Real-Time Implementation of a Full Hyperspectral Unmixing Chain on Graphics Processing Units

Sergio Sánchez[a] and Antonio Plaza[a]

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

ABSTRACT

Hyperspectral unmixing is a very important task for remotely sensed hyperspectral data exploitation. It amounts at estimating the abundance of pure spectral signatures (called endmembers) in each mixed pixel of the original hyperspectral image, where mixed pixels arise due to insufficient spatial resolution and other phenomena. The full spectral unmixing chain comprises three main steps: 1) dimensionality reduction, in which the original hyperspectral data is brought to an adequate subspace; 2) endmember extraction, in which endmembers are automatically identified from the image data; and 3) abundance estimation, in which the fractional coverage of each endmember is estimated for each pixel of the hyperspectral scene. The hyperspectral unmixing process can be time-consuming, particularly for high-dimensional hyperspectral images. Parallel computing architectures have offered an attractive solution for fast unmixing of hyperspectral data sets, but these systems are expensive and difficult to adapt to on-board data processing scenarios, in which low-weight and low-power integrated components are essential to reduce mission payload and obtain analysis results in (near) real-time. In this paper, we develop a real-time implementation of a full unmixing chain for hyperspectral data on graphics processing units (GPUs). These hardware accelerators can bridge the gap towards on-board processing of this kind of data. The considered chain comprises principal component analysis (PCA) for dimensionality estimation, extraction of endmembers using the N-FINDR algorithm, and unconstrained linear spectral unmixing. The proposed GPU implementation is shown to perform strictly in real-time for hyperspectral data sets collected by the NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS).

Keywords: Hyperspectral imaging, spectral unmixing, dimensionality reduction, endmember extraction, abundance estimation, graphics processing units (GPUs).

1. INTRODUCTION

Hyperspectral imaging instruments are capable of collecting hundreds of images, corresponding to different wavelength channels, for the same area on the surface of the Earth.[1] For instance, NASA is continuously gathering imagery data with instruments such as the Jet Propulsion Laboratory's Airborne Visible-Infrared Imaging Spectrometer (AVIRIS), which is able to record the visible and near-infrared spectrum (wavelength region from 0.4 to 2.5 micrometers) of reflected light in an area 2 to 12 kilometers wide and several kilometers long, using 224 spectral bands.[2]

One of the main problems in the analysis of hyperspectral data cubes[3] is the presence of mixed pixels,[4] which arise when the spatial resolution of the sensor is not fine enough to separate spectrally distinct materials. In this case, several spectrally pure signatures (*endmembers*) are combined into the same (mixed) pixel. Spectral unmixing involves the separation of a pixel spectrum into its pure component endmember spectra,[5,6] and the estimation of the abundance value for each endmember.[7–9] A popular approach for this purpose in the literature has been linear spectral unmixing,[10] which assumes that the endmember substances interact linearly within the field of view of the imaging instrument.[11] In practice, the linear model is flexible and can be easily adapted to different analysis scenarios. The linear unmixing chain is graphically illustrated by a flowchart in Fig. 1. It

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

Satellite Data Compression, Communications, and Processing VII, edited by Bormin Huang, Antonio J. Plaza, Carole Thiebaut, Proc. of SPIE Vol. 8157, 81570F · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.892284

