Local Contrast Enhancement with Multiscale Filtering

Kohei Hayashi*, Yoshihiro Maeda† and Norishige Fukushima*
* Nagoya Institute of Technology, Japan
Web: https://fukushima.web.nitech.ac.jp/en/
† Tokyo University of Science, Japan

***Abstract*—Multiscale processing is fundamental for image processing. Most multiscale processing splits signals into base and detail signals and then manipulates the detail signal. Local Laplacian filtering (LLF), multiscale processing, has an unusual form: manipulation and then separation. This structure brings out the high performance of the filter. However, LLFs are limited to the form of Laplacian pyramids, which requires downsampling to create image pyramids; it causes misalignments around edges. In this study, we break away from the pyramid and extend the interpretation to multiscale filters that locally enhance the contrast. We propose a local difference of Gaussian (DoG) filter (LDF) for the solution. DoG filtering can remove the downsampling in the filter. Moreover, we further improve the computational speed using the precomputing technique introduced in LLF. Furthermore, to accelerate it, we utilize $O(1)$ Gaussian filtering for large-scale filtering. Experimental results show that the proposed method suppresses overshoot and undershoot around sharp edges caused by the misalignment. Furthermore, the proposed method can keep computational speed even when the convolutional radius is large by $O(1)$ Gaussian filtering.**

I. Introduction

When capturing images, the contrast of the subject cannot be perfectly transferred due to diffraction of light, aberrations, surface reflections, and other physical characteristics of light and lenses. In particular, contrast is low in regions of high spatial frequency. The modulation transfer function expresses this relationship. Therefore, the image is multiscaled, and a nonlinear contrast transformation function enhances the contrast. In this study, the value referenced by the contrast transformation function is changed for each pixel to improve the local contrast. By improving the contrast at a specific spatial frequency, a locally good-looking image is generated, and by multiscaling the image and retaining only the coefficients of the relevant pixel among the coefficients after processing, more precise edge information can be obtained than by performing contrast transformation after multiscaling. In this way, an image with improved contrast at all positions in the image is obtained by processing each pixel so that the modulation transfer function is changed for each pixel.

Manipulation of images with multiscale analysis is a challenging task. For multiscale processing, an input image is decomposed into multiple layers by various methods: Gaussian and Laplacian pyramids [1], wavelet transform [2], and difference of Gaussians (DoG) form the scale-space analysis [3]. Multiscale processing manipulates coefficients of pyramid, wavelet, or scale-space representations. Then, the layered signals are collapsed to obtain an output image.

Unsharp masking is a straightforward multiscale process. The method has two layers: a base layer and a detail layer. The base layer is a Gaussian/box-filtered image, and the detail layer is the subtraction of the base layer from the input image. The detail layer is linearly multiplied to control the enhancement level. However, this approach produces large halos.

The Laplacian pyramid is utilized to extend two layers to multiple layers. Pyramid-based processing directly enhances the pyramid coefficient [4]–[6]. Instead of linear enhancement of coefficients, S-tone or Gaussian weighting is used to suppress large amplitude. In the Gaussian pyramid-based approach, gradients in pyramid images are optimized to scale factors to avoid halos [7].

The signal decomposition is extended to have an edge-preserving property. Seminal work uses bilateral filtering [8] to generate base signals in two-layer decomposition [9] instead of linear filtering. The cost-consuming part of bilateral filtering is accelerated by Fast Fourier transform (FFT) [9], and further accelerated by following accelerated methods [10], [11]. The two-layer decomposition based on bilateral filtering is extended to multiple layers using iterative bilateral filtering. Multiscale bilateral decomposition is accelerated by skipped filtering [12]. Skipping is further accelerated by hardware [13]. This approach has difficulty in determining parameters to suppress halo and other edge-reversal artifacts. The following work computes a multiscale edge-preserving decomposition with a least-square scheme instead of bilateral filtering [14]. The other extension uses iterative local linear regression filtering [15], similar to guide image filtering [16], [17].

