A HIGH THROUGHPUT ENCODER FOR HIGH DYNAMIC RANGE IMAGES

Firas Hassan and *Joan Carletta*

Department of Electrical and Computer Engineering
The University of Akron
Akron, Ohio USA

ABSTRACT

An encoder for high dynamic range (HDR) images is proposed that is compatible with the JPEG2000 compression engine and can provide three different versions of the image to the user at the receiver end depending on his needs. The first version is a gray scale HDR image. The second is a local tone-mapped version of the color HDR image suitable for display on conventional devices. By combining data from the first two versions, the original color HDR image can be reconstructed. The proposed HDR encoder was implemented on a field programmable gate array, and has high enough throughput to support a system processing 1024×768 images at a rate of 60 frames per second.

Index Terms— high dynamic range, field programmable gate array, JPEG2000

1. INTRODUCTION

The sheer volume of data required to represent a HDR image is a bottleneck limiting the use of HDR images. Current research is investigating encoding of HDR images, removing redundant or perceptually unimportant information, in ways suitable to interface with lossy compression engines such as JPEG2000. It is highly desirable to be able to do this encoding in real-time, and embedded within a camera. However, published HDR encoders have been implemented mostly on workstations and graphics cards without regard to resource or time constraints. The work described in this paper develops a HDR image encoder that operates in real-time on an embedded system with high throughput and low cost.

The paper is organized as follows. Section 2 describes related work, while Section 3 describes the proposed HDR image encoder in detail. Section 4 shows the results obtained from synthesizing and verifying the design. Section 5 draws conclusions.

2. BACKGROUND

The pioneering compression algorithms for HDR images, such as RGBE [1] and LogLuv [2], used lossless methods and were not efficient. An early HDR image encoder compatible with lossy compression techniques was presented in [3]. Mantiuk et al. transform the HDR data to the Luv space (similar to logLuv) and quantize the color components linearly. Then they use a global tone mapping operator to perceptually quantize the luminance component. The resulting image is sent to an MPEG video compressor. In [4], Mantiuk et al. introduced a compact reconstruction function that is used to decompose the HDR image into a standard low dynamic range (LDR) stream and a residual stream that together can be used to reconstruct the original image. A similar approach was suggested by Ward et al. in [5]. Both [4] and [5] targeted MPEG video compression; because the MPEG protocol does not allow for additional channels, the residual image must be sent in a subband. As a result, the methods concentrated on decorrelating the residual from the LDR image and coding the data so that it will fit in a subband.

Use of JPEG2000 for high dynamic range applications has two advantages: JPEG2000 allows for more precision than MPEG, and it allows for additional channels that can be used to send luminance data. Xu et al. develop in [6] an encoder for HDR images that can be used as a front end for a JPEG2000 compression engine. They send a 16-bit logarithm of every color component to the engine. However, they are not able to downsample the chrominance red and chrominance blue inside the compression engine (as is usually done in JPEG2000 to decrease the size of the compressed image) because in the log domain the chrominance spaces bear significant information. Finally, Munkberg et al. and Roimela et al. suggest textural compression of HDR images in [7] and [8], respectively; both methods were implemented on graphics cards.

Our encoder uses a local tone mapping operator that is a variant of the operator suggested by Reinhard et al. in [9]. This operator has been implemented on graphics cards by Goodnight et al. in [10] and Krawczyk et al. in [11]. The implementation in [11] processes 1024×768 images at a rate of 14 frames per second; this is not considered real-time. Our work, which implements a hardware-friendly variant of the Reinhard operator inside the encoder, has been implemented on a field programmable gate array and has high enough throughput for real-time applications.

