



Local look-up table upsampling for accelerating image processing

Tepei Tsubokawa^{1,4} · Hiroshi Tajima^{2,4} · Yoshihiro Maeda³ · Norishige Fukushima⁴

Received: 9 April 2022 / Revised: 29 May 2023 / Accepted: 21 July 2023 /
Published online: 28 August 2023
© The Author(s) 2023

Abstract

The resolution of cameras is increasing, and speedup of various image processing is required to accompany this increase. A simple way of acceleration is processing the image at low resolution and then upsampling the result. Moreover, when we can use an additional high-resolution image as guidance formation for upsampling, we can upsample the image processing results more accurately. We propose an approach to accelerate various image processing by downsampling and joint upsampling. This paper utilizes per-pixel look-up tables (LUTs), named local LUT, which are given a low-resolution input image and output pair. Subsequently, we upsample the local LUT. We can then generate a high-resolution image only by referring to its local LUT. In our experimental results, we evaluated the proposed method on several image processing filters and applications: iterative bilateral filtering, ℓ_0 smoothing, local Laplacian filtering, inpainting, and haze removing. The proposed method accelerates image processing with sufficient approximation accuracy, and the proposed outperforms the conventional approaches in the trade-off between accuracy and efficiency. Our code is available at <https://fukushimalab.github.io/LLF/>.

Keywords Joint upsample · Local LUT · Acceleration

1 Introduction

Imaging applications improve the quality of photographs using various techniques, such as noise removal [5], outline emphasis [3], high dynamic range imaging [12], haze removing [25],

✉ Norishige Fukushima
fukushima@nitech.ac.jp

¹ DENSO Cooperation, Showa-cho 1-1, Kariya 448-8661, Aichi, Japan

² Canon Inc., Simomaruko 3-30-2, Ohta-ku 146-8501, Tokyo, Japan

³ Department of Electrical Engineering, Tokyo University of Science, 6-3-1, Nijjuku, Katsuhika-ku 125-8585, Tokyo, Japan

⁴ Department of Engineering, Nagoya Institute of Technology, Showa-ku, Gokiso-cho, Nagoya 466-8555, Aichi, Japan

stereo matching [28, 40], free-viewpoint imaging [30], and depth map enhancement [41]. These applications use edge-aware filters, including bilateral filtering [54], non-local means filtering [5], discrete cosine transform (DCT) filtering [62], BM3D [11], guided image filtering [26], domain transform filtering [19], adaptive manifold filtering [20], local Laplacian filtering [45], weighted least square filtering [33], and ℓ_0 smoothing [59]. However, these filters have high-computational costs.

Each filter has its own acceleration methods, such as bilateral filtering [1, 6–8, 12, 18, 22, 38, 39, 46–48, 60], non-local means filtering [1, 14], local Laplacian filtering [2, 49], DCT filtering [13, 15], guided image filtering [16, 24, 44] and weighted least square filtering [43]. Each approach successively accelerates filtering; however, the computational time of these filters also depends on image resolution. Image sizes are rapidly increasing (for example, smartphones already have 12-Megapixel cameras). Such high-resolution images require more acceleration techniques, regardless of the image size or filtering algorithm.

Some acceleration techniques were proposed for general image processing. A simple approach is to process images at low resolution by downsampling an image, processing the image, and upsampling the result. This solution accelerates various types of image processing and dramatically reduces processing time; however, the approximated accuracy is also significantly decreased. The downsampling loses significant details (such as edges) in images; hence, the resulting images also lose the information. Therefore, upsampling must restore the information lost during the downsampling to improve the approximation accuracy.

Sophisticated upsampling methods use a high-resolution image as a guidance signal. Kopf et al. [31] proposed joint bilateral upsampling (JBU), which uses bilateral weights [54] for upsampling. Liu et al. proposed joint geodesic upsampling (JGU) [37], which uses the geodesic distance for bilateral weights instead of using the Euclidean distance. JBU and JGU work well for smoothed output images but are unsuitable for edge-enhanced output. Fukushima et al. [17] use non-local weight [5] for application-specific upsampling of depth map upsampling, i.e., highly smoothed cases.

He and Sun [24] proposed guided image upsampling (GIU), which computes the coefficients of a linear transformation—the guided image filter’s coefficient [26]—at the downsampled domain, and then, the coefficients are upsampled. Finally, the upsampled coefficients transform the high-resolution input image into the approximated output. GIU can handle the linear relationship between input and output images. GIU is suitable for enhanced images but not better than JBU in smoothed cases.

Chen et al. [9] proposed bilateral guided upsampling (BGU), which obtained the output intensity by an affine transformation of the input intensity using the bilateral space [4]. BGU is well-balanced in both smoothed and enhanced cases; however, its computational cost is higher than JBU and GIU. BGU can capture a piece-wise linear relationship between input and output images. Chen et al. [10] proposed an acceleration method of image processing with a fully convolutional network (CNN). This approach has high approximation accuracy with fast computational time and can be used for various image processing applications. However, it requires enormous datasets for training and tuning the parameters.

Approximation accuracy and computational cost are essential trade-offs, and supporting a wide range of applications is also crucial. Also, learning-based methods require enormous learning for a general acceleration of image processing; thus, non-learning-based methods are preferred. In this study, we propose a new upsampling method that can support a wide range of image processing applications and is not based on machine learning. Figure 1 shows an overview of our method. First, we apply image processing in the downsampled domain for acceleration. Next, we generate a per-pixel look-up table (LUT) using the correspondence between a low-resolution input image and a low-resolution output image. Finally, we convert

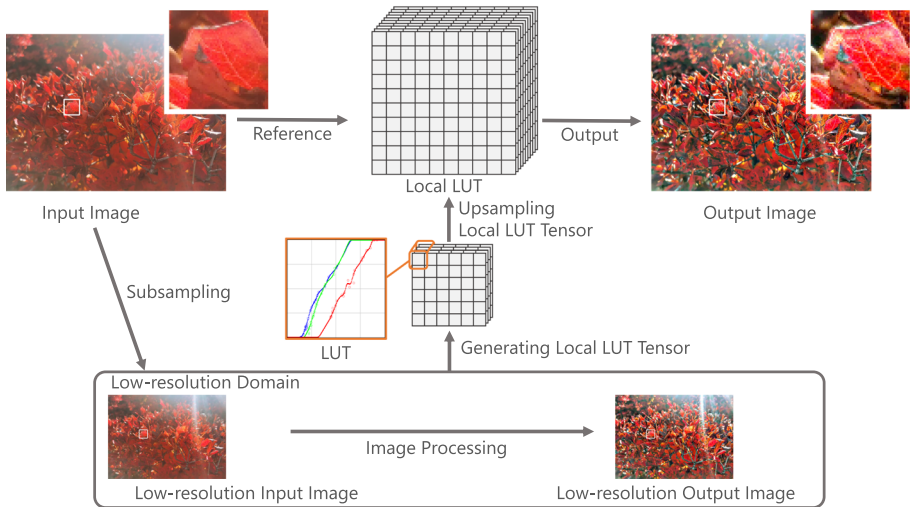


Fig. 1 Local LUT upsampling: An input image is downsampled. Subsequently, the downsampled image is smoothed by arbitrary image processing. Next, we create per-pixel LUTs (LUT tensor) using the correspondence between the subsampled input and the output images. Finally, we convert the input image into the approximated image by referring to and upsampling the local LUT tensor

the high-resolution input image into the high-resolution output image. We call it the *local LUT upsampling (LLU)*.

The main contributions of this study are as follows:

- The proposed method can capture a non-linear relationship between input and output images by LUT, covering a broad range of applications.
- We investigate various local LUT manipulation methods (building LUTs, boundary conditions of LUTs, quantization, and interpolation of LUT).
- We show the connection between the proposed and conventional methods: joint bilateral upsampling, guided image upsampling, and bilateral guided upsampling.

This paper is an extended version of our conference papers [50, 51]. In the building LUT process, dynamic programming (DP) and winner-take-all (WTA) approaches, which are reviewed in Sections 3.2.1 and 3.2.2, were introduced in [50]. Moreover, the 0–255 boundary condition, which is shown in Section 3.3, was introduced in [50]. An accelerated building LUT process for the WTA strategy, which is reviewed in Section 3.2.3, was introduced in [51].

The newly proposed parts are as follows: This study generalizes the accelerated the WTA approach to the ℓ_n norm minimization approach, which is shown in Section 3.2.3. Furthermore, we introduce an offset map for quantization in the building process. We add various boundary conditions: min-max, replication, and linear. For LUT upsampling, previous methods [50, 51] used linear and cubic upsampling. In addition, this study also uses box and Gauss upsampling. Moreover, this study shows the relationship between the proposed method and the conventional upsamples; JBU, GIU, and BGU. Finally, we optimize the code for our method by using AVX/AVX2, a vectorization instruction set in CPU, which is almost $\times 10$ faster than the conference version. These factors are newly verified in Section 5.

2 Acceleration by downsampling and upsampling

This section reviews accelerating methods of image processing by processing an image in the downsampled domain and then upsampling it. Let I and J be input and output images, respectively. Each image $I, J : S \mapsto \mathcal{R}$ is a D -dimensional R -tone gray/color image, where $S \subset \mathbb{Z}^D$ and $\mathcal{R} \subset \mathbb{Z}^{\{1,3\}}$ denote the spatial domain and the range domain (generally, $D=2$ and $R=256$), respectively. An image processing $J = F(I)$ can be approximately accelerated by downsampling and upsampling. An approximated output $\tilde{J} : S \mapsto \mathcal{R}$ is defined by:

$$\tilde{J} = F(I_{\downarrow})_{\uparrow}, \quad (1)$$

where \downarrow is a image downsampling operator, and \uparrow is an upsampling operator.

The image processing cost depends on the image size; thus, downsampling accelerates the speed. We can define an acceleration ratio of A for image processing:

$$A = \frac{C_F}{C_D + C_U + C_S}, \quad (2)$$

where C_F and C_S are processing costs of the full-sample and downsampled domains, respectively. C_D and C_U are downsampling time and upsampling time, respectively. The processing is accelerated when the total time of C_D , C_U , and C_S is shorter than C_F . C_S depends on the downsampled image size, and C_D and C_U depend on the downsampling and upsampling methods. Smaller-size processing can be accelerated more, but it requires more sophisticated upsampling for accurate approximation.

There are various upsampling [34, 64, 65] and single-image super-resolution methods [32, 53]. These approaches upsample a low-resolution input image to a high-resolution image. By contrast, joint upsample methods [4, 9, 10, 17, 24, 31, 37, 61] have better performance when we can use an additional high-resolution guidance image for upsampling.

Joint bilateral upsampling [31] and joint geodesic upsampling [37] work well for smooth images. Depth map upsampling [17, 61] is a particular case of smooth upsampling. Guided image upsampling [24] and bilateral guided upsampling [4, 9] can support various types of image processing. CNN-based upsampling [10] also supports various applications.

This study develops a joint upsampling method, which supports a wide range of image processing applications and is not based on machine learning.

