IMAGE TILING FOR CLUSTERING TO IMPROVE STABILITY OF CONSTANT-TIME COLOR BILATERAL FILTERING

Takahisa Miyamura[*], *Norishige Fukushima*[*], *Muhammad Waqas*[*],
Kenjiro Sugimoto[†], and Sei-ichiro Kamata[†]

[*]Nagoya Institute of Technology, Japan
[†]Waseda University, Japan

ABSTRACT

Bilateral filtering is a typical edge-preserving smoothing and it is used in various applications. The main issue of bilateral filtering is the processing time. In order to solve this problem, constant-time bilateral filtering has been proposed. The constant-time bilateral filter is an effective method for grayscale images, but it takes high cost for color images because of the curse of dimensionality. Some algorithms specialize in constant-time color bilateral filtering for color images by using clustering. However, the clustering has randomness, and computational cost itself is also high. In this paper, we propose an acceleration method of clustering by using K-means++, tiling, and subsampling, and also achieve improvement of the stability.

***Index Terms*—** edge-preserving filter, color bilateral filter, constant-time bilateral filter, tiling, K-means++

1. INTRODUCTION

Edge-preserving smoothing can preserve contours in an image while its smoothing process. A typical example of the edge-preserving filter is bilateral filtering (BF) [1]. The kernel in BF composites two kernels according to pixel positions (spatial kernel) and pixel value (range kernel). BF is used in various image processing applications, such as denoising [2], deblurring [3], detail enhancement [4], HDR [5], haze removing [6], stereo matching [7], and optical flow [8].

The problem of BF is its computational complexity. The computational time of BF is much higher than that of a linear filter, such as Gaussian filtering (GF), since BF is spatially variant filtering.

There are some acceleration methods for BF in gray and color images. For grayscale image, early works use FFT or separable filtering for acceleration [5, 9–12]. Recent works use constant-time Gaussian filtering (GF) [13, 14] for spatial convolution, then the cost of BF becomes constant per pixel [15–18], i.e., computational time does not depend on the kernel radius. The $O(1)$ BF is effective for grayscale images; however, its computational complexity increases exponentially for color or multi-channel images because of the curse of dimensionality.

For the curse of dimensionality in constant-time color bilateral filtering ($O(1)$ CBF), early works spatially subsample images for acceleration [16, 19]. Following works use random sampling of approximating functions of the range kernel to reduce the number of convolutions [20–22]

Recent works [23–25] further improve the performance by using clustering. Therefore, the accuracy of $O(1)$ CBF depends on the clustering result. Also, reasonable clustering methods utilize a randomized algorithm; thus, the filtering results have variance. Moreover, the computational cost of the clustering itself becomes overhead.

In this paper, we propose a method to solve the problems of the clustering-based $O(1)$ CBF [24, 25], such as high computational cost and randomness, by using K-means++ clustering [26], tiling [17], and subsampling. The contributions of this paper are as follows:

- Revealing the problem of randomness in clustering-based CBF and suppressing the variation of the accuracy due to random sampling by using tiling and K-means++.
- Showing that the tiling and K-means++ also improve the approximation accuracy per cluster.
- Showing that clustering is insensitive to subsampling, and then subsampling improves the speed of clustering processing itself. Also, tiling improves the speed of filtering.

2. $O(1)$ COLOR BILATERAL FILTER

We firstly define color bilateral filtering (CBF). A D-dimensional R-tone color image $\boldsymbol{f} : \mathcal{S} \mapsto \mathcal{R}$, where $\mathcal{S} \subset \mathbb{Z}^D$ and $\mathcal{R} \subset \mathbb{R}^3$ denotes the spatial domain and range domain (generally, $D = 2$ and $R = 256$), respectively. Using a pixel position $\boldsymbol{p} \in \mathcal{S}$, its color intensity vector $\boldsymbol{f_p} \in \mathcal{R}$, and its neighboring pixels $N(\boldsymbol{p}) \subset \mathcal{R}$, CBF [1] is defined as follows;

