Hyperspectral Unmixing on NVidia™ Graphics Processing Units

Antonio Plaza[a], Sergio Sánchez[a], Javier Plaza[a]

[a]Department of Technology of Computers and Communications
University of Extremadura, Avda. de la Universidad s/n
E-10071 Cáceres, Spain

ABSTRACT

Spectral unmixing is an important task for remotely sensed hyperspectral data interpretation. It comprises both the determination of spectrally pure signatures (endmembers) and an unmixing process that interprets mixed pixels as combinations of endmembers are computationally expensive procedures. An exciting recent development in the field of commodity computing is the emergence of programmable graphics processing units (GPUs), which are now increasingly being used address the ever-growing computational requirements introduced by hyperspectral imaging applications. In this paper, we develop three new GPU-based implementations of automatic endmember extraction algorithms from hyperspectral image data: the pixel purity index (PPI), a kernel version of the PPI (KPPI), and the automatic morphological endmember extraction (AMEE) algorithm. We also provide a GPU-based implementation of a fully constrained linear spectral unmixing algorithm. Combined, these processing modules allow us to implement the full hyperspectral unmixing chain on NVidia™ GPUs. Experimental results are provided using hyperspectral data provided by NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) operated by Jet Propulsion Laboratory. The considered application is focused on a mineral mapping problem over the Cuprite mining district in Nevada. This problem is analyzed in terms of the accuracy obtained after applying linear spectral unmixing techniques and also from the viewpoint of the parallel efficiency of the proposed GPU implementation.

Keywords: S pectral unmixing, hyperspectral imaging, commodity graphics hardware, endmember extraction

1. INTRODUCTION

Hyperspectral imaging is concerned with the measurement, analysis, and interpretation of spectra acquired from a given scene (or specific object) at a short, medium or long distance by an airborne or satellite sensor.[1] For instance, NASA is continuously gathering hyperspectral data with Earth-observing sensors such as JPL's Airborne Visible-Infrared Imaging Spectrometer (AVIRIS),[2] able to record the visible and near-infrared spectrum (wavelength region from 0.4 to 2.5 micrometers) of the reflected light of an area 2 to 12 kilometers wide and several kilometers long using 224 spectral bands. The resulting hyperspectral data cube is a stack of images (see Fig. 1) in which each pixel (vector) has an associated spectral signature or *fingerprint* that uniquely characterizes the underlying objects, and the resulting data volume typically comprises several GBs per flight.

Spectral mixture analysis has been an alluring exploitation goal since the earliest days of hyperspectral imaging.[3] No matter the spatial resolution, in natural environments the spectral signature for a nominal pixel is invariably a mixture of the signatures of the various materials found within the spatial extent of the ground instantaneous field view of the sensor. In hyperspectral imagery, the number of spectral bands usually exceeds the number of pure spectral components, called *endmembers* in hyperspectral analysis terminology, and the unmixing problem is cast in terms of an over-determined system of equations in which, given the correct set of endmembers allows determination of the actual endmember abundance fractions through a numerical inversion process. Since each observed spectral signal is the result of an actual mixing process, the driving abundances must obey two constraints. First, all abundances must be non-negative. Second, the sum of abundances for a given pixel must be unity.[4]

Send correspondence to Antonio J. Plaza:
E-mail: aplaza@unex.es; Telephone: +34 927 257000 (Ext. 51662); URL: http://www.umbc.edu/rssipl/people/aplaza