3 Proposed method

3.1 Concept of local LUT

We typically use a LUT for all pixels in the LUT-based image intensity transformation (such as contrast enhancement, gamma correction, and tone mapping). We call this global LUT. The global LUT can represent any point-wise operations, and the LUT can be precomputable. However, this approach cannot represent area-based operations like image filtering.

By contrast, our approach generates per-pixel LUTs, called local LUT. The local LUT maps each pixel value of an input image to that of an image processing result pixel-by-pixel. The local LUT $T_p \subset \mathcal{R}$ at a pixel $p \in S$ has the following relationship between an input pixel value I_p and an output value J_p :

$$J_p = T_p[I_p], \quad (3)$$

where $T_p[\cdot]$ is the LUT reference operation. When we have image processing output, we can easily generate local LUTs and transform the input image into the image processing result by referring to the local LUTs. However, there is a contradiction: the output image is required for generating the output image itself. Therefore, we generate a local LUT for acceleration in the downsampled image domain. The proposed method generates per-pixel LUTs from the correspondence between a low-resolution input image and its processing result.

The local LUT for each pixel is created from the neighboring pairs' relationship around a target pixel between a low-resolution processed image and a downsampled input image. Intensities around the neighboring region have high correlations. Moreover, the correlation between the processing output and the input image is high. Therefore, we aggregate the mapping relationship around neighboring pixels.

Figure 1 shows the flowchart of our method. First, we downsample the input high-resolution image I to the input low-resolution image I_\downarrow :

$$I_\downarrow = S_s(I), \tag{4}$$

where $S_s(\cdot)$ represents the subsampling operator for the spatial domain. Subsequently, we apply image processing for I_\downarrow and then obtain the low-resolution output J_\downarrow . The process is represented as follows:

$$J_\downarrow = F(I_\downarrow), \tag{5}$$

where $F(\cdot)$ is a function of arbitrary image processing.

Next, we generate a local LUT on a pixel from the correspondence between I_\downarrow and J_\downarrow . Let p_\downarrow be the target pixel position in the downsampled images I_\downarrow and J_\downarrow , respectively. The local LUT at p_\downarrow is defined as follows:

$$T_{p_\downarrow} = \mathcal{L}(I_\downarrow, J_\downarrow, p_\downarrow, W_{p_\downarrow}), \tag{6}$$

where $\mathcal{L}(\cdot)$ is a generating function of the local LUT reviewed in Section 3.2, and W_{p_\downarrow} is a set of neighboring pixel around p_\downarrow .

Moreover, we consider quantizing the intensity to trim the LUT's size. A quantized intensity $\hat{I}_{p_\downarrow} \in \mathcal{R}$ is defined as follows:

$$\hat{I}_{p_\downarrow} = \lfloor I_{p_\downarrow}/a \rfloor, \tag{7}$$

where a is a quantization parameter. The output of the local LUT has $L = \lfloor 256/a \rfloor \in \mathbb{N}$ levels. We call the number of intensity candidates as *the number of bins*. The intensity quantization accelerates its speed but decreases the approximation accuracy according to the quantization parameter a .

Thus, we introduce an offset map to compensate for the degraded intensity. The offset map has quantization errors in the downsampled input image. The offset map $O_{p_\downarrow} \in \mathcal{R}$ is defined as follows:

$$O_{p_\downarrow} = I_{p_\downarrow} - (a\hat{I}_{p_\downarrow}). \tag{8}$$

The offset map is used to compensate for the output quantization error. For example, the original input intensity $I_{p_\downarrow} \in \mathcal{R}$ is restored as follows:

$$I_{p_\downarrow} = \lfloor a\hat{I}_{p_\downarrow} + O_{p_\downarrow} \rfloor. \tag{9}$$

Furthermore, the offset map reconstructs the local LUT reviewed in Section 3.5.

3.2 Generating local LUT

We generate the local LUT for each pixel by aggregating neighborhood correspondences between the downsampled input and the downsampled output intensities.

Figure 2 shows a scatterplot of intensity pairs between a local patch of an input image and a processed image. The correspondence plot roughly represents the local LUT relationship, but there are two problems: multiple output candidates and gaps. LUT requires one-on-one mapping for each input and output intensity; however, this scatterplot has one-on-zero or one-on- n matching. To solve these problems, we introduce three approaches: dynamic programming, WTA, and ℓ_n approaches.

3.2.1 Dynamic programming (DP)

In the DP approach, we generate a local LUT based on an appearance frequency of the correspondence between the input and output intensities in the neighboring pixel set, W_{p_\downarrow} , which has sampled pixels in a rectangle window centered at p_\downarrow and its size is $(2r + 1) \times (2r + 1)$, $r = 1, 2, 3, 4, 5$, typically. In the $r = 1$ case, the set $W_{p_\downarrow} = \{p_\downarrow + (-1, -1), p_\downarrow + (0, -1), p_\downarrow + (1, -1), p_\downarrow + (-1, 0), p_\downarrow + (0, 0), p_\downarrow + (1, 0), p_\downarrow + (-1, 1), p_\downarrow + (0, 1), p_\downarrow + (1, 1)\}$. We introduce a frequency map f_{p_\downarrow} , which counts the number of times an intensity pair appears between the input and output intensity in the neighboring pixel set. f_{p_\downarrow} is defined by

$$f_{p_\downarrow}(s, t) = \sum_{q_\downarrow \in W_{p_\downarrow}} \delta(\hat{I}_{q_\downarrow}, s) \delta(\hat{J}_{q_\downarrow}, t), \tag{10}$$

where δ denotes the Kronecker delta function. The variables $s \in \mathcal{R}$ and $t \in \mathcal{R}$ are input and output intensities, respectively. Subsequently, we apply DP using the frequency map as

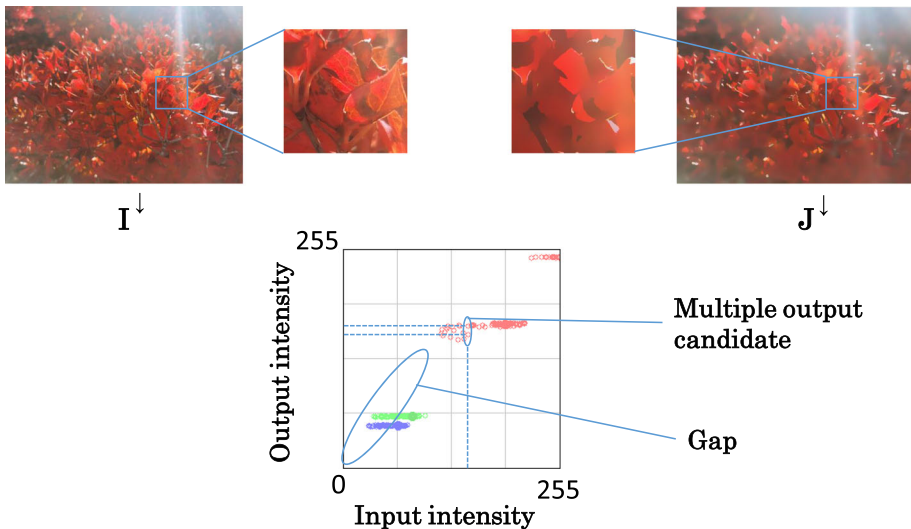


Fig. 2 Problems with generating a local LUT: a scatterplot for correspondence between input and output subsampled images in a local window. The scatterplot almost represents the LUT for mapping input to output. However, there are two problems: gaps and multiple-output candidates

a cost function. DP determines the output intensity for all input intensity ranges. Therefore, this approach can solve both problems (multiple candidates and gaps).

Algorithm 1 represents DP. In our DP approach, we will find a path from $f_{p_{\downarrow}}(0, 0)$ to $f_{p_{\downarrow}}(L - 1, L - 1)$. The path increases monotonically and prefers diagonal connection, which increases 1 both the x and y axis, e.g., from $f(0, 0)$ to $f(1, 1)$ and $f(100, 150)$ to $f(101, 151)$. First, the frequency map is initialized as 0. Next, we count the frequency map in the set $W_{p_{\downarrow}}$, where the size is $N = (2r + 1)^2$. Then, we aggregate the frequency map through the three types of connection: horizontal, vertical, and diagonal connection. For example, the horizontal connection is like $f(0, 0)$ to $f(1, 0)$ and the vertical connection is like $f(0, 0)$ to $f(0, 1)$. We compute boundary case such as $f(0, *)$ or $f(*, 0)$, where * indicates wild card. Then we start the main recursive loop. In the loop, we find the max value of f in the 3 connection types. We give priority to diagonal connections. Parameter \mathcal{P} in this algorithm represents an offsetting penalty to enforce a diagonal connection between LUT gaps. After filling the cost function, we trace back the function to determine a LUT from $f(L - 1, L - 1)$.

The value from the traceback determines the output value in a LUT.

$$T_{p_{\downarrow}}[s] = DP(f_{p_{\downarrow}}(s, t), s, t, \mathcal{P}). \tag{11}$$

Note that this approach ensures monotonicity in the local LUT.

3.2.2 Winner-take-all (WTA)

The second solution is the WTA approach in Algorithm 2. In this approach, we generate the local LUT based on the appearance of frequency as described in DP, and the output value of $T_{p_{\downarrow}}$ is determined as the most frequent intensity in the output image for intensity s in the input image:

$$T_{p_{\downarrow}}[s] = \arg \max_t f_{p_{\downarrow}}(s, t). \tag{12}$$

This approach can solve the multiple-candidate problem; however, the problem with LUT gaps remains. We linearly interpolate the LUT along the gap by $Interpolate(\cdot)$ in Algorithm 2, which is discussed in Section 3.3.

3.2.3 ℓ_n Norm minimization

DP and WTA require a frequency map, i.e., a 2D histogram. The number of elements in the map is L^2 , where L is the number of bins. In the no-quantization case, $L = 256$, we should initialize 65536 elements in a frequency map by zero for each pixel. By contrast, the local window size is $(2r + 1) \times (2r + 1)$, and typically $r = 2, 3, 4, 5$; thus, the initialization of the frequency map is the dominant processing. Moreover, the memory access cost for a massive array is higher than the arithmetic operation costs in current computers [23]. Thus, generating such a large frequency map could be more efficient.

The ℓ_n norm minimization approach solves this problem. When we have multiple candidates, we adopt the pixel intensity with the nearest position from the local window’s center (instead of the maximum counts used in WTA).

$$T_{p_{\downarrow}}[s] = \hat{I} \arg \min_{q_{\downarrow} \in W_{p_{\downarrow}}} \|p_{\downarrow} - q_{\downarrow}\|_n. \tag{13}$$

Algorithm 3 shows the process, where $sort(W_{p_{\downarrow}}, distance)$ indicates sorting the array of neighboring set $W_{p_{\downarrow}}$ in descending order by ℓ_n norm keying. In this study, we implemented

Algorithm 1 : DP approach.

