Multicore Real-Time Implementation of a Full Hyperspectral Unmixing Chain

Sergio Bernabé, Luis Ignacio Jiménez, Carlos García, Javier Plaza, *Senior Member, IEEE*, and Antonio Plaza, *Fellow, IEEE*

***Abstract*— Solving the mixture problem in remotely sensed hyperspectral images remains a challenging task. In particular, solutions are needed in order to obtain a response for applications with real-time constraints. In the last decade, several efforts have been developed, many of them using graphics processing units (GPUs) and focused on the exploitation of spectral information alone. However, a few spectral unmixing chains have been developed using other architectures such as multicore processors, field programmable gate arrays, or Intel Xeon Phi coprocessors. In this letter, we develop a new parallel unmixing chain for multicore processors. Compared with other approaches, the proposed spatial–spectral alternative takes advantage of the complementary information provided by the spatial correlation of the pixels in the image in addition to the spectral information. Our implementation has been optimized using the application program interface OpenMP and the Intel Math Kernel Library on two multicore architectures, and using real analysis scenarios. The results reveal competitive real-time performance compared with another compute unified device architecture implementation previously developed for GPUs.**

***Index Terms*— Graphics processing units (GPUs), hyperspectral unmixing, multicore processors.**

I. Introduction

HYPERSPECTRAL imaging sensors collect a large number of images in different wavelength channels for the same area of the surface of the Earth [1], [2]. As a consequence, each pixel of the image is characterized by a vector with hundreds of components representing the materials in these areas at different wavelength values as a spectral signature. These signatures provide distinguishing features and allow us to describe the imaged materials in great level of detail [3], [4]. Accordingly, a hyperspectral image can be represented as a 3-D data cube, where the first two dimensions represent the spatial coordinates in a 2-D space and the last one captures the spectral singularity of each pixel in the scene [5], [6].

Spectral unmixing is among the most popular techniques to process hyperspectral images. In the literature, two models have been mainly used to characterize mixed pixels in hyperspectral images. The nonlinear mixture model [7] assumes that the interactions between the endmembers follow a nonlinear model with multiple scattering effects [8] and other nonlinearities. The linear mixture model [9] considers that the mixed pixels present in the scene can be modeled as a linear combination of the pure spectral signatures (endmembers) weighted by their corresponding abundances. The linear mixture model can be expressed in mathematical form as follows:

$$\mathbf{Y} = \mathbf{EA} + \mathbf{N} \tag{1}$$

where $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$, $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p]$ is a matrix of l-dimensional endmember signatures containing p endmembers, $\mathbf{A} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ contains the abundance fractions α_i associated with each endmember in each pixel of the scene, and $\mathbf{N}$ accounts for the noise.

Many linear spectral unmixing techniques exhibit a high computational cost but, at the same time, need to satisfy the demands of real-time applications. To address this problem, several parallel systems have been exploited to achieve real-time performance; from small clusters of computers and general-purpose multicore processors to specific accelerators such as field programmable gate arrays (FPGAs), graphics processing units (GPUs) or Intel Xeon Phi coprocessors.

The goal of this letter is to explore the possibility of achieving real-time spectral unmixing on the state-of-the-art multicore architectures, which are quite general and flexible. Our aim is to optimize these architectures by developing a parallel version of a variety of techniques that cover all the stages of the hyperspectral unmixing chain. A comparative evaluation of the performance of our newly developed multicore implementations against single core and GPU versions developed in previous works is also carried out. The analysis is conducted using real hyperspectral scenes widely used as benchmarks.

The remainder of this letter is structured as follows. Section II briefly describes our hyperspectral unmixing chain. Section III is focused on the parallel implementation of the techniques proposed for each stage of the unmixing chain. In Section IV, we describe the experimental process conducted and analyze the results obtained. Finally, Section V concludes with some remarks and hints at plausible future research.

