Advanced Science and Technology Letters
Vol.58 (Signal Processing 2014), pp.87-90
http://dx.doi.org/10.14257/astl.2014.58.18

Improved Parallel Image Processing Algorithms by CUDA in GPU Environment

ChangWon Lee[1] and Tae-Young Choe[1],

[1]Department of Computer Engineering, Kumoh national Institute of Technology, 61 Daehak-ro (yangho-dong), Gumi, Gyeongbuk 730-701, Korea
lcw3564@naver.com, choety@kumoh.ac.kr

Abstract. Integral histogram enables constant time histogram computation of an area. Mark Harris proposed a parallel prefix sum algorithm in CUDA GPGPU for integral histogram initialization. Because of the restricted number of threads in a block in CUDA, Harris' algorithm divides a large image into multiple blocks. Such division increases the number of global memory access and becomes a major reason of performance degradation. In this paper we propose an allocation scheme that maps multiple pixels into a thread when the integral histogram is initialized for a large image. The proposed allocation scheme fully utilizes shared memory and reduces the number of accesses to global memory. Experimental results shows that the execution time of the proposed algorithm is 94.7% ~ 99.8% compared to that of Harris' algorithm.

Keywords: Integral histogram, parallel prefix sum, CUDA, shared memory.

1 Introduction

Histogram is a major method to represent characteristics of an image. Time to calculate a histogram in a strict way increases in accordance with the size of an interesting area. Integral histogram is the accumulated histograms of all pixels in an area of an image. Integral histogram enables to compute a histogram of an area within a constant time. Unfortunately, construction of integral histogram requires linear computation time compared to the size of image. Recently, video resolution with 4096 size is adopted by brand new monitors and movie media. On the basis of these features, fast construction of integral histogram affects performance of applications that uses such high resolution video. Thus some parallel constructions of integral histogram have been proposed. Parallel prefix sum algorithm is a method to construct integral histogram in parallel.

Mark Harris proposed a parallelintegral histogramconstruction algorithm and implemented it in CUDA GPGPU. The algorithm reduces not only the amount of calculations but also the execution time compared to the previous parallel prefix sum algorithms, which is possible by sophisticated usage of shared memory in GPGPU.The algorithm divides an image into multiple blocks in order to process using shared memory which has restricted size. Such division increases the number of global memory accesses and becomes a major reason of performance degradation.

In this paper, we propose an improved parallel construction algorithm of integral

ISSN: 2287-1233 ASTL

histogram that eliminates the global memory accesses by mapping multiple pixels in to a thread. The remainder of this paper is organized as follows: Section 2 presents related works. The proposed algorithm is explained in Section 3. Section 4 gives experimental results. Finally, Section 5 concludes.

2 Related Works

Mark Harrisproposed a parallel version of prefix sum algorithm called *work efficient parallel prefix sum*[1]. It is possible to construct a prefix sum in fewer operations than a simple parallel sum algorithm [2], [3].Although the number of steps of Harris' algorithm is greater than that of simple one,manipulating shared memoryand reduced addition operations enablefaster execution time.

Senguptaet al. implemented segmented scan in CUDA [4]. The algorithm uses head flag in order to divide entire array to multiple different sized blocks. For each block, different operation can be applied.Another algorithm proposed by Sengupta et al. divides an array into multiple blocks with warp size in order to maximize warp occupancy [5].

Above three algorithms utilize the work efficient parallel prefix sum which consists of two phases:an up-sweep and a down-sweep.The up-sweep phase is similar to bottom up tree traverse such as value of a node is added to its right sibling node recursively until root node. In the down-sweep phase, values of nodes are exchanged or added in order to make partial sum. Since threads communicate each other frequently, a shared memory for a block is used to communicate between threads.

At a start time, a thread covers two pixels in Harris' algorithm. Thus, the algorithm can deal images such that width or height of the imagesare smaller than or equal to double of the maximum number of threads in a block. Unfortunately, the maximum number of threads in a block are limited 1024 in the case of NVDIA GTX670, butemerging video size is 4K (UHDTV of 3840×2160 or DCI-4K of 4096×2160) which exceeds 2048.Some GPGPUs like NVDIA Tesla architectures with compute capability is 1.3 or less support only 512 threads in a block. In order to solve the problem, Harris' algorithmdivides a large image into multiple appropriate sized sub-images. Each sub-image is stored in a shared memory of a block and prefix sum for the sub-image is computed by threads in the block. The largest value of each block is broadcasted to right side blocks in order to compute total partial sum.

The advantage of Harris' algorithm is that it can cover images with any size. The disadvantage of Harris' algorithm is the overhead occurred by big images. Broadcasting of partial sum is the major overhead and it requires global memory access that should be processed sequentially. Another overhead is adding the received values to all partial sums in the block in parallel.

