GPU-ACCELERATED LOCAL TONE-MAPPING FOR HIGH DYNAMIC RANGE IMAGES

Qiyuan Tian[†], *Jiang Duan*[††], *Guoping Qiu*[†††]

[†]Department of Electrical Engineering, Stanford University
[††]School of Economic Information Engineering, Southwestern University of Finance and Economics, Chengdu, China
[†††]School of Computer Science, The University of Nottingham, UK

ABSTRACT

This paper presents a very fast local tone mapping method for displaying high dynamic range (HDR) images. Though local tone mapping operators produce better local contrast and details, they are usually slow. We have solved this problem by designing a highly parallel algorithm, which can be easily implemented on a Graphics Processing Unit (GPU) to harvest high computational efficiency. At the same time, the proposed method mimics the local adaption mechanism of the human visual system and thus gives good results for a wide variety of images.

Index Terms— Local tone mapping, high dynamic range, parallel computation, GPU, CUDA

1. INTRODUCTION

Dynamic range of a scene or an image is defined as the ratio of the highest to the lowest luminance. The real world scenes often have a very wide range of luminance (Fig. 1), sometimes exceeding 10 orders of magnitude. To reproduce these scenes presents a challenge for conventional digital capture and display devices, which suffer a limited dynamic range of only 2 orders of magnitude. Radiance maps [1, 2], obtained by merging a sequence of low dynamic range (LDR) images of the same scene taken under different exposure intervals (Fig. 1), are able to record the full dynamic range of the scene in 32-bit floating-point number format. However, LDR reproduction devices such as CRT monitor and printer are usually only 8-bit per color channel. Tone Mapping or Tone Reproduction is the process to compress the dynamic range of the radiance maps to fit into that of the display devices, while preserving as much of visibility and visual contrast as possible.

This paper addresses this issue by presenting a novel GPU-accelerated local tone mapping method for displaying HDR images. The organization of the paper is as follows. In the next section, we briefly review previous works of tone mapping and tone mapping on the GPU. We describe our algorithm in Section 3 and its GPU implementation in Section 4 in detail. Section 5 presents experimental results and Section 6 concludes the paper.

Fig.1. Selected multi-exposed image set of the same scene.

2. REVIEW OF TONE MAPPING METHODS

Tone mapping operators are usually classified as either global or local. Global tone mapping techniques apply the same appropriately designed mapping function to every pixel across the image. [3] and [4] are pioneering works. The operators attempt to match the display brightness with real world sensations, and match the perceived contrast between the displayed image and the scene respectively. Later, [5] proposes a technique based on a comprehensive visual model, successfully simulating important visual effects like adaptation and color appearance. Further, [6] presents a method based on logarithmic compression of luminance values, imitating the human response to light. Recently, [7] formulates the tone mapping problem as a quantization process and employs an adaptive conscience learning strategy to obtain mapped images. Perhaps the most comprehensive technique is still that of [8], which first improves histogram equalization and then extends this idea to incorporate models of human contrast sensitivity, glare, spatial acuity, and color sensitivity effects.

Local tone mapping techniques use spatially varying mapping functions. [9-12] are based on the same principle of decomposing an image into layers and differently compressing them. Usually, layers with large features are strongly compressed to reduce the dynamic range while layers of details are untouched or even enhanced to preserve details. [13] presents a method based on a multiscale version of the Retinex theory of color vision. [14] attempts to incorporate traditional photographic techniques to the digital domain for reproducing HDR images. [15] compresses dynamic range through the manipulation of the gradient domain in the logarithmic space. More recently, [16] proposes a novel method by adjusting the local histogram.

There has been little published research to explore rendering HDR images on the GPU [17,18] and these works

have two noticeable drawbacks: (1) implementing already existing tone mapping methods instead of designing a new algorithm with the GPU in mind, in which case the high parallelism may not be guaranteed; (2) transmitting computation to the GPU by mapping general purpose tasks to graphics pipeline, which requires proficiency of graphics programming language like OpenGL. In comparison, our method is inherently parallel and can be easily implemented on the GPU using Compute Unified Device Architecture (CUDA) [19].