3. HDR IMAGE ENCODER

The block diagram of our HDR image encoder is shown in Figure 1 and is made up of five subblocks. At the input, the high dynamic range image is represented as an *RGB* triplet, where each element is a 32-bit fixed-point value, with 16 bits of integer and 16 bits of fraction. The *luminance calculation subblock* scales the *RGB* triplet by 2^{16} in order to transform the numbers to integer and

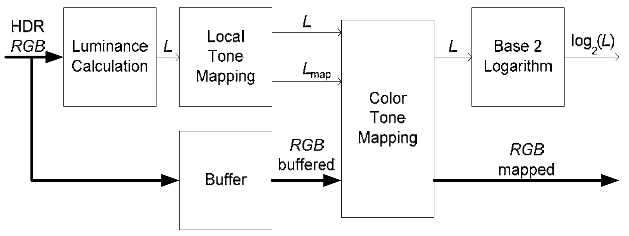


Figure 1. The proposed HDR image encoder.

calculates the luminance component of a pixel in fixed-point mathematics as:

$$L = 0.299 \times R + 0.587 \times G + 0.114 \times B\,. \tag{1}$$

The *local tone mapping subblock* then uses a Reinhard-like operator to get the tone mapped version of the luminance component L_{map}. This subblock is described in more detail in a subsequent subsection.

The *RGB* triplet is buffered in *external memory* during the local tone mapping of the luminance component. The latency of the local tone mapping subblock is 28 rows of the input image. Hence, for 1024x768 images three external memories, each of size 28×1024×32 bits, are needed.

Each buffered *RGB* triplet is fed to the *color tone mapping subblock* simultaneously with its luminance before and after local tone mapping (L and L_{map}, respectively). Tone mapped color components are computed using the approach developed by Hunt in [12] and adopted in [11], which considers the sensitivity of the rods in the human vision system and the blue shift of the subjective hue of colors for night scenes. The computation is:

$$R_{\text{map}} = \frac{(R + 2752) \times L_{\text{map}}}{L + 2621} \tag{2}$$

$$G_{\text{map}} = \frac{(G + 2542) \times L_{\text{map}}}{L + 2621} \tag{3}$$

$$B_{\text{map}} = \frac{(B + 3328) \times L_{\text{map}}}{L + 2621}\,. \tag{4}$$

Before entering this subblock L_{map} is saturated to a maximum value of 255. The arithmetic operations are done in floating-point; division is simplified by using an iteration based on the Newton-Raphson method to find the reciprocal, after retrieving an initial guess for the iteration from a look-up table that is indexed on a limited number of bits of the mantissa. The procedure is such that one look-up and one iteration are enough to achieve the required precision. The mapped *RGB* triplet is converted to fixed-point at the output of the subblock. To convert to fixed point, the mantissa of every mapped component is sent to a barrel shifter controlled by its exponent. Each mapped color component is represented by 10 bits of integer.

