Analysis and Optimizations of Global and Local Versions of the RX Algorithm for Anomaly Detection in Hyperspectral Data

José Manuel Molero, Ester M. Garzón, Inmaculada García, and Antonio Plaza, *Senior Member, IEEE*

***Abstract*—Anomaly detection is an important task for hyperspectral data exploitation. A standard approach for anomaly detection in the literature is the method developed by Reed and Xiaoli, also called RX algorithm. A variation of this algorithm consists of applying the same concept to a local sliding window centered around each image pixel. The computational cost is very high for RX algorithm and it strongly increases for its local versions. However, current advances in high performance computing help to reduce the run-time of these algorithms. So, for the standard RX, it is possible to achieve a processing time similar to the data acquisition time and to increase the practical interest for its local versions. In this paper, we discuss several optimizations which exploit different forms of acceleration for these algorithms. First, we explain how the calculation of the correlation matrix and its inverse can be accelerated through optimization techniques based on the properties of these particular matrices and the efficient use of linear algebra libraries. Second, we describe parallel implementations of the RX algorithm, optimized for multicore platforms. These are well-known, inexpensive and widely available high performance computing platforms. The ability to detect anomalies of the global and local versions of RX is explored using a wide set of experiments, using both synthetic and real data, which are used for comparing the optimized versions of the global and local RX algorithms in terms of anomaly detection accuracy and computational efficiency. The synthetic images have been generated under different noise conditions and anomalous features. The two real scenes used in the experiments are a hyperspectral data set collected by NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) system over the World Trade Center (WTC) in New York, five days after the terrorist attacks, and another data set collected by the HYperspectral Digital Image Collection Experiment (HYDICE). Experimental results indicate that the proposed optimizations can significantly improve the performance of the considered algorithms without reducing their anomaly detection accuracy.**

Manuscript received October 02, 2012; revised January 01, 2013; accepted January 04, 2013. Date of publication January 15, 2013; date of current version May 13, 2013. This work was supported by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117 and AYA2011-29334-C02-02), Junta de Andalucia (P10-TIC-6002) and Junta de Extremadura (PRI09A110 and GR10035), in part financed by the European Regional Development Fund (ERDF). The work was also supported by the European Community's Marie Curie Research Training Networks Programme under reference MRTN-CT-2006-035927 (HYPER-I-NET).

J. M. Molero and E. M. Garzón are with the Supercomputing and Algorithms Group, University of Almería, 04120 Almería, Spain (e-mail: jmp384@ual.es, gmartin@ual.es).

I. García is with the Department of Computer Architecture, University of Málaga, Campus de Teatinos, 29071 Málaga, Spain (e-mail: igarciaf@uma.es).

A. Plaza is with the Hyperspectral Computing Laboratory, University of Extremadura, E-10071 Cáceres, Spain (corresponding author, 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.2013.2238609

***Index Terms*—Anomaly detection, hyperspectral processing, local RX, multicore platforms, RX algorithm.**

I. Introduction

HYPERSPECTRAL imaging [1] is concerned with the measurement, analysis, and interpretation of spectra acquired from a given scene (or specific object) at a short, medium or long distance by an airborne or satellite sensor [2]. Hyperspectral imaging instruments such as the NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) [3] are now able to record the visible and near-infrared spectrum (wavelength region from 0.4 to 2.5 micrometers) of the reflected light of an area of 2 to 12 kilometers wide and several kilometers long using 224 spectral bands. The resulting "image cube" is a stack of images in which each pixel (vector) has an associated spectral signature or *fingerprint* that uniquely characterizes the underlying objects [4]. The resulting data volume typically comprises several GBs per flight [5].

Anomaly detection is an important task for hyperspectral data exploitation. An anomaly detector enables one to detect spectral signatures which are spectrally distinct from their surroundings with no *a priori* knowledge. Frequently, anomalies are composed of a set of isolated pixels with anomalous signatures (when compared to the image background) which represents a very small piece of the full image, and they only occur in the image with low probabilities [6], [7]. The RX algorithm, developed by Reed and Xiaoli, is a well-known approach for anomaly detection which has shown success for multispectral and hyperspectral images [4], [8]. The RX algorithm uses the pixel currently being processed as the matched signal and computes the Mahalanobis distance, which has been widely used in hyperspectral imaging applications [9].

Variants of the RX algorithm consist of applying locally the same concept to a sliding window centered around each image pixel [10], [11]. These variations have been widely used in order to detect small anomalies [12]–[15]. However, they are computationally more expensive than the original RX because they involve the computation of a covariance matrix and its inverse for every local window as opposed to the standard RX which performs the same calculation for the whole image. So far, few works in the published literature have reported (near) real-time performance for the RX algorithm or its local version [16]–[19]. Here, by real-time performance we refer to the fact that the processing can be performed without delay as the data are collected

but not necessarily immediately after the data is collected. In the following, we will avoid the term real-time processing on purpose and refer to *processing at the same time as the data are collected*. Nevertheless, current advances in high performance computing (HPC) [20] offer an unprecedented opportunity to strongly reduce the runtimes for the RX detectors to identify anomalies. In particular, modern multicore architectures [12], [21]–[23] represent an inexpensive, widely available and well-known technology in the HPC field. Moreover, a wide set of linear algebra libraries are available to exploit such multi-core systems. These libraries can be used to significantly reduce the time for solving linear algebra-related problems and to solve more efficiently the mathematical computations involved in the algorithm [24], [25].

