2011 International Symposium on Nonlinear Theory and its Applications
NOLTA2011, Kobe, Japan, September 4-7, 2011

Bilateral Filtering Using Constant-Time Convolution

Masaki Igarashi†, Masayuki Ikebe‡, Sohsuke Shimoyama† and Junichi Motohisa‡

†‡Graduate School of Information Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, 060-0814, Japan
Email: †{igarashi, simoyama}@impulse.ist.hokudai.ac.jp, ‡{ikebe, motohisa}@ist.hokudai.ac.jp

Abstract—We propose a constant-time algorithm for bilateral filter. Bilateral filter can be converted into operation of three-dimensional (3D) convolution. By formulation of moving sum using recurrence formula, we can reduce the number of calculation needed to construct pseudo Gaussian filter. Applying one-dimensional Guassian filter to the 3D convolution, we achieved constant-time bilateral filter. We used a 3-GHz CPU without downsampling, SIMD instructions, or multi-thread operations. We confirmed our proposed bilateral filter is processed in constant time. In some conditions, high PSNR over 40 dB is obtained.

1. Introduction

In the field of image processing, image filters are widely used for the various purpose such as noise reduction, image enhancement and edge detection. 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 [3]. 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 $O(n^2)$ computational cost per output pixel (n: filter size). Several studies have been made on speeding up of the bilateral filter. Porikli proposed the $O(1)$ bilateral filter using an integral histogram [5]. Yang et al. also proposed a $O(1)$ algorithm based on Durand's method [2][6]. Paris et al. studied the bilateral filter as 3D convolution. This method uses fast fourier transform (FFT) for the convolution operation, so it takes a large amount of calculation.

The purpose of this study is to propose the constant-time algorithm for the bilateral filter by extension of Paris's method.

2. Bilateral Filtering

In this section, we explain the basis of the bilateral filter. Main notation in this paper is summarized in Table 1.

The bilateral filter uses a spatial and a range kernel for update of each pixel in an image. Let $\boldsymbol{p} \in \mathcal{S}$ denote a pixel position in the image, $I_{\boldsymbol{p}}$ be the intensity of pixel $\boldsymbol{p}$. Bilateral filter updates intensity value as follows:

$$\begin{aligned} I_{\boldsymbol{p}}^{\mathrm{bf}} &= \frac{1}{W_{\boldsymbol{p}}^{\mathrm{bf}}} \sum_{\boldsymbol{q}\in\mathcal{S}} w_{\mathrm{s}}(\boldsymbol{p},\boldsymbol{q}) w_{\mathrm{r}}(I_{\boldsymbol{p}}, I_{\boldsymbol{q}}) I_{\boldsymbol{q}} \\ W_{\boldsymbol{p}}^{\mathrm{bf}} &= \sum_{\boldsymbol{q}\in\mathcal{S}} w_{\mathrm{s}}(\boldsymbol{p},\boldsymbol{q}) w_{\mathrm{r}}(I_{\boldsymbol{p}}, I_{\boldsymbol{q}})\ . \end{aligned} \tag{1}$$

where w_{s} and w_{r} are the weighting functions in the spatial and range domains respectively. Usually, a constant and a Gaussian weighting function are used for w_{s} and the Gaussian weighting function is used as w_{r}.

Two dimensional (2D) bilateral filter can be rewritten as 3D convolution [4] as follows:

$$\forall x, y, z \in \mathcal{S} \times \mathcal{R}$$

$$I^*_{x,y,z} = \begin{cases} I_{x,y} & (z = I_{x,y}) \\ 0 & (z \neq I_{x,y}) \end{cases} \tag{2}$$

$$\begin{pmatrix} I'_{x,y,z} W'_{x,y,z} \\ W'_{x,y,z} \end{pmatrix} = \sum_{x' \in \mathcal{W}} \left[G_{\sigma_s}(x' - x) \begin{pmatrix} I^*_{x',y,z} \\ 1 \end{pmatrix} \right] \tag{3}$$

$$\begin{pmatrix} I''_{x,y,z} W''_{x,y,z} \\ W''_{x,y,z} \end{pmatrix} = \sum_{y' \in \mathcal{H}} \left[G_{\sigma_s}(y' - y) \begin{pmatrix} I'_{x,y',z} \\ 1 \end{pmatrix} \right] \tag{4}$$

