OpenCL-library-based Implementation of SCLSU Algorithm for Remotely Sensed Hyperspectral Data Exploitation: clMAGMA versus viennaCL

Sergio Bernabé[1], Guillermo Botella[1], Carlos Orueta[1], José M. R. Navarro[1], Manuel Prieto-Matias[1] and Antonio Plaza[2]

[1]Department of Computer Architecture and Automation, Complutense University of Madrid, E-28040 Madrid, Spain

[2]Hyperspectral Computing Laboratory, University of Extremadura, E-10003 Cáceres, Spain

ABSTRACT

In the last decade, hyperspectral spectral unmixing (HSU) analysis have been applied in many remote sensing applications. For this process, the linear mixture model (LMM) has been the most popular tool used to find pure spectral constituents or endmembers and their fractional abundance in each pixel of the data set. The unmixing process consists of three stages: (i) estimation of the number of pure spectral signatures or endmembers, (ii) automatic identification of the estimated endmembers, and (iii) estimation of the fractional abundance of each endmember in each pixel of the scene. However, unmixing algorithms can be very expensive computationally, a fact that compromises their use in applications under real-time constraints. This is, mainly, due to the last two stages in the unmixing process, which are the most consuming ones. In this work, we propose parallel opencl-library-based implementations of the sum-to-one constrained least squares unmixing (P-SCLSU) algorithm to estimate the per-pixel fractional abundances by using mathematical libraries such as clMAGMA or ViennaCL. To the best of our knowledge, this kind of analysis using OpenCL libraries have not been previously conducted in the hyperspectral imaging processing literature, and in our opinion it is very important in order to achieve efficient implementations using parallel routines. The efficacy of our proposed implementations is demonstrated through Monte Carlo simulations for real data experiments and using high performance computing (HPC) platforms such as commodity graphics processing units (GPUs).

Keywords: Hyperspectral imaging, spectral unmixing, high performance computing (HPC), OpenCL, clMAGMA, viennaCL

1. INTRODUCTION

Spectral unmixing analysis[1] have been a hot topic in the last decade using hyperspectral imaging. This kind of images help in many remote sensing applications[2] through its high spectral resolution. However, this resolution presents the following problem: "mixed pixels". To solve this problem, the community is using linear and nonlinear spectral unmixing approaches.[3,4] In this work, linear mixture model (LMM) is used to solve the main problem.

In practice, the LMM is more flexible to be adapted to different analysis scenarios and the most used in the community to unmix remotely sensed hyperspectral data. This process is composed of three stages: 1) estimation of the number of pure spectral signatures (endmembers); 2) automatic identification of the estimated endmembers, and (3) estimation of the fractional abundance of each endmember in each pixel of the scene. In each stage, several methods have been proposed using parallel techniques to accelerate the process and use it in applications under real-time and energy/power constraints.[5]

Different parallel frameworks have been used to accelerate the previous stages: OpenMP, CUDA, VHDL or Verilog. However, OpenCL is a free alternative to the previous frameworks whose main advantages concern the shorter time to market, faster implementations and acceptable performance portable codes[6] on different heterogeneous systems such as graphics processing units (GPUs),[7] multi-core processors,[8] the Intel Xeon Phi[9] and other custom devices.[10] Based on these advantages, OpenCL is selected to be used in the last stage in a fully

High-Performance Computing in Geoscience and Remote Sensing VI, edited by Bormin Huang, Sebastián López, Zhensen Wu, Jose M. Nascimento, Jun Li, Valeriy V. Strotov, Proc. of SPIE Vol. 10007, 100070B
 · doi: 10.1117/12.2241524

operational unmixing chain whose two first steps have been implemented trough OpenCL using high performance computing (HPC) platforms*.

In this paper, we propose a parallel implementation of the sum-to-one constrained least squares unmixing (P-SCLSU) algorithm for hyperspectral unmixing on different platforms using the OpenCL framework. Mathematical libraries such as clMAGMA[11] or ViennaCL have been used to compute matrix products operations to accelerate the most time-consuming parts. Experimental results conducted on a heterogeneous platform equipped with an NVidia GeForce GTX 980 GPU and 2×Intel Xeon E5-2695 multi-core processors reveal satisfactory performance results despite the data transfers overheads between the device and the host memories.

The rest of this paper is organized as follows: Section 2 details the SCLSU method. Section 3 describes the proposed parallel implementation. Section 4 presents experimental results of the proposed implementation in terms of accuracy, parallel performance and energy consumption using real data set on different heterogeneous platforms. Finally, Section 5 presents a few concluding remarks and possible future research lines.