In related work, several optimizations have been developed for the RX algorithm on different platforms [12], [26]–[28]. However, none of them have combined the integrated use of software optimizations (sequential and parallel) and HPC-related optimizations, resulting in the fact that it was difficult to apply the algorithm at the same time as the data were collected. Further, as compared to the RX, a local version exhibits a very large computational burden that generally prevents its sequential execution on single-core systems. However, the design and development of these optimizations and the parallelization of this kind of algorithms are not simple tasks, and involve a deep knowledge of both the algorithm and the architecture.

In this work our interest is focused on the RX version introduced by Chang [14] in order to adapt this detector to online analysis scenarios by replacing the sample covariance matrix by the sample correlation matrix. Thus, two variations of RX are analyzed and optimized for multicore platforms: (i) the so-called global RX (GRX), which uses the sample correlation matrix for the full hyperspectral image to detect anomalies, and (ii) the local RX (LRX) which consists of applying the same concept of RX to a single local window around each pixel of the hyperspectral image. Our aim is to compare the global and local versions of RX in terms of effectiveness (detection accuracy) and efficiency (computing time). This paper thus analyses optimized versions of GRX and LRX which exploit several forms of acceleration, with the ultimate goal of achieving similar time for processing and data acquisition for GRX and to explore practical interest of LRX.

To reduce the runtime of GRX and LRX, first we have developed several software optimization techniques in the sequential code of both algorithm variants. These optimizations are based on: (i) taking advantage of the properties of the correlation matrices involved in both algorithms (such as the symmetry and a specific recurrence relation for LRX), thus, the computation is strongly reduced; and (ii) exploitation of optimized linear algebra libraries (such as BLAS [29] and MKL [30]) in order to accelerate the calculation of the matrix and to avoid the explicit computation of its inverse which is crucial in both algorithms. Second, we have develop parallel implementations optimized for multi-core platforms, which represent a well known, inexpensive and widely available HPC technology. In this way, the exploitation of the particular properties of the correlation matrices involved in both algorithms, the integration of optimized libraries and the parallelization using HPC techniques can assist in the development of more efficient implementations of these algorithms for anomaly detection.

The remainder of the paper is organized as follows. Section II briefly describes the GRX algorithm and its local variant, LRX. Section III explains the software and multithreaded optimizations carried out for the GRX and LRX algorithms. Section IV describes the (synthetic and real) hyperspectral data sets used in our experimental evaluation. Section V presents a detailed quantitative and comparative evaluation of GRX and LRX in terms of their anomaly detection accuracy and parallel performance. Specifically, their capacity to detect anomalies is evaluated using receiver operating characteristics (ROC) [31], and their parallel performance is analyzed by evaluating the speedup or acceleration factor obtained with regards to optimized versions of the serial implementations of both algorithms.

II. Anomaly Detection Using Global RX (GRX) and Local RX (LRX) Algorithms

A. GRX Algorithm

The GRX algorithm has been widely used in signal and image processing [8]. A variant of this algorithm was proposed by Chang [14], which consists of replacing the covariance matrix by the sample correlation matrix $\mathbf{R}$ and removing the subtraction of the mean vector to each B-dimensional hyperspectral pixel vector $\mathbf{x} = [x^{(0)}, x^{(1)}, \cdots, x^{(B)}]$. This modification allows for online exploitation of the RX algorithm without penalization in its ability to detect anomalies. The sample correlation matrix-based GRX filter was introduced by Chang in [4] and given by:

$$\delta^{GRX}(\mathbf{x}) = \mathbf{x}^T \mathbf{R}^{-1} \mathbf{x}, \tag{1}$$

later we will explain the implementation details which make GRX an optimized variant on multicore platforms.

It is important to note that the anomaly detection results generated by the GRX algorithm can be visualized as a grayscale image in which, the higher the probability of detecting an anomaly, the higher the value of the pixel. Anomalies can be categorized in terms of the value returned by GRX, so that the pixel with the highest value of $\delta^{GRX}(\mathbf{x})$ can be considered the first anomaly, and so on.