The edge-avoiding wavelet [18], [19] is an edge-preserving extension of the wavelet transform. The method constructs a basis based on the edge content of the images. The wavelet becomes non-linear filtering. Also, iterative bilateral filtering-based decomposition [12] is one of À-Trous wavelet methods [20]. The method is extended to multilateral filtering for the computer graphics context [21].

The previous edge-preserving scale-space methods have difficulty handling each iterative and enhancement processing

This work was supported by JSPS KAKENHI (21H03465, 21K17768) and Environment Research and Technology Development Fund (JPMEERF20222M01) of the Environmental Restoration and Conservation Agency of Japan.

parameter. Local Laplacian filtering (LLF) [22] solves the difficulty. LLF is an extension of the Laplacian pyramid [23] to have an edge-aware pyramid. This approach locally enhances the contrast quickly and generates Laplacian pyramids for each pixel for image detail control. LLF has better visual quality than other multiscale processing; however, LLF has a high processing cost. The following works accelerate the construction per pixel of the Laplacian pyramid: fast LLF [24] and Gaussian Fourier pyramid [25]. Also, LLF is accelerated by FPGA [26]. Edge-aware upsampling can also accelerate filtering, e.g., joint bilateral upsampling [27], guided image upsampling [28], bilateral guided upsampling [29], and local-LUT upsampling [30].

The processing flow is the most significant difference between the previous methods and LLF. Conventional methods, such as linear filter, edge-preserving smoothing, and wavelet methods, all perform the following steps: 1) separation, 2) enhancement, and 3) integration. However, the structure has difficulty controlling the smoothing level of scale-space and amplifying each level. Also, it has drawbacks of edge ambiguity in downsampling, and downsampling-based approaches do not preserve the translation invariance property.

In contrast, LLF is different: 1) local enhancement, 2) separation, and 3) integration, where the order of enhancement and separation is reversed. This processing flow more easily controls the parameters. In this paper, we break away from the concept of pyramids and consider a new edge-preserving scale-space analysis based on the concept of the different processing orders. Moreover, the new scale-space method solves the inevitable aliasing problem of pyramid processing, like the concept of stationary wavelet transforms [31].

The contributions of this paper are as follows:

- We replace the Laplacian pyramid with the multiscale filtering of DoG in LLF to increase the controllability and generality of scale-space named local DoG filter (LDF).
- We accelerate the DoG computation by recent $O(1)$ Gaussian filtering (GF) with a fast LLF formulation to accelerate LDF.

II. Preliminary

A. Gaussian Pyramid and Laplacian Pyramid

In this section, we review the Laplacian pyramid. Laplacian pyramid is used for various applications, such as compression [23], texture synthesis [32], and harmonization [33].

Here, we introduce the Gaussian pyramid. Let I be an input image, and let the Gaussian pyramid be defined as the set of $G_\ell[I]$, where $\ell \in \mathcal{L} = \{0, 1, 2, \ldots, \ell_{\max}\} \subset \mathbb{N}$. The lowest level of the Gaussian pyramid is $G_0[I] = I$, and the other level of pyramids $G_{\ell+1}[I]$ are equal to the downsampled blurred image of $G_\ell[I]$:

$$G_\ell[I] = \mathrm{downsampling}(\mathcal{G}_\sigma * G_{\ell-1}[I]), \tag{1}$$

where $\mathcal{G}_\sigma *$ means Gaussian convolution with standard deviation σ. The image of $G_{\ell+1}[I]$ has half the width and height of $G_\ell[I]$. Usually, the Gaussian convolution for pyramid building is implemented using the binomial distribution with five taps for integer operations, and the regarded standard deviation is about $\sigma = 1$.

The difference between the current-level pyramid and the upsampled next-level pyramid defines the Laplacian pyramid:

$$L_\ell[I] = G_\ell[I] - \mathrm{upsampling}(G_{\ell+1}[I]), \tag{2}$$

where $\mathrm{upsampling}(\cdot)$ is an operator that doubles the width and height of the image size by using a smoothing kernel. Usually, the smoothing kernel is the same kernel as $\mathcal{G}_\sigma$. Note that the highest level of the Laplacian pyramid is $L_{\ell_{\max}}[I] = G_{\ell_{\max}}[I]$.

