Parallel Implementation of a Full Hyperspectral Unmixing Chain Using OpenCL

Sergio Bernabé, Guillermo Botella, Gabriel Martín, Manuel Prieto-Matias, and Antonio Plaza, *Fellow, IEEE*

***Abstract*—Spectral unmixing is an important task for remotely sensed hyperspectral data exploitation. Due to the fact that the spatial resolution of the sensor may not be able to separate different spectrally pure components (endmembers), spectral unmixing faces important challenges in order to characterize mixed pixels. As a result, several hyperspectral unmixing chains have been proposed to find the spectral signatures for each endmember and their associated abundance fractions. However, unmixing algorithms can be computationally expensive, which compromises their use in applications under real-time constraints. In this paper, we describe a new parallel hyperspectral unmixing chain based on three stages: 1) estimation of the number of endmembers using the geometry-based estimation of number of endmembers algorithm; 2) automatic identification of the spectral signatures of the endmembers using the simplex growing algorithm; and 3) estimation of the fractional abundance of each endmember in each pixel of the scene using the sum-to-one constrained least-squares unmixing algorithm. These algorithms have been specifically selected due to their successful performance in different applications. We have developed new parallel implementations of the aforementioned algorithms and assembled them in a fully operative unmixing chain using an hybrid implementation with the OpenCL framework and `clMAGMA` library. As a result, this is one of the first real-time implementations of a full unmixing chain in an open computing language. This methodology can be executed on different heterogeneous platforms such as CPU (multicore) and GPU platforms, in which accuracy, performance, and power consumption terms have been considered.**

***Index Terms*—Geometry-based estimation of number of endmembers algorithm (GENE), high-performance computing (HPC), hyperspectral images, openCL, simplex growing algorithm (SGA), spectral unmixing.**

I. Introduction

HYPERSPECTRAL images are characterized by their high spectral resolution and volume. These images can be obtained by satellite or airborne sensors, which collect hundreds or even thousands of spectral bands. As an example, the images obtained by the Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) have a size of 614 lines with 512 samples and 224 spectral bands, with spectral resolution of 10 nm in an area of 2–12 km wide and several kilometers long. The resulting multidimensional data cube comprises several GBytes per flight. As a result, the computational requirements needed to store, manage, and process these images are enormous. These constraints generally require the use of high-performance computing (HPC) techniques to obtain a fast response in remote sensing applications [1] such as environmental monitoring, mineral detection, or military and defense/security purposes.

An important problem in hyperspectral image processing is the presence of mixed pixels. Due to the low spatial resolution and other phenomena, several spectrally pure signatures (called endmembers) are combined into the same (mixed) pixel. Spectral unmixing[2] is an important technique to solve this problem identifying pure spectral components and their abundance fractions in each pixel. Popular approaches for this purpose have been the linear mixture model (LMM) and nonlinear mixture models (NLMM). In practice, the LMM is more flexible to adapt it to different analysis scenarios and the most used for the community to unmix remotely sensed hyperspectral data because it is simple to implement and it is unsupervised. Let us denote our hyperspectral data as $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_{n_s}]$, with the pixels $\mathbf{y}_i$ as columns of matrix $\mathbf{Y}$; therefore, $\mathbf{Y} \in \mathbb{R}^{\mathbf{L}\times\mathbf{n_s}}$, where L is the number of bands of our data and n_s is the number of pixels. The LMM assumes that the mixed spectral signatures may be represented as a linear combination of the endmembers and the abundance fractions associated with each endmember. The LMM may be represented in matrix form as

$$\mathbf{Y} \equiv \mathbf{EA} + \mathbf{W} \tag{1}$$

where $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_p]$ is a mixing matrix, which contains p endmember signatures, $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{n_s}]$ is the matrix that contains the fractional abundances $\mathbf{a}_i$ associated with the pixel i, and, finally, $\mathbf{W}$ represents the noise introduced in the model due to the acquisition process. The spectral unmixing chain considered in this work comprises three stages (see Fig. 1):

1) estimation of the number of pure spectral signatures (*endmembers*), p, in the hyperspectral scene;
2) identifying a collection of endmembers $\mathbf{E}$;
3) estimating the abundances, in which the fractional coverage of each endmember is estimated for each pixel.

Manuscript received September 29, 2016; revised February 23, 2017; accepted May 10, 2017. Date of publication June 11, 2017; date of current version July 17, 2017. This work was supported by the EU (FEDER) and the Spanish MINECO under Grant TIN2015-65277-R and Grant TIN2012-32180, the Formación Postdoctoral program (FPDI-2013-16280), and the Portuguese Science and Technology Foundation under Grant SFRH/BPD/94160/2013. *(Corresponding author: Sergio Bernabé.)*

S. Bernabé, G. Botella, and M. Prieto-Matias are with the Complutense University of Madrid, E-28040 Madrid, Spain (e-mail: sebernab@ucm.es; gbotella@ucm.es; mpmatias@dacya.ucm.es).

G. Martín is with the Instituto de Telecomunicações, 1049-1 Lisbon, Portugal (e-mail: g.martin.he@gmail.com).

A. Plaza is with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, E-10003 Cáceres, Spain (e-mail: aplaza@unex.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTARS.2017.2707541

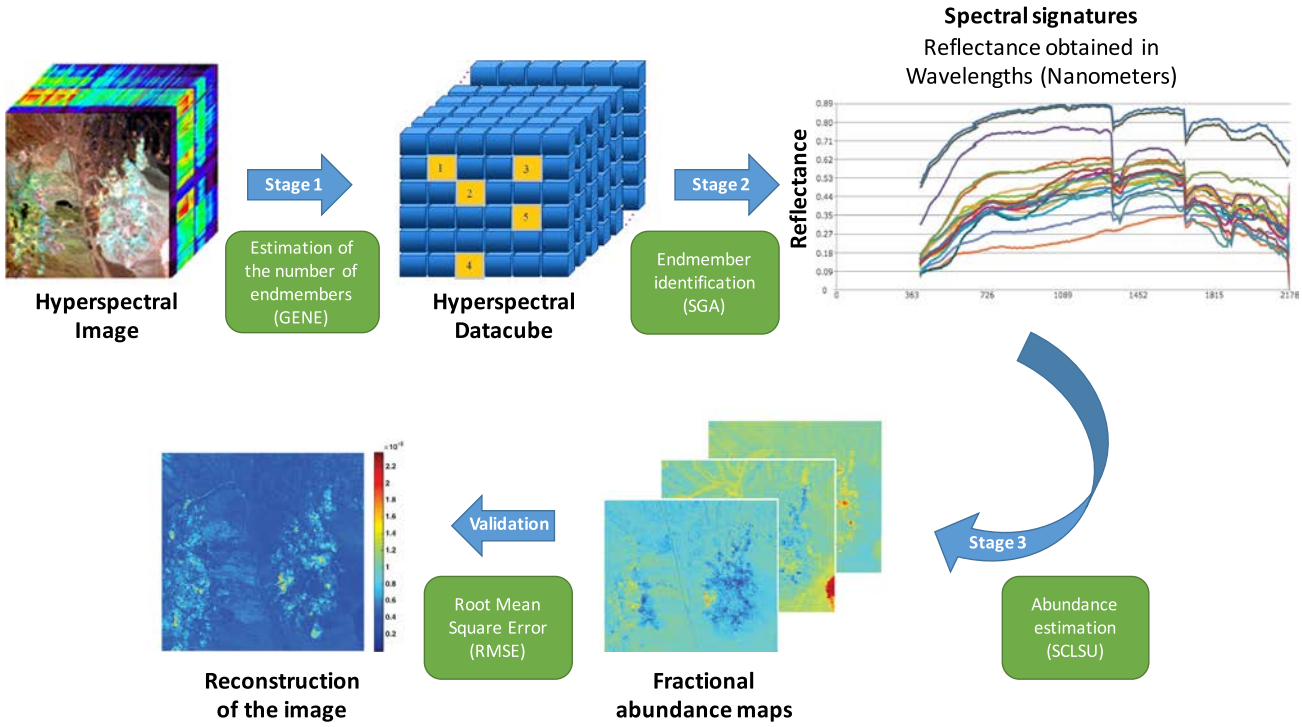

Fig. 1. Block diagram illustrating the proposed full unmixing chain applied on hyperspectral images analysis.

