Local tone mapping algorithm and hardware implementation

J. Yang, A. Hore and O. Yadid-Pecht✉

A novel tone mapping algorithm and hardware implementation for displaying wide dynamic range (WDR) images are proposed. The algorithm processes WDR images in a pixel-by-pixel fashion in the logarithmic domain, and it uses the block-interpolated minimum and maximum pixel values. The hardware implementation can achieve real-time processing of WDR images and is resource-efficient. Experimental results show that the proposed algorithm and hardware implementation can produce images with good brightness and high contrast.

Introduction: The dynamic range is defined as the ratio of the intensity of the brightest point to the intensity of the darkest point in a scene or image. For natural scenes, this ratio can reach up to the order of millions. Wide dynamic range (WDR) images, also called high dynamic range (HDR) images, are images that exhibit larger dynamic range than common photographs. Generally speaking, there are two main methods of obtaining WDR images: one can capture WDR image either in different frames [1] or via advanced image sensors in the same frame [2, 3]. However, the dynamic range of WDR images usually exceeds the dynamic range of conventional display devices, making a proper direct display of WDR image quite impossible. To address the issue of displaying WDR on conventional and existing LDR display devices, tone mapping algorithms have been developed, which aim at efficiently compress WDR image in order to produce LDR images that fit LDR display devices with the implicit or explicit constraints of preserving the original image quality to some extent (contrast, visible details, brightness or texture to name a few). Tone mapping algorithms can be classified into two categories: global and local tone mapping algorithms. Global tone mapping algorithms employ a single function for all pixels and disregard pixel's neighbour statics. In general, they are relatively easy to implement in hardware [4]. However, they may be prone to loss of details in images as well as insufficient contrast. Local tone mapping takes pixel neighbour statistics into account, and they can produce images with more contrast and brightness than global tone mapping algorithms. However, many local tone mapping algorithms are computationally expensive and require a significant amount of hardware resources for implementation [5–8]. In this Letter, we first propose a novel local tone mapping algorithm which processes the WDR image in a pixel-by-pixel manner, and then we present the hardware implementation of the algorithm.

Algorithm: Fig. 1 shows the overall processing flow of the proposed algorithm. It first divides the WDR image into $m \times n$ blocks (white rectangles), then maximum and minimum values of each block are obtained (shown as circles in Fig. 1). The maximum and minimum values constitute two $m \times n$ matrices. The two matrices are then bilinearly interpolated to expand their dimension to the size of the WDR image. We denote the two matrices as $M_{\max}$ and $M_{\min}$. At image location (i, j), the WDR pixel value is $p(i, j)$ and there are two corresponding values $M_{\max}(i, j)$ and $M_{\min}(i, j)$ in the matrices. Our algorithm tone maps the pixel value using logarithm function

$$d(i,j) = \frac{\log(p(i,j)) - \log(M_{\min}(i,j))}{\log(M_{\max}(i,j)) - \log(M_{\min}(i,j))} \times (D_{\max} - D_{\min}) + D_{\min} \tag{1}$$

where $D_{\max}$ and $D_{\min}$ are the maximum and minimum display levels of the visualisation devices (which are usually 255 and 0). Equation (1) tone maps a WDR image using the logarithmic function in a pixel-by-pixel fashion. Note that in (1), the compression level for a pixel $p(i, j)$ is determined by the $M_{\max}(i, j)$ and $M_{\min}(i, j)$ values which take local pixel fluctuations into account, which gives our algorithm its local (instead of global) flavour. The compression level between adjacent pixels is close to each other since the values in $M_{\max}$ and $M_{\min}$ arrays are interpolated. The gradually changing compression level of pixels in the algorithm helps in reducing any visible artefacts.