In the Gaussian pyramid, the filtering parameter of each level, σ, is fixed. Iterative processing and downsampling change each level's actual parameter, i.e., equivalent to the full resolution parameter. The relationship between each scale and actual σ_ℓ at each level is essential for scale-space analysis.

Here, we show the relationships. Iterative Gaussian convolutions can be merged and represented by one large standard deviation value by the semi-group property of Gaussian:

$$\mathcal{G}_{\sigma 1} * \mathcal{G}_{\sigma 2} = \mathcal{G}_\sigma, \quad \sigma = \sqrt{\sigma_1^2 + \sigma_2^2}. \tag{3}$$

The Gaussian convolution is applied for each level. The σ for each level becomes the sum of each variance until the level. Note that the image is downsampled for each level; thus, the supporting distribution is doubled by the level. The actual standard deviation for each level σ_ℓ is defined as follows:

$$\sigma_\ell = \sigma \sqrt{\sum_{l=1}^{\ell} 2^l} \quad (\ell > 0), \quad \sigma_0 = 0, \tag{4}$$

where σ is the base value of a standard deviation for constructing Gaussian and Laplacian pyramids.

B. Manipulating Laplacian Pyramid

We show the example of manipulating the Laplacian pyramid for detail enhancement.

We boost the pyramid coefficients for multiscale detail enhancement except for the coarsest layer of $\sup \mathcal{L}$. We usually use an S-tone-like function to suppress overshoot for boosting, while the most straightforward function is $r(x, 0) = kx$, where $k > 1$ is a constant value. If $k = 1$, there is no boosting, and the resulting image is the input image. Note that the representation of the argument 0 in $r(x, 0)$ is unnecessary for this case; however, it is used to match the form of the latter remapping function for the local Laplacian filtering. The later section refers to the remapping function. Finally, remapped signals are collapsed to obtain the output image:

$$O = L_{\ell_{\max}}[I] + \sum_{\ell \in \mathcal{L} \setminus \{\ell_{\max}\}} r(L_\ell[I], 0). \tag{5}$$

C. Local Laplacian Filtering

We review the basic structure of local Laplacian filtering (LLF). In LLF, an output pixel value O_p at pixel $p = (x, y)$ is defined by constructing a pixel-dependent Laplacian pyramid of the output image, $L[O]_p$, called the local Laplacian pyramid.

Each level ℓ of pyramid $L_\ell[O]_p$ is constructed per pixel. Finally, the output pixel O_p is the summation of the value on p for each level, i.e., collapsing pyramid:

$$O_p = \sum_{\ell \in \mathcal{L}} L_\ell[O]_{p_\ell}, \tag{6}$$

where p_ℓ is the relative position of p for each level ℓ, and becomes sub-pixel. The size of the next level image in the pyramid is half, so the accessing position of pixel p also becomes half. For assessing sub-pixel, we interpolate intensity.

There are three steps to compute the coefficient itself $L_\ell[O]_{p_\ell}$. First, a remapping function $r(\cdot)$ is applied for the input image I and then creates the intermediate image $\tilde{I} = r(I, G_\ell[I]_p)$, where $r(\cdot)$ depends on the coefficient of the Gaussian pyramid at level ℓ at p. We will discuss the details of the remapping function later.

Next, we construct a Laplacian pyramid of the remapped image $L_\ell[\tilde{I}]$, and then we select a pyramid coefficient at p that is $L_\ell[\tilde{I}]_p$ as an output coefficient of $L_\ell[O]_p$. By repeating this process in all positions p and levels ℓ, we can get the Laplacian pyramid of the output pyramid $L[O]$.