The estimation error can be computed by reconstructing the original image (using the extracted endmembers and the derived abundances) and comparing the reconstructed image with the original one.

Previous research studies have implemented different approaches to solve the aforementioned problem under the LMM assumption (see [2], among several others), but all of them are quite expensive in computational terms. Introducing HPC techniques, several studies have accelerated hyperspectral unmixing methods using multicore processors [3], graphical processing units (GPUs) [4], or field-programmable gate arrays (FPGAs) [5]–[7] using OpenMP, CUDA, and VHDL, respectively. One of the main problems in previous studies is that the proposed implementations cannot be executed across different platforms. On the other hand, OpenCL is a free alternative to the previous frameworks whose main advantages include the short development time and the possibility to obtain fast implementations with completely portable codes [8] across different heterogeneous systems such as GPUs [9], multicore processors [10], the Intel Xeon Phi [11], and other custom devices [12]. Based on these advantages, OpenCL and GPU platforms are selected to be used in a fully operational unmixing chain.

Here, we specifically develop a full spectral unmixing chain implemented using a hybrid implementation with the OpenCL framework and `clMAGMA` library. Although previous work has discussed the implementation of full unmixing chains on GPUs, the implementation of a portable OpenCL code for a diverse set of heterogeneous platforms has not been discussed in previous contributions, to the best of our knowledge. The considered GPU platform is evaluated in terms of accuracy, performance, and power consumption. In our experiments, we have selected an Intel Xeon processor E5-2695 v3 at 2.30 GHz (multicore CPU) and an NVidia GeForce GTX 980 (GPU). This study reveals that GPU fully meets real-time constraints in hyperspectral imaging applications using real and synthetic datasets.

The remainder of this paper is organized as follows. Section II briefly describes the different algorithms used to implement the unmixing chain. Section III describes the proposed parallel implementations using the OpenCL framework. Section IV presents an experimental evaluation of the proposed implementation in terms of accuracy, parallel performance, and power consumption using two datasets on heterogeneous platforms. Finally, Section V outlines the conclusions of this paper with some remarks and pointers to future work.

II. Hyperspectral Unmixing Chain

The full hyperspectral unmixing chain considered in this work comprises three steps:

1) geometry-based estimation of number of endmember (GENE) for estimating the number of endmembers;
2) simplex growing algorithm (SGA) for estimating the endmember signatures;
3) sum-to-one constrained least-squares unmixing (SCLSU) for estimating the fractional abundances for each endmember.

In the following, we describe these algorithms.

A. GENE Algorithm

The GENE method makes use of the key geometric characteristics of the hyperspectral data (see original proposal [13]). From previous work [14], the GENE-AH algorithm (see

Algorithm 1: : Pseudocode of the GENE Algorithm.

INPUT: $\mathbf{Y}, \mathbf{W}, N > 3, P_f \geq 0$
% $\mathbf{Y}$ is $L \times n_s$ matrix with the hyperspectral dataset, where L is the number of spectral bands and n_s is the number of pixels.
% $\mathbf{W}$ is $L \times n_s$ matrix with the noise estimation from the scene.
% N and P_f are the maximum number of estimated pixels and false-alarm probability values.
OUTPUT: $\widehat{p}$
% $\widehat{p}$ is the estimated number of endmembers.
$\mathbf{C}_w := (\mathbf{W} - \bar{\mathbf{W}})(\mathbf{W} - \bar{\mathbf{W}})^T / n_s$
% $\mathbf{C}_w$ is $L \times L$ matrix with the covariance from the noise estimation.
$\mathbf{C}_y := (\mathbf{Y} - \bar{\mathbf{Y}})(\mathbf{Y} - \bar{\mathbf{Y}})^T / n_s - \mathbf{C}_w$
% $\mathbf{C}_y$ is $L \times L$ matrix with the covariance from the hyperspectral data.
$\mathbf{U} := \mathbf{U}\Sigma\mathbf{U}^T \equiv \mathbf{C}_y$ {SVD decomposition}
$\tilde{\mathbf{Y}} := \mathbf{U}(\mathbf{Y} - \bar{\mathbf{Y}})$ {Reduced image}
$\tilde{\mathbf{C}}_w := \mathbf{U}\mathbf{C}_w\mathbf{U}^T$
$k := 1$
$\widehat{\mathbf{e}}_k := \tilde{\mathbf{y}}_{l_k}$ for $l_k \in \arg\max_i \|\tilde{\mathbf{y}}_i\|_2$ {First iteration selects the pixel with higher ℓ_2 norm.}
$\mathbf{Q} := \widehat{\mathbf{e}}_k$
for $k = 2$ to N **do**
  $\widehat{\mathbf{e}}_k := \tilde{\mathbf{y}}_{l_k}$ for $l_k \in \arg\max_i \|\mathbf{P}_{\mathbf{Q}}^{\perp}\mathbf{y}_i\|_2$ {ATGP iteration}
  $\theta := \arg\min_{1_{k-1}^T\theta=1} \|\widehat{\mathbf{e}}_k - \mathbf{Q}\theta\|_2^2$ { SCLSU algorithm over the selected endmember}
  $\mathbf{s} := \widehat{\mathbf{e}}_k - \mathbf{Q}\theta$ {Compute the residual error}
  $\mathbf{r} := \mathbf{s}^T((1 + \theta^T\theta)\tilde{\mathbf{C}}_w)^{-1}\mathbf{s}$ {Neyman–Pearson classifier}
  $\psi := 1 - \frac{\gamma(r/2,(N-1)/2)}{\Gamma((N-1)/2)}$
  if $\psi > P_f$ **then**
    $\tilde{\mathbf{E}} := \mathbf{Q}$
    $\widehat{p} := k - 1$
    EXIT
  end if
  $\mathbf{Q} := [\mathbf{Q}, \widehat{\mathbf{e}}_k]$
end for

Algorithm 2: Pseudocode of noise estimation.

INPUTS: Y
$\mathbf{Z} \leftarrow \mathbf{Y}^T$, $\widehat{\mathbf{R}} \leftarrow \mathbf{Z}^T\mathbf{Z}$
$\mathbf{R}' \leftarrow \widehat{\mathbf{R}}^{-1}$
for $i = 1$ to L **do**
$\widehat{\beta}_i \leftarrow ([\mathbf{R}']_{\alpha_i,\alpha_i} - ([\mathbf{R}']_{\alpha_i,i}[\mathbf{R}']_{i,\alpha_i}) / [\mathbf{R}']_{i,i})[\mathbf{R}']_{\alpha_i,i}$ % Note that $\alpha_i =$ [1, ..., i-1, i+1, ..., L]
end for
$\mathbf{W} \leftarrow \mathbf{Z} - \mathbf{Z}\widehat{\beta}$
OUTPUT: W
% $L \times n_s$ matrix with the estimated noise

Algorithms 1 and 2), called GENE hereinafter, was selected since it is more robust than GENE-CH against the absence of pure pixels in the data. It has differences and similarities with the most popular methods for estimating the number of endmembers. These are VD [15] and Hysime [16]; both of them were included in previously developed full spectral unmixing chains for HPC, and thus, this is another reason for including the GENE method in our proposed unmixing chain. The GENE method uses a Neyman–Pearson classifier rule as in VD, but it works with different hypothesis; while VD compares the differences of the eigenvalues between the correlation and the covariance matrices, the GENE method evaluates the reconstruction error of the endmembers against the others. Also, this is a different perspective compared with HySime, in which the number of estimated endmembers is chosen by minimizing both the reconstruction error and the noise power of the whole dataset simultaneously. The GENE method works with the dimension-reduced hyperspectral data, due to the following:

1) under the LMM assumption it preserves the true information about the mixing process;
2) it reduces the complexity of the unmixing process;
3) it reduces the impact of the noise in the hyperspectral data.

