REAL TIME AUTOMATED TONE MAPPING SYSTEM FOR HDR VIDEO

Chris Kiser

University of New Mexico
ECE Dept. & Contrast Optical

Erik Reinhard

University of Bristol
Dept. of Computer Science

Mike Tocci, Nora Tocci

Contrast Optical
Sandia Park, NM

ABSTRACT

Tone Mapping of HDR Images has been studied extensively since the introduction of digital HDR capture methods. However, until recently HDR video has not been realized in a viable form as that given by Tocci et al.[1], which will lead to readily available HDR video cameras. Because they capture video with much broader dynamic range than can currently be displayed, these cameras present a unique challenge. In order to maintain backward-compatibility with legacy broadcast, recording, and display equipment, HDR video cameras need to provide a real-time tonemapped LDR video stream without the benefit of post-processing steps. The purpose of this research is to present a complete TMO solution that can be implemented in real time hardware to yield a high-quality automated LDR video stream suitable for direct use by today's recording, broadcast, and display equipment.

Index Terms— HDR, TMO, video

1. INTRODUCTION

In this work we present a complete real time video tone mapping system, which addresses the challenges of tone mapping video data while still fulfilling the requirements that the solution be implementable in real-time in an HDR video camera.

Simply applying the well-tested photographic operator [2] using automatic parameters settings [3] theoretically should be sufficient to tone map HDR video data. However, through extensive research with a wide variety of real-world HDR video data captured using the camera system described by Tocci et al. [1], two major challenges have been identified as serious obstacles to such a simple solution: sub-optimal utilization of the output (display) dynamic range, and flickering of a nearly-static-scene's overall brightness level.

Ideally, the output LDR video stream would utilize as much of the display's LDR bit depth as possible, in order to reduce quantization effects. But real-world camera data anomalies can cause the TMO to limit its output dynamic range so that it sometimes underutilizes the display's dynamic range and potentially changes the contrast distribution of the output image. This can cause implementations of the otherwise robust TMOs to look sub-optimal.

The flickering intensity problem is caused by internal TMO parameters, which can change significantly with very minor changes in the imaged scene. Although these changing TMO parameters might be insignificant when tonemapping a single still image, the implication for video is that the brightness of each tone mapped frame can change quite noticeably from one video frame to the next, leading to flicker in the final video stream.

Our tone mapping system seeks to resolve these two challenges while building on the photographic operator [2], because its parameter settings can be automated [3]. Our complete system comprises a method to clamp the black and white levels of the HDR image prior to tone mapping, the photographic operator [2], [3], and a *leaky integrator* [4] to give internal TMO parameters a sort of "temporal memory" to reduce intensity flicker.

2. PREVIOUS WORK

Tone reproduction for HDR images is a well-researched field [5]. Tone reproduction for video, on the other hand, is still a largely unexplored area due to the absence of viable sources of HDR video, although this is beginning to change [1]. HDR video cameras require fully automated real-time tonemapped output both for backward compatibility and for driving the view finder. To our knowledge no single operator can currently achieve this, as most operators are too complex to implement in hardware, require manual parameter tuning, or create temporal artifacts.

In particular, local operators [6], [7], [8] require too many passes over the image to compute local adaptation, may require optimization [9], or might require significant hardware resources to implement in consumer level products [10]. Simple global operators apply a compressive function to each pixel identically, and as such could be implemented in real-time, but tend to require manual parameter tuning. Only few global operators are fully automatic [2], Ward [11], although one of them requires some iterative processing with associated penalty in terms of hardware implementation [11]. This leaves the photographic operator [3] with automatic parameter tuning [2] as a viable candidate for video processing, as it is fast and produces plausible results, as determined in psychophysical experiments [12]. However, as with other operators, it leads to temporal artifacts if applied to video, a problem solved in the current paper.

3. OUR TONE MAPPING SYSTEM

3.1. Clamping black and white levels

The function of a TMO is to re-map a range of input (HDR) world luminance $L_w(x,y)$ values to a smaller range of output (LDR) display luminance values $L_d(x,y)$, which is required to maintain backward-compatibility with legacy broadcast, recording, and display equipment. Because of the limited dynamic range of this equipment, it may also be desirable to utilize as much of this limited display range as possible to maximize tonal accuracy of the displayed image. The world luminance $L_w(x,y)$ is computed from RGB triplets using $L_w(x,y) = 0.27R(x,y) + 0.67G(x,y) + 0.06B(x,y)$.