B. LRX Algorithm

The classic RX algorithm can be considered as a global anomaly detector because the correlation matrix is computed using all the pixels of the image and the background is defined with reference to the full image. The LRX algorithm [32] can be considered as a local anomaly detector because each pixel of the image has its own correlation matrix, which is computed just considering a small set of neighboring pixels around the pixel under test. This implementation uses the concept of a sliding local window for every pixel in the image. For each pixel $\mathbf{x}$, the LRX filter is computed using a square window of size $\kappa \times \kappa$ pixels, centered at pixel $\mathbf{x}$. So, a matrix $\mathbf{R}_{\kappa\times\kappa}(\mathbf{x})$ is calculated for every pixel $\mathbf{x}$ based on its own local window. Consequently, the filter is defined by:

$$\delta_{\kappa}^{LRX}(\mathbf{x}) = \mathbf{x}^T \mathbf{R}_{\kappa\times\kappa}(\mathbf{x})^{-1} \mathbf{x}, \tag{2}$$

It is important to emphasize that, in the local implementation, the correlation matrix is computed using the local window instead of the full image. In other words, for each pixel under test the algorithm applies the RX filter using the local information. This variation considers a local approach estimation, to determine whether the image pixels are anomalous or not. As a result, LRX can be considered as a process over each pixel of the image, using local information provided by the data of the sliding window and applying the RX filter in the local area.

C. Structure of the RX Algorithms

Bearing in mind the previous descriptions, three stages can be identified in both algorithms, GRX and LRX:

- *Phase 1*. In this stage the correlation matrix is evaluated. So, the matrix $\mathbf{R}$ is globally evaluated for the whole image when GRX is applied; or $\mathbf{R}_{\kappa\times\kappa}(\mathbf{x})$, for every pixel of the image, $\mathbf{x}$, if LRX is computed.
- *Phase 2*. The second phase is focused on the computation of the intermediate vector $\mathbf{y}(\mathbf{x})$ for every pixel defined by the expression $\mathbf{y}(\mathbf{x}) = \mathbf{R}^{-1}\mathbf{x}$ for GRX, or $\mathbf{y}(\mathbf{x}) = \mathbf{R}_{\kappa\times\kappa}(\mathbf{x})^{-1}\mathbf{x}$ for LRX.
- *Phase 3*. The output filter $\delta(\mathbf{x}) = \mathbf{x}^T\mathbf{y}$ is the same for both versions of the algorithms.

There are several approaches to compute $\mathbf{y}(\mathbf{x})$. The classic one is based on the computation of the inverse matrix, $\mathbf{R}(\mathbf{x})^{-1}$, followed by the evaluation of the product $\mathbf{y}(\mathbf{x}) = \mathbf{R}^{-1}\mathbf{x}$ for GRX. However, an alternative approach to reduce the high cost of inverse matrices computation consists on solving the linear system $\mathbf{x} = \mathbf{R}\mathbf{y}(\mathbf{x})$ for GRX. The same considerations can be applied to LRX. Consequently, in this work, the implementations of both versions of RX are based on solving the linear systems due to its advantages in terms of performance. Additionally, this approach has numerical advantages because the instabilities caused by ill-conditioned correlation matrices are better managed solving the linear equations.

III. Optimizations of Global RX (GRX) and Local RX (LRX) Algorithms

In order to reduce the runtime of GRX and LRX algorithms, their implementations have been improved and optimized, for both the sequential and the parallel versions. In this work, parallel versions are based on the exploitation of multi-core architectures.

A. Optimizations of Sequential GRX and LRX Algorithms

Our interest has been focused on the reduction of the run-time for the most computationally expensive stages; i.e. the first and second stages described on Section II-C, which compute: (1) the correlation matrices and (2) the vector $\mathbf{y}(\mathbf{x})$.

At the first stage, focusing on the GRX algorithm, the correlation matrix is computed by the expression:

$$R^{(m,s)} = \frac{1}{LS}\sum_{p=1}^{L}\sum_{l=1}^{S} x_{p,l}^{(s)} \cdot x_{p,l}^{(m)}; \quad 1 \le m, s \le B \tag{3}$$

where $R^{(m,s)}$ is the element of row m and column s of the correlation matrix; $x_{p,l}^{(s)}$ denotes the spectral (wavelength) component s of the pixel spatially located by indexes p, l in the image; L, S and B are the number of lines, samples and bands of the image, respectively. A similar expression is used to compute the correlation matrices in the LRX algorithm:

$$R_{\kappa\times\kappa}^{(m,s)}(\mathbf{x}_{i,j}) = \frac{1}{\kappa^2}\sum_{p=i-\frac{\kappa}{2}}^{i+\frac{\kappa}{2}}\sum_{l=j-\frac{\kappa}{2}}^{j+\frac{\kappa}{2}} x_{p,l}^{(s)} \cdot x_{p,l}^{(m)}; \quad 1 \le m, s \le B \tag{4}$$