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} = \frac{\sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) w_r(\boldsymbol{f_p}, \boldsymbol{f_q}) \boldsymbol{f_q}}{\sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) w_r(\boldsymbol{f_p}, \boldsymbol{f_q})}, \tag{1}$$

[*]This work was supported by JSPS KAKENHI JP17H01764, JP18K19813

[†]This work was supported by JSPS KAKENHI JP17H01764, JP18K18076.

where $w_s : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$ is a spatial kernel and $w_r : \mathcal{R} \times \mathcal{R} \mapsto \mathbb{R}$ is a range kernel. The most common choice of the kernels is Gaussian distribution:

$$w_s(\boldsymbol{p}, \boldsymbol{q}) = e^{-\frac{\|\boldsymbol{q}-\boldsymbol{p}\|_2^2}{2\sigma_s^2}}, \qquad w_r(\boldsymbol{a}, \boldsymbol{b}) = e^{-\frac{\|\boldsymbol{b}-\boldsymbol{a}\|_2^2}{2\sigma_r^2}}, \tag{2}$$

where $\sigma_s \in \mathbb{R}_+$ is spatial scale and $\sigma_r \in \mathbb{R}_+$ is range scale.

In $O(1)$ CBF, the color information is clustered using the K-means method, and the dominant color vectors are determined at K points, where $\boldsymbol{\mu}_k \in \mathbb{R}^3, (k = 1 \ldots, K)$ is the dominant color vectors. The main idea of $O(1)$ CBF is to decompose BF into the sum of multiple Gaussian filters, using the dominant colors as sampling points.

2.1. Soft-assignment Coding [24]

Sugimoto et al. apply soft-assignment coding [27, 28] CBF [24] Soft-assignment coding is known in general object recognition tasks. This approximation generates component images corresponding to some dominant color vectors. The approximate accuracy is enhanced by utilizing linear combinations of them derived from soft-assignment coding. The soft-assignment coding based CBF is defined by;

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} \approx \sum_{k=1}^{K} \alpha_k(\boldsymbol{f}_{\boldsymbol{p}}) \frac{\boldsymbol{\xi}_{\boldsymbol{\mu}_k}(\boldsymbol{p})}{\zeta_{\boldsymbol{\mu}_k}(\boldsymbol{p})}, \tag{3}$$

and $\boldsymbol{\xi}_{\boldsymbol{\mu}_k} : \mathcal{S} \mapsto \mathcal{R}$ and $\zeta_{\boldsymbol{\mu}_k} : \mathcal{S} \mapsto \mathbb{R}$ are defined by

$$\boldsymbol{\xi}_{\boldsymbol{\mu}_k} = \sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \{ w_r(\boldsymbol{\mu}_k, \boldsymbol{f}_{\boldsymbol{q}}) \boldsymbol{f}_{\boldsymbol{q}} \}, \tag{4}$$

$$\zeta_{\boldsymbol{\mu}_k} = \sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \{ w_r(\boldsymbol{\mu}_k, \boldsymbol{f}_{\boldsymbol{q}}) \}, \tag{5}$$

where α_k is weights derived by soft-assignment coding, $\boldsymbol{\mu}_k$ is dominant color vector determined by clustering.

$$\alpha_k(\boldsymbol{x}) = \frac{\exp\left(-\lambda \|\boldsymbol{x} - \boldsymbol{\mu}_k\|_2^2\right)}{\sum_{l=1}^{K} \exp\left(-\lambda \|\boldsymbol{x} - \boldsymbol{\mu}_l\|_2^2\right)}, \tag{6}$$

where λ is a smoothing parameter. Note that the convolution of $\sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \{\cdot\}$ can be calculated in $O(1)$ by constant-time Gaussian filtering ($O(1)$ GF) [14]. The required number of convolutions is $4K$, where K in the denominator and $3K$ in the numerator.

