Send Orders for Reprints to reprints@benthamscience.ae

Image Based Virtual Dimension Compute Unified Device Architecture of Parallel Processing Technology

Baozhong Liu* and Xin Xu

Chongqing College of Electronic Engineering, Chongqing, China

Abstract: There are a number of virtual dimension typical targets in hyperspectral image. Determining the virtual dimension is the first step in many applications of hyperspectral image. In view of the virtual dimension calculation method of having high time complexity problem, according to the calculation of highly parallel features, in this paper graphics processing unit (GPU) using the Compute Unified Device Architecture (CUDA) and its extended linear algebraic toolbox of CULA and Thrust are studied, to realize virtual dimension calculation. The parallel realization of the algorithm in each step was further optimized to obtain greater acceleration performance. Through the function of CUDA on GPU parallel computing and CPU non parallel calculation, and experimental verification of the virtual dimension, it was fund that the CUDA parallel computing can significantly speed up the implementation of the algorithm.

Keywords: Compute unified device architecture, graphics processing unit, harsanyifarrand-chang, virtual dimension, hyperspectral images.

1. INTRODUCTION

Because of the complexity and variety of remote sensor spatial resolution limit and the nature of objects, mixed pixels exist in the remote sensing image. In order to improve the precision of remote sensing application, we must solve the problem of the decomposition of mixed pixels. To determine the number of endmembers is a prerequisite for the decomposition of mixed pixels [1, 2]. A virtual dimension method was proposed by Chang *et al*. (Virtual Dimensionality, VD method) (2) which is used to determine the number of endmembers at present. Given the huge amounts of data of hyperspectral image, using traditional methods can't meet the real-time application need of hyperspectral image. Analysis of VD algorithm found that its parallelism is very strong, and thus, very suitable for parallel processing. From the development of high performance computing in remote sensing (High Performance Computing, HPC) to the research implementation has been used to accelerate the processing of hyperspectral data parallel computing [3, 4]. The literature [5] proposes meter high spectral unmixing based on the GPU system. The endmember extraction algorithm HFC is based on the mixed programming of CPU and GPU to achieve non-parallel and parallel computing respectively. Only in finding solution from the correlation matrix to covariance matrix change, the parallel library CUDA provides parallel but no real implementation of HFC. In this paper, the pure GPU version of the HFC algorithm, compiled with the CUDA Based on NVIDIA C code, and the optimized operation further speedup 46.07 times in the NVIDIA GeForce GTX 660Ti device.

*Address correspondence to this author at the Chongqing College of Electronic Engineering, Chongqing, China; E-mail: jianqiu27@163.com

2. THE CUDA C PROGRAM

The computing industry is moving from the "central processing using only CPU to CPU and GPU and the development of cooperative processing". For the creation of the new computing paradigm (Ying Weida, NVIDIA ® it is the invention of the CUDA (Compute) Unified Device Architecture, the Compute Unified Device Architecture), this programming model in the application is to make full use of the advantages of CPU and GPU respectively. Now, the framework has been applied to the GeForce&reg (sperm IT), ION (Yiyang IT), Quadro and Tesla GPU (graphics processing unit), for application developers; this is a huge market.

In the consumer market, almost every important consumer video applications have to accelerate the use of CUDA or soon will use CUDA to speed up, including Elemental Technologies Company, MotionDSP Company and LoiLo company's products.

In the research community, CUDA has always been Repeng. For example, CUDA is now able to use AMBER for accelerated drug research. AMBER is a molecular dynamics simulation program; the world has more than 60000 researchers using this program to accelerate drug research in academia and the pharmaceutical companies.

In the financial markets, Numerix and CompatibL are two new counterparty risk applications released to support CUDA and achieve 18 times faster computing. Numerix is widely used in nearly 400 financial institutions.

The wide application of CUDA gives rise to GPU computing *i.e.* Tesla GPU. The global fortune five hundred companies now have installed more than 700 GPU clusters. These enterprises are involved in various fields, such as the energy field of the J Len Bbe Semyon and Chevron and Bank of Paris.

1874-4443/15

With the Microsoft Windows 7 and Apple's Snow Leopard operating system, GPU computing will become a mainstream. In the new operating system, GPU will not only be the graphics processor, it will also become a common to parallel processor that can use all applications.

CUDA is a maker of graphics chips for NVIDIA development, on the GPU development platform for general computing purpose [6]. NVIDIA provides a simple and rapid method used in the preparation of CUDA based on GPU code, which is a kind of advanced language based on C: C for CUDA, referred to as CUDA C; It is NVIDIA extensions and restrictions of C language, that support the majority of C language instruction and grammar [7].