Manuscript received November 15, 2017; revised January 29, 2018; accepted February 16, 2018. Date of publication March 19, 2018; date of current version April 20, 2018. This work was supported by EU (FEDER) and the Spanish MINECO under Grant TIN 2015-65277-R. *(Corresponding author: Sergio Bernabé.)*

S. Bernabé and C. García are with the Department of Computer Architecture and Automation, Complutense University of Madrid, 28040 Madrid, Spain (e-mail: sebernab@ucm.es; garsanca@ucm.es).

L. I. Jiménez is with the Research, Technological Innovation and Supercomputing Center of Extremadura, 10071 Cáceres, Spain (e-mail: luisignacio.jimenez@cenits.es).

J. Plaza and A. Plaza are with the Hyperspectral Computing Laboratory, University of Extremadura, E-10071 Cáceres, Spain (e-mail: jplaza@unex.es; aplaza@unex.es).

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

Digital Object Identifier 10.1109/LGRS.2018.2810600

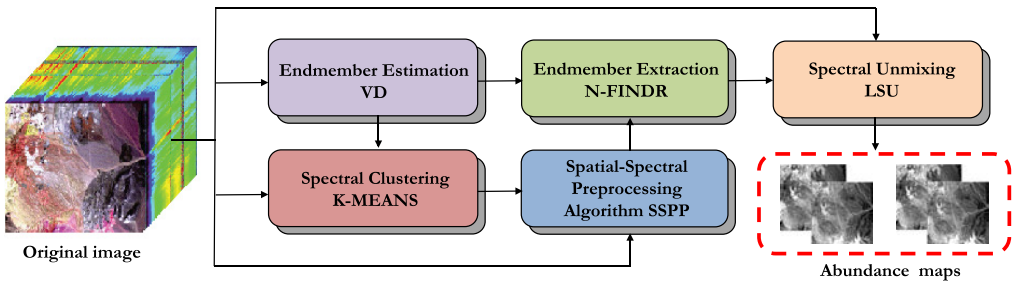


Fig. 1. Graphical illustration of the different stages involved in the considered hyperspectral unmixing chain.

II. Hyperspectral Unmixing Chain

Grounding this letter on the premises of the linear mixture model, a series of stages are differentiated in the unmixing chain as a set of methods with a specific purpose and with the ultimate goal of extracting the pure spectral signatures in the scene. *Endmember estimation* can be considered an initial stage, which estimates the number of pure signatures present in the hyperspectral image [10], [11]. For this purpose, we selected the virtual dimensionality (VD) [12], which calculates the eigenvectors of the correlation and the covariance matrices for each band of the hyperspectral image. This number is used in subsequent steps such as the number of endmembers to be extracted or the number of bands retained after a spectral reduction. *Image preprocessing* is considered an optional step and could be inserted not necessarily after the endmember estimation. The main goal of this stage is to reduce the complexity of the hyperspectral images spatially discarding a set of pixels as candidates to be endmembers or spectrally reducing the number of bands to be processed in other algorithms. As a direct consequence, the computation process in subsequent steps will be reduced. In this letter, we used the spatial–spectral preprocessing (SSPP) algorithm [13] for this purpose. As its name suggests, it works on both domains, selecting a set of pixel candidates for each class contained in the image based on a previous clustering process. To generate the cluster map needed by the SSPP, we use the noted k-means [14] algorithm. The *endmember extraction* step is focused on the selection of the final spectral signatures, which are going to be used as endmembers. Many approaches were developed on the literature such as the N-FINDR [15], which uses a technique based on identifying the simplex with maximum volume that can be formed with the data points but working with a spectrally reduced version of the original image [i.e., principal component analysis (PCA) [16] domain]. This is the main reason for which it has been selected in this letter. It should be noted that the PCA is also computed by the SSPP algorithm, so both methods can be overlapped and the spatial reduction computed by the SSPP will lead to faster results in the N-FINDR. The final process of *Abundance Estimation* defines the concentration of the endmembers selected on each pixel in the hyperspectral image. The popular least squares unmixing method [5] is considered in this letter.