Hardware implementation: The hardware architecture of the proposed algorithm is shown in Fig. 2. It mainly consists of six modules: *block div module*, *pixel status module*, *parameter module*, *regfile module*, *interp module* and *compute module*. The *pixel status module* keeps track of the row and column number of the current input pixel and outputs the row and column number pair (i, j) to *block div module* where decisions are made to keep the maximum and minimum pixel values of each block. The values are then passed to and stored in the *regfile module* for the use of the next frame because when operating in real time, we reasonably assume without any exaggeration that there is very little variation between successive image frames [5, 8]. Hence, image statistics (like minimum and maximum) acquired from one frame can be used to process the subsequent frame. The *interp module* fetches corresponding data from a *regfile module* based on the row and column number pair (i, j) and interpolates the corresponding $M_{\max}(i, j)$ and $M_{\min}(i, j)$ values. The *compute module* carries out the calculation of (1) to obtain the final tone mapped value of the pixel $p(i, j)$. The *parameter module* stores user-defined parameters such as image resolution and predefined size of each block.

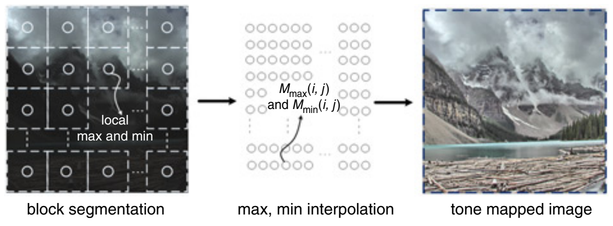

Fig. 1 *Overall processing flow of the proposed algorithm*

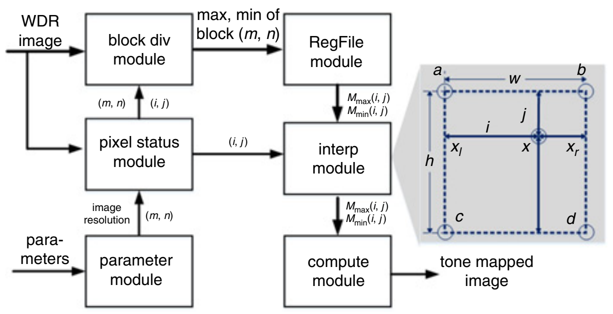

Fig. 2 *Implemented hardware architecture*

To reduce hardware resources, the log computation in the *compute module* is approximated with Taylor expansion. However, Taylor expansion only converges for natural logarithm when the approximated number is between −1 and 1, thus we first separate the pixel value into a fractional part which is smaller than 1 and a multiplicative factor which is an order of 2

$$\log(x) = \log\left(\frac{x}{2^N} \times 2^N\right) = \log\left(\frac{x}{2^N}\right) + \log(2^N), \quad 2^N \geq x \tag{2}$$

The number N is found as the location of the MSB of x. The fractional part then can be approximated by second-order Taylor expansion

$$\log\left(\frac{x}{2^N}\right) = \frac{x}{2^N} - 1 - \left(\frac{x}{2^N} - 1\right)^2 \tag{3}$$

If we change the base of natural logarithm, we get the following for the multiplicative factor part:

$$\log(2^N) = \frac{\log_2(2^N)}{\log(2)} = \frac{N}{\log(2)} \approx N \times \left(1 + \frac{1}{2} - \frac{1}{16}\right) \tag{4}$$

The approximation of $1/\log(2)$ is in the form of power of 2 numbers, which enables to use shift operations instead of a division of computing $\log(2^N)$.

The bilinear interpolation is realised by the *interp module*. It conducts three 1D interpolations as shown in the shaded area of Fig. 2. If the block size in Fig. 1 is $h \times w$ and the four corners are extracted maximum or minimum values, we need to interpolate the values between the four corners to form the $M_{\max}$ and $M_{\min}$ matrices. Suppose the point to be interpolated is located in (i, j) of the block. Then two intermediate values x_l and x_r are first interpolated using the following equations:

$$x_l = a + \frac{j \times (c - a)}{h}, \quad x_r = b + \frac{j \times (d - b)}{h} \tag{5}$$

The final x values are further interpolated with x_l and x_r using a similar method. In our implementation, we choose block size w and h as a power of 2 such as 32, 64, 128 so that the division is done by simple shift operations.

