, K.: Digital photography with flash and no-flash image pairs. In: Siggraph. Volume 23. (2004) **1, 2**
- [11] Ramanath, R., Snyder, W.E.: Adaptive demosaicking. In: Journal of Electronic Imaging. Volume 12. (2003) 633–642 **1**
- [12] Winnemoller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. In: Siggraph. Volume 25. (2006) 1221–1226 **1, 6**
- [13] Xiao, J., Cheng, H., Sawhney, H., Rao, C., Isnardi, M.: Bilateral filtering-based optical flow estimation with occlusion detection. In: ECCV. (2006) **1**
- [14] Sand, P., Teller, S.: Particle video: Long-range motion estimation using point trajectories. In: ECCV. (2006) **1**
- [15] Yang, Q., Wang, L., Yang, R., Stewénus, H., Nistér, D.: Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. PAMI (2008) **1**
- [16] Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. PAMI **28** (2006) 650–656 **1**
- [17] Elad, M.: On the bilateral filter and ways to improve it. IEEE Transactions On Image Processing **11** (2002) 1141–1151 **1**
- [18] Pham, T.Q., van Vliet, L.J.: Separable bilateral filtering for fast video preprocessing. In: International Conference on Multimedia and Expo. (2005) **1**
- [19] Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. In: ECCV. (2006) **1, 4**
- [20] Weiss, B.: Fast median and bilateral filtering. In: Siggraph. Volume 25. (2006) 519–526 **1**
- [21] Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. In: Siggraph. Volume 26. (2007) **1**
- [22] Viola, P., Jones, M.: Robust real-time face detection. In: ICCV. (2001) 747–750 **2, 3**
- [23] Deriche, R.: Recursively implementing the gaussian and its derivatives. In: ICIP. (1992) 263–267 **2, 3**
- [24] Eisemann, E., Durand, F.: Flash photography enhancement via intrinsic relighting. Siggraph **23** (2004) 673–678 **2**
- [25] Crow, F.: Summed-area tables for texture mapping. In: Siggraph. (1984) **3**
- [26] Hewlett-Packard: (Hp halo telepresence and video conferencing solutions) <http://www.hp.com/halo>. **6**