2.2. Nyström Approximation [25]

Nair et al. [25] apply Nyström approximation [29] to the problem of eigen value decomposition (EVD) in $O(1)$ CBF. First, we explain $O(1)$ BF [30, 31] using EVD of the range kernel matrix. Let $\mathcal{T} = \{\boldsymbol{f}_{\boldsymbol{x}} : \boldsymbol{x} \in \mathcal{S}\}$ be a list consisting of pixel values and let $\mathcal{T} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_m\}$ be some ordering of the elements in $\mathcal{T}$, where m is the number of elements. This means that, given $l \in [1, m], \boldsymbol{t}_l = \boldsymbol{f}_{\boldsymbol{x}}$ for some $\boldsymbol{x} \in \mathcal{S}$. We track the correspondence by the index map;

$$\tau(\boldsymbol{x}) = l \quad \text{if } \boldsymbol{t}_l = \boldsymbol{f}_{\boldsymbol{x}}. \tag{7}$$

We next define the kernel matrix $\boldsymbol{W} \in \mathbb{R}^{m \times m}$ given by

$$\boldsymbol{W}(i, j) = w_r(\boldsymbol{t}_i, \boldsymbol{t}_j). \tag{8}$$

Substituting (8) for (1) gives

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} = \frac{\sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \boldsymbol{W}(\tau(\boldsymbol{p}), \tau(\boldsymbol{q})) \boldsymbol{f}_{\boldsymbol{q}}}{\sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \boldsymbol{W}(\tau(\boldsymbol{p}), \tau(\boldsymbol{q})))}. \tag{9}$$

Since $\boldsymbol{W}$ is a symmetric matrix, EVD of $\boldsymbol{W}$ is as follows;

$$\boldsymbol{W} = \sum_{k=1}^{m} \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^{\mathrm{T}}, \tag{10}$$

where $\lambda_k (\lambda_1 \geq \ldots \geq \lambda_m) \in \mathbb{R}$ are its eigenvalues, and $\boldsymbol{u}_k \in \mathbb{R}^m$ are the corresponding eigenvectors. Substituting (10) to (9) gives

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} = \frac{\sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \sum_{k=1}^{m} \lambda_k \boldsymbol{u}_k(\tau(\boldsymbol{p})) \boldsymbol{u}_k(\tau(\boldsymbol{q})) \boldsymbol{f}_{\boldsymbol{q}}}{\sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \sum_{k=1}^{m} \lambda_k \boldsymbol{u}_k(\tau(\boldsymbol{p})) \boldsymbol{u}_k(\tau(\boldsymbol{q}))}. \tag{11}$$

On switching the sums, this becomes

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} = \frac{\sum_{k=1}^{m} \lambda_k \boldsymbol{u}_k(\tau(\boldsymbol{p})) \sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \{\boldsymbol{u}_k(\tau(\boldsymbol{q})) \boldsymbol{f}_{\boldsymbol{q}}\}}{\sum_{k=1}^{m} \lambda_k \boldsymbol{u}_k(\tau(\boldsymbol{p})) \sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \{\boldsymbol{u}_k(\tau(\boldsymbol{q}))\}}. \tag{12}$$

Let $\hat{\boldsymbol{W}} \in \mathbb{R}^{m \times m}$ be the matrix of low rank approximation of $\boldsymbol{W}$ using the top K ($K \ll m$) eigenvalues and eigenvectors.

$$\hat{\boldsymbol{W}} = \sum_{k=1}^{K} \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^{\mathrm{T}}, \tag{13}$$

Using $\hat{\boldsymbol{W}}$ instead of $\boldsymbol{W}$ in the formula of (12), CBF can be approximated as