$$\begin{pmatrix} I'''_{x,y,z} W'''_{x,y,z} \\ W'''_{x,y,z} \end{pmatrix} = \sum_{z' \in \mathcal{R}} \left[G_{\sigma_r}(z' - z) \begin{pmatrix} I''_{x,y,z'} \\ 1 \end{pmatrix} \right] \tag{5}$$

$$I^{\mathrm{bf}}_{x,y} = I'''_{x,y,I_{x,y}}\ . \tag{6}$$

Table 1: Notation used in this paper

$\mathbb{N}$	Set of natural numbers
W, H	Image width, height
$\mathcal{W}$	Set of x-coordinates: $\{0, 1, \cdots, W-1\}$
$\mathcal{H}$	Set of y-coordinates: $\{0, 1, \cdots, H-1\}$
$\mathcal{S}$	Spatial domain: $\mathcal{W} \times \mathcal{H}$
$\mathcal{R}$	Range domain
$\boldsymbol{p} = (x, y) \in \mathcal{S}$	Pixel position
$I_{\boldsymbol{p}}$	Brightness value at $\boldsymbol{p}$
$G_\sigma(x)$	One dimensional Gaussian: $\exp(-\frac{x^2}{2\sigma^2})$
$\sigma_{\mathrm{s}}, \sigma_{\mathrm{r}}$	Standard deviation of Gaussian (space, range)
$I^{\mathrm{bf}}_{x,y}$	Result of the bilateral filter

3. Proposed Bilateral Filtering Method

In this section, we propose a method of bilateral filtering using constant-time convolution.

Table 2: Weight map

		s_n	s'_n	s''_n
	1	1, 1, 1	1, 2, 3, 2, 1	1, 3, 6, 7, 6, 3, 1
r	2	1, 1, 1, 1, 1	1, 2, 3, 4, 5, 4, 3, 2, 1	1, 3, 6, 10, 15, 18, 19, 18, 15, 10, 6, 3, 1
	3	1, 1, 1, 1, 1, 1, 1	1, 2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1	1, 3, 6, 10, 15, 21, 28, 33, 36, 37, 36, 33, 28, 21, 15, 10, 6, 3, 1

3.1. Constant-Time Convolution

Let $x_{n\in\{0,1,\cdots,N-1\}}$ denote data sequence, where N is the length of the sequence. We define the moving sum s_n which calculates the summation from x_{n-r} to x_{n+r} as follows:

$$s_n = \sum_{i=-r}^{r} x_{n+i} \qquad (r \in \mathbb{N}) , \tag{7}$$

where r means radius. Here we are not concerned with handling the elements located out of the sequence $(x_{i<0}, x_{i\geq N})$[1]. We can convert Eq. (7) into recurrence formula:

$$s_0 = \sum_{i=-r}^{r} x_i$$
$$s_{n+1} = s_n + x_{n+r+1} - x_{n-r} . \tag{8}$$

When calculating successive values, new sum s_{n+1} is calculated by adding/subtracting two elements x_{n+r+1}, x_{n-r} to/from the old sum s_n, meaning a full summation each time is unnecessary.

Then we define the moving sum s'_n which calculates the summation of s_n as follows:

$$s'_n = \sum_{i=-r}^{r} s_{n+i} . \tag{9}$$

Similarly, we can rewrite Eq. (9) as follows:

$$s'_0 = \sum_{i=-r}^{r} s_i$$
$$s'_{n+1} = s'_n + s_{n+r+1} - s_{n-r} . \tag{10}$$

We can obtain s'_{n+1} by adding/subtracting two elements s_{n+r+1}, s_{n-r} to/from the old sum s'_n. Similarly, we can define s''_n which calculates summation of s'_n , s'''_n and so on. The calculation amount of these summation is independent of r by calculating based on recurrence formula.

Let us focus on the weight on the data sequence. Weighted sum such as s'_n and s''_n can be expanded as follows:

$$\begin{aligned} s'_n &= \sum_{i=-2}^{2} s_{n+i} = \sum_{i=-2}^{2} \{x_{n+i-2} + x_{n+i-1} + x_{n+i} + x_{n+i+1} + x_{n+i+2}\} \\ &= x_{n-4} + 2x_{n-3} + 3x_{n-2} + 4x_{n-1} + 5x_n \\ &+ 4x_{n+1} + 3x_{n+2} + 2x_{n+3} + x_{n+4} . \end{aligned} \tag{11}$$