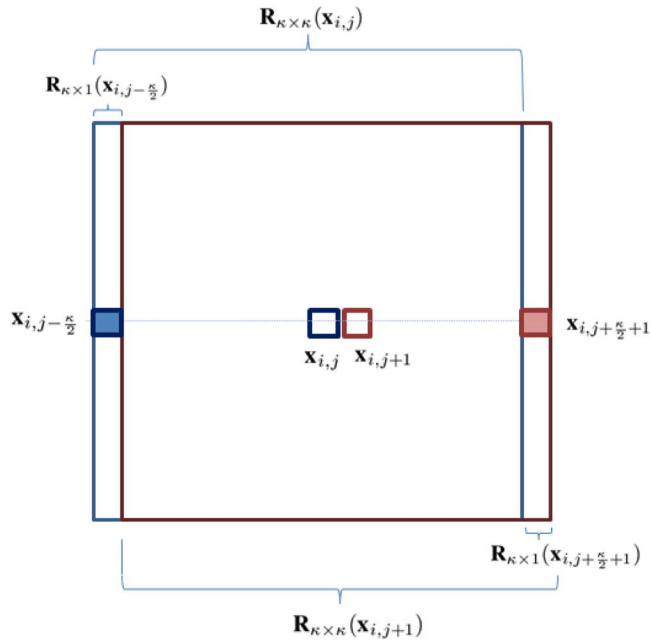


Fig. 1. Computation of the correlation matrix, taking advantage of the computation of the neighboring window.

To optimize the computation of (4) the following characteristics have been considered:

- The symmetry property of the correlation matrix, which can be exploited by both GRX and LRX algorithms, allows us to compute only half of the correlation matrix and then duplicate the result.
- For the LRX algorithm it is also possible to take advantage of the recurrence relation among the correlation matrices associated to neighboring pixels. This means that is possible to reuse the common computation of contiguous pixels. So, the computational cost to compute the correlation matrix associated to pixel $i, j+1$ can be strongly reduced because most of the computations have already been performed for the correlation matrix associated to pixels i, j (see Fig. 1).

In order to analyze this recurrence relation, let $\mathbf{x}_{i,j}$ and $\mathbf{x}_{i,j+1}$ denote two contiguous pixels located in the line i and let $\mathbf{R}_{\kappa\times1}(\mathbf{x}_{i,l})$ be the correlation matrix evaluated in the window centered in the pixel $\mathbf{x}_{i,l}$ of dimension $\kappa \times 1$, i.e. column l of the $\kappa \times \kappa$ window centered at $\mathbf{x}_{i,j}$; then (4) can be rewritten as:

$$\mathbf{R}_{\kappa\times\kappa}(\mathbf{x}_{i,j}) = \frac{1}{\kappa}\sum_{l=j-\frac{\kappa}{2}}^{j+\frac{\kappa}{2}} \mathbf{R}_{\kappa\times1}(\mathbf{x}_{i,l}) \tag{5}$$

and the correlation matrix $\mathbf{R}_{\kappa\times\kappa}(\mathbf{x}_{i,j+1})$ can be expressed by the following recurrent expression:

$$\mathbf{R}_{\kappa\times\kappa}(\mathbf{x}_{i,j+1}) = \mathbf{R}_{\kappa\times\kappa}(\mathbf{x}_{i,j}) + \frac{1}{\kappa}\left\{\mathbf{R}_{\kappa\times 1}\left(\mathbf{x}_{i,j+\frac{\kappa}{2}+1}\right) - \mathbf{R}_{\kappa\times 1}\left(\mathbf{x}_{i,j-\frac{\kappa}{2}}\right)\right\} \quad (6)$$

Fig. 1 illustrates the computation of the correlation matrix taking advantage of the computation of the neighboring window as indicated in (6). The computational cost to calculate the correlation matrices is $O((B^2\kappa^2\cdot S\cdot L)/2)$ and the cost to calculate $\mathbf{R}_{\kappa\times 1}(\mathbf{x})$ is $O(B^2\kappa/2)$. This means that our approach based on the recurrence relation provides an improvement in performance of κ (the dimension of the sliding window).

The other stage with a high computational cost is the evaluation of $\mathbf{y}(\mathbf{x})$. As mentioned above, the standard approach for this stage is based on the computation of the inverse of the correlation matrix and then the product matrix vector [17], [26]. However, the best alternative to determine $\mathbf{y}(\mathbf{x})$ in terms of both stability and performance is to consider the solution of the linear system: $x = Ry(\mathbf{x})$ for GRX, or $x = R_{\kappa\times\kappa}(\mathbf{x})\mathbf{y}(\mathbf{x})$ for LRX (for more details, Section 3.4.11 of [33] and Chapter 2 of [34]). Direct methods for solving systems of equations are based on the factorization of the matrix which defines the system. Several factorization methods systems have been described in the literature [33]. We have selected the QR factorization due to its numerical stability when the systems are solved as least square problems (Section 5.3.3 of [33]).

The library LAPACK (Linear Algebra PACKage) supplies implementations of different methods for solving systems of linear equations. LAPACK has been designed to run efficiently on a wide range of modern high-performance computers. Specifically, the routine `dgels` can solve systems of equations defined as least square problems by the QR decomposition [29], [35]. The routine `dgels`, included in the library Intel MKL,[1] exploits efficiently the resources of INTEL modern multicore processors. Our introspection is that an efficient solution of linear systems can be provided by the `dgels` routine which is optimized in terms of performance and numerical stability.

