International Journal of Grid and Distributed Computing
Vol.7, No.2 (2014), pp.99-114
http://dx.doi.org/10.14257/ ijgdc.2014.7.2.10

Multi-GPU Accelerated Parallel Algorithm of Wallis Transformation for Image Enhancement

Han Xiao[1, 2,*] Yu-Pu Song[3] and Qing-Lei Zhou[1]
[1] *School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China*
[2] *School of Information Science and Technology, Zhengzhou Normal University, Zhengzhou 450044, China*
[3] *School of Computer Science, Shangqiu Vocational & Technical College, Shangqiu 476100, China*
[*] *Corresponding author: e-mail: xiaohan70@163.com*

Abstract

With the development of satellite remote sensing technology, satellite remote sensing data obtained by the amount will increase rapidly. Consequently, the process of Wallis transformation is faced with such challenges as large data size, high intensity, high computational complexity and large computational quantity, and so on. A fast algorithm and efficient implementation of Wallis filtering based on Compute Unified Device Architecture (CUDA) is proposed. The parallel hardware architecture and software development process of multiple graphic processing unit (multi-GPU) is introduced firstly. On the basis of the parallel architecture and hardware characteristic of GPU, some algorithms are focused on computation and optimization of multiplicative coefficients, additive coefficients and updated gray values of the image pixels to improve the computing speed, and the shared memory is used to improve the data access efficiency. The experimental results clearly show that, in the same image quality, the average acceleration rate is 107 times, and the algorithm on multi-GPU can achieve better performance.

Key Words: *Graphic Processing Unit (GPU), Compute Unified Device Architecture (CUDA), Single Instruction Multiple Thread (SIMT), Wallis transform, Image enhancement*

1. Introduction

An image pre-processing algorithm can be applied to enhance the images for subsequent image processing. Research conducted within the reported project has found that Wallis filter is warranted to provide greater detail in shadowed areas and saturated areas simultaneously. And thus to allow a greater number of interest points to be detected [1].

With the increasing of people's earth observe ability, profited from the development of space remote sensing technology, the spatial resolution of satellite imagery has been substantially improved [2]. While providing more details it is hard to construct a real-time system for high spatial resolution image processing, because it takes much more time to process them with conventional algorithms [3]. Traditional serial processing algorithms facing many difficulties, therefore many research organization and scholars have studied parallel processing scheme [4]. But most of these schemes are based on special super computers or clusters: hardware is very expensive and the software development process is also complex, therefore it is not a good popularized scheme [5].

Recently, performances of programmable Graphics Processing Unit have rapidly developed. GPU has possessed powerful parallel computational capability [6]. At present,

ISSN: 2005-4262 IJGDC

Flops of advanced single-chip GPU has reached 1Tflops/s and the memory bandwidth is up to 192.4GB/s which have exceeded that of mainstream CPU more than 10 times [7], in other words, the advanced GPU can be equal to a small computer cluster [8]. Moreover, now the mainstream GPU's scale has exceeded significantly the CPU's. The transistor number of NVIDIA Geforce GTX 580 has been more than 30 billion [9]. From the above, we can see the powerful computational capability of the GPU. Moreover, as the programmability and parallel processing emerge [10], GPU begins being used in some non-graphics applications, which is general-purpose computing on the GPU (GPGPU). GPU is a special designed chip for computing task with intensive computing, intensive data and high parallelism, has more units for data processing, not for data cache or flow controlling, and also has high store and transmission bandwidth. In one word, it has became a high parallelism, multi-threads and multi-core processing unit [11].

As a result, we propose the parallel algorithm of Wallis transformation based on multiple graphic processing unit (multi-GPU) under Compute Unified Device Architecture (CUDA) in this paper, making use of the normal PC's multi-GPU to realize image processing algorithms' fine-grained parallel, and what is more important is that this algorithm can easily integrate with cluster to more improve efficiency.

This paper is organized as follows: The architecture of GPU and programming model of CUDA are presented in Section 3, and Wallis image enhancement algorithms are briefly reviewed in Section 4. Section 5 describes how to implement the parallel algorithm of Wallis transformation using multi-GPU. In Section 6, we present experimental results and show performance comparisons analysis in computation speed between CPU and GPU, Section 7 conclude our work.

2. Related Works

The Wallis filter [12], is a digital processing function that enhances the contrast levels of an image and it can be applied in order to optimize images for subsequent image matching [13]. Studies have found that interest operators typically find more suitable points on imagery that has been pre-processed with the Wallis filter [14]. The algorithm is adaptive and adjusts pixel brightness values in local areas only, as opposed to a global contrast filter which applies the same level of contrast throughout an entire image. The resulting image contains greater detail in both low and high level contrast regions concurrently, ensuring that good local enhancement is achieved throughout the entire image.

Chang Li [15], M.W.Sun [16], Yu Lijun [17], Lina Zhu [18] and Chunhua Zhu[19] both used the Wallis filter on the original image for image enhancement preprocessing, which enhance contrast and reduce noise for the processed images. Li zhang [20] also researched Wallis transform for image matching in CPU. At present, people's study stay only in simple applications of the Wallis transformation original algorithm, and for improving the efficiency of algorithm and system no more study.