The dimensionality reduction proposed in [13] assumes that the statistics of the noise are known and the multiplied-regression-analysis-based noise covariance estimation method reported in [16] is used as suggested by the authors of GENE. The dimensionality reduction is similar to the computation of the principal components of the data but removing the noise statistics of the data from the covariance matrix. Later, the GENE method is used in combination with the TRI-P algorithm with the ℓ_2 norm to identify endmembers iteratively: on every iteration, the algorithm identifies a potentially new endmember, which GENE tries to decompose into a linear combination of the previously identified endmembers. By using the residual error of the decomposition and the noise statistics of the data, the algorithm performs a Neyman–Pearson classifier rule to determine if the dimensionality of the new subspace is higher than the previous iteration, i.e., the output of the Neyman–Pearson classifier is used as GENE stopping criteria.

B. SGA Algorithm

The SGA method for spectral unmixing was originally developed in [17], and it is described in the Algorithm 3. This method assumes the presence of pure pixels on the data. It iteratively chooses as endmembers the image pixels, which maximize the volume of the simplex. The first endmember is chosen using a randomly generated target pixel $\mathbf{t}$ (step 4 in Algorithm 3). Experimental results have shown that different selections for the target pixel $\mathbf{t}$ do not affect to the final set of endmembers. Later, the algorithm iteratively chooses as endmembers those image pixels that maximize the volume defined in step 7 of Algorithm 3. Finally, the algorithm stops when a number of endmembers p have been selected, where p is the number of endmembers estimated by the GENE method.

C. SCLSU Algorithm

For the abundance estimation part of our full unmixing chain, the SCLSU algorithm was used. This method belongs to the

Algorithm 3: Pseudocode of the SGA.

1: INPUT: $\mathbf{Y}, \widehat{p} > 0$
   % $\mathbf{Y}$ is $L \times n_s$ matrix with the hyperspectral dataset, where L is the number of spectral bands and n_s is the number of pixels.
   % $\widehat{p}$ is an estimated value generated by the GENE algorithm.
2: OUTPUT: $\widehat{\mathbf{E}}$
   % $\widehat{\mathbf{E}}$ is $L \times \widehat{p}$ matrix with the spectral signatures for each endmember.
3: $n = 1$
   % **r** is an array corresponding to a pixel with all the spectral bands.
4: $\mathbf{e}_n := \arg\left\{ \max_{\mathbf{r}} \left[\left| \det \begin{bmatrix} 1 & 1 \\ \mathbf{t} & \mathbf{r} \end{bmatrix} \right| \right] \right\}$ % First initial end member pixel
5: $\widehat{\mathbf{E}}_n := [\mathbf{e}_n]$
6: **for** n to $\widehat{p} - 1$ **do**
7: $\mathbf{V}(e_1, \ldots, e_n, \mathbf{r}) := \dfrac{\left| \det \begin{bmatrix} 1 & 1 & \ldots & 1 & 1 \\ \mathbf{e_1} & \mathbf{e_2} & \ldots & \mathbf{e}_n & \mathbf{r} \end{bmatrix} \right|}{n!}$
8: $\mathbf{e}_{n+1} := \arg\left\{ \max_{\mathbf{r}} [\mathbf{V}(\mathbf{e}_1, \ldots, \mathbf{e}_n, \mathbf{r})] \right\}$
9: $\widehat{\mathbf{E}} := [\widehat{\mathbf{E}}\ , \mathbf{e}_{n+1}]$
10: **end for**

Algorithm 4 : Pseudocode of the SCLSU Algorithm

1: INPUT: $\mathbf{Y}, \mathbf{E}$
   % $\mathbf{Y}$ is $L \times n_s$ matrix with the hyperspectral dataset, where L is the number of spectral bands and n_s is the number of pixels.
   % $\mathbf{E}$ is $L \times \widehat{p}$ matrix with the estimated endmembers generated by the SGA algorithm and $\widehat{p}$ is the number of estimated endmembers by the GENE method.
2: OUTPUT: $\mathbf{A}$
   % $\mathbf{A}$ is $n_s \times \widehat{p}$ matrix containing the abundance fractions for each of the $\widehat{p}$ endmembers in $\mathbf{Y}$.
3: $\mathbf{Y} := \mathbf{Y} / \|\mathbf{Y}\|_F$
4: $\mathbf{E} := \mathbf{E} / \|\mathbf{Y}\|_F$
5: $\mathbf{E}^{\#} := (\mathbf{E}^T \mathbf{E})^{-1}$
6: $\mathbf{F} := \mathbf{E}^{\#} \mathbf{1} (\mathbf{1}^T \mathbf{E}^{\#} \mathbf{1})^{-1}$
7: $\mathbf{G} := \mathbf{E}^{\#} - \mathbf{F} \mathbf{1}^T \mathbf{E}^{\#}$
8: $\mathbf{A} := \mathbf{G} \mathbf{E}^T \mathbf{Y} + \mathbf{F} \mathbf{1}$

third stage in the unmixing process called abundance extraction algorithms and was proposed in [18] for spectral unmixing. Algorithm 4 imposes the abundance sum-to-one constraint, while ignoring the abundance nonnegativity constraint. The main advantage of this method is that the solution can be obtained by an unconstrained least-squares solution plus an error correction term. However, its solution does not guarantee that the estimated abundance fractions are nonnegative as in nonnegativity constrained least-squares unmixing or fully constrained least-squares unmixing algorithms [18]. The SCLU algorithm has been computed efficiently as described in Algorithm 4, where $\mathbf{1}$ and $\mathbf{1}^T$ represent column and row vectors of 1's, respectively, with suitable size. Note that the computation of $\mathbf{F}$ in Algorithm 4 is very simple to compute by summing the elements of each row of $\mathbf{E}^{\#}$ and dividing by the sum of all the elements of $\mathbf{E}^{\#}$, and the computation of $\mathbf{G}$ may be performed element by element in a simple way. Therefore, the most computational expensive part of the algorithm is the computation of the matrix multiplications $\mathbf{G}\mathbf{E}^t\mathbf{Y}$ and the computation of the matrix $\mathbf{E}^{\#}$.

III. Parallel Unmixing Chain

In this section, we describe our parallel implementation of the proposed unmixing chain, where a hybrid approach that only uses the device to accelerate the main parts of the algorithm has been opted using `clMAGMA` library function calls for mathematical purposes. We have identified those stages with a profiling of an optimized serial implementation based on the BLAS library. Before explaining the parallel implementation, we will describe the OpenCL framework.

OpenCL is a framework for parallel implementation that allows the execution of parallel programs on heterogeneous platforms. It is currently supported by several hardware devices, such as CPUs, GPUs, DSPs, FPGAs, and other accelerators. OpenCL is based on the host-device model, where we have one or more kernels to express the parallelism in our program. The main goal in this kind of implementations is to use a common code written in OpenCL for the different platforms.