3 Proposed Algorithm

It is noticed that the amount of shared memory for a block can accommodate a line of big image like 4K video frame. For example, NVDIA GTX670 allocates 48Kbytes to

a block [6]. Thus the size of shared memory is not restriction for prefix sum computation within a block. Only the maximum number of threads in a block limits size of image in the case of Harris' algorithm.

We solve the problem by allocating multiple pixels to a thread instead of using multiple blocks to cover entire image. Let l be the length of an image, that is, maximum of width and height, and N_t be the maximum number of available threads in a block of a GPU. If $l \leq 2N_t$, the number of threads to be used is $\lceil \frac{l}{2} \rceil$ and the algorithm works as same asHarris' algorithm does. Otherwise, the number of threads to be used is N_t. Let integer Qsatisfy range $2^Q N_t < l \leq 2^{Q+1} N_t$. Thread t_i $(0 \leq i < N_t)$covers pixel index $2^{Q+1} i \sim 2^{Q+1}(i+1) - 1$ at the initial step. The number of step increases up to Q.

The disadvantage of the algorithm is that the amount of used threads is restricted. Thus maximum parallelism of the algorithm is N_t. Such disadvantage can be overcome by parallel access of local memory because the algorithm does not wait sequential global memory access.

4 Performance Analysis

4.1 Experimental Environment

Harris' algorithm and our proposed algorithm are implemented in C programming language using CUDA.Two algorithms are compiled in Visual Studio 2010 with CUDA version 5.5. Experimental environment is Windows 7 operating system, Intel core i5 750 (2.67GHz), 8Gbytes main memory, and NVDIA GeForce GTX 670 graphics card.

4.2Performance Analysis

Error! Reference source not found. shows performances of Harris' algorithm and our proposed algorithm. Four types of images are used: FHD (1920×1080), QHD (2560×1440), UHD (3840×2160), and DCI-4K (4096×2160). Two algorithms are executed 10 times. Since GTX 670 graphic card allows a block to contain 1024 threads, FHD can be processed within a block by Harris' algorithm. As the result, the performances of the two algorithms are almost the same in the case of FHD image. The performance of the proposed algorithm passes aheadthat of the Harris' algorithm from QHD image because Harris' algorithm should process two blocks and access global memory. The relative execution time of the proposed algorithm against Harris' algorithm is 94.7% when DCI-4K image is processed.

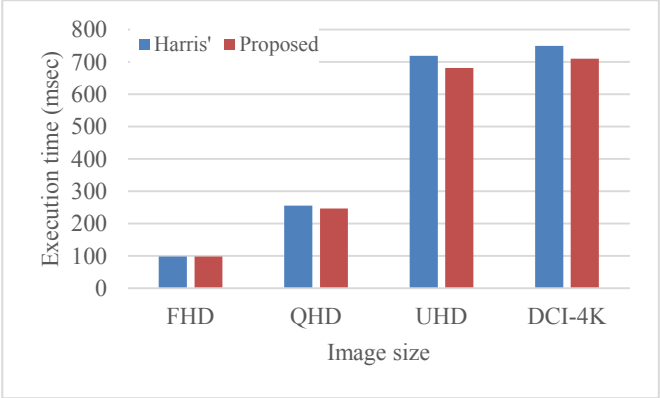

Fig.1. Performance Comparison

5 Conclusion

Integral histogram can extract a histogram of a specific area within a constant time. However, time to construct an entire integral histogram increases in accordance with the image size. In this paper, we propose an improved parallel prefix sum algorithm that can be used to construct integral histogram for big sized image. The proposed algorithm eliminates global memory access and maximizes the parallelism within threads in a block.

References

1. Harris M., Sengupta S., OwensJ. D.:Parallel Prefix Sum(Scan) with CUDA:GPU gems, vol. 3 no. 39, pp.851-876(2007)
2. Hills W. D., Steele JR.G. L.: Data Parallel Algorithms: Communications of the ACM, vol. 29, no. 12, pp. 1170–1183, ACM Press(1986)
3. HornD.: Stream reduction operations for GPGPU applications: GPU Gems, 2, Ch. 36, pp. 573–589, Addison Wesley (2005)
4. Sengupta S., Harris M., Zhang Y., OwensJ. D.: Scan Primitives for GPU Computing:ACM, Graphics Hardware, pp.1-11(2007)
5. Dotsenko, Y., Govindaraju, N. K., Sloan, P. P., Boyd, C., Manferdelli, J.: Fast scan algorithms on graphics processors: In Proceedings of the 22nd annual international conference on Supercomputing, pp. 205-213. ACM. (2008)
6. Silberstein, M., Schuster, A., Geiger, D., Patney, A., Owens, J. D.: Efficient computation of sum-products on GPUs through software-managed cache: In Proceedings of the 22nd annual international conference on Supercomputing, pp. 309-318, ACM(2008, June).