B. Parallel GRX and LRX Algorithms

One goal of this work is to accelerate GRX and LRX by the exploitation of multicore architectures. Despite the similarity of both algorithms, their parallel implementations are different due to the characteristics of each version, as described in Section II. This subsection describes the key aspects of the parallel implementation of both algorithms. The aim is to parallelize the most expensive stages, that is, the computation of the correlation matrices and the solution of linear systems. Hereinafter Th denotes the number of threads mapped on the multicore architecture.

1) Multicore GRX: The GRX algorithm consists of three consecutive tasks over the whole image:

- *Phase 1*. The computation of the correlation matrix, $\mathbf{R}$, whose computational cost is $O((B^2\cdot L\cdot S)/2)$ since it is evaluated on the whole image and its symmetry is considered to optimize the computation. This phase is parallelized using a cyclic distribution of the Th threads among the number of bands of the image, so every thread independently computes a subset of elements of R.
- *Phase 2*. The computation of $\mathbf{y}(\mathbf{x})$ for every pixel $\mathbf{x}$ of the image. The goal of this phase is to compute the matrix $\mathbf{Y}$ whose columns are the vectors $\mathbf{y}(\mathbf{x})$ for all pixels, $\mathbf{x}$, of the hyperspectral image. According to the previous consideration for avoiding the computation of $\mathbf{R}^{-1}$, the matrix $\mathbf{Y}$ can be obtained by the solution of the matricial equation $\mathbf{X} = \mathbf{RY}$, where the image pixels define the columns of the matrix $\mathbf{X}$. So, the matrix $\mathbf{Y}$ can be obtained as the output of the multithreaded `dgels` routine of Intel MKL Library. This kind of libraries are able to exploit different parallelism levels of the multicore architecture.
- *Phase 3*. The output filter $\delta^{GRX}(\mathbf{x}) = \mathbf{x}^T\mathbf{y}(\mathbf{x})$. In this phase a cyclic distribution of threads is used in one spatial dimension because our experimental evaluation has shown that this strategy achieves slightly better performance than block distribution.

2) Multicore LRX: LRX algorithm could be considered as a set of independent procedures or tasks, where each task computes the local window associated to the pixel under test. The recurrence relation of the correlation matrices strongly reduces the sequential runtime. However, it increases the data dependence of the local tasks since the computation of the correlation matrix of a pixel involves the computation of the matrix associated to the previous (neighboring) pixel. The recurrence relation can be applied to the two spatial dimensions and, consequently, LRX can be decomposed in tasks with front-wave dependence [36], [37]. However, this scheme cannot take advantage of the computation by lines of the image, which is very relevant to achieve online processing.

Therefore, our parallel local version considers the recurrence relation among the matrices associated to every line of the image in the computation for every thread, so the filter for every line is computed by every thread. In this way, the parallel LRX computes all the filters related to Th lines of the image. Moreover, this approach maintains the local features of the sequential version of the LRX algorithm, that is, it can compute the RX filter without requiring the full image. Fig. 2 shows a generic flow path for the multithreaded version of the LRX algorithm. On the left side of Fig. 2, we represent the hyperspectral image. On the right side of Fig. 2, the output filters have been represented. The LRX algorithm can be simply understood by connecting the leftmost and rightmost parts of Fig. 2.

To conclude this section, we emphasize that our implementation of LRX considers that only one processing window is used to define the spatial context around each pixel. In the literature, several anomaly detection solutions have been described that use more than one processing window per pixel [17]. In this regard, it is important to note that our proposed parallel implementation could be easily adapted to this framework with several processing windows since the parallel implementation would be identical but with different window sizes. Specifically, the concept of "dual rectangular window" in hyperspectral image processing has been widely used to separate a local area in two regions using a second small window (inner window region) inside the large window (outer window region). Again, we reiterate that our proposed formulation can be easily adapted to such dual processing window frameworks.

[1]http://software.intel.com/en-us/articles/intel-mkl/.

