A Comparative Analysis of GPU Implementations of Spectral Unmixing Algorithms

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

Spectral unmixing is a very important task for remotely sensed hyperspectral data exploitation. It involves the separation of a mixed pixel spectrum into its pure component spectra (called endmembers) and the estimation of the proportion (abundance) of each endmember in the pixel. Over the last years, several algorithms have been proposed for: i) automatic extraction of endmembers, and ii) estimation of the abundance of endmembers in each pixel of the hyperspectral image. The latter step usually imposes two constraints in abundance estimation: the non-negativity constraint (meaning that the estimated abundances cannot be negative) and the sum-to-one constraint (meaning that the sum of endmember fractional abundances for a given pixel must be unity). These two steps comprise a hyperspectral unmixing chain, which can be very 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 perform an inter-comparison of parallel algorithms for automatic extraction of pure spectral signatures or endmembers and for estimation of the abundance of endmembers in each pixel of the scene. The compared techniques are implemented in graphics processing units (GPUs). These hardware accelerators can bridge the gap towards on-board processing of this kind of data. The considered algorithms comprise the orthogonal subspace projection (OSP), iterative error analysis (IEA) and N-FINDR algorithms for endmember extraction, as well as unconstrained, partially constrained and fully constrained abundance estimation. The considered implementations are inter-compared using different GPU architectures and hyperspectral data sets collected by the NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS).

Keywords: Hyperspectral imaging, spectral unmixing, 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. Linear spectral unmixing is a popular approach to characterize mixed pixels in hyperspectral images.[5] It can be simply defined as follows:

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

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

High-Performance Computing in Remote Sensing, edited by Bormin Huang, Antonio J. Plaza,
Proc. of SPIE Vol. 8183, 81830E · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.897329

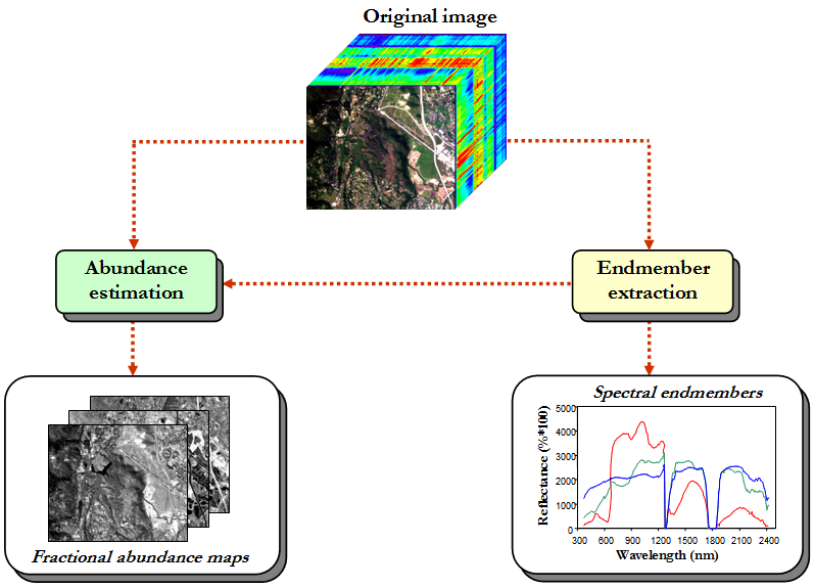


Figure 1. Hyperspectral unmixing chain.

where $\mathbf{x}$ is a hyperspectral pixel vector of the original hyperspectral image $\mathbf{X}$ given by a collection of values at different wavelengths, $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{p}$ is a matrix containing p 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. Two physical 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., $\sum_{i=1}^{p} a_i = 1$.[6] Solving the linear mixture model involves: 1) identifying a collection of $\{\mathbf{e}_i\}_{i=1}^{p}$ endmembers in the image, and 2) estimating their abundance in each pixel (see Fig. 1). The unmixing process is quite computationally expensive, due to the extremely high dimensionality of hyperspectral data cubes.[7–11]

