Gaussian Fourier Pyramid for Local Laplacian Filter

Yuto Sumiya, Tomoki Otsuka, Yoshihiro Maeda, *Member, IEEE*, and Norishige Fukushima, *Member, IEEE*

***Abstract*—Multi-scale processing is essential in image processing and computer graphics. Halos are a central issue in multi-scale processing. Several edge-preserving decompositions resolve halos, e.g., local Laplacian filtering (LLF), by extending the Laplacian pyramid to have an edge-preserving property. Its processing is costly; thus, an approximated acceleration of fast LLF was proposed to linearly interpolate multiple Laplacian pyramids. This paper further improves the accuracy by Fourier series expansion, named Fourier LLF. Our results showed that Fourier LLF has a higher accuracy for the same number of pyramids. Moreover, Fourier LLF exhibits parameter-adaptive property for content-adaptive filtering.**

***Index Terms*—Local Laplacian filter, Laplacian pyramid, Fourier series expansion, scale-space.**

I. Introduction

Multi-scale processing is an essential tool for detail manipulation of images and is used for various detail and contrast enhancement: high-dynamic-range imaging [1], detail enhancement [2], and contrast enhancement [3]. Multi-scale processing decomposes an input image into multiple layers using Gaussian and Laplacian pyramids [4], wavelet transform [5], and difference of Gaussians for scale-space analysis [6].

Unsharp masking is one of the simplest multi-scale methods; it consists of two layers: an input and a detail layer. The detail layer is a subtraction of the blurred image from the input image. This simple method produces large halos in steep edges; thus, nonlinear enhancement is used for coefficients to suppress large amplitudes.

The halo problem is resolved by edge-preserving multi-scale decomposition. Instead of linear filtering, bilateral filtering [7] is used to generate a base layer [1]. The bilateral filter is accelerated through fast Fourier transform (FFT) [1], and is further accelerated by state-of-the-art methods [8]–[10]. Next, iterative bilateral filtering is extended to decompose multiple layers [11]; however, it is difficult to determine the parameters in bilateral filtering, which still suffers from halo artifacts. Moreover, multi-scale decomposition is represented by other filters, such as least squares [2] and iterative local linear regression filtering [12], which is similar to guide image filtering [13]. Wavelet-based methods are also proposed for edge-preserving decomposition. Edge-avoiding wavelets [14] construct a basis according to the edge content of the images. Iterative bilateral filtering-based decomposition [11] is one of the À-Trous wavelet methods [15]. The method is extended to multilateral filtering for the computer graphics context [16]. These approaches generate detailed signals and then manipulate the signals through remap functions: linear-tone curve, gamma curve, and S-tone curve.

Local Laplacian filtering (LLF) [17] manipulates the contrast of the input signals using remap functions and then generates detailed signals. LLF locally enhances the image contrast and constructs a Laplacian pyramid for each pixel; thus, LLF can reduce halos with clearer images than the other multi-scale methods. Parameter adaptation for the remap function further improves the quality.

The per-pixel construction of the Laplacian pyramid is costly; thus, an accelerated method is proposed, called fast LLF [18]. Fast LLF reduces the number of Laplacian pyramids by selecting at finite sampled points. Then, the pyramids are linearly interpolated. However, when the number of sample points is insufficient, the approximation accuracy is low. Moreover, the parameter-adaptive version of fast LLF [19] is not an approximation of naïve LLF since the precomputed pyramids are based on a fixed parameter.

For function approximations, Fourier series expansion is a better interpolation method than the linear one. Therefore, this study proposes an approximation for LLF using Fourier series expansion, named *Fourier LLF*. Fourier LLF generates Fourier pyramids, which include cosine and sine pyramids, and then product-sums the pyramids. Fourier LLF improves the accuracy and produces parameter-adaptive functionality. The generated Fourier pyramids are independent of the remap function; thus, we can change the function by switching only the coefficient for the pyramids. Once the pyramids are generated, we can change the parameter by $O(1)$ operation.