The selection of the different blocks of the unmixing chain considered in this letter was done based on the following reasons. First and foremost, we can perform a comparison of our implementation to a recent GPU version of the same chain in [17]. Second, our adopted chain is quite general, including all the steps necessary for a full unmixing framework. These individual stages can be easily replaced by other equivalent algorithms but are characterized by their high computational cost. This makes our selection flexible while representative enough of the complexity of unmixing algorithms.

Most hyperspectral unmixing methods, regardless of the kind of unmixing chain that they implement, are very demanding in terms of computational complexity. The trend is to develop solutions that increase the efficiency of these methods to be able to adapt them to the increasing computational demands of real-time applications. In Section III, a new multicore parallel implementation of the unmixing chain described above (summarized in Fig. 1) is proposed.

III. Parallel Implementation

In this section, a parallel implementation of the proposed unmixing chain will be described. For this purpose, OpenMP API and the Intel Math Kernel Library, which includes the well-known linear algebra package (LAPACK) library, are used to take advantage of the underlying hardware parallelism. In Sections III-A–III-E, we summarize the main techniques used for the multicore implementation.

A. Parallel Virtual Dimensionality Algorithm

The stages for this algorithm are implemented using different strategies. First, we need to compute the covariance matrix $\mathbf{K}$. It involves to compute the mean value for each band, which is optimized with a *#pragma parallel for*. Then, a group of elements are vectorized using a *#pragma omp simd* in order to subtract the mean value computed to all the pixels in the same band $\bar{\mathbf{Y}}$. The resulting matrix is multiplied by its transpose using the *cblas_dgemm* routine. The next step consists in calculating the correlation matrix $\mathbf{R}$. For this purpose, the parallel coding is carried out with a *#pragma parallel for*, where a group of elements are vectorized by means of *#pragma omp simd* to get $\mathbf{R}_{ij} = \mathbf{K}_{ij} + \bar{\mathbf{Y}}_i \bar{\mathbf{Y}}_j$. Finally, the correlation-eigenvalues and covariance-eigenvalues for SVD calculation are computed by the *LAPACKE_dgesvd* function. After that, the Neyman–Pearson test for the estimation of the number of endmembers is parallelized with a *#pragma parallel for*, where unrolling has been used to optimize it.

B. Parallel k-Means Algorithm

This algorithm is based on several reduction stages, which supposes a limit in terms of scalability. The strategy is based

on the data memory layout change to hide its dependences in successive reduction stages, and thus to achieve improved efficiency rates. Initially, the Euclidean norm for the matrix **Y** is established as a maximum threshold. For this purpose, the bands' distribution processing is carried out by *#pragma parallel for reduction*, where a group of elements are also vectorized using a *#pragma omp simd*. Subsequently, the band sequential image layout is changed to band interleaved by pixel using a *#pragma parallel for*. This aspect is crucial to hide data dependences in the next reduction stages, which are also parallelized obtaining the possible image centroids.

C. Parallel Spatial–Spectral Preprocessing Algorithm

Different strategies to parallelize the stages of this algorithm are studied. First of all, the multiscale Gaussian filtering step has been optimized using the convolution of two 1-D filtering. This optimization consists in fixed-loop distribution by means of *#pragma omp parallel for schedule (static, 32)* and internal-loop unrolling, which also permits Gaussian filtering process vectorization. Second, a vectorized reduction process is performed using the root-mean-square error to calculate the spatial homogeneity index for each pixel in the data set. Additionally, for each pixel, we have used a *#pragma omp parallel for* to split the loop iterations between the spawned threads and compute the square for each pixel in the scene.