Although the unmixing chain in Fig. 1 maps nicely to high performance computing systems such as commodity clusters,[11] 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)[12] 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.[13–19]

In this paper, we perform an inter-comparison of parallel algorithms for automatic extraction of endmembers and for estimation of the abundance of endmembers in hyperspectral images. Specifically, three classic algorithms for endmember extraction and three techniques for abundance estimation (unconstrained, partially constrained and fully constrained) are considered, and their possibility for real-time performance on NVidia GPUs is investigated. The proposed techniques have 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 outlines the different algorithms for endmember extraction and abundance estimation considered for implementation of the unmixing chain in this work. Section 3 describes the GPU implementation of these algorithms. 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. ALGORITHMS

2.1 Endmember extraction

Three classic algorithms for endmember extraction are considered in this work:

1. **Orthogonal subspace projection (OSP)**.[20] This algorithm starts by selecting the pixel vector with maximum length in the scene as the first endmember $\mathbf{e}_1 = \arg\left\{\max_i \sum_{i=1}^{s\times l} \mathbf{x}_i \cdot \mathbf{x}_i^T\right\}$, where s and l respectively denote the number of samples and the number of lines in the hyperspectral image (meaning that $s \times l$ is the total number of pixels) and the superscript 'T' denotes the vector transpose operation. The algorithm now assigns $\mathbf{U}_1 = [\mathbf{e}_1]$. Then, it looks for the pixel vector with the maximum absolute projection in the space orthogonal to the space linearly spanned by the initial pixel, and labels that pixel as the second endmember $\mathbf{e}_2 = \arg\left\{\max_i \left[\left(P^{\perp}_{\mathbf{U}_1}\mathbf{e}_i\right)^T \left(P^{\perp}_{\mathbf{U}_1}\mathbf{e}_i\right)\right]\right\}$, where $P^{\perp}_{\mathbf{U}_1} = \mathbf{I} - \mathbf{U}_1\left(\mathbf{U}_1^T\mathbf{U}_1\right)^{-1}\mathbf{U}_1^T$ is an orthogonal subspace projection operator. Now the algorithm assigns $\mathbf{U}_2 = [\mathbf{e}_1\mathbf{e}_2]$ and looks for a third endmember is found by applying an orthogonal subspace projector to the original image,[20] where the signature that has the maximum orthogonal projection in the space orthogonal to the space linearly spanned by the first two endmembers. This procedure is repeated until the desired number of endmembers, p, is found.[21]

2. **N-FINDR**.[22] 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 image pixels as endmembers $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \cdots, \mathbf{e}_p^{(0)}\}$. 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. This means that an initial volume is calculated as follows:

$$V(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \cdots, \mathbf{e}_p^{(0)}) = \frac{\left|\det\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{e}_1^{(0)} & \mathbf{e}_2^{(0)} & \cdots & \mathbf{e}_p^{(0)} \end{bmatrix}\right|}{(p-1)!}. \tag{2}$$

   For each pixel vector $\mathbf{x}_i$ in the input hyperspectral data, recalculate the volume by testing the pixel in all p endmember positions, i.e., first calculate $V(\mathbf{x}_i, \mathbf{e}_2^{(0)}, \cdots, \mathbf{e}_p^{(0)})$, then calculate $V(\mathbf{e}_1^{(0)}, \mathbf{x}_i, \cdots, \mathbf{e}_p^{(0)})$, and so on until $V(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \cdots, \mathbf{x}_i)$. If none of the p recalculated volumes is greater than $V(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(k0)}, \cdots, \mathbf{e}_p^{(0)})$, then no endmember is replaced. Otherwise, the combination with maximum volume is retained. The replacement step is repeated for all the pixel vectors in the input data until all the pixels have been exhausted and a maximum volume combination has been found. This combination gives the final set of p endmembers.