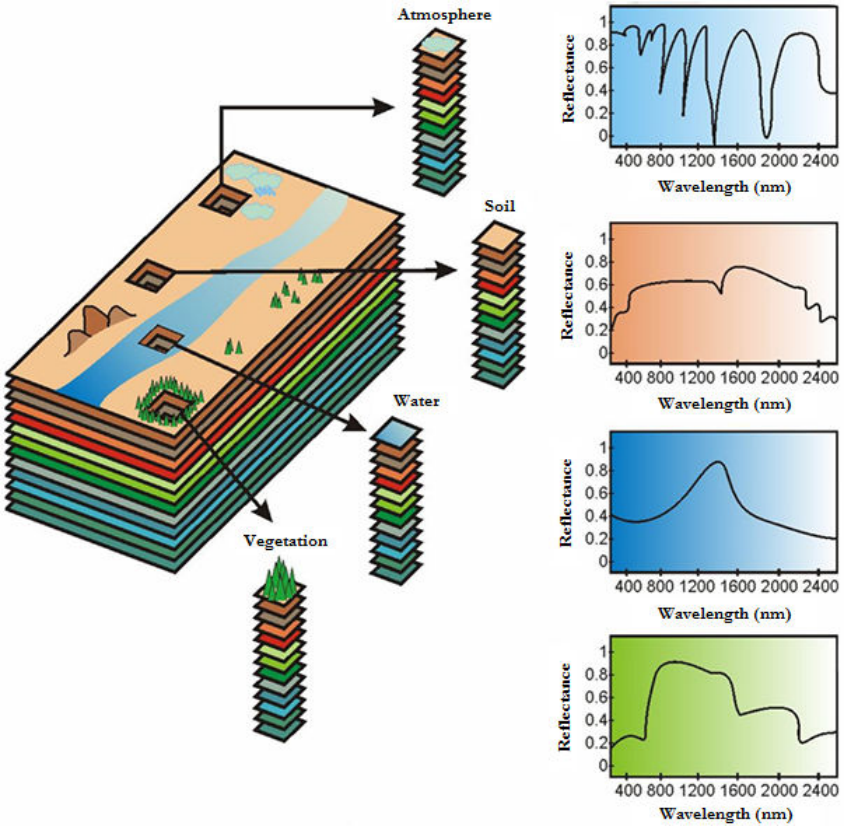


Figure 1. The concept of hyperspectral imaging.

Over the last decade, several algorithms have been developed for automatic extraction of endmembers from hyperspectral data sets.[5] A popular approach focused on exploiting the spectral information in the data has been the pixel purity index (PPI).[6] Other algorithms, such as the automatic morphological endmember extraction (AMEE),[7] integrate the spatial and the spectral information in the data. While hyperspectral imaging algorithms such as PPI or AMEE hold great promise for Earth science image analysis, they also introduce new processing challenges, in particular, for very high-dimensional data sets. From a computational perspective, these algorithms exhibit inherent parallelism at multiple levels: across pixel vectors (coarse grained pixel-level parallelism), across spectral information (fine grained spectral-level parallelism), and even across tasks (task-level parallelism). As a result, they map nicely to massively parallel systems such as clusters.[8] Unfortunately, these systems are expensive and difficult to adapt to onboard data processing scenarios,[9] in which low-weight and low-power integrated components are mandatory to reduce mission payload.[10]

An exciting recent development in the field of commodity computing is the emergence of programmable graphics processing units (GPUs),[11,12] which bear many features of vector processors. The speed of graphics hardware doubles approximately every six months, which is much faster than the improving rate of the CPU. The ever-growing computational requirements introduced by hyperspectral imaging applications can benefit from this hardware and take advantage of the compact size and relatively low cost of these units, which make them appealing for on-board data processing at much lower costs than those introduced by other hardware devices.[13] In this paper, we develop three new GPU-based implementations of endmember extraction algorithms: the PPI, a kernel version of the PPI, and the spatial-spectral AMEE algorithm. We also provide a GPU-based implementation of the fully constrained linear spectral unmixing algorithm. The algorithms have been implemented on an NVidia GeForce 8800 GTX GPU*, using NVidia's CUDA† programming language.

