<table>
<thead>
<tr>
<th>Title</th>
<th>O(1) bilateral filtering with low memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Igarashi, Masaki; Ikebe, Masayuki; Shimoyama, Sousuke; Yamano, Kenta; Motohisa, Junichi</td>
</tr>
<tr>
<td>Citation</td>
<td>2010 17th IEEE International Conference on Image Processing (ICIP): 3301-3304</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2010-09</td>
</tr>
<tr>
<td>Doc URL</td>
<td><a href="http://hdl.handle.net/2115/48855">http://hdl.handle.net/2115/48855</a></td>
</tr>
<tr>
<td>Rights</td>
<td>© 2010 IEEE. Reprinted, with permission, from Igarashi, M., Ikebe, M., Shimoyama, S., Yamano, K., Motohisa, J., O(1) bilateral filtering with low memory usage, 2010 17th IEEE International Conference on Image Processing (ICIP), Sep. 2010. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of Hokkaido University products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to <a href="mailto:pubs-permissions@ieee.org">pubs-permissions@ieee.org</a>. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.</td>
</tr>
<tr>
<td>Type</td>
<td>proceedings (author version)</td>
</tr>
<tr>
<td>File Information</td>
<td>ICIP2010_3301-3304.pdf</td>
</tr>
</tbody>
</table>
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 p denote a pixel in the image, K(p) be a set of pixels in the kernel of p. I_p be the intensity of pixel p. Bilateral filter updates intensity value as follows:

\[ I^\text{new}_p = \frac{\sum_{p' \in K(p)} w_s(p, p') w_r(I_p, I_{p'}) I_{p'}}{\sum_{p' \in K(p)} w_s(p, p') w_r(I_p, I_{p'})} \]  (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 \( B_{\text{max}} \) be the number of bins in the histogram. Generally, a histogram \( H_p \) from kernel \( K(p) \) is given as follows:

\[ b \in \{0, 1, \cdots, B_{\text{max}} - 1\} \]  (2)

\[ B_b = \frac{b B_{\text{max}}}{B_{\text{max}}} \]  (3)

\[ H_p(b) = \sum_{p' \in K(p)} f_b(I_{p'}) \]  (4)

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

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

\[ H^w_p(b) = \sum_{p' \in K(p)} f_b(I_{p'}) w_s(p, p') \]  (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.

\[ I^\text{new}_p = \frac{\sum_b H^w_p(b) w_r(I_p, b B_{\text{max}} \frac{B_{\text{max}}}{B_{\text{max}}})}{\sum_b H^w_p(b) w_r(I_p, b B_{\text{max}} \frac{B_{\text{max}}}{B_{\text{max}}})} \]  (6)

This means that we create a O(1) bilateral filtering if the weighted histogram can be calculated in constant time.
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 \( p = (x, y) \), the kernel is

\[
K(x, y) = \{ (x+i, y+j) \mid -r \leq i, j \leq r \}
\]

where \( r \) is the radius of the filter.

Assume that \( w_z \) 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:

\[
H^R_{x, y}^{<0, 0>} (b) = f_b(I_{x, y})
\]

(8)

\[
H^R_{x, y}^{<0, n>} (b) = \sum_{i=-r}^{r} H^R_{x, y+i}^{<0, n-1>} (b)
\]

(9)

\[
H^R_{x, y}^{<m, n>} (b) = \sum_{i=-r}^{r} H^R_{x+i, y}^{<m-1, n>} (b)
\]

(10)

where \( c_r \) is a normalization coefficient defined by \( c_r = \frac{1}{2r+1} \). \( H^R_{x, y}^{<1, 1>} \) is the standard histogram. We can take \( H^R_{x, y}^{<m, n>} \) 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:

\[
H^R_{x, y}^{<0, n>} (b) = H^R_{x, y}^{<0, n-1>} (b) - H^R_{x, y-r, n-1} (b)
\]

(11)

\[
H^R_{x, y}^{<m, n>} (b) = H^R_{x, y}^{<m-1, n>} (b) - H^R_{x, y-r, n-1} (b)
\]

(12)

This means that we can obtain histogram \( H^R_{x, y}^{<m, n>} \) 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 \( H^R_{x, y}^{<2, 2>} \) calculation in algorithm 1. Then, we show a diagram about composition of \( h_{22} \) in Fig.1.