3. **Iterative error analysis (IEA)**.[23] In this algorithm, a series of fully constrained abundance estimation operations is performed to reconstruct the original image $\mathbf{X}$ using an approximation $\hat{\mathbf{X}}$ given by the linear mixture model in Eq. (1), each time selecting as endmembers the pixels that minimize the remaining error in the unmixed image. An initial vector (usually the mean spectrum of the data) is chosen to start the process. A fully constrained linear spectral unmixing in terms of this vector is performed, and the error image, formed by the errors remaining at each pixel after the unmixing operation, is calculated. The spectral vector corresponding to the pixel with the single largest error is found, and this pixel is now included in the endmember set. The process is repeated until p endmembers have been selected. A distinguishing feature with regards to OSP and N-FINDR is that IEA not only produces a set of endmembers but also their abundances in each pixel of the scene. As a result, it implements a full hyperspectral unmixing chain.

2.2 Abundance estimation

Three techniques for abundance estimation have been considered in this work:

1. **Unconstrained linear spectral unmixing (ULSU)**. Once a set of p endmembers $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^p$ has been identified using any of the algorithms discussed in the previous subsection, these endmembers can be used to unmix a certain pixel vector $\mathbf{x}$ by estimating the vector of abundances $\mathbf{a}$ containing the fractional abundances of each of the p endmembers in the pixel using the following expression:[6] $\mathbf{a} \approx (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T\mathbf{x}$. However, it should be noted that the fractional abundance estimation $\mathbf{a}$ does not satisfy the ASC or the ANC constraints.

2. **Non-negative linear spectral unmixing (NNLSU)**. Imposing the ANC constraint in abundance estimation results in the following optimization problem: $\min_{\mathbf{a}\in\Delta}\left\{(\mathbf{x}-\mathbf{a}\cdot\mathbf{E})^T(\mathbf{x}-\mathbf{a}\cdot\mathbf{E})\right\}$, subject to: $\Delta=\{\mathbf{a}|\sum_{i=1}^{p}a_i=1\}$. In order to solve this optimization problem, in this work we resort to the image space reconstruction algorithm (ISRA),[24] which imposes the ANC constraint while guaranteeing convergence in a finite number of iterations as well as positive values in the abundance estimation results for any input set of endmembers.

3. **Fully constrained linear spectral unmixing (FCLSU)**. Imposing the ASC constraint results in the optimization problem: $\min_{\mathbf{a}\in\Delta}\left\{(\mathbf{x}-\mathbf{a}\cdot\mathbf{E})^T(\mathbf{x}-\mathbf{a}\cdot\mathbf{E})\right\}$, subject to: $\Delta=\{\mathbf{a}|a_i\geq 0 \text{ for all } 1\leq i\leq p\}$. As indicated in,[6] a fully constrained abundance estimate (i.e. ASC-constrained and ANC-constrained) can be obtained by considering the two constraints simultaneously.

The endmember extraction and, particularly, the abundance estimation techniques described in this section are quite complex in computational terms, particularly if they are applied to unmix high-dimensional hyperspectral data sets. In the following section we describe GPU implementations for the endmember extraction and abundance estimation algorithms described in this section.

3. GPU IMPLEMENTATIONS

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. Once the hyperspectral image is mapped onto the GPU memory, a structure in which the number of blocks equals the number of lines in the hyperspectral image and the number of threads equals the number of samples is created, thus ensuring that as many pixels as possible are processed in parallel. The amount of pixels processed in parallel depends of the memory and register resources available in the GPU.

3.1 GPU Implementation of Endmember Extraction Algorithms

The GPU implementations of endmember extraction algorithms in this work can be summarized as follows:

1. **GPU implementation of OSP (GPU-OSP)**. In order to implement the OSP algorithm in GPUs, we first calculate the brightest pixel $\mathbf{e}_1$ in the original hyperspectral scene by means of a `CUDA` kernel (`CalculateBright`) which computes (in parallel) the dot product between each pixel vector $\mathbf{x}$ in the original hyperspectral image and its own transposed version $\mathbf{x}^T$. Then, the kernel `MaxBright` calculates $\mathbf{e}_1$ from the output provided by `CalculateBright`. Once the brightest pixel in the original hyperspectral image has been identified as the first endmember, the pixel (vector of bands) is allocated as the first column in the $\mathbf{U}_1$ matrix by $\mathbf{U}_1=[\mathbf{e}_1]$. A kernel (called `UtxU`) with `i` blocks and `i` threads is now applied to calculate $\mathbf{U}_1^T\mathbf{U}_1$, where `i` is the iteration number (between 1 and the desired number of endmembers, p). Now the algorithm calculates the inverse of the previous product using the Gauss-Jordan elimination