*http://www.nvidia.com/page/geforce_8800.html
†http://www.nvidia.com/object/cuda_home.html

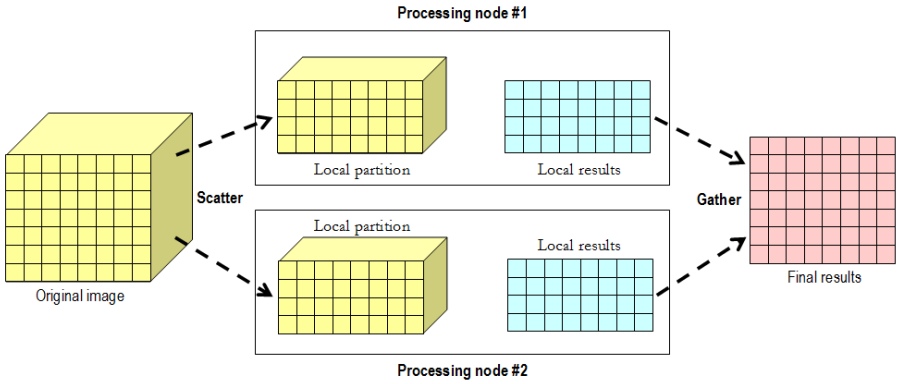


Figure 2. Spatial-domain decomposition.

2. SPECTRAL UNMIXING METHODOLOGY

A standard approach to decompose mixed pixels in hyperspectral images is linear spectral unmixing, which comprises the following stages. Firstly, a set of spectral endmembers are extracted from the input data set. For this purpose, we have considered three standard algorithms in the literature: the PPI algorithm, a kernel version of the PPI (called KPPI), and the AMEE algorithm. Then, the fractional abundances of such endmembers in each pixel of the scene is obtained using an inversion process. A brief description of the considered endmember extraction and spectral unmixing algorithms follows:

- The PPI proceeds by generating a large number of random, N-dimensional unit vectors called "skewers" through the dataset.[6] Every data point is projected onto each skewer, and the data points that correspond to extrema in the direction of a skewer are tallied. The pixels with the highest tallies are considered the final endmembers.

- On the other hand, the AMEE begins by searching spatial neighborhoods around each pixel in the image for the most spectrally pure and mostly highly mixed pixel. This task is performed by using extended mathematical morphology operators of dilation and erosion, respectively.[7] Each spectrally pure pixel is assigned an "eccentricity" value, which is calculated as the spectral angle distance between the most spectrally pure and mostly highly mixed pixel for the given spatial neighborhood. This process is repeated iteratively for larger spatial neighborhoods up to a maximum size that is pre-determined. The final endmember set is obtained by applying a threshold to the "eccentricity" image.

- Finally, the KPPI can be seen as a hybrid between the PPI and the AMEE. It begins by searching spatial neighborhoods around each pixel in the image, but instead of using morphological operations of erosion and dilation, it looks for the most spectrally pure pixels in the local neighborhood by applying a variant of the original PPI in each local neighborhood.

3. GPU IMPLEMENTATIONS

The algorithms were implemented using NVidia's CUDA, a collection of C extensions and a runtime library. CUDA's functionality primarily allows a developer to write C functions to be executed on the GPU. CUDA also includes memory management and execution configuration, so that a developer can control the number of GPU processors and threads that are to be invoked during a function's execution.

The first issue that needs to be addressed is how to map a hyperspectral image onto the memory of the GPU. Since the size of hyperspectral images usually exceeds the capacity of such memory, we split them into multiple spatial-domain partitions[8] made up of entire pixel vectors (see Fig. 2), i.e., each spatial-domain partition incorporates all the spectral information on a localized spatial region and is composed of spatially adjacent pixel vectors. Each spatial-domain parition is further divided into 4-band tiles (called spatial-domain tiles), which are

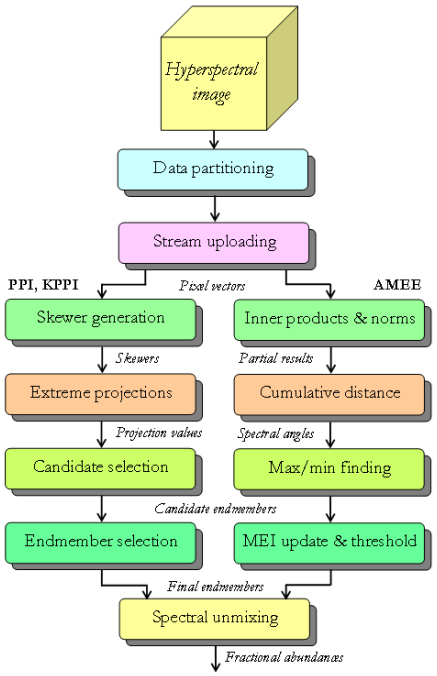


Figure 3. Flowchart of the stream-based GPU implementations.

arranged in different areas of a 2-D texture. Such partitioning allows us to map four consecutive spectral bands onto the RGBA color channels of a texture element.

