J Real-Time Image Proc (2014) 9:397–405
DOI 10.1007/s11554-012-0276-3

SPECIAL ISSUE

Fast determination of the number of endmembers for real-time hyperspectral unmixing on GPUs

Sergio Sánchez · Antonio Plaza

Received: 25 April 2012 / Accepted: 4 September 2012 / Published online: 10 October 2012

Abstract Spectral unmixing is a very important task for remotely sensed hyperspectral data exploitation. It amounts at identifying a set of spectrally pure components (called *endmembers*) and their associated per-pixel coverage fractions (called *abundances*). A challenging problem in spectral unmixing is how to determine the number of endmembers in a given scene. Several automatic techniques exist for this purpose, including the virtual dimensionality (VD) concept or the hyperspectral signal identification by minimum error (HySime). Due to the complexity and high dimensionality of hyperspectral scenes, these techniques are computationally expensive. In this paper, we develop new fast implementations of VD and HySime using commodity graphics processing units. The proposed parallel implementations are validated in terms of accuracy and computational performance, showing significant speedups with regards to optimized serial implementations. The newly developed implementations are integrated in a fully operational unmixing chain which exhibits real-time performance with regards to the time that the hyperspectral instrument takes to collect the image data.

Keywords Hyperspectral imaging · Spectral unmixing · Endmembers · Graphics processing units (GPUs)

S. Sánchez · A. Plaza (✉)
Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politecnica de Cáceres, University of Extremadura, Cáceres, Spain
e-mail: aplaza@unex.es

S. Sánchez
e-mail: sersanmar@unex.es

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 μm) of reflected light in an area 2–12 km wide and several kilometers long, using 224 spectral bands [2]. The resulting multidimensional data cube typically comprises several GBs per flight (see Fig. 1). The wealth of spectral information provided by latest-generation hyperspectral imaging instruments has opened ground-breaking perspectives in many applications [3], many of which require real-time response of algorithm analysis [4–6].

Spectral unmixing has been an alluring exploitation goal from the earliest days of hyperspectral imaging [1] to our days [7, 8]. No matter the spatial resolution, the spectral signatures collected in natural environments are invariably a mixture of the signatures of the various materials found within the spatial extent of the ground instantaneous field view of the imaging instrument [9]. For instance, the pixel vector labeled as "vegetation" in Fig. 1 may actually be a mixed pixel comprising a mixture of vegetation and soil, or different types of soil and vegetation canopies. Linear spectral unmixing is a standard technique for spectral unmixing that infers a set of pure spectral signatures, called *endmembers* [10, 11], and the fractions of these endmembers, called *abundances* [12], in each pixel of the scene. This model assumes that the spectra collected by the imaging spectrometer can be expressed in the form of a linear combination of endmembers, weighted by their