The main contributions of this study are as follows:

- We formulate remap functions in LLF using Fourier series expansion, which has higher accuracy than that of the fast LLF [18].
- We verify that Fourier LLF exhibits parameter-adaptive property for the remap function.

Manuscript received June 20, 2021; revised September 25, 2021; accepted October 1, 2021. Date of publication October 19, 2021; date of current version January 20, 2022. This work was supported by JSPS KAKENHI under Grants 18K19813, 19K24368, 21H03465, and 21K17768. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jarvis Haupt. *(Corresponding author: Norishige Fukushima.)*

Yuto Sumiya and Norishige Fukushima are with the Nagoya Institute of Technology, Nagoya 466-8555, Japan (e-mail: y.sumiya.228@stn.nitech.ac.jp; fukushima@nitech.ac.jp).

Tomoki Otsuka is with the Nagoya Institute of Technology, Nagoya 466-8555, Japan (e-mail: ot.19970326@gmail.com).

Yoshihiro Maeda is with the Tokyo University of Science, Tokyo 125-8585, Japan (e-mail: ymaeda@rs.tus.ac.jp).

This article has supplementary downloadable material available at http://doi.org/10.1109/LSP.2021.3121198, provided by the authors.

Digital Object Identifier 10.1109/LSP.2021.3121198

II. Preliminary

A. Gaussian Pyramid and Laplacian Pyramid

We introduce Gaussian and Laplacian pyramids, which are the bases of LLF. The Laplacian pyramid is used for multi-scale processing and analysis of images for compression [20], texture synthesis [21], and harmonization [22]. The traditional Laplacian pyramid processing directly enhances the coefficients of the pyramid [11], [23], [24].

First, the Gaussian pyramid is introduced. Let $\boldsymbol{I} : \mathcal{S} \mapsto \mathcal{R}$ be a D-dimensional R-tone input grayscale image, where $\mathcal{S} \subset \mathbb{Z}^D$ and $\mathcal{R} \subset \mathbb{R}$ denote the spatial and range domain (generally, $D = 2$ and $R = 256$), respectively. The Gaussian pyramid is defined as the set of $G_\ell[\boldsymbol{I}] \in \mathbb{R}$, where $\ell \in \mathcal{L} = \{0, 1, 2, \dots, \ell_{\max}\} \subset \mathbb{Z}$. The lowest level of the pyramid is $G_0[\boldsymbol{I}] = \boldsymbol{I}$, and its other level $G_{\ell+1}[\boldsymbol{I}]$ is the downsampled blurred image of $G_\ell[\boldsymbol{I}]$. This relationship can be described as follows:

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

where $\mathcal{G}_\sigma *$ is a Gaussian convolution with the standard deviation σ. $\downarrow$ is a downsampling operator, which halves the image size. The size of $G_{\ell+1}[\boldsymbol{I}]$ is the half-width and height of $G_\ell[\boldsymbol{I}]$. Usually, the Gaussian convolution is based on the binomial distribution for integer operations with five taps. The regarded standard deviation is approximately $\sigma = 1$.

The Laplacian pyramid is defined by the difference between the Gaussian pyramids of successive levels:

$$L_\ell[\boldsymbol{I}] = G_\ell[\boldsymbol{I}] - (G_{\ell+1}[\boldsymbol{I}])_\uparrow, \tag{2}$$

where $\uparrow$ is an upsampling operator that doubles the image width and height using a smoothing kernel. Usually, the kernel is identical to $\mathcal{G}_\sigma$. The highest level of the Laplacian pyramid is $L_{\ell_{\max}}[\boldsymbol{I}] = G_{\ell_{\max}}[\boldsymbol{I}]$.

