Local LUT Upsampling for Acceleration of Edge-preserving Filtering

Hiroshi Tajima, Teppei Tsubokawa, Yoshihiro Maeda and Norishige Fukushima*
Nagoya Institute of Technology, Japan
**https://fukushima.web.nitech.ac.jp/en/*

Keywords: Edge-preserving Filtering, Local LUT Upsampling, Acceleration, Bilateral Filtering, L_0 Smoothing.

Abstract: Edge-preserving filters have been used in various applications in image processing. As the number of pixels of digital cameras has been increasing, the computational cost becomes higher, since the order of the filters depends on the image size. There are several acceleration approaches for the edge-preserving filtering; however, most approaches reduce the dependency of filtering kernel size to the processing time. In this paper, we propose a method to accelerate the edge-preserving filters for high-resolution images. The method subsamples an input image and then performs the edge-preserving filtering on the subsampled image. Our method then upsamples the subsampled image with the guidance, which is the high-resolution input images. For this upsampling, we generate per-pixel LUTs for high-precision upsampling. Experimental results show that the proposed method has higher performance than the conventional approaches.

1 INTRODUCTION

Edge-preserving filtering smooths images while maintaining the outline in the images. There are various edge-preserving filters for various purposes of image processing, such as bilateral filtering (Tomasi and Manduchi, 1998), non-local means filtering (Buades et al., 2005), DCT filtering (Yu and Sapiro, 2011), BM3D (Dabov et al., 2007), guided image filtering (He et al., 2010), domain transform filtering (Gastal and Oliveira, 2011), adaptive manifold filtering (Gastal and Oliveira, 2012), local Laplacian filtering (Paris et al., 2011), weighted least square filtering (Levin et al., 2004), and L_0 smoothing (Xu et al., 2011). The applications of the edge-preserving filters include noise removal (Buades et al., 2005), outline emphasis (Bae et al., 2006), high dynamic range imaging (Durand and Dorsey, 2002), haze removing (He et al., 2009), stereo matching (Hosni et al., 2013; Matsuo et al., 2015), free viewpoint imaging (Kodera et al., 2013), depth map enhancement (Matsuo et al., 2013).

The computational cost of edge-preserving filtering is the main issue. There are several acceleration approaches for each filter, such as bilateral filtering (Durand and Dorsey, 2002; Yang et al., 2009; Adams et al., 2010; Chaudhury et al., 2011; Chaudhury, 2011; Chaudhury, 2013; Sugimoto and Kamata, 2015; Sugimoto et al., 2016; Ghosh et al., 2018; Maeda et al., 2018b; Maeda et al., 2018a), non-local means filtering (Adams et al., 2010; Fukushima et al., 2015), local Laplacian filtering (Aubry et al., 2014), DCT filtering (Fujita et al., 2015), guided image filtering (Murooka et al., 2018; Fukushima et al., 2018b) and weighted least square filtering (Min et al., 2014). The computational time of each filter, however, depends on image resolution, and it is rapidly increasing, e.g., smartphone's cameras even have 12M pixels. For such a high-resolution image, we require more acceleration techniques.

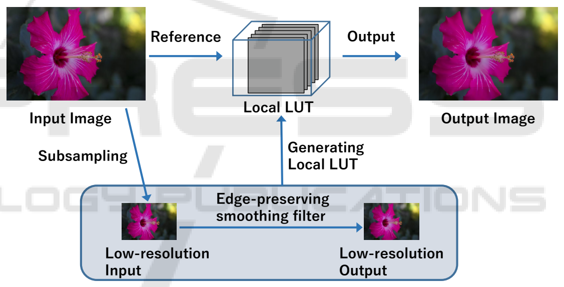


Figure 1: Local LUT upsampling: an input image is downsampled and then the image is smoothed by arbitrary edge-preserving filtering. Next, we create per-pixel LUT by using the correspondence between subsampled input and output images. Finally, we convert the input image into the approximated image by the LUT.

Processing with subsampling and then upsampling is the most straightforward approach to accelerate image processing. This approach dramatically reduces processing time; however, the accuracy of the

Tajima, H., Tsubokawa, T., Maeda, Y. and Fukushima, N.
Local LUT Upsampling for Acceleration of Edge-preserving Filtering.
DOI: 10.5220/0007364601310138
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 131-138
ISBN: 978-989-758-354-4

approximation is also significantly decreased. Subsampling loses high-frequency and large edges in images; hence, the resulting images also lose the information. Different from super-resolution problems, subsampling/upsampling for acceleration can utilize a high-resolution input image as a guidance signal. Joint bilateral upsampling (Kopf et al., 2007) and guided image upsampling (He and Sun, 2015) utilize the high-resolution image for high-quality upsampling as extensions of joint bilateral filtering (Petschnigg et al., 2004; Eisemann and Durand, 2004). However, both upsampling are specific for accelerating bilateral filtering and guided image filtering. Note that More specific upsampling, e.g., depth map upsampling (Fukushima et al., 2016), improve the upsampling quality.

In this paper, we propose a method to accelerate arbitrary edge-preserving filters by using subsampling and upsampling with guidance high-resolution images. We named the proposed method *local LUT upsampling*. Figure 1 indicates the overview of the proposed method. In the proposed, edge-preserving filtering, which has the highest cost in the processing, is performed in the downsampled domain for acceleration. Then our method utilizes high-resolution information for accurate upsampling.