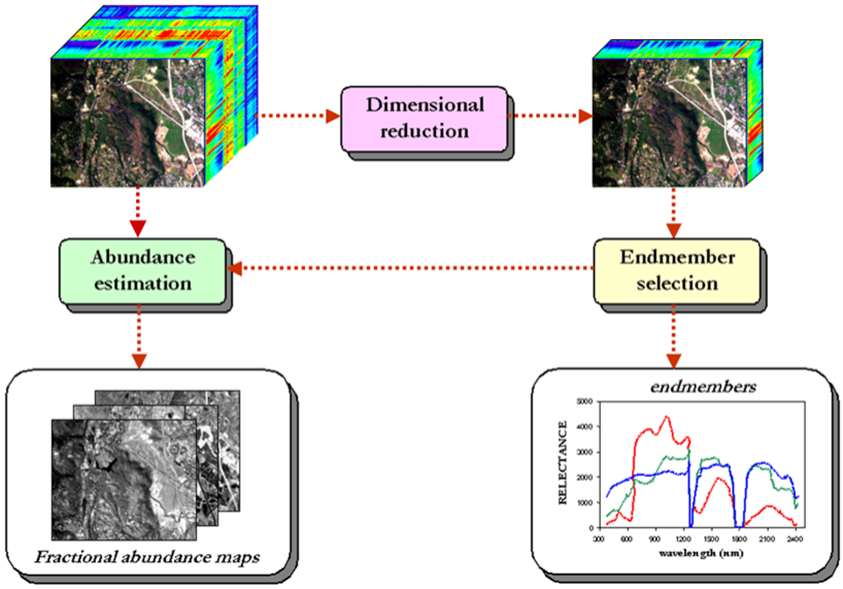


Figure 1. Standard hyperspectral unmixing chain.

generally comprises three stages: 1) reduction of the dimensionality of the original image to a proper subspace; 2) automatic identification of pure spectral signatures (called endmembers); and 3) estimation of the fractional abundance of each endmember in each pixel of the scene. The unmixing process is quite computationally expensive, due to the extremely high dimensionality of hyperspectral data cubes.[12–16]

Although the unmixing chain maps nicely to high performance computing systems such as commodity clusters,[16] these systems are difficult to adapt to on-board processing requirements introduced by applications with real-time constraints such as wild land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination. In those cases, low-weight integrated components such as commodity graphics processing units (GPUs)[17] are essential to reduce mission payload. In this regard, the emergence of GPUs now offers a tremendous potential to bridge the gap towards real-time analysis of remotely sensed hyperspectral data.[18–24]

In this paper, we develop a real-time implementation of the full spectral unmixing chain for GPUs. The proposed methodology has been implemented using NVidia's compute device unified architecture (`CUDA`), and tested on an NVidia Tesla C1060 GPU using a hyperspectral image collected by AVIRIS. The remainder of the paper is organized as follows. Section 2 describes the different modules that conform the considered unmixing chain. Section 3 describes the GPU implementation of these modules. Section 4 presents an experimental evaluation of the proposed implementations in terms of both unmixing accuracy and parallel performance. Section 5 concludes the paper with some remarks and hints at plausible future research lines.

2. HYPERSPECTRAL UNMIXING CHAIN

The proposed unmixing chain is based on the concept of linear spectral unmixing, which can be defined in mathematical terms as follows. Let $\mathbf{x}$ be a pixel vector given by a collection of values at different wavelengths. In the context of linear spectral unmixing, such vector can be modeled as:

$$\mathbf{x} \approx \mathbf{E}\mathbf{a} + \mathbf{n} = \sum_{i=1}^{p} \mathbf{e}_i a_i + \mathbf{n}, \tag{1}$$

where $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{p}$ is a matrix containing p pure spectral signatures (endmembers), $\mathbf{a} = [a_1, a_2, \cdots, a_p]$ is a p-dimensional vector containing the abundance fractions for each of the p endmembers in $\mathbf{x}$, and $\mathbf{n}$ is a noise term. Solving the linear mixture model involves: 1) estimating the right subspace for the hyperspectral data,

2) identifying a collection of $\{\mathbf{e}_i\}_{i=1}^{p}$ endmembers in the image, and 3) estimating their abundance in each pixel of the scene. In this work, the dimensional reduction step is performed using principal component analysis (PCA),[25,26] a popular tool for feature extraction in different areas including remote sensing. For the endmember selection part, we rely on the well-known N-FINDR algorithm,[27] which is a standard the hyperspectral imaging community. Finally, abundance estimation is carried out using unconstrained least-squares (UCLS) estimation[9] due to its simplicity. In the following, we describe the three modules adopted in our hyperspectral unmixing chain implementation: PCA, N-FINDR and UCLS.