When a single HDR image is tone mapped and displayed, the image often must be manually adjusted to compensate for any ill-behaved data such as pixels with digital number (DN) values lower than the scene's overall black level (cold pixels) and pixels with

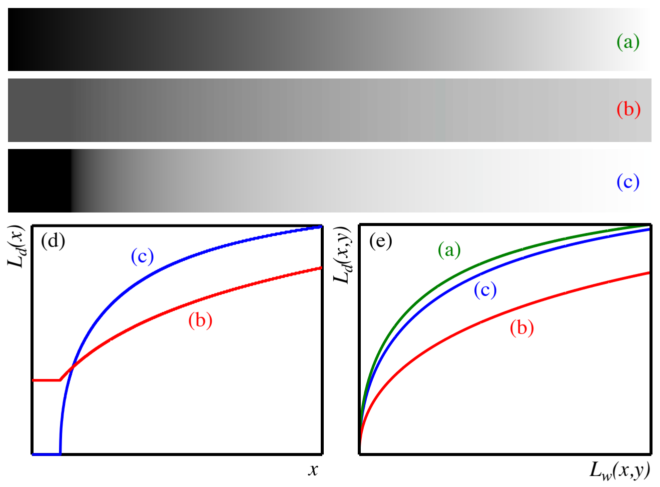


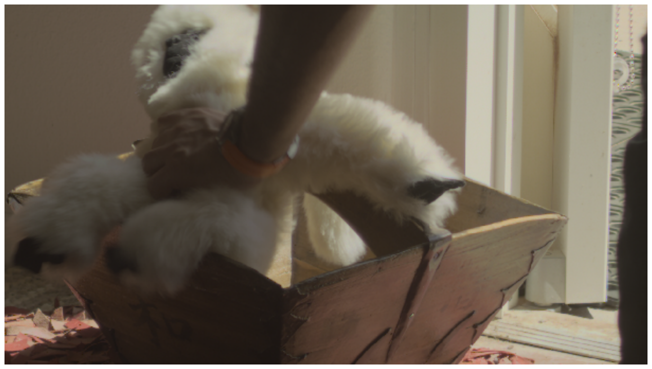
Fig. 1. (a) Example HDR Image with 11 stops of dynamic range tone mapped with Reinhard et al. (b) HDR image with black level shift and single hot pixel tone mapped with Reinhard et al. (c) HDR image with black level shift and single hot pixel pre-scaled then tone mapped with Reinhard et al. (d) Cross-sectional view of tone mapped images (b) in red and (c) in blue. (e) $L_w(x, y)$ to $L_d(x, y)$ mapping curves for tone mapped image (a) in blue, (b) in red, and (c) in green

anomalously high DN values (hot pixels). Cold pixels can cause the TMO to set the tone mapped image's black level too high making black portions of the scene gray. Similarly, hot pixels can cause bright portions of the image to appear dimmer. Such ill-behaved data types are common in real-world digital camera images and proper treatment is a necessity for a well performing automated video tone mapping system.

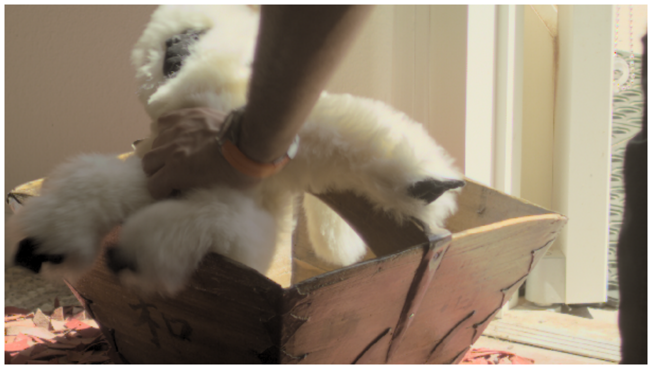
A common technique to improve stability is to remove outliers from the tonemapped imagery by clamping, usually around 1% of the brightest and 1% of the dimmest pixels. Excluding these pixels reduces the effects of ill-behaved data compared to no exclusion. However, simply excluding the data can still lead to reduced dynamic range utilization. Take for example the test image in Fig 1a, which has been tone mapped according to [2]. This image has a dynamic range of 11 stops starting at zero on the left and linearly increasing to 2048 on the right. The blue curve in Fig 1e shows the world luminance $L_w(x, y)$ to display luminance $L_d(x, y)$ mapping of the TMO where 2048 input values are mapped to 256 output values.