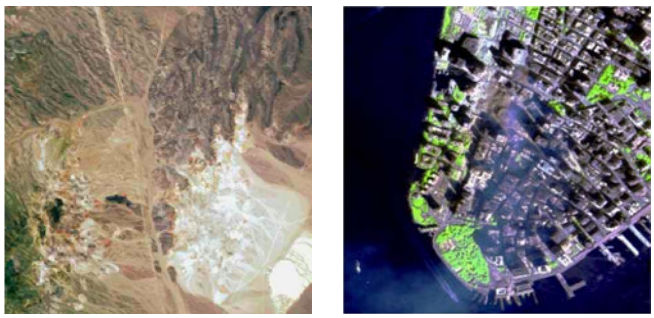
After that, the spectral purity index is computed using the PCA method. For this purpose, vectorized reduction processes that are used to calculate the normalized image and to subtract the mean value to each pixel in the data set are performed. Later, the *cblas_dgemm* routines are used to compute matrix multiplications to obtain the reduced image. Finally, the spectral clustering and fusion of spatial and spectral information operation is optimized declaring a *#pragma omp parallel for schedule (static, 32)* for maximum and minimum reductions.

D. Parallel N-FINDR Algorithm

The most computationally expensive part of this stage (volume calculation step) has been selected for parallelization after a detailed profiling. For this purpose, this part is performed by means of *LAPACKE_dgetrf* routine for lower and upper decomposition. The remainder steps are not parallelized due to their lower processing times, except for its inverse matrix computation where *LAPACKE_dgetri* routine was used. The output of this stage is the endmember matrix **E**.

E. Parallel Least Squares Algorithm

For the last stage in our unmixing chain, several optimized LAPACK functions have been used to speed it up. First, we need to compute a multiplication between the obtained endmember matrix by the N-FINDR stage and its transpose through the *cblas_dgemm* routine. Second, its inverse operation is performed by the *LAPACKE_dgetri* function. The result is then multiplied by the endmember matrix using *cblas_dgemm*. Finally, the result of the previous step is now multiplied by the hyperspectral image **Y** through the *cblas_dgemm* invocation. As a final result, we obtain a set of fractional abundances for each endmember, **A**.



(a) (b)

Fig. 2. (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada. (b) False color composition of an AVIRIS hyperspectral image collected by NASA's Jet Propulsion Laboratory over lower Manhattan on September 16, 2001.

IV. Experimental Results

A. Data Set Description

The experiments are carried out using two popular hyperspectral images collected by NASA's AVIRIS sensor. The first data set is named AVIRIS Cuprite [see Fig. 2(a)], a scene collected in the summer of 1997 and available online in reflectance units after atmospheric correction.[1] The subset used in our experiments corresponds to a 350×350 pixels, which comprises 188 spectral bands in the range from 400 to 2500 nm and a total size of around 50 MB. The site is well understood mineralogically and has several exposed minerals of interest.

The second hyperspectral data set considered in experiments is the well-known AVIRIS World Trade Center (WTC) scene. This scene has been collected in New York City on September 16, 2001, just five days after the terrorist attacks that collapsed the two main towers and other buildings in the WTC complex. The data set consists of 614×512 pixels, which comprises 224 spectral bands and a total size of (approximately) 140 MB. The spatial resolution is 1.7 m/pixel. Fig. 2(b) shows a false color composite of the data set selected for experiments using the 1682-, 1107-, and 655-nm channels displayed as red, green, and blue, respectively. Vegetated areas appear green in Fig. 2(b), while burned areas appear dark gray.

Even though Cuprite scene is commonly used to validate the accuracy of unmixing algorithms and WTC scene for target detection purposes, the main feature that is sought to exploit is the amount of information contained in the images in order to accomplish performance experiments over different architectures.

B. Working Environment

The considered full hyperspectral unmixing chain has been tested on two multicore heterogeneous systems.

1) *Architecture 1:* Intel Core i7-6700K processor with four cores running at 4 GHz using 16 GB of RAM (denoted hereinafter as MC1). This architecture uses an MSI NVIDIA GeForce GTX 1080 GPU with 2560 CUDA

[1] [Online]. Available: http://aviris.jpl.nasa.gov.

