Implementation of Spectral Angle Mapper (SAM) Algorithm on a Graphic processing unit (GPU)

Balaji Vengatesh M.
Information Technology, SRM UNIVERSITY, Chennai, India
balajivengatesh1991@gmail.com

Abstract— **The Need for Hyper spectral Images for Exploration of Oil and Other Minerals are so massive. We can tap the high computational power available now for faster tracking of those minerals underneath. In this paper, we Implement an Algorithm called Spectral angle mapper(SAM) using compute unified device architecture(CUDA) framework on a GPU. The SAM algorithm is fit for parallel and distributed computing, but we use a Graphic processing unit to implement it in parallel. This paper studied the balance between resource acquirement of each thread and the number of active blocks, and the impact of computational complexity on speedup. We also Improved the SAM algorithm to use several training samples instead of one. At the end of this paper we show quantitative results of comparison on speedup with the earlier use of ENVI, the only software for the analysis of hyper spectral images with our latest implementation on CUDA.**

***Index Terms*—ENVI, CUDA**

I. INTRODUCTION

In recent years several efforts were directed towards incorporating high performance computing in remote sensing missions. A relevant example of the use of HPC techniques (such as parallel and distributed computing) is hyper spectral remote sensing, in which an imaging spectrometer collects hundreds or even thousands of measurements (at multiple wavelength channels) for the same area on the surface of the earth [1]–[3]. Antonio *et al.* Researchers have been done already on this with other specialized hardware devices such as FPGA and other GPUs. Algorithms such as pixel purity index(PPI),Principal Component analysis(PCA) were already implemented on a GPU with Compute unified device architecture(CUDA) and achieved significant speedups [1]. Yang *et al.* [7], [8] implemented SAM algorithm based on CUDA using a specific pixel value as reference spectra without transferring image data asynchronously. Lu and Park *et al.* [9]–[11] researched the key factors in design and evaluation of image processing algorithms on the massive parallel GPU using CUDA and proposed metrics to show the suitability in predicting the effectiveness of an application for parallel implementation. Now the GPU architecture with low cost, low power dissipation is gathering more attention than onboard real time analysis of remote sensing data.

SAM algorithm is a classical identification or classification method for hyper spectral data, which is suitable for parallel and distributed computing without any mutual influences between different pixels in computing spectral angle [8].

A plurality of spectral angles between spectrum of each pixel and training samples of the same class are calculated in the first. If the maximum spectral angle is larger than a certain threshold, this pixel is regarded as a target pixel. In the actual operation, we use morphological erosion when extracting training samples so that the mixed pixels are omitted. Similarly, morphological opening operation is performed after recognition to eliminate noise. Here, it is taken as an example to illustrate the parallel implementation based on GPU and distributed clusters.

II. KEY TECHNIQUE

In this section we introduce spectral angle mapper algorithm , the mathematical implementation and finally the implementation in CUDA.

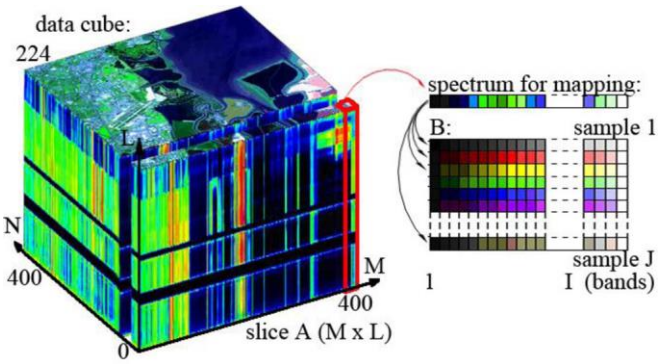


Fig.1. A hyper spectral image pixel and reference spectra

A. SAM Algorithm for Target Detection

SAM algorithm is an identification method that maps the target on the image by calculating the spectral similarity between the image spectra and the reflectance spectra. Reflectance spectra are readily available or it should be taken from the image itself or extracted from the laboratory setting. The spectral similarly is calculated by calculating the angle between the two spectra by considering the vectors in n-dimensional space. This method is rapid to identify the targets on any given hyper spectral image. It is one of the robust techniques because it restrains the influence of shading effects to accentuate the influence of target reflectance characteristics.

