A Fixed-Point Local Tone Mapping Operation for HDR Images

Toshiyuki DOBASHI*, Masahiro IWAHASHI† and Hitoshi KIYA*
*Tokyo Metropolitan University, Tokyo, 191–0065, Japan
†Nagaoka University of Technology, Niigata, 980-2188, Japan

***Abstract*—This paper proposes a fixed-point local tone mapping operation (TMO) for high dynamic range (HDR) images. A TMO is classified in two types: local and global. Although a local TMO offers better results than global one, it requires more resources such as a computational cost and memory space. The proposed method uses fixed-point arithmetic with short data to solve this problem. The method uses an intermediate format which is composed of 8-bit mantissa part and 8-bit exponent part instead of IEEE754 standard floating-point format. Moreover, these mantissa part and exponent part are processed separately as two integer numbers. As a result, the method reduces the memory space. In addition, the method also reduces numerical range of calculations; it facilitates to implement the method with fixed-point arithmetic. The experimental results show that the method reduces the memory and computational costs, and offers high quality of tone mapped images comparable to the conventional method.**

I. Introduction

High dynamic range (HDR) images are increasingly being used in many fields: photography, computer graphics, medical imaging, and others. In contrast, display devices which can express the pixel values of HDR images are not popular yet. Therefore, the importance of a tone mapping operation (TMO) which generates a low dynamic range (LDR) image from an HDR image is growing.

Various research works on tone mapping have so far been done [1]–[7]. Many of these focus on compression techniques or quality of tone mapped images. Most of those were concentrated on finding a tone mapping function suitable for human visual systems [1]–[4]. Recently, some papers dealt with reducing communication cost combining with data compression technologies [5]–[7]. Unlike these previous research works, this paper focuses on "resources" of a TMO such as a computational cost and memory space for easy implementation.

There are two types of TMOs: global operators and local operators. The former use the same tone mapping function for all pixels in the HDR image. On the other hand, the latter try to find optimal function for each pixel. They have advantages in terms of contrast preservation compared with the former. However, local TMOs require more resources than global ones.

To reduce resources of a TMO, an integer TMO approach which deals with resource reduction is proposed in [8]–[12]. The method in [8], [9] treats a floating-point number as two 8-bit integer numbers which correspond to a exponent part and a mantissa part, and applies tone mapping to these integer numbers separately. The method reduces the memory space by using two 8-bit integer data instead of 64-bit floating-point data such as IEEE754 [13]. Moreover, using 8-bit integer data facilitates executing calculations with fixed-point arithmetic because it eases the limitation of the bit length. Fixed-point arithmetic is often utilized in image processing and embedded systems because of the advantages such as low-power consumption, the small circuit size and high-speed computing [14]–[16]. The method in [10]–[12] implements the integer TMO with fixed-point arithmetic, and therefore it reduces the computational cost as well. However, these methods are all global TMOs.

On the other hand, fast local TMOs are proposed in [17], [18]. These methods utilize parallel processing for high-speed computing. The method in [17] replaces a Gaussian filter with a box filter to reduce a computational cost during convolution. However, these are for graphics processing unit (GPU) implementation. In other words, these are not for low-memory or fixed-point implementation.

This paper solves these problem by extending an integer TMO for local operators. The proposed method processes the exponent part and the mantissa part separately in a local TMO including the convolution. Moreover, the method can conduct all the calculations of the TMO with only fixed-point arithmetic. By these features, the method can be executed under limited resources, such as processors without a FPU or low-memory. The experiments and evaluation confirmed that the proposed method reduces the computational cost and the memory cost, and keeps the quality of tone mapped images, compared to the conventional method with floating-point arithmetic.

II. Tone Mapping Operation

The procedure of a TMO is described in this section. It generates an integer LDR image from a floating-point HDR image. This section describes "Photographic Tone Reproduction" which is one of well-known tone mapping procedures [1]. In this section, the global operator is described at first. Then, the difference between the global and the local is described.

A. Global Operator

(a) World Luminance
First, it calculates the world luminance $L_w(x,y)$ of each pixel x,y as

$$L_w(x,y) = 0.27R(x,y) + 0.67G(x,y) + 0.06B(x,y), \quad (1)$$

where $R(x, y)$, $G(x, y)$, and $B(x, y)$ are floating-point RGB values of the input HDR image.