[1]This handling is dependent on the implementations (e.g. $x_i = x_{-i-1}$ or $x_i = 0$ etc. if $i < 0$).

$$\begin{aligned} s''_n &= \sum_{i=-1}^{1} s'_{n+i} = \sum_{i=-1}^{1} \{s_{n+i-1} + s_{n+i} + s_{n+i+1}\} \\ &= \sum_{i=-1}^{1} \{x_{n+i-2} + 2x_{n+i-1} + 3x_{n+i} + 2x_{n+i+1} + x_{n+i+2}\} \\ &= x_{n-3} + 3x_{n-2} + 6x_{n-1} + 7x_n + 6x_{n+1} + 3x_{n+2} + x_{n+3} . \end{aligned} \tag{12}$$

The center weights such as coefficient of x_n are large. Meanwhile, the weights to the data apart from center tend to be small. By extending s_n and s'_n to s''_n, s'''_n and using average operation instead of summation, $s_n^{'''\cdots}$ approaches Gaussian filter according to central limit theorem. We show the weight map in Table 2.

As stated above, $s_n^{'''\cdots}$ is calculated in constant time and gives Gaussian-like weight to data. Consequently, we can achieve pseudo Gaussian filter processed in constant time by using $s_n^{'''\cdots}$. We show a pseudo code of the algorithm for pseudo Gaussian filter corresponding to s''_n in Procedure 1.

Procedure 1 Convolve(x, r): Pseudo Gaussian using constant-time convolution (e.g. s''_n).

```
Input: Data array $x$, radius $r$
Output: Filtered data $y$
1:  $N \leftarrow$ length of array $x$
2:  for $n = 0$ to $N - 1$ do
3:      if $n = 0$ then
4:          Initialize variables such that
            $g_{1LL} \leftarrow s_{-2r-2}$, $g_{1LR} \leftarrow s_{-1}$,
            $g_{1RL} \leftarrow s_{-1}$, $g_{1RR} \leftarrow s_{2r}$,
            $g_{2L} \leftarrow s'_{-r-1}$, $g_{2R} \leftarrow s'_r$, $g_3 \leftarrow s''_0$
            based on definition.
5:      else
6:          The calculation amount of following operation is independent of radius $r$.
7:          $g_{1LL} \leftarrow g_{1LL} + x[n-r-2] - x[n-3r-3]$
8:          $g_{1LR} \leftarrow g_{1LR} + x[n+r-1] - x[n-r-2]$
9:          $g_{1RL} \leftarrow g_{1RL} + x[n+r-1] - x[n-r-2]$
10:         $g_{1RR} \leftarrow g_{1RR} + x[n+3r] - x[n+r-1]$
11:         $g_{2L} \leftarrow g_{2L} + g_{1LR} - g_{1LL}$
12:         $g_{2R} \leftarrow g_{2R} + g_{1RR} - g_{1RL}$
13:         $g_3 \leftarrow g_3 + g_{2R} - g_{2L}$
14:     end if
15:     Normalize: $y[n] \leftarrow \frac{g_3}{(2r+1)^3}$
16: end for
```

We can adjust the standard deviation of Gaussian by controlling the radius of moving sum. Equation 13 shows the relation between radius r and standard deviation σ in the case of s''_n.

$$\sigma = \frac{2r+1}{\sqrt{2\pi}} \tag{13}$$

3.2. Constant-Time Bilateral Filtering Using Fast Convolution

According to Eq. (2-6), the bilateral filter is processed by 3D convolution. Paris et al. used FFT for 3D convolution. On the other hand, we apply the method of the constant-time convolution to the bilateral filter based on 3D convolution. We present a pseudo-code of the algorithm for the bilateral filter in Procedure 2.