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} \approx \frac{\sum_{k=1}^{K} \lambda_k \boldsymbol{u}_k(\tau(\boldsymbol{p})) \sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \{\boldsymbol{u}_k(\tau(\boldsymbol{q})) \boldsymbol{f}_{\boldsymbol{q}}\}}{\sum_{k=1}^{K} \lambda_k \boldsymbol{u}_k(\tau(\boldsymbol{p})) \sum_{\boldsymbol{q} \in N(\boldsymbol{p})} w_s(\boldsymbol{p}, \boldsymbol{q}) \{\boldsymbol{u}_k(\tau(\boldsymbol{q}))\}}. \tag{14}$$

As a result, CBF can be processed in constant time [30, 31], similar to Sugimoto et al. [24]. The required number of convolutions is $4K$, where K in the denominator and $3K$ in the numerator.

In the grayscale case, the size of the matrix $\boldsymbol{W}$ is 256×256. However, in the color case, the matrix size is millions $\times$ millions, i.e., 256^3. Therefore, EVD of $\boldsymbol{W}$ is difficult. For Nyström approximation [29] of EVD of $\boldsymbol{W}$ [25], we first construct a small range kernel matrix $\boldsymbol{A}$ and then extrapolate its eigenvectors to approximate those of $\boldsymbol{W}$. $\boldsymbol{A} \in \mathbb{R}^{K \times K}$ is defined using dominant color vectors $\boldsymbol{\mu}_k$ determined by clustering as

$$\boldsymbol{A}(i, j) = w_r(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) \quad (i, j \in [1, K]). \tag{15}$$

The size of $\boldsymbol{A}$ is much smaller than that of $\boldsymbol{W}$. Thus, we can easily compute EVD:

$$\boldsymbol{A} = \sum_{k=1}^{K} \lambda_k \boldsymbol{v}_k \boldsymbol{v}_k^{\mathrm{T}}, \tag{16}$$

where $\lambda_k \in \mathbb{R}$, and $\boldsymbol{v}_k \in \mathbb{R}^K$. We next construct $\boldsymbol{B} \in \mathbb{R}^{K \times m}$:

$$\boldsymbol{B}(i, j) = w_r(\boldsymbol{\mu}_i, \boldsymbol{t}_j) \quad (i \in [1, K], j \in [1, m]). \tag{17}$$

This matrix is used to extrapolate $\boldsymbol{u}_k$ as follows;

$$\boldsymbol{u}_k = \frac{1}{\lambda_k}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{v}_k. \tag{18}$$

These calculations eliminate the need to solve EVD of $\boldsymbol{W}$ and solve the problem of the curse of dimensionality.

3. PROPOSED METHOD

In this section, we introduce three techniques for clustering in CBF, such as K-means++, tiling, and subsampling. First of all, formulating $O(1)$ CBF, Sugimoto et al. is as follows;

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} \approx \sum_{k=1}^{K} \alpha_k \frac{\sum_{\boldsymbol{q}\in N(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\{w_r(\boldsymbol{\mu}_k,\boldsymbol{f}_{\boldsymbol{q}})\boldsymbol{f}_{\boldsymbol{q}}\}}{\sum_{\boldsymbol{q}\in N(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\{w_r(\boldsymbol{\mu}_k,\boldsymbol{f}_{\boldsymbol{q}})\}}, \tag{19}$$

and Nair et al. is as follows;

$$\hat{\boldsymbol{f}}_{\boldsymbol{p}} \approx \frac{\sum_{k=1}^{K} \frac{1}{\lambda_k} X_{(\boldsymbol{f}_{\boldsymbol{p}},k)} \sum_{\boldsymbol{q}\in N(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\{X_{(\boldsymbol{f}_{\boldsymbol{q}},k)}\boldsymbol{f}_{\boldsymbol{q}}\}}{\sum_{k=1}^{K} \frac{1}{\lambda_k} X_{(\boldsymbol{f}_{\boldsymbol{p}},k)} \sum_{\boldsymbol{q}\in N(\boldsymbol{p})} w_s(\boldsymbol{p},\boldsymbol{q})\{X_{(\boldsymbol{f}_{\boldsymbol{q}},k)}\}} \tag{20}$$

where α_k is coefficient and $X_{(\boldsymbol{f},k)} = \sum_{j=1}^{K} w_r(\boldsymbol{\mu}_j,\boldsymbol{f})\boldsymbol{v}_k(j)$. From these equations, filtering results depend on the value of $\boldsymbol{\mu}_k$. In other words, the accuracy can be improved by improving how to take $\boldsymbol{\mu}_k$, i.e., sampling points.

