Implementation of Parallel Components of High Dynamic Range Images Algorithm Using FPGA

Salim A. Mohammed Ali*, Aymen M. Al-Kadhimi
Al-Nahrain University, College of Information Engineering, Department of Computer Networks Engineering, Baghdad, Iraq

Email: salimm@ieee.org*, aymen.mohammed@coie-nahrain.edu.iq

Abstract. High Dynamic Range (HDR) images capturing, and reproduction have been an essential area to investigate by researchers, especially after the escalating acceleration in computers processing capabilities and the widespread of digital cameras. In this paper, parallel components of HDR algorithm are employed using field programmable gate array (FPGA) to produce HDR image. The implementation involves different pre-processing stages to separate images, color matrices, and then group the similar colors to perform a series of arithmetic operations. In consequence, the algorithm has inherited parallelism to exploit so that the FPGA provides parallel computational speed. The algorithm is implemented using Simulink and ModelSim-Altera MATLAB software to produce the response curve and HDR color components signals. After preforming the tone mapping, the results show that the HDR image provides much greater details and a truer color.

1. Introduction

A High Dynamic Range (HDR) image was first introduced in 1995 by Mann and Picard [1], and the idea was to recover and extend the dynamic illumination range of an image captured by a digital camera. The camera sensor limitation forces the captured image to have less illumination range than what human eyes actually recognize. One way to overcome this issue is to combine multiple snapshots of the same image as having different exposure times. Unlike other low-level computer vision algorithms, implementing HDR images requires many computation steps using floating points, addition, multiplication, exponential computation, and the intricacy increases with the number of images to be combined. For this reason, the challenge now is how to produce HDR for high quality images or moving pictures in real-time manner. Several hardware implementations have been introduced recently using GPU and FPGA for real-time purposes, but it is still not at the quality which is required by the marketplace nowadays.

Our goal is to implement the parallel components of the HDR algorithm optimized for running on an FPGA. Basically, the algorithm will have to combine the data from three image matrices. Some of the images will be intentionally under exposed and others will intentionally be over exposed. Simulink MATLAB are used as a simulation and design tool for the FPGA platform along with ModelSim-Altera to produce response curve and HDR color components signals. The paper is organized as follows: the next section provides a background study for partial and full implementations; the third section describes the methodology and the work design; the fourth section discusses the results; and finally, the paper ends with conclusions.
2. Literature Review

In terms of pixel weighting techniques used for combining low dynamic range LDR images into an HDR image, many techniques have been contributed to literature. The weighting function proposed by Mann and Picard [1] in which the weights are assigned according to deriving the inverse of camera response curve. In consequence, the quantization errors are reduced due to the assigning less weight to lower digital number valued pixels. That work was extended by Roberson et al. [2] by presenting a different weighting function that quadratically rises with exposure time. As a result, the extra quadratic exposure part leads to applying more weight for pixels whose exposure time is the longest. Afterwards, another weighting function was proposed by Reinhard et al. [3] in which they added a hat function to aid limiting the under and over exposed pixels, and hence a weighting function with optical signal to noise ratio SNR is offered. on the other hand, a very effective method for HDR pixel combining is the winner takes all technique [4-6]. In this technique, the pixel is used with the longest exposure if the value is not saturated yet. Also, a weighting function is mimicked in this approach, but zero weight is placed into any pixel that is not a non-saturated pixel with the longest exposure. This technique is argued according to the case that higher SNRs are addressed for those non-saturated pixels with longer exposure compared to the shorter ones [5]. Most of the techniques used for weighting functions are presented in [7] along with showing results and camera response curves. In this paper, the weighting function that is used is the one proposed by Debevec and Malik [8] in which the 1 stop exposure difference between images is the optimized spacing to minimize noise with maximum dynamic range. This technique is implemented in Simulink using altera DSP Blocks as discussed in detail in Section 3.

In addition, new methods have been developed that improve the HDR images. O. Kwon et al. [10] proposed a new HDR image coding scheme defined for JPEG XT by applying adaptive prediction technique. On the other hand, D. Kundu et al. [11] presented a novel model and algorithm for HDR images with elevated performance in terms of image quality prediction. Since processing needs to be done in real time, whatever system is running the algorithm needs to be able to keep up. Since the fused image method still has innate parallelism in it, an FPGA system might be used to keep up with the real time needs of the system. Parallel implementation methods of algorithm components will facilitate of using heterogeneous computing devices to reach higher speed-ups, which then reduces processing time by many levels [12-13].

3. Methodology

FPGAs can provide a major speed advantage to the HDR computation by parallelizing many of the steps. However, to take advantage of the parallelism provided by FPGAs, the algorithm needs to be designed to make sure the data is provided in a way that it can be operated on in parallel. It is also important that the algorithm has inherited parallelism that can be exploited. Luckily for us the HDR algorithm has many portions that can be done in parallel. Below in Figure 1 is a flow diagram that gives a basic understanding of how the HDR algorithm works.