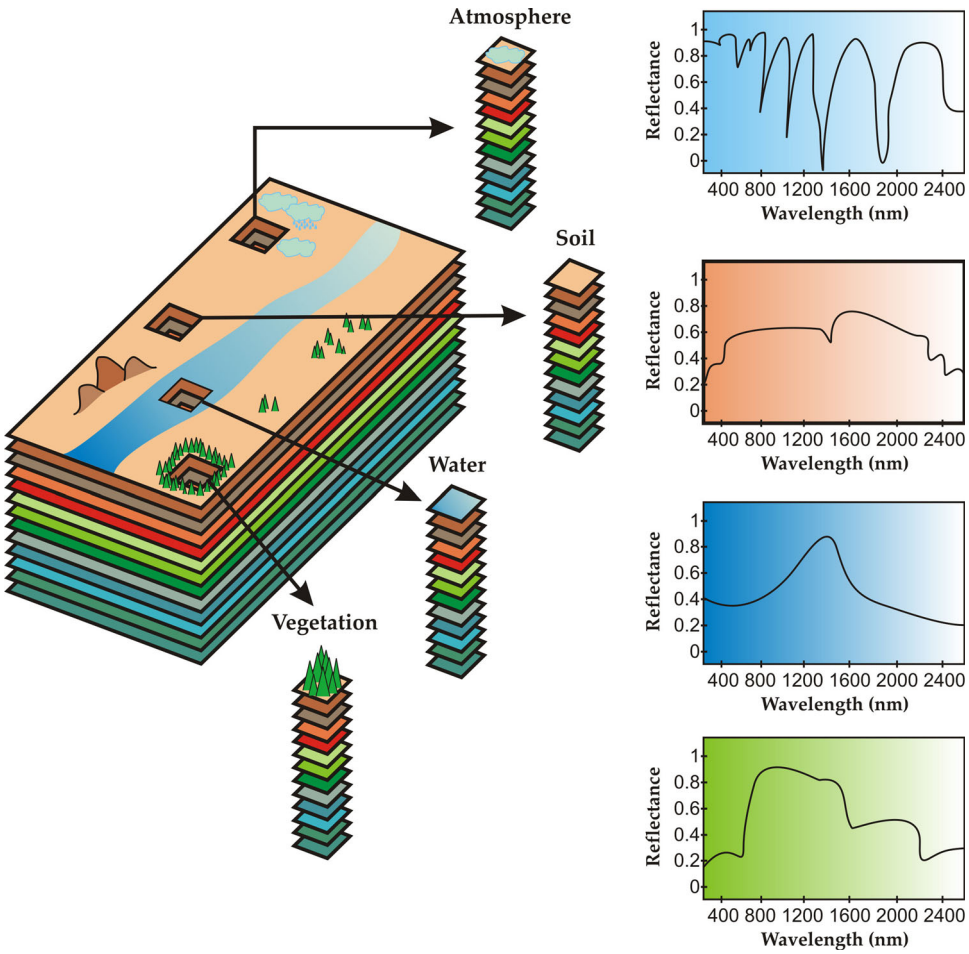

Fig. 1 The concept of remotely sensed hyperspectral imaging

corresponding abundances. In turn, nonlinear unmixing describes mixed spectra (in physical [13], or statistical [14] sense) by assuming that part of the source radiation is multiply scattered. In the following, we adopt the linear model due to practical advantages such as ease of implementation and flexibility in different applications [15].

The solution of the linear unmixing model relies on two major requirements: (1) a successful estimation of how many endmembers are present in the input hyperspectral scene, and (2) an accurate identification of such endmembers. While a significant number of techniques have been developed for endmember identification purposes (see [10, 11], among many others), fewer approaches have been developed for automatically estimating the number of endmembers. In this paper, we focus on this part of the spectral unmixing chain which is crucial for the subsequent endmember identification and abundance estimation steps. Two of the most successful approaches for estimating the number of endmembers are the virtual dimensionality (VD) concept [16] and the hyperspectral signal identification by minimum error (HySime) [17]. Depending on the complexity and dimensionality of the hyperspectral scene, the aforementioned techniques may be computationally very expensive. This fact limits the possibility of utilizing those algorithms in time-critical applications [18]. Despite the growing interest in parallel hyperspectral imaging research [5], no parallel implementations of techniques for automatic estimation of the number of endmembers (a critical component of the spectral unmixing chain) exist in the open literature. However, with the recent explosion in the amount and dimensionality of hyperspectral imagery, parallel versions of spectral unmixing algorithms are expected to become relevant in many remote sensing missions.

In this paper, we develop new parallel implementations of VD and HySime for automatically determining the number of endmembers in hyperspectral images using commodity graphics processing units (GPUs), a low-weight hardware platform that now offers a tremendous potential to bridge the gap towards real-time analysis of remotely sensed hyperspectral data [6, 19–21]. We further demonstrate that our implementations can be integrated in a fully operational unmixing chain which exhibits real-time performance for the first time in the literature. The remainder of the paper is organized as follows. Section 2 briefly describes the VD and HySime methods. Section 3 describes their parallel implementation in GPUs. Section 4

analyzes the accuracy and computational performance of the developed implementations in the framework of a fully operational, real-time unmixing chain. Section 5 concludes the paper with some remarks and future lines.

2 Methods

2.1 VD method

Let us define $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$ as a hyperspectral image with N pixel vectors, each with L spectral bands. The VD first calculates the eigenvalues of the covariance matrix $\mathbf{K}_{L\times L} = 1/N(\mathbf{Y} - \overline{\mathbf{Y}})^T(\mathbf{Y} - \overline{\mathbf{Y}})$ and the correlation matrix $\mathbf{R}_{L\times L} = \mathbf{K}_{L\times L} + \overline{\mathbf{Y}}\overline{\mathbf{Y}}^T$, referred to as covariance-eigenvalues and correlation-eigenvalues, for each of the spectral bands in the original hyperspectral image $\mathbf{Y}$. The VD concept follows the "pigeon-hole principle". If we represent a signal source by a pigeon and a spectral band by a hole, we can use a spectral band to accommodate one source. Thus, if a distinct spectral signature makes a contribution to the eigenvalue-represented signal energy in one spectral band, then its associated correlation eigenvalue will be greater than its corresponding covariance eigenvalue in this particular band. Otherwise, the correlation eigenvalue would be very close to the covariance eigenvalue, in which case only noise would be present in this particular band. By applying this concept, a Neyman–Pearson detector [16] is introduced to formulate the issue of whether a distinct signature is present or not in each of the spectral bands of $\mathbf{Y}$ as a binary hypothesis testing problem. Here, the decision is made based on an input parameter of the algorithm which is called the false alarm probability or P_{F}, which is used to establish the sensitivity of the algorithm in terms of how much error can be tolerated in the identification of the actual number of endmembers in the image data. With this interpretation in mind, the issue of determining an estimate $\hat{p}$ for the number of endmembers is further simplified and reduced to a specific value of P_{F} that is preset by the Neyman–Pearson detector.