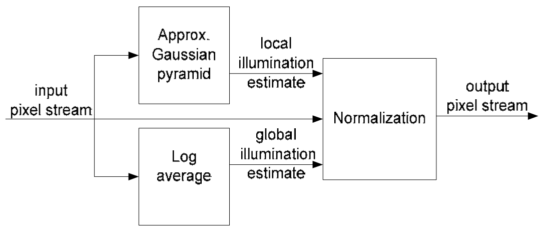


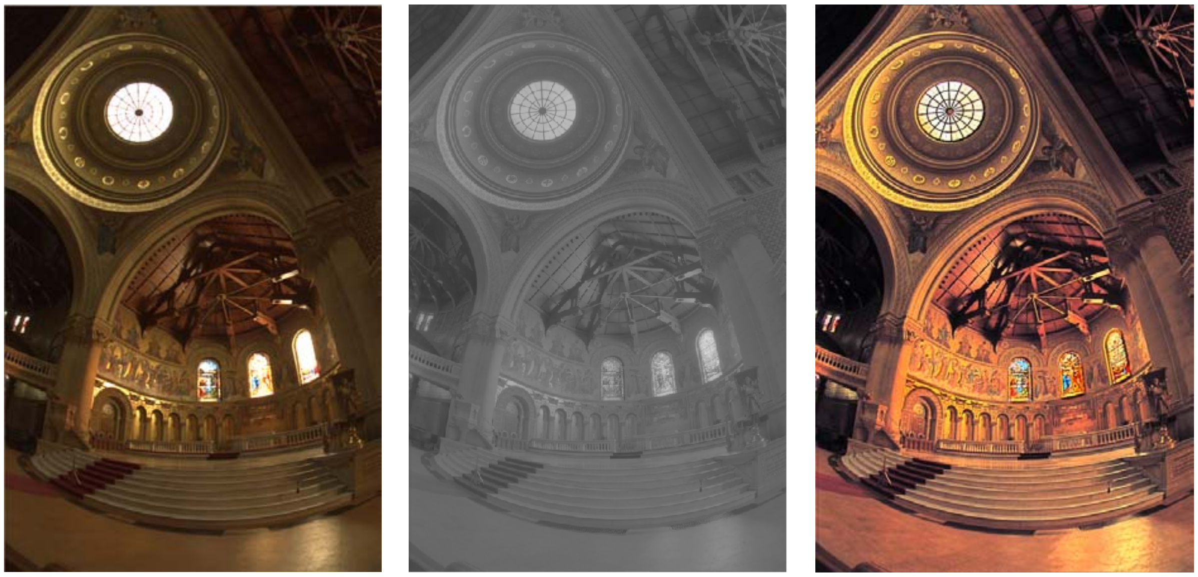
Figure 2. The block diagram of the embedded version of the Reinhard operator.

Finally, the *base-2 logarithm subblock* calculates the log-luminance $log_2(L)$ of each pixel. An estimate of the logarithm can be found based on the number of leading zeros in L; the integer part of the base-2 logarithm is determined solely by the position of the most significant '1' in L, while the fraction part is determined by the remaining bits in L via lookup table. The size of the lookup table and the error in the estimation depends on how many bits we choose to use as the address and how many fraction bits we choose to store in the table; for the work described here, we used an address size of eleven and word size of eleven; the log-luminance is represented as 16-bit fixed-point, with 5 bits of integer and 11 bits of fraction.

3.1. The local tone mapping subblock

Local tone mapping of the luminance components of the image is done via a modified Reinhard operator. We developed hardware that approximates the original method well, but is simple enough for real-time embedded processing. Full details, including a study that compares the original operator to our approximation, are given in [13]. A block diagram of our hardware-friendly variant of the Reinhard operator is given in Figure 2. The input to this subsystem is the luminance component L of every pixel. In the Reinhard operator, a nine-scale Gaussian pyramid is used to estimate the illumination local to a pixel; our hardware-friendly variant has an *approximate Gaussian pyramid* that replaces the nine scales with four scales (of size 8x8, 14x14, 28x28 and 56x56 pixels, respectively) and uses rising and falling geometric series, which can be computed with simple accumulator structures, in place of the exponential rise and decay of a true Gaussian. This approximation is a key to being able to do the local tone mapping in real time on embedded low cost hardware.

A global *logarithmic average* of the luminance is also computed to provide an estimate of the global illumination. This is done by computing the sum of the base-2 logarithms of all the pixels in the image, using the already described technique to make the logarithm calculation hardware-friendly. The global illumination is estimated as 2 raised to this sum, and corresponds to the average of all the pixels. Because this global average requires access to the entire image, it can not be computed until a frame has been received in its entirety. To achieve real-time performance, the log average luminance of the previous frame can be used to normalize the pixels of the current video frame. This choice is justifiable for because global illumination generally does not change dramatically between two consecutive video frames.



(a) original HDR image (color, containing high dynamic range and processed using software from [1] to display) (b) log-luminance (containing information needed to reconstruct a gray scale high dynamic range image) (c) mapped RGB (color, with low dynamic range appropriate for display)

Figure 3. The Memorial HDR image from [1], and the output of the proposed HDR encoder.