3. ALGORITHM

Local tone mapping methods involve spatial processing and therefore have advantages in preserving details and local contrast over global ones. However, it implies in a larger amount of computational cost, making local tone mapping methods slow and unsuitable in real time applications such as HDR video. We solve this problem by rendering individual pixels in parallel. Local tone mapping computation is based on individual pixels and achieves localization by considering local pixel statistics and contexts, which damages the parallelism of the system and therefore is unsuitable to be implemented on the GPU. Our method divides images into non-overlapping rectangular blocks and reproduces contrast and brightness in each of them simultaneously using a highly parallel global tone mapping operator. This strategy subtly addresses the issue that local operators are hard to parallelize and provides promising methods for GPU acceleration. On the other hand, our design is also consistent with how the human visual system copes with high contrast scenes [16] and consequently ensures visual quality of the mapped images.

3.1. Global tone mapping in local regions

First, our algorithm segments images into independent rectangular blocks, in which we conduct *HALEQ* [16], a fast global tone mapping method with potential to be paralleled as discussed in Section 4.2. *HALEQ* works by striking a balance between linear compression and histogram equalized compression in the mapping process as Eq. (1). The left image in Fig. 2 shows the mapping result.

$$d(x,y) = \beta \cdot EC[D(x,y)] + (1-\beta) \cdot LC[D(x,y)] \quad (1)$$

$D(x, y)$ is the input luminance while $d(x, y)$ is the output display intensity level. EC and LC is the histogram equalization and linear mapping function respectively. β controls the contrast enhancement level.

3.2. Boundary and halo artifacts elimination

The next step is to fight boundary artifacts between blocks and "halo" around edges (like between the building and the sky in the left image of Fig. 2), which are due to the fact that pixels with similar values but on the opposite sides of the boundary can be projected to have very different values [16].

Fig.2. Left: direct *HALEQ* result; middle: after eliminating boundary and halo artifacts; right: denoised final result.

Our solution is to incorporate spatial information within different blocks to obtain the final result for a pixel. To be specific, the final mapped pixel value is the weighted average of the results from mapping functions $HALEQ_n$ of neighboring blocks as Eq. (2). The middle image in Fig. 2 shows the mapped result until this step. The weighting process can be applied to individual pixels concurrently as all mapping functions $HALEQ_n$ ($1 \le n \le R$, R is the number of segmented blocks) have been already derived in Section 3.1. The parallelism benefits from the segmentation design.

$$d(x,y) = \frac{\sum_{n=1}^{n=N} HALEQ_n[D(x,y)] \cdot w_d(n) \cdot w_s(n)}{\sum_{n=1}^{n=N} w_d(n) \cdot w_s(n)} \quad (2)$$

$$w_d(n) = e^{-(d_n/\sigma_n)}, w_s(n) = e^{-(s_n/\sigma_s)} \quad (3)$$

N is the number of used blocks. w_d and w_s is the distance weighting function and pixel value similarity weighting function respectively. d_n is the Euclidean distance between the current pixel position and the center of used blocks. s_n is the normalized difference between the current pixel value and the average pixel value of block n. σ_d and σ_s control the smoothness between blocks. Larger values of N, σ_d and σ_s facilitate the elimination of boundary and halo artifacts but produce images with less local contrast.

3.3. Local contrast enhancement adjustment

Another problem of the algorithm is to introduce noises in uniform areas (like the sky in the middle image of Fig. 2) if a common parameter β is applied in Eq. (1). This is because the contrast enhancement is so strong for uniform areas, although proper for the others, that similar pixels are mapped to have quite different values. We adaptively decrease β for uniform areas as Eq. (4) to deal with the issue.

$$\beta = 0.6 \cdot [1 - e^{-(SD_{n_{\max}} - SD_n)}] \quad (SD_n > \eta) \quad (4)$$

SD_n is the deviation of histogram population for each region. The threshold η is empirically set for different images and determines at what level the region is regarded as uniform. The right image in Fig. 2 shows the denoised image.

4. GPU IMPLEMENTATION

4.1. Basics of CUDA

GPU has gained considerable computational power and the introduction of programmability has enabled its use outside the original application domain of computer graphics for more general purposed computing tasks. CUDA is a newly emerged scalable parallel programming model and a software environment for parallel computing on the GPU [19]. It allows almost direct translation of C codes onto the GPU, with the syntax consisting of minimal extensions of the C language.

CUDA programmers accomplish computation tasks on the GPU via launching kernels. One important way in which kernels differ from normal C functions is that they are executed in parallel, over a large number of CUDA threads. Individual threads concurrently execute the same kernel program on different data. Threads are organized into blocks and blocks make up grids, as shown in Fig. 3. Built-in variables threadIdx, blockIdx and gridIdx, up to three dimensions, help locate a thread and determine what data it works on. The tricky parts of CUDA programming are to decide the grid and block size, and identify target data using the mentioned ID variables. A kernel program is launched as:

kernel<<<grid_size, block_size>>>(arg);

Constructing local mapping functions (Section 3.1) and the weighting process (Section 3.2) are the two most computationally demanding parts of our algorithm. The next two sections describe their CUDA implementation.