2. SCLSU METHOD

The SCLSU algorithm was proposed in[12] for spectral unmixing. This method belongs to the third stage in the unmixing process called abundance extraction algorithms (AEA), which pseudocode can be seen in Algorithm 1. This method imposes the abundance sum-to-one constraint (ASC), while ignoring the abundance nonnegativity constraint (ANC). 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 the estimated abundance fractions are nonnegative as in nonnegativity constrained least squares unmixing (NCLSU) or fully constrained least squares unmixing (FCLSU) algorithms.[12] The SCLU algorithm has been computed efficiently as described in Algorithm 1, 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 1 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 more 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 $\mathbf{E}^{\#}$.

Algorithm 1 : Pseudocode of the Sum-to-one Constrained Least Squares Unmixing algorithm

1: INPUT: $\mathbf{Y}, \mathbf{E}$
   % $\mathbf{Y}$ is $L \times n_s$ matrix with the hyperspectral data set, 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: $\mathbf{A} := \mathbf{G}\mathbf{E}^T\mathbf{Y} + \mathbf{F}\mathbf{1}$

3. OPENCL IMPLEMENTATION

In this section, we show our mapping of the P-SCLSU algorithm, where a hybrid approach that only uses the device to accelerate the main parts of the algorithm has been opted (see Fig. 1). We have identified those stages with a profiling of an optimized serial SCLSU implementation based on the BLAS library. As described below, some of them can be efficiently implemented on heterogeneous platforms using mathematical libraries

*http://eprints.ucm.es/38483/1/TG%202016-18.pdf

such as clMAGMA and viennaCL for OpenCL. Before explaining the parallel implementation, we will describe the OpenCL framework.

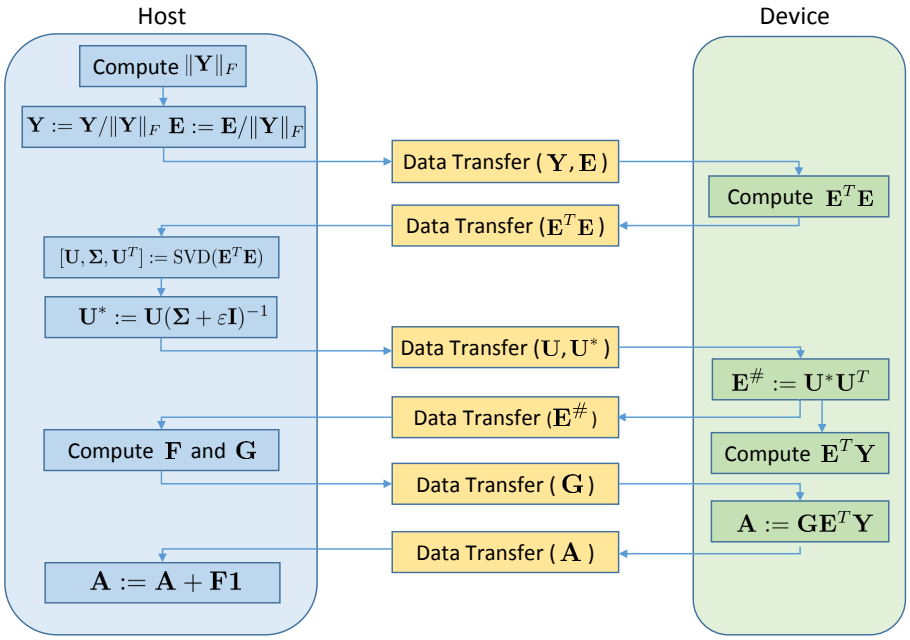


Figure 1. Block diagram illustrating our mapping of the P-SCLSU method using OpenCL.

3.1 Framework

OpenCL (Open Computing Language) is an open and royalty-free standard based on C99 that allows the execution of parallel programs on heterogeneous platforms. It is currently supported by several hardware devices, such as CPUs, GPUs, FPGAs, DSPs and other processors or hardware accelerators. This standard is based on the host-device model, where our program is divided in regions executed on the host and other regions, called kernel to express the parallelism, in the compute device. The OpenCL standard facilitates the usage of parallelism with vector types, operations, synchronization and functions to work with *work-items* and *work-groups*. 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, OpenCL defines four different memory spaces for the compute device. Global memory is read-write accessible by all processing elements across all *work-groups*, and it usually corresponds to the DRAM memory device, which carries a high access latency. Local memory is shared by a group of processing elements, and it usually involves 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*.