Input: $I_{\downarrow}, J_{\downarrow}, p_{\downarrow}, W_{p_{\downarrow}}, \mathcal{P}$
Output: $T_{p_{\downarrow}}[s]$

Initialization: $N = (2r + 1)^2$
Initialization: $L =$ the number of bins
Initialization: $f_{p_{\downarrow}}(s, t) = 0 \ \forall s, t$
 Let q_{\downarrow} be a neighborhood pixel ($\{q_{1\downarrow}, \dots, q_{N\downarrow}\} \in W_{p_{\downarrow}}$)
for $n = 1$ **to** N **do** //(1) **count up frequency**
 $f_{p_{\downarrow}}(\hat{I}_{q_{n\downarrow}}, \hat{J}_{q_{n\downarrow}}) = f_{p_{\downarrow}}(\hat{I}_{q_{n\downarrow}}, \hat{J}_{q_{n\downarrow}}) + 1$
end for
for $i = 1$ **to** $L - 1$ **do** //(2) **left and bottom boundary loop**
 $f_{p_{\downarrow}}(i, 0) = f_{p_{\downarrow}}(i, 0) + f_{p_{\downarrow}}(i - 1, 0)$
 $f_{p_{\downarrow}}(0, i) = f_{p_{\downarrow}}(0, i) + f_{p_{\downarrow}}(0, i - 1)$
end for
for $s = 1$ **to** $L - 1$ **do** //(3) **main recursive loop**
 for $t = 1$ **to** $L - 1$ **do**
 $C_1 = f_{p_{\downarrow}}(s - 1, t - 1) + \mathcal{P}$
 $C_2 = f_{p_{\downarrow}}(s - 1, t)$
 $C_3 = f_{p_{\downarrow}}(s, t - 1)$
 $f_{p_{\downarrow}}(s, t) = \max(C_1, C_2, C_3) + f_{p_{\downarrow}}(s, t)$
 index(s, t) = $\arg \max_{m \in \{1, 2, 3\}} C_m$
 end for
end for
 $s = L - 1, t = L - 1$
while $s \geq 0$ and $t \geq 0$ **do** //(4) **trace back**
 $T_{p_{\downarrow}}[s] = t$
 if **index**(s, t) == 1 **then**
 $s = s - 1, t = t - 1$
 else if **index**(s, t) == 2 **then**
 $s = s - 1$
 else if **index**(s, t) == 3 **then**
 $t = t - 1$
 end if
end while

$\ell_1, \ell_2,$ and ℓ_{∞} norm approaches, i.e., Manhattan, Euclidean, and Chebyshev distances. The ℓ_n norm approach can directly compute a local LUT for a pixel without counting the frequency map. We can determine the LUT by scanning the local window at once. Moreover, we scan pixels in the presorted order from the pixel farther from the center pixel in the local window and overwrite the output intensity, which has already been defined in the local LUT. Therefore, we can determine a local LUT without comparing distances.

However, the problem with LUT gaps remains (as in the WTA approach); thus, the ℓ_n norm approach also needs interpolation.

3.3 Interpolation and boundary conditions of local LUT

WTA and ℓ_n norm approaches can solve the problem with multiple candidates; however, gaps remain. We interpolate the gaps in a local LUT by linear interpolation. Linear interpolation needs two samples. However, there are usually no samples outside of LUT. Thus, boundary regions require extrapolation.

In this section, we implement four extrapolation approaches to determine the boundary conditions. For simplicity, we assume that there is no quantization, i.e., $a = 1$.

Algorithm 2 : WTA approach.

Input: $I_{\downarrow}, J_{\downarrow}, p_{\downarrow}, W_{p_{\downarrow}}$
Output: $T_{p_{\downarrow}}[s]$
Initialization: $N = (2r + 1)^2$
Initialization: $L =$ the number of bins
Initialization: $f_{p_{\downarrow}}(s, t) = 0 \forall s, t$
 Let q_{\downarrow} be a neighborhood pixel ($\{q_{1\downarrow}, \dots, q_{N\downarrow}\} \in W_{p_{\downarrow}}$)
for $n = 1$ **to** N **do**
 $f_{p_{\downarrow}}(\hat{I}_{q_{n\downarrow}}, \hat{J}_{q_{n\downarrow}}) = f_{p_{\downarrow}}(\hat{I}_{q_{n\downarrow}}, \hat{J}_{q_{n\downarrow}}) + 1$
end for
for $s = 0$ **to** $L - 1$ **do**
 $T_{p_{\downarrow}}[s] = \arg \max_t f_{p_{\downarrow}}(s, t)$
end for
for $s = 0$ **to** $L - 1$ **do**
 if $T_{p_{\downarrow}}[s] == 0$ **then**
 Interpolate($T_{p_{\downarrow}}[s]$)
 end if
end for

Algorithm 3 : ℓ_n norm approach.

Input: $I_{\downarrow}, J_{\downarrow}, p_{\downarrow}, W_{p_{\downarrow}}, n$
Output: $T_{p_{\downarrow}}[s]$
Initialization: $N = (2r + 1)^2$
Initialization: $L =$ the number of bins
 Let q_{\downarrow} be a neighborhood pixel ($\{q_{1\downarrow}, \dots, q_{N\downarrow}\} \in W_{p_{\downarrow}}$)
for $i = 1$ **to** N **do**
 distance[i] = $\|p_{i_{\downarrow}} - q_{i_{\downarrow}}\|_n$ //computable in advance
end for
 sort($W_{p_{\downarrow}},$ distance) //computable in advance

for $i = 1$ **to** N **do** //sorted order loop
 $T_{p_{\downarrow}}[\hat{I}_{q_{i_{\downarrow}}}] = \hat{J}_{q_{i_{\downarrow}}}$ //overwrite without comparison
end for
for $s = 0$ **to** $L - 1$ **do**
 if $T_{p_{\downarrow}}[s] == 0$ **then**
 Interpolate($T_{p_{\downarrow}}[s]$)
 end if
end for

The first method is the 0–255 boundary. We set both ends of the local LUT to 0 and 255. Subsequently, we apply linear interpolation for the location where the output intensities are not defined:

$$\begin{cases} T_{p_{\downarrow}}[0] = 0 \\ T_{p_{\downarrow}}[255] = 255. \end{cases} \tag{14}$$

The second method is the *min-max* boundary. We set both ends of the local LUT to the minimum and maximum values in the local window. Next, we apply linear interpolation:

$$\begin{cases} T_{p_{\downarrow}}[0] = \min_{q_{\downarrow} \in W_{p_{\downarrow}}} J q_{\downarrow} \\ T_{p_{\downarrow}}[255] = \max_{q_{\downarrow} \in W_{p_{\downarrow}}} J q_{\downarrow}. \end{cases} \tag{15}$$

The third method is the *replication* boundary. This method replicates the minimum and maximum intensity, which are both edges of sampled points in a local LUT, in the local window from both edges of the LUT, i.e., 0 and 255, to the edge of the sampled points. Moreover, the minimum or maximum intensities are controlled by an offset value \mathcal{O} :

$$\begin{cases} T_{p\downarrow}[l] = T_{p\downarrow}[I_{p\downarrow}^{\min}] - \mathcal{O} & (l \leq I_{p\downarrow}^{\min}) \\ T_{p\downarrow}[l] = T_{p\downarrow}[I_{p\downarrow}^{\max}] + \mathcal{O} & (l \geq I_{p\downarrow}^{\max}), \end{cases} \tag{16}$$

where $I_{p\downarrow}^{\min} = \min_{q\downarrow \in W_{p\downarrow}} I_{q\downarrow}$, $I_{p\downarrow}^{\max} = \max_{q\downarrow \in W_{p\downarrow}} I_{q\downarrow}$.

The fourth method is the *linear* boundary. We compute a gradient from the minimum and maximum values of the input intensity in the local window. Next, we define both ends of the local LUT based on this gradient. Finally, we apply linear interpolation:

$$\begin{cases} T_{p\downarrow}[0] = T_{p\downarrow}[I_{p\downarrow}^{\min}] - d \cdot I_{p\downarrow}^{\min} \\ T_{p\downarrow}[255] = T_{p\downarrow}[I_{p\downarrow}^{\max}] + d \cdot (255 - I_{p\downarrow}^{\max}). \end{cases} \tag{17}$$

The gradient of d is computed as follows:

$$d = \frac{T_{p\downarrow}[I_{p\downarrow}^{\max}] - T_{p\downarrow}[I_{p\downarrow}^{\min}]}{I_{p\downarrow}^{\max} - I_{p\downarrow}^{\min}}, \tag{18}$$

These four methods solve the problem of LUT gaps. Figure 3 shows the gap-filled results of a local LUT for various combinations of LUT methods and boundary conditions.

3.4 Smoothing local LUT

After solving the problems with multiple candidates and gaps in the local LUT, we smooth the LUT by moving average filtering. The smoothed LUT $T'_{p\downarrow}[l]$ at level l is defined as follows:

$$T'_{p\downarrow}[l] = \frac{1}{|M|} \sum_{m \in M} T_{p\downarrow}[l + m], \tag{19}$$

where $M \subset \mathbb{Z}$ is smoothing region, and $|M|$ is its size.

The smoothing is an optional method. We use T instead of T' as the smoothed LUT from the next section for simplicity.

3.5 Upsampling of local LUT

The generated local LUT has three-dimensional information: intensity level n and pixel position $p = (x, y)$. All LUT elements are subsampled. Therefore, we upsample the local LUT in the spatial and intensity domain. The upsampling for the local LUT is defined as follows:

$$\tilde{T} = \mathcal{S}_c^{-1}(\mathcal{S}_s^{-1}(T_{\downarrow})), \tag{20}$$

where $\tilde{T} : \mathcal{S} \times \mathcal{R} \mapsto \mathcal{R}$ and $T_{\downarrow} : \mathcal{S} \times \mathcal{R} \mapsto \mathcal{R}$ are tensors, which contain local LUTs for each pixel. $\mathcal{S}_c^{-1}(\cdot)$ and $\mathcal{S}_s^{-1}(\cdot)$ are tensor upsampling operators for the intensity and spatial domains of the tensors, respectively. We use linear interpolation by adding the offset map