TABLE I

SET OF GEEKBENCH BENCHMARKS ENGINEERED TO MEASURE PROCESSOR AND MEMORY PERFORMANCE FOR THE TWO MULTICORES SYSTEMS (MC1 AND MC2) USED IN OUR EXPERIMENTS

Benchmark	Description	MC1	MC2
Canny (Mpixels/sec)	single-core	64.2	36.3
	multi-core	232.5	668.6
SGEMM (Gflops)	single-core	114.9	53.0
	multi-core	171.3	1360.0
SFFT (Gflops)	single-core	13.8	7.18
	multi-core	58.8	175.0
Gaussian Blur (Mpixels/sec)	single-core	86.8	44.5
	multi-core	399.1	1220.0
Memory Copy (GB/sec)	single-core	10.1	3.16
	multi-core	10.5	17.2
Memory Latency (Moperations/sec)	single-core	8.28	5.93
	multi-core	8.20	6.51
Memory Bandwidth (GB/sec)	single-core	13.5	10.7
	multi-core	14.0	42.8

cores at 1.823 GHz and 8 GB of dedicated memory (denoted hereinafter as GPU1).

2) *Architecture 2:* The 2×Intel Xeon E5-2695v3 processors with 14 cores each, running at 2.30 GHz, and 64 GB of DDR3 RAM memory (denoted hereinafter as MC2). An NVIDIA GeForce GTX 1080 GPU with 2560 CUDA cores operating at 1.772 GHz and dedicated memory of 8 GB (denoted hereinafter as GPU2).

For illustrative purposes, Table I provides an indication of the processor and memory performance for both multicore processors used in our experiments. These measures have been obtained using Geekbench[2] (version 4).

C. Analysis of Parallel Performance

Before describing our parallel performance results, it is important to emphasize that the result of the proposed unmixing chain can be evaluated in terms of the quality of the reconstruction of the original data set using the extracted endmembers and the estimated fractional abundances. However, due to the random component resulting from the k-means and N-FINDR algorithms, such comparison would be highly dependent of the random seeds on both methods. Since the considered unmixing chain is well established, we focus on the analysis of the parallel performance of the proposed implementation.

A comparison using our single-core and multicore versions, and a previous GPU implementation [17], is shown in Table II. The multicore version has been compiled using the Intel C++ Compiler 17.0 and OpenMP 4.0 with $-O3$, $-restrict$, and $-qopenmp$ flags. On the other hand, the GPU version has been compiled using NVIDIA CUDA version 8.0 with $-O3$ flag and using BLAS/CUBLAS libraries.

In Table II, the results obtained for Cuprite and WTC scenes reveal a significant speedup for the MC2 architecture, achieving real time in all considered platforms. However, the multicore version does not achieve a better performance when the image size is increased for the MC1 architecture.

[2]http://geekbench.com.

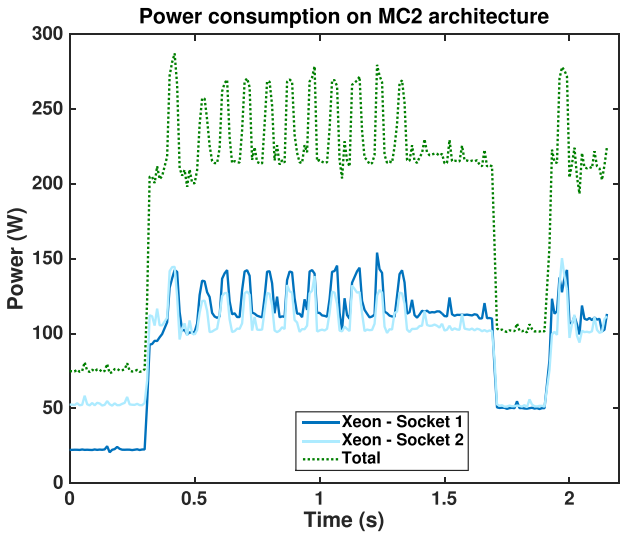


Fig. 3. Power consumption of the proposed full unmixing chain executed on the MC2 architecture considering the WTC data set.