3.2 Libraries

For this work, two different libraries have been considered: viennaCL 1.7.1 and clMAGMA 1.3.0. Both libraries are used to compute matrix products in a faster way. ViennaCL[†] is a free open-source linear algebra library for computations on many-core architectures. This library is written in C++ and supports several parallel programming languages such as OpenCL. On the other hand, clMAGMA[‡] is an OpenCL port of MAGMA. MAGMA is a project to create a new generation of linear algebra libraries, in particular optimized GPU OpenCL BLAS and CPU optimized BLAS and LAPACK libraries that achieves the fastest possible time to an accurate solution on heterogeneous architectures.

[†]http://viennacl.sourceforge.net/
[‡]http://icl.cs.utk.edu/magma/software/view.html?id=190

3.3 Parallel implementation

By observing the program flow in Algorithm 1, it is possible to identify the main potential bottlenecks in the SCLSU algorithm (profiling the C Code) highlighted in green. In this case, we focused on different matrix multiplications where both clMAGMA and viennaCL libraries have been evaluated from a performance point of view.

Firstly and following Fig. 1, the implementation starts diving the elements of both the endmembers and the data by the Frobenius norm of $\mathbf{Y}$. After that, the next step is to compute the pseudo-inverse of the endmember matrix $\mathbf{E}$. In our implementation, this step is carried out through the Singular Value Decomposition (SVD) of $\mathbf{E}^T\mathbf{E}$ as follows:

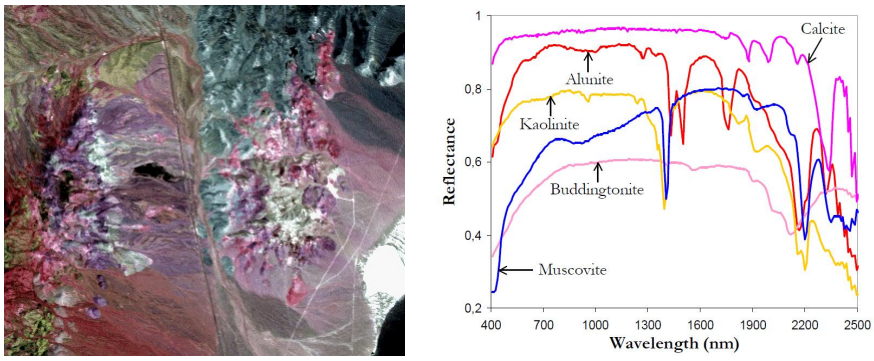
$$\mathbf{E}^{\#} \equiv (\mathbf{E}^T\mathbf{E})^{-1} \equiv \mathbf{U}(\mathbf{\Sigma} + \varepsilon\mathbf{I})^{-1}\mathbf{U}^T \tag{1}$$

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 Figure 1 shows, the matrix multiplication $\mathbf{E}^T\mathbf{E}$ is performed in the GPU while the SVD decomposition is performed in the CPU because of the small size of the matrices promotes low occupancy in the GPU for the SVD operation. Later the matrix $\mathbf{E}^{\#}$ is obtained by performing the right hand matrix multiplication of the eigenvectors in the GPU. After that, the more computationally expensive part of the algorithm which is the computation of $\mathbf{E}^T\mathbf{Y}$ is performed in the GPU while asynchronously the CPU is computing the matrices $\mathbf{F}$ and $\mathbf{G}$. Once the matrix $\mathbf{G}$ has been computed and transferred to the GPU, it computes the matrix multiplication between $\mathbf{G}$ and $\mathbf{E}^T\mathbf{Y}$, which is already computed and stored in the GPU. Finally the abundances are obtained by adding the term $\mathbf{F1}$ to the result of the product.

4. EXPERIMENTAL RESULTS

4.1 Real Data set

The experiments are carried out using the well-known AVIRIS Cuprite scene [see Fig. 2(a)], which has been widely used to study the accuracy of unmixing algorithms. This data set is available online in reflectance units after atmospheric correction. The portion used in our experiments corresponds to a 350 × 350 pixels 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*. Fig. 2(b) displays the reference ground signatures of the above minerals, which are available in the United States Geological Survey (USGS) library [§].



(a) (b)

Figure 2. (a) False color composition of an AVIRIS hyperspectral image collected by NASA's Jet Propulsion Laboratory over the Cuprite mining district in Nevada. (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

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

4.2 Work Environment

To carry out the tests on OpenCL, we have used a multicore heterogeneous system equipped with: two Intel Xeon E5-2695 v3 at 2.30 GHz with 64 GBytes of DDR3 RAM memory and an NVidia GeForce GTX 980 GPU with 2048 cores operating at 1.126 GHz and dedicated memory of 4 GBytes.