(b) Geometric Mean
The geometric mean $\bar{L}_w$ of the world luminance $L_w(x, y)$ is defined as

$$\bar{L}_w = \exp\left(\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\log_e\left(L_w(x, y)\right)\right), \tag{2}$$

where M and N are the width and the height of the input HDR image, respectively. Note that Eq. (2) has singularity due to zero value of $L_w(x, y)$. It is avoided by introducing a small value in [1]. However, its arbitrariness is not negligible for pixel values in a floating-point format, since its pixel value is also small. Therefore, only non-zero values are included in the geometric mean in this paper.

(c) Scaled Luminance
Next, the scaled luminance $L(x, y)$ is calculated as

$$L(x, y) = \alpha \cdot \frac{L_w(x, y)}{\bar{L}_w}, \tag{3}$$

where $\alpha \in [0, 1]$ is a parameter called "key value."

(d) Display Luminance
Then, display luminance $L_d(x, y)$ is computed with a tone mapping function $y()$ as

$$L_d(x, y) = y(L(x, y)), \tag{4}$$

where Reinhard's global operator [1] is specified as

$$y_{\text{Global}}(L(x, y)) = \frac{L(x, y)}{1 + L(x, y)}. \tag{5}$$

(e) LDR Image Generation
The values $C_F(x, y) \in \{R_F(x, y), G_F(x, y), B_F(x, y)\}$ which are floating-point RGB pixels calculated by

$$C_F(x, y) = L_d(x, y) \cdot \frac{C(x, y)}{L_w(x, y)}, \tag{6}$$

where $C(x, y) \in \{R(x, y), G(x, y), B(x, y)\}$ are the floating-point RGB values of input HDR image.
Finally, the 24-bit color integer RGB values $C_I(x, y) \in \{R_I(x, y), G_I(x, y), B_I(x, y)\}$ of the resulting LDR image are generated as

$$C_I(x, y) = \text{round}\left(C_F(x, y) \cdot 255\right), \tag{7}$$

where $\text{round}(x)$ rounds x to the nearest integer value.

B. Local Operator

The local operator is obtained by replacing $L(x, y)$ in the denominator of Eq. (5) to $V(x, y, s)$ which is derived from $L(x, y)$ and the Gaussian filter $G(x, y, s)$ with various scale s as

$$V(x, y, s) = L(x, y) \otimes G(x, y, s), \tag{8}$$

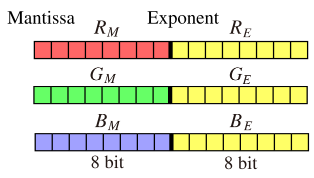

Fig. 1. The bit allocation of the proposed intermediate format.

where $\otimes$ is the convolution operator. The method finds the largest scale s_m which satisfies the following condition

$$|W(x, y, s_m)| < \epsilon, \tag{9}$$

where

$$W(x, y, s_i) = \frac{V(x, y, s_i) - V(x, y, s_{i+1})}{2^\phi \alpha / s^2 + V(x, y, s_i)}. \tag{10}$$

The parameter ϕ is a sharpening factor. In [1], $\phi = 8$ and $\epsilon = 0.05$ are used as default values.
The local operator is finally defined as

$$y_{\text{Local}}(L(x, y)) = \frac{L(x, y)}{1 + V(x, y, s_m)}. \tag{11}$$

As described above, the global operator uses the same function for all pixel. On the other hand, the local one tries to find the optimal function for each pixel. Because of this, although it gives better result than the global one, more resources are required.

III. Proposed Method

In this section, the proposed intermediate format utilized in the method is described at first. Then, the integer TMO for local operators is described.