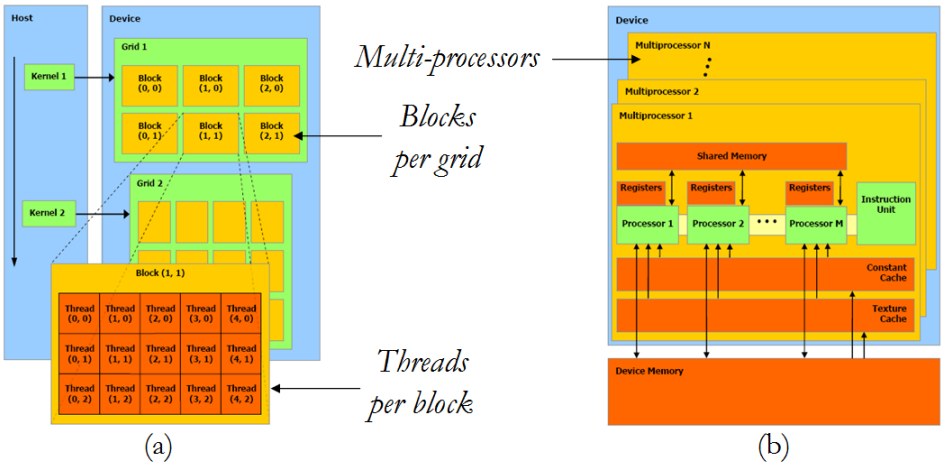


Figure 2. Schematic overview of a GPU architecture. (a) Threads, blocks and grids. (b) Execution model in the GPU.

method,[12] thus obtaining $\left(\mathbf{U}_1^T\mathbf{U}_1\right)^{-1}$. A new kernel (`Uxinv`) with `i` blocks and `num_bands` threads is now applied to multiply $\mathbf{U}_1$ and the inverse of the previous step, thus obtaining $\mathbf{U}_1\left(\mathbf{U}_1^T\mathbf{U}_1\right)^{-1}$. Another kernel (`AnsxUt`) with `num_bands` blocks and `num_bands` threads is then applied to multiply the previous result by $\mathbf{U}_1^T$, thus obtaining $\mathbf{U}_1\left(\mathbf{U}_1^T\mathbf{U}_1\right)^{-1}\mathbf{U}_1^T$. The subtraction of the identity matrix $\mathbf{I}$ is calculated by a kernel (`SustractIdentity`) with `num_bands` blocks and `num_bands` threads, thus obtaining the orthogonal subspace projector $P_{\mathbf{U}_1}^{\perp} = \mathbf{I} - \mathbf{U}_1\left(\mathbf{U}_1^T\mathbf{U}_1\right)^{-1}\mathbf{U}_1^T$. Finally, a new kernel (`PixelProjection`) in which the number of blocks equals the number of lines (l) in the hyperspectral image and the number of threads equals the number of samples (s) is applied to project the orthogonal subspace projector to each pixel $\mathbf{x}_i$ in the image. Finally, the maximum of all projected pixels is calculated using the kernel `MaxProjection`, which obtains the second endmember as follows: $\mathbf{e}_2 = \arg\left\{\max_i\left[\left(P_{\mathbf{U}_1}^{\perp}\mathbf{e}_i\right)^T\left(P_{\mathbf{U}_1}^{\perp}\mathbf{e}_i\right)\right]\right\}$. The GPU-OSP now extends the reference matrix $\mathbf{U}_2 = [\mathbf{e}_1\mathbf{e}_2]$ and repeats from step 3 until the desired number of endmembers (specified by the input parameter p) has been extracted. The output of the algorithm is a set of endmembers $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{p}$.