4.3 Accuracy Evaluation

The results of spectral unmixing can be evaluated in terms of the quality of the reconstruction of the original data set using the extracted endmembers, the estimated fractional abundances, and the linear mixture model. In this work, the metric employed to evaluate the goodness of the reconstruction is the root mean square error (RMSE) obtained after comparing the original scene with the reconstructed one. This metric showed in Eq. (2) 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 the reconstructed version on a per-pixel basis.

$$RMSE(Y,\hat{Y}) \leftarrow \frac{1}{n_s}\sum_{i=0}^{n_s}(\sum_{j=0}^{L}(y_i^j - \hat{y}_i^j)^2)^{\frac{1}{2}} \tag{2}$$

For illustrative purposes, Fig. 3 represents graphically the per-pixel RMSE obtained in the reconstruction process for real hyperspectral data set. 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.

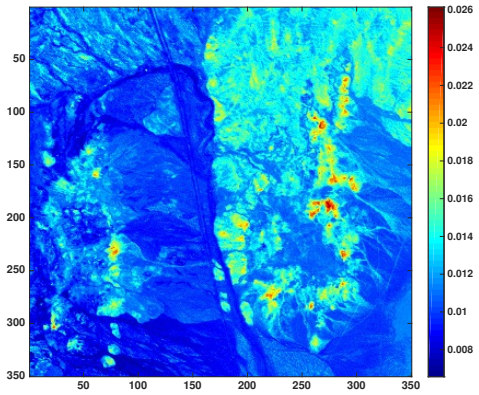


Figure 3. Per-pixel RMSE obtained in the reconstruction process for the AVIRIS Cuprite scene whose overall RMSE was 0.0115.

4.4 Performance Evaluation

In order to analyze the performance of the proposed implementation, it is important to emphasize that our parallel SCLSU provides exactly the same results as the serial approach of the algorithm, implemented using C code and the GNU `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 1 shows partial times, where *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. In our experiments, clMAGMA library offers a good performance compared to viennaCL using GPU platform and real hyperspectral data set. Observing the time results, clMAGMA obtains a better performance using GPU platform. However, viennaCL is very fast executing on device but the transfers data penalize its use. So, clMAGMA library is a good choice when we have several data transfers and OpenCL is needed.

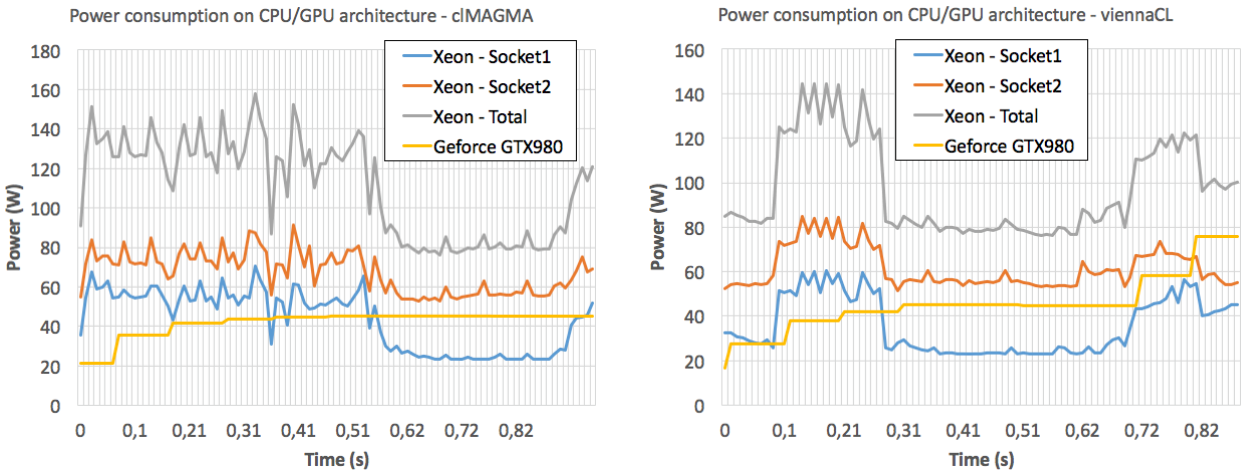
Table 1. Processing times (in seconds) and speedups achieved for the proposed implementations in GPU platform based on the Algorithm 1 and tested on the AVIRIS Cuprite scene.