B. Sequential Implementation of SAM

Spectral angle mapping is the most time consuming part of the entire algorithm. So we apply parallel methods to accelerate this

The Original serial implementation is as follows

1) Record all bands of values through triple loop for each spectrum
2) Calculate spectral angle between the current pixel and the reference spectra
3) Combine computational results

From Fig.1 the source hyper spectral data is shown as a three dimensional matrix data (MxNxL M lines of source

data,N columns of source data, L bands of source data) Matrix A (MxL) comes from a slice of source data and Matrix B stands for training samples coming from source data (IxJ, I:bands of reference spectra, J:number of samples). The matching process will be conducted by using the following formula:

$$\alpha = \cos^{-1}\left[\frac{\sum_{i=1}^{nb} t_i r_i}{\sqrt{\left(\sum_{i=1}^{nb} t_i^2\right)}\sqrt{\left(\sum_{i=1}^{nb} r_i^2\right)}}\right] \quad (1)$$

Where nb represents the number of bands

Algorithm:
For i ROW= 1~N;
For i Column=1~N
Spectra for compare ← Spectrum of pixel(iRow,iColumn)

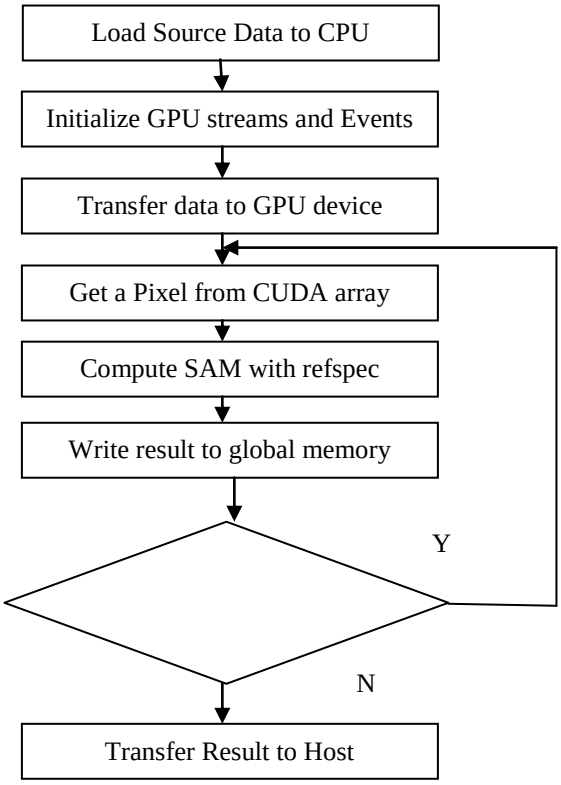
Calculate α between spectra for compare and each training sample by (1)
Find max α
Compare to threshold
Record the result
End
End

C. *Parallel implementation of SAM*

As the main goal of this paper lies in parallel implementation to achieve speedup, I describe the implementation method using CUDA. The implementation can be clearly explained with a flow chart. But before that we have created a step by step procedure of the process as follows

1) Load the image data into memory
2) Initialize the GPU device
3) Create Streams and Initialization
4) Create events ,i.e. monitor the device progress and record exact execution time
5) Load m kernels corresponding to the lines of source data
6) Run the kernels with the image for process.

As the main aim of our implementation is only to achieve speedup. We rely on the SIMD devices for performance. GPU's have massive amount of threads more than a CPU to process data. Every pixel gets processed in a different thread, according to our implementation. The stream function of CUDA platform allows us to occupy both the computational engine and the data transfer engine simultaneously so the whole task is divided into several pieces each for a single stream. Within the stream, all the steps are executed sequentially, but the executing sequence between streams are generally random. While one stream is computing, another can begin its data transfer step at the same time. So that the data transfer latency can be covered.



III. EXPERIMENTALRESULTS

A. *Experimental Setup*

1) *Experimental Data:* The source hyper spectral Data (400: lines of source data, 400: columns of source data, 224: bands of source data) is a low altitude AVIRIS image whose resolution is3.5 m and radiation wavelengths is from 0.4–2.5 m. The truth map is extracted from the original image.
2) *Software Environment:* Operating System: Windows 7
3) Development platform: CUDA 4.0
4) *Hardware Environment:* GPU: NVIDIA GTX 560, INTEL CORE I5 3.2GHZ PROCESSOR

B. *Experimental Results*

Different parallel architectures are fit for different application environments. Even though under the same parallel architecture, different speed-up ratios can be achieved by using different parameters [16]. In order to obtain comparable data, several group experiments have been conducted. The results are given

TABLE I
COMPUTATIONAL TIME FOR CPU AND GPU(MS)

Item	A	B	C	D	E	F	G
$Time_1$	400.90	29.44	22.99	29.46	22.96	---	---
$Time_2$	400.90	99.62	93.21	81.70	75.08	54.87	34.70

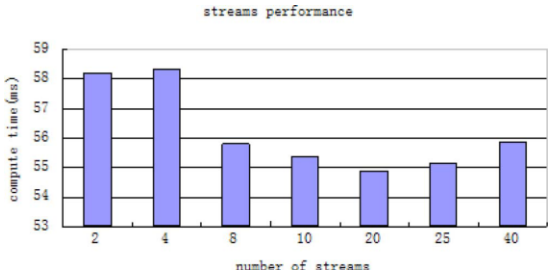


Fig.2.Influence of Streams

To make the efficient of page-locked memory clear, we set the number of training sample as 1. The experimental results are given in Table II.