Finally, we get the output image by collapsing the pyramid $L[O]$. The computational cost of this process is $O(\sigma|\mathcal{L}|N \log N)$ per pixel, where N is the number of image pixels. The separable Gaussian filtering requires $O(\sigma)$ for each level $|\mathcal{L}|$, and a pyramid requires contains $N \log N$ pixels.

Careful implementation can reduce the processing pyramid size. Let $R \in \mathbb{N}$ be the convolution radius: $R = \lceil n\sigma \rceil, n \in \mathbb{R}$, usually $n = 3.0$. The deep pyramid layer hierarchy requires outside pixels of the shallower layers. The required outside radius at level ℓ is defined as follows:

$$R_l = R \sum_{\ell=0}^{l} 2^\ell, \tag{7}$$

Therefore, we require $2 * R_l + 1$ pixels to build a pyramid. For example, $R_l \in \{1, 3, 7, 15, 31, 63, 127, \ldots\}$. R_l is exponentially increasing; thus, the trimmed pyramid size can rapidly reach the maximum size of N.

The approximated acceleration of LLF, called fast LLF [24], can reduce the order $O(\sigma|\mathcal{L}||\mathcal{I}|)$ per pixel, where $|\mathcal{I}| \in \mathbb{N}$ is the number of approximation samples. The method builds a limited number of pyramids and then blends the pyramid to approximate the result.

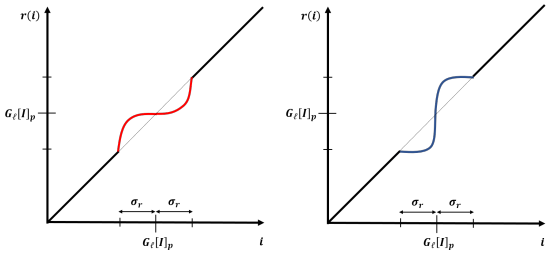
D. Remapping Function

The remapping function is defined as:

$$r(x, g) = x - (x - g)f(x - g), \tag{8}$$

where, x is input signal intensity, and g is the controlling center value. This function enhances/smooths signals. In LLF, $g = G_\ell[I]_p$, where $G_\ell[I]_p$ is a coefficient of a Gaussian pyramid, while f is specifically defined for each application. This paper used the Gaussian function with multiplying factor m as f.

$$f(x - g) = m \exp\left(\frac{(x - g)^2}{-2\sigma_r^2}\right) \tag{9}$$



(a) detail smoothing (b) detail enhancement

Fig. 1: . Remapping functions ($m = 1$).

The sign of m switches enhancing or smoothing signals, and the magnitude control strength of signal control. Figure 1 shows an example of remapping functions.

III. Proposed Method

We extend the concept of LLF to a multiscale filter that manipulates the contrast in advance, named Local DoG Filter (LDF). Furthermore, the LDF is accelerated using the fast LLF form. Moreover, we use $O(1)$ Gaussian filtering (GF) for further acceleration.

A. Definition

Under the concept of LDF, The difference between LDF and LLF is the scale-space structure: pyramid or DoG. In conventional LLF, the image at level $\ell + 1$ is a quarter-size image at ℓ due to downsampling. In contrast, the LDF's image size of each level ℓ is equal due to no downsampling.

First, we build the scale-space blurring representation instead of the Gaussian pyramid. The definition of scale-space $G_\ell[I]$ is as follows:

$$G_\ell[I] = \begin{cases} I & (\ell = 0) \\ \mathcal{G}_{\sigma_\ell} * I & (\text{otherwise}). \end{cases} \tag{10}$$

Here, we redefine $L_\ell[O]$ at a pixel p by using DoG. DoG filtering means the difference between two GFs that have different σ values; thus, $L_\ell[O]_p$ is defined as follows:

$$L_\ell[O]_p = \begin{cases} (\mathcal{G}_{\sigma_\ell} * r(I, G_\ell[I]_p) - \mathcal{G}_{\sigma_{\ell+1}} * r(I, G_\ell[I]_p))_p \\ \qquad\qquad (0 \le \ell \le \ell_{\max} - 1) \\ (\mathcal{G}_{\sigma_\ell} * I)_p \qquad (\ell = \ell_{\max}), \end{cases} \tag{11}$$

where σ_ℓ is the Gaussian scale parameter for each level ℓ. Steps between each level σ_ℓ depend on the applications and have more flexibility than the Laplacian pyramid. For example, we can set a small step when fine scaling is required. In this paper, however, we apply (4) for comparison with conventional LLF.