Procedure 2 Fast bilateral filter using constant-time convolution.

```
Input: Image $I$, radius $r_s, r_r$
Output: Filtered image $I^{bf}$
1:  for all $x, y, z \in \mathcal{W} \times \mathcal{H} \times \mathcal{R}$ do
2:      $I^*[x][y][z] \leftarrow I[x][y]$ if $z = I[x][y]$, 0 else
3:      $W^*[x][y][z] \leftarrow 1$ if $z = I[x][y]$, 0 else
4:  end for
5:  for all $y, z \in \mathcal{H} \times \mathcal{R}$ do
6:      $I'[:][y][z] \leftarrow \text{Convolve}(I^*[:][y][z], r_s)$
7:      $W'[:][y][z] \leftarrow \text{Convolve}(W^*[:][y][z], r_s)$
        Colon(:) operator indicates a specific 1D array retrieved from multidimensional data (e.g. $I[:][y][z]$ means $I[0][y][z], I[1][y][z], \cdots, I[W-1][y][z]$)
8:  end for
9:  for all $z, x \in \mathcal{R} \times \mathcal{W}$ do
10:     $I''[x][:][z] \leftarrow \text{Convolve}(I'[x][:][z], r_s)$
11:     $W''[x][:][z] \leftarrow \text{Convolve}(W'[x][:][z], r_s)$
12: end for
13: for all $x, y \in \mathcal{W} \times \mathcal{H}$ do
14:     $I'''[x][y][:] \leftarrow \text{Convolve}(I''[x][y][:], r_r)$
15:     $W'''[x][y][:] \leftarrow \text{Convolve}(W''[x][y][:], r_r)$
16: end for
17: for all $x, y \in \mathcal{W} \times \mathcal{H}$ do
18:     $I^{bf}[x][y] \leftarrow \frac{I'''[x][y][I[x][y]]}{W'''[x][y][I[x][y]]}$
19: end for
```

4. Experimental Results

We tested the proposed bilateral filter method and evaluated the processing time and image quality. These tests were run on a PC with Intel Core i7 CPU (3.07 GHz) and 8GB main memory. Our implementation is written in C++. We did not use downsampling, multithread operations, or SIMD instructions. The implementation of Paris's technique we used is available at his website[2].

The computation times are given in Fig. 1. Our proposed method is several faster than Paris's one. Calculation time of both methods became larger according to increase of image size. Next, we examined the relation between the processing time and standard deviation of Gaussian. Figure 2 shows the results of measurement. We found that the processing time of the proposed method has less dependency of the standard deviation. In the Paris's method, however, processing time varies according to the standard deviation.

We analyzed the filter accuracy by performing filtering operation and by calculating the Peek Signal-to-Noise Ratio (PSNR) defined in Eq. 14. For the analysis, the filtering image with exact Gaussian spatial weight were set to the source images. Table. 3 shows PSNR of bilateral filters with pseudo Gaussian kernel (using s''_n). In some conditions, high PSNR (over 40 dB) is obtained.

Figure 3 shows 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.

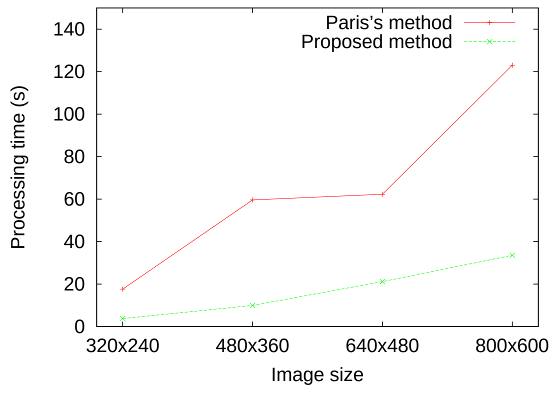

Figure 1: Comparison of processing time between two methods (natural image, $\sigma_s = \sigma_r = 13.17$).

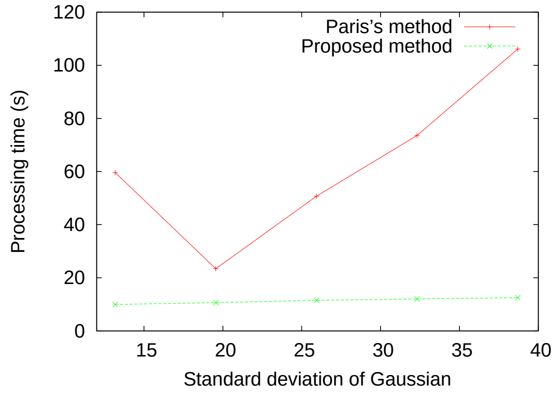

Figure 2: Relation between the processing time and standard deviation (natural image: 480×360).