We now give the test image an offset of 200 counts for the first 200 columns and a simulated hot pixel at location 1500. Fig 1b shows the new tone mapped test image. The mapping of $L_w(x, y)$ to $L_d(x, y)$ for the new test image is shown by the red curve in Fig 1e. A cross-sectional view of the tone mapped test image of Fig 1b is shown by the red curve in Fig 1d. This cross-sectional view shows the tone mapped image of Fig 1b now has a black level offset and contains less contrast from blacks to whites. The tone mapped image in Fig 1b is using only around 130 counts of 256 possible output counts thus not using half of the dynamic range.

Our tone mapping system seeks to build on the results of display dynamic range utilization in the TMO by clamping white and black levels and also black level shifting the image using histogram percentiles as a pre-clamping step. The method computes the histogram H of the HDR image then the cumulative sum C_S of the histogram by $C_S(n) = \sum_{k=1}^{n} H$. A percentage of pixels are allocated to the output dynamic range as β. Experience working with a wide variety of HDR video data shows a good value of β is around 0.999. An upper and lower boundary is then found by $B_{upper} = \beta \max C_S$ and $B_{lower} = (1 - \beta) \max C_S$ respectively. These values are in numbers of pixels on the y-axis of the cumulative sum graph. To convert these to values we simply find the nearest bin on the x-axis that corresponds to the required boundary value on the y-axis.



(a)

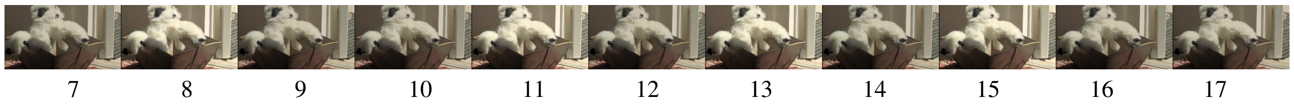


(b)

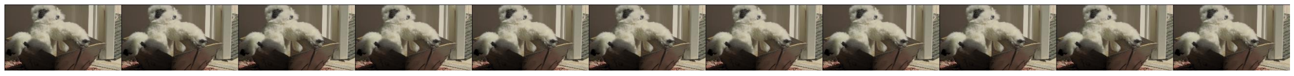
Fig. 2. Standard tone mapped monkey image in (a). Tone mapped monkey image using pre-scaling in (b)

Fig 1c shows the resultant tone mapped image when the clamping step has been applied and the cross-sectional view is shown by the blue curve in Fig 1d. The 200 count offset is positioned back to black level zero and the hot pixel value does not affect the white levels. In addition, Fig 1e shows a key feature of our application of the pre-scale and black level shift, that is to retain the basic shape of the $L_w(x, y)$ to $L_d(x, y)$ mapping curve. The red mapping curve of Fig 1e shows dark value pixels receive less contrast enhancement because of the shallower slope in the lower DN region. Lighter pixels gain more contrast enhancement from the higher slope in the higher DN region. Applying a post-clamping step would have retained this distorted shape; however, the green mapping curve from pre-clamping in Fig 1e is similar in shape to the original blue mapping curve and thus the distribution of the output DNs is also similar.

Another tone mapped HDR image is shown in Fig 2a, which is a single frame of an HDR video sequence that was captured and merged using the apparatus and method of Tocci [1]. Fig 2b shows the same Monkey test image tone mapped after using the



(a)



(b)

Fig. 3. Frames 7 - 17 of tone mapped movie. Frames 8,11,13, and 15 from (a) cause flicker in resultant video. (b) shows results of reduce flicker

pre-clamping step. The image in Fig 2b shows brighter whites and darker blacks, and effectively uses more of the available output dynamic range than the non-pre-clamped TMO image in Fig 2a.

3.2. Tone Mapping

Our tone reproduction operator of choice is the photographic operator, as its parameters can be automated. However, the parameter estimation requires the minimum and maximum luminance of the image $L_{\min}$ and $L_{\max}$, which are by definition outliers. With L_{av} the geometric mean of the image, the key of the scene a is estimated as:

$$a = 0.18 * 2^{2(B-A)/(A+B)} \tag{1a}$$
$$A = L_{\max} - L_{\text{av}} \tag{1b}$$
$$B = L_{\text{av}} - L_{\min} \tag{1c}$$

Photographic tone reproduction then maps the geometric mean of the image to the estimated key value, $L(x,y) = a\, L(x,y)/L_{\text{av}}$ before applying a non-linear compression:

$$L_d(x,y) = \frac{L(x,y)\left(1 + \frac{L(x,y)}{L^2_{\text{white}}}\right)}{1 + L(x,y)} \tag{2}$$

where L_{white} is a user parameter specifying the smallest input value that should be mapped to white.

