International Journal of Applied Earth Observation and Geoinformation 26 (2014) 312–321

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

journal homepage: www.elsevier.com/locate/jag

# Implementation and performance of a general purpose graphics processing unit in hyperspectral image analysis

H.M.A. van der Werff*, W.H. Bakker

*Faculty for Geo-Information Science and Earth Observation, University of Twente, Enschede 7500 AE, The Netherlands*

ARTICLE INFO

*Article history:*
Received 18 June 2013
Accepted 16 August 2013

*Keywords:*
Hyperspectral
Classification
Graphicshardware
GPGPU
IDL

ABSTRACT

A graphics processing unit (GPU) can perform massively parallel computations at relatively low cost. Software interfaces like NVIDIA CUDA allow for General Purpose computing on a GPU (GPGPU). Wrappers of the CUDA libraries for higher-level programming languages such as MATLAB and IDL allow its use in image processing. In this paper, we implement GPGPU in IDL with two distance measures frequently used in image classification, Euclidean distance and spectral angle, and apply these to hyperspectral imagery. First we vary the data volume of a synthetic dataset by changing the number of image pixels, spectral bands and classification endmembers to determine speed-up and to find the smallest data volume that would still benefit from using graphics hardware. Then we process real datasets that are too large to fit in the GPU memory, and study the effect of resulting extra data transfers on computing performance. We show that our GPU algorithms outperform the same algorithms for a central processor unit (CPU), that a significant speed-up can already be obtained on relatively small datasets, and that data transfers in large datasets do not significantly influence performance. Given that no specific knowledge on parallel computing is required for this implementation, remote sensing scientists should now be able to implement and use GPGPU for their data analysis.



## 1. Introduction

Spectroscopy is the study of light as a function of wavelength that has been emitted, reflected or scattered from a solid, liquid or gas (van der Meer, 2006). Hyperspectral image analysis is the measurement, analysis and interpretation of infrared spectra acquired from a short, medium or long distance observation by airborne or spaceborne sensors (Paz and Plaza, 2010). The objective of hyperspectral remote sensing, also referred to as imaging spectrometry or imaging spectroscopy, is to identify and quantify components of the Earth System from calibrated (radiance, reflectance or emissivity) spectra acquired as images in many, narrow and contiguous spectral bands (van der Meer et al., 2012). Hyperspectral image data have been available for civilian use as of the 1970s, with multiple high spectral resolution sensors developed since (van der Meer et al., 2012). The continuous increase of spatial and spectral resolution over these years has led to a substantial increase in data volume, a trend that is expected to continue in the future (Plaza et al., 2011).

While in the past most hyperspectral sensors were operated from airborne platforms, several spaceborne hyperspectral missions are currently under development (van der Meer et al., 2012). Examples of these upcoming missions are: PRISMA or PRecursore IperSpettrale of the application mission, developed by the Italian Space Agency (ISA) for launch in 2015; Sentinel missions (Berger and Aschbacher, 2012), developed by the European Space Agency (ESA) under the Global Monitoring for Environment and Security (GMES) programme; HISUI or Hyperspectral Imager SUIte, formerly called Hyper–X, developed by Japan for the Advanced Land Observation Satellite 3 (ALOS-3) satellite with a launch scheduled for 2014); EnMAP or Environmental Mapping and Analysis Program, developed by the German Aerospace Center (DLR) and the German Research Centre for Geosciences (GFZ) for launch in 2017; HYPXIM or HYPer Spectral IMagerie, developed by the Centre National d'Etudes Spatiales (CNES) for launch in 2019; and HYSPIRI or Hyperspectral Infrared Imager, developed by NASA, with a launch tentatively scheduled for 2020. Spaceborne hyperspectral sensors would allow a frequent revisit of targets, for example every 3–4 days for EnMAP (Segl et al., 2010), and will as such contribute to the expected increase in data volume. A regular availability of data will also support many current and future applications that involve real-time or near real-time processing capabilities (Plaza, 2009; Lee et al., 2011; Plaza et al., 2011). As the amount of data and complexity of processing rises, the demand for processing power increases (Lee et al., 2011; Plaza et al., 2011).

There are significant opportunities to increase computational power through parallel processing on modern graphics hardware (Christophe et al., 2011). In the recent years, computer graphics

* Corresponding author. Tel.: +31 53 4874 535; fax: +31 53 4874 336.
*E-mail address:* harald.vanderwerff@utwente.nl (H.M.A. van der Werff).

0303-2434/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jag.2013.08.009

hardware has become increasingly popular in scientific computing (Kyoung-Su Oh and Jung, 2004; Walsh et al., 2009) as it can perform massively parallel scientific computations (Block et al., 2010). The technique of using a graphics processing unit (GPU), which was designed for handling computer graphics, to perform computations traditionally handled by the central processing unit (CPU) is called General-Purpose computation on Graphics Processing Units (GPGPU). Of the modern personal computers, 90% have GPUs integrated on a video card or directly on the main board (Fresse et al., 2010) and are now commodity hardware. As a result, this high performance comes at relatively low cost, not only compared to CPUs but also to other high-performance computing solutions such as field programmable gate arrays (FPGAs, Setoain et al., 2008) or computer grids and clusters (Paz and Plaza, 2010).