Finally, an output value O_p is obtained by collapsing the DoG stack:

$$O_p = \sum_{\ell \in \mathcal{L}} L_\ell[O]_p. \tag{12}$$

We do not need upsampling and downsampling, unlike the pyramid-based LLF.

The required number of pixels for a DoG kernel is σ^2, and each level does not have dependency. Moreover, we can compute a pixel using a convolution. In this case, separable filtering is ineffective since separable filtering is used to convolute whole pixels. Therefore, the order is $O(|\mathcal{L}|\sigma^2)$ per pixel. LLF requires $O(\sigma|\mathcal{L}|N\log N)$, larger than our representation.

B. Acceleration

Here, we rewrite the LDF representation for acceleration. We call the accelerated method *fast LDF*. The concept is that $r(I, G_\ell[I]_p)$ in (11) has a limited number of variations. Let a sample i be an integer value, $i \in \mathcal{I} \subset \mathbb{Z}$. First, remapping images $\mathcal{R}_i$ are prepared for each intensity i:

$$\mathcal{R}^i = r(I, i) \quad \forall i \in \mathcal{I}. \tag{13}$$

Second, the DoGs for each remapping image are built:

$$L_\ell[\mathcal{R}_i] = \mathcal{G}_{\sigma_\ell} * \mathcal{R}^i - \mathcal{G}_{\sigma_{\ell+1}} * \mathcal{R}^i \quad \forall i \in \mathcal{I}. \tag{14}$$

Eqs. (13) and (14) do not exist pixel accessing operator p since processing for each pixel is the same. Thus, the GF in (14) can be regarded as the usual GF for entire images. GF can be accelerated using the usual approaches, such as FFT, discussed in the following.

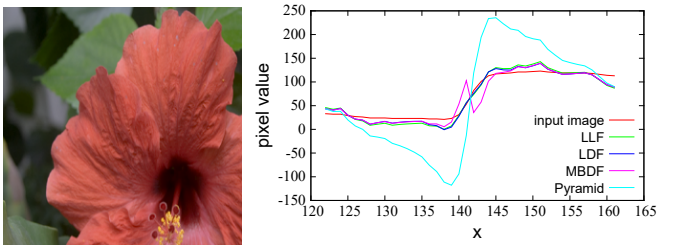
Next, DoGs of the remapped images are combined with the local Laplacian DoG by referring to the Gaussian stacks of (10):

$$L_\ell[O]_p = \sum_{i\in\mathcal{I}} w(G_\ell[I]_p, \mathcal{R}^i_p) L_\ell[\mathcal{R}^i]_p, \tag{15}$$

where w is the weight function for the linear interpolation. Equation (14) sums up each DoG of the remapped image weighted by w. Finally, we can obtain an output value O_p by collapsing the DoG based on (12).

Here, we discuss Gaussian convolution in (14). The order of fast LDF with the naïve implementation of GF is $O(|\mathcal{I}|\sigma^2)$. We perform $|\mathcal{I}|$ times GFs for each pixel, and the order of GF is $O(\sigma^2)$ per pixel. A separable representation can reduce the GF order to $O(\sigma)$. In this case, the LLF order is $O(N|\mathcal{I}|\sigma)$, which is the same as the conventional method of "fast" LLF. The GF is further accelerated by constant-time algorithms, which have two representations: infinite impulse response (IIR) and finite impulse response (FIR).