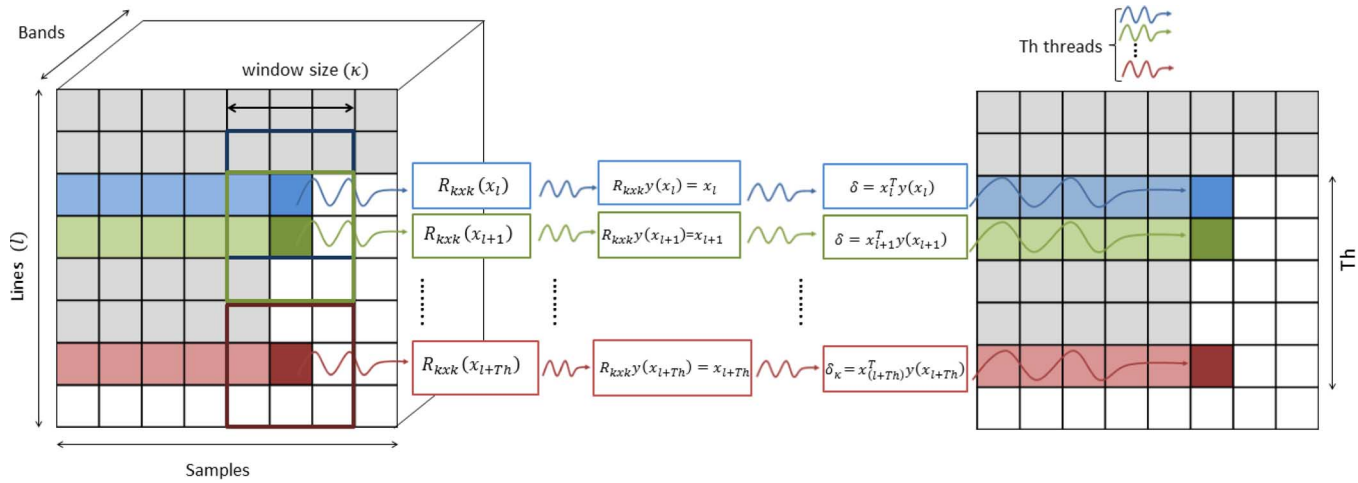


Fig. 2. Flow path of the multithreaded version of the LRX algorithm.

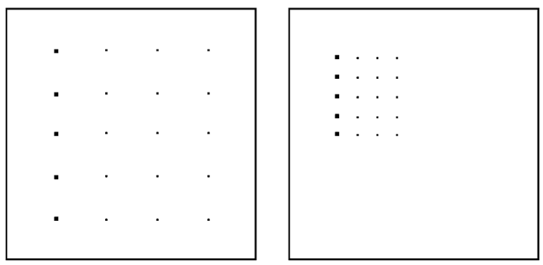


Fig. 3. Two different panel locations adopted in the simulation of SimHyd simulated scenes.

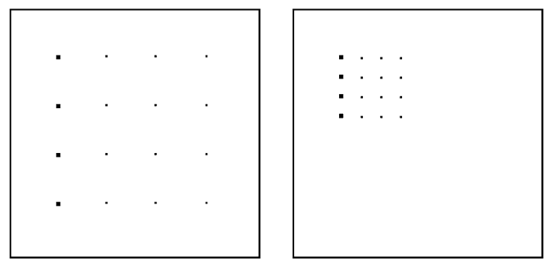


Fig. 4. Two different panel locations adopted in the simulation of SimUsgs simulated scenes.

IV. Hyperspectral Data Sets

Several different hyperspectral data sets have been used in this work to evaluate the accuracy (effectiveness) and computational performance (efficiency) of the proposed optimized algorithms in the task of detecting anomalies. For this purpose, we have used a collection of synthetic and real hyperspectral data sets comprising different characteristics (such as their spectral resolution, the spatial location in the image of the anomalous signatures, the hyperspectral instrument used for data collection, the image size or the size of the anomalies to be detected). Their characteristics are described next.

A. Synthetic Hyperspectral Data

Two simulated hyperspectral data sets have been used in our experiments:

- *SimHyd*: These simulated data are based on a real HYDICE scene that will be fully described in the following section. In order to generate the simulated data, we have extracted different spectral signatures from the HYDICE scene. As described in Section IV-B, the spectral signatures in the HYDICE scene are composed of 169 spectral bands. The background of the simulated image has been generated from random mixtures of spectral signatures of vegetation and soil. In order to generate the anomalies, a set of panels has been placed in a spatial grid, using five different spectral signatures extracted from the real image. The anomalies has been distributed in five rows, one for each different spectral signature extracted.
- *SimUsgs*: These simulated data are based on a set of spectral signatures provided by the USGS,[2] which have been convolved to the wavelengths available in the AVIRIS hyperspectral sensor in order to obtain spectral signatures composed by 224 spectral bands. The background spectra have been generated from random mixtures of the spectral signatures of two different minerals, such as *alunite* and *kaolinite*. In order to generate the anomalies, a set of spectral signatures that represent different minerals have been spatially distributed in the image in the form of a regular spatial grid. The signatures used for simulation purposes are *buddingtonite* (first row), *calcite* (second row), *muscovite* (third row), and a mixture of *buddingtonite* and *calcite* (fourth row).

In both cases, the size in pixels is 128 × 128 but different panel locations have been used in the simulation of each synthetic scene, as illustrated in Fig. 3 and Fig. 4. In the SimHyd01 and SimUsgs01 scenes [see Fig. 3(a) and Fig. 4(a)] the panels occupy the entire image equidistantly. In the SimHyd02 and SimUsgs02 scenes [see Fig. 3(b) and Fig. 4(b)] the panels are grouped in a small area. Also, anomalies with different sizes have been distributed in four columns of the grid: the first column shows panels of 4 pixels in a square (i.e., 2 ×

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

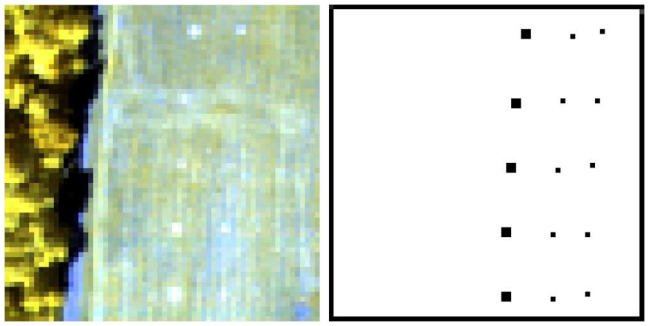


Fig. 5. False color representation of the HYDICE hyperspectral scene and its associated ground-truth information.

