Fast Implementation of Box Filtering

Masahiro Nakamura and Norishige Fukushima*
Nagoya Institute of Technology
Nagoya, Japan
Email: *fukushima@nitech.ac.jp

***Abstract*—In this paper, we propose fast and cache efficient box filtering, and also fast guided filtering using our box filtering. Box filtering is one of the smoothing filters, which computes averaged pixel values in the kernel. Also, guided filtering is one of the edge-preserving filters. This filtering performs multiple times of box filterings to obtain average and variance values in the local window. Therefore, we can use fast box filtering for accelerating guided filtering. Experimental results show that our box filtering is faster than the conventional implementations, and also faster than the conventional implementation of guided filtering.**

***Keywords*—box filtering, guided filtering, cache efficiency, acceleration**

I. Introduction

Box filtering is one of the smoothing filters, which compute the averaged pixel values in the kernel. Box filtering is often used in various applications for image processing and computer vision. One of the application examples of box filtering is guided filtering [1], [2].

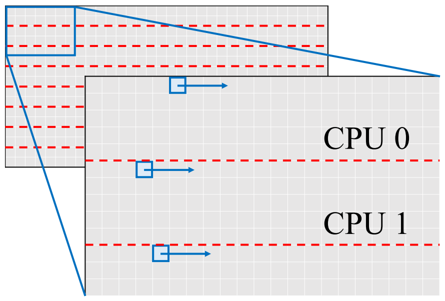
Guided filtering is one of the applications of box filtering and the efficient edge-preserving filters. The filtering is effective for blur or gradient regions thanks to the local linearity of this filtering. This filtering consists of multiple times of box filterings because of using average and variance values in the local window centered at the target pixel. For implementing brute-force guided filtering, which also uses brute-force box filtering, we must iteratively allocate-and-free cache areas every box filtering processes. This process is wasteful for guided filtering and easily makes cache disorder. Therefore, we implement fast box filtering, which has high cache efficiency for multiple times of filterings. Also, we accelerate guided filtering using our box filtering.

II. Proposed methods

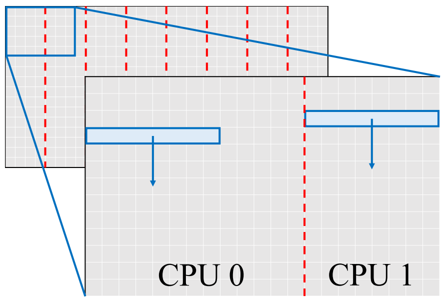
A. Box Filtering

Box filtering is a simple linear filter with a square kernel, and compute the average values in the kernel. For accelerating the filtering, there are two algorithms; the integral image [3] and the summed area table [4] methods. In our implementation, we use the summed area table approach with separability of the box filtering kernel.

When r is a kernel radius, let R be a summation of intensities in the $1 \times (2r+1)$ narrow window. The summation in the $(2r+1) \times (2r+1)$ rectangular window C is also the accumulation of intensities in the R by using separability.



(a) Parallelization of Eq. (1).



(b) Parallelization and Vectorization of Eq. (2).

Fig. 1: Diagram of implementation of proposed box filtering.

The summation R and C centered at (x, y) are computed as follows:

$$R(x,y) = R(x-1,y) + I(x+r,y) - I(x-r-1,y), \quad (1)$$

$$C(x,y) = C(x,y-1) + R(x,y+r) - R(x,y-r-1), \quad (2)$$

where $I(x, y)$ is a pixel value of the input image. The output image O is normalized as follows:

$$O(x,y) = \frac{1}{(2r+1)^2} C(x,y), \quad (3)$$

In Eq. (1) and (2), we implement parallel processing using OpenMP for acceleration (See Fig. 1). We divide the input image into horizontal direction in Eq. (1). We also divide the image which computed in Eq. (1) into vertical direction, and use vector operation with adjoining 8 pixels using AVX (Intel Advanced Vector Extensions) in Eq. (2). In our implementation, we can parallelly deal with 8 pixels in Eq. (1) and 64 pixels in Eq. (2), because we use 4-core 8-thread CPU. Furthermore, we fix cache areas for every filtering steps for efficient implementation.