**Algorithm 1** Calculate histogram \( h_{22} (H^R_{x, y}^{<2, 2>}) \) in constant time.

<table>
<thead>
<tr>
<th>width</th>
<th>width of image.</th>
</tr>
</thead>
<tbody>
<tr>
<td>h02[width]</td>
<td>histogram ( H^R_{x, y}^{&lt;0, 2&gt;} ) array</td>
</tr>
<tr>
<td>h01[top][width]</td>
<td>histogram ( H^R_{x, y}^{&lt;0, 1&gt;} ) array</td>
</tr>
<tr>
<td>h01[bottom][width]</td>
<td>histogram ( H^R_{x, y}^{&lt;0, 1&gt;} ) array</td>
</tr>
</tbody>
</table>

for \( i = 0 \) to \( \text{height} \) do

if \( i == 0 \) then

initialize \( h_{02}[j] \)

initialize \( h_{01[top]}[j], h_{01[bottom]}[j] \)

else

for \( j = 0 \) to \( \text{width} \) do

remove \( \text{pixel} (i - 1, j) \) from \( h_{01[top]}[j] \)

add \( \text{pixel} (i - 1, j) \) to \( h_{01[top]}[j] \)

remove \( \text{pixel} (i - 1, j) \) from \( h_{01[bottom]}[j] \)

add \( \text{pixel} (i + 1, j) \) to \( h_{01[bottom]}[j] \)

\( h_{02}[j] = h_{02}[j] - h_{01[top]}[j] + h_{01[bottom]}[j] \)

end for

end if

\( h_{22}, h_{12[left]}, h_{12[right]} \): histograms

for \( j = 0 \) to \( \text{width} \) do

if \( j == 0 \) then

initialize \( h_{22}, h_{12[left]}, h_{12[right]} \)

else

\( h_{12[left]} = h_{12[left]} - h_{02}[j - r - 1] + h_{02}[j - 1] \)

\( h_{12[right]} = h_{12[right]} - h_{02}[j - 1] + h_{02}[j + r] \)

\( h_{22} = h_{22} - h_{12[left]} + h_{12[right]} \)

end if

end for

end for

---

**Fig. 1.** Composition of recursive histogram \( h_{22} : H^R_{x, y}^{<2, 2>} \)
According to Eq.11, \( H_{R<0,n}^{R<0,n}\) is calculated easily by keeping the pre-calculated value of \( H_{x,y}^{R<0,n} \). Considering the moving direction of a filter window, we should keep \( \{ H_{R<0,n}^{0,y} \cdots H_{R<0,n}^{0,y-1} \} \). For \( h_{22} (H_{R<2,2}^{2,2}) \) calculation, we prepared three line buffers. One is used for the \( h_{02} (H_{R<0,2}^{R<0,2}) \) process while the others are used for the \( h_{01_{top}} (H_{R<0,1}^{R<0,1}) \) and \( h_{01_{bottom}} (H_{R<0,1}^{R<0,1}) \) processes. The \( h_{01} \) line buffer stores the \( h_{01} \)s of the previous line. The \( h_{01_{top}} \) and the \( h_{01_{bottom}} \) are updated by adding and subtracting pixels (Fig.1 left). Because the \( h_{02} \)s of previous line have been stored to line buffer, the \( h_{02} \) can be updated by adding/subtracting the \( h_{01_{top}} \) and the \( h_{01_{bottom}} \) (Fig.1 center). Updated column histograms(\( h_{01} \) and \( h_{02} \)) are stored in the line buffers again.

The \( h_{12} \) can be updated by adding and subtracting the pre-calculated \( h_{02} \) in the line buffer. Adding and subtracting the \( h_{12_{left}} \) and the \( h_{12_{right}} \) to/from the \( h_{22} \) enables us to obtain the updated \( h_{22} \) (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 require 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 \( H_{R<2,2}^{R<2,2} \)). 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 \( H_{R<2,2}^{R<2,2} \). 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
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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>box $H^{R&lt;2,2&gt;}$</td>
<td>$B\times W \times H$</td>
<td>$4\times W \times H$</td>
<td>$B\times W + B$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$3\times (B\times W + B)$</td>
</tr>
</tbody>
</table>

5. REFERENCES