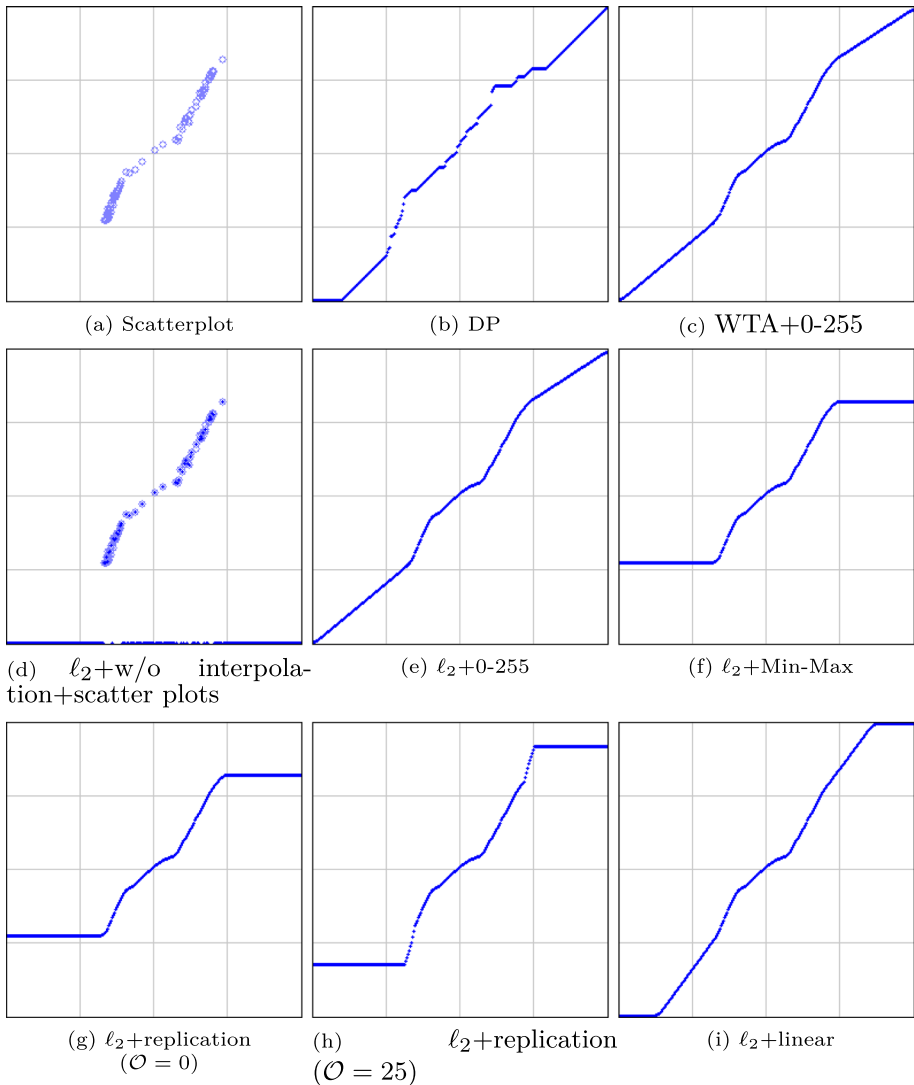


Fig. 3 Scatterplot of input and output values (local Laplacian filtering), and the results of each approach for generating local LUT on p . The radius of the local window is $r = 2$

for intensity domain upsampling. Moreover, we use four upsampling methods for the spatial domain: bi-linear, bi-cubic, box, and Gaussian upsampling.

We can reconstruct the output image by upsampling the tensor by (3) and referring to the tensor by (20). However, reconstructing the full LUT tensor implies redundant computations. We can narrow down the problem by computing the LUT values referred to the input image.

Let $\tilde{J}_p \in \mathcal{R}$ be an approximated output value at p . Given an input image I_p , the image is quantized by a ; \hat{I}_p . We compute two spatial tensor upsampling results S_s^{-1} , i.e., $I_p^C \in \mathbb{R}$ and

$I_p^H \in \mathbb{R}$, for intensity interpolation S_c^{-1} . The partial upsampling for a tensor $T : \mathcal{S} \times \mathcal{R} \mapsto \mathcal{R}$ is defined as follows:

$$I_p^L = \eta \sum_{q \downarrow \in \Omega_{p \downarrow}} w_{p \downarrow, q \downarrow} (a T_{q \downarrow} [\hat{I}_p] + O_{q \downarrow}), \quad (21)$$

$$I_p^H = \eta \sum_{q \downarrow \in \Omega_{p \downarrow}} w_{p \downarrow, q \downarrow} (a T_{q \downarrow} [\hat{I}_p + 1] + O_{q \downarrow}), \quad (22)$$

$$\eta = \frac{1}{\sum_{q \downarrow \in \Omega_{p \downarrow}} w_{p \downarrow, q \downarrow}}, \quad (23)$$

$$\tilde{J}_p = \text{round} \left(\frac{I_p - a \hat{I}_p}{a} I_p^L + \frac{a(\hat{I}_p + 1) - I_p}{a} I_p^H \right), \quad (24)$$

where $\Omega_{p \downarrow}$ is a position set of referred local LUTs for upsampling, $O_{q \downarrow}$ is an offset value, w is a spatial weight function, and η is its normalization term. The spatial weights are defined by each method: bi-linear, bi-cubic, box, and Gauss. Bi-linear and bi-cubic methods are well known; thus, we introduce only the box and Gauss methods in this paper. Box upsampling weight is $w_{p, q} = 1 / |\Omega_{p \downarrow}|$. Gaussian upsampling weight is $w_{p, q} = \exp(-\frac{\|p - q\|^2}{2\sigma^2})$.

4 Relationship to guided image upsample, bilateral guided upsample, and joint bilateral upsample

We review the relationship between the proposed method and the following conventional joint upsampling methods: guided image upsampling [24, 26], bilateral guided upsampling [9], and joint bilateral upsampling [31].

First, we review guided image upsampling (GIU) [24, 26]. GIU is a local linear transformation of an input image, which is defined as

$$\tilde{J}_p = \alpha_{\downarrow \uparrow p} I_p + \beta_{\downarrow \uparrow p}, \quad (25)$$

where $\alpha_{\downarrow \uparrow p}$ and $\beta_{\downarrow \uparrow p}$ are upsampled coefficients for the linear transformation. The computing method of these coefficients is written in [24, 26]. Equation (25) means that the local coefficients linearly transform input intensities. The transformation is conducted by the proposed local LUT upsampling method if we compute linear fitting or least squares for scatterplots obtained when building LUT. After fitting, LUTs become linear transformation functions.

Bilateral guided upsampling [9] is an extension of GIU. For simplicity, we assume that the range kernel of the bilateral part in BGU is a box kernel, while the original study used a Gaussian kernel. BGU divides the input image by the intensity level and then applies guided image upsampling for each level. BGU is defined by

$$\tilde{J}_p = \begin{cases} \alpha^0_{\downarrow \uparrow p} I_p + \beta^0_{\downarrow \uparrow p} & (l_0 \leq I_p < l_1) \\ \alpha^1_{\downarrow \uparrow p} I_p + \beta^1_{\downarrow \uparrow p} & (l_1 \leq I_p < l_2) \\ \dots & \\ \alpha^n_{\downarrow \uparrow p} I_p + \beta^n_{\downarrow \uparrow p} & (l_n \leq I_p < l_{n+1}) \end{cases}, \quad (26)$$

where $l_n = R/N_{BGU} \cdot n$, R is the number of intensity levels, N_{BGU} is the number of divisions of range dimension for BGU. $\alpha^n_{\downarrow \uparrow p}$ and $\beta^n_{\downarrow \uparrow p}$ are coefficients for each level n . BGU

is a piece-wise linear transformation for each divided level. Our method also realizes BGU if we compute linear fitting or least square for scatterplots divided by each level.

GIU and BGU can be represented by (3). By contrast, we cannot directly represent joint bilateral upsampling (JBU) [31] by our local LUT due to JBU's normalization factor. The normalization factor is changed by input pixel position \mathbf{p} ; thus, we should prepare local LUTs per pixel position of the high-resolution image. The position adaptive LUT $T_{p,q\downarrow}[i]$ is defined as follows:

$$T_{p,q\downarrow}[i] = \frac{\exp(\frac{\|I_{q\downarrow}-i\|^2}{-2\sigma^2})J_{q\downarrow}}{\sum_{r \in \Omega_p} \exp(\frac{\|I_{r\downarrow}-i\|^2}{-2\sigma^2})}. \quad (27)$$

If we switch the local LUT per high-resolution pixel position \mathbf{p} , we can represent JBU by local LUT. However, the size of the local LUT is huge. The size will be $|\mathbf{I}|$ times the original, where $|\mathbf{I}|$ is the input image size.

5 Experimental results

We evaluated the proposed method on three image processing filters and two image processing applications: iterative bilateral filtering (IBF) [54], ℓ_0 smoothing (ℓ_0) [59], local Laplacian filtering (LLF) [45], haze removing (HR) [25], and additionally inpainting [52].

First, we analyzed the effect of changing the proposed method's parameters and settings. Next, we compared the proposed method of the local LUT upsampling (LLU) with the conventional upsampling methods: cubic upsampling (CU), joint bilateral upsampling (JBU) [31], guided image upsampling (GIU) [24], and bilateral guided upsampling (BGU) [9] by approximation accuracy and computational time. We used the peak-signal-to-noise ratio (PSNR) for accuracy evaluation and regarded the naïve filtering results as the ground-truth results. We used two datasets, the high-resolution high-precision images dataset (HRHP) and the Transform Recipes dataset (T-Recipes) [21]. HRHP has seven high-resolution test-images¹: artificial (3072×2048), bridge (2749×4049), building (7216×5412), cathedral (2000×3008), deer (4043×2641), fireworks (3136×2352), flower (2268×1512), and tree (6088×4550). T-Recipes includes 143 images with different resolutions, e.g., from 1408×896 to 5760×3840 . HRHP dataset is used to verify the parameter and various options in the proposed method, and T-Recipes dataset is used for comparing various upsampling methods.

The following parameters were used in image processing applications. For iterative bilateral filtering, iteration = 10, $\sigma_s = 10$, $\sigma_c = 20$, $r = 3\sigma_s$. For ℓ_0 smoothing, $\lambda = 0.005$ and $\kappa = 1.5$. For local Laplacian filtering, $\alpha = 0.5$, $\beta = 0.5$, $\sigma_r = 0.3$. For haze remove, $r = 15$, $r_{\min} = 4$, $\epsilon = 0.6$, and haze ratio = 0.1.

We implemented the proposed method and competitive methods² written in C++ with OpenMP parallelization and AVX vectorization. For downsampling, we used the nearest-neighbor downsampling after applying Gaussian filtering. For this downsampling, we used OpenCV with Intel IPP for operations with the cv::INTER_NN option. Experiments were performed on Intel Core i7 7700K (4.2 GHz 4 cores / 8 threads), and the code was compiled by Visual Studio 2022. We used $r = 2$ for the local patch to generate the local LUT. Moreover, we set $M = 7$ for LUT smoothing.

¹ http://www.imagecompression.info/test_images

² GIU is accelerated by [55] from the conference version [51].

5.1 Comparison of parameters and settings

We evaluated four factors in local LUT upsampling using the HRHP dataset; (1) building LUT methods, (2) boundary conditions, (3) LUT quantization and offset map, and (4) tensor upsampling methods:

5.1.1 Building LUT

We evaluated the PSNR accuracy and computational time for each method of generating the local LUT: DP, WTA, and ℓ_n norm. All results are average of seven images in the HRHP dataset. We used the linear method for boundary conditions and 4×4 box upsampling for upsampling the local LUT. We verified four sampling rate settings: $\{1/4, 1/16, 1/64, 1/256\}$. The $1/4$ subsampling rate indicates the width and height of images, which are downsampled by $1/2$ (for example, $1/16$ means $1/4 \times 1/4$).