B. *Manipulation of the Laplacian Pyramid*

The multi-scale detail enhancement with the Laplacian pyramid amplifies the pyramid coefficients except for the coarsest layer $\sup \mathcal{L}$. For enhance functions, an S-tone-like function can suppress an overshoot, while the straightforward function is $r(i, 0) = mi$, where $m \in \mathbb{R}$ is a constant value. If $m = 1$, the resulting image is the input image. The argument 0 in $r(i, 0)$ is unnecessary in this case; it is used to match the latter remap function for LLF. Finally, the remapped signals are collapsed to obtain the output:

$$\boldsymbol{O} = L_{\ell_{\max}}[\boldsymbol{I}] + \sum_{\ell \in \mathcal{L} \setminus \{\ell_{\max}\}} r(L_\ell[\boldsymbol{I}], 0), \tag{3}$$

where $\setminus$ indicates the excluding operator from the set.

C. *Local Laplacian Filtering*

The output pixel value of LLF $\boldsymbol{O}_p$ is defined by the Laplacian pyramid $L[\boldsymbol{O}]_p$ at p called the local Laplacian pyramid, where $p = (x, y) \in \mathcal{S}$ is the pixel position. Collapsing the pyramid generates the output image:

$$\boldsymbol{O}_p = \sum_{\ell \in \mathcal{L}} L_\ell[\boldsymbol{O}]_p. \tag{4}$$

Let us consider the procedure for calculating $L_\ell[\boldsymbol{O}]_p$.

1) Repeat 2)– 4) for all the pixels p and levels ℓ.
2) Build the Gaussian pyramid $G_\ell[\boldsymbol{I}]$ at level ℓ.
3) Apply a remap function discussed later and create a contrast-transformed image $\tilde{\boldsymbol{I}} = r(I, G_\ell[\boldsymbol{I}]_p)$.
4) Construct a Laplacian pyramid for $\tilde{\boldsymbol{I}}$, $L_\ell[\tilde{\boldsymbol{I}}]$, and copy the pyramid coefficient at p, $L_\ell[\tilde{\boldsymbol{I}}]_p$, as an output Laplacian pyramid of $L_\ell[\boldsymbol{O}]_p$. The remapping value changes according to the level ℓ at p.

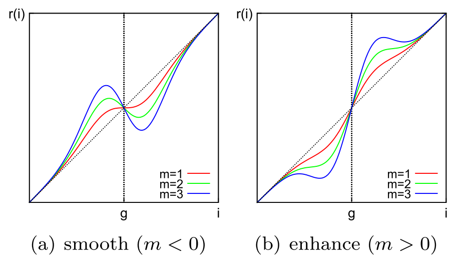


Fig. 1. Remap functions of detail smoothing and enhancement. σ_r is used to distinguish the details and edges.

5) Collapse the pyramid using Eq. (4) to obtain $\boldsymbol{O}$.

The flow requires per-pixel and per-level construction of the pyramid; thus, its order is $O(N|\mathcal{L}| \cdot N \log |\mathcal{L}|)$, where N and $|\mathcal{L}|$ are the number of pixels and levels, respectively. $O(N \log |\mathcal{L}|)$ is the order of the pyramid building.

D. *Fast Local Laplacian Filtering*

The naïve implementation accesses all layers of the pyramid for all pixels, while fast LLF [18] requires a limited number of pyramids for approximation. The algorithm of fast LLF is defined as follows:

1) Build the Gaussian pyramid $G_\ell[\boldsymbol{I}] \quad \forall \ell \in \mathcal{L}$.
2) Sample the intensities $k \in \mathcal{I} \subset \mathcal{R}$ at equal intervals τ.
3) Compute the remap images $\boldsymbol{R}_k = r(\boldsymbol{I}, \tau k)$ for each k and build their Laplacian pyramids $L_\ell[\boldsymbol{R}_k]$.
4) For each pyramid and pixels ($\ell \in \mathcal{L}$, $p \in \mathcal{S}$):
   a) Obtain the pixel value $G_\ell[\boldsymbol{I}]_p$ in the Gaussian pyramid.
   b) Find $a (0 \leq a \leq 1)$ such that $G_\ell[\boldsymbol{I}]_p = \tau((1-a)k + a(k+1)), j = \lfloor G_\ell[\boldsymbol{I}]_p \rfloor / \tau$.
   c) Compute the output pyramid by linearly interpolating the precomputed pyramids: $L_\ell[\boldsymbol{O}]_p = (1-a) L_\ell[\boldsymbol{R}_k]_p + a L_\ell[\boldsymbol{R}_{k+1}]_p$.
5) Collapse the pyramid using (4) to obtain $\boldsymbol{O}_p$.