A. Intermediate Format

An input HDR image is converted to the intermediate format (Figure 1) at the first step of the proposed method. The proposed method can be applied for various HDR image formats by converting the input image to the intermediate format. This format can be applied for the RGBE [19], the OpenEXR [20] and the long-integer formats [11], [12]. Figure 1 shows the bit allocation of the intermediate format. The encode functions which yield the exponent part I_E and the mantissa part I_M of each RGB channel I are defined as

$$I_E = \lceil \log_2 I + 128 \rceil, \tag{12}$$
$$I_M = \left\lfloor I \cdot 2^{136 - I_E} \right\rfloor, \tag{13}$$

where $\lceil x \rceil$ rounds x to the nearest integer greater than or equal to x, and $\lfloor x \rfloor$ rounds x to the nearest integer less than or equal to x. On the other hand, the decode function which yields the original RGB value from the intermediate format is defined as

$$I = (I_M + 0.5) \cdot 2^{I_E - 136}. \tag{14}$$

B. Integer TMO for Local Operator

The integer TMO is defined as the TMO which is implemented with integer input and integer output. The integer TMO defines new processes and replaces each tone mapping process

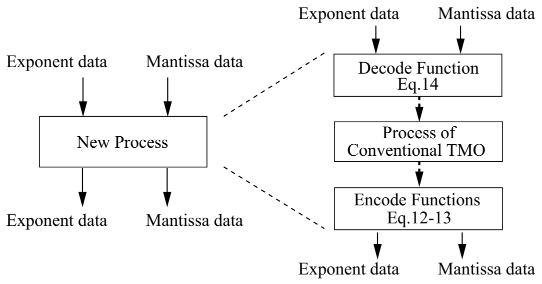

Fig. 2. A new process defined in the proposed integer TMO.

by them. These new processes are composite functions shown in Figure 2. In the proposed TMO, the numerical range in the processes is significantly reduced because the exponent part and the mantissa part are separated as two integer numbers. Note that this technique of the integer TMO works well by using the proposed intermediate format. The technique does not work well for the IEEE754 format because it has denormalized numbers as well as the OpenEXR [9].

The proposed integer TMO converts RGB values $C(x,y)$ into the intermediate format at the first step. The exponent parts $C_E(x,y) \in \{R_E(x,y), G_E(x,y), B_E(x,y)\}$ and the mantissa parts $C_M(x,y) \in \{R_M(x,y), G_M(x,y), B_M(x,y)\}$ are calculated as

$$C_E(x,y) = \lceil \log_2 C(x,y) + 128 \rceil, \tag{15}$$

$$C_M(x,y) = \left\lfloor C(x,y) \cdot 2^{136 - C_E(x,y)} \right\rfloor. \tag{16}$$

(a') World Luminance
The exponent part $L_{w_E}(x,y)$ and the mantissa part $L_{w_M}(x,y)$ of the world luminance $L_w(x,y)$ are given as

$$L_{wE}(x,y) = \lceil \log_2 ML(x,y) - 8 \rceil, \tag{17}$$

$$L_{wM}(x,y) = \left\lfloor ML(x,y) \cdot 2^{-L_{wE}(x,y)} \right\rfloor, \tag{18}$$

$$\begin{aligned} ML(x,y) = 0.27(R_M(x,y)+0.5)\cdot 2^{R_E(x,y)} + \\ 0.67(G_M(x,y)+0.5)\cdot 2^{G_E(x,y)} + \\ 0.06(B_M(x,y)+0.5)\cdot 2^{B_E(x,y)}, \end{aligned} \tag{19}$$

where $ML(x,y)$ is set to zero if $R_M(x,y) = G_M(x,y) = B_M(x,y) = 0$. $L_{wE}(x,y)$ and $L_{wM}(x,y)$ are also set to zero in this case.

(b') Geometric Mean
The exponent part $\bar{L}_{wE}(x,y)$ and the mantissa part $\bar{L}_{wM}(x,y)$ of the geometric mean $\bar{L}_w$ are derived as

$$\bar{L}_{wE} = \lceil SL_{wM} + SL_{wE} + 128 \rceil, \tag{20}$$

$$\bar{L}_{wM} = \left\lfloor 2^{SL_{wM} + SL_{wE} - \bar{L}_{wE} + 136} \right\rfloor, \tag{21}$$

$$SL_{wE} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (L_{wE}(x,y) - 136), \tag{22}$$

$$SL_{wM} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \log_2 (L_{wM}(x,y) + 0.5). \tag{23}$$