4.2. Local mapping function construction acceleration

Besides deriving the local mapping functions in each region concurrently, we also propose a parallel implementation of *HALEQ* operator as shown in Fig. 4. This recursive binary cut approach first divides the range of *D(I)* into two segments according to Eq. (1) [16] on level 1. The two new segments are then independently divided into 2 segments similarly on level 2. The process is recursively applied to each resultant segment until the predefined number of segments (256) are created. Each segment is allocated a displayable value (an integer within [0, 255]). Since the 2^{i-1} cuts are created independently on level i, we calculate them in parallel by launching one kernel program on each level as:

DeriveHALEQ_i<<<Grids, 2^{i-1}>>>(arg); (i = 1, 2 ... 8)

DeriveHALEQ_i is the calculation on level i. Grids is a two dimensional variable with each component equal to the number of blocks vertically and horizontally. Kernels are so launched to ensure that one CUDA block is responsible for constructing the mapping function in one local block and each thread serves to create a new cut between segments.

4.3. Weighting process acceleration

As discussed in Section 3.2, the weighting process can be conducted as Eq. (2) for all pixels concurrently. We pre-calculate the distance weighting function and the similarity weighting function, and then launch only one kernel as:

Weighting <<<Grids, Blocks>>>(arg);

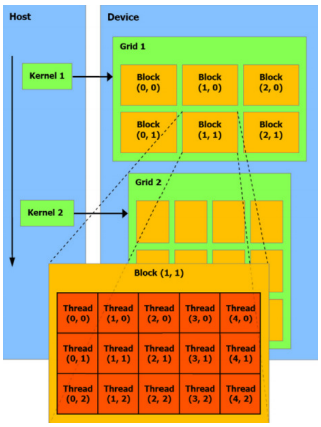

Fig.3. CUDA threads organization. Courtesy of NVIDIA.

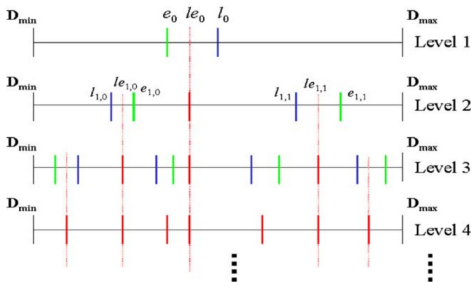

Fig.4. Recursive binary cut approach for *HALEQ*.

Weighting is the kernel program to calculate Eq. (2). Grids is the same variable as that of the kernel in Section 4.2. Blocks is a two dimensional variable, whose first and second dimension size is equal to the number of pixels of a local block horizontally and vertically. In this manner, each CUDA thread is in charge of the weighting process for one pixel to get the final mapping result.

5. EXPERIMENTAL RESULTS

5.1. Mapping results

Fig. 5 shows two examples of the resultant images, which are comparable to those of state-of-the-art techniques like [11, 14]. Compared with Durand and Dorsey's super mapping operator [11] (Fig. 6), our method has the advantage of preserving more details (like in the window areas) while theirs produces images with more local contrast.

5.2. Computational efficiency

To demonstrate the computational efficiency of our method, we implement it on both the CPU and GPU. For the test image Clock Building with 768×1024 pixels (Fig. 2) divided into 64×64 blocks, it takes 1.477s for an i5-2410M CPU at 2.30Hz with 4GB RAM running 64-bit Windows 7 Ultimate to compute the final result. Local mapping function construction and weighting process occupies 0.392s and 0.899s respectively. The GPU experimental platform is NVIDIA GeForce GT 550M with 2 multiprocessors.

Without carefully considering optimizations of memory use and the cooperation between the CPU and GPU, CUDA codes shorten the time to 0.358s. Specifically, the mapping function construction time is reduced to 0.172s while the weighting process time to 0.103s, from which we experience about 2 and 9 times speedup. The reason that the weighting process has gained a higher ratio of acceleration is that it has more computations with potential to be paralleled.

The current running time is satisfying, which is about 5 times faster than Fattal et al.'s operator [15], 6 times faster than Reinhard et al.'s operator [14], and comparable to Durand and Dorsey's method [11]. More importantly, the CUDA model has so fine scalability that a GPU with more multiprocessors easily brings further acceleration in scale, which is very likely to decrease the experimental time 0.358s down to a real time level. This feature provides our novel GPU accelerated tone mapping method a promising advantage for practical applications.