2 RELATED WORK

In this section, we review two edge-preserving filters used in the section of experimental results, i.e., bilateral filtering and L_0 smoothing. Also, we introduce guided image upsampling, which is an acceleration method for edge-preserving filtering.

2.1 Bilateral Filtering

Bilateral filtering is a standard edge-preserving filtering, and it is one of the finite impulse response (FIR) filters. The processing of the bilateral filter for an input image $\boldsymbol{I}$ can be expressed as:

$$\boldsymbol{J_p} = \frac{\sum_{\boldsymbol{q}\in N_{\boldsymbol{p}}} f_{\mathrm{bf}}(\boldsymbol{p},\boldsymbol{q})\boldsymbol{I_q}}{\sum_{\boldsymbol{q}\in N_{\boldsymbol{p}}} f_{\mathrm{bf}}(\boldsymbol{p},\boldsymbol{q})}, \quad (1)$$

where $\boldsymbol{J}$ is an output image of the bilateral filter, $\boldsymbol{p}$ and $\boldsymbol{q}$ are target and reference pixels, respectively. $N_{\boldsymbol{p}}$ is a set of the neighboring pixel of the target pixel $\boldsymbol{p}$. f_{bf} represents a weight function, and it is defined as:

$$f_{\mathrm{bf}}(\boldsymbol{p},\boldsymbol{q}) = \exp\left(\frac{\|\boldsymbol{p}-\boldsymbol{q}\|_2^2}{-2\sigma_s^2}\right)\exp\left(\frac{\|\boldsymbol{I_p}-\boldsymbol{I_q}\|_2^2}{-2\sigma_c^2}\right), \quad (2)$$

where $\|\boldsymbol{p}-\boldsymbol{q}\|_2^2$ and $\|\boldsymbol{I_p}-\boldsymbol{I_q}\|_2^2$ show the difference of the spatial distance and luminance value between the target pixel and the reference pixel, respectively. σ_s and σ_c are smoothing parameters for the Gaussian distributions of the spatial distance and luminance difference, respectively. The bilateral filter smooths images with the range weight in addition to the spatial weight. As a result, a large weight is given to a reference pixel whose distance and luminance from target pixel is close.

2.2 L_0 Smoothing

L_0 smoothing can emphasize rough edges and can reduce fine edges. The L_0 smoothing smooths weak edge-parts in images more strongly than the bilateral filtering. Therefore, it is useful for extracting edges and generating non-photorealistic effects. An output is obtained by solving the following equation:

$$\min_{J,h,v}\left\{\sum_{\boldsymbol{p}}(\boldsymbol{J_p}-\boldsymbol{I_p})^2+\lambda C(h,v) + \beta((\partial_x\boldsymbol{J_p}-h_p)^2+(\partial_y\boldsymbol{J_p}-v_p)^2)\right\}, \quad (3)$$

where $\boldsymbol{I}$ and $\boldsymbol{J}$ are input and output images, respectively. h_p, v_p are auxiliary variables corresponding to $\partial_x\boldsymbol{J_p}$ and $\partial_y\boldsymbol{J_p}$, where $\boldsymbol{p}$ is a target pixel coordinate, respectively. $C(h,v)$ is the number of $\boldsymbol{p}$ that $|h_p|+|v_p| \neq 0$, and β is determined from the gradient of (h,v). In order to solve Eq. (3), it is divided into two sub-problems. The first is computing $\boldsymbol{J}$. Excluding the term not including $\boldsymbol{J}$ from Eq. (3), the following equation is obtained:

$$E = \sum_p \left(\boldsymbol{J}_p-\boldsymbol{I}_p\right)^2 + \beta\left((\partial_x\boldsymbol{J}_p-h_p)^2+(\partial_y\boldsymbol{J}_p-v_p)^2\right). \quad (4)$$

$\boldsymbol{J}$ is obtained by minimizing equation (4). In this method, derivative operators are diagonalized after the fast Fourier transform (FFT) for speedup:

$$\boldsymbol{J} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\boldsymbol{I})+\beta(\mathcal{F}(\partial_x)^*\mathcal{F}(h)+\mathcal{F}(\partial_y)^*\mathcal{F}(v))}{\mathcal{F}(\boldsymbol{I})+\beta(\mathcal{F}(\partial_x)^*\mathcal{F}(\partial_x)+\mathcal{F}(\partial_y)^*\mathcal{F}(\partial_y))}\right), \quad (5)$$

where $\mathcal{F}$ is the FFT operator and * denotes the complex conjugate. The second is computing (h,v). The objective function for (h,v) is defined as follows:

$$E' = \sum_p \min_{h_p,v_p}\left\{(h_p-\partial_x\boldsymbol{J_p})^2+(v_p-\partial_y\boldsymbol{J_p})^2+\frac{\lambda}{\beta}H(|h_p|+|v_p|)\right\}, \quad (6)$$

where $\boldsymbol{H}(|h_p|+|v_p|)$ is a binary function returning 1, if $|h_p|+|v_p| \neq 0$ and 0 otherwise. When Eq. (6) is rewritten to the function E_p of each pixel $\boldsymbol{p}$, the following expression is obtained.