2.1 PCA for dimensionality reduction

The implementation of PCA adopted in this work is the one described in.[28] This technique is an approximation to PCA from which principal components can be derived in sequential fashion. More specifically, the algorithm is based on sequentially deriving the eigenvectors that maximize the variance over all samples, and has been proven to be computationally efficient.[28] Let us denote the original hyperspectral image $\mathbf{X}$ as a set of pixel vectors as follows: $\mathbf{X} = \{\mathbf{x}_j\}_{j=1}^{m}$, where m is the total number of pixels in the original image. The considered approach for dimensionality reduction is based on the following decomposition:

$$y_n = (\mathbf{v}_n^k)^T \mathbf{x}_j, \tag{2}$$

where $\mathbf{v}_n^k$ is an eigenvector that represents the n-th principal component and k is number of repetitions. An initial vector, $\mathbf{v}_n^0$, is first set to random values. A threshold function is also used:

$$f(y_n, \mathbf{x}_j) = \begin{Bmatrix} \mathbf{x}_j & \text{if} & y_n \geq 0 \\ 0 & \text{otherwise} & \end{Bmatrix}. \tag{3}$$

The iterative procedure is performed using the following expression:

$$\mathbf{v}_n^{k+1} = \frac{\sum_j f(y_n, \mathbf{x}_j)}{\left\| \sum_j f(y_n, \mathbf{x}_j) \right\|}, \tag{4}$$

where $\mathbf{v}_n^{k+1}$ is the eigenvector approximated after $k+1$ iterations. Here, the value of the output function is calculated by using $\mathbf{v}_n^k$, which is the result in the previous iteration. So far the method of finding one principal component has been described. In order to find the next principal component, the effect of the first principal component is removed from the dataset, so as to avoid being found again. This is performed by subtracting the component of each datum which is parallel to the principal component from that datum, leaving only an orthogonal component:

$$\mathbf{x}'_j = \mathbf{x}_j - (\mathbf{v}_n^{k+1} \cdot \mathbf{x}_j) \cdot \mathbf{v}_n^{k+1}. \tag{5}$$

After the component is removed, the principal component is evaluated and similar calculations are performed iteratively. This produces an approximate solution without the need for calculating a variance-covariance matrix and then diagonalizing it (as do most traditional methods) and does not depend on learning. Most importantly, for high dimensional datasets such as hyperspectral images, this strategy is shown in[28] to be faster than other existing techniques and it is also easy to implement as a batch or an adaptive technique.

2.2 N-FINDR for endmember selection

The N-FINDR algorithm[27] is one of the most widely used and successfully applied methods for automatically determining endmembers in hyperspectral image data without using a priori information. This algorithm looks for the set of pixels with the largest possible volume by *inflating* a simplex inside the data. The procedure begins with a random initial selection of pixels. Every pixel in the image must be evaluated in order to refine the estimate of endmembers, looking for the set of pixels that maximizes the volume of the simplex defined by selected endmembers. The corresponding volume is calculated for every pixel in each endmember position by replacing that endmember and finding the resulting volume. If the replacement results in a an increase of volume, the pixel replaces the endmember. This procedure is repeated in iterative fashion until there are no more endmember replacements. The method can be summarized by a step-by-step algorithmic description which is given below for clarity:

1. *Feature reduction*. Apply a dimensionality reduction transformation such as the PCA to reduce the dimensionality of the data from n to $p-1$, where p is an input parameter to the algorithm (number of endmembers to be extracted).

2. *Initialization*. Let $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \cdots, \mathbf{e}_p^{(0)}\}$ be a set of endmembers randomly extracted from the input data.

3. *Volume calculation*. At iteration $k \geq 0$, calculate the volume defined by the current set of endmembers as follows:
$$V(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \cdots, \mathbf{e}_p^{(k)}) = \frac{\left|\det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{e}_1^{(k)} & \mathbf{e}_2^{(k)} & \cdots & \mathbf{e}_p^{(k)} \end{bmatrix}\right|}{(p-1)!}. \quad (6)$$

4. *Replacement*. For each pixel vector $\mathbf{x}_j$ in the input hyperspectral data, recalculate the volume by testing the pixel in all p endmember positions, i.e., first calculate $V(\mathbf{x}_j, \mathbf{e}_2^{(k)}, \cdots, \mathbf{e}_p^{(k)})$, then calculate $V(\mathbf{e}_1^{(k)}, \mathbf{x}_j, \cdots, \mathbf{e}_p^{(k)})$, and so on until $V(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \cdots, \mathbf{x}_j)$. If none of the p recalculated volumes is greater than $V(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \cdots, \mathbf{e}_p^{(k)})$, then no endmember is replaced. Otherwise, the combination with maximum volume is retained. Let us assume that the endmember absent in the combination resulting in the maximum volume is denoted by $\mathbf{e}_i^{(k+1)}$. In this case, a new set of endmembers is produced by letting $\mathbf{e}_i^{(k+1)} = \mathbf{x}_j$ and $\mathbf{e}_l^{(k+1)} = \mathbf{e}_l^{(k)}$ for $l \neq i$. The replacement step is repeated for all the pixel vectors in the input data until all the pixels have been exhausted.

