O(1) BILATERAL FILTERING WITH LOW MEMORY USAGE

Masaki Igarashi, Masayuki Ikebe, Sousuke Shimoyama, Kenta Yamano, Junichi Motohisa

Graduate School of Information Science and Technology
Hokkaido University, Sapporo, Japan

ABSTRACT

We propose a $O(1)$ algorithm for bilateral filter with low memory usage. Bilateral filter can be converted into weighted histogram operation. Applying line buffers of column histograms, we can reduce the number of calculation needed to construct recursive center-weighted local histogram. Also our method have advantage in terms of memory requirements. We used a 2-GHz CPU with our method and achieved one million pixels per 0.5 sec operation and high PSNR over 40 dB without the need for temporary frame buffers or additional instructions (downsampling, SIMD instructions, or multi-thread operations).

Index Terms— Bilateral filter, constant time algorithm

1. INTRODUCTION

Tomasi et al. proposed a bilateral filter, which is an edge preserving smoothing filter [1]. It has been used for several applications such as noise reduction, dynamic range compression [2], and estimation of illuminance in Retinex. The output images are obtained by using weight averaging in the spatial and range (intensity) domains.

Until recently, the bilateral filter was computationally expensive because brute-force implementation has a $O(r^2)$ computational cost per output pixel (r: filter radius). Weiss developed the $O(\log\, r)$ algorithm by more efficiently calculating the local histogram. However, the processing speed depends on the filter radius. Moreover, the spatial weight of this method is limited to the constant weight not the center-weighted kernel. Porikli proposed the $O(1)$ bilateral filter using an integral histogram [3]. Yang et al. also proposed a $O(1)$ algorithm [4]. These methods require several times the size of image memory.

In this paper, we describe a new $O(1)$ bilateral filtering method that uses less memory space. Our method requires only several line buffers.

The authors would like to extend their appreciation to Kensuke Yamaoka and Yuji Osumi of Dai Nippon Printing Co.,Ltd. for their helpful guidance and insightful discussions regarding the development of a processing system in this research.

2. BILATERAL FILTERING

A bilateral filter contains a spatial and a range kernel. Let $\boldsymbol{p}$ denote a pixel in the image, $K(\boldsymbol{p})$ be a set of pixels in the kernel of $\boldsymbol{p}$, $I_{\boldsymbol{p}}$ be the intensity of pixel $\boldsymbol{p}$. Bilateral filter updates intensity value as follows:

$$I_{\boldsymbol{p}}^{new} = \frac{\sum_{\boldsymbol{p'}\in K(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{p'}) w_r(I_{\boldsymbol{p}}, I_{\boldsymbol{p'}}) I_{\boldsymbol{p'}}}{\sum_{\boldsymbol{p'}\in K(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{p'}) w_r(I_{\boldsymbol{p}}, I_{\boldsymbol{p'}})} \tag{1}$$

where w_s and w_r are the weighting functions in the spatial and range domains respectively. Usually, a constant and a center-weighted function are used for w_s and the Gaussian weighting function is used as w_r.

2.1. Conversion of bilateral filter into weighted histogram operation

In this section, we convert a bilateral filter into a weighted histogram operation. Let B_{max} be the number of bins in the histogram. Generally, a histogram $H_{\boldsymbol{p}}$ from kernel $K(\boldsymbol{p})$ is given as follows:

$$b \in \{0, 1, \cdots, B_{max}-1\} \tag{2}$$

$$B_b = [\frac{bI_{max}}{B_{max}}, \frac{(b+1)I_{max}}{B_{max}}) \tag{3}$$

$$H_{\boldsymbol{p}}(b) = \sum_{\boldsymbol{p'}\in K(\boldsymbol{p})} f_b(I_{\boldsymbol{p'}}) \tag{4}$$

where $H_{\boldsymbol{p}}(b)$ is the b-th bin of the histogram and f_b is defined as $f_b(x) = \textbf{if } x \in B_b \textbf{ then } 1 \textbf{ else } 0$.