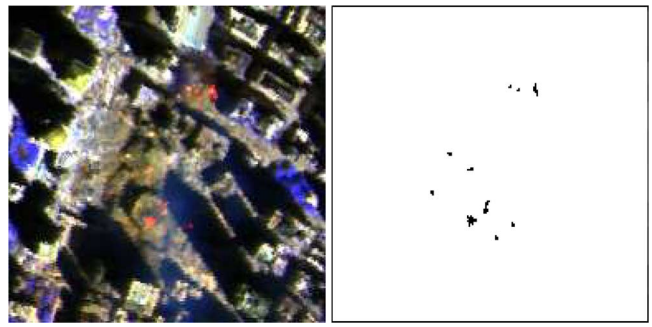


Fig. 6. False color representation of the WTC hyperspectral scene and its associated ground-truth information.

2-pixel panels); the second column is made up of single pixel panels; the third column is made up of sub-pixels with 50% abundance of the panel and 50% abundance of the background; and finally the fourth column is made up of sub-pixels with 25% abundance of the panel and 75% abundance of the background. For illustrative purposes, Table I shows the properties of all the synthetic images used in our experiments. Finally, Gaussian noise has been added (in different proportions) to all the synthetic scenes, using the definition of signal-to-noise ratio (SNR) in [38]. Specifically, three different levels of SNR have been considered: 10:1, 30:1 and 50:1.

B. Real Hyperspectral Data

Two real hyperspectral data sets, collected by different instruments, have been used in experiments:

- *HYDICE*: The first real hyperspectral data set used was collected in the framework of the HYperspectral Digital Image Collection Experiment (HYDICE). The scene is extensively described and used in [4]. It is an image scene with size of 64 $\times$ 64 pixels and 15 panels in the scene. A ground-truth map is available for the scene, indicating the spatial location of the panels (real targets). The sizes of the panels in the first, second, and third columns have a size (in meters) of: 3 $\times$ 3, 2 $\times$ 2 and 1 $\times$ 1, respectively. This image was acquired by 210 spectral bands with a spectral coverage from 0.4 to 2.5 microns. Low signal/high noise bands: 1–3 and 202–210; and water vapor absorption bands: 101–112 and 137–153, were removed prior to experiments so a total of 169 bands were finally used. The spatial resolution of the scene is 1.56 meters (i.e., the last column of targets are sub-pixel in size) and the spectral resolution is 10 nanometers. Fig. 5 depicts the HYDICE scene and its associated ground-truth.
- *WTC*: The second real hyperspectral data set used was collected by the AVIRIS instrument, flown by NASA's Jet Propulsion Laboratory, over the World Trade Center (WTC) area in New York City on September 16, 2001. The size of the full scene is 614 $\times$ 512 pixels and 224 spectral bands, for a total size of about 140 MB (this is the standard size of the data chunks collected by the AVIRIS instrument before saving the data to disk in the onboard data collection). A subset of this scene has also been selected for experiments, centered at the region of interest (hot spots zone at the WTC), and consisting of 192 $\times$ 192 pixels and 224 spectral bands. The leftmost part of Fig. 6 shows a false color composite of the data set selected for experiments. Extensive reference information, collected by U.S. Geological Survey (USGS), is available for the WTC scene.[3] In this work, we use a U.S. Geological Survey thermal map[4] which shows the locations of the thermal hot spots (which can be seen as anomalies) at the WTC area. The anomalies (fires) are displayed as bright red, orange and yellow spots in Fig. 6, which is centered at the region where the towers collapsed. The ground-truth map displayed in the rightmost part of Fig. 6 will be used in this work to validate the anomaly detection accuracy of the proposed parallel implementations.