The flow requires $|\mathcal{I}| + 1$ pyramids, where $|\mathcal{I}|$ is the number of sampled intensities. The order is $O((|\mathcal{I}| + 1) \cdot N \log |\mathcal{L}|)$.

E. *Remap Function*

The remap function for an intensity $i \in \mathcal{R}$ is continuous and changes the input i around a reference value $g \in \mathcal{R}$:

$$r(i, g) = i - (i - g) f(i - g). \tag{5}$$

LLF uses $g = G_\ell[I]_p$. $f \in \mathbb{R}$ is an even function usually and is specifically defined for each application. This study uses a Gaussian function with multiplication factor m denoted as $w_r(i)$ for f, which is defined as:

$$w_r(i) = m \exp\left(\frac{i^2}{-2\sigma_r^2}\right). \tag{6}$$

When $m > 0$, the function enhances images, while when $m < 0$, the function smooths images, as shown in Fig. 1. The dashed lines indicate the $y = x$ line, which is no remap function. The difference between the $y = x$ line and the desired remap function indicates the degree of emphasis.

III. Proposed Method

A. Formulation

This study approximates the remap function using Fourier series expansion. The derivative function of the Gaussian function w_r in the remap function is:

$$w'_r(i-g) = \sigma_r^{-2}(i-g)w_r(i-g)$$
$$\therefore (i-g)w_r(i-g) = \sigma_r^2 w'_r(i-g). \qquad (7)$$

Substituting (7) into the remap function (5), the form is:

$$\begin{aligned} r(i,g) &= i-(i-g)w_r(i-g) \\ &= i-\sigma_r^2 w'_r(i-g). \end{aligned} \qquad (8)$$

The Gaussian function in $w_r(i)$ is an even function, which can be approximated by the number of finite cosine terms K of the Fourier series expansion [25]:

$$w_r(i-g) \approx m\left(\alpha_0 + 2\sum_{k=1}^{K}\alpha_k\cos(\omega_k(i-g))\right), \qquad (9)$$

where, $\alpha_k = \frac{\sigma_r\sqrt{2\pi}}{T}\exp\left(-\frac{1}{2}(\omega_k\sigma_r)^2\right)$ and $\omega_k = \frac{2\pi}{T}k$. T is the period. The derivative function of (9) is

$$w'_r(i-g) \approx -2m\sum_{k=1}^{K}\alpha_k\sin(\omega_k(i-g))\omega_k. \qquad (10)$$

Using (10) and the addition theorem of trigonometric functions, we can approximate the remap function (8) as follows:

$$r(i,g) \approx i - m\sum_{k=1}^{K}\tilde{\alpha_k}\left(\sin(\omega_k g)\cos(\omega_k i) - \cos(\omega_k g)\sin(\omega_k i)\right), \qquad (11)$$

where $\tilde{\alpha}_k = 2\sigma_r^2\alpha_k\omega_k$.