$$E_p = (h_p-\partial_x\boldsymbol{J_p})^2+(v_p-\partial_y\boldsymbol{J_p})^2+\frac{\lambda}{\beta}H(|h_p|+|v_p|), \quad (7)$$

```
Algorithm 1: $L_0$ Gradient Minimization.
Input: image $I$, smoothing weight $\lambda$, parameters $\beta_0$, $\beta_{max}$, and rate $\kappa$
  Initialization: $J \leftarrow I, \beta \leftarrow \beta_0, i \leftarrow 0$
  repeat
    With $J^{(i)}$, solve for $h_p^{(i)}$ and $v_p^{(i)}$ in Eq. (8)
    With $h^{(i)}$ and $v^{(i)}$, solver for $J^{(i+1)}$ with Eq. (5)
    $\beta \leftarrow \kappa\beta, i++$
  until $\beta \geq \beta_{max}$
Output: result image $J$
```

which reaches its minimum E_p^* under the condition:

$$(h_p, v_p) = \begin{cases} (0,0) & ((\partial_x \boldsymbol{J_p})^2 + (\partial_y \boldsymbol{J_p})^2 \leq \lambda/\beta) \\ (\partial_x \boldsymbol{J_p}, \partial_y \boldsymbol{J_p}) & (otherwise.) \end{cases} \quad (8)$$

After computing the minimum energy E_p^* for each pixel $\boldsymbol{p}$, calculating $\sum_p E_p^*$ becomes the global optimum for Eq. (6). The alternating minimization algorithm is sketched in Algorithm 1. Parameter β is automatically adapted for each iteration. β is starting from a small value β_0, and then it is multiplied by κ each time.

2.3 Guided Image Upsampling

We now summarize guided image upsampling. The upsampling is an extension of the guided image filtering. The filtering converts local patches in an input image by a linear transformation of a guidance image. Let the guide signal be G, and it is possible to be $G = I$, where I is an input image. The output patch $J_{\boldsymbol{p}}$ should be satisfied for the following relation:

$$J_{\boldsymbol{p}} = a_{\boldsymbol{k}} G_{\boldsymbol{p}} + b_{\boldsymbol{k}}, \forall \boldsymbol{p} \in \omega_{\boldsymbol{k}}, \quad (9)$$

where $\boldsymbol{k}$ indicates a center position of a rectangular patch $\omega_{\boldsymbol{k}}$, and $\boldsymbol{p}$ indicates a position of a pixel in the patch. $a_{\boldsymbol{k}}$ and $b_{\boldsymbol{k}}$ are coefficients for the linear transformation. The equation represents the coefficients linearly convert the guide signals in a patch.

The coefficients are calculated by a linear regression of the input signal I and Eq. (9),

$$\arg\min_{a_k, b_k} = \sum_{\boldsymbol{p} \in \omega_{\boldsymbol{k}}} ((a_{\boldsymbol{k}} J_{\boldsymbol{p}} + b_{\boldsymbol{k}} - I_{\boldsymbol{p}})^2 + \varepsilon a_{\boldsymbol{k}}^2). \quad (10)$$

The coefficients are estimated as follows:

$$a_{\boldsymbol{k}} = \frac{cov_{\boldsymbol{k}}(G, I)}{var_{\boldsymbol{k}}(G) + \varepsilon}, \quad b_{\boldsymbol{k}} = \bar{I}_{\boldsymbol{k}} - a_{\boldsymbol{k}} \bar{G}_{\boldsymbol{k}}, \quad (11)$$

where ε indicates a parameter of smoothing degree. $\bar{\cdot}_{\boldsymbol{k}}$, $cov_{\boldsymbol{k}}$ and $var_{\boldsymbol{k}}$ indicate the mean, variance, and covariance values of the patch $\boldsymbol{k}$. The coefficients are overlapping in the output signals; thus, these coefficients are averaged;

$$\bar{a}_{\boldsymbol{i}} = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{p}}} a_{\boldsymbol{k}}, \quad \bar{b}_{\boldsymbol{i}} = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{p}}} b_{\boldsymbol{k}}, \quad (12)$$

where $|\cdot|$ indicates the number of elements in the set. Finally, the output is calculated as follows:

$$J_{\boldsymbol{i}} = \bar{a}_{\boldsymbol{i}} G_{\boldsymbol{i}} + \bar{b}_{\boldsymbol{i}}. \quad (13)$$

For color filtering, let input, output and guidance signals be $\boldsymbol{I} = \{I^1, I^2, I^3\}$, J^n $(n = 1, 2, 3)$, and $\boldsymbol{G}$, respectively. The per-channel output is defined as:

$$J_{\boldsymbol{p}}^n = \bar{\boldsymbol{a}}_{\boldsymbol{p}}^{n\,\mathrm{T}} \boldsymbol{G}_{\boldsymbol{p}} + \bar{b}_{\boldsymbol{p}}^n, \quad (14)$$

$$\bar{\boldsymbol{a}}_{\boldsymbol{p}}^n = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{p}}} \boldsymbol{a}_{\boldsymbol{k}}^n, \quad \bar{b}_{\boldsymbol{p}}^n = \frac{1}{|\omega|} \sum_{\boldsymbol{k} \in \omega_{\boldsymbol{p}}} b_{\boldsymbol{k}}^n. \quad (15)$$

The coefficients $\boldsymbol{a}_{\boldsymbol{k}}^n$, $b_{\boldsymbol{k}}^n$ are obtained as follows:

$$\boldsymbol{a}_{\boldsymbol{k}}^n = \frac{cov_{\boldsymbol{k}}(\boldsymbol{G}, I^n)}{var_{\boldsymbol{k}}(\boldsymbol{G}) + \varepsilon \boldsymbol{E}}, \quad b_{\boldsymbol{k}}^n = \bar{I}_{\boldsymbol{k}}^n - \boldsymbol{a}_{\boldsymbol{k}}^{n\,\mathrm{T}} \bar{\boldsymbol{G}}_{\boldsymbol{k}}, \quad (16)$$

where $\boldsymbol{E}$ is an identity matrix. When the output signal is a color image, $cov_{\boldsymbol{k}}$ is the covariance matrix of the patch in I and $\boldsymbol{G}$. Also, $var_{\boldsymbol{k}}$ is the variance of the R, G, and B components, which will be covariance matrix, in the patch of $\boldsymbol{G}$. The division of the matrix is calculated by multiplying the inverse matrix of the denominator from the left. We use box filtering for the calculation of per pixel mean, variance, and covariance. The fast implementation of the box filtering (Fukushima et al., 2018a) accelerates the filtering.