3. GPU Description

General-purpose processing on the GPU, known as GPGPU is currently an active research area since GPUs are widely available and continue to improve in performance faster than CPUs. The capabilities of the GPU have increased dramatically in the past few years and the current generations of GPUs have higher floating point performance than the most powerful (multicore) CPUs. Many algorithms have been implemented on the GPU, and the results are often a significant speed-up over the sequential CPU implementation of the same algorithm.

3.1. Hardware Architecture

The hardware architecture is illustrated in Figure 1[21]. The device is logically composed of arrays of Single Instruction Multiple Data (SIMD) Streaming Multiprocessor (SM). Each SM contains a number of physical CUDA cores with single instruction multiple threads (SIMT) architecture that CUDA core executes the same instruction but operates on different data, at each clock cycle. The device has its own DRAM referred to device memory which has global memory, constant memory and texture memory that all can communicate with host memory. Each SM has register, shared memory, a constant cache speeding up read from constant memory and a texture cache speeding up read from texture memory four types on-chip memory.

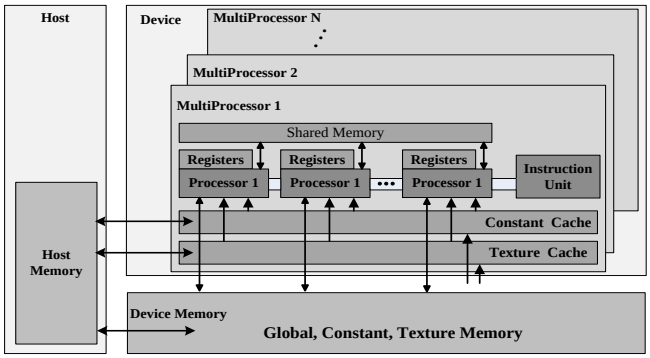

Figure 1. Hardware Model

3.2. CUDA Parallel Programming Model

The philosophical and architectural underpinning of CUDA is to create mass of thread level parallelism that can be dynamically exploited by hardware. The CUDA programming model regards GPU as a compute device which is capable of executing a high number of threads in parallel and operating as a co-processor to the host Central Processing Unit (CPU). Under CUDA's heterogeneous program model, serial code is performed on host's CPU, and parallel code is run on GPU as multi-threads mode. In other words, a portion of an application that is executed many times on different data, can be divided into a function that is executed on the device as many different threads. Such a function running on GPU is called kernel. As shown in Figure 2[22], a kernel is executed by a grid of thread blocks and a thread block is a group of 1024 threads at most that executes in parallel operating on different data based on thread IDs. The devices support thousands of threads at the same time, multiplexed on their far smaller set of cores by a dedicated hardware scheduler that avoids the overhead usually associated to context switching.

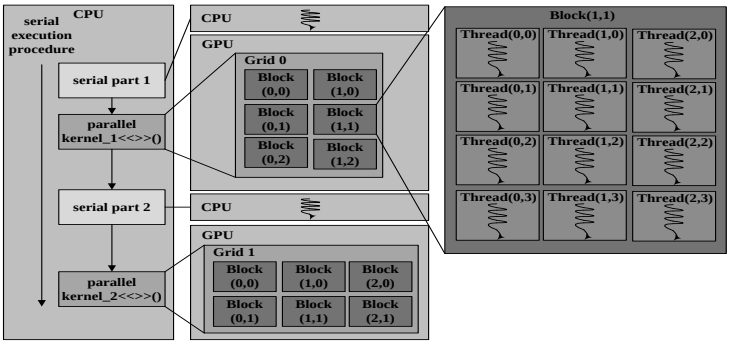

Figure 2. CUDA Programming Model

3.3. CUDA Executing Model

The executing model of CUDA is SIMT where a single instruction is executed in multiple threads. Each thread specified by the programmer is mapped to one scalar processor core by the SIMT unit of the SM. Each thread is able to be executed independently, with its own instruction address and register state, and a group of 32 threads called warps are created, managed, and scheduled by the SIMT unit. Consequently, hundreds or thousands of threads are executed simultaneously, providing the ability to hide much of the memory latency incurred on the GPU.

The executing model of CUDA is illustrated in Figure 3[23]. During execution, threads within a block are grouped into warps of 32 parallel threads, which are the granular multi-threading scheduling unit. Warps are formed from continuous sections of threads in a thread block: the first 32 threads in a block form the first warp, *etc*. A scoreboard indicates when all of a warp's operands are ready for execution. It then executes the same instruction for the 32 threads in the warp. An SM can perform zero-overhead scheduling to interleave warps and hide the latency of global memory accesses and long latency arithmetic operations. When one warp stalls, the SM can quickly switch to a ready warp resident in the SM. The SM stalls only if there are no warps with ready operands available. Scheduling freedom is high in many applications because threads in different warps are independent with the exception of explicit barrier synchronizations among threads in the same thread block.