The period T for an R-tone image is usually determined such that the following equation becomes minimum [25]; the equation approximates the formula for the error between the Gaussian function and its approximation:

$$\operatorname*{arg\,min}_{T} E_k(T) = \operatorname{erfc}\left(\frac{\pi\sigma}{T}(2K+1)\right) + \operatorname{erfc}\left(\frac{T-R}{\sigma}\right). \qquad (12)$$

The output local Laplacian pyramid is defined by

$$L_\ell[\boldsymbol{O}]_p = G_\ell[r(\boldsymbol{I}, G_\ell[\boldsymbol{I}]_p)] - G_{\ell+1}[r(\boldsymbol{I}, G_\ell[\boldsymbol{I}]_p)]_\uparrow. \qquad (13)$$

Recall that $G_0[\boldsymbol{I}] = \boldsymbol{I}$, $L_{\ell_{\max}}[\boldsymbol{O}] = G_{\ell_{\max}}[\boldsymbol{I}]$, and $r(\boldsymbol{I}, \boldsymbol{I}_p) = \boldsymbol{I}$; thus, the form is expressed as follows by substituting (11):

case : $\ell = 0$

$$\begin{aligned} L_0[\boldsymbol{O}] \approx \boldsymbol{I} - G_1[\boldsymbol{I}]_\uparrow + m\sum_{k=1}^{K}\tilde{\alpha_k}(&\sin(\omega_k G_0[\boldsymbol{I}])\tilde{C}_{1,k_\uparrow} \\ &- \cos(\omega_k G_0[\boldsymbol{I}])\tilde{S}_{1,k_\uparrow}), \end{aligned}$$

case : $1 \le \ell \le \ell_{\max}-1$

$$\begin{aligned} L_\ell[\boldsymbol{O}] \approx G_\ell[\boldsymbol{I}] - m\sum_{k=1}^{K}\tilde{\alpha_k}(&\sin(\omega_k G_\ell[\boldsymbol{I}])\tilde{C}_{\ell,k} \\ &- \cos(\omega_k G_\ell[\boldsymbol{I}])\tilde{S}_{\ell,k}) \\ - G_{\ell+1}[\boldsymbol{I}]_\uparrow + m\sum_{k=1}^{K}\tilde{\alpha_k}(&\sin(\omega_k G_\ell[\boldsymbol{I}])\tilde{C}_{\ell+1,k_\uparrow} \\ &- \cos(\omega_k G_\ell[\boldsymbol{I}])\tilde{S}_{\ell+1,k_\uparrow}), \end{aligned}$$

where $\tilde{C}_{\ell,k} = G_\ell[\cos(\omega_k\boldsymbol{I})]$ and $\tilde{S}_{\ell,k} = G_\ell[\sin(\omega_k\boldsymbol{I})]$. (14)

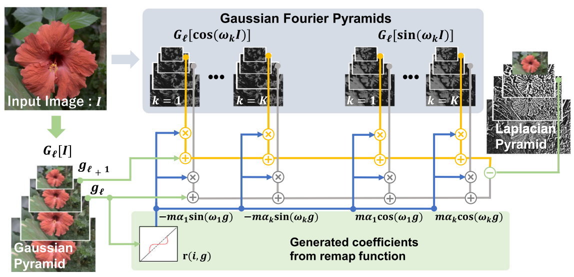

Gaussian Fourier Pyramids
$G_\ell[\cos(\omega_k I)]$
$G_\ell[\sin(\omega_k I)]$
Input Image : I
k = 1
k = K
k = 1
k = K
Laplacian Pyramid
$G_\ell[I]$
$g_{\ell+1}$
g_ℓ
Gaussian Pyramid
$-m\alpha_1\sin(\omega_1 g)$
$-m\alpha_k\sin(\omega_k g)$
$m\alpha_1\cos(\omega_1 g)$
$m\alpha_k\cos(\omega_k g)$
r(i,g)
Generated coefficients from remap function