Apart from the tiles, we also allocate additional memory to hold other information, such as the skewers which are generated at the host (CPU) and then transmitted to the GPU, and also intermediate results such as dot products, norms, and pointwise distances. Figure 3 shows a flowchart describing the kernels involved in our GPU-based implementations. There are two kernels which are common to the three endmember extraction algorithms: the *data partitioning* stage performs the spatial-domain decomposition of the original hyperspectral image. In the *stream uploading* stage, the spatial-domain partitions are uploaded as a set of tiles onto the GPU memory. The kernels which are specific to the PPI and KPPI implementations can be described as follows:

- *Skewer generation.* This kernel provides the skewers, using NVidia's parallel implementation of the Mersenne twister pseudo-random number generator on the GPU.
- *Extreme projections.* The tiles are input streams to this stage, which obtains all the dot products necessary to compute the required projections. The implementation of this stage is based on a multi-pass kernel that implements an element-wise multiply and add operation, thus producing four partial inner products stored in the RGBA channels of a texture element.
- *Candidate selection.* This kernel uses as inputs the projection values generated in the previous stage, and produces a stream for each pixel vector, containing the relative coordinates of the pixels with maximum and minimum distance after the projection onto each skewer. A complementary kernel is then used to find pixels which have been selected repeatedly during the process.
- *Endmember selection.* For each endmember candidate, this kernel computes the cumulative SAD with all the other candidates. It is based on a single-pass kernel that computes the SAD between two pixel vectors using the dot products and norms produced by the previous stage. A complementary kernel is then used to discard those candidates with cumulative SAD scores below a threshold angle.

It should be noted that both PPI and KPPI use the same kernels. The only difference between these two algorithms is in the way pixels are fed in the *stream uploading* stage. In the case of PPI, the pixels are fed

without any spatial arrangement, while in the KPPI the pixels are fed in local neighborhoods according to the kernel size parameter. This strategy is also used in the AMEE implementation, which is based on the following specific kernels:

- *Inner products & norms.* The tiles are input streams to this kernel, which obtains all the inner products and norms necessary to compute the point-wise spectral angle distances involved in the calculations.
- *Cumulative distance.* For each pixel vector, this kernel accumulates the spectral angle with all the neighboring pixels. It is based on a single-pass kernel that computes the spectral angle distance between two pixel vectors using the inner products and norms produced by the previous kernel. Finally, the kernel calculates, for each pixel vector, the cumulative spectral angle between the pixel and all its neighbors.
- *Max/min finding.* Morphological erosion and dilation are finalized at this stage through a kernel that applies minimum and maximum reductions. This kernel uses as inputs the cumulative values generated in the previous stage and produces a stream containing (for each pixel) the relative coordinates of the neighboring pixels with maximum and minimum cumulative distance.
- *Eccentricity update.* This kernel updates the morphological eccentricity scores using the maximum/minimum and point-wise distance streams. A complementary kernel applies a threshold to select a set of final endmembers at the end of the process.
- *Spectral unmixing.* Finally, this kernel (common to all endmember extraction algorithms) uses as inputs the final endmembers selected in the previous stage and produces the endmember fractional abundances for each pixel by means of an inversion process.[14]

4. EXPERIMENTAL RESULTS

4.1 GPU platform

Our experiments were performed on a 2006-model HP xw8400 workstation based on dual Quad-Core Intel Xeon processor E5345 running at 2.33 GHz with 1.333 MHz bus speed and 3 GB RAM. The computer was equipped with an NVidia GeForce 8800 GTX with 16 multiprocessors, each composed of 8 processors operating at 1350 Mhz. Each multiprocessor has 8192 registers, a 16 KB parallel data cache of fast shared memory, and access to 768 MB of global memory.