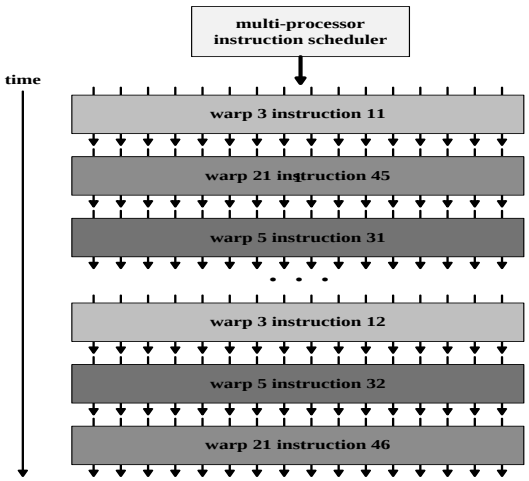

Figure 3. Scheduling Warp in SM

4. Wallis Filter and its Characteristics

Wallis filter is a special filter which enhances image contrast and suppresses noise at the same time. This filter aims at projecting image gray mean and variance to appointed values. In fact, it is a local image transform that makes gray mean and variance of different part approximate. Namely, it increases gray contrast of parts where contrast is small and decreases gray contrast contrarily. Because of smooth operator introduced when computing gray mean and variance, Wallis filter enhances useful information and suppresses noise simultaneously. The general formula of Wallis filter is:

$$f(x,y) = g(x,y)r_1 + r_0 \tag{1}$$

$$r_1 = (cs_f)/(cs_g + s_f/c); \quad r_0 = bm_f + (1-b-r_1)m_g \tag{2}$$

where $g(x,y)$ is the gray of primary image, $f(x,y)$ is the gray after Wallis transform. r_1 is a multiplicative coefficient and r_0 is an additive coefficient. If r_1>1, transformation (2) is a

high pass filter; else if $r_1<1$ transformation (2) becomes a low pass filter. m_g can be regarded as the DC component of image signal, which expresses gray scale mean of image in a certain area of a pixel. s_g can be called the AC component of image signal, which means gray scale variance of image in a certain area of a pixel. m_f is a target value of gray scale mean, and it must be a median of dynamic image range. s_f is a target value of gray scale variance, and it decides the contrast of image, generally speaking, the bigger the better. c is a contrast stretching constant of image, $c\in[0,1]$. $b\in[0,1]$ is a image brightness coefficient. When $b\rightarrow1$, the mean of image is compelled to m_f, and when $b\rightarrow0$, the mean of image is forced to m_g. In order to keep primary image gray mean, we should use smaller b [12, 14].

5. More Grain Hybrid Wallis Image Enhancement Parallel Algorithm Analysis and Design

The digital image will be divided into non-overlapping grid region and overlapping filtering region in Wallis transformation, then will be divided into non-overlapping interpolation grid region. Calculation of local gray scale mean and local gray scale variance in filtering region, calculation of multiplicative coefficients and additive coefficients in interpolation grid region are all the same with the formula between pixels, independent of the calculation process, and high parallelism. Therefore, Wallis algorithm based on the staggered grid is suitable to GPU hardware architecture that possesses lots of thread processors with parallel computing and shared memory with multi-level high-speed. For features of GPU, the digital image is divided into many computing areas, each thread block corresponds to a computational domain, and each thread corresponds to a computing grid, shown in Figure 4[24].

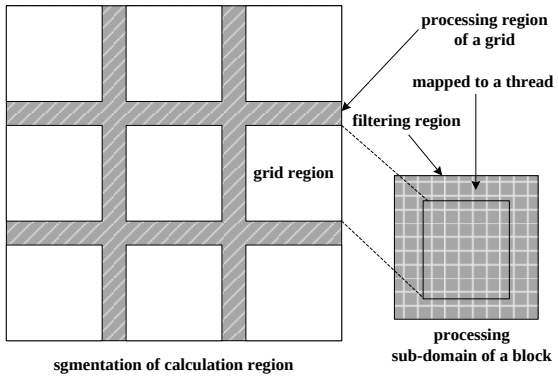

Figure 4. Schematic Diagram of Computing Domain Decomposition in Wallis Image Transforms

The Wallis transformation process based on CUDA-enabled GPU can be decomposed into the two main parts which include computing multiplicative coefficient, additive coefficient of center pixel of filtering region and computing multiplicative coefficients and additive coefficients of pixels in interpolation grid region. Since the two functions must be processed in sequence, GPU can not handle these functions once according to the character of CUDA parallel computing. The process is accomplished by two collaborative computings of CPU and GPU. It avoids the frequent data transfer between host memory and device memory, which become probably the bottleneck of the performance, just as handling these functions by GPU once.

5.1. Parallel Computing Model of Wallis Image Enhancement

The main idea of collaborative computing Wallis transformation by using multi-core CPU and multi-GPU in single computer is: taking advantage of efficient floating point processing power of GPU and good task allocation and scheduling capabilities of multi-core CPU, multiplicative coefficient computing, additive coefficient computing and interpolation computing will be designed parallel computing process for multi-GPU architecture and scheduled on multi-GPU. The operation of pre-computing and reading the file will be designed serial process and scheduled on multi-core CPU.

In the whole Wallis transformation process, the system uses three different granularities of the parallel speedup: coarse-grained parallelism on multi-GPU, middle-grained parallelism on thread block and fine-grained parallelism on thread. Logic course of Wallis filtering by multi-GPU parallel processing is shown in Figure 5. The gray value of image subdomain and the arguments to the execution configuration for each GPU, together with preprocessed the boundary information of subdomain are passed to the GPU parallel computing module, which computes multiplicative coefficients and additive coefficients of the digital image pixels, as shown within the thin frame of Figure 5. Parallel computing of multi-core CPU can be executed within the rough frame of Figure 5. Wallis transformation for each parallel subdomain is executed parallelly by multi-core CPU with parallel programming model of POSIX.