A CUDA C program usually consists of two parts: one part in the host (CPU) for the order of execution, another part of the equipment (GPU) starts thousands of threads executed in parallel. NVIDIA Company developed the C compiler (NVCC) during compilation to distinguish these threads. The device is mainly based on some kernel function, part of which is used to complete the whole process; each kernel function will usually generate a large number of threads to use parallel data [8]. Usually a large project will start with a number of kernel functions in different stages of the completion of the corresponding calculation. If a kernel does not have the data dependency for starting its work, the program can only start sequencing after the completion of the execution of all kernels so that all the results will be returned to the host. A grid sequence generates each of which starting a kernel function (grid). The grid will be organized into the thread block (block), the same grid block contains the same number of threads. The grid and the dimensions of the block are set through the configuration parameters.

3. VIRTUAL DIMENSION CALCULATION METHOD

The characteristics of hyperspectral image covariance matrix $K_{L\times L}$ and $R_{L\times L}$ autocorrelation matrix set of values $\lambda_1 \geq \lambda_2 \geq ...\lambda_L$ and $\{\overline{\lambda}_1 \geq \overline{\lambda}_2 \geq ...\overline{\lambda}_L\}$ respectively, where L denotes the number of bands of hyperspectral image. To determine the number of problem endmembers, a dualistic problem testing hypothesis can be formulated as:

$$H_0 : z_1 = \overline{\lambda}_1 - \lambda_1 = 0, H_1 : z_1 = \overline{\lambda}_1 - \lambda_1 > 0 \quad (1)$$

If H_1 is true, in addition to the noise and the signal source (endmembers in hyperspectral images), it suggests feature related matrix. The difference between the $\overline{\lambda}_1$ and λ_1 as H_0 and H_1 is based on the conditional probability, conditional probability density function corresponding to:

$$p_0(z_l) = p(z_l | H_0) \cong N(0, \sigma_{zl}^2), p_1(z_l) = p(z_l | H_1) \cong N(\mu_l, \sigma_{zl}^2) \quad l = 1,...,L \quad (2)$$

In the formula, μ_l is unknown. When the image of sample N is large enough, the variance is $\sigma_{zl}^2 = 2\overline{\lambda}_1^2 / N + 2\lambda_1^2 / N$. The definition of false alarm rate formula for $P_F = \int_{\tau_1}^{\infty} P_0(z)dz$ and the detection rate of $P_D = \int_{\tau_1}^{\infty} P_1(z)dz$.

The VD algorithm is described as follows:

1) Calculate the covariance matrix of the image data of $K_{L\times L}$ and $R_{L\times L}$ correlation matrix;

2) calculate the covariance matrix and correlation matrix of the value sets, denoted as $\{\lambda_1 \geq \lambda_2 \geq ...\lambda_L\}$ and $\{\overline{\lambda}_1 \geq \overline{\lambda}_2 \geq ...\overline{\lambda}_L\}$, where L is the number of spectral bands;

3) Approximation in obtaining spectral image noise variance in 1 Band $\sigma_{z1}^2 = 2\overline{\lambda}_1^2 / N + 2\lambda_1^2 / N$, where N denotes the number of elements in the image;

4) calculate the probability density function $p_0(z_1) = \frac{1}{\sqrt{2\pi}\sigma_{z1}} e^{-\frac{z_1^2}{2\sigma_{z1}^2}}$

5) The false alarm probability is given according to the P_F, the false alarm rate and the detection rate is defined to obtain the τ_1 value type;

6) Based on the Neyman-Pearson theory, as the $\overline{\lambda}_1 - \lambda_1 > \tau_1$ indicates a signal source; the L band is used to judge such a detection, which finally obtains the total signal source as the value of VD.

4. PARALLEL IMPLEMENTATION OF CUDA BASED ON HFC

The realization process is shown in Fig. (**1**) in parallel CUDA based on HFC.

The host is responsible for the original hyperspectral data input to the device and removing the calculation results from the device, whereas calculation of the whole process occurred in the terminal equipment. This algorithm defines 4 kernel equipments in the end. Because each kernel function is data dependent so the serial mode in order to start the realization process, is as follows:

1) For first kernel to complete the correlation matrix and the covariance matrix of the basic linear algebra, using the CUDA development kit that provides a set of procedures (CUDA Basic Linear Algebra Subprograms, CUBLAS) cublas Sgemm function in [9];

2) Results obtained from the first kernel will be used as input for the second kernel to find the characteristics of the two matrix, which were used in the other function based on CUDA to realize the extension of