2.2 HySime method

The HySime method consists of two parts. Algorithm 1 describes the noise estimation part, which obtains a $N \times L$ matrix $\widehat{\xi}$ containing an estimation of the noise present in the original hyperspectral image $\mathbf{Y}$ [17]. This algorithm follows an approach which addresses the high correlation exhibited by close spectral bands. The main advantage of Algorithm 1 is that the computational complexity is substantially lower than that of other algorithms for noise estimation in hyperspectral data in the literature. Additional details about Algorithm 1 can be found in [17] and we do not repeat them for space considerations. On the other hand, Algorithm 2 describes the signal subspace identification part of the algorithm, which first computes the noise correlation matrix $\widehat{\mathbf{R}}_n$ and then computes the signal correlation matrix $\widehat{\mathbf{R}}_x$. Next, the eigenvectors of the signal correlation matrix are obtained and sorted in ascending order. Finally, a minimization function is applied to obtain an estimate $\hat{p}$ of the number of endmembers in the subspace $\widehat{\mathbf{X}}$. The main purpose of this algorithm is to select the subset of eigenvectors that best represents the signal subspace in the minimum mean squared error sense.

Algorithm 1 Noise estimation

1: **INPUT**: $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$
2: $\mathbf{Z} := \mathbf{Y}^T$, $\widehat{\mathbf{R}} := (\mathbf{Z}^T\mathbf{Z})$
3: $\mathbf{R}' := \widehat{\mathbf{R}}^{-1}$
4: **for** i:=1 to L **do**
5: $\quad \widehat{\beta}_i := ([\mathbf{R}']_{\alpha_i,\alpha_i} - [\mathbf{R}']_{\alpha_i,i}[\mathbf{R}']_{i,\alpha_i}/[\mathbf{R}']_{i,i})[\widehat{\mathbf{R}}]_{\alpha_i,i}$
$\quad$ {Note that $\alpha_i = 1, \ldots,$ i-1, i+1, $\ldots, L$ }
6: $\quad \widehat{\xi}_i := \mathbf{z}_i - \mathbf{Z}_{\alpha_i}\widehat{\beta}_i$
7: **end for**
8: **OUTPUT**: $\widehat{\xi}$ {$N \times L$ matrix with the estimated noise}

Algorithm 2 Signal subspace estimation

1: **INPUTS**: $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$, $\widehat{\mathbf{R}}_y \equiv (\mathbf{Y}\mathbf{Y}^T)/N$
2: $\widehat{\mathbf{R}}_n := 1/N\sum_i (\widehat{\xi}_i\widehat{\xi}_i^T)$
3: $\widehat{\mathbf{R}}_x := 1/N\sum_i ((\mathbf{y}_i - \widehat{\xi}_i)(\mathbf{y}_i - \widehat{\xi}_i^T))$ {estimate of $\widehat{\mathbf{R}}_x$}
4: $\mathbf{E} := [\mathbf{e}_1, \ldots, \mathbf{e}_L]$ {$\mathbf{e}_i$ are eigenvectors of $\widehat{\mathbf{R}}_x$}
5: $\delta := [\delta_1, \ldots, \delta_L]$
6: $(\widehat{\delta}, \widehat{\pi}) := \mathrm{sort}(\delta)$
$\quad$ {sort δ_i by ascending order; save the permutation $\widehat{\pi}$}
7: $\widehat{p}$:=number of terms $\widehat{\delta}_i < 0$
8: $\widehat{\mathbf{X}} = \langle[\widehat{\mathbf{e}}_{i1}, \cdots, \widehat{\mathbf{e}}_{ip}]\rangle$

3 GPU implementations

In this section, we describe the newly developed GPU implementation of VD and HySime, carried out using the compute unified device architecture (CUDA)[1] developed by NVidia™. The architecture of a GPU can be seen as a set of multiprocessors (MPs). Each multiprocessor is characterized by a single instruction multiple data (SIMD) architecture, i.e., in each clock cycle, each processor 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. 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

[1] http://www.nvidia.com/object/cuda_home_new.html.

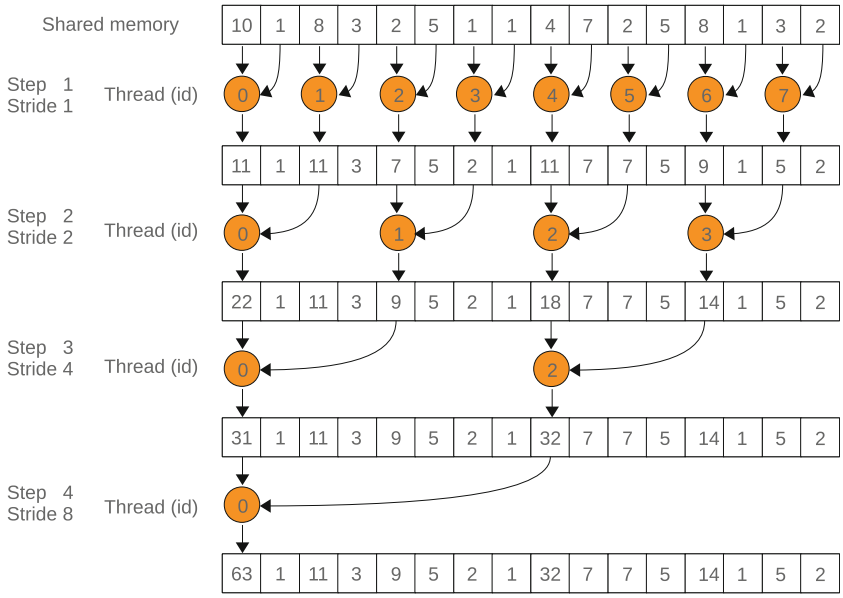

Fig. 2 Reduction process in shared memory

are constructed by chaining so-called *kernels* which operate on entire streams and which are executed by a multiprocessor, 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, where each block is composed by a group of threads that share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory. As a result, there are different levels of memory in the GPU for the thread, block and grid concepts. While the number of threads that can run in a single block is limited, the number of threads that can be concurrently executed is much larger as several blocks can be executed in parallel. This comes at the expense of reducing the cooperation between threads since threads in different blocks cannot synchronize among themselves. In the following, we describe the different steps that we followed for the development of the GPU versions of VD and HySime.