Here, we define the weighted histogram H^w as follows:

$$H_{\boldsymbol{p}}^w(b) = \sum_{\boldsymbol{p'}\in K(\boldsymbol{p})} f_b(I_{\boldsymbol{p'}}) w_s(\boldsymbol{p},\boldsymbol{p'}) \tag{5}$$

Unlike the standard histogram, a weighted histogram stores spatial weight w_s instead of the number of pixels for each tone value. So we can rewrite Eq(1) for a bilateral filter in the following way.

$$I_{\boldsymbol{p}}^{new} = \frac{\sum_b H_{\boldsymbol{p}}^w(b) w_r(I_{\boldsymbol{p}}, \frac{bI_{max}}{B_{max}}) \frac{bI_{max}}{B_{max}}}{\sum_b H_{\boldsymbol{p}}^w(b) w_r(I_{\boldsymbol{p}}, \frac{bI_{max}}{B_{max}})} \tag{6}$$

This means that we create a $O(1)$ bilateral filtering if the weighted histogram can be calculated in constant time.

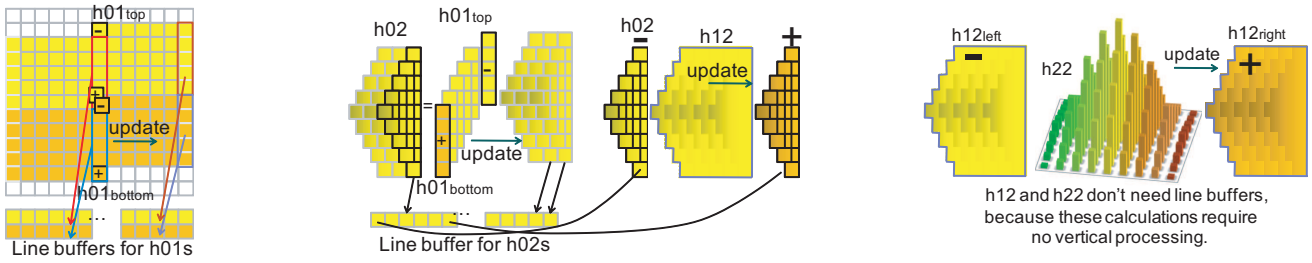


Fig. 1. Composition of recursive histogram $h22$: $H^{R<2,2>}$

2.2. O(1) Bilateral Filtering with Constant Spatial Weight

We are currently researching a 2D bilateral filtering in which the spatial filter has a constant weight (which is called box spatial filter). Given that the shape of a kernel is rectangular and pixel $\boldsymbol{p} = (x, y)$, the kernel is

$$K(x, y) = \{(x + i, y + j) \mid -r \leq i, j \leq r\} \tag{7}$$

where r is the radius of the filter.

Assume that w_s is constant, the weighted histogram H^w is the equal to standard histogram H. The standard histogram can be easily computed in constant time [5].

2.3. O(1) Bilateral Filtering with center-weighted local histograms

Here, we define recursive histogram H^R as follows:

$$H_{x,y}^{R<0,0>}(b) = f_b(I_{x,y}) \tag{8}$$

$$H_{x,y}^{R<0,n>}(b) = \sum_{i=-r}^{r} H_{x,y+i}^{R<0,n-1>}(b) \tag{9}$$

$$H_{x,y}^{R<m,n>}(b) = \sum_{i=-r}^{r} H_{x+i,y}^{R<m-1,n>}(b) \tag{10}$$

where c_r is a normalization coefficient defined by $c_r = \frac{1}{2r+1}$. $H_{x,y}^{R<1,1>}$ is the standard histogram. We can take $H_{x,y}^{R<m,n>}$ as the repetition of the moving sum for each bins of the histogram. As m and n increase, the weight of the recursive histogram approaches the Gaussian distribution centered at (x, y) according to central limit theorem.