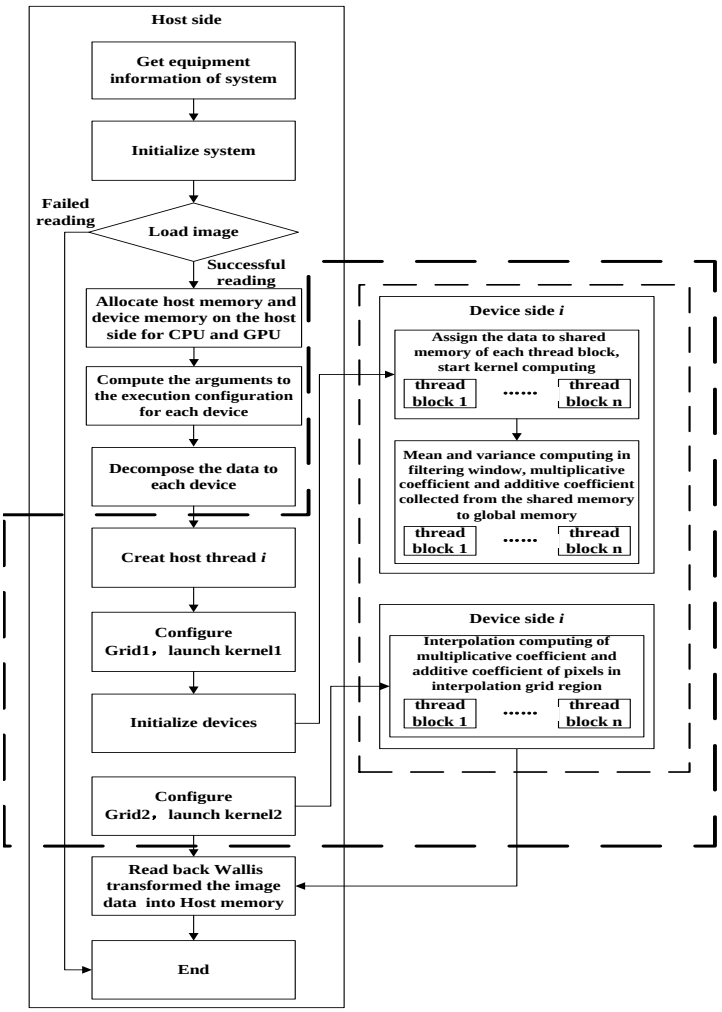

Figure 5. Flow Chart of Wallis Transformation with Multi-GPU and Multi-core CPU Collaborative Computing

Specifically, using CUDA to process Wallis filtering as follows:

(1). The digital image is divided into non-overlapping rectangular regions, the regional scale corresponding to enhance the scale of texture model. Before the system computing, CPU needs to read the original image data and initialization information.

(2). Allocating memory space in device memory by $cudaMalloc((void **), size)$, the original image that will be provided to GPU computing is copied from host memory to global memory by $cudaMemcpy(dst, src, size, cudaMemcpyHostToDevice)$.

(3). Executing kernel function. The input data are assigned to the thread blocks and threads with the configuration of kernel function. Wallis transform algorithm need use two kernel functions $wallisParameter$ and $wallis\_Interplot\_Image$ that compute severally multiplicative coefficient r_1, additive coefficient r_0 of center point in the filter window and multiplicative coefficient r_1, additive coefficient r_0 of the pixels in interpolation grid region. Original image size is $row \times col$, grid window size is $WALLIS\_gridWindowSize$, and filtering window size is $WALLIS\_filterWindowSize$ in the Wallis transform algorithm.

$$\begin{aligned} xblock &= [row - (WALLIS\_filterWindowSize - WALLIS\_gridWinowSize)]/WALLIS\_gridWindowSize \\ yblock &= [col - (WALLIS\_filterWindowSize - WALLIS\_gridWinowSize)]/WALLIS\_gridWindowSize \\ blockNumber &= xblock * yblock \end{aligned} \quad (3)$$

In equation (3), $blockNumber$ is regarded as thread block configuration of $wallisParameter$ kernel function in the Wallis parallel algorithm.

(4). Thread index built-in variables $threadIdx.x$ are regarded as marker of different thread to distinguish different pixels in kernel function $wallisParameter$. Threads in the block access synchronously the corresponding pixel location in global memory according to $threadIdx.x$, read the data from global memory into shared memory, and write result into the corresponding shared memory of the block.

(5). Determining the standard parameters (target mean and standard variance) based on obtained the data. Each block counts gray mean and standard variance of each filter window. The thread in each block calculates multiplicative coefficient r_1 and additive coefficient r_0 on the region of the original image corresponding to the block, results are assigned to center pixel in the filter window. Finally, each block will pass the results from the shared memory to global memory according to the different index value.

(6). Launching kernel function $wallis\_Interplot\_Image$. As a result of non-overlapping grid region, each thread in the block accesses synchronously multiplicative coefficient r_1 and additive coefficient r_0 of the corresponding pixel in global memory according to coordinates of the four reference interpolation points in bilinear interpolation, and then calculates multiplicative coefficient r_1, additive coefficient r_0 of the pixels in interpolation grid region and updated gray value of the corresponding pixel.