The OpenCL programming model divides a program workload into *work groups* and *work items*. We can use a *work group* with a limited number of *work items* depending on the selected device. Each *work group* is executed independently with respect to other *work groups*. On the other hand, the memory model distinguishes different memory regions: global, local, constant, and private memories. Global memory is read–write accessible by all *work items* across all *work groups*, and it usually corresponds to the DRAM memory device, which carries a high latency memory access. Local memory is a shared read–write memory accessible from all *work items* of a single *work group*, and it habitually involves a low latency memory access. Constant memory is a read-only memory that is visible to all *work items* across all *work groups*, and private memory, as the name suggests, is only accessible by a single *work item*.

Before performing any processing on the platform, the hyperspectral image is loaded band by band from the host to device global memory. With this data layout, the access to consecutive pixels in the same spectral band performed by adjacent work items can be coalesced into fewer memory transactions. With the above ideas in mind, our proposed parallel implementation of the hyperspectral unmixing chain comprises the three following steps.

A. P-GENE Implementation

Once we load the full hyperspectral image $\mathbf{Y}$ from host memory to the global memory of the *device*, the first step is to compute the noise estimation following Fig. 2. A matrix multiplication is performed using the `clMAGMA` library.

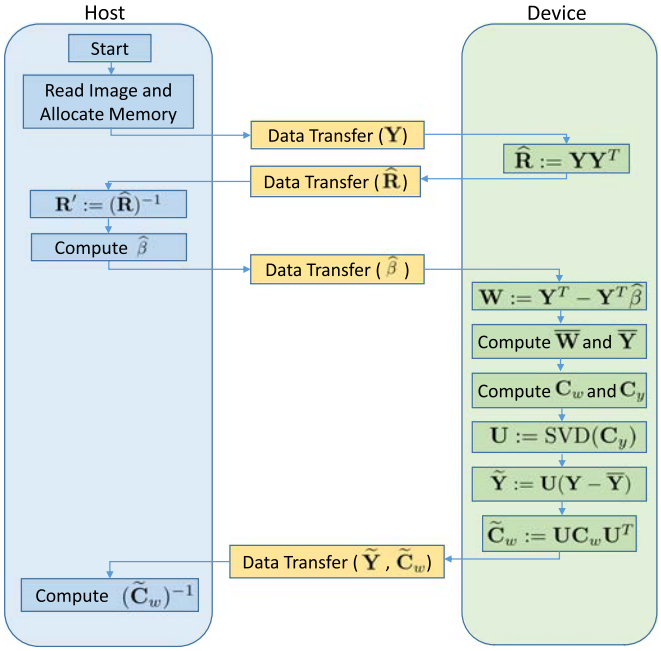

Fig. 2. Block diagram illustrating our mapping of the first part belongs to the P-GENE method using OpenCL.

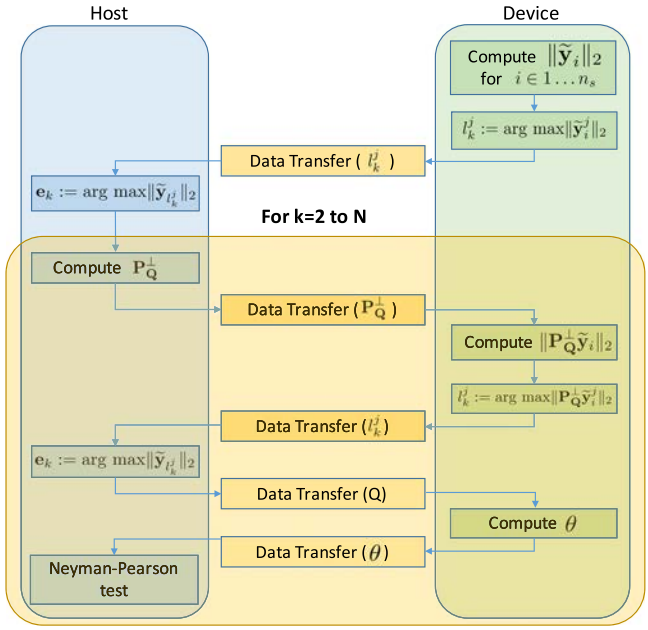

Fig. 3. Block diagram illustrating our mapping of the second part belongs to the P-GENE method using OpenCL.

Specifically, we use the *magma_dgemm* function of `clMAGMA` to obtain $\widehat{\mathbf{R}}$. After that, the next step is to compute β and $\mathbf{R}'$ (the inverse of the matrix $\widehat{\mathbf{R}}$). Experimentally, we have checked that both operations can be computed very fast in the *host*. Then, an ad-hoc kernel implemented in OpenCL, called *mean_reduction*, computes the mean value of each band of $\mathbf{W}$ using a reduction process and subtract it to all the pixels in the same band. This step has been optimized using local memory and coalesced memory accesses. The resulting matrix $(\mathbf{W} - \bar{\mathbf{W}})$ is used to finalize the calculation of the covariance matrix $\mathbf{C}_w$ by means of a matrix multiplication operation $(\mathbf{W} - \bar{\mathbf{W}})\,(\mathbf{W} - \bar{\mathbf{W}})^T$, which is mapped on the *device* using the *magma_dgemm* function. The same strategy is performed later to compute the covariance matrix $\mathbf{C}_y$. After that, the singular value decomposition (SVD) on matrix $\mathbf{C}_y$ is performed very efficiently on the *device* using the *magma_dgesvd* function. Then, we take advantage of the *device* computing the $\tilde{\mathbf{Y}}$ and $\tilde{\mathbf{C}}_w$ matrices through *magma_dgemm* functions. Both results are transferred to the *host*.

The next step of the algorithm uses the ATGP method [19]. Following Fig. 3 to compute the first endmember $\mathbf{e}_1$, we need to calculate the brightest pixel in the reduced image, $\tilde{\mathbf{Y}}$. For this purpose, we have developed an ad-hoc kernel to compute the ℓ_2 norm, called *first_endmember*, for each pixel vector and another ad-hoc kernel to retain the maximum value of all them called *maximum_reduction* (for more details, see the parallel implementation described in [20]). The pixel with maximum ℓ_2 norm is computed in two steps. The first step is performed on the *device* and consist of partitioning the reduced image $\tilde{\mathbf{Y}}_i^j$ between the different blocks of the *device*, where j denotes the block in which the pixel is allocated. Each block perform a reduction process to obtain the maximum of its data partition, retaining the index of the maximum of its partition l_k^j. These indexes are then transferred to the *host*, which computes the maximum of all the blocks. After that, the algorithm iteratively computes the most orthogonal vector to those obtained thus far, as follows. First, the orthogonal projection operator $\mathbf{P}_{\mathbf{Q}}^{\perp}$ is computed in the *host*. Second, an ad-hoc kernel computes the projection of each vector with regard to the orthogonal operator. Finally, the pixel with maximum projection is computed in two steps by both *device* and *host* in a similar process to that described in the aforementioned *maximum_reduction* kernel. To finish this first stage, θ is computed on the device by using the P-SCLSU algorithm described in Section III-C. On the other hand, we have experimentally tested that the Neyman–Pearson test does not exhibit enough data parallelism to be performed by the *device*, and therefore, it is performed on the *host* with negligible performance degradation.