3.1. K-means++

K-means++ [26] is a variant of K-means. K-means randomly takes initial centroids. By contrast, K-means++ determines the initial centroids by the following procedure. Let $\boldsymbol{c}_i \in \mathbb{R}^3, (i = 1\ldots, K)$ be the initial centroids to be determined.

1. Choose one centroid $\boldsymbol{c}_1$ at random from data points.
2. For each data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, which is the shortest distance between $\boldsymbol{x}$ and the closest centroid, where we have already chosen.
3. Stochastically choose one new centroid $\boldsymbol{c}_i$ according to weighted probability distribution $\frac{D(\boldsymbol{x})^2}{\sum D(\boldsymbol{x})^2}$.
4. Repeat Step 2 until K-point centroids are chosen.

Each initial centroid is placed away from each other by K-means++ because of weighted probability distribution; thus, bias among sampling points are smaller than K-means. Therefore, K-means++ improves clustering accuracy and stability.

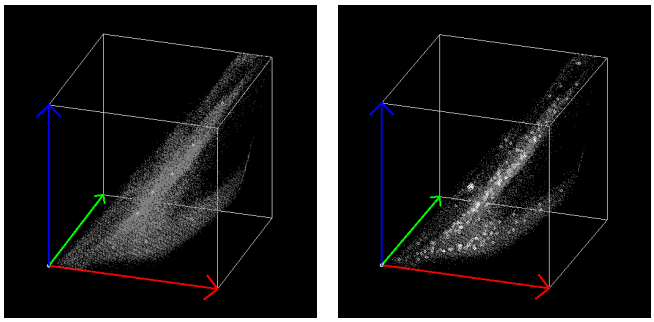
Table 1 shows the minimum, maximum, variance, and the average of the within-cluster sum of squares (WCSS) [32] when the color vector clustering is performed 1000 times for the image "Kodim04", where the number of clusters is $K = 15$). The clustering is performed by setting the luminance range of the image from 0-255 to 0-1. The statistics show that stability and accuracy are improved from K-means.

3.2. Tiling

Tiling is a technique that divides an input image into multiple images (sub-images) and then performs image processing for each sub-image. Finally, the processed sub-images are combined into an output image. When applying the tiling method, it is necessary to pad the sub-images by the kernel radius. The padding introduces redundant processing.

Table 1: Within-cluster sum of squares (WCSS) statistics for each method ($K = 15$). * is the total WCSS for each tile.

Statistics\Method	K-means	K-means++	K-means with tiling*	K-means++ with tiling*
minimum	1495.99	1493.02	1000.21	831.40
maximum	3002.58	1713.60	1436.63	875.64
variance	42195.58	1126.32	5229.76	58.94
average	1836.37	1544.17	1178.75	849.36



(a) without tiling (b) with tiling

Fig. 1: 3D plot of pixel values of image "Kodim04" and clustered color vectors (gray points and white points).

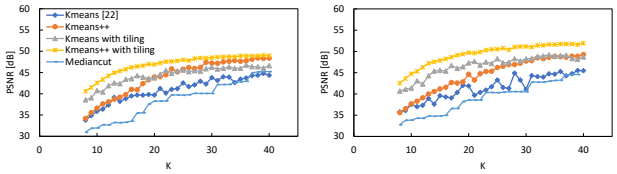
The tiling technique limits the dynamic range of input images to improve the performance in the grayscale bilateral filtering case [17]. For the color image case, the color space of each sub-image is generally smaller than that of the whole image. Therefore, by performing $O(1)$ CBF for each sub-image, $O(1)$ CBF works effectively with fewer dominant color vectors. Also, the clustering result becomes more stable.