Algorithm 1 Guided Filtering.

```
1: $\text{average}_J = f_{\text{box}}(J, r)$
   $\text{average}_p = f_{\text{box}}(p, r)$
   $\text{corr}_J = f_{\text{box}}(J. * J, r)$
   $\text{corr}_{Jp} = f_{\text{box}}(J. * p, r)$
2: $\text{var}_J = \text{corr}_J - \text{average}_J . * \text{average}_J$
   $\text{cov}_{Jp} = \text{corr}_{Jp} - \text{average}_J . * \text{average}_p$
3: $a = \text{cov}_{Jp} ./ (\text{var}_J + \epsilon)$
   $b = \text{average}_p - a. * \text{average}_J$
4: $\text{average}_a = f_{\text{box}}(a, r)$
   $\text{average}_b = f_{\text{box}}(b, r)$
5: $q = \text{average}_a . * J + \text{average}_b$
```

TABLE I: Number of box filterings in guided filtering.

Src	Guide	Number of times
Gray	Gray	6
	Color	17
Color	Gray	18
	Color	51

B. Guided Filtering

Guided filtering [1], [2] is one of the efficient edge-preserving filters, and effective for blur or gradient regions thanks to the local linearity of this filter. We assume that output image q is a linear transform of the guidance image J.

When the guidance image is grayscale, the output image is computed as follows:

$$q_{\boldsymbol{i}} = \bar{a}_{\boldsymbol{i}} J_{\boldsymbol{i}} + \bar{b}_{\boldsymbol{i}}, \tag{4}$$

$$\bar{a}_{\boldsymbol{i}} = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{i}}} a_{\boldsymbol{k}}, \quad \bar{b}_{\boldsymbol{i}} = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{i}}} b_{\boldsymbol{k}}, \tag{5}$$

where $\boldsymbol{i}$ and $\boldsymbol{k}$ are pixel position, $\omega_{\boldsymbol{i}}$ is sets of neighborhood pixels around the pixel $\boldsymbol{i}$, $|\omega|$ is the number of pixels of $\omega_{\boldsymbol{i}}$, and $a_{\boldsymbol{k}}$ and $b_{\boldsymbol{k}}$ are some linear coefficients, which define Eq. (6) and (7), respectively. The linear coefficients are computed as follows:

$$a_{\boldsymbol{k}} = \frac{\frac{1}{|\omega|} \sum_{\boldsymbol{i} \in \omega_{\boldsymbol{k}}} J_{\boldsymbol{i}} p_{\boldsymbol{i}} - \mu_{\boldsymbol{k}} \bar{p}_{\boldsymbol{k}}}{\sigma_{\boldsymbol{k}}^2 + \epsilon}, \tag{6}$$

$$b_{\boldsymbol{k}} = \bar{p}_{\boldsymbol{k}} - a_{\boldsymbol{k}} \mu_{\boldsymbol{k}}, \tag{7}$$

where $\mu_{\boldsymbol{k}}$ and $\sigma_{\boldsymbol{k}}^2$ are the average and variance of J in $\omega_{\boldsymbol{k}}$, $\bar{p}$ is the average of input image p in $\omega_{\boldsymbol{k}}$, and ϵ is regularization parameter.

When the guidance image is color, a color guidance image can better preserve the edges that are not distinguishable in grayscale. The output image is computed as follows:

$$q_{\boldsymbol{i}} = \bar{\boldsymbol{a}}_{\boldsymbol{i}}^T \boldsymbol{J}_{\boldsymbol{i}} + \bar{b}_{\boldsymbol{i}}, \tag{8}$$

$$\bar{\boldsymbol{a}}_{\boldsymbol{i}} = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{i}}} \boldsymbol{a}_{\boldsymbol{k}}, \quad \bar{b}_{\boldsymbol{i}} = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{i}}} b_{\boldsymbol{k}}, \tag{9}$$

TABLE II: Computational times of box filtering [msec].

	Gray	Color
Ours	**5.03**	**15.92**
OpenCV 3.0	8.26	24.88
Integral image [3]	14.42	44.25
Brute-fource	33.70	83.78

TABLE III: Computational times of guided filtering [msec].