![Figure 1. HDR flow chart.](image-url)
As shown each image can be thought of as a set of matrices. Each matrix’s elements symbolize the intensity value of the red, green, or blue for the pixels of the image. Since the HDR algorithm acts on the three-color matrices separately, the operations on all three matrices can be performed at the same time. Once the images are separated and the separate colors grouped, a series of arithmetic operations are performed to compute a response function for each color. The implementation of Debevec and Malik [8], is based on the following equation:

\[
\ln E_i = \frac{\sum_{j=1}^{p} w(Z_{ij})(g(Z_{ij}) - \ln t_j)}{\sum_{j=1}^{p} w(Z_{ij})}
\]

(1)

where \( Z_{ij} \) represents the \( i \)th pixel from the \( j \)th image. \( t_j \) is exposure time of the \( j \)th image such that optimal HDR reconstruction uses exposure spacing of 1. \( E_i \) are the recovered irradiance values. The function \( g(Z) \) is the camera response curve which recovers the radiance from the measured pixel values and \( w(Z_{ij}) \) is a weighting function used in the averaging process. This response function is what the HDR algorithm uses to decide the color intensity value for each pixel. A major part of ensuring the algorithm can operate in parallel is preparing the data to be processed. For this reason, the images go through a pre-processing phase in MATLAB. So, the design basically implemented in subsystems such as: weight matrix builder, B-matrix, and final radiance map lookup table. The following design steps were carried out:

### 3.1 Building weight matrix

Weight matrix is arbitrary in structure, and different implementations have been presented to produce an efficient matrix as in [9]. In this work, the Debevec and Malik [8] weight function, which is basically a triangular Gaussian hat, is implemented. The following equation was implemented in Simulink using altera DSP Blocks.

\[
w(z) = \begin{cases} 
z - Z_{\text{min}} & \text{for } z \leq \frac{1}{2}(Z_{\text{min}} + Z_{\text{max}}) \\
Z_{\text{max}} - z & \text{for } z > \frac{1}{2}(Z_{\text{min}} + Z_{\text{max}}) \end{cases}
\]

(2)

Figure 2.a shows the blocks component of the design,

**Figure 2.** (a) Simulink block design of the weight matrix. (b) Weight matrix curve (triangular Gaussian hat).

The graph in Figure 2.b presents the output weight matrix. The purpose of the weight matrix is to build the other HDR algorithm parameters, like matrices A, B, G, and E.
3.2 Building algorithm parameters

First, B-vector is the part of the algorithm that if it is multiplied with the inverse of matrix A, it will produce the expected camera radiance curve, G vector. The inputs of the block that computes the above vector are the exposure times of each LDR image and the weight matrix. The following equation shows the calculation of the B matrix:

\[ B = W(i,j) \ast \log(T(j)) \]  

Where \( i \) is the intensity of the pixel, \( j \) is LDR image number, \( W \) is the weight function, and \( T \) is the exposure time. The output of the block was redirected to the MATLAB workspace for further processing. The implementation of the B-Matrix is shown in Figure 3.

3.3 Building the Radiance Map Model

After all the parameters of equation 1, are computed, the final step is mapping each mid exposure image color channel to its correspondence radiance map. The method was implemented as follows:

a) The LDR images color channels were transformed to 1-D time series, so, it can be used as input to our final step model.

b) Each color channel were mapped separately and independently, so, parallelism will be exploited in this step.

c) The time series data forwarded to a lookup table 8-bit at a time, this ensures the lookup table will pick up the exact mapping for each pixel of the proper exposed LDR image.

d) The output were forwarded toward the sink, which will then combine with other color channels to produce the High dynamic range image.

e) At this point the HDR image cannot be displayed on a 8-bit display devices. Therefore, is has to be toned by a tone mapper.

Figure 4.a reveals the camera response curve which recovers the radiance from the measured pixel values.
Figure 4.b represents the lookup table hardware design block for the radiance map, since it is much faster to use memory mapping than calculating it using math operations. Figure 5 shows all the components that are implemented for the HDR algorithm. It is easily can be identified that these components can run in parallel leading to the achievement of low latency processing especially when large input images are used. Hence, LDR images are not limited to 3, to obtain better HDR images we can combine as many as LDR images depending on the hardware system capacity. When large number of LDR images is used, conventional processing power will fail to produce HDR image in real-time. Parallel processing of such application will be promising.