2.3 UCLS for abundance estimation

Once the set of endmembers $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^p$ have been extracted, their correspondent abundance fractions $\mathbf{a} = [a_1, a_2, \cdots, a_p]$ in a specific pixel vector $\mathbf{x}$ of the scene can be simply estimated (in least squares sense) by the following unconstrained expression:[9]
$$\hat{\mathbf{a}} = (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T\mathbf{x}. \quad (7)$$

Two additional constrains can be imposed into the model described in (1), these are the abundance non-negativity constraint (ANC), i.e., $a_i \geq 0$, and the abundance sum-to-one constraint (ASC), i.e., $\sum_{i=1}^p a_i = 1$.[9] However, in this work we focus on the unconstrained estimation as it is much faster and it has been shown in practice to provide satisfactory results if the model endmembers are properly selected.

3. GPU IMPLEMENTATION

GPUs can be abstracted in terms of a *stream model*, under which all data sets are represented as streams (i.e., ordered data sets). Algorithms are constructed by chaining so-called *kernels*, which operate on entire streams, taking one or more streams as inputs and producing one or more streams as outputs. Thereby, data-level parallelism is exposed to hardware, and kernels can be concurrently applied without any sort of synchronization. The kernels can perform a kind of batch processing arranged in the form of a grid of blocks, as displayed in Fig. 2(a), where each block is composed by a group of threads which share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory. There is a maximum number of threads that a block can contain but the number of threads that can be concurrently executed is much larger (several blocks executed by the same kernel can be managed concurrently, at the expense of reducing the cooperation between threads since the threads in different blocks of the same grid cannot synchronize with the other threads). Fig. 2(a) displays how each kernel is executed as a grid of blocks of threads. On the other hand, Fig. 2(b) shows the execution model in the GPU, which can be seen as a set of multiprocesors. Each multiprocessor is characterized by a single instruction multiple data (SIMD) architecture, i.e., in each clock cycle each processor of the multiprocessor executes the same instruction but operating on multiple data streams. Each processor has access to a local shared memory and also to local cache memories in the multiprocessor, while the multiprocessors have access to the global GPU (device) memory. With the above ideas in mind, our GPU implementation of the hyperspectral unmixing chain comprises three stages: 1) GPU implementation of PCA (called GPU-PCA); 2) GPU implementation of N-FINDR (called GPU-FINDR); and 3) GPU implementation of UCLS (called GPU-UCLS).

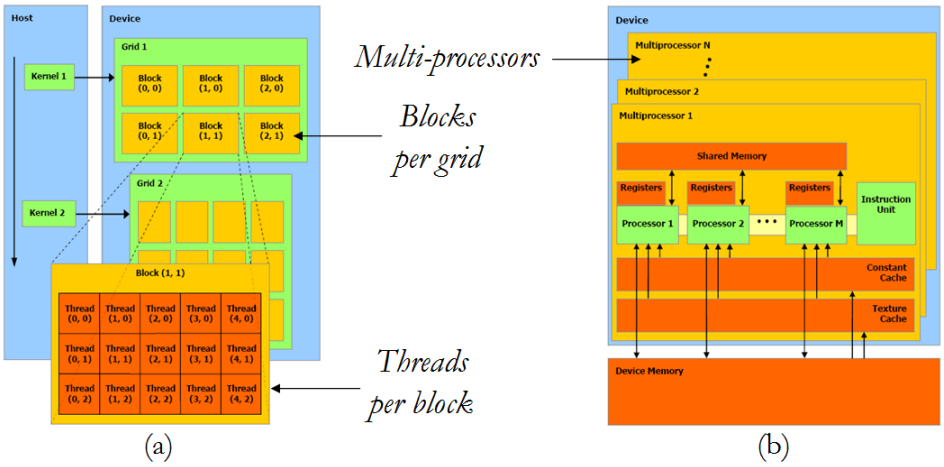


Figure 2. Schematic overview of a GPU architecture. (a) Threads, blocks and grids. (b) Execution model in the GPU.