2. **GPU implementation of N-FINDR (GPU-FINDR)**. The most time-consuming computation in the N-FINDR algorithm is the calculation of the determinants needed for the volume computation. 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:

$$\begin{aligned} V_M^{(1)} &= \begin{bmatrix} 1 & \cdots & 1 & 1 \\ \mathbf{e}_2^{(0)} & \cdots & \mathbf{e}_p^{(0)} & \mathbf{x}_i \end{bmatrix}, \text{ and} \\ \bar{V}_M^{(1)} &= \begin{bmatrix} 1 & \cdots & 1 \\ \mathbf{e}_2^{(0)} & \cdots & \mathbf{e}_p^{(0)} \end{bmatrix}. \end{aligned} \tag{3}$$

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 $\hat{M} = \begin{bmatrix} 1\ 1\ \ldots\ 1 \\ \bar{M}^T \end{bmatrix}$, 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 $s \times l$ 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 $s \times l \gg p$, this implies a significant reduction of the computational complexity of the original algorithm.

3. **GPU implementation of IEA (GPU-IEA)**. The most time consuming step of the IEA algorithm is the calculation of spectral unmixings in iterative fashion as more endmembers become available, as well as the calculation of a reconstructed version $\hat{\mathbf{X}}$ of the original hyperspectral image $\mathbf{X}$ with more endmembers at each iteration. An `IEA` kernel implements unconstrained spectral unmixing and reconstruction in our GPU implementation of IEA. It first calculates a compute matrix given by $\left(\mathbf{E}^T\mathbf{E}\right)^{-1}\mathbf{E}^T$, where $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^q$ is formed by the q endmembers extracted by IEA until the current iteration. This matrix is be multiplied (in parallel) by all pixels $\mathbf{x}_i$ in the original image. Then, the reconstruction error is calculated. An iterative application of this kernel (using the mean spectrum of the original image the initial one and iterating until p endmembers are found) produces a GPU implementation of the IEA algorithm

3.2 GPU Implementation of Abundance Estimation Algorithms

The GPU implementations of abundance estimation algorithms in this work can be summarized as follows:

1. **GPU implementation of ULSU (GPU-ULSU)**. Our GPU version of the linear spectral unmixing-based abundance estimation algorithm uses a set of $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^p$ endmembers derived by a certain algorithm and produces a set of endmember abundance maps as follows. The first step is to calculate a compute matrix $\left(\mathbf{E}^T\mathbf{E}\right)^{-1}\mathbf{E}^T$. 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. The compute matrix is then multiplied by each pixel $\mathbf{x}_i$ in the hyperspectral image, thus obtaining a set of abundance vectors containing the fractional abundances of the p endmembers in $\mathbf{x}_i$. 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.

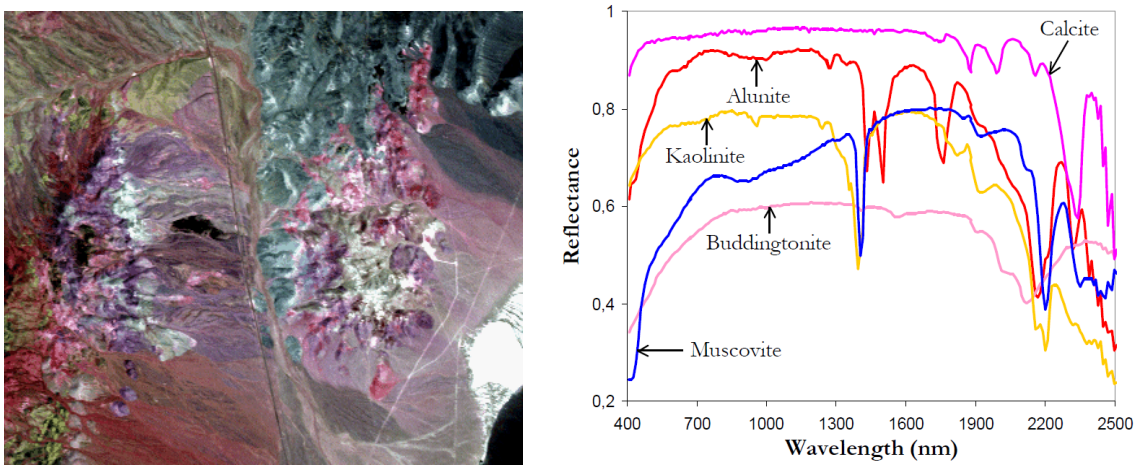
2. **GPU implementation of NNLSU (GPU-NNLSU) and FCLSU (GPU-FCLSU)**. In addition to the GPU-LSU, which does not incorporate the ANC and ASC unmixing constraints, we have also implemented the non-negative (NNLSU) and fully constrained (FCLSU) abundance estimation algorithms. Specifically, our GPU version of the NNLSU is obtained by a kernel called `ISRA`, which implements the corresponding algorithm for imposing the ANC constraint in abundance estimation. The algorithm converges in a finite number of iterations and provides positive values in the abundance estimation results for any input set of endmembers. In order to implement FCLSU we simply normalize the abundances provided by GPU-NNLSU to provide both the ANC and ASC constraints at the same time. This is accomplished by a kernel called `FCLSU`.