Src	Gray		Color	
Guide	Gray	Color	Gray	Color
Ours	**98.39**	**340.73**	**313.54**	**1030.03**
OpenCV	122.63	398.79	374.22	1202.37

where $\boldsymbol{J}_{\boldsymbol{i}}$ and $\boldsymbol{a}_{\boldsymbol{k}}$ represent 3×1 vectors. $\boldsymbol{a}_{\boldsymbol{k}}$ and $b_{\boldsymbol{k}}$ are linear coefficients, which define Eq. (10) and (11), respectively. The linear coefficients are computed as follows:

$$\boldsymbol{a}_{\boldsymbol{k}} = (\boldsymbol{\Sigma}_{\boldsymbol{k}} + \epsilon \boldsymbol{U})^{-1} \left(\frac{1}{|\omega|} \sum_{\boldsymbol{i} \in \omega_{\boldsymbol{k}}} \boldsymbol{J}_{\boldsymbol{i}} p_{\boldsymbol{i}} - \boldsymbol{\mu}_{\boldsymbol{k}} \bar{p}_{\boldsymbol{k}} \right), \tag{10}$$

$$b_{\boldsymbol{k}} = \bar{p}_{\boldsymbol{k}} - \boldsymbol{a}_{\boldsymbol{k}}^T \boldsymbol{\mu}_{\boldsymbol{k}}, \tag{11}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{k}}$ is the 3×3 covariance matrix of $\boldsymbol{J}$ in $\omega_{\boldsymbol{k}}$, and $\boldsymbol{U}$ is a 3×3 identity matrix.

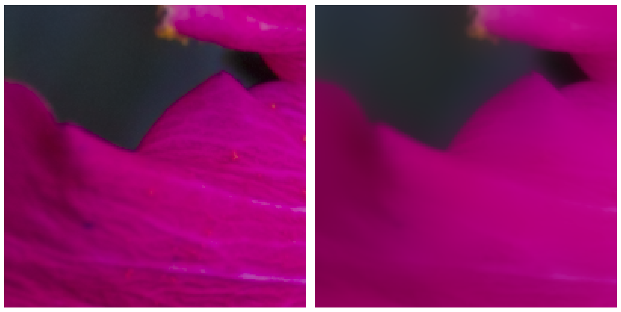
A pseudocode is in Algorithm 1. In this algorithm, f_{box} is a box filtering with a kernel radius r. The abbreviations of correlation (corr), variance (var), and covariance (cov) indicate the meaning of these variables.

The average and variance using guided filtering are computed by box filtering. TABLE I shows the number of times of box filtering, which performs in guided filtering process. When we implement guided filtering using the conventional box filtering, we must iteratively allocate-and-free cache areas every box filtering processes. This process is wasteful of processing time and easily makes cache disorder. In our implementation, we allocate cache areas using box filtering only once before the process of guided filtering and fix those cache areas. Therefore, we can reduce processing times which allocate-and-free cache areas every box filtering processes. Also, we can perform high cache efficiency box filtering and guided filtering because cache areas do not disorder.

III. Experimental Results

In the experiments, we compare our implementation with conventional implementations. The input and guidance images whose resolution is 2268×1512 are grayscale or color images. The kernel size which uses box filtering is 21×21. Note that the computational time does not depend on kernel radius. We have implemented our proposed and competition methods written in C++ with Visual Studio 2013 on Windows 10 64bit. The CPU for the experiment is 3.40 GHz Intel Core i7-6700 (4-core 8-thread).

TABLE II shows the computational times of each box filtering. We compare our proposed box filtering with three conventional methods. The conventional methods are the OpenCV 3.0's implementation, integral image implementation [3] and brute-force implementation. The proposed box filtering is



(a) Input image. (b) Guided filter.

Fig. 2: Zoomed result of our guided filtering.

faster than the conventional methods in both grayscale and color image.

TABLE III shows the computational times of brute-force guided filtering using proposed box filtering and using OpenCV 3.0's box filtering. We compare with a total of four types of grayscale and color image as input and guidance images. The guided filtering using our box filtering is faster than using OpenCV 3.0's box filtering. Fig. 2 shows the filtering result of our proposed guided filtering. Our method preserves edge and also flat regions.

IV. Conclusion

In this paper, we proposed fast box filtering implementation, which has high cache efficiency and also proposed fast guided filtering using our box filtering. Proposed box filtering used the summed area table approach [4] with separability of the filtering kernel, and vector operation using AVX. In the implementation of guided filtering, we allocated and fixed cache areas using box filtering before the process of guided filtering. Experimental results showed that our box filtering was faster than the conventional methods, and brute-force guided filtering using our box filtering was faster than using OpenCV 3.0's box filtering.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP15K16023.

References

[1] K. He, J. Shun, and X. Tang, "Guided image filtering," in *Proc. European Conference on Computer Vision (ECCV)*, 2010, pp. 1–14.

[2] ——, "Guided image filtering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, 2013.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 511–518.

[4] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. ACM SIGGRAPH*, 1984, pp. 207–212.