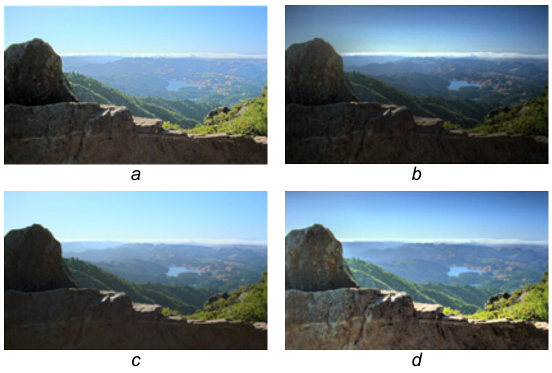

Fig. 3 *Tone mapped images using different algorithms*

a Result of [5]
b Result of [6]
c Result of [7]
d Result of the proposed algorithm

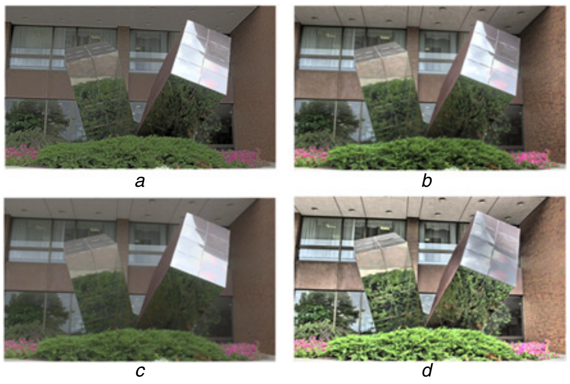

Fig. 4 *Tone mapped images using different algorithms*

a Result of [5]
b Result of [6]
c Result of [7]
d Result of the proposed algorithm

Experimental results and comparisons: The proposed hardware architecture for the tone mapping algorithm was modelled in Verilog HDL and synthesised using Altera Quartus II 13.1 toolset. An Altera Cyclone III FPGA (EP3C120F780) development kit was our targeted platform. Figs. 3 and 4 show images tone mapped with different algorithms. As we can notice, the tone mapped image of our proposed algorithm is globally brighter and many details are visible when compared with other works. To assess the image quality of our tone mapped images, we have used the tone mapped image quality index (TMQI) [9]. The TMQI combines a multi-scale structural fidelity measure and a measure of image naturalness and provides a single quality score of an entire image. The TMQI measurements of images in Figs. 3 and 4 are shown in Table 1. The high quality score of our algorithm suggests that it can produce images that satisfy good quality criteria. Our hardware implementation was designed for minimising hardware resources and real-time processing. The synthesised working clock frequency of our hardware implementation is 100 MHz. Since our algorithm works in pixel-by-pixel fashion, this means that our implementation can process 100 Mega-pixels in 1 ms. The logic utilisation of our hardware implementation on a cyclone III FPGA (119 K logic elements) is only 11%. To evaluate the hardware implementation efficiency, we compare our work with four other similar works. The results are shown in Table 2. Hassan and Carletta [7] reported an FPGA implementation of a local tone mapping algorithm. This design can achieve a processing speed of 60 frames per second (FPS); however, the implementation requires a large number of hardware resources. Vytla *et al.* [6] have implemented the gradient domain WDR compression algorithm; their design requires a fewer logic resource, but it employs 88 DSP blocks for the computation. Recently, Ambalathankandy *et al.* [5] have implemented a global-local tone mapping method; this work also requires more logical and memory resources than our implementation. The implementation of Shahnovich *et al.* [4] use less logic resource, but it can only carry out simple logarithmic compression.