When we take into consideration the overlap region between adjacent kernels, we can rewrite Eqs. 8-10 as follows:

$$\begin{aligned} H_{x,y}^{R<0,n>}(b) = {} & H_{x,y-1}^{R<0,n>}(b) - H_{x,y-r-1}^{R<0,n-1>}(b) \\ & + H_{x,y+r}^{R<0,n-1>}(b) \end{aligned} \tag{11}$$

$$\begin{aligned} H_{x,y}^{R<m,n>}(b) = {} & H_{x-1,y}^{R<m,n>}(b) - H_{x-r-1,y}^{R<m-1,n>}(b) \\ & + H_{x+r,y}^{R<m-1,n>}(b) \end{aligned} \tag{12}$$

This means that we can obtain histogram $H_{x,y}^{R<m,n>}$ by recursively combining the pre-calculated histogram. In addition, we found that the amount of calculation necessary for the histogram is independent of the filter radius r. However, in the recursive histogram calculation, if x-direction processing of all rows are performed after y-direction processing of all columns, frame buffers are indispensable. Also, these processing is not suitable for scan line processing. Here, focusing on line buffers of column histograms, we propose the constant time recursive histogram calculation without frame buffers. We present a pseudo-code of the algorithm for a recursive histogram $h22$ ($H_{x,y}^{R<2,2>}$) calculation in algorithm 1. Then, we show a diagram about composition of $h22$ in Fig.1.

Algorithm 1 Calculate histogram $h22$ ($H^{R<2,2>}$) in constant time.

```
width: width of image.
h02[width]: histogram H^{R<0,2>} array
h01_top[width]: histogram H^{R<0,1>} array
h01_bottom[width]: histogram H^{R<0,1>} array
for i = 0 to height do
  if i==0 then
    initialize h02[j]
    initialize h01_top[j], h01_bottom[j]
  else
    for j = 0 to width do
      remove pixel(i − r − 1, j) from h01_top[j]
      add pixel(i − 1, j) to h01_top[j]
      remove pixel(i − 1, j) from h01_bottom[j]
      add pixel(i + r, j) to h01_bottom[j]
      h02[j] = h02[j] − h01_top[j] + h01_bottom[j]
    end for
  end if
  h22, h12_left, h12_right: histograms
  for j = 0 to width do
    if j == 0 then
      initialize h22, h12_left, h12_right
    else
      h12_left = h12_left − h02[j − r − 1] + h02[j − 1]
      h12_right = h12_right − h02[j − 1] + h02[j + r]
      h22 = h22 − h12_left + h12_right
    end if
  end for
end for
```

According to Eq.11, $H_{x,y}^{R<0,n>}$ is calculated easily by keeping the pre-calculated value of $H_{x,y-1}^{R<0,n>}$. Considering the moving direction of a filter window, we should keep $\{H_{0,y-1}^{R<0,n>}, \cdots, H_{width-1,y-1}^{R<0,n>}\}$. For $h22$ ($H^{R<2,2>}$) calculation, we prepared three line buffers. One is used for the $h02$ ($H^{R<0,2>}$) process. while the others are used for the $h01_{top}$ ($H_{top}^{R<0,1>}$) and $h01_{bottom}$ ($H_{bottom}^{R<0,1>}$) processes. The $h01$ line buffer stores the $h01$s of the previous line. The $h01_{top}$ and the $h01_{bottom}$ are updated by adding and subtracting pixels (Fig.1 left). Because the $h02$s of previous line have been stored to line buffer, the $h02$ can be updated by adding/subtracting the $h01_{top}$ and the $h01_{bottom}$ (Fig.1 center). Updated column histograms($h01$ and $h02$) are stored in the line buffers again.

The $h12$ can be updated by adding and subtracting the pre-calculated $h02$ in the line buffer. Adding and subtracting the $h12_{left}$ and the $h12_{right}$ to/from the $h22$ enables us to obtain the updated $h22$ (Fig.1 center, right). These processes are independent of the filter kernel size and require no frame buffers.

2.3.1. Memory usage of proposed method

Table 2 shows the comparison result of the memory usage between proposed method and two conventional methods reported in [4]. Even though the box spatial filter is used for 8-bit grayscale images, these conventional methods requires a large amount of memory which is proportional to image size (number of pixels). For box and center-weighted filter, our method uses only line buffers of histograms and several temporary histograms, therefore it requires low amount of memory which is proportional to image width not height.

3. EXPERIMENTAL RESULTS

We tested the proposed bilateral filter method and evaluated the processing time and image quality. We used an Athlon Dual Core CPU (2 GHz). We did not use downsampling, multithread operations, or SIMD instructions.