Figure 1 shows the images of 8 clustered color vectors without tiling and 8 clustered color vectors for each sub-image using tiling, where we divided the image into 4×4 for tiling. Gray points represent pixel values in the image "Kodim04" and white points represent clustered color vectors. This figure suggests that the use of tiling covers a broader range of cluster points. Table 1 shows the total WSCC for each tile. This table means that the pixel values are dense near the sampling points due to tiling.

Furthermore, the tiling method can accelerate $O(1)$ CBF itself [17]. The tiling method adopts the granularity of the parallelization and improves its efficiency. Also, the locality of data access increases, which improves cache efficiency.

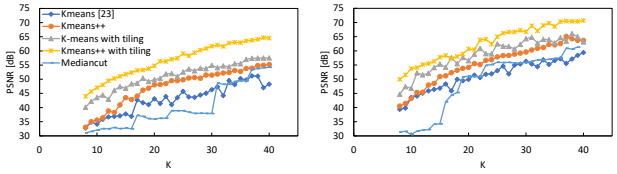
3.3. Subsampling

To accelerate clustering, we subsample an input image for clustering. If we subsample the input image for entire image processing, the quality of filtering becomes obviously degraded. However, we utilize the subsampled image only for clustering. Subsampled and full-sampled 3D image histogram, e.g., Fig. 1, is similar; thus, the side-effect of the subsampling is small. Also, the variance of the tiled sub-image is smaller than the full image; thus, we can keep higher quality with tiling than the non-tiled case.



(a) $\sigma_s = 5, \sigma_r = 30$ (b) $\sigma_s = 5, \sigma_r = 45$

Fig. 2: K w.r.t. PSNR [dB] on Sugimoto et al.



(a) $\sigma_s = 5, \sigma_r = 30$ (b) $\sigma_s = 5, \sigma_r = 45$

Fig. 3: K w.r.t. PSNR [dB] on Nair et al.

4. EXPERIMENTAL RESULTS

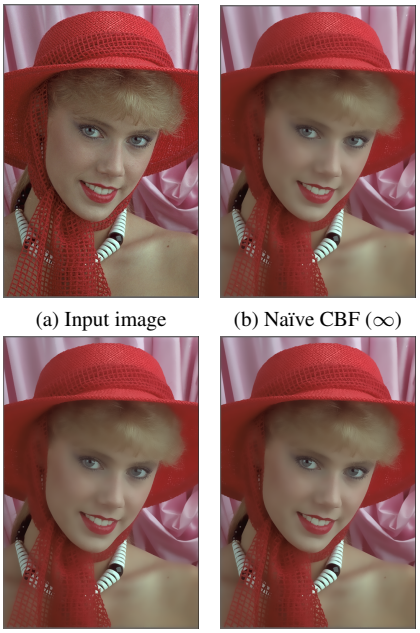
We evaluated the performance of our method for approximation accuracy and stability. All the codes were implemented in C++ and developed on Visual Studio 2017, parallelized by OpenMP and vectorized by AVX on Intel Core i9-9900K @ 3.60GHz. The accuracy was quantified as peak signal-to-noise ratio (PSNR) between the naïve CBF and $O(1)$ CBF. For clustering, we used K-means (conventional), K-means++, K-means with tiling, K-means++ with tiling, and median cut. The median cut is a typical color quantization, e.g., used in GIF image format, and the method does not have randomness. In the experiment, we used $O(1)$ GF [14] for the spatial convolutions. We used the test image Kodak 24 image set (512×768). For tiling, we divided input images into 4×4. For Sugimoto et al. [24], we set a parameter $\lambda = \frac{0.5}{255}$. The kernel radius was $3\sigma_s$ for both methods. We subsampled the input image width and height into $1/2$ before clustering.