(c') Scaled Luminance
The exponent part $L_E(x,y)$ and the mantissa part $L_M(x,y)$ of the scaled luminance $L(x,y)$ are calculated as

$$L_E(x,y) = \lceil \log_2(AL_w(x,y)) + L_{wE}(x,y) - \bar{L}_{wE} + 128 \rceil, \tag{24}$$

$$L_M(x,y) = \left\lfloor AL_w(x,y) \cdot 2^{136 + L_{wE}(x,y) - L_E(x,y) - \bar{L}_{wE}} \right\rfloor, \tag{25}$$

$$AL_w(x,y) = k \cdot \frac{L_{wM}(x,y) + 0.5}{\bar{L}_{wM} + 0.5}. \tag{26}$$

(d') Display Luminance
The exponent part $L_{dE}(x,y)$ and the mantissa part $L_{d_M}(x,y)$ of the display luminance $L_d(x,y)$ is calculated with a tone mapping function. This calculation depends on the tone mapping function to be used. Here, the local operator of Eq. (11) is used as an example,

$$L_{dE}(x,y) = \lceil \log_2(FL(x,y)) + 128 \rceil, \tag{27}$$

$$L_{dM}(x,y) = \left\lfloor FL(x,y) \cdot 2^{136 - L_{dE}(x,y)} \right\rfloor, \tag{28}$$

$$FL(x,y) = \frac{(L_M(x,y) + 0.5) \cdot 2^{L_E(x,y)}}{2^{136} + (V_M(x,y,s_m) + 0.5) \cdot 2^{V_E(x,y,s_m)}}. \tag{29}$$

The exponent $V_E(x,y,s)$ and mantissa $V_M(x,y,s)$ of $V(x,y,s)$ are defined as

$$\begin{aligned} V_E(x,y,s) = \lfloor \log_2((L_M(x,y)+0.5)\cdot 2^{L_E(x,y)-136} \\ \otimes (G_M(x,y,s)+0.5)\cdot 2^{G_E(x,y,s)-136} + 128 \rfloor, \end{aligned} \tag{30}$$

$$\begin{aligned} V_M(x,y,s) = \lfloor (L_M(x,y)+0.5)\cdot 2^{L_E(x,y)} \\ \otimes (G_M(x,y,s)+0.5)\cdot 2^{G_E(x,y,s)}) \cdot 2^{V_E(x,y,s)-136} \rfloor, \end{aligned} \tag{31}$$

where $G_E(x,y,s)$ and $G_M(x,y,s)$ are the exponent part and the mantissa part of the Gaussian filter $G(x,y,s)$, and $\otimes$ is the convolution operator.
The method finds the largest scale s_m which satisfies following condition

$$(W_M(x,y,s_m)) + 0.5) \cdot 2^{W_E(x,y,s_m)-136} < \epsilon, \tag{32}$$

where

$$W_E(x,y,s_i) = \lceil \log_2(|FW(x,y,s_i)|) + 128 \rceil, \tag{33}$$

$$W_M(x,y,s_i) = \left\lfloor |FW(x,y,s_i)| \cdot 2^{136 - V_E(x,y,s)} \right\rfloor, \tag{34}$$

$$FW(x,y,s_i) =$$

$$\frac{(V_M(x,y,s_i)+0.5)\cdot 2^{V_E(x,y,s_i)} - (V_M(x,y,s_{i+1})+0.5)\cdot 2^{V_E(x,y,s_{i+1})}}{2^{\phi+136}\alpha/s^2 + (V_M(x,y,s_i)+0.5)\cdot 2^{V_E(x,y,s_i)}}. \tag{35}$$

However, these calculations of $V_E(x,y,s)$ and $V_M(x,y,s)$ require high computational cost. The method can compute these calculations efficiently by following steps.
First, the provisional exponent part $V'_E(x,y,s)$ is calculated as

$$\begin{aligned} V'_E(x,y,s) = \max\{G_E(u,v,s) + L_E(x-u, y-v)), \\ u = 0,1,\cdots A-1, v = 0,1,\cdots,B-1\} - 136, \end{aligned} \tag{36}$$

where A and B are the width and the height of the Gaussian filter, respectively. Next, the provisional mantissa part $V'_M(x,y,s)$ is calculated as

$$V'_M(x,y,s) = \sum_{u=0}^{A-1}\sum_{v=0}^{B-1}(G_M(u,v,s)+0.5)\cdot(L_M(x-u,y-v)+0.5)\cdot 2^{SV'_M}, \quad (37)$$