B. P-SGA Implementation

The second stage in our proposed unmixing chain is made up of three ad-hoc kernels as shown in Fig. 4. First, an ad-hoc kernel called *determinant_calculation* computes the determinant of a matrix of size $(\widehat{p}+1) \times (\widehat{p}+1)$, (being $\widehat{p}$ the input parameter that specifies the desired number of endmembers) using a LU decomposition. In this decomposition, each *work item* calculates its value, so that we can obtain a value equal to 0 under the main diagonal after processing each row of the matrix. For the first endmember, a random candidate is selected although it will be replaced by the real at the end of the first iteration. It is important to note that for each iteration, the computed and new endmembers are stored in local and global memory, respectively. With this in mind and once calculated the determinant, the volume is computed, where each *work item* obtains its relative value, so it can be divided later on by its factorial in order to obtain the absolute value.

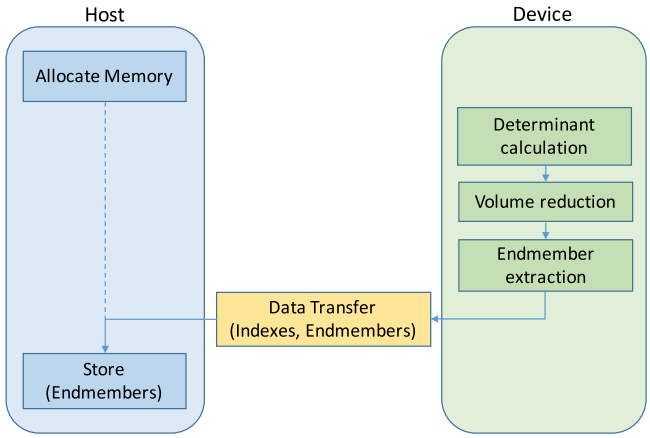

Fig. 4. Block diagram illustrating our mapping of the P-SGA method using OpenCL.

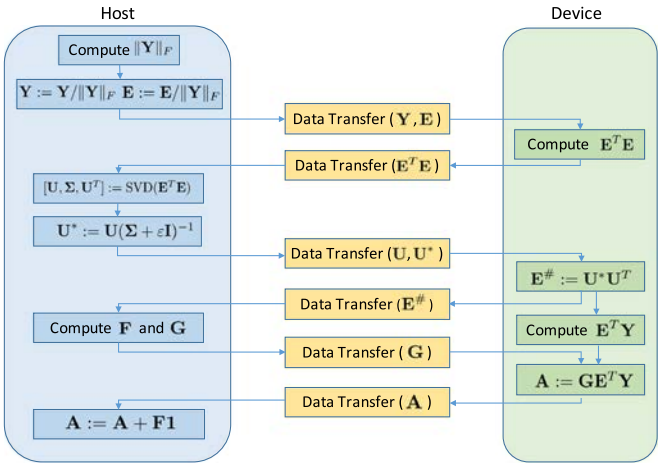

Fig. 5. Block diagram illustrating our mapping of the P-SCLSU method using OpenCL.

Second, another ad-hoc kernel called *volume_reduction* is performed for each volume values along two steps. In the first step, each *work item* compares each element to obtain the maximum value and store it in an array of size $\widehat{p}$, located in the local memory. The second step performs a small reduction in a serial way, so the maximum volume is obtained. Once the maximum volume has been calculated, an ad-hoc kernel called *endmember_extraction* is performed in order to extract all the spectral bands from the considered endmember. During the extraction, each *work item* stores each element in a resultant array with a stride of $\widehat{p}$ elements.

These steps are repeated until the algorithm reaches up the desired number of endmembers (parameter $\widehat{p}$). Finally, the array with all the obtained endmembers is transferred to the host memory.

C. P-SCLSU Implementation

Fig. 5 shows the flowchart of the different operations and data transfers performed between the *device* and the *host* for the P-SCLSU algorithm. It starts dividing the elements of both endmembers and data by the Frobenius norm of $\mathbf{Y}$. After that, the next step is to compute the pseudoinverse of the endmember matrix $\mathbf{E}$. In our implementation, this step is carried out through the SVD of $\mathbf{E}^T\mathbf{E}$ as follows:

$$\mathbf{E}^\# \equiv (\mathbf{E}^T\mathbf{E})^{-1} \equiv \mathbf{U}(\mathbf{\Sigma} + \varepsilon\mathbf{I})^{-1}\mathbf{U}^T \tag{2}$$

with $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T \equiv \mathbf{E}^T\mathbf{E}$; thus, $\mathbf{U}$ represents the eigenvectors and $\mathbf{\Sigma}$ the diagonal matrix with the eigenvalues of $\mathbf{E}^T\mathbf{E}$, ε a scalar value close to 0, and $\mathbf{I}$ the identity matrix of suitable size. As Fig. 5 shows, the matrix multiplication $\mathbf{E}^T\mathbf{E}$ is performed on the *device*, while the SVD decomposition is performed on the *host* because of the small size of the matrices promotes low occupancy in the *device* for the SVD operation. Later, the matrix $\mathbf{E}^\#$ is obtained by performing the right-hand matrix multiplication of the eigenvectors in the *device*. After that, the more computationally expensive part of the algorithm that is the computation of $\mathbf{E}^T\mathbf{Y}$ is performed on the *device*, while asynchronously the *host* is computing the matrices $\mathbf{F}$ and $\mathbf{G}$. Once the matrix $\mathbf{G}$ has been computed and transferred to the *device*, it computes the matrix multiplication between $\mathbf{G}$ and $\mathbf{E}^T\mathbf{Y}$. Finally, the abundances are obtained by adding the term $\mathbf{F1}$ to the result of the product.

IV. Experimental Results

A. Experimental Configuration

The multicore heterogeneous system used for experimental evaluations is equipped with: two Intel Xeon E5-2695 v3 CPUs (28 cores in total) at 2.30 GHz, with 64 GBytes of DDR3 RAM memory and Debian GNU/Linux 8 as the operating system installed. Moreover, an NVidia GeForce GTX 980 GPU with 2048 cores operating at 1.126 GHz and dedicated memory of 4 GBytes is connected by PCI-e.

The experiments were conducted using two hyperspectral scenes. The first dataset used is the well-known AVIRIS Cuprite scene,[1] which has been widely used to study the accuracy of unmixing algorithms. This dataset is available online in reflectance units after atmospheric correction. The portion used in our experiments corresponds to a 350 × 350 pixel subset with a spatial resolution of 20-m pixels, which comprises 188 spectral bands in the range from 0.4 to 2.5 μm and a total size of around 46 MB. The site is well understood mineralogically and has several exposed mineral of interest including *alunite*, *buddingtonite*, *calcite*, *kaolinite*, and *muscovite*. The reference ground signatures of the above minerals are available in the United States Geological Survey (USGS) library.[2] On the other hand, a second simulated dataset was considered for experiments. This scene is composed of 30 signatures from the USGS library. The procedure is described in [21] to simulate natural spatial patterns, composed of a total size of 750 × 650 pixels and a total size of around 437 MB. Fig. 6 displays a color-scale composition and three examples of ground-truth abundance maps from the simulated image.

[1] [Online]. Available: http://aviris.jpl.nasa.gov/data/free_data.html
[2] [Online]. http://speclab.cr.usgs.gov/spectral-lib.html

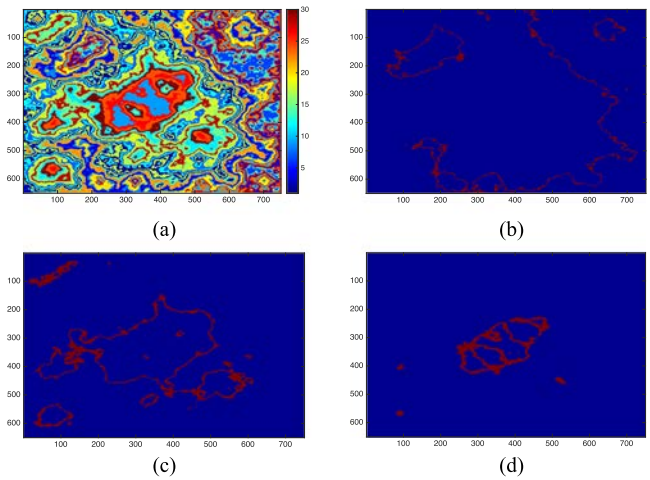

Fig. 6. (a) Color-scale composition and three examples of ground-truth abundance maps of endmembers in the simulated hyperspectral data. (b) Endmember #3. (c) Endmember #15. (d) Endmember #26.