Finally, the *normalization subblock* divides the input luminance component by a weighted sum of the local and global illumination estimates, using:

$$L_{map} = \frac{L}{L_{ave\_local} + a * L_{ave\_global}} \qquad (5)$$

where a is a weighting factor, $L_{\text{ave\_local}}$ is the local estimate of illumination around the pixel, $L_{\text{ave\_global}}$ is the global illumination estimate and L_{map} is the mapped luminance represented in floating point.

4. IMPLEMENTATION AND RESULTS

The output of our HDR image encoder can be fed to a four-channel JPEG2000 lossy compression engine. The three usual JPEG2000 channels can be used to compress the mapped RGB triplet. An additional channel can be allocated to compress the log-luminance component. At the decoder side, the user may choose to receive only the log-luminance component; this component, which is shown for the Memorial image from [1] in Figure 3(b), contains high dynamic range information, and can be used to reconstruct a high dynamic range gray scale version of the image (by applying an inverse log transform).

Alternatively, the user can choose to receive the mapped RGB channels only. These channels provide a low dynamic range color image appropriate for immediate display on a conventional display device. The mapped RGB version of our example image is shown in Figure 3(c).

If desired, the user can also choose to reconstruct the original HDR image. In this case, the user needs both the log-luminance and mapped RGB channels. The original luminance component can be reconstructed by taking the inverse base-2 logarithm of the log-luminance channel, and the luminance fed into an implementation of the proposed Reinhard-like local tone mapper on the user side to reproduce the mapped luminance component L_{map}. Finally, L, L_{map} and a buffered version of the mapped RGB triplet can be fed to an inverse color tone mapping subblock to reconstruct the original HDR RGB triplet.

The HDR image encoder was implemented using an Altera FPGA from the Stratix II family. The architecture was sized in order to accommodate high resolution images of high dynamic range with 768×1024 pixels and 32 bits per pixel. The architecture was described in VHDL, and synthesized for the Stratix device using Altera's Quartus II v5.0 toolset; functionality of the system was verified by simulating the processing of test images using Modelsim. Table 1 summarizes the synthesis report from Quartus. The simplicity of hardware is reflected in the clock speed achieved, and in the low utilization of look-up tables.

Typically, a video frame has horizontal blanking periods of 64 pixels, and a vertical blanking period of 32 rows. Given that we would like to achieve a video frame rate of 60 frames per second, and that there are (1024+64)*(768+32) or 870,400 pixels in the frame when we include the blanking periods, we need to be able to process 60*870,400 = 52.24 megapixels per second. Our architecture, which has a maximum operating frequency of 68.03 MHz, can accommodate this by taking in one pixel per clock.

To verify the design, we implemented a fixed-point Matlab simulation that matches the hardware exactly. A HDR video

Table 1. Summary of hardware synthesis report.

family	Stratix II
device	EP2SQ30F780C4
size of memory	3.23 Mbits
no. of flip-flops	17,151
no. of ALUTs	21,532
max operating frequency	68.03 MHz

Table 2. Peak signal-to-noise ratio of reconstructed HDR images relative to the original images.

	$PSNR_R$	$PSNR_G$	$PSNR_B$
Nave	98	97	88
Memorial	90	88	84
groveC	86	86	82
groveD	75	73	73
Rosette	86	86	65
Vinesunset	72	61	56

decoder was also implemented in Matlab. Using a set of images from the Debevec library we checked the PSNR of the reconstructed RGB triplet at the output of the decoder relative to the 32-bit RGB triplet at the input of the system. Table 2 shows the PSNR for a set of images from the Debevec library. Note that the numbers shown indicate only the degree to which information is lost in the proposed encoding/decoding process; they do not include losses in the JPEG2000 compression itself. Note also that the size of the noise is comparable for all the images; however, the PSNR values vary because PSNR depends on the dynamic range of the particular test image. The dynamic range of the images in Table 2 varies from six bits of integer for Vinesunset to twelve bits of integer for Nave. The PSNR values show that the images can be successfully reconstructed.