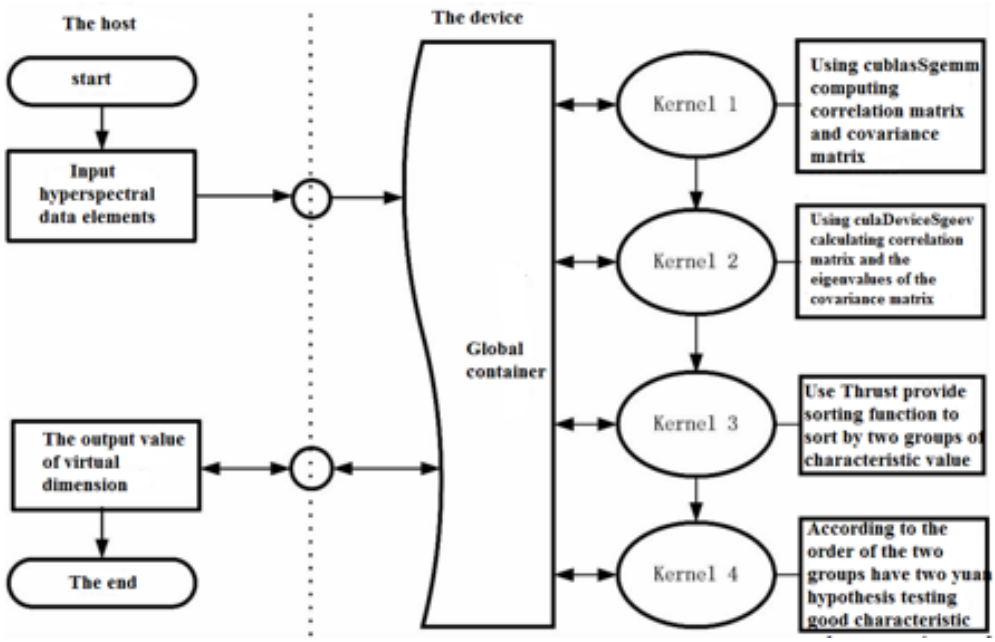

Fig. (1). HFC algorithm flow chart based on CUDA architecture.

linear algebra tool box (CUDA Linear, Algebra, CULA) CUDA Device Sgeev [10] for calculating the characteristic value;

3) Next to the correlation matrix value and covariance matrix eigenvalue sorting, is the use of the sort function (CUDA development tool in the Thrust package) [11], and this function can be transmitted into the CPU operation according to the attribute parameters (parameters of the allocation of memory on the host side) or GPU (end of operation equipment in the parameters of the distribution);

4) It's because of the above 3 kernels are correct and effective, and function of packaging is highly optimized; so, it can guarantee the results. Finally, a kernel VD is calculated; *e.g.* if the number of eigenvalues is less than 512 then just open a block, parallel with each thread.

Finally, compare, and then use the API atomic Add function to meet the conditions to accumulate the results.

5. EXPERIMENTAL RESULTS AND ANALYSIS

The HFC algorithms based on CUDA and C standard were used in a 350 * 400 * 50 cuprite hyperspectral image and a 120 x 250 x 124 crops of hyperspectral image, Figs. (**2**, & **3**) respectively, show cuprite and crop hyperspectral image false color synthesis.

The test platform is Windows-7 32 Intel core i5-3580p and NVIDIA GTX 660Ti and CUDA 5.5 development kits, finally achieved the expected value of VD on two pieces of hyperspectral images. The experimental results are shown in Table **1**, we can see that the HFC algorithm based on GPU compared with the standard C HFC algorithm to deal with two different hyperspectral images, have different degrees of speedup. Because both the HFC algorithm based on the

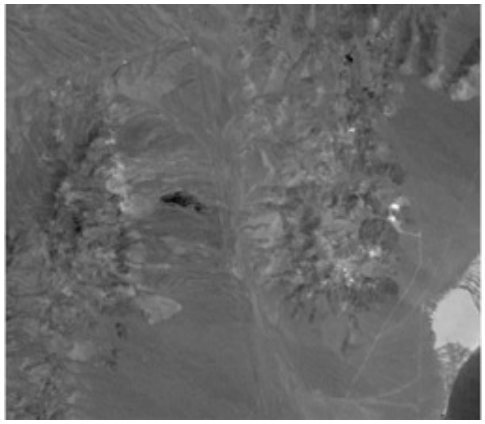

Fig. (2). Cuprite hyperspectral false color composite image.

Fig. (3). Crop hyperspectral false color composite image.

Table 1. The experimental results of two hyperspectral images.

Graphics Category	The CPU Run Time Running Time/s	The CPU Run Time Running Time/s	Acceleration Ratio
Cuprite	5.43	0.198	27.42
Crop figure	6.45	0.14	46.07

Table 2. Parts of the CUDA C implementation algorithm of HFC time-consuming.