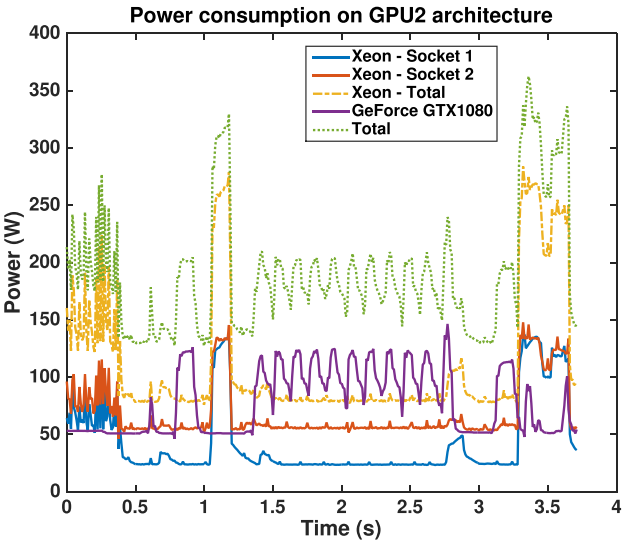


Fig. 4. Power consumption of the proposed full unmixing chain executed on the GPU2 architecture considering the WTC data set.

Between them, there is a slight performance improvement in the multicore approach, mostly involving the first two steps of the unmixing chain (VD and k-means). On the other hand, the rest of the stages exhibit similar performance. When the image size and the number of endmembers are increased (as it is the case in the WTC scene), real-time results are achieved, but the speedup does not keep the same values obtained with the other image. The GPU version shows a better performance in the preprocessing and endmember extraction steps, compensated in the clustering stage by the multicore approach. Moreover, we have found FPGA implementations for the endmember estimation [18], the endmember extraction [19], and the spectral clustering steps [20]. However, it should be noted that we are not aware of a full implementation of a complete unmixing chain in the literature and separately, the performance obtained for our implementations is higher.

TABLE II

MEAN EXECUTION TIMES AND SPEEDUPS (IN THE PARENTHESES) FOR THE PARALLEL VERSIONS OF THE PROPOSED CHAIN EXECUTED ON TWO DIFFERENT PLATFORMS (MULTICORE AND GPU) AFTER 10 MONTE CARLO RUNS

AVIRIS CUPRITE						
	VD	**k-means**	**SSPP**	**N-FINDR**	**LSU**	**Total**
Parallel GPU1	0.395	0.398	0.169	0.003	0.061	1.026
Parallel GPU2	0.659	0.377	0.244	0.034	0.060	1.374
Parallel MC1 [4 threads]	0.161 (2.45)	0.336 (1.18)	0.288 (0.59)	0.002 (1.50)	0.022 (2.77)	0.809 (1.27)
Parallel MC2 [26 threads]	0.146 (4.51)	0.204 (1.85)	0.248 (0.98)	0.004 (8.50)	0.007 (8.57)	0.609 (2.25)
AVIRIS WTC						
	VD	**k-means**	**SSPP**	**N-FINDR**	**LSU**	**Total**
Parallel GPU1	0.651	1.598	0.465	0.009	0.189	2.912
Parallel GPU2	1.017	1.691	0.680	0.085	0.171	3.644
Parallel MC1 [4 threads]	0.488 (1.33)	1.786 (0.89)	0.995 (0.56)	0.007 (1.29)	0.080 (2.36)	3.356 (0.87)
Parallel MC2 [26 threads]	0.426 (2.39)	0.949 (1.78)	0.723 (0.94)	0.014 (6.07)	0.020 (8.55)	2.132 (1.71)

To conclude this section, we have evaluated the power consumption considering both MC2 and GPU2 architectures using the software solution PowerMeter daemon (pmlib) [21], which gathers power results periodically from the tools provided by manufacturers (namely, running average power limit for the Intel Xeon CPU and NVIDIA Management Library for the NVIDIA GPU).