$$SV'_M = G_E(u,v,s)+L_E(x-u,y-v)-V'_E(x,y,s)-136. \quad (38)$$

Then, $V_E(x,y,s)$ is obtained by adding a overflowed value of $V'_M(x,y,s)$ to $V'_E(x,y,s)$ as

$$O = \lfloor \log_2(V'_M(x,y,s))-7\rfloor, \quad (39)$$

$$V_E(x,y,s) = V'_E(x,y,s)+O. \quad (40)$$

Finally, $V_M(x,y,s)$ is obtained by

$$V_M(x,y,s) = \left\lfloor V'_M(x,y,s)\cdot 2^{-O}\right\rfloor. \quad (41)$$

(e') LDR Image Generation

The 24-bit color RGB value $C_I(x,y)$ of the resulting LDR image is obtained by

$$C_I(x,y) = \text{round}\left(RL(x,y)\cdot 2^{C_E(x,y)+L_{dE}(x,y)-L_{wE}(x,y)-136}\cdot 255\right), \quad (42)$$

$$RL(x,y) = \frac{(L_{dM}(x,y)+0.5)(C_M(x,y)+0.5)}{L_{wM}(x,y)+0.5}. \quad (43)$$

In the above processes, the input and output data of each calculation are all 8-bit integer data. The memory cost can be reduced by using integer data. The next section describes fixed-point implementation of the method.

IV. Fixed-Point Implementation

In the integer TMO, only the data is converted to integer, and the memory cost is reduced. However, the internal arithmetic of the integer TMO is still with floating-point. This section describes the way to execute the internal arithmetic with fixed-point arithmetic. The proposed method introduces fixed-point arithmetic to reduce the computational cost as well.

Most of equations can be calculated with fixed-point arithmetic because each variable is expressed in 8-bit integer [11]. Nevertheless, Eq. (35) is difficult to be calculated without floating-point arithmetic because the numerical range can be very wide. Eq. (35) is used to find the scale s_m. The equation does not affect on the pixel values directly. Therefore, the method calculates it by branching and approximation. First, the method deforms Eq. (35) as follows

$$FW(x,y,s_i) = \frac{1-\frac{V_M(x,y,s_{i+1})+0.5}{V_M(x,y,s_i)+0.5}\cdot 2^{V_E(x,y,s_{i+1})-V_E(x,y,s_i)}}{\frac{\alpha/s^2}{V_M(x,y,s_i)+0.5}\cdot 2^{\phi+136-V_E(x,y,s_i)}+1} \quad (44)$$

Furthermore, the method branches Eq. (44) into three cases and approximates it based on the power of two in the denominator as follows.

Case 1:If $\phi+136-V_E(x,y,s_i) > 16$ in Eq. (44), '1' in the denominator and the numerator can be ignored because the left part of the denominator is very large, and so it is approximated as

$$FW(x,y,s_i) \approx -\frac{V_M(x,y,s_{i+1})+0.5}{\alpha/s^2}\cdot 2^{V_E(x,y,s_{i+1})-\phi-136}. \quad (45)$$

Case 2:If $\phi+136-V_E(x,y,s_i) < -16$ in Eq. (44), the left part of the denominator can be ignored because it is very small, and so it is approximated as

$$FW(x,y,s_i) \approx 1-\frac{V_M(x,y,s_{i+1})+0.5}{V_M(x,y,s_i)+0.5}\cdot 2^{V_E(x,y,s_{i+1})-V_E(x,y,s_i)}. \quad (46)$$

Case 3:Otherwise, it can be calculated with fixed-point arithmetic.

In addition, the method uses pre-calculated tables for calculations of 2^x (in Eq. (21)) and $\log_2$ (in Eq. (23)). Each table consists of 16×256 bits.

The method can calculate all equations of the TMO with only fixed-point arithmetic by these branching, approximation, and tables. Note that the conventional method [1] consists of floating-point data and floating-point arithmetic. In contrast, the proposed method consists of integer data and fixed-point arithmetic.