Graphics Category	Algorithm Phase				
	The Covariance Matrix and Correlation Matrix	To Solve the Eigenvalue	Sorting Characteristic Value	Binary Hypothesis Test	The Data Transfer
Cuprite	74.22 ms	5.85 ms	1.187 ms	0.010 ms	16.72 ms
Crop figure	42.10 ms	85.24 ms	2.038 ms	0.011 ms	10.61 ms

Table 3. Two parallel HFC algorithm calculating the same in all parts of hyperspectral image time-consuming.

The Algorithm Contrast	Algorithm Phase				
	The Covariance Matrix and Correlation Matrix	To Solve the Eigenvalue	Sorting Characteristic Value	Binary Hypothesis Test	The Data Transfer
Algorithm in this paper	167.76 ms	121.47ms	1.272 ms	0.011 ms	44.487 ms
Algorithm in the literature [5]	246.00 ms	–	–	–	–

standard C, and HFC version of the GPU algorithm based on the first need to read data, pre reading data, and the consumed time is the same, the following algorithm gives the total speedup time respectively 27.42 and 46.07; time of each part of the HFC algorithm, CUDA and the C are shown in Table **2**. We can find that the Cuprite calculates the correlation matrix and the covariance matrix of the longest, and crop map calculates the characteristic value of the most time-consuming operation, which is related to the size dimension of hyperspectral image transmission. Also the two image data processing also occupied a considerable amount of time, to help further analyze and optimize the procedures and eliminate performance bottlenecks of these data analyses. Two kinds of parallel HFC algorithms with a high spectral image of each part's time, are shown in Table **3**. Table **3** lists the comparison of each part of the algorithm for solving the consumption of virtual dimension parallel algorithm proposed in this paper and given in ref. [5] with a 350*350 *188 hyperspectral image data.

Because in this algorithm each step is done in parallel at the device end [5], only the first step solves the HFC parallel implementation, subsequent executions are host serial implementations. So in the Table **3**, there is no specific time given for each step of operation.

CONCLUSION

By using the method of parallel technology, the traditional hyperspectral data processing is modified to accelerate data processing to satisfy a real-time hyperspectral application trend. This paper discusses the technology based on the CUDA framework to complete the parallel implementation of a virtual dimension algorithm, and CPU algorithm based on the comparison of serial. The experimental data show that, for large data computing intensive operation, the parallel processing of GPU is compared with CPU serial processing in the speed that will accelerate orders of magnitude. The program design of GPU involves the communication of CPU and GPU, initialization of fixed time overhead, and calculation precision of intermediate results, and when designing the algorithm, should be fully considered.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the Chongqing Education Commission scientific research topic "Research on landslide monitoring and early warning system based on Wireless Sensor Network" the project number: kj132204.

REFERENCES

[1] B. Luo, J. Chanussot, S. Douté, and L. Zhang, "Empirical automatic estimation of the number of endmembers in hyperspectral images," *Geoscience and Remote Sensing Letters*, *IEEE,* vol. 10, pp. 24-28, 2013.

[2] C.-I. Chang, and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, pp. 608-619, 2004.

[3] Z. HaiJun, C. ShengBo, Z. XuQing, and W. YaNan, "GPU-Based denoising to remotely sensing images," *Urban Geotechnical Investigation & Surveying,* vol. 2, p. 034, 2010.

[4] Y. Luo, K. Guo, and S. Zhao, "Minimum noise fraction of hyperspectral remote sensing in parallel computing based on GPU," *Journal of Sichuan Normal University (Natural Science),* vol. 3, p. 036, 2013.

[5] S. Bernabe, S. Sanchez, A. Plaza, S. Lopez, J. A. Benediktsson, and R. Sarmiento, "Hyperspectral unmixing on GPUs and multi-core processors: a comparison," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 6, pp. 1386-1398, 2013.

[6] C. Nvidia, *NVIDIA Compute Unified Device Architecture Programming Guide Version 2.0*, NVIDIA Corporation, Santa Clara, CA, USA, 2010, pp. 10-12.

[7] D. Qiu, *GPGPU: The Art of Acceleration*, Mechanical Industry Press: Beijing, 2012, p.111.

[8] D. B. Kirk, and W. H. Wen-mei, *Programming Massively Parallel Processors,* Tsinghua University Press, Beijing, 2010, pp. 32-78.

[9] C. Nvidia, *Cublas Library User Guide,* NVIDIA: Santa Clara, 2013, pp. 67-69.

[10] C. Nvidia, *CULA Reference Manual*, NVIDIA: Santa Clara, 2012, pp. 17-20.

[11] C. Nvidia, *Thrust Quick Start Guide*, NVIDIA: Santa Clara, 2013, p. 11.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014