3.1 GPU-PCA

Our implementation of PCA for GPUs can be summarized by the following steps:

1. The first step is to generate a random vector, $\mathbf{v}_n^0$. This task is done in CPU due to its simplicity. Now the vector $\mathbf{v}_n^0$ is multiplied by each vector $\mathbf{x}_j$, and this operation is accomplished in the GPU by means of a so-called `Reduction_yn` kernel. The output of the kernel is $\mathbf{y}_n$ in (2).

2. Once the vector $\mathbf{y}_n$ is calculated, the threshold function in (3) is applied and then we add all the results to obtain $\sum_j f(y_n, \mathbf{x}_j)$ in (4). This task is also accomplished in the GPU by means of a `Compute_sum` kernel.

3. Now we obtain $\mathbf{v}_n^{k+1}$ using (4). This task is accomplished in CPU. This vector, obtained after $k+1$ iterations, is an eigenvector.

4. In order to compute (5), we use two kernels. First, the `Reduction_aux` kernel computes $(\mathbf{v}_n^{k+1} \cdot \mathbf{x}_j)$, and the `Compute_x` kernel completes the calculation. Now the four steps are repeated as many times as the number of principal components needed.

3.2 GPU-FINDR

The most time-consuming computation in the N-FINDR algorithm is the calculation of the determinants. The determinant of a nonsingular matrix V is usually obtained from the factorization $PV = LU$ (where P is a permutation matrix, L is a unit lower triangular matrix, and U is an upper triangular matrix) as the product of the diagonal elements of U. This decomposition is known as *Gaussian elimination* or the LU factorization (with partial row pivoting). The repeated volume calculations of the N-FINDR algorithm can be reduced by exploiting some basic properties of the LU factorization and matrix determinants. Consider, e.g., the $p \times p$ and $p \times p - 1$ matrices:

$$\begin{aligned} V_M^{(1)} &= \begin{bmatrix} 1 & \dots & 1 & 1 \\ \mathbf{e}_2^{(0)} & \dots & \mathbf{e}_p^{(0)} & \mathbf{x}_j \end{bmatrix}, \text{ and} \\ \bar{V}_M^{(1)} &= \begin{bmatrix} 1 & \dots & 1 \\ \mathbf{e}_2^{(0)} & \dots & \mathbf{e}_p^{(0)} \end{bmatrix}. \end{aligned} \tag{8}$$

obtained after the feature reduction obtained from PCA. Note that $\left|\det(V_M^{(1)})\right| = V^{(1)}$ as $V_M^{(1)}$ is simply a column permutation of the matrix. Assume that we have computed the LU factorization (with partial pivoting) $P_M \bar{V}_M^{(1)} = L_M U_M$. Then, the LU factorization (with partial pivoting) of $V_M^{(1)}$ is simply given by $P_M V_M^{(1)} = [U_M(L_M^{-1} P_M^T \mathbf{x}_j)]$. Therefore, the LU factorizations required in the volume calculations of the N-FINDR algorithm

can be all computed by simply forming the $p \times s$ matrix $\hat{M} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ & & \hat{M}^T & \end{bmatrix}$, and then computing $L_M^{-1} P_M^T \hat{M}$. This is one of the parts that we accomplished in the GPU by means of a `VolumeCalculation` kernel to obtain the volume of each pixel for one iteration. The s volumes required in the first iteration of the N-FINDR algorithm are obtained from the product of the determinant of U_M times each one of the entries in the last row of $L_M^{-1} P_M^T \hat{M}$. By means of a `Reduction_vol` kernel, we get the value of the maximum volume and the coordinates of the pixel that produces such volume. Given that $s \gg p$, this implies a significant reduction of the computational complexity of the original algorithm.