(7). Feedbacking updated image data in the global memory to host memory

By analysis of parallel character of Wallis image enhancement algorithm, Wallis parallel algorithm will implement mixed multi-granularity parallel at different levels of

parallel granularity, that is, coarse-grained parallelism, middle-grained parallelism and fine-grained parallelism.

5.2. Coarse-grained Parallel based on Multi-GPU

The digital image will be divided into several parallel subdomains in parallel process, and each GPU processes a subdomain for Wallis filter parallel algorithm. Communication and information transmission do not occur between main threads in parallel mode of GPU and CPU. As shown in Figure 6, computing on parallel subdomains are achieved by the main thread forking sub-threads in multi-core CPU. Each CPU thread transfers data, launches kernels, transfers results, and controls a device in parallel computing. Because multiplicative coefficients and additive coefficients are computed on the GPU, interpolation operation of multiplicative coefficient and additive coefficient can be also processed on the GPU for image. CUDA and POSIX are no mutual restraint. All calculations on subdomains are implemented on the GPU. CPU controls synchronization between main threads and the final output is merged by the host. It forms calculation mode of CPU-GPU-CPU.

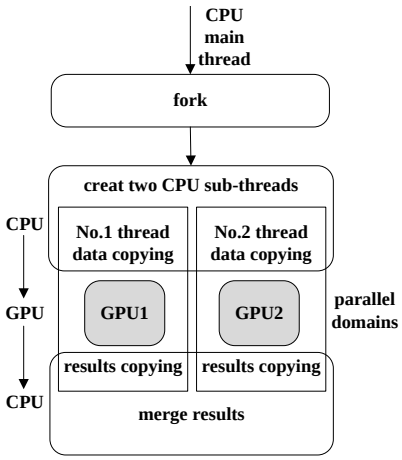

Figure 6. Schematic Diagram of Multi-core CPU and GPUs Collaborative Computing

5.3. Middle-grained Parallel based on Data Block

After the problem is divided into unrelated coarse-grained tasks, each task is divided into smaller sub-problems in parallel computing. Parallel operation within the same sub-problem can be coordinated. The task partitioning methods can implement not only communicating between the threads in sub-problem, but also make the system get the scalability. Each sub-problem can run on any CUDA core, so CUDA application system can run properly in processor with different CUDA cores.

According to non-overlapping number of grid regions, an image pixel array can be divided into $(xblock-1)\times(yblock-1)$ subdomains. Each subdomain is a basic processing unit, a large number of subdomains calculation can be processed by the thread blocks and grid. A SM can process simultaneously 8 subdomains from the data level and GeForce GTX 580 possesses 16 SMs. Therefore, GPU can process parallelly many subdomains. $(xblock-1)\times(yblock-1)$ is regarded as block value of kernel $wallis\_Interplot\_Image$, the size of grid regions $WALLIS\_gridWindowSize \times WALLIS\_gridWindowSize$ are regarded as number of threads in each block. Kernel configuration for the grid:


```
dim3 interplot_Blocks( yblock - 1 , xblock - 1 );
dim3 interplot_Threads( WALLIS_gridWindowSize , WALLIS_gridWindowSize );
wallis_Interplot_Image <<< interplot_Blocks, interplot_Threads >>> ( …… );
```

The configuration will distribute the processing task of filtering window to SMs and the processing task of each pixel to CUDA cores so as to achieve the best effect in parallel.

5.4. Fine-grained Parallel based on Threads

The arithmetic logic units are assigned as evenly as possible for the computing tasks, which improves the algorithm's parallelism is one of the important measures to raise the efficiency of GPU computing. How to load evenly computation to each thread is a major task in parallel algorithm design of Wallis filter. Thus, in order to replace cycle operation of on filter window, grid window and interpolation grid area in the CPU algorithm, the coordinates of digital image pixels can be mapped to threads. The mapping relationship between data and thread is shown in the following. Data is mapped to thread with computing the address of thread processing data through the thread ID and block ID.

```
dim3 parameter_Blocks( yblock, xblock );
dim3 parameter_Threads( WALLIS_filterWindowSize * WALLIS_filterWindowSize );
const int offset = col_share – WALLIS_filterWindowSize;
const int baseAddress = blockIdx.y * WALLIS_gridWindowSize * col_share +
                        blockIdx.x * WALLIS_gridWindowSize + tid;
const int index = blockIdx.y * yblock_share + blockIdx.x;
const int parameter_Index = blockIdx.y * yblock_share + blockIdx.x;
const int value_Index = ((WALLIS_filterWindowSize >> 1) + threadIdx.y +
                         WALLIS_gridWindowSize * blockIdx.y) * col_share +
                        (WALLIS_filterWindowSize >> 1) + threadIdx.x +
                         WALLIS_gridWindowSize * blockIdx.x;
```

Starting the number of threads per block is dynamic with changing of size of filter window in parallel algorithm of Wallis filter. Each thread block is limited up to accommodate 1024 threads in Fermi GPU at present. Therefore, parallel algorithm can take advantage of the maximum number of threads to achieve Wallis transformation effect of the greater filtering window and improve furtherly computational efficiency of parallel algorithm.