3.1 GPU implementation of VD

Once we load the hyperspectral image $\mathbf{Y}$ pixel by pixel in the main memory of the GPU, the first step is to compute the covariance matrix $\mathbf{K}_{L \times L}$. For this purpose, we need to calculate the mean value of each band of the image and substract this mean value to all the pixels in the same band. To perform this calculation in the GPU, we use a kernel called `meanSub` configured with as many blocks as the number of bands L in the hyperspectral image. In each block, all available threads perform a reduction process using shared memory and coalesced memory accesses to add the values of all the pixels in the same band. Figure 2 illustrates the adopted reduction process in graphical form.

Once this process is completed, another thread divides the computed value by the number of pixels in the original image, N, and the mean value is obtained. The resulting mean values of each band $\overline{\mathbf{Y}}$ are stored in a structure as they will be needed for the calculation of the correlation matrix $\mathbf{R}_{L \times L}$. Next, all threads subtract the corresponding mean values from every pixel in the same band, obtaining the matrix $\mathbf{Y} - \overline{\mathbf{Y}}$ which is used to finalize the calculation of the covariance matrix in the GPU by means of a matrix multiplication operation $(\mathbf{Y} - \overline{\mathbf{Y}})^T(\mathbf{Y} - \overline{\mathbf{Y}})$. This operation is performed using the cuBLAS library,[2] an implementation of basic linear algebra subprograms (BLAS) on top of NVidia™ CUDA. Specifically, we use the `cublasSgemm` function of cuBLAS. The next step is to calculate the correlation matrix $\mathbf{R}_{L \times L}$ in the GPU. To achieve this, we use a kernel `Corr` which launches as many threads as elements are present in $\mathbf{R}_{L \times L}$, where each thread computes an element of the resulting matrix as follows: $\mathbf{R}_{ij} = \mathbf{K}_{ij} + \overline{\mathbf{Y}}_i\overline{\mathbf{Y}}_j$. Finally, we have observed that the remaining steps in the VD calculation (i.e., extraction of correlation-eigenvalues, covariance-eigenvalues and Neyman-Pearson test for estimation of the number of endmembers) can be computed very fast in the CPU.

3.2 GPU implementation of HySime

After loading the hyperspectral image $\mathbf{Y}$ in the GPU, we first implement the noise estimation part of HySime (see Algorithm 1). For this part, the first step is to compute $\widehat{\mathbf{R}}$

[2] http://developer.nvidia.com/cuBLAS.

and its inverse, $\mathbf{R}'$. The former is calculated in the GPU by means of a standard cuBLAS matrix multiplication (using the `cublasSgemm` function) while the latter is implemented in the CPU to avoid the high computational cost of the inverse operation in parallel. Then, the `for` loop in step 4 of Algorithm 1 repeats the same set of operations for each band of the hyperspectral image. Since the operations in this loop are independent, and all of them calculate a part of the final matrix $\widehat{\xi}$ which provides the noise estimation matrix, we compute these operations in parallel in the GPU using two kernels and a big matrix–matrix multiplication using cuBLAS. The first kernel, `Compute_Beta`, performs the computation of all the $\widehat{\beta}_i$ and stores them in a matrix $\widehat{\beta}$ of size $L \times L$. The main idea of this kernel is to assign one iteration of the `for` loop to one block of a compute grid. Thus, for this kernel, we have as many blocks as spectral bands in the original image, L. Once we have calculated the matrix $\widehat{\beta}$, the next step is to compute the operation $\widehat{\xi} := \mathbf{z} - \mathbf{Z}_\alpha \widehat{\beta}$. For this purpose, the multiplication $\mathbf{P} = \mathbf{Z}_\alpha \widehat{\beta}$ is done using cuBLAS, followed by the calculation of $\widehat{\xi} := \mathbf{z} - \mathbf{P}$. We can take advantage of this operation to substract each element of $\mathbf{P}$ from each element of $\mathbf{z}$ in parallel. For this purpose, we use a kernel `Compute_elements`. This is a simple kernel in which each thread performs a simple substraction operation, and with as many threads as elements in $\widehat{\xi}$.

The second part of the HySime algorithm is the signal subspace estimation (see Algorithm 2). The first step of this part is the calculation of the noise correlation matrix $\widehat{\mathbf{R}}_n$. This could be performed as a standard matrix multiplication in cuBLAS. However, in the next steps of the algorithm, we only consider the diagonal elements of this matrix. Hence, it is not necessary to compute the rest of the values. As a result, we just need to compute $\widehat{\mathbf{R}}_{n_{i,i}}$. To achieve this, we use a kernel called `CorrHS` to compute these elements in parallel. Since the set given by the i-th element of different $\widehat{\xi}$ vectors provides elements located in consecutive positions, we only need to compute L values. Hence, we use L threads to ensure coalesced accesses to memory. As a result, each thread reads N elements to perform the computation of an element of the diagonal of matrix $\widehat{\mathbf{R}}_n$.