6. CONCLUSION AND FUTURE WORK

We have presented a novel GPU-accelerated tone mapping method, which has been demonstrated fast and effective for displaying HDR images. Future work focuses on optimizing the CUDA implementation and using a better GPU to render HDR videos in real time.

7. ACKNOWLEDGEMENTS

Radiance maps used in this paper courtesy of corresponding author(s). This project Sponsored by National Natural Science Foundation of China (Grant No. 60903128), the Scientific Research Foundation for the Returned Overseas Chinese Scholars and the Program for New Century Excellent Talents in University of State Education Ministry, and Excellent Youth Foundation of Sichuan Scientific Committee (2012jq0017).

8. REFERENCES

[1] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs", Proc. ACM SIGGRAPH'97, pp. 369 – 378, 1997.
[2] T. Mitsunaga, S. K. Nayar, Radiometric self calibration, "Proceedings of the Computer Vision and Pattern Recognition, vol.1, 1999, pp.374–380.
[3] J. Tumblin and H. Rushmeier, "Tone reproduction for realistic images", IEEE Computer Graphics and Applications, vol. 13, pp. 42– 48, 1993.
[4] G. Ward, A contrast-based scalefactor for luminance display, in: Graphics Gems IV, Academic Press, 1994, pp. 415–421.
[5] J.A. Ferwerda, S.N. Pattanaik, P. Shirley, D.P. Greenberg, A model of visual adaptation for realistic image synthesis, in: Proceedings of the SIGGRAPH'96, 1996, pp. 249–258.
[6] F. Drago, K. Myszkowski, T. Annen and N. Chiba, "Adaptive Logarithmic Mapping For Displaying High Contrast Scenes", The Journal of Computer Graphics Forum, Vol.22, No, 3, pp. 419-426, 2003.9.

Fig.5. Two mapped results of our method.

Fig.6. Comparison with other local operators. From left to right: result of ours, Durand & Dorsey's [11], Reinhard's [14].

[7] J. Duan, G. Qiu, G. M. D. Finlayson, Learning to display high dynamic range images, Pattern Recognition 40 (10) (2007) .
[8] G. W. Larson, H. Rushmeier, C. Piatko, "A visibility matching tone reproduction operator for high dynamic range scenes", IEEE Trans on Visualization and Computer Graphics, vol. 3, pp. 291 – 306, 1997.
[9] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang and K. Zimmerman, "Spatially nonuniform scaling functions for high contrast images", Proc. graphics Interface'93, pp. 245 – 253, 1993
[10] J. Tumblin and G. Turk, "LCIS: A boundary hierarchy for detail preserving contrast reduction", In Proc. of ACM SIGGRAPH'99, pp. 83-90.
[11] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images", ACM Trans. Graph. (special issue SIGGRAPH 2002) 21, 3, 257-266, 2002.
[12] X. Li, K. Lam, L. Shen, "An adaptivea lgorithm for the display of high-dynamic range images", Journal of Visual Communication and Image Representation, 18 (5) (2007) 397–405.
[13] D. J. Jobson, Z. Rahman and G. A. Woodell, "A multiscale Retinex for bridging the gap between color images and the human observation of scenes", IEEE Transactions on Image processing, vol. 6, pp. 965-976, 1997
[14] E. Reinhard, M. Stark, P. Shirley and J. Ferwerda, "Photographic tone reproduction for digital images", Proc. ACM SIGGRAPH'2002.
[15] R. Fattal, D. Lischinski and M. Werman, "Gradient domain high dynamic range compression", Proc. ACM SIGGRAPH'2002.
[16] J. Duan, M. Bressan, C. Dance and G. Qiu, "Tone-mapping high dynamic range images by novel histogram adjustment". Pattern Recognition, vol. 43, no.5, pp. 1847-18622010.
[17] Goodnight N., Wang R., Woolley C., Humphreys G.: Interactive time-dependent tone mapping using programmable graphics hardware. In Proceedings of the 14th Eurographics Workshop on Rendering, pp. 26–37, June 2003
[18] Zhao, H. and Jin, X. and Shen, J., Real-Time Tone Mapping for High-Resolution HDR Images, International Conference on Cyberworlds, pp. 256--262, 2008.
[19] http://developer.nvidia.com/object/cuda.html.