6. Experiment Results and Analysis

6.1. Testing Setup

We have implemented the Wallis parallel algorithm using CUDA and tested it with NVIDIA GeForce GTX580. The GTX580 GPU uses the latest GF110 architecture and it has 16 SMs, 512 1.544GHz processing cores, with 1.5GB on board memory. The GPU is installed on a desktop computer equipped with a 2.67GHz Intel Quad-core CPU Core 2 Quad i7, 4 GB of DDR2 DRAM running at 800 MHz. The GPU based implementation was programmed using the CUDA Programming Toolkit, version 2.3. On the CPU side, we used the Microsoft Visual Studio 2005 C++ compiler.

6.2. Experimental Results

Experimental data are used for the intercepting a regional satellite image. For comparison of multiple sets of experimental data, first the original image data pre-processed, images resolution obtain 12 sets of experimental data by reducing access, like 734×876, 1230×1388, 1564×1652, 2135×2467, 2814×3028, 3724×4126, 4893×5321, 5868×6083, 7145×7267, 8745×9258, 9165×9730 and 9465×10530.

In Figure 7, (a) is the original image with 1564×1652 image resolution, (b) is the CPU serial processing result of Wallis filtering on original image with 7×7 grid window, and (c) is the multi-GPU parallel processing result of Wallis filtering on original image with 7×7 grid window.

(a) Original Image

(b) Image Filtered based on CPU Algorithm

(c) Image Filtered based on multi-GPU Algorithm

Figure 7. Wallis Filter

For contrast experiment, respectively, running Wallis image enhancement system on CPU and multi-GPU for pre-processed 12 different image of images resolution with 3×3 grid window, and recording processing time GPU vs. CPU for each, as Table 1, CPU version of Wallis filter system has been used in DPGrid(Digital Photogrammetry Grid). For Wallis filter contrast experiment with 5868×6083 image resolution, respectively, running Wallis image enhancement system on CPU and multi-GPU for different grid window and recording processing time GPU vs. CPU for each, as Table 2. And the executed time is averaged over 1000 iterations per test.

Table 1. Comparison in Serial and Parallel Performance of Wallis Filter Algorithm with Different Image Resolution

Image resolution	Serial processing time (ms)	Parallel processing time (ms)	Speedup
734×876	344.670	152.509	2.26
1230×1388	905.323	155.287	5.83
1564×1652	1406.343	194.515	7.23
2135×2467	2857.812	261.226	10.94
2814×3028	4638.524	321.450	14.43
3724×4126	8361.713	446.196	18.74
4893×5321	14169.891	626.432	22.62
5868×6083	19465.578	785.853	24.77
7145×7267	28265.047	1058.616	26.70
8745×9258	44099.369	1553.889	28.38
9165×9730	49262.163	1695.188	29.06
9465×10530	54915.362	1873.605	29.31

Table 2. Comparison in Serial and Parallel Performance of Wallis Filter Algorithm with Different Filter Window

Grid window size	Serial processing time (ms)	Parallel processing time (ms)	Speedup
3×3	15650.343	1020.231	15.34
5×5	10009.578	402.476	24.87
7×7	8115.629	221.436	36.65
9×9	6971.845	152.657	45.67
11×11	6570.651	115.762	56.76
13×13	6642.904	98.326	67.56
15×15	6261.031	79.586	78.67
17×17	6237.634	71.328	87.45
19×19	6305.886	64.451	97.84
21×21	6042.760	56.348	107.24

6.3. Performances Analysis of Wallis Parallel Algorithm for Image Enhancement

Experimental results show that GPU can improve significantly computing speed as the increase of image resolution and grid window. For example, system of Wallis transformation can obtain 29 times speedup for image of 9465×10530 image resolution, 107 times speedup for image of 21×21 grid window size, execution speed increases 2 orders of magnitude. However, upgrading multiple of parallel processing performance is not a linear growth, but shows gently upward trend with increase of image resolution.

Figure 8 and Figure 9 show speedup curves of the Wallis parallel algorithm. Observed from Figure 8, speedup of multi-GPU accelerating the parallel algorithm of Wallis image enhancement increase considerably, and the speedup curve shows a faster rising trend as image resolution from 734×876 expanded to 4893×5321. We can see that as the image resolution increases, more and more CUDA blocks are launched, more concurrency is incurred, hence CUDA cores are able to run at full speed without interruption. It shows

that the parallel algorithm of Wallis image enhancement taking advantage of parallel computing power of multi-GPU hide effectively host starting CPU thread time, data communication latency and synchronization latency between threads from a whole, multi-GPU system highlights advantage of massive parallel computing during this period. From 4893×5321 to further scale-up in the image resolution, the speedup curve still maintains upward trend, but increasing amplitude of parallel algorithm speedup of Wallis filtering is smaller and speedup curve becomes more flat. When the image data is larger, due to bandwidth restrictions, data communication delay time between host and device increase sharply, which become a performance bottleneck, limit the overall speed of the GPU computing performance upgrade and decrease the advantages amplitude of the Wallis transformation parallel system processing a large image. Therefore, it causes increasing amplitude of system speedup slowdown trend.