TABLE I
GENE ESTIMATES OF $\widehat{p}$ ON REAL AND SIMULATED DATASETS FOR VARIOUS FALSE ALARM PROBABILITIES (P_f) AND MAXIMUM NUMBER OF ENDMEMBERS, N

P_f	N	AVIRIS Cuprite	Simulated
$10^{-1},\ldots,10^{-8}$	45	38	29
0	45	35	29
10^{-1}	43	40	30
$10^{-2},\ldots,10^{-8}$	43	40	29
0	43	40	29
$10^{-1},\ldots,10^{-3}$	40	36	30
$10^{-4},\ldots,10^{-8}$	40	36	29
0	40	36	29
10^{-1}	35	35	**30**
$10^{-2},\ldots,10^{-8}$	35	35	29
0	35	34	29
$10^{-1},\ldots,10^{-5}$	34	34	29
$10^{-6},\ldots,10^{-8}$	34	33	29
0	34	**33**	29
10^{-1}	30	30	30
$10^{-2},\ldots,10^{-8}$	30	30	28
0	30	30	28
10^{-1}	29	29	29
$10^{-2},\ldots,10^{-8}$	29	29	28
0	29	29	28

B. Accuracy Evaluation

First of all, we evaluate the accuracy for each stage in our proposed unmixing chain. For the first one, Table I provides the values of $\widehat{p}$ (number of endmembers) estimated by the P-GENE method for the two considered hyperspectral datasets using different values of the false alarm probability (P_f) and N.

As shown in Table I, the number of endmembers estimated by P-GENE in the simulated case is very similar in most of the cases with the ground-truth value $p = 30$. In the case of the Cuprite dataset, the estimation of the number of endmembers varies depending on the chosen value for N and P_f; this is expected as the GENE method is based on the assumption that the data follows strictly the LMM, which is not the case in the real scenarios because of the nonlinear reflections of the light. In our case, we take as good the value of $\widehat{p} = 33$ because it is the minimum estimated value by the GENE method in which $\widehat{p} < N$, and therefore, it is the first case in which the Neyman–Pearson classifier rule was fulfilled. Due to the fact that there is not ground-truth value of p for the real Cuprite image, the only thing we may do is compare with the estimated value of the number of endmembers given by the VD and HySime algorithms, which is $\widehat{p} = 31$ with $P_f = 10^{-1}$ and $\widehat{p} = 17$, respectively.

TABLE II
SPECTRAL ANGLE VALUES (IN DEGREES) BETWEEN THE ENDMEMBERS EXTRACTED BY P-SGA AND THE KNOWN GROUND ENDMEMBERS IN THE AVIRIS CUPRITE SCENE

Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite	Average
9.08°	4.33°	5.20°	12.69°	5.94°	7.45°

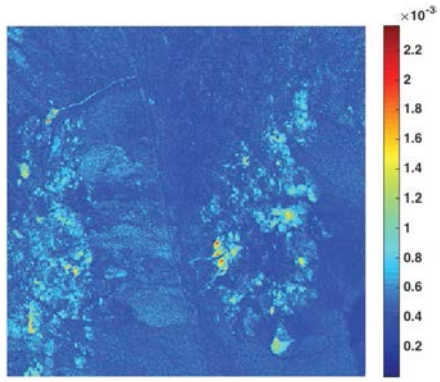

Fig. 7. Per-pixel RMSE obtained in the reconstruction process for the AVIRIS Cuprite scene whose overall RMSE was 0.0005.

On the other hand, we evaluate the accuracy for the second stage using the well-known spectral angle distance (SAD) [18] for the AVIRIS Cuprite scene. This distance gets the worst case at 90° and the best case at 0°. As a result, Table II shows the angles obtained after applying SAD between the extracted endmembers and five known ground-truth spectral signatures obtaining over 7° on average. As we can see, P-SGA obtains very low SAD scores indicating that the implementation provides accurate results.

Finally, the result of our proposed full unmixing chain can be evaluated in terms of the quality of the reconstruction of the original dataset (using the obtained endmembers by the SGA algorithm) and the estimated fractional abundances (by the SCLSU algorithm using the LMM). Therefore, we have employed the root-mean-square error (RMSE) obtained between the original scene and the reconstructed one. This metric shown in (3) is based on the assumption that a set of high-quality endmembers (and their corresponding estimated abundance fractions) may allow reconstruction of the original scene with higher precision compared to a set of low-quality endmembers. In this case, the original scenes are used as a reference to measure the fidelity of

TABLE III
PROCESSING TIMES (IN SECONDS) AND SPEEDUPS ACHIEVED FOR THE PROPOSED IMPLEMENTATIONS IN GPU PLATFORM AND TESTED WITH REAL AND SIMULATED DATASETS

		AVIRIS Cuprite		Simulated	
		Serial CPU	OpenCL GPU	Serial CPU	OpenCL GPU
P-GENE	Transfers	–	0.084 (8.84%)	–	0.399 (10.55%)
	Host	3.446	0.048 (5.04%)	14.033	0.089 (2.34%)
	Device	–	0.814 (86.12%)	–	3.293 (87.11%)
P-SGA	Transfers	–	0.130 (45.49%)	–	0.728 (56.12%)
	Host	19.261	0.000 (0.03%)	55.739	0.000 (0.01%)
	Device	–	0.155 (54.48%)	–	0.569 (43.87%)
P-SCLSU	Transfers	–	0.060 (30.16%)	–	0.212 (26.79%)
	Host	0.386	0.109 (55.24%)	1.714	0.518 (65.54%)
	Device	–	0.029 (14.60%)	–	0.061 (7.67%)
Full Unmixing Chain	Total Time	23.093	1.429	71.486	5.869
	Speedup	–	16.16x	–	12.18x

the reconstructed version on a per-pixel basis

$$\mathrm{RMSE}(Y,\hat{Y}) \leftarrow \frac{1}{n_s}\sum_{i=0}^{n_s}\left(\sum_{j=0}^{L}(y_i^j-\hat{y}_i^j)^2\right)^{\frac{1}{2}}. \qquad (3)$$

For illustrative purposes, Fig. 7 represents graphically the per-pixel RMSE obtained in the reconstruction process for the real dataset. The RMSE value reveals a good spatial distribution of the error with quite low values, indicating a good overall compromise in the reconstruction of the original scene.

C. Performance Evaluation

Before describing our parallel performance results, it is important to emphasize that our parallel and serial versions provide exactly the same results using the `gcc`-4.9.2 compiler with the `-O3` optimization flag to exploit data locality and avoid redundant computations. Note that for the serial implementation, we used the BLAS and LAPACK implementation from Netlib, version 3.5.0. For each experiment, ten runs were performed and the mean values were reported.

Table III summarizes the timing results after processing two hyperspectral images on the considered heterogeneous platform. It should be noted that *Transfers* include data transfers overheads between the device and the host memories. *Host* and *Device* correspond to the time executed on serial and parallel ways, respectively. As shown by Table III, our implementation achieves significant speedup in both datasets between 12× and 16×. On the other hand, it should be noted that the cross-track line scan time in AVIRIS is quite fast (8.3 ms to collect 512 full pixel vectors). This introduces the need to process both images in less than 1.985 and 7.902 s to fully achieve real-time performance.