From Figs. 3 and 4, a few observations can be made about the power dissipation. First, if we analyze the plots, the best platform is the MC2 architecture, which dissipates 193 W on average (411 J) and 287 W at most. Compare this, for example, with the 191 W (696 J) on average for the GPU2 architecture. Second, the best tradeoff solution between performance measured in Mpixel/s and energy consumption is achieved by the MC2: 0.00034 Mpixel/s/J versus 0.00012 for the GPU2 architecture.

V. CONCLUSION

In this letter, we have presented a new implementation of a full unmixing chain for multicore platforms. The obtained results indicate that it is possible to achieve significant speedups and even a higher performance than those achieved for other parallel implementations specifically developed for CPU/GPU architectures. The parallel approaches achieve real-time performance in all cases studied. Future work will focus on the development of hybrid multicore/GPU implementations exploiting the advantages of both architectures, and studying other hyperspectral analysis algorithms that could be significantly improved in terms of computational performance.

REFERENCES

[1] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for earth remote sensing," *Science*, vol. 228, no. 4704, pp. 1147–1153, 1985.

[2] A. F. H. Goetz and B. Kindel, "Comparison of unmixing results derived from AVIRIS, high and low resolution, and Hydice images at Cuprite, NV," in *Proc. IX NASA/JPL Airborne Earth Sci. Workshop*, Pasadena, CA, USA, 1999, pp. 1–10.

[3] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, no. 1, pp. 110–122, Sep. 2009.

[4] M. E. Schaepman, S. L. Ustin, A. J. Plaza, T. H. Painter, J. Verrelst, and S. Liang, "Earth system science related imaging spectroscopy—An assessment," *Remote Sens. Environ.*, vol. 113, pp. 123–137, Sep. 2009.

[5] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York, NY, USA: Kluwer, 2003.

[6] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 44–57, Jan. 2002.

[7] R. Heylen, M. Parente, and P. Gader, "A review of nonlinear hyperspectral unmixing methods," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 1844–1868, Jun. 2014.

[8] C. C. Borel and S. A. W. Gerstl, "Nonlinear spectral mixing models for vegetative and soil surfaces," *Remote Sens. Environ.*, vol. 47, no. 3, pp. 403–416, 1994.

[9] J. M. Bioucas-Dias *et al.*, "Hyperspectral unmixing overview: 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.

[10] A. Zare and P. Gader, "Sparsity promoting iterated constrained endmember detection in hyperspectral imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 4, no. 3, pp. 446–450, Jul. 2007.

[11] A. Zare and P. Gader, "Hyperspectral band selection and endmember detection using sparsity promoting priors," *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 2, pp. 256–260, Apr. 2008.

[12] C.-I. Chang and Q. Du, "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.

[13] G. Martin and A. Plaza, "Spatial-spectral preprocessing prior to endmember identification and unmixing of remotely sensed hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 380–395, Apr. 2012.

[14] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[15] M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data," in *Proc. SPIE*, Oct. 1999, pp. 266–270.

[16] J. A. Richards and X. Jia, *Remote Sensing Digital Image Analysis: An Introduction*. Berlin, Germany: Springer, 2006.

[17] L. I. Jiménez *et al.*, "GPU implementation of spatial–spectral preprocessing for hyperspectral unmixing," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 11, pp. 1671–1675, Nov. 2016.

[18] C. Gonzalez, D. Mozos, S. Lopez, and R. Sarmiento, "A novel FPGA-based architecture for the estimation of the virtual dimensionality in remotely sensed hyperspectral images," *J. Real-Time Image Process.*, pp. 1–12, 2015.

[19] C. Gonzalez, D. Mozos, J. Resano, and A. Plaza, "FPGA implementation of the N-FINDR algorithm for remotely sensed hyperspectral image analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 2, pp. 374–388, Feb. 2012.

[20] D. Lavenier, "FPGA implementation of the k-means clustering algorithm for hyperspectral images," Los Alamos Nat. Lab., Los Alamos, NM, USA, Tech. Rep., 2000.

[21] 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.