Typical examples of the IIR filter are the addition model [34] and the multiplication model [35]. The order of these methods is $O(K)$ per pixel, where K is the approximate order. However, the implementation of IIR is not faster than the implementation of recursive FIR [36]. In terms of FIR approximation methods, sliding DCT-based GF has been proposed [37]–[43]. The computational order of this method is also $O(K)$. Currently, sliding DCT-based GF has the best performance in terms of accuracy and processing speed [42]. Therefore, we use the sliding DCT-based GF for the implementation of LDF. The parameter of K is independent of the filtering parameter of σ, so the approximation has a constant-time property for σ. Using constant-time GF, the order of LDF is $O(|\mathcal{I}|K)$, *named fast $O(1)$ LDF* because the algorithm is constant for σ.

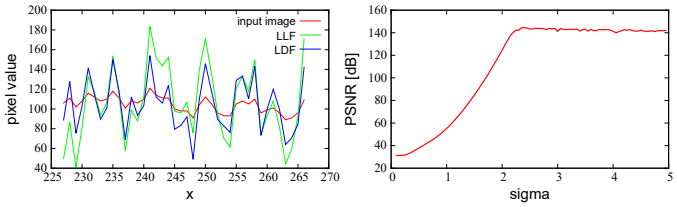


(a) Input Image (b) Profile plot ($\sigma_s = \sqrt{2}$)

(c) Pyr. $\sigma_s = \sqrt{2}$ (d) MBDF $\sigma_s = \sqrt{2}$ (e) LLF $\sigma_s = \sqrt{2}$ (f) LDF $\sigma_s = \sqrt{2}$

(g) Pyr. $\sigma_s = 0.3$ (h) MBDF $\sigma_s = 0.3$ (i) LLF $\sigma_s = 0.3$ (j) LDF $\sigma_s = 0.3$

Fig. 2: Results of each method ($\sigma_r = 20$, $\ell_{\max} = 2$).



(a) Profile plot ($\sigma_s = 0.3$) (b) PSNR

Fig. 3: a) Profile plot of Fig. 2i, Fig 2j, and the input image. b) PSNR between naïve Gaussian filtering and upsampled Gaussian filtering in the Gaussian pyramid.

IV. Experimental Results

We demonstrate the performance of the LDF, fast LDF, and fast $O(1)$ LDF on accuracy and speed. The code is written in C++ with OpenMP parallelization and AVX vectorization. The CPU tested is the Intel Core i9–9980XE (3.0 GHz, 18-cores/36-threads). We remove subnormal numbers in Gaussian distributions for acceleration based on [44].

First, we compare the output images of various methods: pyramid-enhancement of Eq. (5), multiscale bilateral decomposition filter (MBDF) [12], LLF [22] and the proposed LDF. Pyramid and MBDF are preenhancement methods. Figure 2 shows the output of these images and the profile plots of each signal. These images have two σ_s settings: $\sigma_s \in \{0.3, \sqrt{2}\}$. The pyramid has halos, while the other methods solve the problem. MBDF has edge reversals (e.g., the leaf contour in the image's upper left corner), while LLF and LDF solve the problem. LLF and LDF in $\sigma = \sqrt{2}$ have almost the same response, whereas, in the smaller case of $\sigma = 0.3$, the results are different.

Figure 3(a) shows the profile plots then $\sigma = 0.3$. The

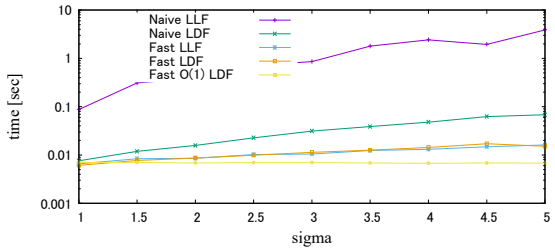


Fig. 4: Processing time ($\ell_{\max} = 2$, $K = 3$). σ_s is the standard deviation of $\ell = 1$, and the radius of the kernel is truncated by $3\sigma_\ell$ for each level in LDF. For conventional LLF, we used a fixed kernel radius, $3\sigma_s$, for each downsampled image. Test images are Kodak 24 images (768×512).