The computation times are given in Fig. 2. We found that the processing time of the proposed method is independent of the filter radius. In a method similar to Durand's method [2], we can optimize weighting in the range domain. By reducing the number of bins, the weighted histogram can be calculated more quickly.

We analyzed the filter accuracy by performing filtering operation and by calculating the PSNR. For the analysis, the filtering image with exact box and gaussian spatial weight were set to the source images. Table. 1 shows PSNR of bilateral filters with center weighted kernel (using histogram $H^{R<n,n>}$). For $H^{R<n,n>}$ calculation, $2^n - 1$ line buffers are required in our method. We found the histograms $n = 2, 3$ provide enough PSNR values (over 40 dB).

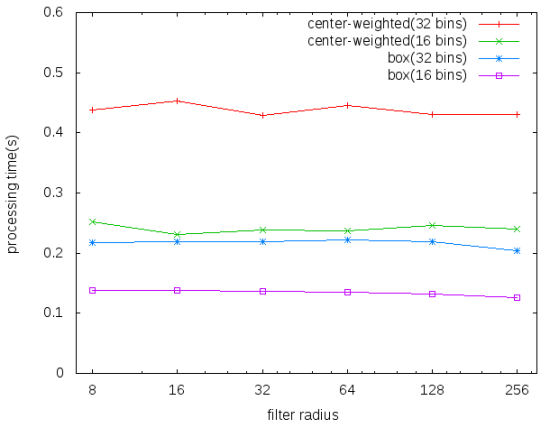


Fig. 2. Processing time of our methods (image size: 800×600, $\sigma_r^2 = 0.016$).

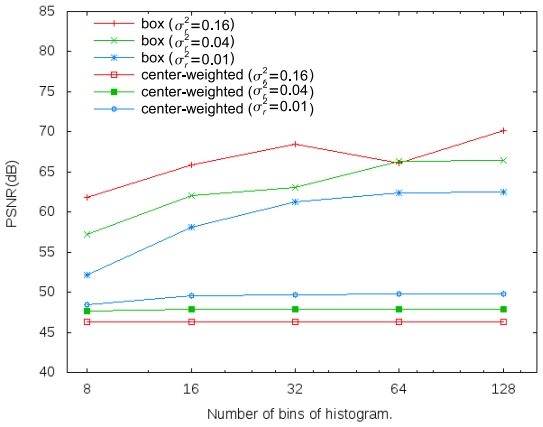


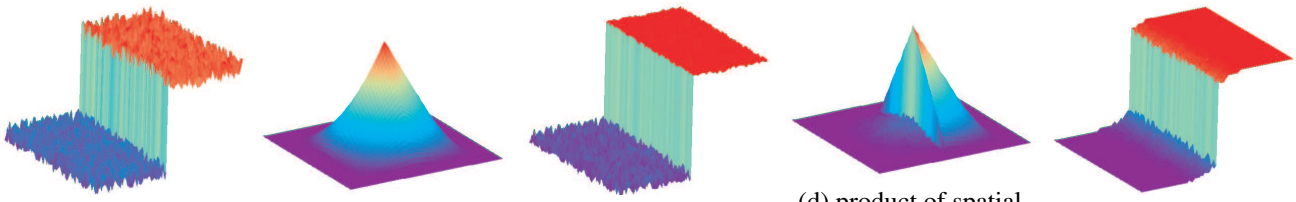
Fig. 3. PSNR accuracy of proposed bilateral filter (image size: 800×600, filter radius: 5)

.

Figure 3 shows PSNR graphs using proposed bilateral filters with box and center-weighted kernel (using histogram $H^{R<2,2>}$) and exact box and gaussian kernel. As visible, even the low bins of histogram provide high PSNR values (over 45 dB). Figure 4 shows the kernel form of our bilateral filter with $H^{R<2,2>}$. An adaptive kernel is formed around the edge of the image, and preserving the edges smoothes out the output image. Figures 5(a)-(d) show the result of applying the proposed method to an image. We cannot distinguish between the resulting images obtained by proposed method and exact images. We found that the proposed method achieved smoothing, while preserving the edge.