4.2 Hyperspectral data

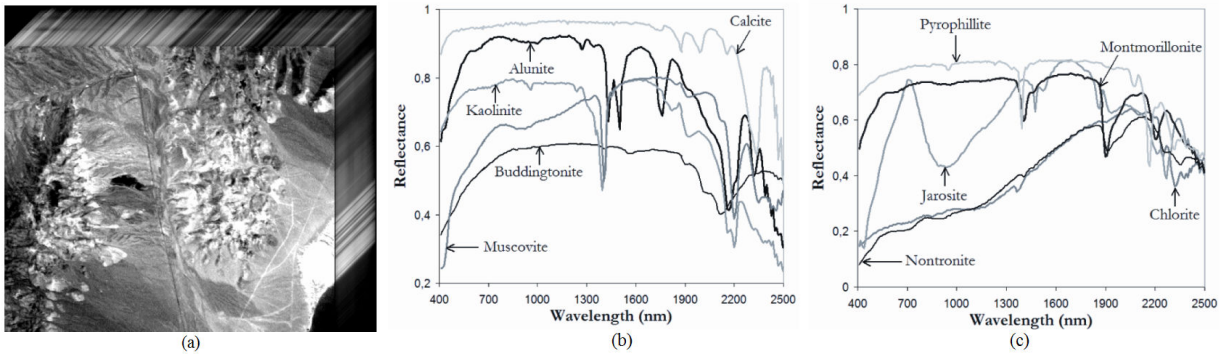
A well-known hyperspectral data set collected over the Cuprite mining district in Nevada was used in experiments to evaluate the algorithms in the context of a real mineral mapping application. The data set[‡] consists of 1939×677 pixels and 224 bands in the wavelength range 0.4–2.5 μm (574 MB in size). The Cuprite site has been extensively mapped by the U.S. Geological Survey (USGS), and there is extensive ground-truth information available, including a library of mineral signatures collected on the field[§]. Fig. 4(a) shows the spectral band at 587 nm wavelength of the AVIRIS scene. The spectra of USGS ground minerals are also displayed in Figs. 4(b-c).

4.3 Hyperspectral image experiments

A comparison in terms of spectral angle distances between the endmembers extracted by the PPI, KPPI and AMEE versus USGS reference signatures (not displayed here) revealed that the three considered algorithms were able to extract good candidate endmembers from the input scene. In order to study the scalability of our CPU and GPU-based implementations, we tested them on a subset (614×512 pixels and 224 bands, for a total size of 140 MB). Table 1 shows the execution times and speedups measured for the GPU-based implementations compared to execution in the dual-core CPU of the system in which the GPU is integrated. The C function *clock()* was used for timing the CPU implementation and the CUDA timer was used for the GPU implementation.

[‡]Available online from http://aviris.jpl.nasa.gov/html/aviris.freedata.html

[§]Available online from http://speclab.cr.usgs.gov/spectral-lib.html



(a) (b) (c)

Figure 4. (a) A portion of the AVIRIS scene over Cuprite mining district. (b-c) Ground-truth mineral spectra provided by USGS.

Table 1. Processing time (seconds) and speedups for the dual-core CPU and GPU implementations.

Algorithm	Processing time (CPU)	Processing time (GPU)	Speedup
PPI	493.78	18.93	26.08
KPPI	177.30	19.29	9.19
AMEE	1345.18	52.60	25.57

The time measurement was started right after the hyperspectral image file was read to the CPU memory and stopped right after the results of the processing chain were obtained and stored in the CPU memory. In all cases, the time for spectral unmixing was included in the calculations.

The speedups achieved by the GPU implementation of the PPI and AMEE algorithms over their respective CPU implementations remained close to 25. It should be noted that the speedup achieved for the GPU implementation of AMEE were independent of the kernel size, even for very large kernels (the results displayed in Table 1 correspond to a kernel size of 5×5 pixels). The GPU implementation of the KPPI was not as effective, probably due to the procedure adopted for incorporating the spatial information into the PPI algorithm. This aspect will be subject to improvements in our future research work. From Table 1, it can be seen that the full AVIRIS data cube could be processed in 18.93 seconds by the PPI. This response is not strictly in real-time since the cross-track line scan time in AVIRIS, a push-broom instrument,[2] is quite fast (8.3 msec). This introduces the need to process the full image cube (614×512 pixels and 224 bands) in less than 5 seconds to achieve real-time performance. Further developments will be pursued in future work in order to approximate real-time performance for onboard data exploitation.