![Figure 5. Overall components design](image)

4. Results and Discussion
The lookup table design, was compiled and simulated on ModelSim-Altera and the results can be shown in Figure 6.a. The results all the waveforms that are implemented in the design, including each color component. Figure 6.b shows a subjective example of 3 captured images with different intensities; exposure times, and then followed by the output of the implemented HDR system. Then the final toned image shows the actual HDR image after tone mapping. It can be easily recognized the effect of HDR by replacing the dark areas with high contrast images while keeping bright areas balanced.
5. Conclusion

To make HDR algorithm run much faster, pipelining can be added to the code of the algorithm. In this paper, the implemented FPGA operates simultaneously on three LDR input color images. Since most of the steps operate on a single pixel independent of all others, more pipelines could be added to do more operations in parallel. By creating more pipelines to the FPGA, the amount and the complexity of the pre-processing will be increased because it is required that the data should pass to the FPGA be properly synchronized. Also, it becomes more complicated when trying to merge all the outputs back into a single stream; however, the complexity of the algorithm would be subdued by the parallelism and would outweigh the extra complexity that is incurred. Working with FPGAs needs to determine the number of inputs limitation and the available bus bandwidth of the FPGAs, as these would be the main limitations in parallelizing the algorithm. This paper shows that FPGAs can be used to construct HDR images. This could be used to create HDR images in real time which is a major interest for using with surveillance systems and many other applications. FPGAs are able to exploit the parallelism in algorithms by enabling to do some types of processing much faster. Through a proper implementation of the algorithm and a proper preprocessing, the FPGA can significantly cut down the operation, and in general the amount of parallelism is usually only limited by the bus bandwidth and the number of input ports, both of which are fairly large.

References

[1] S. Mann and R.W. Picard, “On being „undigital” with digital cameras: Extending Dynamic Range by combining differently exposed pictures”, in proceedings of IS&T 48th Annual Conference Society for Imaging Science and Technology, Cambridge, Massachusetts, May 1995, pp. 422-428.

[2] M.A. Robertson, S. Borman, and R.L. Stevenson. "Estimation-theoretic approach to dynamic range enhancement using multiple exposures." Journal of Electronic Imaging 12, no. 2 pp. 219-229, 2003.

[3] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec, "High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting," in The Morgan Kaufmann Series in Computer Graphics, 309-313 pp., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[4] JAI, Using image fusion to capture high-dynamic range HDR scenes., Tech Note, TN-0903, 2009.

[5] W. Kao, "High Dynamic Range Imaging by Fusing Multiple Raw Images and Tone Reproduction," in IEEE Transactions on Consumer Electronics, vol. 54, no. 1, pp. 10-15, February 2008, doi: 10.1109/TCE.2008.4470017.

[6] W. Kao, C. Hsu, L. Chen, C. Kao and S. Chen, "Integrating image fusion and motion stabilization for capturing still images in high dynamic range scenes," in IEEE Transactions on
Consumer Electronics, vol. 52, no. 3, pp. 735-741, Aug. 2006, doi: 10.1109/TCE.2006.1706464.

[7] A. M. Sa, High Dynamic Range Image Reconstruction, 1st ed., pp. 24-36 pp, 2008.

[8] P. Debevec, and J. Malik, “Recovering high dynamic range radiance maps from photographs,” in SIGGRAPH’97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 369-378, August 1997, doi:10.1145/258734.258884

[9] W.C. Kiser, “High Dynamic Range Video Merging, Tone Mapping, and Real-time Implementation”, PhD dissertation, university of New Mexico, December 2012.

[10] O. Kwon, S. Choi and D. Shin, "Improvement of JPEG XT Floating-Point HDR Image Coding Using Region Adaptive Prediction," in IEEE Access, vol. 6, pp. 3321-3335, 2018, doi: 10.1109/ACCESS.2018.2793228.

[11] D. Kundu, D. Ghadiyaram, A. C. Bovik and B. L. Evans, "No-Reference Quality Assessment of Tone-Mapped HDR Pictures," in IEEE Transactions on Image Processing, vol. 26, no. 6, pp. 2957-2971, June 2017, doi: 10.1109/TIP.2017.2685941.

[12] S. A. M. Ali and E. H. Al-Hemiary, "A Parallel Implementation Method for Solving the Shortest Path Problem for Vehicular Networks," 2020 1st. Information Technology To Enhance e-learning and Other Application (IT-ELA, Baghdad, Iraq, 2020, pp. 121-126, doi: 10.1109/IT-ELA50150.2020.9253086.

[13] S. A. M. Ali, A. M. Al-Kadhimi and S. Hasan, "Investigating the Parallel Components of TLD Algorithm Using OpenCL Computation Framework," 2018 10th Computer Science and Electronic Engineering (CEEC), Colchester, United Kingdom, 2018, pp. 36-40, doi: 10.1109/CEEC.2018.8674189.