Table 1 shows PSNR for each filter. PSNR accuracy is almost identical without DP, and WTA and ℓ_2 norm approaches have a slightly higher PSNR.

Table 2 shows the computational time for each method of generating the local LUT only. ℓ_n norm approach is obviously faster than the other methods. Therefore, the ℓ_n norm approach has the highest performance in terms of accuracy and speed among the compared methods. ℓ_∞ is slightly faster due to its cache efficiency that is near raster-scan order. However, the difference among ℓ_n norm approaches is ignorable.

5.1.2 LUT boundary condition

Table 3 shows PSNR of various image processing methods for each boundary condition. The sampling rate is $1/4$, and the ℓ_2 norm minimization approach is used. The replication method

Table 1 Averaged PSNR accuracy [dB] for each local LUT generating and image processing (iterative bilateral filtering, ℓ_0 smoothing, and local Laplacian filtering)

Sampling rate	DP	WTA	ℓ_1	ℓ_2	ℓ_∞
iterative bilateral filtering					
1/256	26.504	28.860	28.864	28.864	28.860
1/64	27.262	33.037	33.030	33.033	33.022
1/16	27.820	32.517	32.511	32.512	32.510
1/4	28.913	39.167	39.162	39.166	39.146
ℓ_0 smoothing					
1/256	30.302	32.407	32.411	32.412	32.406
1/64	31.357	33.692	33.690	33.691	33.685
1/16	32.824	35.772	35.769	35.771	35.761
1/4	35.024	38.987	38.986	38.990	38.974
local Laplacian filtering					
1/256	31.282	32.020	31.997	31.999	31.995
1/64	32.745	33.521	33.532	33.533	33.527
1/16	35.766	38.144	38.169	38.173	38.151
1/4	39.535	42.098	42.090	42.103	42.045

Table 2 Averaged computational time [ms/Mpix] for each method of generating the local LUT

Sampling rate	DP	WTA	ℓ_1	ℓ_2	ℓ_∞
1/256	328.7	143.7	25.2	25.3	24.9
1/64	1209.9	417.2	29.1	29.1	29.0
1/16	4811.3	3846.4	40.6	41.3	40.6
1/4	19201.5	4684.1	80.6	81.0	80.2

The computational time is normalized by 1-Megapixel. The time does not include image processing, such as iterative bilateral filtering, ℓ_0 smoothing, and local Laplacian filtering

is better for flatter images, such as IBF and L0. For sharper images, such as LLF and Haze, the 0–255 method is better. The linear method is a well-balanced method.

5.1.3 Quantization and offset map

We evaluated the effectiveness of quantization and the offset map. We changed the quantization level as follows: {16, 32, 64, 128, 256}.

Figure 4 shows PSNR w.r.t. time by changing the quantization level with/without the offset map. The sampling rate is 1/4 (upsampling level is 4). PSNR is the average of seven images in HRHP dataset. The computational time is normalized by 1-Megapixel size.

Figure 4 (a) includes iterative bilateral filtering results for PSNR, and its computational time includes only upsampling, i.e., excluding an acceleration target of image processing and downsampling processing. The quantization accelerates processing time, and we can obtain a better trade-off in upsampling with the offset map than without the offset map.

Figure 4(b) shows the total image processing time, which includes downsampling, upsampling, and image processing of iterative bilateral filtering. The image processing is cost-consuming; therefore, $C_U \ll C_S$ in (2). Note that the full-size processing time of iterative bilateral filtering is about 100 seconds. LUT quantization accelerates C_U shown in Fig. 4 (a), but the effect is ignorable in this case. In this case, no quantization is the best solution.

Figure 4(c) and (d) show more fast processing of C_S : domain transform filtering (DTF) [19] and weighted mode filtering (WMF) [42]. The plots include the time of image processing of DTF or WMF. Note that the full-size processing time of DTF is 310 ms, and its WMF is 340 ms. DTF is one of the fast edge-preserving filtering, and weighted mode filtering is one of the histogram-based filters. In Fig. 4(c), we can control the performance trade-off by quantization levels and downsampling rate. In Fig. 4(d), we can accelerate processing more than (c) by quantization because WMF is the histogram-based filtering. The computational order of the histogram-based filtering depends on the intensity levels; thus, the quantization

Table 3 PSNR accuracy for each boundary condition and image processing (PSNR [dB]/rank): Iterative bilateral filtering (IBF); local Laplacian filtering (LLF); ℓ_0 smoothing (ℓ_0); and Haze remove (Haze)

Boundary	IBF	LLF	L0	Haze
Min-Max	38.31/2	38.12/3	35.63/1	37.2/3
0-255	38.03/4	39.62/1	35.36/4	37.64/1
Replication	38.32/1	38.1/4	35.63/1	37.2/3
Linear	38.32/1	39.38/2	35.56/3	37.57/2

accelerates the processing time. The time of most image processing depends only on the image’s spatial domain, and the usual upsampling accelerates the processing time by the spatial factor. In contrast, the proposed method can accelerate processing by both spatial and range domain factors.

5.1.4 Tensor upsampling

Next, we evaluated PSNR and computational time for each method of upsampling the local LUT. We used the l_2 norm approach to build the local LUT and the linear method for the boundary condition. Table 4 shows PSNR for each method. Gauss64, which is 8×8 Gauss upsampling, has a higher PSNR than the other methods.

Table 5 shows the computational time for each upsampling method for the local LUT. Moreover, we include the time of the baseline methods. The time is normalized per 1-Megapixel image size (1024×1024). The computational time for each upsampling (i.e., linear, cubic, box, and Gauss) of the local LUT depends on the convolution size for upsampling. Thus, we denote 2×2 , 4×4 , and 8×8 convolution for upsampling as LLU4, LLU16, and LLU64, respectively. Linear and Box4 have the 2×2 kernel. Box16, Gauss16, and Cubic have the 4×4 kernel. Gauss64 has the 8×8 kernel. Larger size upsampling has higher upsampling quality than the smaller ones but has lower time performance.

Approximation accuracy deference between box 4×4 convolution and Gauss 16×16 is not considerable, so small size upsampling of the box 4×4 tends to be a better trade-off.

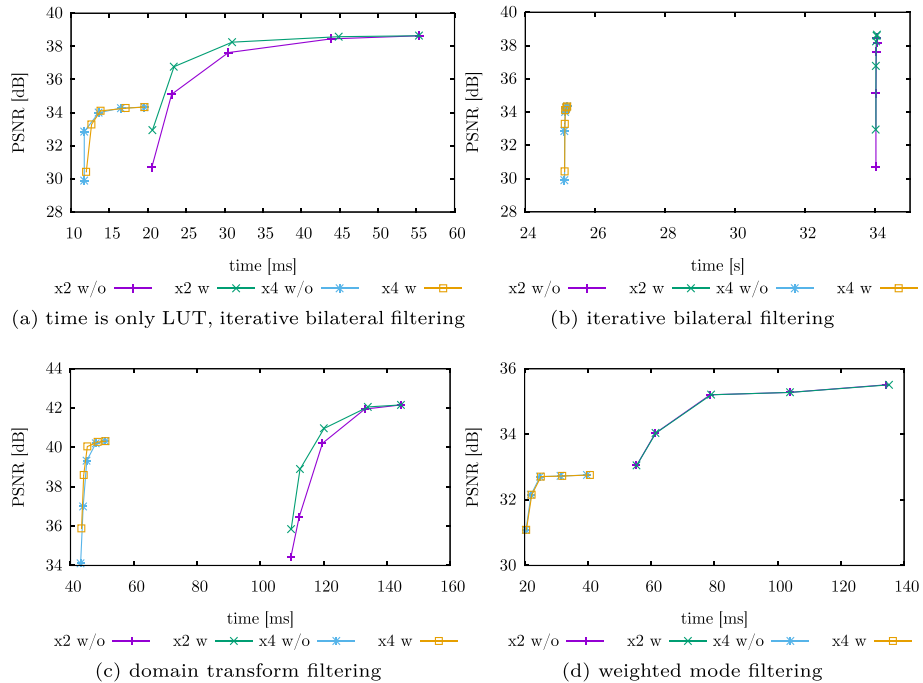


Fig. 4 Performance trade-off of local lut upsampling between time and accuracy of PSNR w.r.t. quantization level. x2 w/o means x2 upsampling without offset map, and x4 w means x4 upsampling with offset map. The computational time is normalized by 1-Megapixel. The unit of the x-axis for (b) is **second**, and the others are milliseconds

Table 4 PSNR accuracy [dB] for each method of upsampling the local LUT (iterative bilateral filtering, ℓ_0 smoothing, and local Laplacian filtering)

Sampling rate	Linear	Box4	Box16	Gauss16	Cubic	Gauss64
iterative bilateral filtering						
1/256	28.54	28.60	28.86	28.80	28.61	28.96
1/64	32.65	32.73	33.03	32.96	32.69	33.26
1/16	32.24	32.29	32.56	32.43	32.27	32.78
1/4	38.82	38.85	39.17	39.12	38.96	39.33
ℓ_0 smoothing						
1/256	32.12	32.18	32.40	32.36	32.18	32.53
1/64	33.45	33.50	33.69	33.64	33.49	33.81
1/16	35.54	35.58	35.77	35.72	35.58	35.91
1/4	38.77	38.79	38.99	38.97	38.86	39.15
local Laplacian filtering						
1/256	31.80	31.85	32.00	31.94	31.75	32.13
1/64	33.31	33.36	33.53	33.49	33.33	33.67
1/16	37.92	37.94	38.17	38.16	38.02	38.26
1/4	41.88	41.88	42.10	42.13	41.98	42.28

5.2 Comparison with other upsampling

Next, we compared the proposed upsampling with conventional methods. In conventional upsampling methods, we used $\sigma_s = 10$ and $\sigma_c = 20$ for JBU, $\epsilon = (5 \times 0.0001 \times 255)^2$ for GIU. The number of divisions in the spatial domain of the bilateral grid was 3, and the number of divisions in the range domain was 8 for BGU. We used 256 bins for the proposed method, which indicates no quantization for the intensity domain. These parameters were adopted because their performance was good on average. In this experiment, we use both HRHP dataset and T-Recipes dataset.

Figure 5 shows PSNR of each acceleration method for iterative bilateral filtering, ℓ_0 smoothing, and local Laplacian filtering on HRHP dataset. The proposed LLU had higher PSNR accuracy than other acceleration methods.