3.3. HDR Video Flicker Removal

If applied to video, using outliers leads to significant changes in parameter settings from frame to frame, and is therefore the source of flicker in tonemapped video. An example filmstrip of a tone mapped video sequence is shown in figure Fig 3. Fig 3a shows frames 7 - 17 using the unmodified TMO. Frames 8,11,13, and 15 have a noticeable increase of average pixel value which causes visible flicker in the video sequence. The red curve in Fig 4a the upper limit A computed by the TMO for the same frames of 7 - 17. Spiked values correspond to the frames that appear lighter, namely frames 8, 11, 13, and 15.

Our tone mapping system seeks to reduce flicker in tone mapped HDR video by using a leaky integrator applied to the parameters of Eq. 1a through Eq. 1c. We use a normalized version of the leaky integrator as a filter to reduce the rapid frame to frame changes in the parameters of the TMO thus giving the TMO algorithm temporal memory. If we applied a leaky integrator to a only, then to eliminate the flicker, α_a would need to be set such that the TMO could not quickly adapt to scene changes. The update equations to compute the filtered A_n, B_n, and a_n for the n^{th} frames are given in Eq. 3a through Eq. 3c

$$A_n = (1-\alpha_A)A_{(n-1)} + (\alpha_A)A \tag{3a}$$
$$B_n = (1-\alpha_B)B_{(n-1)} + (\alpha_B)B \tag{3b}$$
$$a_n = (1-\alpha_a)a_{(n-1)} + (\alpha_a)a \tag{3c}$$

The parameters α_A, α_B, and α_a are time constants $\in [0,1]$ and all are set to 0.98 for our example. They also serve as normalizing factors since the coefficients sum to the value one, i.e. $(1-\alpha_A) + (\alpha_A) = 1$. Eq. 1a through Eq. 1c only need to store a single value for each parameter. Application of the leaky integrator to internal TMO parameters results in reduced flicker shown in the frames of Fig 3b. The blue curve in Fig 4a shows the filtered value A_n for frames 7 - 17 plotted along with the red curve of the unfiltered A value. Fig 4b shows the entire sequence of 250 values of A in red and A_n in blue for our HDR movie sequence.

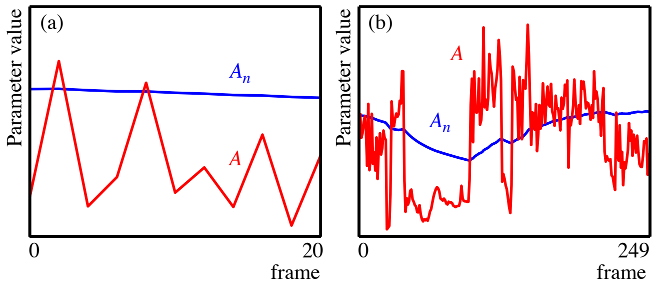


Fig. 4. Internal upper limit A parameter for frames 7 - 17 of HDR video sequence. Red lines are unmodified A, Blue lines are A_n. Spiked values in A for frames 8,11,13, and 15 that lead to flicker in the video sequence.

3.4. FPGA Real Time Implementation

Our pre-clamping method requires the histogram to be computed using $L_w(x,y)$, the cumulative sum, and the boundary values then adjusts the image according to these boundary values. A histogram bins the pixel values in an image into equally spaced containers and returns the number of elements in each container as a single dimensional vector. The world luminance value $L_w(x,y)$ for each pixel (x,y) in the image is the address of the bin to increment in the histogram $H(L_w(x,y))$. In hardware, histograms are computed using a single RAM (Random Access Memory) and a feedback adder circuit. The HDR video captured by the camera system presented in [1] produces 18 bit data, which would require 2^{18} histogram bins in the RAM. This is not feasible in FPGA internal memory. Our method uses a histogram storage solution similar in idea as *.hdr* files. We force every DN value to have the same number of bits of accuracy such that all DNs can be stored using 8 bits of mantissa and a base 2 exponent. Using this storage method, Eq. 4b computes the histogram bin to increment. The function f_{pe} is a priority encoder applied to bits 9 through 18 of $L_w(x, y)$, and $>>$ is a bitshift operator.

$$\nu = f_{pe}(L_w(x, y) >> 8) \tag{4a}$$

$$H(L_w(x, y)) = (L_w(x, y) >> \nu) + \nu * 2^8 \tag{4b}$$

Since we only use 8 bits of mantissa for every luminance value in our image, Eq. 4b requires only 256 bins for every doubling of pixel value. To demonstrate this, Table 1 shows how a range of input luminance values from $min(L_w)$ and $max(L_w)$ will be mapped to our histogram range H_{min} to H_{max}.