The guided image upsampling is an extension of guided image filtering and the upsampling is expressed as:

$$J_{\boldsymbol{p}}^n = S_s^{-1}(S_s(\bar{\boldsymbol{a}}_{\boldsymbol{p}}^n))^{\mathrm{T}} \boldsymbol{G}_{\boldsymbol{p}} + S_s^{-1}(S_s(\bar{b}_{\boldsymbol{p}}^n)), \quad (17)$$

where $S(\cdot)$ and $S^{-1}(\cdot)$ indicate subsampling and upsampling operators, respectively. The upsampling computes coefficients of $\bar{\boldsymbol{a}}_{\boldsymbol{p}}$ and $\bar{b}_{\boldsymbol{p}}$ in the subsampled domain, and then upsamples the coefficients. Notice that we should keep high-resolution signal of $\boldsymbol{G}$.

The coefficients in the downsampled domain is computed as follows:

$$\bar{\boldsymbol{a}'}_{\boldsymbol{p}_\downarrow}^n := S(\bar{\boldsymbol{a}}_{\boldsymbol{p}}^n), \quad \bar{b'}_{\boldsymbol{p}_\downarrow}^n := S(\bar{b}_{\boldsymbol{p}}^n), \quad (18)$$

$$\bar{\boldsymbol{a}'}_{\boldsymbol{p}_\downarrow}^n = \frac{1}{|\omega'|} \sum_{\boldsymbol{k}_\downarrow \in \omega'_{\boldsymbol{p}}} \boldsymbol{a}_{\boldsymbol{k}_\downarrow}^n, \quad \bar{b}_{\boldsymbol{p}_\downarrow}^n = \frac{1}{|\omega'|} \sum_{\boldsymbol{k}_\downarrow \in \omega'_{\boldsymbol{p}}} b_{\boldsymbol{k}_\downarrow}^n, \quad (19)$$

$$\boldsymbol{a'}_{\boldsymbol{k}_\downarrow}^n = \frac{cov_{\boldsymbol{k}_\downarrow}(\boldsymbol{G'}, I'^n)}{var_{\boldsymbol{k}_\downarrow}(\boldsymbol{G'}) + \varepsilon \boldsymbol{E}}, \quad b'^n_{\boldsymbol{k}_\downarrow} = \bar{I'}_{\boldsymbol{k}_\downarrow}^n - a'^{n\,\mathrm{T}}_{\boldsymbol{k}_\downarrow} \bar{\boldsymbol{G}'}_{\boldsymbol{k}_\downarrow}, \quad (20)$$

where $\boldsymbol{p}_{\downarrow}$ indicates pixel coordinate in the downsampled domain. Also, $\boldsymbol{a}'$ and b' are coefficients in the downsampled domain, respectively $\boldsymbol{G}' := S(\boldsymbol{G})$ and $I' := S(I)$ represent downsampled input and guidance images, respectively. The filtering radius is reduced according to the rate of subsampling; thus, ω' indicates a small rectangular patch, which is reshaped to fit the rate of subsampling.

3 PROPOSED METHOD

We propose an upsampling method guided by a high-resolution image for various edge-preserving filtering. The proposed method generates a per-pixel look-up table (LUT), which maps pixel values of the input image to that of the output image of the edge-preserving filtering. If we have the result of edge-preserving filtering, we can easily transform the input image into the edge-preserving result by one-to-one mapping. We generate the mapping function in the subsampled image domain for acceleration. The proposed method generates per-pixel LUTs for the function from the correspondence between a subsampled input and its filtering result.

Luminance values are highly correlated between pixels and its neighboring region in the image. Also, the correlation between the output of edge-preserving filtering and the input image becomes high. Therefore, in this method, a LUT for each pixel is created from a pair of low-resolution input and output images. Then, we refer the subsampled LUT to generate the high-resolution edge-preserving image. Initially, an input image $\boldsymbol{I}$ is subsampled to generate a low-resolution input image $\boldsymbol{I}_{\downarrow}$:

$$\boldsymbol{I}_{\downarrow} = S_s(\boldsymbol{I}), \tag{21}$$

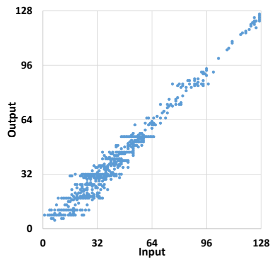
where S_s is a subsampling operator. Then, edge-preserving filter is applied for $\boldsymbol{I}_{\downarrow}$ and the output $\boldsymbol{J}_{\downarrow}$ is obtained as follows:

$$\boldsymbol{J}_{\downarrow} = F(\boldsymbol{I}_{\downarrow}), \tag{22}$$

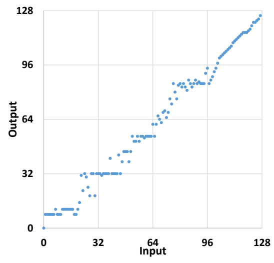
where F is an operator of arbitrary edge-preserving filtering. Then, we count frequency of one-to-one mapping between the images $\boldsymbol{J}_{\downarrow}$ and $\boldsymbol{I}_{\downarrow}$. Let $f_{\boldsymbol{p}}(s,t)$ be a frequency map on a pixel $\boldsymbol{p}$, where s and t are an intensity value on image $\boldsymbol{I}_{\downarrow}$ and $\boldsymbol{J}_{\downarrow}$, respectively. The frequency is counted by gathering around a pixel. The map f is defined as follows:

$$f_{\boldsymbol{p}}(s,t) = \sum_{\boldsymbol{q}\in W_{\boldsymbol{p}}} \delta(S_c(\boldsymbol{I}_{\boldsymbol{q}\downarrow}),s)\delta(S_c(\boldsymbol{J}_{\boldsymbol{q}\downarrow}),t), \tag{23}$$

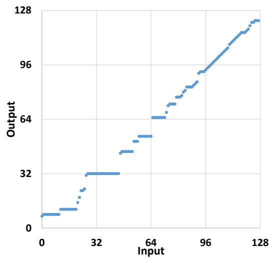
$$S_c(x) = \lfloor x/l \rfloor \tag{24}$$



(a) Frequency map



(b) Winner-takes-all



(c) Dynamic programming

Figure 2: Frequency map $f_{\boldsymbol{p}}(s,t)$ and its results of winner-takes-all and dynamic programming approaches for local LUT on $\boldsymbol{p}$. The number of bins is 128, i.e., quantization level is $l=2$. The radius is $r=20$ for visibility of this figure. In the experiment, small radius setting, e.g., $r=2,3$ have better performance.

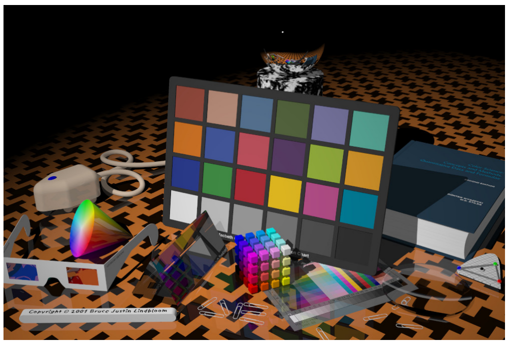
where $\delta(a,b)$ indicates the Kronecker delta function. $\boldsymbol{p}=(x,y)$ indicates a pixel position in the images of $\boldsymbol{I}_{\downarrow}$ and $\boldsymbol{J}_{\downarrow}$. $\boldsymbol{q}$ is a neighboring pixel around $\boldsymbol{p}$. $W_{\boldsymbol{p}}$ indicates the set including $\boldsymbol{q}$. $S_c(x)$ is a quantization function, which is divided by l. The output values have $256/l$ candidates. The redaction also accelerates the processing. We call the number of intensity candidates as the number of bins.

Next, we determine a local LUT on a pixel $\boldsymbol{p}$. With the simplest solution, we select the maximum frequency argument from the f by the winner-takes-all approach (WTA). The LUT under the downsampled domain is defined as follows:

$$T_{\boldsymbol{p}\downarrow}[s] = \arg\max_{t} f(s,t) \tag{25}$$

The operator $[\cdot]$ is an array operator in programming languages, such as C programming language. We cannot determine the value where $f(s,t)=0|_{\forall t}$. In this case, we linearly interpolate the values by the nearest non-zero values around s. For boundary condition, we set $T_{\boldsymbol{p}\downarrow}[0]=0$ and $T_{\boldsymbol{p}\downarrow}[255]=255$, when $f(s,t)=0|_{\forall t}$. The solution cannot keep monotonicity for the LUT. The fact sometimes generates negative edges.

To keep the monotonicity, we can solve the problem by using dynamic programming (DP). We assume that $T_{\boldsymbol{p}}[0]=0$ and $T_{\boldsymbol{p}}[n] >= T_{\boldsymbol{p}}[n]$. We define a cost function $C(s,t)$ for the DP under this condition:


(a) artificial (3072×2048)

(b) tree (6088×4550)

(c) bridge (2749×4049)

(d) cathedral (2000×3008)

(e) fireworks (3136×2352)

Figure 3: High-resolution test images.

$$C(s,t) = \max\Big(C(s-1,t-1) + O, C(s-1,t)), C(s,t-1)\Big) + f(s,t) \tag{26}$$

where $C(0,0) = f(0,0)$, $C(s,t) = 0 \quad (s < 0 \vee t < 0)$. The parameter O represent an offset value to enforce diagonal matching. The cost function can be recursively updated. After filling the cost function, then we trace back the function to determine a solid pass.

$$T_{\boldsymbol{p}\downarrow} = \underset{\boldsymbol{pass}}{\arg \text{DP}} \quad C(s,t), \tag{27}$$

where ***pass*** shows the tracing back pass of the cost matrix C by using DP. Figure 2 shows the frequency map and the resulting LUTs obtained by WTA and DP. After WTA and DP, the matrix indicates one-to-one mapping. We call this per-pixel LUT $T_{\boldsymbol{p}}$ as local LUT.