On the other hand, Table IV compares the obtained results with other developments in the literature, such as CUDA [3], OpenMP [3], or CUDA + OpenMP [4] implementations and considering the AVIRIS Cuprite scene. From this table, we can outline the following aspects. In all cases, the main goal is achieved after processing the scene in real time except in [3], where OpenMP processes in near real time. In our work, the processing time is little more than in previous works, but we extract more endmembers and we propose an implementation to execute across a range of processing devices including multicore processors, GPUs, and other accelerators using an OpenCL code. Moreover, we have evaluated the power consumption dissipated because it is a constraint for real-time requirements.

TABLE IV
EXECUTION TIMES FOR THE AVIRIS CUPRITE SCENE COMPARING OUR RESULTS WITH OTHER DEVELOPMENTS IN THE LITERATURE

Implementation	$\widehat{p}$	Time execution (s)
CUDA [3]	19	0.590
CUDA + OpenMP [4]	18	0.631
OpenCL	33	1.428
OpenMP [3]	19	1.853

D. Power Consumption Evaluation

Last but not least, we evaluate the power consumption using the software solution *PowerMeter daemon* (*pmlib*) [22]. Gathering power results periodically from the tools provided by manufacturers, namely: RAPL for the Intel Xeon CPU, NVML for the NVidia GPU, and Intel MicMGMT for the Xeon Phi. For this work, RAPL and NVML for the proposed hybrid implementation have been selected.

Fig. 8 shows the power consumption for the proposed implementation using the described simulated dataset on CPU/GPU architecture throughout the execution time. If we analyze the plot, our full unmixing chain dissipates 149.05 W on average (874.48 J), and 255.07 W at most. Evaluating these results considering performance measured in Mega pixel per second (Mpps) and power consumption, we obtain 0.085 Mpps/Watt considering a 1024^2 pixel image.

For illustrative purposes, Fig. 9 shows the energy consumption on average for three different simulated datasets (100 × 100, 100 × 500, and 400 × 500 pixels) with 10, 20, and 30 endmembers. In this figure, the dependence with the content of the hyperspectral image unmixed is shown. Evaluating these

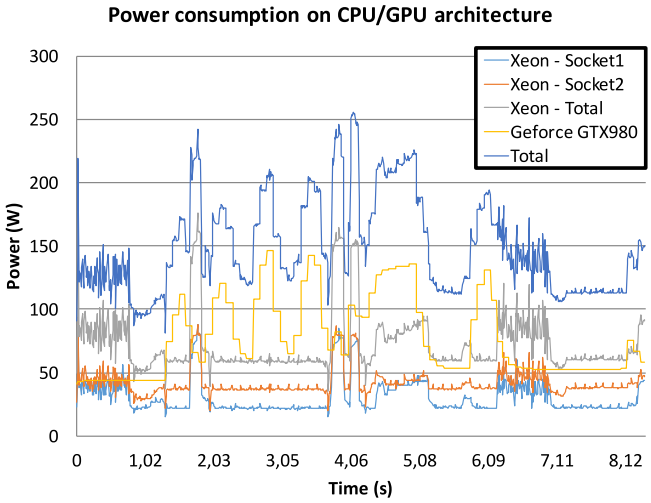

Fig. 8. Power consumption of the proposed full unmixing chain executed on the CPU/GPU architecture considering the simulated dataset.

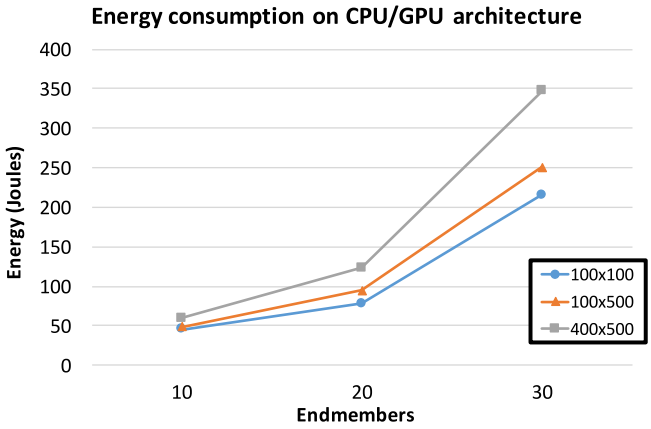

Fig. 9. Energy consumption of the proposed full unmixing chain executed on CPU/GPU architecture considering three different simulated datasets (100 × 100, 100 × 500, and 400 × 500 pixels) with 10, 20, and 30 endmembers.

results, we can see that the number of endmembers is decisive for large datasets increasing the energy consumption by the processed chain. However, the size of the image impacts to a lesser measure. These results can be compared with a previous effort presented in [23], observing similar trends with regards to those analyzed in this study based on different architectures.

V. Conclusion and Future Research Lines

In this paper, we have developed a new parallel implementation of a full hyperspectral unmixing chain using ad-hoc OpenCL kernels and the `clMAGMA` library on heterogeneous platforms. This chain is made up of the GENE algorithm for estimating the number of endmembers, the SGA for identifying the endmember signatures, and the SCLSU for estimating the fractional abundance of each endmember. The parallel unmixing chain, which is one of the first ones of its kind developed in OpenCL, fully meets real-time constraints for both data whose processing times are below acquisition times using GPU platform. As future research, we plan to use additional real scenes and another algorithms using low-power HPC devices to be used on onboard processing modules in airborne and (particularly) spaceborne Earth observation missions.

Acknowledgment

The authors would like to take this opportunity to gratefully thank the Associate Editor and the Anonymous Reviewers for their detailed comments and suggestions, which greatly helped them to improve the technical quality and presentation of this manuscript.

References

[1] A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*. Boca Raton, FL, USA: Taylor & Francis, 2007.
[2] J. M. Bioucas-Dias *et al.*, "Hyperspectral unmixing: Geometrical, statistical, and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 354–379, Apr. 2012.
[3] S. Bernabe, S. Sanchez, A. Plaza, S. Lopez, J. A. Benediktsson, and R. Sarmiento, "Hyperspectral unmixing on GPUs and multi-core processors: A comparison," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 3, pp. 1386–1398, Jun. 2013.
[4] E. Torti, G. Danese, F. Leporati, and A. Plaza, "A hybrid CPU-GPU real–time hyperspectral unmixing chain," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 2, pp. 945–951, Feb. 2016.
[5] C. Gonzalez, S. Lopez, D. Mozos, and R. Sarmiento, "FPGA implementation of the HySime algorithm for the determination of the number of endmembers in hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2870–2883, Jun. 2015.
[6] C. Gonzalez, S. Bernabe, D. Mozos, and A. Plaza, "FPGA implementation of an algorithm for automatically detecting targets in remotely sensed hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 9, pp. 4334–4343, Sep. 2016.
[7] C. Gonzalez, J. Resano, A. Plaza, and D. Mozos, "FPGA implementation of abundance estimation for spectral unmixing of hyperspectral data using the image space reconstruction algorithm," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 1, pp. 248–261, Feb. 2012.
[8] S. Bernabe, F. D. Igual, G. Botella, C. Garcia, M. Prieto-Matias, and A. Plaza, "Performance portability study of an automatic target detection and classification algorithm for hyperspectral image analysis using OpenCL," *Proc. SPIE*, vol. 9646, pp. 1–9, 2015.
[9] G. M. Callico, S. Lopez, B. Aguilar, J. F. Lopez, and R. Sarmiento, "Parallel implementation of the modified vertex component analysis algorithm for hyperspectral unmixing using OpenCL," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 8, pp. 3650–3659, Aug. 2014.
[10] S. Bernabe, F. D. Igual, G. Botella, M. Prieto-Matias, and A. Plaza, "Parallel implementation of the multiple endmember spectral mixture analysis algorithm for hyperspectral unmixing," *Proc. SPIE*, vol. 9646, pp. 1–6, 2015.
[11] S. Bernabe *et al.*, "A fast parallel hyperspectral coded aperture algorithm for compressive sensing using OpenCL," in *Proc. IEEE 16th Int. Conf. Comput. as a Tool*, 2015, pp. 1–6.
[12] C. Rodriguez-Doñate, G. Botella, C. Garcia, E. Cabal-Yepez, and M. Prieto-Matias, "Early experiences with OpenCL on FPGAs: Convolution case study," in *Proc. IEEE 23rd Int. Symp. Field-Programmable Custom Comput. Mach.*, 2015, p. 235.
[13] A. Ambikapathi, T.-H. Chan, C.-Y. Chi, and K. Keizer, "Hyperspectral data geometry-based estimation of number of endmembers using p-norm-based pure pixel identification algorithm," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 5, pp. 2753–2769, May 2013.
[14] S. Bernabe, G. Martin, G. Botella, M. Prieto-Matias, and A. Plaza, "Parallel implementation of a hyperspectral data geometry-based estimation of number of endmembers algorithm," *Proc. SPIE*, vol. 9897, pp. 1–7, 2016.
[15] Q. Du and C.-I. Chang, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 608–619, Mar. 2004.
[16] J. M. Bioucas-Dias and J. M. P. Nascimento, "Hyperspectral subspace identification," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 8, pp. 2435–2445, Aug. 2008.