Figure 6 shows the computational time of each acceleration method on HRHP dataset. The horizontal line of the graph shows the computational time of naïve implementation. The

Table 5 The computational time [ms/Mpix] for each method

Sampling rate	LLU4	LLU16	LLU64	CU	JBU	GIU	BGU
1/256	2.88	6.24	25.29	1.12	7.62	16.99	16.77
1/64	5.46	8.97	29.07	1.13	7.72	17.05	20.25
1/16	15.07	19.55	41.31	1.23	8.31	17.94	32.85
1/4	51.38	55.37	80.96	1.73	11.76	22.66	81.75

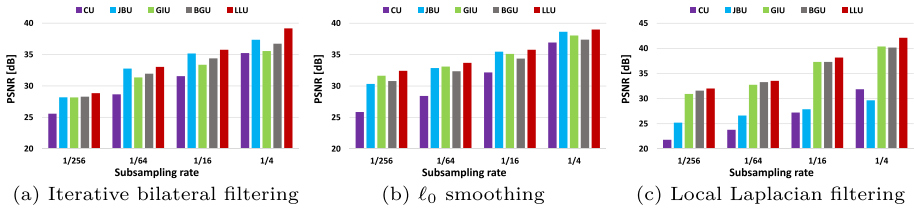


Fig. 5 Approximation accuracy of PSNR for each acceleration method with changing subsampling rate for spatial domain on **HRHP dataset** (generating method: ℓ_2 norm; interpolating method: linear)

computational time is an average of 10 trials. Note that the vertical axis of this graph is logarithmic. CU is the simplest and fastest method, but its accuracy performance is not high.

Figure 7 shows the trade-off between PSNR and the computational time by changing the subsampling rate for the spatial domain from 1/4 to 1/256 on HRHP dataset. The results show that LLU has the best trade-off performance among the competitive acceleration methods.

Tables 6, 7, 8 show the accuracy results on T-Recipes dataset. The evaluation metrics are PSNR, SSIM [56], and gradient magnitude similarity deviation GMSD [58]. All results are average of resulting 143 images in the dataset. The downsampled processing results are generated from the full sampling result of image processing by downsampling. The proposed LLU is always the best in the PSNR results (Tab. 6). On the other hand, the SSIM results (Tab. 7) show that the proposed method is still better in the overall results. However, the smoothing evaluation (iterative bilateral filtering and ℓ_0 smoothing) shows that JBU is often better in some cases. This is because SSIM prefers smoother images. Although SSIM is a universal metric for image distortion, current image quality metrics have shown that GMSD is closer to human subjectivity than SSIM; thus, we next evaluate GMSD. The GMSD results (Tab. 8) show that the proposed LLU is always the best result.

Figure 8 shows the visual comparison of each image processing and acceleration method. The proposed method has visually similar to the naïve output for each image processing application.

5.3 Subjective evaluation

We also performed a subjective evaluation for image quality assessment (IQA). We evaluated 4 image processing methods (IBF, ℓ_0 , LLF, HR) that were expanded with 5 up-sampling methods (CU, JBU, GIU, BGU, and proposed LLU). IBF and ℓ_0 are smoothing processings,

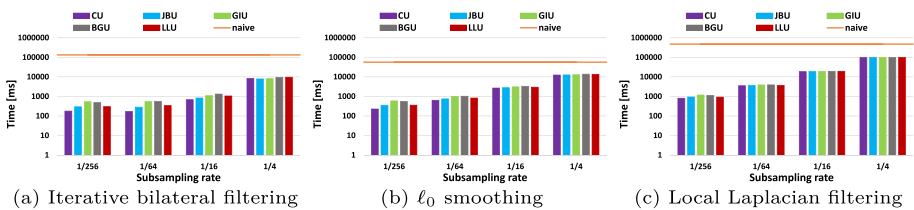


Fig. 6 Computational time for naïve implementation and each acceleration method with changing subsampling rate for spatial domain on **HRHP dataset** (generating method: ℓ_2 norm; interpolating method: linear.)

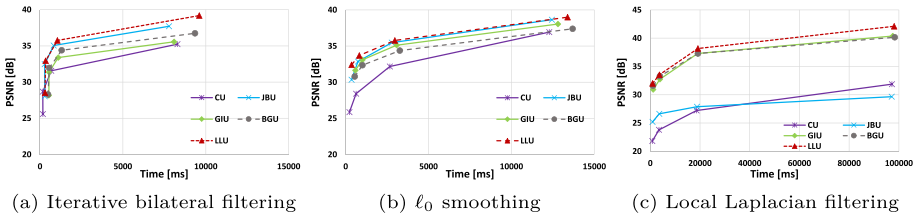


Fig. 7 Changing subsampling rate performance in PSNR w.r.t. the computational time of each acceleration method on **HRHP dataset** (generating method: ℓ_2 norm; boundary condition: linear)

Table 6 Accuracy comparison (PSNR [dB]) with the other upsampling methods for each sampling rate on **T-Recipes dataset**

ratio	CU	JBU	GIU	BGU	LLU
iterative bilateral filtering					
1/4	42.60	<u>45.71</u>	41.27	43.83	49.17
1/16	37.58	<u>43.47</u>	38.29	40.75	44.68
1/64	33.69	<u>40.19</u>	35.62	37.84	40.24
1/256	30.39	<u>36.18</u>	33.30	35.51	36.58
ℓ_0 smoothing					
1/4	43.81	<u>44.60</u>	41.46	43.05	45.83
1/16	38.03	<u>41.19</u>	38.28	40.19	42.16
1/64	33.56	<u>38.32</u>	35.70	37.81	39.21
1/256	29.94	<u>35.87</u>	33.51	35.81	36.78
local Laplacian filtering					
1/4	29.28	26.08	36.19	<u>37.31</u>	39.62
1/16	23.74	24.71	33.41	<u>34.76</u>	36.72
1/64	20.49	23.76	31.01	<u>32.40</u>	34.10
1/256	18.18	22.97	28.69	<u>30.01</u>	31.52
haze remove					
1/4	37.09	34.37	47.22	<u>49.52</u>	49.78
1/16	31.35	32.81	43.30	<u>45.18</u>	46.45
1/64	27.84	31.57	38.80	<u>41.02</u>	42.63
1/256	25.25	30.36	34.82	<u>37.71</u>	38.84
average					
1/4	38.19	37.69	41.53	<u>43.43</u>	46.10
1/16	32.66	35.54	38.32	<u>40.22</u>	42.50
1/64	28.89	33.46	35.28	<u>37.27</u>	39.05
1/256	25.94	31.34	32.58	<u>34.76</u>	35.93
All	31.42	34.51	36.93	<u>38.92</u>	40.89

Bold font is the best result, and underling is the second best. Higher values are better

Table 7 Accuracy comparison (SSIM) with the other upsampling methods for each sampling rate on **T-Recipes dataset**. Bold font is the best result, and underling is the second best. Higher values are better

ratio	CU	JBU	GIU	BGU	LLU
iterative bilateral filtering					
1/4	0.9865	<u>0.9934</u>	0.9874	0.9895	0.9961
1/16	0.9627	<u>0.9883</u>	0.9756	0.9780	0.9914
1/64	0.9379	0.9822	0.9584	0.9599	<u>0.9821</u>
1/256	0.9193	0.9722	0.9336	0.9380	<u>0.9641</u>
L0 smoothing					
1/4	0.9867	<u>0.9924</u>	0.9851	0.9841	0.9935
1/16	0.9597	<u>0.9834</u>	0.9710	0.9726	0.9853
1/64	0.9282	<u>0.9735</u>	0.9540	0.9601	0.9741
1/256	0.9052	0.9638	0.9364	0.9470	<u>0.9619</u>
local Laplacian filtering					
1/4	0.8643	0.7574	<u>0.9795</u>	0.9778	0.9869
1/16	0.6799	0.6819	0.9616	<u>0.9650</u>	0.9753
1/64	0.5324	0.6602	0.9399	<u>0.9492</u>	0.9610
1/256	0.4470	0.6631	0.9144	<u>0.9276</u>	0.9443
haze remove					
1/4	0.9388	0.9095	0.9895	<u>0.9966</u>	0.9973
1/16	0.8504	0.8846	0.9831	<u>0.9936</u>	0.9962
1/64	0.7801	0.8783	0.9740	<u>0.9884</u>	0.9943
1/256	0.7405	0.8790	0.9603	<u>0.9810</u>	0.9904
average					
1/4	0.9441	0.9132	0.9854	<u>0.9870</u>	0.9935
1/16	0.8632	0.8846	0.9728	<u>0.9773</u>	0.9870
1/64	0.7946	0.8736	0.9566	<u>0.9644</u>	0.9779
1/256	0.7530	0.8695	0.9362	<u>0.9484</u>	0.9652
All	0.8387	0.8892	0.9563	<u>0.9693</u>	0.9780

and LLF and HR are detail enhancement processing. We have 5 magnification levels: 1/1, 1/4, 1/16, 1/64, and 1/256. Since there are 5 different input images for each image processing, a total of 420 images were tested: $5 \times 4 \times 5 \times 4 = 400$ images for distorted image; processings \times upsamplings \times images \times levels, and $4 \times 5 = 20$ images for 1/1 magnification, i.e., just copies of the images; processings \times images. The evaluating 5 images were selected from the tested images of objective evaluations. However, since the subjective evaluation is also affected by the size of the images, the images were cropped to 1024×1024 to match the resolution. Note that 3 image data were changed only for the haze removal since the image does not contain haze (Fig. 9).

The experimental procedure was to display the non-degraded and degraded images on the left and right and perform a binary evaluation to see if the left and right images matched or not. The showing image pair is randomly selected for each subject. This procedure was

Table 8 Accuracy comparison (GMSD) with the other upsampling methods for each sampling rate on **T-Recipes dataset**. Bold font is the best result, and underling is the second best. Lower values are better

ratio	CU	JBU	GIU	BGU	LLU
iterative bilateral filtering					
1/4	0.00532	<u>0.00285</u>	0.00751	0.00291	0.00189
1/16	0.01426	0.00559	0.01380	<u>0.00558</u>	0.00434
1/64	0.02507	0.00902	0.02128	0.00996	0.00772
1/256	0.03759	0.01503	0.03030	0.01515	0.01298
ℓ_0 smoothing					
1/4	0.00466	<u>0.00252</u>	0.00707	0.00357	0.00202
1/16	0.01397	<u>0.00640</u>	0.01330	0.00659	0.00487
1/64	0.02606	0.01126	0.02107	<u>0.01042</u>	0.00887
1/256	0.04019	0.01712	0.02956	<u>0.01470</u>	0.01374
local Laplacian filtering					
1/4	0.01920	0.03003	0.00866	<u>0.00513</u>	0.00257
1/16	0.05191	0.04806	0.01629	<u>0.00976</u>	0.00590
1/64	0.07928	0.06086	0.02575	<u>0.01646</u>	0.01088
1/256	0.10188	0.07190	0.03730	<u>0.02581</u>	0.01779
haze remove					
1/4	0.00799	0.01094	0.00681	<u>0.00201</u>	0.00086
1/16	0.02587	0.02001	0.01179	<u>0.00438</u>	0.00233
1/64	0.04435	0.02768	0.01919	<u>0.00854</u>	0.00498
1/256	0.06167	0.03525	0.02877	<u>0.01409</u>	0.00954
average					
1/4	0.0093	0.0116	0.0075	<u>0.0034</u>	0.0018
1/16	0.0265	0.0200	0.0138	<u>0.0066</u>	0.0044
1/64	0.0437	0.0272	0.0218	<u>0.0113</u>	0.0081
1/256	0.0603	0.0348	0.0315	<u>0.0174</u>	0.0135
All	0.0350	0.0234	0.0187	<u>0.0096</u>	0.0070

repeated for 420 images. The display was EIZO CS 270 (27inch 2560 × 1440 / IPS), and the viewing distance was 0.5 m. The number of subjects was 11, not including the authors. The evaluation time per subject was approximately 2 hours. This time it included breaks to rest the eyes.