$min(L_w)$	$max(L_w)$	ν	H_{min}	H_{max}
0	255	0	0	255
256	511	1	256	511
512	1023	2	512	767
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
65536	131071	9	2304	2559
131072	262143	10	2560	2815

Table 1. $L_w(x, y)$ mapping to histogram H of $L_w(x, y)$

Our optimized histogram storage method requires only 2816 RAM locations vs. the normal 262,144 locations. A 30Hz 1080p video has a vertical blanking time of 1.6ms. Since an FPGA Microblaze processor can compute the cumulative sum in 163 μs (4 clocks per value * 3072 values = 12288 cycles or 163 μs @ 75Mhz processor frequency), the cumulative sum can be computed during the vertical blank. The standard cumulative sum requires 14 ms., well beyond the vertical blanking period. The processor computes Eq. 1b, Eq. 1c, and Eq. 1a, then the TMO parameters are updated according to equations Eq. 3a through Eq. 3c. All processor computations are easily computed in the vertical blanking time for the n^{th} image which can be applied to the $(n + 1)^{\text{th}}$ image since the scene in video changes little from image to image.

The boundary values and tone mapping of $L_w(x, y)$ to $L_d(x, y)$ is then applied to the $(n + 1)^{\text{th}}$ using the updated parameters computed from the n^{th} image. These operations are implemented as pipelined operations in the FPGA. Not including the processor, Table 2 summarizes the resources required to implement our real time HDR video tone mapping operation in a Xilinx Spartan-6 LX150T.

Resource	Required/Available	%
Slices	1000/184304	0.5%
BRAMs	3/268	1%
LUTs	874/92152	1%
DSP48s	17/180	9%

Table 2. Hardware resource usage for Xilinx Spartan-6 LX150T

4. CONCLUSION

In this paper we presented a two-part solution to two relatively new problems in standard TMO's, which have just recently surfaced because of the recent emergence of camera-captured HDR video. We also described how this two-part solution along with the photographic operator [3] can be implemented in low cost FPGA hardware for use as a real time tone mapping system. Our results show that our real time tone mapping system is independent of user input, fully automatically adaptable to changes in scenery, and implementable in real time for video data rates. As a result the real time tone mapping system presented is a camera ready solution for tone mapping HDR video.

5. REFERENCES

[1] M. Tocci, C. Kiser, N. Tocci, and P. Sen, "A versatile hdr video production system," *ACM Transactions on Graphics*, vol. 30(4), pp. 41, 2011.

[2] E. Reinhard, G. Ward, P. Debevec, P. Sumanta, W. Heidrich, and K. Myszkowski, "Parameter estimation for photographics tone reproduction.," *Journal of Graphics Tools*, vol. 7(1), pp. 45–51, 2003.

[3] E. Reinhard, Stark M., Shirley P., and Ferwerda J., "Photographic tone reproduction for digital images.," *ACM Transactions on Graphics*, vol. 21(3), pp. 267–276, 2002.

[4] Randy Yates and Richard. Lyons, "Dc blocker algorithms.," *IEEE Signal Processing Magazine*, pp. 132–134, March, 2008.

[5] E. Reinhard, G. Ward, P. Debevec, P. Sumanta, W. Heidrich, and K. Myszkowski, *High Dynamic Range Imaging.*, Morgan Kaufmann Publishers, San Francisco, 2nd edition, 2010.

[6] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images.," *ACM Transactions on Graphics*, vol. 21(3), pp. 257–266, 2002.

[7] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression.," *ACM Transactions on Graphics*, vol. 21(3), pp. 249–256, 2002.

[8] Y. Li, L. Sharan, and E. Adelson, "Compressing and companding high dynamic range images with subband architectures.," *ACM Transactions on Graphics*, vol. 24(3), pp. 836–844, 2005.

[9] R. Mantiuk, S. Daly, and L. Kerofsky, "Display adaptive tone mapping.," *ACM Transactions on Graphics*, vol. 27(3), pp. 68, 2008.

[10] F. Hassan and J. Carletta, "An fpga-based architecture for a local tone-mapping operator.," *Real-Time Image Processing*, vol. 2, pp. 293–308, 2007.

[11] G. Ward, H. Rushmeier, and C. Piatko, "A visibility matching tone reproduction operator for high dynamic range scenes.," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3(4), pp. 291–306, 1997.

[12] P. Ledda, A. Chalmers, T Troscianko, and H. Seetzen, "Evaluation of tone mapping operators using a high dynamic range display.," *ACM Transactions on Graphics*, vol. 24(3), pp. 640–648, 2005.