V. Experimental Results

In this section, the proposed method with the intermediate format and the conventional method [1] with floating-point numbers were compared. The peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [21] of the resulting LDR images were measured to evaluate the accuracy of the proposed method. Moreover, the memory space and processing time comparisons between the methods were also performed. The 32 HDR images in the RGBE format, 42 HDR images in the OpenEXR format, and 74 HDR images in the long-integer (16-bit) format were used in these experiments. The conventional method was executed with 64-bit floating-point arithmetic. On the other hand, the proposed method was executed with 32-bit fixed-point arithmetic. All floating-point numbers were computed and stored in the IEEE754 double-precision (64-bit) format. The parameters $\phi = 8$, $\epsilon = 0.05$, and $\alpha = 0.5$ were used.

A. Tone Mapped LDR Image Quality

Table I shows the maximum, minimum, and average PSNR and average SSIM. In all cases, high PSNR and SSIM values were obtained in the proposed method. It was confirmed that effects of the errors due to fixed-point arithmetic on the image quality were sufficiently small. Moreover, the method offered high quality LDR images with various HDR image formats.

TABLE I
THE MAXIMUM, MINIMUM, AND AVERAGE PSNR AND THE AVERAGE SSIM OF THE METHODS.

	PSNR [dB]			SSIM
	Maximum	Minimum	Average	
RGBE	54.36	46.56	50.52	0.9992
OpenEXR	61.08	39.95	51.48	0.9991
Long-integer	59.04	43.63	52.27	0.9989

TABLE II
THE MEMORY SPACE OF THE CONVENTIONAL METHOD [1] AND THE PROPOSED METHOD.

The data used in the methods	Memory Space [bits]	
	Conventional [1]	Proposed
HDR RGB Value	$M \times N \times 192$	$M \times N \times 48$
World Luminance	$M \times N \times 64$	$M \times N \times 16$
Geometric Mean	64	16
Scaled Luminance	$M \times N \times 64$	$M \times N \times 16$
Display Luminance	$M \times N \times 64$	$M \times N \times 16$
Gaussian Filter	$A \times B \times 64$	$A \times B \times 16$

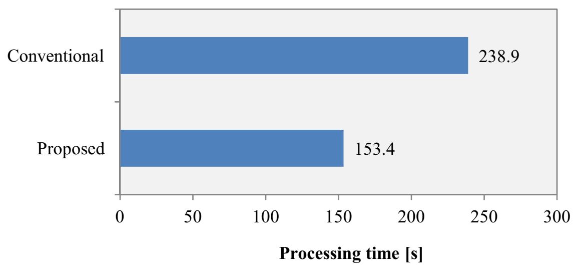

Fig. 3. The processing time of the proposed method and the conventional method [1].

B. Memory Space

Table II shows the memory space of each calculation with $M \times N$ sized HDR image and $A \times B$ sized Gaussian filter. It indicates that the proposed method reduces memory resources compared to the conventional method [1]. The memory space of the proposed method is 75% less than the conventional method [1].

C. Comparison of the Processing Time

This experiment applied tone mapping for an HDR image with 346 $\times$ 512 pixels in the OpenEXR format using the proposed method with fixed-point arithmetic and the conventional method [1] with floating-point arithmetic. The experimental environment was with Marvell PXA270 ARM Processor 624MHz and 128MB RAM. Note that this processor does not have a FPU.

Figure 3 compares the processing time of the methods. The proposed method was 1.6 times faster than the conventional method. Therefore, this experiment confirmed that the proposed method reduced the computational cost by using fixed-point arithmetic.

VI. Conclusion

This paper proposed a local TMO with fixed-point arithmetic and low-memory. The proposed method processes the exponent part and the mantissa part of an HDR image separately as two 8-bit integer numbers; the numerical range in the tone mapping process is greatly reduced. From this, the method can be implemented with fixed-point arithmetic and low-memory. The experimental results confirmed that the method reduces the memory and computational cost, and offers high quality of tone mapped images comparable to the conventional method.

References

[1] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic Tone Reproduction for Digital Images," *ACM Trans. Graphics*, Vol.21, No.3, p.267-276, Jul. 2002.