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

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



(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.

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.[25] Table 1 shows the spectral angles[4] (in degrees) between the most similar endmember pixels detected by the GPU implementations of OSP, N-FINDR and IEA 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 three algorithms 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. Overall, the most spectrally similar endmembers (on average) with regards to the USGS reference signatures were provided by the N-FINDR, but the three algorithms provided very similar results. Since no reference information is available regarding the true abundance fractions of minerals in the AVIRIS Cuprite data, no quantitative experiments could be conducted for the three considered abundance estimation techniques (UCLSU, NNLSU and FCLSU) although we observed that the obtained mineral maps exhibited similar correlation with regards to previously published maps[‡], particularly when the ASC and ANC constrants were imposed for the unmixing. Since these results have been discussed in previous work (see for instance[12]), we do not display them here.

4.3 Analysis of parallel performance

Before describing our parallel performance results, it is first important to emphasize that our GPU versions of OSP, N-FINDR, IEA, UCLSU, NNLSU and FCLSU all 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 a CPU Intel Q9450 with 4 cores, which uses a motherboard ASUS Striker II NSE (with NVidiaTM 790i

[†]http://speclab.cr.usgs.gov/spectral-lib.html
[‡]http://speclab.cr.usgs.gov/cuprite.html

Table 1. Spectral angle values (in degrees) between the pixels extracted by OSP, N-FINDR and IEA from the AVIRIS Cuprite scene and the USGS reference signatures (the results for the serial algorithms were exactly the same as those obtained by the corresponding GPU implementations).

Algorithm	Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite	Average
OSP	4.81^o	4.16^o	9.52^o	10.76^o	5.29^o	6.91^o
N-FINDR	4.81^o	4.29^o	7.60^o	9.92^o	5.05^o	6.33^o
IEA	4.81^o	4.29^o	7.60^o	12.23^o	6.03^o	6.99^o

Figure 4. The NVidia GeForce GTX 275 GPU used in experiments.

chipset) and 4 GB of RAM memory at 1333 MHz. The GPU used in experiments was the NVidia GeForce GTX 275 GPU (see Fig. 4), which features 240 processor cores operating at 1.550 GHz, 80 texture processing units, a 448-bit memory interface, and a 1792 Megabyte GDDR3 framebuffer at a 2520 MHz. It is based on the GT200 architecture[§].

Table 2 summarizes the obtained results in terms of parallel performance (the GPU and serial processing times are reported in each case) for each of the considered algorithms. In the endmember extraction algorithms (OSP, N-FINDR and IEA) we fixed the number of endmembers to be detected to $p = 19$. In all cases, the C function `clock()` was used for timing the CPU implementations, and the CUDA timer was used for the GPU implementations. The time measurement was started right after the hyperspectral image file was read to the CPU memory and stopped right after the results provided by the considered algorithm are stored in the CPU memory. The serial algorithms were executed in one of the available cores. For each experiment, fifty runs were performed and the mean values are reported on Table 2 (these times were always very similar, with differences on the order of a few milliseconds only). It should be noted that the GPU implementations have 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 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.