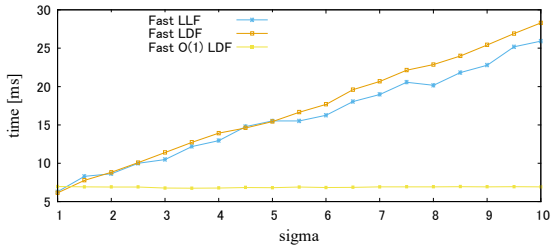


Fig. 5: Processing time of selected methods in Fig. 4. The Y scale is *not the log scale*, and the unit is *ms*.

responses differ, while the case $\sigma = \sqrt{2}$ is almost the same. The difference between high-level signals and low-level signals generates a Laplacian pyramid. The low-level signals are an approximation of Gaussian filtering using the semigroup property and downsampling. Figure 3(b) shows the approximation accuracy of Gaussian filtering. Due to downsampling, aliasing is produced in a small σ_s case; therefore, the accuracy of the approximation is low. The low approximation accuracy produces the difference between LLF and LDF.

Figure 4 shows the computational time (log scale in Y) of each method: naïve LLF [22], fast LLF [24], naïve LDF, fast LDF, fast O(1) LDF. Naïve LLF requires many processing areas, while naïve LDF needs less area; thus, the processing time of LDF is faster than the naïve LLF. For the other case, Fig. 5 shows faster plots with the removal of the logarithmic scale. The computational time of fast LDF and fast LLF is proportional to σ, while fast $O(1)$ LDF exhibits a constant time property. Except for the tiny sigma case (around $\sigma = 1$), the proposed method is faster than the other. Note that due to cache efficiency, when σ is small, the naïve GF implementation is faster than the $O(1)$ GF-based one.

V. Conclusion

This study proposed a local DoG filter (LDF) that creates image pyramids without downsampling. The pure definition of the proposed method generates DoG per pixel; however, it is more efficient than naïve LLF. Furthermore, we accelerate it by changing the form to share the convolution between pixels. Moreover, we use constant-time Gaussian filtering to accelerate large-scale filtering. In our future work, we extend the method to have color space [45] and to perceptually enhance images [46].

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, 1984.

[2] S. Mallat, *A wavelet tour of signal processing*. Elsevier, 1999.

[3] T. Lindeberg, *Scale-space theory in computer vision*. Springer Science & Business Media, 2013, vol. 256.

[4] P. Vuylsteke and E. P. Schoeters, "Multiscale image contrast amplification (musica)," in *Proceedings of SPIE*, vol. 2167, 1994, pp. 551–560.

[5] S. Dippel, M. Stahl, R. Wiemker, and T. Blaffert, "Multiscale contrast enhancement for radiographies: Laplacian pyramid versus fast wavelet transform," *IEEE Transactions on Medical Imaging*, vol. 21, no. 4, 2002.

[6] Y. Li, L. Sharan, and E. H. Adelson, "Compressing and companding high dynamic range images with subband architectures," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 836–844, 2005.

[7] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 249–256, 2002.

[8] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 1998, pp. 839–846.

[9] 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.

[10] 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.

[11] 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.

[12] R. Fattal, M. Agrawala, and S. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections.," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 51, 2007.

[13] N. Fukushima, T. Tsubokawa, and Y. Maeda, "Vector addressing for non-sequential sampling in fir image filtering," in *IEEE International Conference on Image Processing (ICIP)*, 2019.

[14] 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. 1–10, 2008.

[15] B. Gu, W. Li, M. Zhu, and M. Wang, "Local edge-preserving multiscale decomposition for high dynamic range image tone mapping," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 70–79, 2012.

[16] K. He, J. Shun, and X. Tang, "Guided image filtering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, 2013.

[17] 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.

[18] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 1–10, 2009.

[19] Y. Sumiya, H. Kamei, K. Ishikawa, and N. Fukushima, "Vectorized computing for edge-avoiding wavelet," in *International Workshop on Advanced Image Technology (IWAIT)*, vol. 12177, 2022, pp. 23–28.