Firstly, we compared the approximation accuracy. Figures 2, 3 show the results of accuracy comparison, and plot the relationship between the number of clustered colors K and averaged PSNR with 1000 trials. Note that the number of convolutions is $4K$. From these results, PSNR is slightly improved by K-means++ or tiling. Moreover, it is significantly improved by K-means++ with tiling.

Secondly, we evaluated the stability because the accuracy of $O(1)$ CBF using clustering varies depending on the clustering result. Table 2 shows the minimum, maximum, variance, and average PSNR of 1000 trails of $O(1)$ CBF. We confirm that our approach decreases the variance and improves the average PSNR; thus, the stability is improved.

Thirdly, we visually compared Naïve CBF and $O(1)$ CBF (using K-means++ with tiling). Figure 4 shows the filtered images of each method. We confirm that the approximation of $O(1)$ CBF is sufficiently accurate.

Lastly, we compared the computational time. Table 3 shows the computational time [ms] for each method. The tiling method is faster in full/sub-sampling cases. In addition to the reason described in section 3.2, the overhead of clustering for large numbers of data is considerable.



(a) Input image (b) Naïve CBF (∞)

(c) Sugimoto (44.9 dB) (d) Nair (50.7 dB)

Fig. 4: Visual comparison and PSNR [dB] between naïve and $O(1)$ CBF ($\sigma_s = 5, \sigma_r = 30, K = 15$).

Table 2: PSNR statistics for each method with sub/full sample ($\sigma_s = 5, \sigma_r = 30, K = 15$) (a) K-means, (b) K-means++, (c) K-means with tiling, (d) K-means++ with tiling. * means full-sampled, and no mark means sub-sampled. Note that (a)* is conventional conpetitives [24, 25].

Sugimoto et al.

Statistics\Method	(a)	(b)	(c)	(d)	(a)*	(b)*	(c)*	(d)*
minimum	32.49	37.17	37.65	44.95	32.94	37.83	38.19	44.91
maximum	44.84	44.00	45.76	46.68	45.37	44.56	46.37	46.73
variance	4.75	1.26	2.21	0.06	4.70	1.01	1.48	0.06
average	38.48	40.29	43.05	45.77	39.45	41.77	43.61	45.78

Nair et al.

Statistics\Method	(a)	(b)	(c)	(d)	(a)*	(b)*	(c)*	(d)*
minimum	30.14	30.22	34.94	50.34	30.27	30.55	34.82	50.44
maximum	47.94	47.45	52.35	53.80	48.54	47.53	53.16	53.90
variance	16.84	8.95	10.18	0.24	14.93	3.20	8.49	0.17
average	38.56	42.22	47.20	51.66	40.04	44.05	48.30	51.61

Table 3: Computational time [ms] for each method with sub/full sampling ($\sigma_s = 5, \sigma_r = 30, K = 15$). * means conventional method [24, 25].

$O(1)$ CBF\Method	K-means (a)	K-means++ (b)	K-means with tiling (c)	K-means++ with tiling (d)
Sub-Sugimoto	80	79	56	52
Sub-Nair	120	120	58	58
Full-Sugimoto	*196	218	142	145
Full-Nair	*236	260	159	159

5. CONCLUSION

This paper proposed a method to improve accuracy of $O(1)$ CBF using K-means++ and tiling, and accelerate it by tiling and subsampling. In the experiment, our method showed higher performance in accuracy, stability, and computational time.

6. REFERENCES

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 1998.

[2] M. Zhang and B. K. Gunturk, "Multiresolution bilateral filtering for image denoising," *IEEE Transactions on image processing*, vol. 17, no. 12, pp. 2324–2333, 2008.