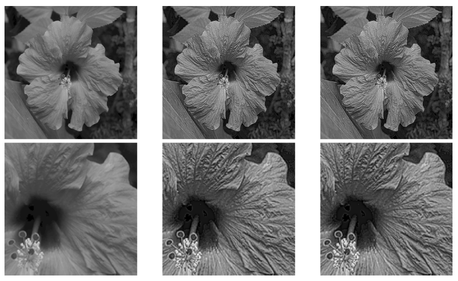
Fig. 2. Overview of the Fourier LLF, which shows the computational flow of (14). First, we create a Gaussian pyramid of the input image ($G_\ell[I]$) and Gaussian Fourier pyramids ($G_\ell[\sin(\omega_k I)], G_\ell[\cos(\omega_k I)]$). Next, we compute $m\alpha_k\cos(\omega_k g)$ and $m\alpha_k\sin(\omega_k g)$ according to the value of the Gaussian pyramid g, and the remap function. The output Laplacian pyramid is approximated by the product-sum of the Gaussian Fourier pyramids and coefficients, except the top.

Fig. 2 represents (14), visually. $\tilde{C}_{\ell,k}$ and $\tilde{S}_{\ell,k}$ represent the ℓ-th layer of the k-th Gaussian Fourier pyramids (blue shaded area), and $\cos(\omega_k G_\ell[I])$ and $\sin(\omega_k G_\ell[I])$ multiplied by $m\tilde{\alpha_k}$ are the coefficients (green shaded area). In (14), all image operators (such as $L_\ell[\cdot], G_\ell[\cdot], \tilde{C}_{\ell,k}, \tilde{S}_{\ell,k}$) can be followed by the pixel position p. We can remove the operator p from (13) because there is no loop dependency for p. Because the arguments of all the pyramids are given by the input image and constant variables, the pyramids are independently computed; thus, the output image $O_p = \sum_{\ell\in\mathcal{L}} L_\ell[O]_p$ is obtained by the computing cost of $2K+1$ pyramids: K cosine and K sine pyramids, and $G_\ell[I]$. Therefore, the computational order of our method is $O((2K+1)\cdot N\log|\mathcal{L}|)$.

B. Pixel-By-Pixel Enhancement

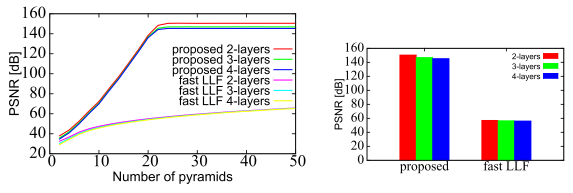
Changing the remap function for each pixel is essential for enhancement, such as the enhancement of flat regions to avoid noise signal boosting. The naïve implementation can locally change the function without any additional footprint; however, the computation itself is inefficient. The paper [19] also uses adaptive remap functions for fast LLF; however, the implementation is not an approximation of LLF. Fast LLF requires Laplacian pyramids for each remap function to interpolate the pyramid; thus, the multiple of the number of remap functions and pyramids is required for approximating LLF.

By contrast, the proposed method is independent of the remap functions for generating pyramids because the Gaussian Fourier pyramids of $\boldsymbol{I}$ do not depend on the remap function. Only the coefficients exhibit a dependency on the remap function; thus, we change the coefficient for the pixel-level parameter adaptation. Let $\tilde{\alpha}_{k,p}$ be the k-th coefficient for p. The remap function can be switched by replacing $\tilde{\alpha}_{k,p}$ pixel-by-pixel in (14) instead of $\tilde{\alpha}_k$. Similarly, we can also change the magnification factor of m to m_p for pixel-by-pixel adaptation. Therefore, we need $2K+1$ pyramids for this case, which are identical to the parameter-fixed case.



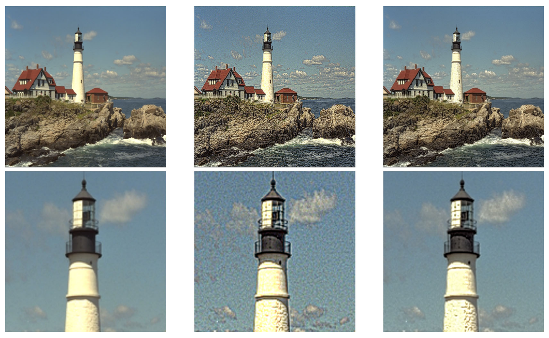
(a) input (b) fast LLF (c) Fourier LLF

Fig. 3. Input and output images (2-layer and 21 pyramids).