The average value of whether the images matched or not for each subject was used as the evaluation method, called the identification ratio. The binary evaluation is called just noticeable difference (JND). JND was used because it is difficult to evaluate the multi-gradation of very close images (e.g., Mean opinion score (MOS)). JND is also used in the recent IQA, such as learned perceptual image patch similarity (LPIPS) [66]. In this evaluation, a good agreement, i.e., 100%, indicates no degradation and high quality, while 0% indicates a clear degradation. The experimental flow has the same protocol of the research [27] for IQA for visually near-lossless image coding.

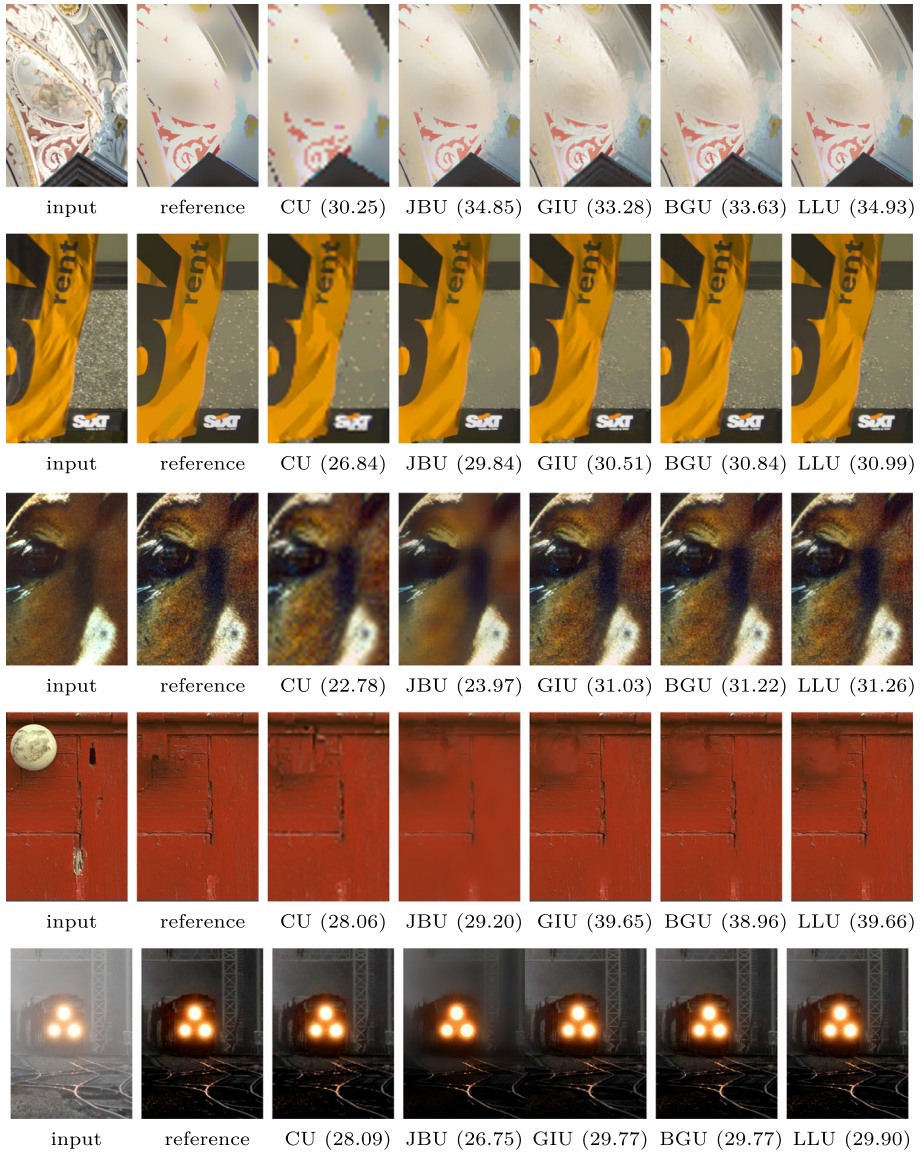


Fig. 8 Quality comparison of five imaging operators. Topline: iterative bilateral filtering. Second line: ℓ_0 smoothing. Third line: local Laplacian filtering. The fourth line: inpainting. The fifth line: haze removal. The values of PSNR are shown in (PSNR). We used the ℓ_2 norm method for building the local LUT, the linear method for boundary conditions, and 4×4 box upsampling for upsampling the local LUT tensor

Figure 10 shows the results of the subjective IQA. The all averaging results (ALL) show that the proposed method is the highest performance. Next, a breakdown of each process is given separately for smoothing (IBF and ℓ_0) and detail enhancement (LLF and HR). For smoothing cases, JBU and LLU have good performance. The IBF results indicate that JBU is the best since both image processing and upsampling use bilateral filtering weight. This is a special case. In the ℓ_0 case, the proposed method with a small downscaling factor (1/4) is



Fig. 9 Images for subjective image quality assessment. From top to bottom: source images, iterative bilateral filtering, ℓ_0 smoothing, local Laplacian filtering, and haze removing. For haze removing, 3 images are changed since there is no haze in the images

the best. GIF, BGU, and LLU for detail enhancement cases have good performance. In the LLF case, LLU has the highest performance, while JBU has a low performance since JBU has smooth images. The HR results are more remarkable.

Experimental results showed that the proposed method performed well not only in objective evaluation but also in subjective evaluation.

6 Conclusion

In this paper, we proposed an acceleration method for image processing by image upsampling. The local LUT upsampling has higher approximation accuracy than conventional methods. In addition, the local LUT upsampling accelerates $\times 100$ faster than the naïve implementation. Moreover, the proposed method can output more accurate results with the ℓ_n norm approach

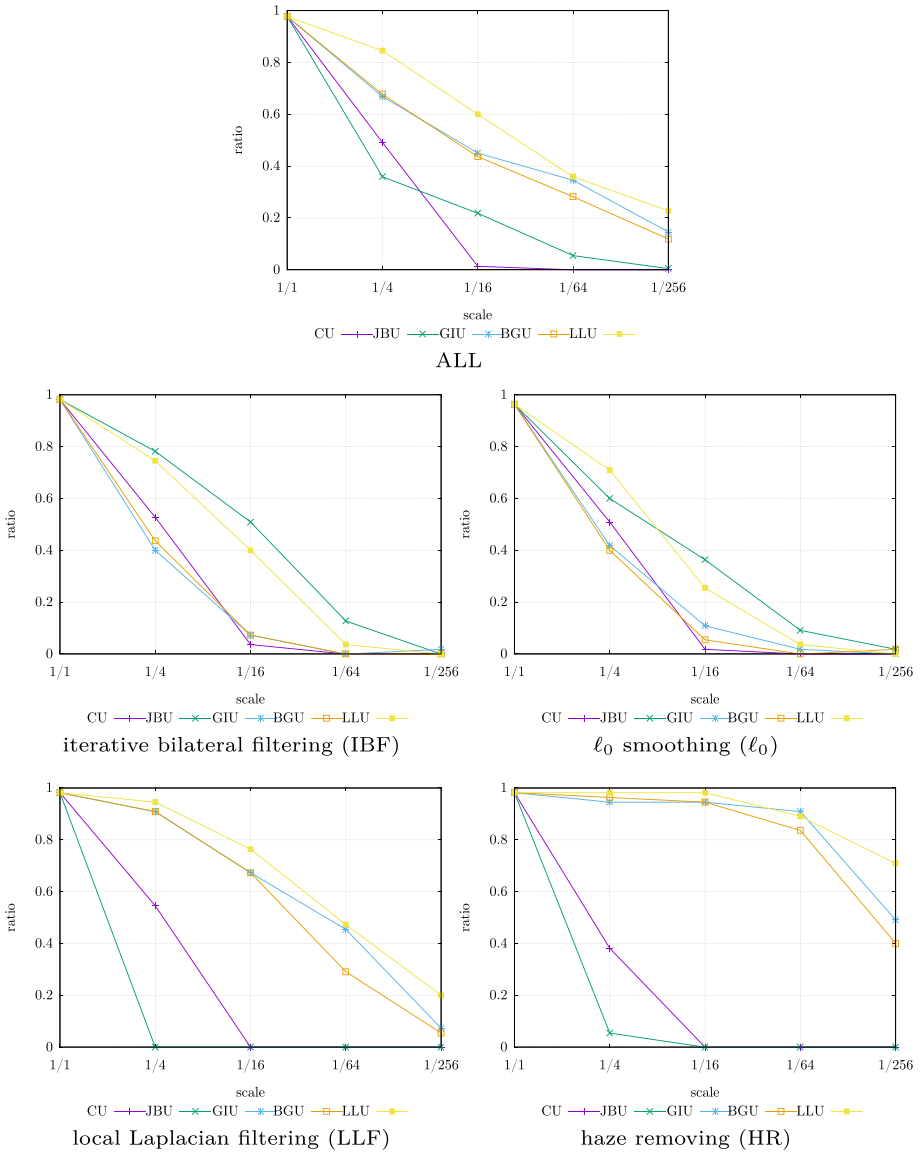


Fig. 10 Subjective IQA results for each image processing and upsampling. *y*-axis is the identification ratio, and higher is better. IBF and ℓ_0 are smoothing images, and LLF and HR are detail-enhanced images. ALL is the average of each image processing: IBF, ℓ_0 , LLF, and HR

with upsampling for generating the local LUT. The following conclusions were reached regarding the parameters of the proposed method.

- Building LUT: ℓ_2 minimization
- Building LUT parameters: $r = 2$ and $M = 7$
- LUT boundary condition: Linear
- Sampling rate: trade-off factor between accuracy and speed.

- Quantization: trade-off factor between accuracy and speed.
- Tensor upsampling: trade-off factor between accuracy and speed.

The limitation of the proposed method is color transformation. We apply the proposed method for each color channel individually because the proposed method is intended for grayscale processing. However, the proposed method does not work well when image processing includes color-twisting processing. In addition, this method is limited to image processing, which outputs images because it is a method to obtain approximated images. Speeding up methods that include decomposition results such as frequency transform, principal component analysis, and singular value decomposition in image processing [13, 15, 29, 35, 36, 48, 57, 63] is also a future challenge.

Acknowledgements This work was supported by JSPS KAKENHI Grant Number (18K19813, 21K17768, 21H03465) and Environment Research and Technology Development Fund (JPMEERF20222M01) of the Environmental Restoration and Conservation Agency of Japan. We would like to thank Editage for English language editing.