3.3 GPU-UCLS

Our GPU version of the linear spectral unmixing-based abundance estimation algorithm uses the endmember set produced by the GPU-FINDR algorithm to produce a set of endmember abundance maps as follows:

1. The first step is to calculate a so-called compute matrix $\left(\mathbf{E}^T\mathbf{E}\right)^{-1}\mathbf{E}^T$, where $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{p}$ is formed by the p endmembers extracted by the GPU-FINDR. This compute matrix will be multiplied by all pixel vectors $\mathbf{x}_j$ in the original image. In our implementation, the compute matrix is calculated in the CPU mainly due to two reasons: 1) its computation is relatively fast, and 2) once calculated, the compute matrix remains the same throughout the whole execution of the code.
2. The compute matrix calculated in the previous step is now multiplied by each pixel $\mathbf{x}_j$ in the hyperspectral image, thus obtaining a set of abundance vectors $\mathbf{a}_j = [a_{1j}, a_{2j}, \cdots, a_{pj}]$, each containing the fractional abundances of the p endmembers in $\mathbf{x}_j$. This is accomplished in the GPU by means of an `Unmixing` kernel, in which the outcome of the unmixing process (i.e. the p endmember abundance maps) is produced.

4. EXPERIMENTAL RESULTS

4.1 Hyperspectral image data

The hyperspectral image scene used in experiments is the well-known AVIRIS Cuprite scene [see Fig. 3(a)], collected in the summer of 1997 and available online in reflectance units after atmospheric correction*. The portion used in experiments corresponds to a 350×350-pixel subset of the sector labeled as f970619t01p02_r02_sc03.a.rfl in the online data, which comprises 188 spectral bands in the range from 400 to 2500 nanometers and a total size of around 50 Megabytes. Water absorption and low SNR bands were removed prior to the analysis. The site is well understood mineralogically, and has several exposed minerals of interest including *alunite*, *buddingtonite*, *calcite*, *kaolinite* and *muscovite*. Reference ground signatures of the above minerals [see Fig. 3(b)], available in the form of a U.S. Geological Survey library (USGS)† will be used to assess endmember signature purity in this work.