5. CONCLUSIONS

This paper presents a novel HDR color image encoder. The encoder uses a separate channel for a 16-bit log-luminance component, and three channels for a 30-bit local tone mapped RGB triplet. The output of the encoder can be fed to a JPEG2000 compression engine where the log-luminance is compressed in a separate channel, while the mapped color channels are treated as a conventional low dynamic range color image. We believe that our encoding method will give more freedom to the user than previous methods in [4] and [5] that use a residual ratio image. The residual ratio image is not useful to the user in itself, while the log-luminance component here can be used to reconstruct a HDR gray scale image. By using JPEG2000, we also can use more precision than approaches based on MPEG; this should result in better reconstruction. While we have not yet studied overall efficiency of our method when coupled with JPEG2000, we anticipate that JPEG2000 will be able to compress our encoded images more effectively than those in [6], because our mapped RGB triplets can be treated by JPEG2000 as a regular low dynamic range image, and therefore downsampling of the chrominance components can be enabled. Furthermore, our encoder has been implemented in real-time on low-cost hardware suitable for embedding in applications such as cameras.

REFERENCES

[1] P. E. Debevec and J. Malik, “Recovering high dynamic range radiance maps from photographs,” *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 369-378, 1997.

[2] G. W. Larson, “Overcoming gamut and dynamic range limitations in digital images,” *Proceedings of the 6th Color Imaging Conference*, pp. 214-219, 1998.

[3] R. Mantiuk, G. Krawczyk, K. Myszkowski, and H.-P. Seidel, “Perception-motivated high dynamic range video encoding,” *ACM Transaction on Graphics*, vol. 23, no. 3, pp. 733-741, 2004.

[4] R. Mantiuk, A. Efremov, K. Myszkowski, and H.-P. Seidel, “Backward compatible high dynamic range MPEG video compression,” *SIGGRAPH 2006*, pp. 713-723, 2006.

[5] G. Ward and M. Simmons, “Subband encoding of high dynamic range imagery,” *Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization,* pp. 83-90, 2004.

[6] R. Xu, S. N. Pattanaik and C. E. Hughes, “High-dynamic-range still-image encoding in JPEG 2000,” *IEEE Computer Graphics and Applications*, pp. 57-64, 2005.

[7] J. Munkberg, P. Clarberg, J. Hasslegren and T. Akenine-Moller, “High dynamic range texture compression for graphics hardware,” *SIGGRAPH 2006*, pp. 698-706, 2006.

[8] K. Roimela, T. Aarnio and J. Itaranta, “High dynamic range texture compression,” *SIGGRAPH 2006*, pp.707-712, 2006.

[9] E. Reinhard, M. Stark, P. Shirley and J. Ferwerda, “Photographic tone reproduction for digital images,” *Proceedings of the* 29th *Annual Conference on Computer Graphics and Interactive Techniques*, pp. 267 - 276, 2002.

[10] N. Goodnight, R. Wang, C. Woolley, and G. Humphreys, “Interactive time-dependent tone mapping using programmable graphics hardware,” *Proceedings of the 14th Eurographics Workshop on Rendering,* vol. 44, pp. 26-37, 2003.

[11] G. Krawczyk, K. Myszkowski and H-P. Seidel, “Perceptual effects in real-time tone mapping,” *Proceedings of the 21st Spring Conference on Computer Graphics*, pp. 195-202, 2005.

[12] R. Hunt, *The Reproduction of Colour in Photography, Printing and Television: 5th Edition*, Fountain Press, 1995.

[13] F. Hassan, J. Carletta, “A real-time FPGA-based architecture for a Reinhard-like tone mapping operator,” *Proceedings of Graphics Hardware*, San Diego, California, Aug. 2007.