The second step of Algorithm 2 is to calculate the signal correlation matrix $\widehat{\mathbf{R}}_x$. For this purpose, we start by performing the operation $(\mathbf{y}_i - \widehat{\xi}_i^T)$ using a kernel called `Compute_elements`, which substracts two structures of the same size using as many threads as elements in the structures. After this, we use again cuBLAS matrix multiplication to obtain $\widehat{\mathbf{R}}_x$. Step 4 of Algorithm 2 performs an eigen-decomposition of $\widehat{\mathbf{R}}_x$ to calculate the eigenvectors of the signal correlation matrix, and steps 5–6 sort them in ascending order. This operation is performed in the CPU using an optimzied function (`dsyevr_`) of the widely used linear algebra package (LAPACK),[3] which provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The terms $\widehat{\delta}_i$ are obtained based on the quadratic forms given by $p_{ij} = e_{ij}^T \widehat{\mathbf{R}}_y e_{ij}$ and $\sigma_{ij}^2 = e_{ij}^T \widehat{\mathbf{R}}_n e_{ij}$. We perform these operations using cuBLAS matrix multiplication. Finally, we apply a minimization function to get the number of different materials $\widehat{p}$ in the CPU. Since we only need the number of materials, we do not retrieve the signal subspace $\widehat{\mathbf{X}}$.

To conclude this section, we emphasize that our implementations have been carefully designed to exploit as much as possible the CPU and the GPU concurrently. In other words, in our parallel implementations, there is no need for the GPU to wait for the CPU to finish and vice versa as they operate concurrently whenever there are no data dependences involved. This aspect has been carefully optimized in our implementation to optimize the performance of parallel calculations taking advantage of both the CPU and the GPU.

4 Experimental results

4.1 Hyperspectral image data

The first data set used in our experiments was collected by the AVIRIS sensor over the World Trade Center (WTC) area in New York City on 16 September 2001, just 5 days after the terrorist attacks that collapsed the two main towers and other buildings in the WTC complex. The data set consists of 614 $\times$ 512 pixels, 224 spectral bands and a total size of (approximately) 140 MB. The spatial resolution is 1.7 m per pixel. The leftmost part of Fig. 3 shows a false-color composite of the data set selected for experiments using the 1,682, 1,107 and 655 nm channels, displayed as red, green and blue, respectively. Vegetated areas appear green in the leftmost part of Fig. 3, while burned areas appear dark gray. Smoke coming from the WTC area (in the red rectangle) and going down to south Manhattan appears bright blue due to high spectral reflectance in the 655 nm channel. Extensive reference information, collected by US Geological Survey (USGS), is available online for the WTC scene. The rightmost part of Fig. 3 shows the target locations of the thermal hot spots at the

[3] http://www.netlib.org/lapack/.

Fig. 3 False color composition of an AVIRIS hyperspectral image collected by NASA's Jet Propulsion Laboratory over the World Trade Center (WTC) on 16 Sept. 2001 (*left*). Location of thermal hot spots in the fires observed in the WTC area, available online: http://pubs.usgs.gov/of/2001/ofr-01-0429/hotspot.key.tgif.gif (*right*)

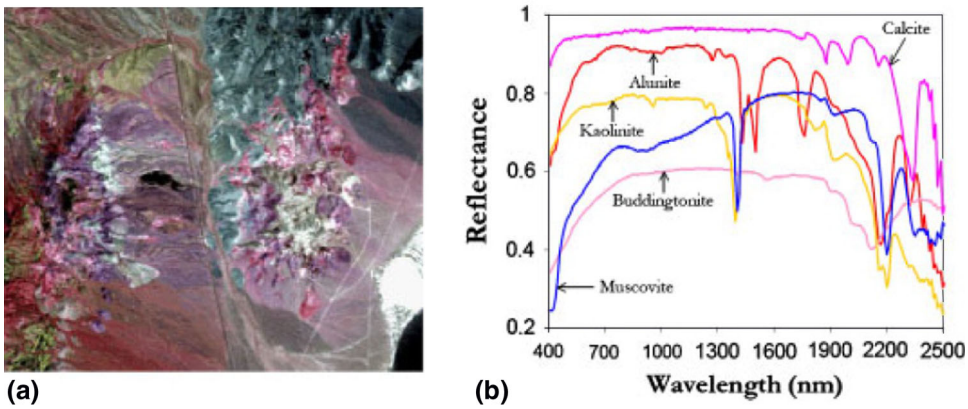

Fig. 4 **a** False color composition of the AVIRIS cuprite hyperspectral scene and **b** US Geological Survey mineral spectral signatures used for validation purposes

WTC area, displayed as bright red, orange and yellow spots.