Table 1: Quantitative measure of Figs. 3 and 4

Algorithms	Image1	Image2
[5]	0.9228	0.8058
[6]	0.8479	0.8576
[7]	0.8348	0.7620
Ours	0.9351	0.9182

Table 2: Comparison with other tone mapping hardware implementations

Works	Image size	FPS	Logic elements	Memory (bits)
Hassan and Carletta [7]	1024 × 768	60	34,806	3,153,048
Vytla *et al.* [6]	1 Megapixel	100	9019 + 88 DSP	307,200
Ambalathankandy *et al.* [5]	1024 × 768	126	93,989	87,176
Shahnovich *et al.* [4]	1024 × 768	126	4020	270,336
This work	1024 × 768	126	13,216	77,408

Conclusion: This Letter introduces a novel WDR image tone mapping algorithm with a hardware implementation. The algorithm can compress WDR images in a pixel-by-pixel fashion with different compression levels based on local pixel statistics. The hardware implementation is resource-efficient and can achieve real-time processing speed. Some experimental results obtained have shown the good performance of the proposed algorithm, and comparisons with some other hardware works have shown that our implementation is hardware-efficient.

Acknowledgment: This work was supported by the Alberta Innovates Technology Futures (AITF) and Natural Sciences and Engineering Research Council of Canada (NSERC).

© The Institution of Engineering and Technology 2018
Submitted: *29 August 2017* E-first: *6 April 2018*
doi: 10.1049/el.2017.3227
One or more of the Figures in this Letter are available in colour online.

J. Yang, A. Hore and O. Yadid-Pecht (*Integrated Intelligent Sensors Laboratory (I2Sense) Lab, Schulich School of Engineering, University of Calgary, 2500 University Drive NW Calgary, AB T2N 1N4, Canada*)

✉ E-mail: orly.yadid-pecht@ucalgary.ca

A. Hore: Also with IT GLORY Inc.

References

1 Debevec, P.E., and Malik, J.: 'Recovering high dynamic range radiance maps from photographs'. ACM SIGGRAPH 2008 Classes, Los Angeles, CA, USA, August 2008, p. 31
2 Yadid-Pecht, O., and Fossum, E.R.: 'Wide intrascene dynamic range cmos aps using dual sampling', *Trans. Electron Devices*, 1997, **44**, (10), pp. 1721–1723
3 Spivak, A., Belenky, A., Fish, A., *et al.*: 'Wide-dynamic-range cmos image sensors comparative performance analysis', *Trans. Electron Devices*, 2009, **56**, (11), pp. 2446–2461
4 Shahnovich, U., Hore, A., and Yadid-Pecht, O.: 'Hardware implementation of a real-time tone mapping algorithm based on a mantissa-exponent representation'. 2016 IEEE Int. Symp. on Circuits and Systems (ISCAS), Montreal, Canada, May 2016, pp. 2210–2213
5 Ambalathankandy, P., Horé, A., and Yadid-Pecht, O.: 'An FPGA implementation of a tone mapping algorithm with a halo-reducing filter', *J. Real-Time Image Process.*, 2016, pp. 1–17, DOI: 10.1007/s11554-016-0635-6
6 Vytla, L., Hassan, F., and Carletta, J.E.: 'A real-time implementation of gradient domain high dynamic range compression using a local Poisson solver', *J. Real-Time Image Process.*, 2013, **8**, (2), pp. 153–167
7 Hassan, F., and Carletta, J.E.: 'An FPGA-based architecture for a local tone-mapping operator', *J. Real-Time Image Process.*, 2007, **2**, (4), pp. 293–308
8 Ofili, C., Glozman, S., and Yadid-Pecht, O.: 'Hardware implementation of an automatic rendering tone mapping algorithm for a wide dynamic range display', *J. Low Power Electron. Appl.*, 2013, **3**, (4), pp. 337–367
9 Yeganeh, H., and Wang, Z.: 'Objective quality assessment of tone-mapped images', *Trans. Image Process.*, 2013, **22**, (2), pp. 657–667