[17] C.-I. Chang, C. Wu, W. Liu, and Y. C. Ouyang, "A new growing method for simplex-based endmember extraction algorithms," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 10, pp. 2804–2819, Oct. 2006.
[18] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York, NY, USA: Kluwer, 2003.
[19] H. Ren and C.-I. Chang, "Automatic spectral target recognition in hyperspectral imagery," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1232–1249, Oct. 2003.
[20] S. Bernabe, S. Lopez, A. Plaza, and R. Sarmiento, "GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 221–225, Mar. 2013.
[21] G. S. Miller, "The definition and rendering of terrain maps," *ACM SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 39–48, 1986.
[22] S. Barrachina *et al.*, "An integrated framework for power-performance analysis of parallel scientific workloads," in *Proc. 3rd Int. Conf. Smart Grids, Green Commun. IT Energy-aware Technol.*, 2013, pp. 114–119.
[23] S. Sanchez, G. Leon, A. Plaza, and E. Quintana-Orti, "Assessing the performance-energy balance of graphics processors for spectral unmixing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2305–2316, Jun. 2014.

Sergio Bernabé received the Computer Engineering degree and the M.Sc. degree in computer engineering from the University of Extremadura, Cáceres, Spain, in 2010, and the joint Ph.D. degree from the University of Iceland, Reykjavík, Iceland, and the University of Extremadura, Badajoz, Spain, in 2014.

He has been a Visiting Researcher at the Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, Las Palmas, Spain, and also at the Computer Vision Laboratory, Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil. He was a Postdoctoral Researcher (funded by FCT) with the Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal, and a Postdoctoral Researcher (funded by the Spanish Ministry of Economy and Competitiveness) with the Complutense University of Madrid, Madrid, Spain. His research interests include the development and efficient processing of parallel techniques for different types of high-performance computing architectures.

Dr. Bernabé is an Active Reviewer of international conferences and international journals, including the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATION AND REMOTE SENSING (JSTARS), the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS. He received Best Paper Award of the JSTARS journal in 2013 and the Best Ph.D. Dissertation Award at the University of Extremadura, Cáceres, Spain, in 2015.

Guillermo Botella received the M.A.Sc. degree in physics in 1999, the M.A.Sc. degree in electronic engineering in 2001, and the Ph.D. degree in computer engineering in 2007, all from the University of Granada, Granada, Spain.

He was a Research Fellow funded by EU working with the Department of Architecture and Computer Technology, Universidad de Granada, and the Vision Research Laboratory, University College London, London, U.K. Then, he joined the Department of Computer Architecture and Automation, Complutense University of Madrid, Madrid, Spain, as an Assistant Professor. From 2008 to 2012, he was a Visiting Professor with the Department of Electrical and Computer Engineering, Florida State University, Tallahassee, FL, USA. His current research interests include digital signal processing for very large scale integration, field-programmable gate arrays, GPGPUs, vision algorithms, and IP protection of digital systems.

Gabriel Martín received the Computer Engineering degree, M.Sc., and Ph.D. degrees in computers and communications technologies from the University of Extremadura, Cáceres Spain, in 2008, 2010, and 2013, respectively. He has obtained several prices for his Ph.D. dissertation such as the "Best Iberian Ph.D. dissertation in information system and technolgies" awarded by the Iberian Association for Information Systems and Technologies, and the "Outstanding Ph.D. dissertation award" by the University of Extremadura.

He was a Predoctoral Research Associate (funded by the Spanish Ministry of Science and Innovation) with the Hyperspectral Computing Laboratory, University of Extremadura, and is now a Postdoctoral Researcher (funded by FCT) with the Instituto de Telecomunicaçoes, Lisbon, Portugal. His research interests include the development of new techniques for unmixing and compressive sensing of hyperspectral data sets, as well as the efficient processing and interpretation of these data in different types of high-performance computing architectures. Dr. Gabriel Martín has authored or co-authored more than 38 scientific publications.

Dr. Martín has served as a Reviewer for the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING and the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.

Manuel Prieto-Matias received the Ph.D. degree in computer science from the Complutense University of Madrid (UCM), Madrid, Spain, in 2000.

He is now associate professor in the Department of Computer Architecture at UCM and serves as vice-dean for External Relations and Research at the School of Computer Science and Engineering. His research interests include areas of parallel computing and computer architecture. Most of his activities have focused on leveraging parallel computing platforms and on complexity-effective micro-architecture design. His current research addresses emerging issues related to asymmetric processors, heterogeneous systems and energy-aware computing, with a special emphasis on the interaction between the system software and the underlying architecture. He has co-written numerous articles in journals and for international conferences in the field of parallel computing and computer architecture. He is a member of the ACM and IEEE Computer Society.

Antonio Plaza (M'05–SM'07–F'15) received the M.Sc. degree in 1999 and the Ph.D. degree in 2002 from the University of Extremadura, Badajoz, Spain, both in computer engineering.

He is the Head of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. He has authored more than 600 publications, including 197 JCR journal papers (more than 140 in IEEE journals), 23 book chapters, and 285 peer-reviewed conference proceeding papers. He has guest edited ten special issues on hyperspectral remote sensing for different journals. His main research interests include hyperspectral data processing and parallel computing of remote sensing data.

Prof. 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), for which he served as Associate Editor in 2007–2012. He is also an Associate Editor for IEEE ACCESS. He was a member of the Editorial Board of the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER (2011–2012) and the IEEE GEOSCIENCE AND REMOTE SENSING MAGAZINE (2013). He was also a member of the steering committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS). He received the Best Column Award of the IEEE SIGNAL PROCESSING MAGAZINE in 2015, the 2013 Best Paper Award of the JSTARS journal, and the most highly cited paper (2005–2010) in the *Journal of Parallel and Distributed Computing*. He received Best Paper Awards at the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology. He served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) in 2011–2012 and as the President of the Spanish Chapter of IEEE GRSS in 2012–2016. He has reviewed more than 500 manuscripts for more than 50 different journals. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.