Time 1: time needed for SAM computation;
Time 2 : total time (for GPU: computation times add data transmission time between host-memory and device-memory);

A: CPU running time based on C code
B–G: GPU running time
B: program in which thread assignment is not optimized
C: after optimization of thread assignment
D: page-locked memory is introduced
E: page-locked memory is introduced and threads assignment Optimized
F: stream and page-locked memory are all introduced and thread assignment optimized
G: the executing time of F on Tesla S2050

It is obvious to draw the conclusion from the chart above that the thread allocation and the CPU and GPU asynchronous collaborative work mode have an efficient impact on the performance of the program. Due to the increased data transfer bandwidth and number of computing cores, Tesla reaches a higher acceleration performance than GTX 560.

The experiments in this section demonstrate that the stream-based algorithm in the CUDA platform accelerates SAM process the most, yielding an optimized executing time of 54.87 ms. Thread assignment and stream number impact speedup a lot. Graphics devices are more suited to compute-intensive applications.

REFERENCES

[1] A. Plaza, Q. Du, and Y.-L. Chang, "High performance computing for hyperspectral remote sensing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 529–543, Sep. 2011.

[2] C. A. Lee *et al.*, "Recent developments in high performance computing for remote sensing: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 508–527, Sep. 2011.

[3] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, pp. S110–S122, 2009.

[4] A. Plaza *et al.*, "Improving the performance of hyperspectral image and signal processing algorithms using parallel, distributed and specialized hardware-based systems," *J. Signal Process. Syst.*, vol. 61, pp. 293–315, 2010.

[5] A. Plaza *et al.*, "An experimental comparison of parallel algorithms for hyperspectral analysis using heterogeneous and homogeneous networks of workstations," *Parallel Computing*, vol. 34, pp. 92–114, 2008.

[6] A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*. Boca Raton, FL, USA: Taylor & Francis, 2007.

[7] J. Yang, Y. Zhang, and G. Dong, "Investigation of parallel method of RS image SAM algorithmic based on GPU," *Science of Surveying and Mapping*, vol. 35, no. 3, pp. 9–11, May 2010.

[8] X. Liu and Y. Kang, "Algorithm of spectrial angle parallel classification on remote sensing image," *Computer Science*, vol. 36, no. 9, pp. 267–269, Sep. 2009.

[9] F. Lu and J. Song, "Survey of CPU/GPU synergetic parallel computing," *Computer Science*, vol. 38, no. 3, pp. 5–9, Mar. 2011.

[10] I. K. Park *et al.*, "Design and performance evaluation of image processing algorithms on GPUs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, pp. 91–104, Jan. 2011.

[11] D. Castaño-Díez *et al.*, "Performance evaluation of image processing algorithms on the GPU," *J. Structural Biology*, vol. 164, pp. 153–160, Oct. 2008.

[12] E. Christophe, J. Michel, and J. Inglada, "Remote sensing processing: From multicore to GPU," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 643–652, Sep. 2011.

[13] W. Fang, Q. Luo, and N. K. Govindaraju, "Mars: AcceleratingMapReduce with graphics processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 4, pp. 608–620, Apr. 2011.

[14] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General- Purpose GPU Programming*. Boston, MA, USA: Pearson Education, 2010.

[15] D. B. Kirk andW.-M.W. Hwu, *Programming Massively Parallel Processor: A Hands-On Approach*. NewYork,NY,USA:ElsevierScience & Technology, 2010.

[16] G. M. Striemer and A. Akoglu, "Sequence alignment with GPU: Performance and design challenges," in *IEEE Int. Symp. Parallel & Distributed Processing*, May 2009, pp. 1–10.

[17] S. Ryoo *et al.*, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proc. 13th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, Feb. 20–23, 2008, pp. 73–82.

[18] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," *J. Parallel Distrib. Comput.*, vol. 68, no. 10, pp. 1370–1380, Oct. 2008.

[19] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on theGPU using CUDA," in *High Performance Computing—HiPC2007*, Berlin/Heidelberg, Germany, 2007, Springer, ser. Lecture Notes in Computer Science.

[20] B. Huang *et al.*, "Development of a GPU-based high-performance radiative transfer model for the infrared atmospheric sounding interferometer (IASI)," *J. Computational Physics*, vol. 230, pp. 2207–2221,2011.

[21] J. Mielikainen, B. Huang, and A.-L. Huang, "GPU-accelerated multiprofile radiative transfer model for the infrared atmospheric sounding interferometer," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 691–700, 2011.