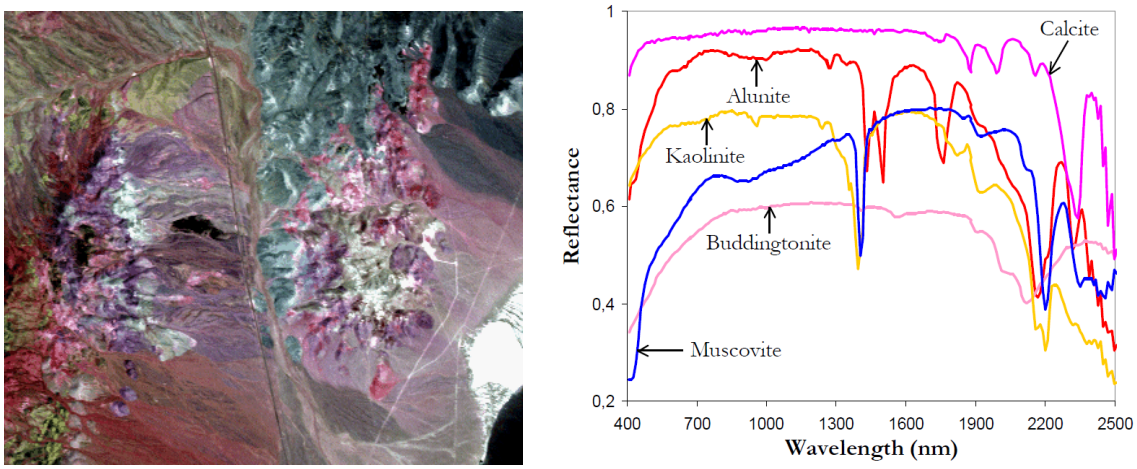
4.2 Analysis of algorithm precision

The full hyperspectral unmixing chain was first tested with the AVIRIS Cuprite scene. The number of endmembers to be detected was set to $p = 19$ after calculating the virtual dimensionality (VD) of the hyperspectral data.[8] This value was also used to decide the number of principal components to be retained by GPU-PCA. Table 1 shows the spectral angles[4] (in degrees) between the most similar endmember pixels detected by GPU-FINDR and the USGS library signatures. The lower the spectral angle, the more similar the spectral signatures are. The range of values for the spectral angle is $[0^o, 90^o]$. As shown by Table 1, the GPU-FINDR extracted endmembers which were very similar, spectrally, to the USGS reference signatures, despite the potential variations (due to possible interferers still remaining after the atmospheric correction process) between the ground signatures and the airborne data. Since no reference information is available regarding the true abundance fractions of minerals in the AVIRIS Cuprite data, no quantitative experiments were conducted although the obtained mineral maps exhibit similar correlation with regards to previously published maps‡. Since these results have been discussed in previous work (see for instance[17]), we do not display them here.

*http://aviris.jpl.nasa.gov

†http://speclab.cr.usgs.gov/spectral-lib.html

‡http://speclab.cr.usgs.gov/cuprite.html



(a) (b)

Figure 3. (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada. (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

Table 1. Spectral angle values (in degrees) between the pixels extracted by GPU-FINDR from the AVIRIS Cuprite scene and the USGS reference signatures.

Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite
$4{,}81^o$	$4{,}16^o$	$9{,}52^o$	$10{,}76^o$	$5{,}29^o$

4.3 Analysis of parallel performance

The proposed GPU implementation of the full hyperspectral unmixing chain has been tested on a Nvidia Tesla C1060 GPU, which features 240 processor cores operating at 1.296 Ghz, with single precision floating point performance of 933 Gflops, double precision floating point performance of 78 Gflops, total dedicated memory of 4 GB, 800 MHz memory (with 512-bit GDDR3 interface) and memory bandwidth of 102 GB/sec[§]. The GPU is connected to an Intel core i7 920 CPU at 2.67 Ghz with 8 cores, which uses a motherboard Asus P6T7 WS SuperComputer. It is important to emphasize that our GPU versions of PCA, N-FINDR and UCLS provide exactly the same results as the serial versions of the same algorithms, implemented using the Intel C/C++ compiler and optimized using several compilation flags to exploit data locality and avoid redundant computations. Hence, the only difference between the serial and parallel algorithms is the time they need to complete their calculations.

The serial algorithms were executed in one of the available cores, and the parallel times were measured in the considered GPU platform. For each experiment, fifty runs were performed and the mean values are reported (these times were always very similar, with differences on the order of a few milliseconds only). Table 2 summarizes the obtained results. It should be noted that the cross-track line scan time in AVIRIS, a push-broom instrument,[2] is quite fast (8.3 milliseconds to collect 512 full pixel vectors). This introduces the need to process the considered AVIRIS Cuprite scene (350×350 pixels and 188 spectral bands) in less than 1985 milliseconds to fully achieve real-time performance. It can be observed in Table 2 that the time measured on the Intel Core i7 920 CPU after applying the proposed linear algebra optimizations was 12168.19 milliseconds, while the execution time measured on the GPU was only 1718.61 milliseconds. This result is strictly in real-time. The C function GETTIMEOFDAY() was used for timing the CPU implementations, and the CUDA timer was used for the GPU implementations. It should be noted that the GPU implementation has been carefully optimized taking into account the specific parameters of each considered architecture, including the global memory available, the local shared memory in each multiprocessor, and also the local cache memories. Whenever possible, we have

[§]http://www.nvidia.com/object/product_tesla_c1060_us.html

Table 2. Processing times (milliseconds) and speedups achieved for the GPU implementations of PCA, N-FINDR and UCLS parts and for the full hyperspectral unmixing chain on an NVidia GPU Tesla C1060 architecture, tested with the AVIRIS Cuprite hyperspectral scene.

	PCA	**N-FINDR**	**UCLS**	**Full Chain**
Time serial	12168.19	1045.83	2146.73	15360.84
Time GPU	1718.61	143.33	73.66	1935.69
Speedup	7.08	7.30	29.14	7.94

accommodated blocks of pixels in small local memories in the GPU in order to guarantee very fast accesses, thus performing block-by-block processing to speed up the computations as much as possible.