[§]http://www.nvidia.com/object/product_geforce_gtx_275_us.html

Table 2. Processing times (seconds) and speedups achieved for the OSP, N-FINDR, IEA, UCLSU, NNLSU and FCLSU implemented with the AVIRIS Cuprite image. It should be noted that the full unmixing chain considered in this work is made up of: 1) endmember extraction (implemented using one out of three possible algorithms: OSP, N-FINDR or IEA); and 2) abundance estimation (implemented using one out of three possible algorithms: LSU, NCLSU or FCLSU).

Algorithm	OSP	N-FINDR	IEA	UCLSU	NNLSU	FCLSU
Time serial (secs)	144.21	10.45	126.05	5.05	485.98	486.21
Time GPU (secs)	13.53	1.43	2.31	0.26	12.49	12.49
Speedup	10.65	7.30	54.56	19.42	38.90	38.92

From Table 2, we can observe that the slowest algorithm among the endmember extraction algorithms was the OSP, both in the serial and in the parallel versions. Quite opposite, the N-FINDR performed very fast in its serial version due to the matrix optimizations introduced to the algorithm in both its serial and parallel versions. Finally, the IEA was the algorithm that achieved the highest speedup when implemented in the GPU due to the acceleration achieved in the iterative unmixing steps performed by this algorithm. Among the abundance estimation methods, both the NNLSU and FLCSU required comparatively much more processing time than the UCLSU (both in the serial and in the parallel versions) due to the additional calculations required by imposing the ASC and ANC constraints.

If we analyze the possibility to apply a full unmixing chain (i.e., endmember extraction plus abundance estimation) in real-time, we should bear in mind 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 combination N-FINDR+UCLSU is the only one that can strictly perform in real-time (1690 milliseconds), while the IEA itself also performs close to real-time (note that this algorithm performs both endmember extraction and abundance estimation simultaneously). For the other tested algorithms, real-time performance could only be achieved by reducing the number of endmembers to be extracted. In any case, the performance of all methods can be significantly increased by resorting to GPU architectures, which provide a source of computational power which is very appealing for accelerating hyperspectral unmixing applications.

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 performed an inter-comparison of parallel implementations of algorithms for extraction of pure spectral signatures (endmembers) and for estimation of the abundance of endmembers in remotely sensed hyperspectral images. The compared techniques are implemented in graphics processing units (GPUs), which offer the potential to bridge the gap towards on-board processing of hyperspectral data. The performance of the implementation has been evaluated (in terms of the quality of the solutions provided and its parallel performance) using an NVidia GeForce GTX 275 GPU. Our experimental results indicate that real-time performance could be obtained using an appropriate combination of endmember extraction plus abundance estimation on the considered GPU device. In future work, we will further optimize the considered algorithms (and other methods not included in this work) in order to tune them for real-time performance using a relatively large number of endmembers.

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. 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.
6. 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.
7. A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*, Taylor & Francis: Boca Raton, FL, 2007.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection," *IEEE Transactions on Geoscience and Remote Sensing* **32**(4), pp. 779–785.
21. H. Ren and C.-I. Chang, "Automatic spectral target recognition in hyperspectral imagery," *IEEE Transactions on Aerospace and Electronic Systems* **39**(4), pp. 1232–1249, 2003.
22. M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral endmember determination in hyperspectral data," *Proceedings of SPIE* **3753**, pp. 266–277, 1999.
23. R. A. Neville, K. Staenz, T. Szeredi, J. Lefebvre, and P. Hauff, "Automatic endmember extraction from hyperspectral data for mineral exploration," *Proc. 21st Canadian Symp. Remote Sens.* , pp. 21–24, 1999.
24. A. R. D. Pierro, "On the relation between ISRA and the EM algorithm for positron emission tomography," *IEEE Transactions on Medical Imaging* **12**, pp. 328–333, 1993.
25. 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.