The LUT has three-dimensional information, such as $n, \boldsymbol{p} = (x,y)$; however, all elements are subsampled. It has not enough to map an input image into a high-resolution image. Therefore, we upsample the LUT by using tri-linear upsampling. The upsampling is defined as follows:

$$T_{\boldsymbol{p}} = S_c^{-1}(S_s^{-1}(T_{\boldsymbol{p}\downarrow})), \tag{28}$$

where S_c^{-1} and S_s^{-1} is upsampling operator for the intensity and spatial domain, respectively. $\boldsymbol{p}$ indicates the image coordinate of pixel in the upsampled domain.

Finally, the output image is referred from the local LUT and input intensity $\boldsymbol{I_p}$. The output is defined by:

$$\boldsymbol{J_p} = T_{\boldsymbol{p}}[\boldsymbol{I_p}]. \tag{29}$$

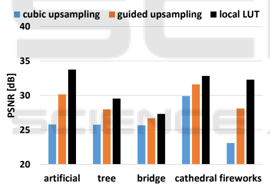
4 EXPERIMENTAL RESULTS

We approximated two edge-preserving filters, such as iterative bilateral filtering and L_0 smoothing by subsampling based acceleration methods. These filters can mostly smooth images; however, the computational cost is high. We compared the proposed method of the local LUT upsampling with the conventional method of the cubic upsampling and guided image upsampling by approximation accuracy and computational time. Also, we compared the proposed method with naïve implementation, which does not subsample images, in computational time. We utilized high-resolution test images, which are shown in Fig. 3[1]. We used two metrics of PSNR and SSIM (Wang et al., 2004) for accuracy evaluation, and we regarded the results of the naïve filtering as ground truth

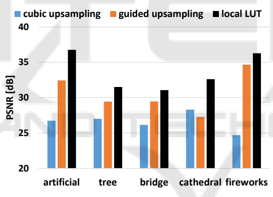
[1]http://imagecompression.info

results. We implemented the proposed method by C++ with OpenMP parallelization. We also used Intel IPP for the cubic upsampling, which is optimized by AVX/AVX2. For guided image upsampling, we also optimized by AVX/AVX2 with OpenCV functions. For downsampling in the proposed method, we use the nearest neighbor downsampling. This downsampling has better performance than the cubic interpolation for our method. We used OpenCV and Intel IPP for the operation with cv::INTER_NN option. The used computer was Intel Core i7 6700 (3.40 GHz) and compiled by Visual Studio 2017. We used $r = 3$ for local LUT upsampling, $r = 10$, $iteration = 10$, $\sigma_s = 10, \sigma_c = 20$ for iterative bilateral filtering, and $\lambda = 0.01$ and $\kappa = 1.5$ for L_0 smoothing.

Figures 4, 5, and 6 show the PSNR/SSIM accuracy and computational time of iterative bilateral filtering and L_0smoothing for large images. The parameters of the proposed method are $l = 2$ and downsampling ratio is 8×8. The computational time is an average of 10 trails. The local LUT upsampling has drastically higher PSNR than the cubic interpolation for each image. Guided image upsampling is slower and lower PSNR than the proposed method. The computation time of the local LUT is $\times 100$ faster than the naïve implementation; however, the cubic upsampling is $\times 10000$ faster. Notice that the code of the proposed method is not vectorized, such as AVX/AVX2; thus, the proposed can speed up by programming, while Intel's code fully optimizes the cubic upsampling. We now optimize the code for the proposed method while reviewing period.

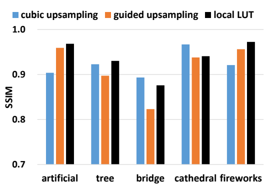


(a) Bilateral filtering

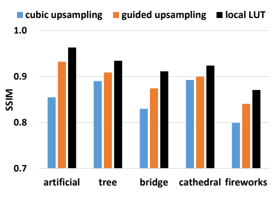


(b) L_0 smoothing

Figure 4: Approximation accuracy of PSNR for each image and method.

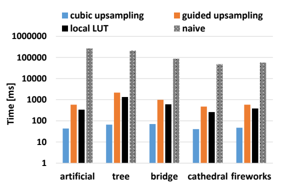


(a) Bilateral filtering

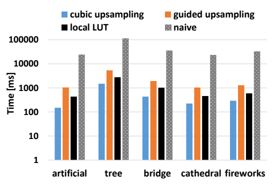


(b) L_0 smoothing

Figure 5: Approximation accuracy of SSIM for each image and method.



(a) Bilateral filtering



(b) L_0 smoothing

Figure 6: Computational time for each image and method.

Figures 7 and 8 show the relationship between spatial/range subsampling ratio and PSNR for iterative bilateral filtering and L_0smoothing. The result shows that $1/4$ range downsampling (64 bin) and 8×8 spatial downsampling do not notable decrease the PSNR quality.

Figures 9 and 10 show the trade-off between the PSNR accuracy and computational time by changing the resizing ratio from 1/16 to 1/1024. We also change the number of bins for local LUT upsampling. These results show that the proposed method has the best trade-off than the cubic interpolation and guided image upsampling.

Figures 11 and 12 depict input image and results of each upsampling method for iterative bilateral filtering and L_0 smoothing, respectively.

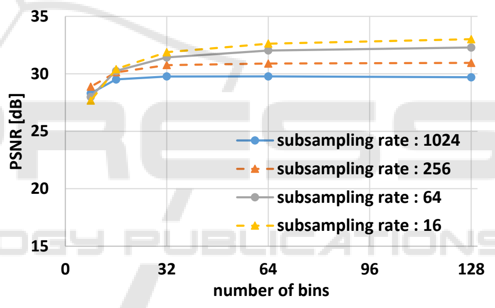


Figure 7: Bilateral filtering result: PSNR w.r.t. the number of bins with changing resizing ratio.