[3] S. Dai, M. Han, Y. Wu, and Y. Gong, "Bilateral back-projection for single image super resolution.," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2007, pp. 1039–1042.

[4] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 67, 2008.

[5] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 257–266, 2002.

[6] N. Fukushima, K. Sugimoto, and S. Kamata, "Guided image filtering with arbitrary window function," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[7] T. Matsuo, S. Fujita, N. Fukushima, and Y. Ishibashi, "Efficient edge-awareness propagation via single-map filtering for edge-preserving stereo matching," in *Proc. Three-Dimensional Image Processing, Measurement (3DIPM), and Applications*, 2015.

[8] S. Fujita, T. Matsuo, N. Fukushima, and Y. Ishibashi, "Cost volume refinement filter for post filtering of visual corresponding," in *Proc. Image Processing: Algorithms and Systems XIII*, 2015.

[9] T. Q. Pham and L. J. V. Vliet, "Separable bilateral filtering for fast video preprocessing," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[10] N. Fukushima, S. Fujita, and Y. Ishibashi, "Switching dual kernels for separable edge-preserving filtering," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[11] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," in *Proc. European conference on computer vision (ECCV)*, 2006.

[12] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[13] K. Sugimoto and S. Kamata, "Fast gaussian filter with second-order shift property of dct-5," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2013.

[14] K. Sugimoto and S. Kamata, "Efficient constant-time gaussian filtering with sliding dct/dst-5 and dual-domain error minimization," *ITE Transactions on Media Technology and Applications*, vol. 3, no. 1, pp. 12–21, 2015.

[15] F. Porikli, "Constant time o(1) bilateral filtering," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[16] Q. Yang, N. Ahuja, and K. H. Tan, "Constant time median and bilateral filtering," *International Journal of Computer Vision*, vol. 112, no. 3, pp. 307–318, 2015.

[17] K. Sugimoto, N. Fukushima, and S. Kamata, "200 fps constant-time bilateral filter using svd and tiling strategy," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2019.

[18] Y. Sumiya, N. Fukushima, K. Sugimoto, and S. Kamata, "Extending compressive bilateral filtering for arbitrary range kernel," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2020.

[19] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 24–52, 2009.

[20] W. Tu, Y. Lai, and S. Chien, "Constant time bilateral filtering for color images," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2016.

[21] C. Karam, C. Chen, and K. Hirakawa, "Stochastic bilateral filter for high-dimensional images," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2015.

[22] S. Ghosh and K. N. Chaudhury, "Fast bilateral filtering of vector-valued images," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2016.

[23] M. G. Mozerov and J. van de Weijer, "Global color sparseness and a local statistics prior for fast bilateral filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 12, 2015.

[24] K. Sugimoto, N. Fukushima, and S. Kamata, "Fast bilateral filter for multichannel images via soft-assignment coding," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016.

[25] P. Nair and K. N. Chaudhury, "Fast high-dimensional kernel filtering," *IEEE Signal Processing Letters*, vol. 26, pp. 377–381, 2019.

[26] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[27] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. Geusebroek, "Visual word ambiguity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1271–1283, 2010.

[28] L. Liu, L. Wang, and X. Liu, "In defense of soft-assignment coding," in *Proc. International Conference on Computer Vision (ICCV)*, 2011.

[29] E. J. Nystrom, "Uber die praktische auflosung von integralgleichungen mit anwendungen auf randwertaufgaben," *Acta Math.*, vol. 54, pp. 185–204, 1930.

[30] K. Sugimoto, T. Breckon, and S. Kamata, "Constant-time bilateral filter using spectral decomposition," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2016.

[31] G. Papari, N. Idowu, and T. Varslot, "Fast bilateral filtering for denoising large 3d images," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 251–261, 2017.

[32] A. W. F. Edwards and L. L. Cavalli-Sforza, "A method for cluster analysis," *Biometrics*, vol. 21, no. 2, pp. 362–375, 1965.