The second data set used in experiments was collected by the AVIRIS sensor over the cuprite mining district in Nevada in the summer of 1997 (see Fig. 4). It is available online (in reflectance units) after atmospheric correction.[4] 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 comprise 188 spectral bands in the range from 400 to 2,500 nm after removing water absorption bands as well as bands with low signal-to-noise ratio (SNR) from the original set of 224 spectral bands, which resulted in a total size of around 50 MB. 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, available in the form of a USGS library[5] have been used in the literature for evaluation purposes [15].

For illustrative purposes, Table 1 provides the values of $\widehat{p}$ (number of endmembers) estimated by the VD method for the two considered hyperspectral scenes, using different values of the false-alarm probability (P_F). The number of endmembers estimated by HySime was $\widehat{p} = 23$ for the WTC scene and $\widehat{p} = 16$ for the cuprite scene. As shown in Table 1, the best compromise between VD and HySime was generally observed for values of $P_F = 10^{-7}$ or $P_F = 10^{-8}$.

4.2 Performance assessment

The GPU platform used in our experiments is the NVidia™ GeForce GTX 580 GPU,[6] which features 512 processor cores operating at 1.544 GHz, with single precision

[4] http://aviris.jpl.nasa.gov/data/free_data.html.

[5] http://speclab.cr.usgs.gov/spectral-lib.html.

[6] http://www.nvidia.com/object/product-geforce-gtx-580-us.html.

Table 1 VD estimates of $\widehat{p}$ for various false-alarm probabilities (P_F)

P_F	Cuprite	WTC
10^{-1}	37	59
10^{-2}	28	44
10^{-3}	25	33
10^{-4}	20	31
10^{-5}	19	30
10^{-6}	19	27
10^{-7}	17	23
10^{-8}	16	21

HySime provided an estimate of $\widehat{p} = 16$ for the cuprite scene and of $\widehat{p} = 23$ for the World Trade Center (WTC) scene

Table 2 Processing times (s) and speedups achieved for the GPU implementations of VD and HySime on an NVidia™ GPU GTX 580 architecture

	Cuprite		WTC	
	VD	HySime	VD	HySime
Time CPU	26.66	119.19	96.45	421.98
Time GPU	0.43	1.29	1.07	2.81
Speedup	60.82	92.05	89.48	149.82

floating point performance of 1,354 Gflops, double precision floating point performance of 198 Gflops, total dedicated memory of 1,536 MB, 2,004 MHz memory (with 384-bit GDDR5 interface) and memory bandwidth of 192.4 GB/s. 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. Before analyzing the parallel performance of the proposed GPU implementations, we emphasize that our parallel versions provide exactly the same results as the corresponding serial versions, executed in one of the cores of the i7 920 CPU and implemented using the `gcc` (gnu compiler default) of the C programming language (with optimization flag `--O3` to exploit data locality and avoid redundant computations). As a result, the only difference between the serial and parallel versions is the time they need to complete their calculations. The C function GETTIMEOFDAY() was used for timing the CPU implementations, and the CUDA timer was used for the GPU implementations.

Table 2 summarizes the obtained results by the CPU and GPU implementations. The reported GPU times are the mean of 50 executions in each platform (the measured times were always very similar, with differences—if any—on the order of only a few milliseconds). It is important to note that the GPU processing times of our implementation of VD do not vary for different values of P_F. As shown in Table 2, relevant speedups were obtained for both methods (VD and HySime) and hyperspectral scenes (WTC and Cuprite). It is also remarkable that the obtained speedups increase with the image size, and that the final processing times are strictly in real-time in all cases. This is because the data aqcuisition ratio by the AVIRIS sensor is known and we have used this information in order to determine if the proposed parallel algorithms could be applied at the same time as the data are collected without delaying the collection procedure at the sensor. Specifically, the cross-track line scan time in AVIRIS, a push-broom instrument [22], is quite fast (8.3 ms to collect 512 full pixel vectors). This introduces the need to process the WTC scene in $<$5.09 s (and the cuprite scene in $<$1.98 s) to fully achieve real-time performance. Although not reported here due to space considerations, we have conducted additional experiments and experimentally observed that our proposed parallel implementations scale linearly with regards to the hyperspectral image size and with regards to the number of spectral bands available.

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. Although the global GPU memory (1.5 GB) is large enough to store the full hyperspectral image in all considered cases, we accommodated blocks of pixels in small local memories in the GPU in order to guarantee very fast accesses. Also, we emphasize that the times of the data transfers between CPU and GPU (including the times for loading the image and writing the final results) are included in the GPU times reported in Table 2.