[2]http://people.csail.mit.edu/sparis/bf/

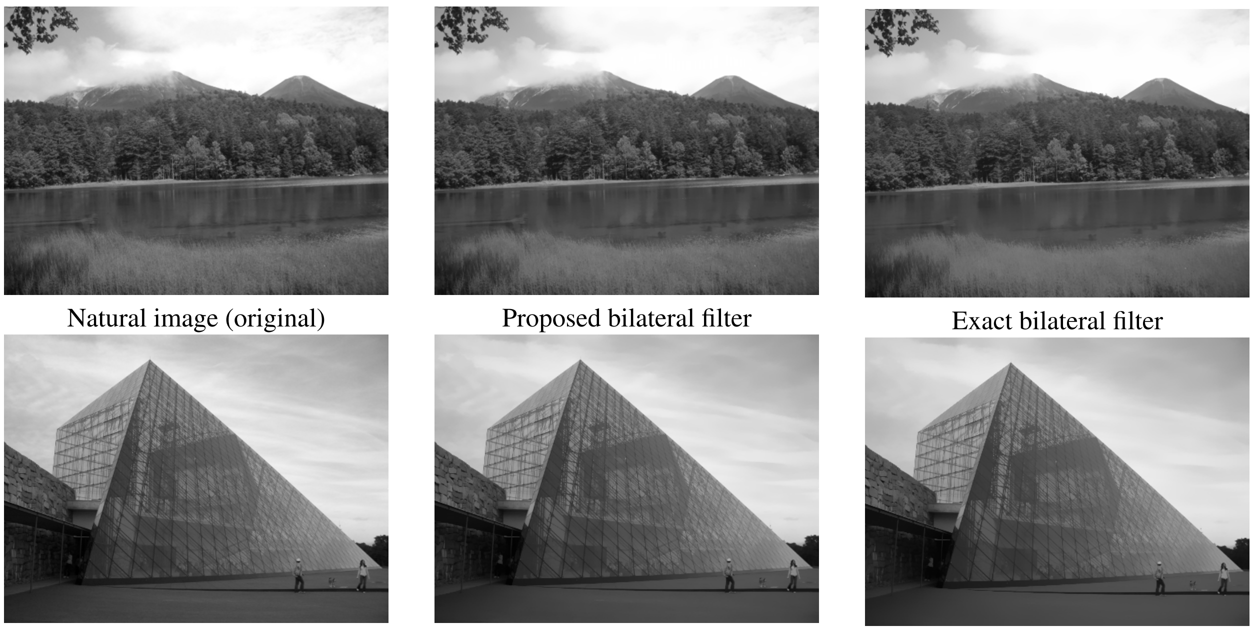
Natural image (original)
Proposed bilateral filter
Exact bilateral filter
Architectural image (original)
Proposed bilateral filter
Exact bilateral filter

Figure 3: Filtered result on some images (natural, architectural). $\sigma_s = \sigma_r = 6.8$.

Table 3: PSNR accuracy [dB] of proposed method (natural image: 480×360).

		r_s 2	4	8	16	32
r_r	2	58.1	59.7	60.6	61.0	61.5
	4	56.6	57.9	58.1	58.0	57.7
	8	54.5	55.4	54.8	54.3	54.1
	16	52.2	52.9	51.9	51.1	50.2
	32	50.1	50.4	48.5	47.0	45.2

$$\mathrm{MSE} = \frac{1}{WH}\sum_{i=0}^{W}\sum_{j=0}^{H}(I_{i,j} - J_{i,j})^2$$
$$\mathrm{PSNR} = 20\log_{10}\left(\frac{255}{\sqrt{\mathrm{MSE}}}\right) \tag{14}$$

5. Conclusion

We proposed a constant-time algorithm for bilateral filter. By formulation of moving sum using recurrence formula, we can reduce the number of calculation needed to construct pseudo Gaussian filter. Applying one-dimensional Gaussian filter to the 3D convolution, we achieved constant-time bilateral filter.

A further direction of this study will be to apply the downsampling of image to our method. Then, we will achieve the additional speeding-up of the proposed bilateral filter.

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *In Proceedings of the International Conference on Computer Vision*, pp.839–846, 1998.

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

[3] L. Edwin, " Retinex Theory of Color Vision," *Scientific American*, Volume 237, pp.108–128, 1977.

[4] S. Paris and F. Durand, "Fast Approximation of the Bilateral Filter using a Signal Processing Approach," *In Proceedings of the European Conference on Computer Vision*, pp.568–580, 2006.

[5] F. Porikli, "Constant Time O(1) Bilateral Filtering," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.1–8, 2008.

[6] Q. Yang, K. Tan and N. Ahuja, "Real-Time O(1) Bilateral Filtering," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.557–564, 2009.