5. CONCLUSIONS

The emergence of low-weight and low-power specialized hardware devices such as graphics processing units (GPUs), whose evolution in terms of performance and cost is mainly due to the advent of video-game industry, is now progressively bridging the gap towards real-time analysis of high dimensional data in many application domains, including remote sensing. This kind of specialized, on-board processing devices are essential to reduce mission payload and obtain analysis results quickly enough for practical use. In this work, we have presented our experience in computationally efficient implementation of algorithms for endmember extraction and spectral unmixing of remotely sensed hyperspectral data using commodity graphics hardwere. Specifically, three algorithms: PPI, KPPI and AMEE have been implemented on an NVidia GeForce 8800 GTX, obtaining speedups on the order of 26 for the PPI and AMEE, and 9 for the KPPI.

6. ACKNOWLEDGEMENT

This work has been supported by the European Community's Marie Curie Research Training Networks Programme under reference MRTN-CT-2006-035927 (HYPER-I-NET). Funding from the Spanish Ministry of Science and Innovation (HYPERCOMP/EODIX project, reference AYA2008-05965-C04-02) is also gratefully acknowledged.

REFERENCES

1. A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for earth remote sensing," *Science* **228**, pp. 1147–1153, 1985.
2. R. O. Green, "Imaging spectroscopy and the airborne visible-infrared imaging spectrometer (AVIRIS)," *Remote Sensing of Environment* **65**, pp. 227–248, 1998.
3. J. B. Adams, M. O. Smith, and P. E. Johnson, "Spectral mixture modeling: a new analysis of rock and soil types at the viking lander 1 site," *Journal of Geophysical Research* **91**, pp. 8098–8112, 1986.
4. C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification*, Kluwer Academic and Plenum Publishers, New York, 2003.
5. A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing* **42**, pp. 650–663, 2004.
6. J. Boardman, F. A. Kruse, and R. O. Green, "Mapping target signatures via partial unmixing of AVIRIS data," in *Summaries of the JPL Airborne Earth Science Workshop*, pp. 23–26, JPL Publication, 1995.
7. A. Plaza, P. Martinez, R. Perez, and J. Plaza, "Spatial/spectral endmember extraction by multidimensional morphological operations," *IEEE Transactions on Geoscience and Remote Sensing* **40**, pp. 2025–2041, 2002.
8. A. Plaza, J. Plaza, and D. Valencia, "Impact of platform heterogeneity on the design of parallel algorithms for morphological processing of high-dimensional image data," *Journal of Supercomputing* **40**, pp. 81–107, 2007.
9. A. Plaza and C.-I. Chang, "Clusters versus FPGA for parallel processing of hyperspectral imagery," *International Journal of High Performance Computing Applications* **22**(4), pp. 366–385, 2008.
10. A. Plaza and C.-I. Chang, *High performance computing in remote sensing*, CRC Press, Boca Raton, 2006.
11. J. Setoain, M. Prieto, C. Tenllado, A. Plaza, and F. Tirado, "Parallel morphological endmember extraction using commodity graphics hardware," *IEEE Geoscience and Remote Sensing Letters* **43**, pp. 441–445, 2007.
12. Y. Tarabalka, T. V. Haavardsholm, I. Kasen, and T. Skauli, "Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and gpu processing," *Journal of Real-Time Image Processing* **4**, pp. 1–14, 2009.
13. J. Setoain, M. Prieto, C. Tenllado, and F. Tirado, "GPU for parallel on-board hyperspectral image processing," *International Journal of High Performance Computing Applications* **22**(4), pp. 424–437, 2008.
14. D. Heinz and C.-I. Chang, "Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing* **39**, pp. 529–545, 2001.