4. CONCLUSION

We proposed a $O(1)$ (constant time) bilateral filtering method with low memory usage by applying line buffers of column



(a) input image (b) spatial weight (c) intensity weight (d) product of spatial and intensity weight (e) output image

Fig. 4. Kernel form of proposed bilateral filter using $H^{R<2,2>}$ and its image processing

Table 1. PSNR[dB] dependent on r and m (r in Eqs.8-10, m of $H^{R<m,m>}$).

(a) 256-bin histogram

		r				
		1	3	7	15	31
m	2	54.8	57.0	56.8	56.5	55.2
	3	60.6	61.3	61.0	60.8	60.2
	4	58.1	61.0	60.2	60.2	58.3
	5	56.1	61.3	60.0	59.8	57.9

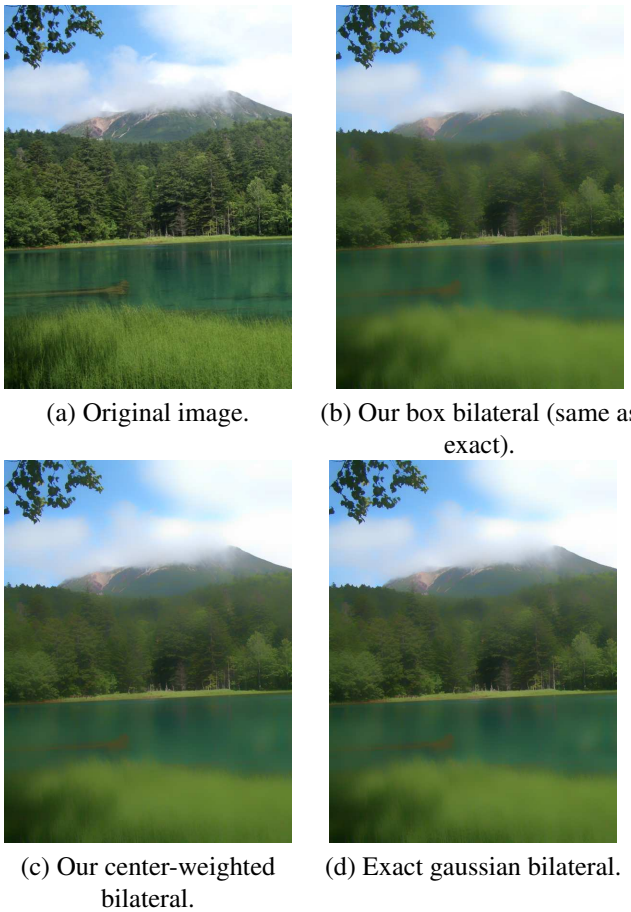
(b) 8-bin histogram

		r				
		1	3	7	15	31
m	2	50.2	48.4	46.8	45.2	43.7
	3	50.6	48.3	46.6	45.0	43.4
	4	49.9	48.0	46.3	44.7	43.1
	5	49.3	47.7	46.0	44.5	42.8

histograms. We used a 2-GHz CPU with our method and achieved one million pixels per 0.5 sec operation and high PSNR over 40 dB without the need for temporary frame buffers or additional instructions (downsampling, SIMD instructions, or multi-thread operations).

Table 2. Comparison of memory usage for bilateral filtering. B is the number of bins in the histogram. W and H are the image width and height respectively.

method	Porikli [3]	Yang [4]	proposed
box	$B \times W \times H$	$4 \times W \times H$	$B \times W + B$
$H^{R<2,2>}$			$3 \times (B \times W + B)$



(a) Original image. (b) Our box bilateral (same as exact). (c) Our center-weighted bilateral. (d) Exact gaussian bilateral.

Fig. 5. Bilateral filtering with box and center-weighted spatial kernels (size: 600x800).

5. REFERENCES

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV '98*, 1998, pp. 839–846.

[2] Frédo Durand and Julie Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," in *SIGGRAPH '02*. 2002, pp. 257–266, ACM.

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

[4] N. Qingxiong Yang, Kar-Han Tan Ahuja, "Real-time o(1) bilateral filtering," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2009*, 2009, pp. 557–564.

[5] Perreault S. and Hebert P., "Median filtering in constant time," *Image Processing, IEEE Transactions on*, vol. 16, pp. 2389–2394, 2007.