Table 3 Processing times (s) and speedups achieved for the GPU implementations of VD, Hysime, N-FINDR and UCLS on an NVidia™ GPU GTX 580 architecture

	Cuprite				WTC			
	VD	HySime	N-FINDR	UCLS	VD	HySime	N-FINDR	UCLS
Time CPU	26.66	119.19	31.38	3.06	96.45	421.98	124.25	14.77
Time GPU	0.43	1.29	0.15	0.04	1.07	2.81	0.58	0.18
Speedup	60.82	92.05	209.05	63.62	89.48	149.82	210.71	82.04

To conclude this section, we analyze the operational capacity of our GPU implementations of VD and HySime to perform as part of a full hyperspectral data processing chain. For this purpose, we applied a full spectral unmixing chain made up of the GPU implementations of VD or HySime, followed by a GPU implementation of the well-known N-FINDR algorithm [23] for endmember extraction, and by a GPU implementation of an unconstrained least-squares (UCLS) technique for fractional abundance estimation [24]. We experimentally tested that, when applied to the WTC and Cuprite scenes, the resulting VD + N-FINDR + UCLS and HySime+N-FINDR+ UCLS chains provided good fractional abundance estimations for the reference targets in Fig. 3, and also endmembers with high spectral similarity (measured in terms of the spectral angle [12]) with regards to the reference spectral signatures in Fig. 4b, respectively. Most importantly, these full and unsupervised unmixing chains could perform in real-time. Specifically, the VD + N-FINDR + UCLS chain was able to process the WTC scene in 1.83 s and the Cuprite scene in 0.62 s. On the other hand, the HySime+N-FINDR+UCLS chain was able to process the WTC scene in 3.57 s and the Cuprite scene in 1.48 s (for illustrative purposes, Table 3 summarizes the obtained results by the CPU and GPU implementations of different hyperspectral unmixing chains that can be formed after combining VD or HySime with N-FINDR for endmember extraction and UCLS for abundance estimation). The results in Table 3 demonstrate that the proposed implementations can be used as modules of a fully operational unmixing chain, able to provide real-time unmixing results in completely unsupervised fashion.

5 Conclusions and future research

In this paper, we have discussed the optimization of an important part of spectral unmixing chain for hyperspectral image analysis: the estimation of the number of spectrally pure components (endmembers). Even though other parts of the unmixing chain (such as the extraction of endmembers, or the estimation of fractional abundances) have already been implemented in parallel, here we develop new parallel implementations of two popular methods for estimating the number of endmembers: VD and HySime. Our experimental results reveal that the proposed techniques can achieve significant speedups in GPUs (an inexpensive, portable and low-weight parallel computing platform). Most importantly, the proposed implementations achieve real-time performance, both as separate processing modules and also as part of a full hyperspectral unmixing chain. As a result, this contribution reports, for the first time in the literature, a real-time implementation of a full hyperspectral unmixing chain for remotely sensed hyperspectral images using GPUs. Although the results reported in this paper are very encouraging, GPUs are still rarely exploited in real missions due to power consumption and radiation tolerance issues, despite improvements in these directions are expected in upcoming years. Currently, we are also experimenting with field programmable gate arrays in order to be able to adapt the proposed algorithms to hardware devices that can be mounted onboard hyperspectral imaging instruments after space qualification by international agencies.

References

1. Goetz, A.F.H.,Vane, G., Solomon, J.E., Rock, B.N.: Imaging spectrometry for Earth remote sensing. Science. **228**, 1147–1153 (1985)
2. Green, R.O.: Imaging spectroscopy and the airborne visible infrared imaging spectrometer (AVIRIS). Remote Sens. Environ. **65**, 227–248 (1998)
3. Plaza, A., Benediktsson, J.A., Boardman, J., Brazile, J., Bruzzone, L., Camps-Valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, J., Marconcini, M., Tilton, J.C., Trianni G.: Recent advances in techniques for hyperspectral image processing. Remote Sens. Environ. **113**, 110–122 (2009)
4. Plaza, A.: Special issue on architectures and techniques for real-time processing of remotely sensed images J. Real Time Image Process. **4**(3), 191–193 (2009)
5. Plaza, A., Plaza, J., Paz, A., Sanchez, S.: Parallel hyperspectral image and signal processing . IEEE Signal. Process. Mag. **28**(3), 119–126 (2011)
6. Lee, C.A., Gasster, S.D., Plaza, A., Chang, C.-I., Huang, B.: Recent developments in high performance computing for remote sensing: a review. IEEE J. Selected Topics Appl. Earth Observ. Remote Sens. **4**(3), 508–527 (2011)
7. Bioucas-Dias, J.M., Plaza, A.: Hyperspectral unmixing: geometrical, statistical, and sparse regression-based approaches. In: Proceedings of SPIE, Image and Signal Processing for Remote Sensing XVI 7830:1–15, Toulouse, France (2010)
8. Plaza, A., Du, Q., Bioucas-Dias, J.M., Jia, X., Kruse, F.: Foreword to the special issue on spectral unmixing of remotely sensed data. IEEE Trans. Geosci. Remote. Sens. **49**(11), 4103–4110 (2011)
9. Adams, J.B., Smith, M.O., Johnson, P.E.: Spectral mixture modeling: a new analysis of rock and soil types at the Viking Lander 1 site. J. Geophys. Res. Atmos. **91**, 8098–8112 (1986)
10. Plaza, A., Martinez, P., Perez, R., Plaza, J.: A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data. IEEE Trans. Geosci. Remote Sens. **42**(3), 650–663 (2004)
11. Raksuntorn, Q., Du. N., Younan, N.H., King, R.L.: End-member extraction for hyperspectral image analysis. Appl. Opt. **47**, 77–84 (2008)
12. Keshava, N., Mustard, J.F.: Spectral unmixing. IEEE Signal Process. Mag. **19**(1), 44–57 (2002)
13. Borel, C.C., Gerstl, S.A.W.: Nonlinear spectral mixing model for vegetative and soil surfaces. Remote Sens. Environ. **47**(3), 403–416 (1994)