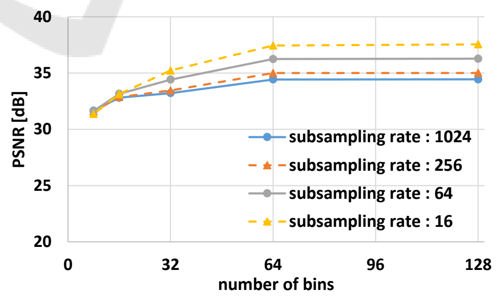


Figure 8: L_0 smoothing result: PSNR w.r.t. the number of bins with changing resizing ratio.

5 CONCLUSIONS

In this paper, we proposed an acceleration method for edge-preserving filtering with image upsampling. We call the upsampling as local LUT upsampling. The lo-

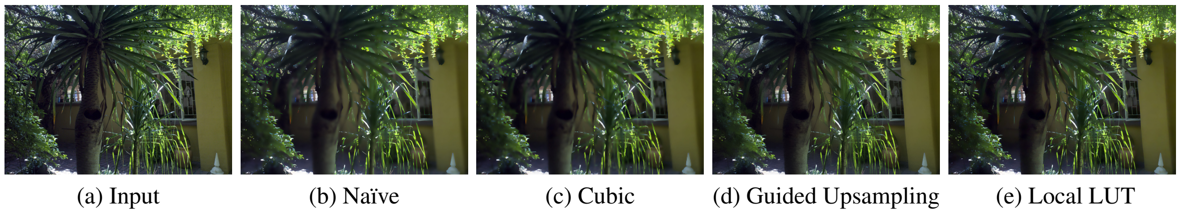


Figure 11: Results of iterative bilateral filtering.

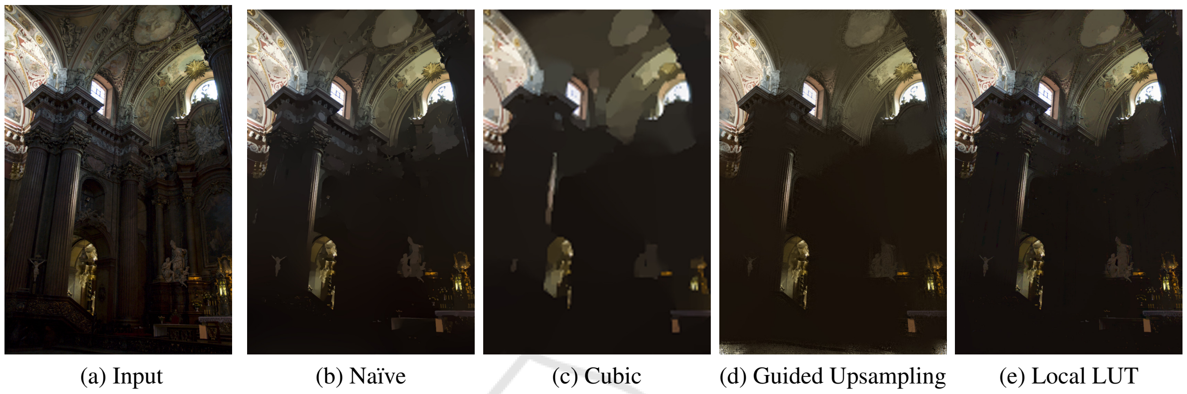


Figure 12: Results of L_0 smoothing.

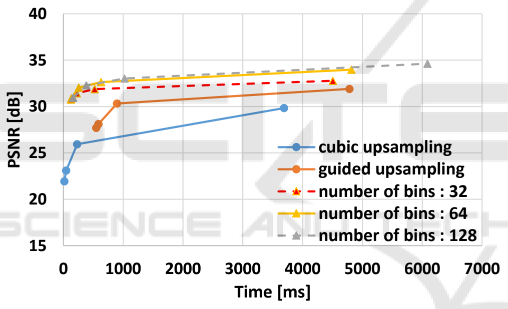


Figure 9: Resizing and changing the number of bins performance in PSNR w.r.t. computational time (bilateral filter). Local LUT samples 32, 64, and 128 bins, respectively.

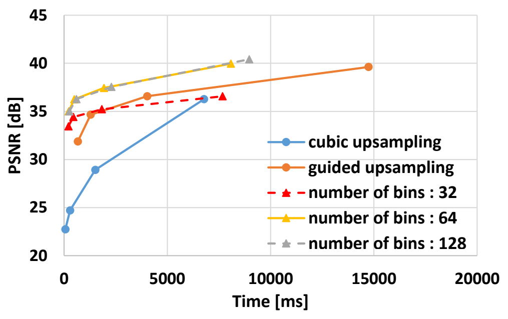


Figure 10: Resizing and changing the number of bins performance in PSNR w.r.t. computational time (L_0 smoothing). Local LUT samples 32, 64, and 128 bins, respectively.

cal LUT upsampling has higher approximation accuracy than the conventional approaches of cubic and guided image upsampling. Also, the local LUT upsampling accelerates $\times 100$ faster than the naïve implementation.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI (JP17H01764, 18K18076).

REFERENCES

Adams, A., Baek, J., and Davis, M. A. (2010). Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2):753–762.