(a) PSNR (b) PSNR at 25 pyramids

Fig. 4. (a) PSNR of Fourier LLF and fast LLF to the number of pyramids (average of 10 images). (b) PSNR when the number of pyramids is 25. The ground truth is the naïve LLF output.



(a) input (b) fixed (c) pixel-by-pixel

Fig. 5. Outputs of pixel-by-pixel enhancement of Fourier LLF.

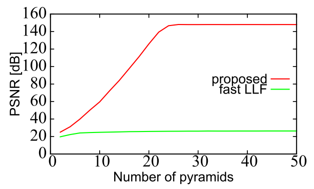


Fig. 6. Pixel-by-pixel enhancement case: PSNR of the proposed method and adaptive fast LLF [19]. The ground truth is the pixel-by-pixel enhanced naïve LLF.

IV. Experimental Results

We compared Fourier LLF with naïve LLF [17] and fast LLF [18]. The processing time was evaluated using two types of CPU: AMD Ryzen Threadripper 3970X (32 cores 3.7 GHz) and Intel Core i9-9980XE (18 cores 3.0 GHz). The code is written in C++, parallelized by OpenMP, and vectorized by AVX. We removed the subnormal numbers for an efficient implementation [26]. For the entire experiment, we set $\sigma_r = 30$ and $T = 405.9$, which is the optimal value when $\sigma_r = 30$. We used 10 traditional grayscale images of size 512×512 as test images.

Fig. 3 shows the input and output images for fast LLF and Fourier LLF with 21 pyramids. In both outputs, the details of the input image are emphasized. The processing time of our method is 16.5 ms, and that of fast LLF is 15.2 ms in Threadripper. In Core i9, it is 20.2 ms for our method and 21.3 ms for fast LLF. The processing time of naïve LLF is about 5 min in both CPUs.

These values are the average times of 100 trials. The processing time difference between the CPUs is not significant. When the number of pyramids is the same, the computation speed is almost the same.

Fig. 4 shows PSNR of fast LLF and our method for the 2-layer, 3-layer, and 4-layer cases. In Fig. 4(a), the x-axis shows the number of pyramids, which can be considered as the processing time because the cost of building the pyramids is a dominant factor in the filtering. For the same number of pyramids, our method always shows a higher PSNR than that of fast LLF. The processing time with the same number of pyramids is almost the same; however, the overall performance of the proposed method is superior to that of fast LLF. Fig. 4(b) shows the PSNR values when the number of pyramids is 25. It can be seen that for both methods, the PSNR decreases slightly as the number of layers increases, owing to accumulation errors in floating-point numbers. Note that over 59 dB images have the same output in the 8-bit level and are consistent with the output of the original paper [17]. Therefore, please see [17] for a visual comparison with other enhancement methods or our supplemental document.

Fig. 5 demonstrates the pixel-by-pixel enhancement. The degree of enhancement is determined according to the variance of the pixels in a local window; the low variance is a small enhancement. In the fixed-parameter case (Fig. 5(b)), the lighthouse is emphasized, while the flat sky is also emphasized, which may degrade the quality of the image. The pixel-by-pixel method (Fig. 5(c)) can enhance the image while maintaining visual of flat areas. For color processing, LLF is applied to the Y channel of the YUV color space, and the other components are kept.

Fig. 6 shows PSNR of parameter-adaptive fast LLF [19] and Fourier LLF with the pixel-by-pixel enhancement. Adaptive fast LLF does not improve PSNR even when the number of pyramids increases because fast LLF assumes that the same remap functions are used for all pixels. Fast LLF with the adaptive remap function [19] is another filter from the parameter adaptive naïve LLF. By contrast, the proposed method maintains high accuracy in the parameter-adaptive case.