14. Raksuntorn, N., Du, Q.: Nonlinear spectral mixture analysis for hyperspectral imagery in an unknown environment. IEEE Geosci. Remote Sens. Lett. **7**(4), 836–840 (2010)
15. Plaza, A., Martin, G., Plaza, J., Zortea, M., Sanchez, S.: Recent developments in spectral unmixing and endmember extraction. In: Prasad, S., Bruce, L.M., Chanussot, J. (eds.) Optical Remote Sensing, ch.12, pp. 235–267. Springer, Berlin (2011)
16. Chang, C.-I., Du, Q.: Estimation of number of spectrally distinct signal sources in hyperspectral imagery. IEEE Trans. Geosci. Remote Sens. **42**(3), 608–619 (2004)
17. Bioucas-Dias, J.M., Nascimento, J.M.P.: Hyperspectral subspace identification. IEEE Trans. Geosci. Remote Sens. **46**(8), 2435–2445 (2008)
18. Plaza, A., Chang, C.-I.: High Performance Computing in Remote Sensing. Taylor and Francis, Boca Raton (2007)
19. Tarabalka, Y., Haavardsholm, T.V., Kasen, I., Skauli, T.: Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and GPU processing. J. Real Time Image Process. **4**, 1–14 (2009)
20. Plaza, A., Du, Q., Chang, Y.-L., King, R.L.: High performance computing for hyperspectral remote sensing. IEEE J. Selected Topics Appl. Earth Observ. Remote Sens. **4**(3), 528–544 (2011)
21. Sanchez, S., Paz, A., Martin, G., Plaza, A.: Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units. Concurr Comput Pract Exp **23**(13), 1538–1557 (2011)
22. Green, R.O., Eastwood, M.L., Sarture, C.M., Chrien, T.G., Aronsson, M., Chippendale, B.J., Faust, J.A., Pavri, B.E., Chovit, C.J., Solis, M., et al. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). Remote Sens. Environ. **65**(3), 227–248 (1998)
23. Winter, M.E,: N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. Proc. SPIE Image Spectr. V **3753**, 266–277 (2003)
24. Sanchez, S., Plaza, A.: Real-time implementation of a full hyperspectral unmixing chain on graphics processing units. Proc. SPIE Satell. Data Compress. Commun. Process. VII **8157**, 1–9 (2011)

Author Biographies

Sergio Sánchez received the M.Sc. degree in 2010 and is currently a Research Associate with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, Spain. His main research interests comprise hyperspectral image analysis and efficient implementations of large-scale scientific problems on commodity graphical processing units (GPUs).

Antonio Plaza received the M.S. and Ph.D. degrees in computer engineering from the University of Extremadura, Caceres, Spain. He was a Visiting Researcher with the Remote Sensing Signal and Image Processing Laboratory, University of Maryland Baltimore County, Baltimore, with the Applied Information Sciences Branch, Goddard Space Flight Center, Greenbelt, MD, and with the AVIRIS Data Facility, Jet Propulsion Laboratory, Pasadena, CA. He is currently an Associate Professor with the Department of Technology of Computers and Communications, University of Extremadura, Caceres, Spain, where he is the Head of the Hyperspectral Computing Laboratory (HyperComp). He was the Coordinator of the Hyperspectral Imaging Network (Hyper-I-Net), a European project designed to build an interdisciplinary research community focused on hyperspectral imaging activities. He has been a Proposal Reviewer with the European Commission, the European Space Agency, and the Spanish Government. He is the author or coauthor of around 300 publications on remotely sensed hyperspectral imaging, including more than 60 Journal Citation Report papers, 20 book chapters, and over 200 conference proceeding papers. His research interests include remotely sensed hyperspectral imaging, pattern recognition, signal and image processing, and efficient implementation of large-scale scientific problems on parallel and distributed computer architectures. Dr. Plaza has coedited a book on high-performance computing in remote sensing and guest edited seven special issues on remotely sensed hyperspectral imaging for different journals, including the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING (for which he serves as Associate Editor on hyperspectral image analysis and signal processing since 2007), the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (for which he serves as a member of the steering committee since 2011), the International Journal of High Performance Computing Applications, and the Journal of Real-Time Image Processing. He is also serving as an Associate Editor for the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER. He has served as a reviewer for more than 280 manuscripts submitted to more than 50 different journals, including more than 140 manuscripts reviewed for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. He has served as a Chair for the IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing in 2011. He has also been serving as a Chair for the SPIE Conference on Satellite Data Compression, Communications, and Processing since 2009, and for the SPIE Remote Sensing Europe Conference on High Performance Computing in Remote Sensing since 2011. Dr. Plaza is a recipient of the recognition of Best Reviewers of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS in 2009 and a recipient of the recognition of Best Reviewers of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING in 2010. He is currently serving as Director of Education activities and member of the Administrative Committee of the IEEE GEOSCIENCE AND REMOTE SENSING SOCIETY.