5. CONCLUSIONS AND FUTURE LINES

The ever increasing spatial and spectral resolutions that will be available in the new generation of hyperspectral instruments for remote observation of the Earth anticipates significant improvements in the capacity of these instruments to uncover spectral signals in complex real-world analysis scenarios. Such capacity demands parallel processing techniques which can cope with the requirements of time-critical applications and properly scale with image size, dimensionality and complexity. In order to address such needs, we have developed a real-time implementation of a full hyperspectral unmixing chain using GPUs. The performance of the implementation has been evaluated (in terms of the quality of the solutions provided and its parallel performance) using an NVidia Tesla C1060 GPU. Experimental results indicate that real-time performance could be obtained using only one GPU device. Further improvements can be obtained by resorting to alternative strategies for dimensionality reduction as this step consumes a significant portion of GPU time (around 90%) in our implementation. Optimizing this part could also allow us to search for a higher number of endmembers in real time. In order to fully substantiate the aforementioned remarks, further experimentation with additional hyperspectral scenes and GPU architectures is also desirable.

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 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, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis, *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sensing of Environment* **65**(3), pp. 227–248, 1998.
3. G. Shaw and D. Manolakis, "Signal processing for hyperspectral image exploitation," *IEEE Signal Processing Magazine* **19**, pp. 12–16, 2002.
4. N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Processing Magazine* **19**(1), pp. 44–57, 2002.
5. Q. Du, N. Raksuntorn, N. H. Younan, and R. L. King, "End-member extraction for hyperspectral image analysis," *Applied Optics* **47**, pp. 77–84, 2008.
6. 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**(3), pp. 650–663, 2004.
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**(9), pp. 2025–2041, 2002.

8. C.-I. Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing* **42**(3), pp. 608–619, 2004.
9. 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.
10. 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.
11. K. J. Guilfoyle, M. L. Althouse, and C.-I. Chang, "A quantitative and comparative analysis of linear and nonlinear spectral mixture models using radial basis function neural networks," *IEEE Trans. Geosci. Remote Sens.* **39**, pp. 2314–2318, 2001.
12. A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*, Taylor & Francis: Boca Raton, FL, 2007.
13. A. Plaza and C.-I. Chang, "Special issue on high performance computing for hyperspectral imaging," *International Journal of High Performance Computing Applications* **22**(4), pp. 363–365, 2008.
14. A. Plaza, "Special issue on architectures and techniques for real-time processing of remotely sensed images," *Journal of Real-Time Image Processing* **4**(3), pp. 191–193, 2009.
15. A. Plaza, Q. Du, Y.-L-Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
16. A. Plaza, J. Plaza, A. Paz, and S. Sanchez, "Parallel hyperspectral image and signal processing," *IEEE Signal Processing Magazine* **28**(3), pp. 119–126, 2011.
17. S. Sanchez, A. Paz, G. Martin, and A. Plaza, "Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units," *Concurrency and Computation: Practice and Experience* **23**(12), 2011.
18. C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
19. S.-C. Wei and B. Huang, "GPU acceleration of predictive partitioned vector quantization for ultraspectral sounder data compression," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
20. H. Yang, Q. Du, and G. Chen, "Unsupervised hyperspectral band selection using graphics processing units," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
21. J. A. Goodman, D. Kaeli, and D. Schaa, "Accelerating an imaging spectroscopy algorithm for submerged marine environments using graphics processing units," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
22. E. Christophe, J. Michel, and J. Inglada, "Remote sensing processing: From multicore to GPU," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
23. J. Mielikainen, B. Huang, and A. Huang, "GPU-accelerated multi-profile radiative transfer model for the infrared atmospheric sounding interferometer," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
24. C.-C. Chang, Y.-L. Chang, M.-Y. Huang, and B. Huang, "Accelerating regular LDPC code decoders on GPUs," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), 2011.
25. D. A. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*, John Wiley & Sons: New York, 2003.
26. J. A. Richards and X. Jia, *Remote Sensing Digital Image Analysis: An Introduction*, Springer, 2006.
27. M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral endmember determination in hyperspectral data," *Proceedings of SPIE* **3753**, pp. 266–277, 1999.
28. M. Partridge and R. Calvo, "Fast dimensionality reduction and simple PCA," *Intelligent Data Analysis* **2**(3), pp. 1657–1672, 1998.