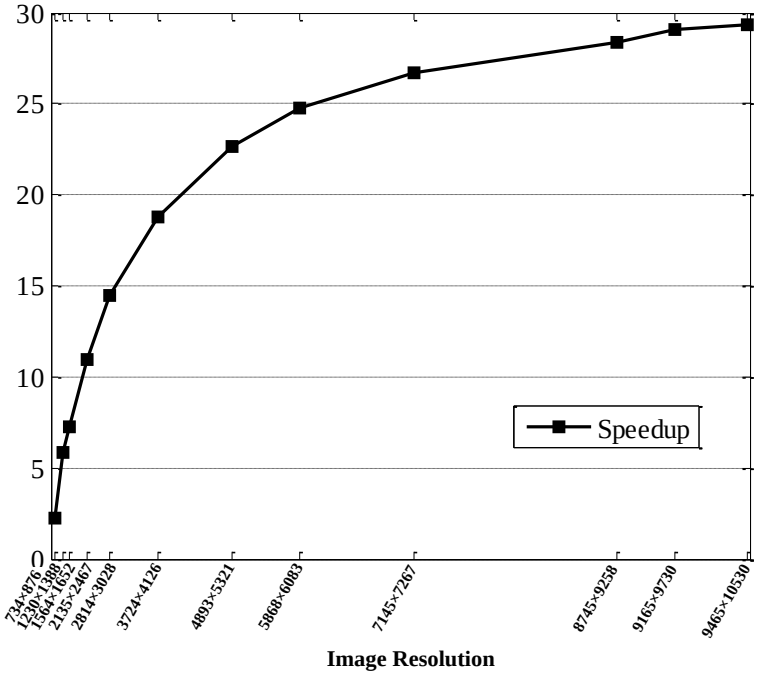

Figure 8. Accelerating Chart of Parallel Wallis Filter Algorithm with Different Image Resolution

Figure 9 shows that the speedup curve shows rising trend as size of the grid window increasing. Increasing the number of threads is included in each thread block as size of the image grid window increasing. Then, the system can run simultaneously more threads to improve the efficiency of accessing global memory and shared memory and hide the memory latency. SM create, manage and implement concurrent threads by hardware, while maintaining zero scheduling overhead. Parallel algorithm of Wallis image enhancement achieves barrier synchronization by an internal instruction_syncthreads(), the fast barrier synchronization, lightweight thread creation and zero-overhead thread scheduling are combined with effective implementation of the fine-grained data parallelism, and thus system obtains the larger speedup. Each thread block can accommodate that the maximum number of threads is 512 in the GPU. The experiment shows that the Wallis transform speedup is the greater if the number of threads created in a thread block is closer to the maximum value that can be handled by the GPU.

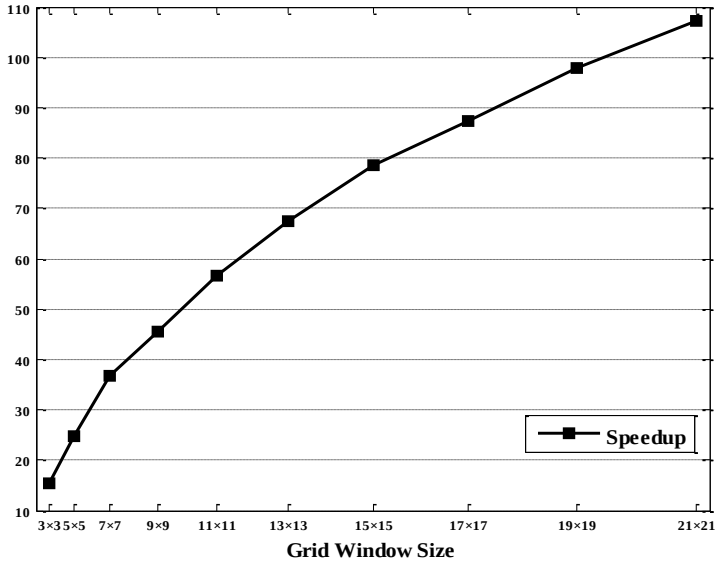

Figure 9. Accelerating Chart of parallel Wallis Filter Algorithm with Different Filter Window

7. Conclusion and Future Works

In this paper, a parallel image processing algorithms, Wallis filter, is presented and implemented on multi-GPU, and compared with the sequential implementations based on CPU. Performance results indicate that significant speedup can be achieved, and the speedup increases with image resolution and size of grid window increasing. The Wallis filter can get speedup of up to 107 times, compared to CPU-based implementations. Obviously, GPU provides a novel and efficient acceleration technique for image processing, and is cheaper in hardware implementation. Future work will involve mapping more complex image processing algorithms into GPU, and a deeper analysis of parallelization strategies to make best of computing resources provided by multi-GPU.

Acknowledgments

This work is supported by the National Basic Research Program of China (973 Program, 2012CB719900); the China Postdoctoral Science Foundation(2012M510110); the Science and Technique Foundation of Henan Province of China(132102310003); the Science and Technique Foundation of Zhengzhou City of China(131PPTGG419-2); the Educational Commission Research Key Foundation of Henan Province of China(13A520354).