Aubry, M., Paris, S., Hasinoff, S. W., Kautz, J., and Durand, F. (2014). Fast local laplacian filters: Theory and applications. *ACM Transactions on Graphics*, 33(5).

Bae, S., Paris, S., and Durand, F. (2006). Two-scale tone management for photographic look. *ACM Transactions on Graphics*, pages 637–645.

Buades, A., Coll, B., and Morel, J. M. (2005). A non-local algorithm for image denoising. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Chaudhury, K. N. (2011). Constant-time filtering using shiftable kernels. *IEEE Signal Processing Letters*, 18(11):651–654.

Chaudhury, K. N. (2013). Acceleration of the shiftable o(1) algorithm for bilateral filtering and nonlocal means. *IEEE Transactions on Image Processing*, 22(4):1291–1300.

Chaudhury, K. N., Sage, D., and Unser, M. (2011). Fast o(1) bilateral filtering using trigonometric range kernels. *IEEE Transactions on Image Processing*, 20(12):3376–3382.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095.

Durand, F. and Dorsey, J. (2002). Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3):257–266.

Eisemann, E. and Durand, F. (2004). Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3):673–678.

Fujita, S., Fukushima, N., Kimura, M., and Ishibashi, Y. (2015). Randomized redundant dct: Efficient denoising by using random subsampling of dct patches. In *Proc. ACM SIGGRAPH Asia 2015 Technical Briefs*.

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

Fukushima, N., Maeda, Y., Kawasaki, Y., Nakamura, M., Tsumura, T., Sugimoto, K., and Kamata, S. (2018a). Efficient computational scheduling of box and gaussian fir filtering for cpu microarchitecture. In *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2018*.

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

Fukushima, N., Takeuchi, K., and Kojima, A. (2016). Self-similarity matching with predictive linear upsampling for depth map. In *Proc. 3DTV-Conference*.

Gastal, E. S. L. and Oliveira, M. M. (2011). Domain transform for edge-aware image and video processing. *ACM Transactions on Graphics*, 30(4).

Gastal, E. S. L. and Oliveira, M. M. (2012). Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics*, 31(4).

Ghosh, S., Nair, P., and Chaudhury, K. N. (2018). Optimized fourier bilateral filtering. *IEEE Signal Processing Letters*, 25(10):1555–1559.

He, K. and Sun, J. (2015). Fast guided filter. *CoRR*, abs/1505.00996.

He, K., Sun, J., and Tang, X. (2009). Single image haze removal using dark channel prior. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

He, K., Sun, J., and Tang, X. (2010). Guided image filtering. In *Proc. European Conference on Computer Vision (ECCV)*.

Hosni, A., Rhemann, C., Bleyer, M., Rother, C., and Gelautz, M. (2013). Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511.

Kodera, N., Fukushima, N., and Ishibashi, Y. (2013). Filter based alpha matting for depth image based rendering. In *Proc. IEEE Visual Communications and Image Processing (VCIP)*.

Kopf, J., Cohen, M. F., Lischinski, D., and Uyttendaele, M. (2007). Joint bilateral upsampling. *ACM Transactions on Graphics*, 26(3).

Levin, A., Lischinski, D., and Weiss, Y. (2004). Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694.

Maeda, Y., Fukushima, N., and Matsuo, H. (2018a). Effective implementation of edge-preserving filtering on cpu microarchitectures. *Applied Sciences*, 8(10).

Maeda, Y., Fukushima, N., and Matsuo, H. (2018b). Taxonomy of vectorization patterns of programming for fir image filters using kernel subsampling and new one. *Applied Sciences*, 8(8).

Matsuo, T., Fujita, S., Fukushima, N., and Ishibashi, Y. (2015). Efficient edge-awareness propagation via single-map filtering for edge-preserving stereo matching. In *Proc. SPIE*, volume 9393.

Matsuo, T., Fukushima, N., and Ishibashi, Y. (2013). Weighted joint bilateral filter with slope depth compensation filter for depth map refinement. In *Proc. International Conference on Computer Vision Theory and Applications (VISAPP)*.

Min, D., Choi, S., Lu, J., Ham, B., Sohn, K., and Do, M. N. (2014). Fast global image smoothing based on weighted least squares. *IEEE Transactions on Image Processing*, 23(12):5638–5653.

Murooka, Y., Maeda, Y., Nakamura, M., Sasaki, T., and Fukushima, N. (2018). Principal component analysis for acceleration of color guided image filtering. In *Proc. International Workshop on Frontiers of Computer Vision (IW-FCV)*.

Paris, S., Hasinoff, S. W., and Kautz, J. (2011). Local laplacian filters: Edge-aware image processing with a laplacian pyramid. pages 68:1–68:12.

Petschnigg, G., Agrawala, M., Hoppe, H., Szeliski, R., Cohen, M., and Toyama, K. (2004). Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672.

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

Sugimoto, K. and Kamata, S. (2015). Compressive bilateral filtering. *IEEE Transactions on Image Processing*, 24(11):3357–3369.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Proc. International Conference on Computer Vision (ICCV)*, pages 839–846.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Xu, L., Lu, C., Xu, Y., and Jia, J. (2011). Image smoothing via l0 gradient minimization. *ACM Transactions on Graphics*.

Yang, Q., Tan, K. H., and Ahuja, N. (2009). Real-time o (1) bilateral filtering. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 557–564.

Yu, G. and Sapiro, G. (2011). Dct image denoising: A simple and effective image denoising algorithm. *Image Processing On Line*, 1:1.