[20] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*, Springer, 1990, pp. 286–297.

[21] H. Dammertz, D. Sewtz, J. Hanika, and H. P. A. Lensch, "Edge-Avoiding A-Trous Wavelet Transform for fast Global Illumination Filtering," in *Proc. ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics*, The Eurographics Association, 2010.

[22] S. Paris, W. Hasinoff, and J. Kautz, "Local laplacian filters: Edge-aware image processing with a laplacian pyramid," *ACM Trans. on Graphics*, vol. 30, no. 4, 2011.

[23] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, 1983.

[24] M. Aubry, S. Paris, J. Kautz, and F. Durand, "Fast local laplacian filters: Theory and applications," *ACM Trans. on Graphics*, vol. 33, no. 5, 2014.

[25] Y. Sumiya, T. Otsuka, Y. Maeda, and N. Fukushima, "Gaussian fourier pyramid for local laplacian filter," *IEEE Signal Processing Letters*, vol. 29, pp. 11–15, 2022.

[26] S. Khandelwal, Z. Choudhury, S. Shrivastava, and S. Purini, "Accelerating local laplacian filters on fpgas," in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2020, pp. 109–114.

[27] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[28] K. He and J. Sun, "Fast guided filter," *CoRR*, vol. abs/1505.00996, 2015.

[29] J. Chen, A. Adams, N. Wadhwa, and S. W. Hasinoff, "Bilateral guided upsampling," *ACM Transactions on Graphics*, vol. 35, no. 6, 2016.

[30] T. Tsubokawa, H. Tajima, Y. Maeda, and N. Fukushima, "Local look-up table upsampling for accelerating image processing," *Multimedia Tools and Applications*, 2023.

[31] G. P. Nason and B. W. Silverman, "The stationary wavelet transform and some statistical applications," in *Wavelets and Statistics*, Springer, 1995, pp. 281–299.

[32] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. annual conference on Computer graphics and interactive techniques*, 1995, pp. 229–238.

[33] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister, "Multi-scale image harmonization," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 1–10, 2010.

[34] R. Deriche, "Fast algorithms for low-level vision," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 12, pp. 78–87, 1990.

[35] L. J. van Vliet, I. T. Young, and P. W. Verbeek, "Recursive gaussian derivative filters," in *Proc. International Conference on Pattern Recognition (ICPR)*, 1998.

[36] H. Takagi and N. Fukushima, "An efficient description with halide for iir gaussian filter," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2020.

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

[38] 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, pp. 12–21, 2015.

[39] D. Charalampidis, "Recursive implementation of the gaussian filter using truncated cosine functions," *IEEE Transactions on Signal Processing*, vol. 64, 2016.

[40] K. Sugimoto, S. Kyochi, and S. Kamata, "Universal approach for dct-based constant-time gaussian filter with moment preservation," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1498–1502, 2018.

[41] N. Fukushima, Y. Maeda, Y. Kawasaki, *et al.*, "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.

[42] T. Otsuka, N. Fukushima, Y. Maeda, K. Sugimoto, and S. Kamata, "Optimization of sliding-dct based gaussian filtering for hardware accelerator," in *Proc. International Conference on Visual Communications and Image Processing (VCIP)*, 2020.

[43] K. Ishikawa, Y. Sumiya, and N. Fukushima, "Polynomial fitting for period prediction in sliding-dct-based filtering," in *International Workshop on Advanced Image Technology (IWAIT),*, International Society for Optics and Photonics, vol. 12177, SPIE, 2022, pp. 143–148.

[44] Y. Maeda, N. Fukushima, and H. Matsuo, "Effective implementation of edge-preserving filtering on cpu microarchitectures," *Applied Sciences*, vol. 8, no. 10, 2018.

[45] Y. Maeda, N. Fukushima, and T. Hamamoto, "Color transformation for compressive computing in image filtering," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2021.

[46] S. Oishi and N. Fukushima, "Retinex-based relighting for night photography," *Applied Sciences*, vol. 13, no. 3, 2023.