References

[1] C. S. Fraser Jazayeri, “Interest operators in close-range object reconstruction”, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing XXXVII(Part B5), **(2008)**, pp. 69-74.
[2] S. Sergio, P. Abel and M. Gabriel, “Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units”, Concurrency and Computation-Practice & Experience, vol. 23, 13, **(2011)**, pp. 1538-1557.
[3] C. Zhaoxi, M. Xiaohong and G. Lianghui, “GICUDA: A parallel program for 3D correlation imaging of large scale gravity and gravity gradiometry data on graphics processing units with CUDA”, Computers & Geosciences, vol. 46, **(2012)**, pp. 119-128.
[4] I. K. Park, N. Singhal and M. H. Lee, “Design and Performance Evaluation of Image Processing Algorithms on GPUs”, IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 1, **(2011)**, pp. 91-104.
[5] S. A. Bibikov, V. A. Fursov and A. V. Nikonorov, “Memory access optimization in recurrent image

processing algorithms with CUDA", Pattern Recognition and Image Analysis, vol. 21, no. 3, **(2011)**, pp. 377-380.
[6] M. Dorgham Osama, D. Laycock Stephen and H. Fisher Mark, "GPU accelerated generation of digitally reconstructed radiographs for 2-D/3-D image registration", IEEE Transactions on Biomedical Engineering, vol. 59, no. 9, **(2012)**, pp. 2594-2603.
[7] O. Eric1, J. de la Calleja and B. Antonio, "Point to point processing of digital images using parallel computing", International Journal of Computer Science Issues, vol. 9, no. 3, **(2012)**, pp. 1-10.
[8] F. Joaquín, B. Gregorio and F. Juan, "The 2D wavelet transform on emerging architectures: GPUs and multicores", Journal of Real-Time Image Processing, vol. 7, no. 3, **(2012)**, pp. 145-152.
[9] C. Raúl1, A. S. Montemayor and J. J. Pantrigo, "High performance memetic algorithm particle filter for multiple object tracking on modern GPUs", Soft Computing, vol. 16, no. 2, **(2012)**, pp. 217-230.
[10] J. Radovan, A. Jukka and H. Jirí, "Real-time PCA calculation for spectral imaging (using SIMD and GP-GPU)", Journal of Real-Time Image Processing, vol. 7, no. 2, **(2012)**, pp. 95-103.
[11] S. Haigang, P. Feifei and X. Chuan, "GPU-accelerated MRF segmentation algorithm for SAR images", Computers & Geosciences, vol. 43, **(2012)**, pp. 159-166.
[12] K. F. Wallis, "Seasonal adjustment and relations between variables", Journal of the American Statistical Association, vol. 69, no. 345, **(1976)**, pp. 18-31.
[13] E. P. Baltsavias, "Multliphoto geometrically constrained matching", America: Institute of Geodesy and Photogrammetry, ETH Zurich, **(1991)**, pp. 49-221.
[14] F. Remondino, "Image-based modelling for object and human reconstruction", America: Institute of Geodesy and Photogrammetry, ETH Zurich, **(2006)**, pp. 91-174.
[15] C. Li, H. Wu and M. Hu, "A novel method of straight-line extraction based on wallis filtering for the close-range building", Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics, **(2009)**, pp. 290-293.
[16] M. W. Sun and J. Q. Zhang, "Dodging research for digital aerial images", The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, vol. XXXVII, Part B4, **(2008)**, pp. 349-353.
[17] Y. Lijun, N. Yueping and Z. Yan, "Remote sensing change detection study of the grand canal and environs-A case study in the yangzhou section, JiangSu, CHINA", The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, vol. XXXVII, Part B7, **(2008)**, pp. 1591-1594.
[18] L. Zhu, J. Zhang and L. Pa, "River change detection based on remote sensing image and vector", Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, IMSCCS'06, vol. 1, **(2006)**, pp. 188-191.
[19] C. Zhu and Y. Nie, "Study on the effects of Grand Canal on city pattern change of Hangzhou based on Remote Sensing", Urban Remote Sensing Event, vol. 69, no. 345, **(2009)**, pp. 1-6.
[20] Z. Li, Z. Zuxun and Z. Jianqing, "The Image Matching Based on WalIis Filtering", Journal Of WuHan Technical University Of Surveying and Mapping, vol. 1, **(1999)**, pp. 24-27.
[21] P. B. Noël, A. M. Walczak and J. Xu, "GPU-based cone beam computed tomography", Computer Methods and Programs in Biomedicine, vol. 98, no. 3, **(2010)**, pp. 271-277.
[22] O. Kutter, R. Shams and N. Navab, "Visualization and GPU-accelerated simulation of medical ultrasound from CT images", Computer Methods and Programs in Biomedicine, vol. 94, no. 3, **(2009)**, pp. 250-266.
[23] E. Liria, D. Higuero and M. Abella, "Exploiting parallelism in a X-ray tomography reconstruction algorithm on hybrid multi-GPU and multi-core platforms", Proceedings of the 2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA, **(2012)**, pp. 867-868.
[24] X. Han, W. Qing-Shuang and F. Na, "Fast Wallis image enhancement algorithm with CUDA", Shenyang Gongye Daxue Xuebao, vol. 33, no. 3, **(2011)**, pp. 293-298.

Authors

Xiao Han, he received his PhD degree in photogrammetry and remote sensing from the School of Remote Sensing and Information Engineering, Wuhan University, China in 2011. Professor in the School of Information Science and Technology, Zhengzhou Normal University. His main research interests include photogrammetry and remote sensing, parallel computing.

Yu-Pu SONG, she Received her MD degree in software engineering from the School of Software, Beijing University of Technology, China in 2008. Associate professor in the Shangqiu Vocational & Technical College. Her main research interests include Software Engineering and parallel computing.

ZHOU Qinglei, he received his PhD degree in software and theory from the Xi'an Jiaotong University, China in 2002. Professor in the School of Information Engineering, Zhengzhou University. His main research interests include performance analysis and optimization on parallel programs.