[2] E. Reinhard, G. Ward, S. Pattanaik, P. Debevec, W. Heidrich, and K. Myszkowski, "High Dynamic Range Imaging - Acquisition, Display and Image based Lighting," Morgan Kaufmann, 2010

[3] F. Drago, K. Myszkowski, T. Annen, and N. Chiba, "Adaptive logarithmic mapping for displaying high contrast scenes," *Computer Graphics Forum*, Vol.22, p.419-426, 2003.

[4] R. Fattal, D. Lischinski, and M. Werman, "Gradient Domain High Dynamic Range Compression," *ACM Trans. Graphics*, Vol.21, No.3, pp.249-256, July. 2002.

[5] M. Iwahashi and H. Kiya, "Efficient Lossless Bit Depth Scalable Coding for HDR Images," *Proc. APSIPA ASC*, no.OS.49-IVM.17-5, Dec. 2012.

[6] M. Iwahashi and H. Kiya, "Two Layer Lossless Coding of HDR Images," *Proc. IEEE ICASSP*, pp.1340-1344, May 2013.

[7] R. Xu, S. N. Pattanaik, and C. E. Hughes, "High-Dynamic-Range Still Image Encoding in JPEG2000," *IEEE Trans. Computer Graphics and Applications*, Vol.25, No.6, pp.57-64, Nov. 2005.

[8] T. Murofushi, M. Iwahashi, and H. Kiya, "An Integer Tone Mapping Operation for HDR Images Expressed in Floating Point Data," *Proc. IEEE ICASSP*, pp.2479-2483, May 2013.

[9] T. Murofushi, T. Dobashi, M. Iwahashi, and H. Kiya, "An Integer Tone Mapping Operation for HDR Images in OpenEXR with Denormalized Numbers," *Proc. IEEE ICIP*, no.TEC-P10.6, Oct. 2014.

[10] T. Dobashi, T. Murofushi, M. Iwahashi, and H. Kiya, "A Fixed-Point Tone Mapping Operation for HDR Images in the RGBE Format," *APSIPA ASC 2013*, no.OS.37-IVM.16-4, Nov. 2013.

[11] T. Dobashi, A. Tashiro, M. Iwahashi, and H. Kiya, "A fixed-point implementation of tone mapping operation for HDR images expressed in floating-point format," *APSIPA Trans. Signal and Info. Process.*, vol.3, no.e11, pp.1-11, Oct. 2014.

[12] T. Dobashi, M. Iwahashi, and H. Kiya, "A Unified Tone Mapping Operation for HDR Images Including Both Floating-Point and Integer Data," *LNCS*, Y. -S. Ho, J. Sang, Y. M. Ro, J. Kim, and F. Wu, Eds. Springer-Verlag, vol.9314, pp.321-333, Sep. 2015.

[13] "Information technology - Microprocessor Systems - Floating-Point arithmetic," ISO/IEC/IEEE 60559, 2011.

[14] T. Viitanen, P. Jaaskelainen, O. Esko, and J. Takala, "Simplified Floating-point Division and Square Root," *Proc. IEEE ICASSP*, pp.2707-2711, May 2013.

[15] C. H. Lampert, and O. Wirjadi, "Anisotropic Gaussian Filtering Using Fixed Point Arithmetic," *Proc. IEEE ICIP*, pp.1565-1568, Oct. 2006.

[16] K. J. Hass, "Synthesizing Optimal Fixed-Point Arithmetic for Embedded Signal Processing," *Proc. IEEE MWSCAS*, pp.61-64, Aug. 2010.

[17] M. Slomp, and M. M. Oliveria, "Real-Time Photographic Local Tone Reproduction Using Summed-Area Tables," *Computer Graphics International*, pp.82-91, 2008.

[18] Q. Tian, J. Duan, and G. Qiu, "GPU-accelerated local tone-mapping for high dynamic range images," *Proc. IEEE ICIP*, pp.377-380, Oct. 2012.

[19] G. Ward, "Real Pixels," Graphics Gems 2., pp.80-83, San Diego, CA, USA: Academic Press, 1992.

[20] F. Kainz, R. Bogart, and D. Hess, "The Openexr Image File Format," *ACM SIGGRAPH Technical Sketches*, 2003.

[21] Z. Wang, A. C. Bovik, H. R. Seikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Processing*, Vol.13, No.4, pp.600-612, Apr. 2004.