V. Conclusion

This study proposes an approximation for LLF using Fourier series expansion, called Fourier LLF. The experimental results showed that our method achieved higher accuracy per pyramid building and better performance than those of the conventional method. We also showed that our method can approximate the adaptive filter. The limitation of the proposed method is that it cannot handle color image distances. Note that the conventional method is also for grayscale images. The aforementioned limitation can be solved by high-dimensional Gaussian kernel approximation methods [27], [28].

Acknowledgment

Code is available online at https://github.com/norishigefukushima/OpenCP/tree/master/MultiScaleFilter/.

REFERENCES

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

[2] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, no. 3, p. 67, 2008.

[3] S. Dippel, M. Stahl, W. Rafael, and T. Blaffert, "Multiscale contrast enhancement for radiographies: Laplacian pyramid versus fast wavelet transform," *IEEE Trans. Med. Imag.*, vol. 21, no. 4, pp. 343–353, Apr. 2002.

[4] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Eng.*, vol. 29, no. 6, pp. 33–41, 1984.

[5] S. Mallat, *A Wavelet Tour of Signal Processing*. New York, NY, USA: Elsevier, 1999.

[6] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Berlin, Germany: Springer, 2013, vol. 256.

[7] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE Int. Conf. Comput. Vision*, 1998, pp. 839–846.

[8] K. N. Chaudhury, "Acceleration of the shiftable O(1) algorithm for bilateral filtering and nonlocal means," *IEEE Trans. Image Process.*, vol. 22, pp. 1291–1300, 2013.

[9] K. Sugimoto, N. Fukushima, and S. Kamata, "200 FPS constant-time bilateral filter using SVD and tiling strategy," in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 190–194.

[10] Y. Sumiya, N. Fukushima, K. Sugimoto, and S. Kamata, "Extending compressive bilateral filtering for arbitrary range kernel," in *Proc. IEEE Int. Conf. Image Process.*, 2020, pp. 1018–1022.

[11] R. Fattal, M. Agrawala, and S. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 51, 2007.

[12] B. Gu, W. Li, M. Zhu, and M. Wang, "Local edge-preserving multiscale decomposition for high dynamic range image tone mapping," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 70–79, Jan. 2013.

[13] K. He, J. Shun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[14] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–10, 2009.

[15] 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*. Berlin, Germany: Springer, 1990, pp. 286–297.

[16] H. Dammertz, D. Sewtz, J. Hanika, and H. P. A. Lensch, "Edge-avoiding Á-trous wavelet transform for fast global illumination filtering," in *Proc. ACM SIGGRAPH/EUROGRAPHICS Conf. High Perform. Graph.*, 2010, pp. 67–75.

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

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

[19] Z. Qtang, L. He, Y. Chen, X. Chen, and D. Xu, "Adaptive fast local Laplacian filters and its edge-aware application," *Multimedia Tools Appl.*, vol. 78, pp. 619–639, 2019.

[20] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 532–540, Apr. 1983.

[21] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. Annu. Conf. Comput. Graph. Interactive Techn.*, 1995, pp. 229–238.

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

[23] P. Vuylsteke and E. P. Schoeters, "Multiscale image contrast amplification (MUSICA)," in *Proc. SPIE, Med. Imag.: Image Process.*, 1994, vol. 2167, pp. 551–560.

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

[25] K. Sugimoto and S. Kamata, "Compressive bilateral filtering," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3357–3369, Nov. 2015.

[26] Y. Maeda, N. Fukushima, and H. Matsuo, "Effective implementation of edge-preserving filtering on CPU microarchitectures," *Appl. Sci.*, vol. 8, no. 10, 2018.

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

[28] T. Miyamura, N. Fukushima, M. Waqas, K. Sugimoto, and S. Kamata, "Image tiling for clustering to improve stability of constant-time color bilateral filtering," in *Proc. IEEE Int. Conf. Image Process.*, 2020, pp. 1038–1042.