The advent of NVIDIA's Compute Unified Device Architecture (CUDA, NVIDIA and Corp, 2013) introduced the possibility of including GPUs in science and engineering applications. CUDA is a C-language environment with extensions for NVIDIA GPU hardware specifically (Tech-X and Corp, 2013). Although CUDA allows GPGPU via application programming interfaces (APIs), software development kits (SDKs) and GPU-enabled C programming language variants, it still requires programmers to be familiar with parallel programming, boundary conditions of GPU hardware (Christophe et al., 2011) and addressing GPU memory (Fresse et al., 2010). Hence Christophe et al. (2011) state about "the final user of the program" that "it is important to keep him isolated from such implementation details". However, the complexity of parallel programming can be avoided by using high-level library wrappers. Such wrappers make CUDA functionality available to a fourth-generation programming languages (4GL) that are frequently used by remote sensing scientists, such as MATLAB (MathWorks, 2013) and IDL (Exelis VIS, 2013).

The use of high performance computing in hyperspectral remote sensing has been extensively discussed before, for example in Plaza et al. (2006b, 2011). Implementations can be found in the work of Setoain et al. (2008) and Plaza et al. (2006a), who demonstrated the use of GPUs for endmember extraction; Tarabalka et al. (2009) who performed anomaly detection related to spectral unmixing; and Plaza et al. (2011) who programmed several parallel implementations of a typical hyperspectral processing chain, including a version for GPUs. In general, these implementations were done in the C or C++ programming languages and performed by computer scientists.

In this paper, we implemented GPGPU in a high-level language that is frequently used by remote sensing scientists. We chose IDL (Exelis VIS, 2013) and the GPULib wrapper for CUDA (Tech-X and Corp, 2013, 2011) to use GPGPU for hyperspectral image classification. In this implementation, specific knowledge of parallel programming is not needed, which makes the use of GPGPU potentially available to a wider community of remote sensing scientists. We evaluate the speed-up obtained by using GPGPU in IDL, when realistically applied to hyperspectral datasets.

## 2. Methods

We first benchmarked our algorithms on synthetic data, and subsequently applied them to real hyperspectral datasets that have been acquired with airborne platforms. For benchmarking the GPU algorithms against the CPU algorithms, two commonly used distance measures in feature space (Section 2.1) were selected for implementation in IDL. We developed IDL and IDL+GPULib code for the CPU and GPU algorithms respectively, validated these algorithms with processing results of real hyperspectral imagery, and subsequently benchmarked the algorithms with synthetic data. After the benchmark, we tested the algorithms on the three airborne hyperspectral datasets. Some of these datasets were too large to fit in the GPU memory, and required multiple subsets of these images to be transferred to the GPU memory for processing, an approach known as "tiling". The resulting extra data transfers are known to reduce overall computing performance, and are hence of importance in the implementation and evaluation of GPGPU in image processing.

### 2.1. Distance measures

Two distance measures in feature space, the Euclidean distance and the spectral or vector angle, were used for benchmarking the GPU and CPU algorithms. These algorithms have been chosen as their implementation in IDL and IDL+GPULib can, apart from added statements for GPU processing, be kept similar. The Euclidean distance (ED) is the most obvious method (Bakker and Schmidt, 2002) to determine the distance between two vectors in feature space. The similarity between two spectra $\vec{v}$ and $\vec{w}$ is determined as the norm of the difference vector $\vec{v}-\vec{w}$:

$$ED_{(\vec{v},\vec{w})} = ||\vec{v}-\vec{w}||$$

As the Euclidean distance measure may be dominated by the intensity component, i.e. the length of the vectors, the intensity and spectral components are in remote sensing classifications often determined separately (Bakker and Schmidt, 2002). The spectral angle (SA) (Kruse et al., 1993) is one of the most widely used distance measures for determining spectral similarity. This method treats image and reference spectra as vectors in an $n$-dimensional space and calculates the angle between these vectors as a measure of similarity (Hecker et al., 2008):

$$SA_{(\vec{v},\vec{w})} = \cos^{-1}\left(\frac{\vec{v}\cdot\vec{w}}{||\vec{v}||\ \ ||\vec{w}||}\right)$$

A difference between the SA and ED implementations is that the SA calculation has in theory an extra computational step when compared to the ED calculation (scaling of the spectra, which is done by dividing vectors by their length). In practice, however, hardware limitations of the GPU requires an addition of extra computational steps in the ED calculation. The IDL and IDL+GPULib implementation of these distance measures is explained in Section 2.3 and shown in Figs. 1–4.

### 2.2. Software and hardware

The computer used in this research was an Intel x86_64 workstation with 12 Gb of RAM and an Intel Core i7-930 CPU running at 2.8 GHz. The GPU is an NVIDIA Tesla C1060, which has 4029 Mb of memory available and is capable of running 240 threads simultaneously. This GPU card has no video connector, and computer graphics therefore did not influence the performance of the GPU. The computer had a 64-bit GNU/Linux operating system (Debian "Wheezy" distribution (The Debian Project, 2013)). Version 4.2 of the NVIDIA CUDA toolkit was used in combination with version 302.17 of the (proprietary) NVIDIA display driver. IDL version 8.2 with GPULib version 1.4.4 (both also 64-bit) were used for code development and benchmarking.

### 2.3. Algorithm design

Four algorithms were initially developed to calculate the Euclidean distance and spectral angle similarity measures, two on a CPU only (using IDL code) and two that make use of a GPU (using IDL+GPULib code). The IDL+GPULib implementation differs mostly from the IDL implementation by the presence of extra statements that regulate data transfers: All data that needs to be processed by the GPU have to be transferred from computer memory to GPU