Declarations

The authors have no conflicts of interest directly relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Adams A, Baek J, Davis MA (2010) Fast high-dimensional filtering using the permutohedral lattice. *Comput Graph Forum* 29(2):753–762
2. Aubry M, Paris S, Hasinoff SW, Kautz J, Durand F (2014) Fast local laplacian filters: Theory and applications. *ACM Trans Graph* 33(5)
3. Bae S, Paris S, Durand F (2006) Two-scale tone management for photographic look. *ACM Trans Graph* 637–645
4. Barron JT, Adams A, Shih Y, Hernandez C (2015) Fast bilateral-space stereo for synthetic defocus. In: *Proc. Computer Vision and Pattern Recognition (CVPR)*
5. Buades A, Coll B, Morel JM (2005) A non-local algorithm for image denoising. In: *Proc. Computer Vision and Pattern Recognition (CVPR)*
6. Chaudhury KN (2011) Constant-time filtering using shiftable kernels. *IEEE Signal Process Lett* 18(11):651–654
7. Chaudhury KN (2013) Acceleration of the shiftable $o(1)$ algorithm for bilateral filtering and nonlocal means. *IEEE Trans Image Process* 22(4):1291–1300
8. Chaudhury KN, Sage D, Unser M (2011) Fast $o(1)$ bilateral filtering using trigonometric range kernels. *IEEE Trans Image Process* 20(12):3376–3382
9. Chen J, Adams A, Wadhwa N, Hasinoff SW (2016) Bilateral guided upsampling. *ACM Trans Graph* 35(6)
10. Chen Q, Xu J, Koltun V (2017) Fast image processing with fully-convolutional networks. In: *Proc. International Conference on Computer Vision (ICCV)* pp 2497–2506
11. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans Image Process* 16(8):2080–2095
12. Durand F, Dorsey J (2002) Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans Graph* 21(3):257–266

13. Fujita S, Fukushima N, Kimura M, Ishibashi Y (2015) Randomized redundant dct: Efficient denoising by using random subsampling of dct patches. In: Proc. ACM SIGGRAPH Asia Technical Briefs
14. Fukushima N, Fujita S, Ishibashi Y (2015) Switching dual kernels for separable edge-preserving filtering. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
15. Fukushima N, Kawasaki Y, Maeda Y (2019) Accelerating redundant dct filtering for deblurring and denoising. In: IEEE International Conference on Image Processing (ICIP)
16. Fukushima N, Sugimoto K, Kamata S (2018) Guided image filtering with arbitrary window function. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
17. Fukushima N, Takeuchi K, Kojima A (2016) Self-similarity matching with predictive linear upsampling for depth map. In: Proc. 3DTV-Conference
18. Fukushima N, Tsubokawa T, Maeda Y (2019) Vector addressing for non-sequential sampling in fir image filtering. In: IEEE International Conference on Image Processing (ICIP)
19. Gastal ESL, Oliveira MM (2011) Domain transform for edge-aware image and video processing. *ACM Trans Graph* 30(4)
20. Gastal ESL, Oliveira MM (2012) Adaptive manifolds for real-time high-dimensional filtering. *ACM Trans Graph* 31(4)
21. Gharbi M, Shih Y, Chaurasia G, Ragan-Kelley J, Paris S, Durand F (2015) Transform recipes for efficient cloud photo enhancement. *ACM Trans Graph* 34(6):228–122812. <https://doi.org/10.1145/2816795.2818127>
22. Ghosh S, Nair P, Chaudhury KN (2018) Optimized fourier bilateral filtering. *IEEE Signal Process Lett* 25(10):1555–1559
23. Hennessy JL, Patterson DA (2017) *Computer Architecture, Sixth Edition: A Quantitative Approach*, 6th edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
24. He K, Sun J (2015) Fast guided filter. *CoRR*. abs/1505.00996
25. He K, Sun J, Tang X (2009) Single image haze removal using dark channel prior. In: Proc. Computer Vision and Pattern Recognition (CVPR)
26. He K, Sun J, Tang X (2010) Guided image filtering. In: Proc. European Conference on Computer Vision (ECCV)
27. Honda S, Maeda Y, Fukushima N (2023) Dataset of subjective assessment for near-lossless image coding based on just noticeable difference. In: Proceedings of International Conference on Quality of Multimedia Experience (QoMEX)
28. Hosni A, Rhemann C, Bleyer M, Rother C, Gelautz M (2013) Fast cost-volume filtering for visual correspondence and beyond. *IEEE Trans Pattern Anal Mach Intell* 35(2):504–511
29. Ishikawa K, Oishi S, Fukushima N (2023) Principal component analysis for accelerating color bilateral filtering. In: Proceedings of International Workshop on Advanced Imaging Technology (IWAIT), vol. 12592, pp 125921. <https://doi.org/10.1117/12.2666984>. SPIE
30. Kodera N, Fukushima N, Ishibashi Y (2013) Filter based alpha matting for depth image based rendering. In: Proc. IEEE Visual Communications and Image Processing (VCIP)
31. Kopf J, Cohen MF, Lischinski D, Uyttendaele M (2007) Joint bilateral upsampling. *ACM Trans Graph* 26(3)
32. Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, Shi W (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
33. Levin A, Lischinski D, Weiss Y (2004) Colorization using optimization. *ACM Trans Graph* 23(3):689–694
34. Li X, Orchard MT (2001) New edge-directed interpolation. *IEEE Trans Image Process* 10(10):1521–1527
35. Lin Y, Ling BW-K, Xu N, Zhou X (2022) Two dimensional quaternion valued singular spectrum analysis with application to image denoising. *J Frankl Inst* 359(8):3808–3830. <https://doi.org/10.1016/j.franklin.2022.03.036>
36. Lin Y, Ling BW-K, Hu L, Zheng Y, Xu N, Zhou X, Wang X (2021) Hyperspectral image enhancement by two dimensional quaternion valued singular spectrum analysis for object recognition. *Remote Sens* 13(3). <https://doi.org/10.3390/rs13030405>
37. Liu M-Y, Tuzel O, Taguchi Y (2013) Joint geodesic upsampling of depth images. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
38. Maeda Y, Fukushima N, Matsuo H (2018) Taxonomy of vectorization patterns of programming for fir image filters using kernel subsampling and new one. *Appl Sci* 8(8):1235
39. Maeda Y, Fukushima N, Matsuo H (2018) Effective implementation of edge-preserving filtering on cpu microarchitectures. *Appl Sci* 8(10)
40. Matsuo T, Fujita S, Fukushima N, Ishibashi Y (2015) Efficient edge-awareness propagation via single-map filtering for edge-preserving stereo matching. In: Proc. SPIE, vol. 9393

41. Matsuo T, Fukushima N, 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)
42. Min D, Lu J, Do MN (2012) Depth video enhancement based on weighted mode filtering. *IEEE Trans Image Process* 21(3):1176–1190. <https://doi.org/10.1109/TIP.2011.2163164>
43. Min D, Choi S, Lu J, Ham B, Sohn K, Do MN (2014) Fast global image smoothing based on weighted least squares. *IEEE Trans Image Process* 23(12):5638–5653
44. Murooka Y, Maeda Y, Nakamura M, Sasaki T, Fukushima N (2018) Principal component analysis for acceleration of color guided image filtering. In: Proc. International Workshop on Frontiers of Computer Vision (IW-FCV)
45. Paris S, Hasinoff SW, Kautz J (2011) Local laplacian filters: Edge-aware image processing with a laplacian pyramid. *ACM Trans Graph* 30(4)
46. Sugimoto K, Kamata S (2015) Compressive bilateral filtering. *IEEE Trans Image Process* 24(11):3357–3369
47. Sugimoto K, Fukushima N, 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)
48. Sugimoto K, Fukushima N, Kamata S (2019) 200 fps constant-time bilateral filter using svd and tiling strategy. In: IEEE International Conference on Image Processing (ICIP)
49. Sumiya Y, Otsuka T, Maeda Y, Fukushima N (2021) Gaussian fourier pyramid for local laplacian filter. *IEEE Signal Processing Letters*
50. Tajima H, Fukushima N, Maeda Y, Tsubokawa T (2019) Local lut upsampling for acceleration of edge-preserving filtering. In: Proc. International Conference on Computer Vision Theory and Applications (VISAPP)
51. Tajima H, Tsubokawa T, Maeda Y, Fukushima N (2020) Fast local lut upsampling. In: Proc. International Conference on Computer Vision Theory and Applications (VISAPP)
52. Telea A (2004) An image inpainting technique based on the fast marching method. *J Graph Tools* 9(1):23–34
53. Timofte R, Agustsson E, Van Gool L, Yang M-H, Zhang L (2017) Ntire 2017 challenge on single image super-resolution: Methods and results. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)
54. Tomasi C, Manduchi R (1998) Bilateral filtering for gray and color images. In: Proc. International Conference on Computer Vision (ICCV), pp 839–846
55. Tsubokawa T, Nakamura M, Maeda Y, Fukushima N (2019) Tiling parallelization of guided image filtering. In: Proc. International Workshop on Frontiers of Computer Vision (IW-FCV)
56. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. *IEEE Trans Image Process* 13(4):600–612
57. Xu Y, Liu B, Liu J, Riemenschneider S (2006) Two-dimensional empirical mode decomposition by finite elements. *Proc R Soc A* 462(2074):3081–3096. <https://doi.org/10.1098/rspa.2006.1700>
58. Xue W, Zhang L, Mou X, Bovik AC (2014) Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. *IEEE Trans Image Process* 23(2):684–695. <https://doi.org/10.1109/TIP.2013.2293423>
59. Xu L, Lu C, Xu Y, Jia J (2011) Image smoothing via l0 gradient minimization. *ACM Trans Graph* 30(6)
60. Yang Q, Tan KH, Ahuja N (2009) Real-time o(1) bilateral filtering. In: Proc. Computer Vision and Pattern Recognition (CVPR) pp 557–564
61. Yang Q, Yang R, Davis J, Nister D (2007) Spatial-depth super resolution for range images. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
62. Yu G, Sapiro G (2011) Dct image denoising: A simple and effective image denoising algorithm. *Image Processing On Line* 1:1
63. Zabalza J, Ren J, Zheng J, Han J, Zhao H, Li S, Marshall S (2015) Novel two-dimensional singular spectrum analysis for effective feature extraction and data classification in hyperspectral imaging. *IEEE Trans Geosci Remote Sens* 53(8):4418–4433. <https://doi.org/10.1109/TGRS.2015.2398468>
64. Zeyde R, Elad M, Protter M (2010) On single image scale-up using sparse-representations. In: International Conference on Curves and Surfaces. Springer
65. Zhang L, Wu X (2006) An edge-guided image interpolation algorithm via directional filtering and data fusion. *IEEE Trans Image Process* 15(8):2226–2238
66. Zhang R, Isola P, Efros AA, Shechtman E, Wang O (2018) The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 586–595. <https://doi.org/10.1109/CVPR.2018.00068>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.