	Xeon CPU	GTX980 GPU	
	Serial	clMAGMA	viennaCL
Transfers	–	0.047 (31.09%)	0.333 (80.14%)
Host	0.226	0.076 (50.53%)	0.075 (18.06%)
Device	–	0.028 (18.38%)	0.008 (1.80%)
Total Time	0.226	0.150	0.415
Speedup ($\frac{Serial\_Time}{Device\_Time}$)	–	1.51x	–

4.5 Power Consumption Evaluation

In our experiments, the power consumption was measured using the software solution *PowerMeter daemon* (*pmlib*).[13] Gathering power results periodically from the tools provided by manufacturers, namely: RAPL for the Intel Xeon CPU and NVML for the NVidia GPU.

Fig. 4 shows power consumption for different settings. If we analyze the graphics, Fig. 4(a) using clMAGMA library dissipates 152.59 Watts on average (22.89 Joules), and 201.77 Watts at most. On the other hand, Fig. 4(b) using viennaCL library dissipates 143.19 Watts on average (59.42 Joules), and 197.30 Watts at most. Comparing both results, the best trade-off solution between performance measured in Mega pixel per second (Mpps) and power consumption is achieved by clMAGMA implementation: 0.008 Mpps/Watt vs 0.025 Mpps/Watt for viennaCL implementation considering a 1024^2 pixel image.



(a) (b)

Figure 4. Power consumption of the P-SCLSU algorithm executed on CPU/GPU architecture with both clMAGMA (left) and viennaCL (right) libraries and considering the AVIRIS Cuprite scene.

5. CONCLUSIONS AND FUTURE RESEARCH LINES

In this work, we have evaluated accuracy, performance and power consumption of a parallel implementation of the sum-to-one constrained least squares unmixing (P-SCLSU) algorithm for spectral unmixing. Our implementation uses mathematical libraries such as clMAGMA and viennaCL to accelerate the most time-consuming parts of this method. P-SCLSU using clMAGMA achieves the best trade-off solution between performance and power consumption on GPU platforms compared to viennaCL library. As future work, we plan to map P-SCLSU into other computing platforms using the OpenCL standard and additional real scenes.

6. ACKNOWLEDGEMENT

This work has been supported by the EU (FEDER) and the Spanish MINECO, under grants TIN2015-65277-R, TIN2012-32180 and the Formación Posdoctoral programme (FPDI-2013-16280).

REFERENCES

1. J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing: geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **5**(2), pp. 354–379, 2012.
2. Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **7**(6), pp. 2094–2107, 2014.
3. N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Processing Magazine* **19**(1), pp. 44–57, 2002.
4. N. Dobigeon, J.-Y. Tourneret, C. Richard, J. C. M. Bermudez, S. McLaughlin, and A. O. Hero, "Nonlinear unmixing of hyperspectral images: Models and algorithms," *IEEE Signal Processing Magazine* **31**(1), pp. 82–94, 2014.
5. E. Torti, G. Danese, F. Leporati, and A. Plaza, "A hybrid CPU-GPU real-time hyperspectral unmixing chain," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **9**(2), pp. 945–951, 2016.
6. 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," *Proceedings of the SPIE Conference on High-Performance Computing in Remote Sensing* **9646**, pp. 1–9, 2015.
7. 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 Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **7**(8), pp. 3650–3659, 2014.
8. 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," *Proceedings of the SPIE Conference on High-Performance Computing in Remote Sensing* **9646**, pp. 1–6, 2015.
9. S. Bernabe, G. Martin, J. M. P. Nascimento, J. M. Bioucas-Dias, A. Plaza, G. Botella, and M. Prieto-Matias, "A fast parallel hyperspectral coded aperture algorithm for compressive sensing using opencl," *IEEE 16th International Conference on Computer as a Tool* , pp. 1–6, 2015.
10. C. Rodriguez-Doñate, G. Botella, C. Garcia, E. Cabal-Yepez, and M. Prieto-Matias, "Early experiences with opencl on fpgas: convolution case study," *IEEE 23rd International Symposium on Field-Programmable Custom Computing Machines* , p. 235, 2015.
11. C. Cao, J. Dongarra, P. Du, M. Gates, L. Piotr, and S. Tomov, "clMAGMA: high performance dense linear algebra with OpenCL," *Proceedings of the International Workshop on OpenCL (IWOCL)* , pp. 1–9, 2014.
12. C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, Kluwer Academic: New York, 2003.
13. S. Barrachina, M. Barreda, S. Catalan, M. Dolz, G. Fabregat, R. Mayo, and E. Quintana-Orti, "An integrated framework for power-performance analysis of parallel scientific workloads," *Smart Grids, Green Communications and IT Energy-aware Technologies. 3rd Int. Conf. on* , pp. 114–119, 2013.