V. Experimental Results

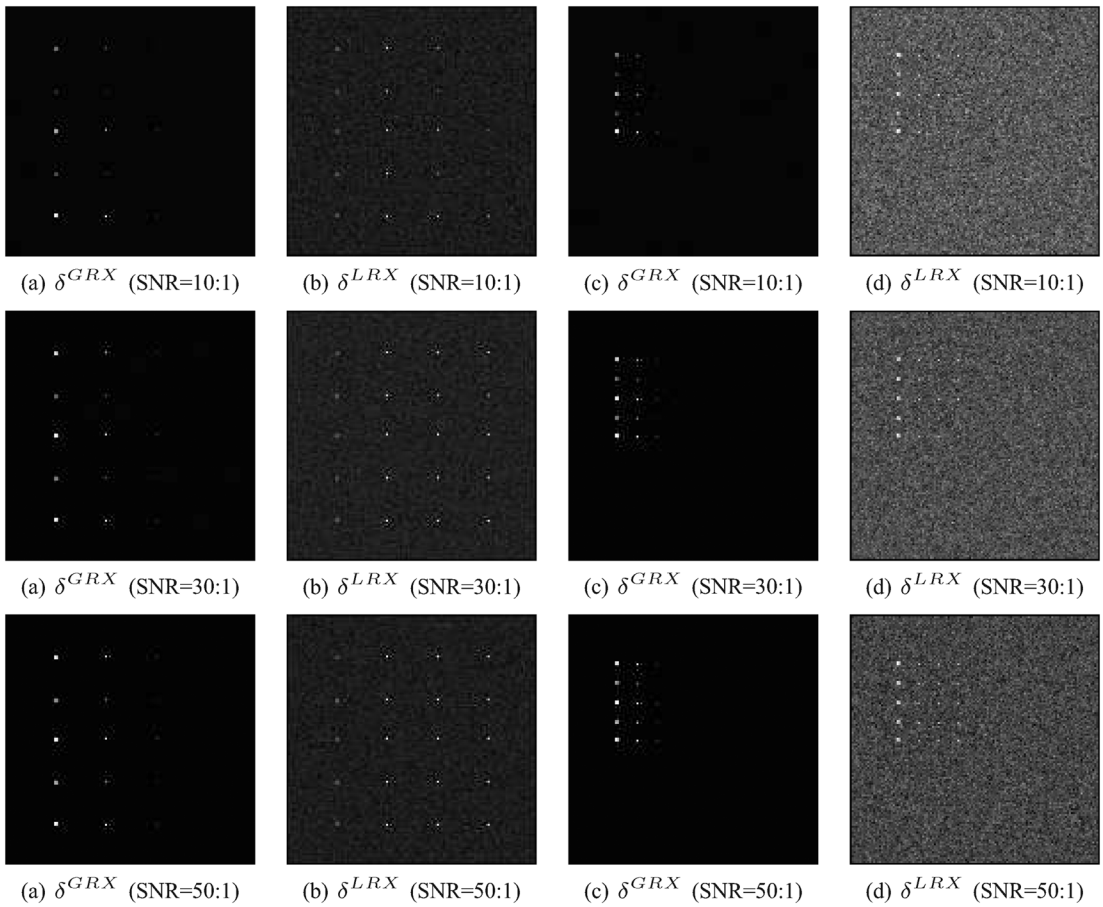
A. Analysis of Anomaly Detection Accuracy

In this subsection we evaluate the anomaly detection accuracy of the proposed parallel implementations of GRX and LRX using the simulated and real hyperspectral data sets described in Section IV. First of all, we emphasize that our parallel versions provided exactly the same results as the sequential ones. For the GRX there are no input parameters, while for the LRX there is only one input parameter, κ, which is the size (in pixels) of the local window used to define the spatial neighborhood around each pixel. In our experiments, we have optimized this parameter empirically after testing several values of κ in the range [3,30]. In general terms, we have observed that $\kappa = 23$ provides a compromise between detection accuracy and computational burden (hereinafter, $\kappa = 23$ is used for all the experiments). At this point, it is important to emphasize that very small window sizes may cause problems in the execution of the LRX due to matrix ill-rank problems, thus causing numerical instability. For this reason, the size of the local processing window should be sufficiently large when compared to the size of the anomaly and cannot be too small. On the other hand, using very small windows is not effective for our parallel implementation since we have empirically observed that the best results (in terms of detection and speedup) are obtained when the window size is sufficiently large.

Fig. 7 shows the output of the $\delta^{GRX}(\mathbf{x})$ and $\delta^{LRX}(\mathbf{x})$ filters for the simulated scenes SimHyd01 and SimHyd02 using different SNR values. As shown in Fig. 7, it is evident that LRX

[3]http://speclab.cr.usgs.gov/wtc.

[4]http://pubs.usgs.gov/of/2001/ofr-01-0429/hotspot.key.tgif.gif.



(a) δ^{GRX} (SNR=10:1) (b) δ^{LRX} (SNR=10:1) (c) δ^{GRX} (SNR=10:1) (d) δ^{LRX} (SNR=10:1)

(a) δ^{GRX} (SNR=30:1) (b) δ^{LRX} (SNR=30:1) (c) δ^{GRX} (SNR=30:1) (d) δ^{LRX} (SNR=30:1)

(a) δ^{GRX} (SNR=50:1) (b) δ^{LRX} (SNR=50:1) (c) δ^{GRX} (SNR=50:1) (d) δ^{LRX} (SNR=50:1)

Fig. 7. Processing output of δ^{GRX} and δ^{LRX} for the simulated scenes SimHyd01 (columns 1–2) and SimHyd02 (columns 3–4) using different SNR values.

TABLE I
PROPERTIES OF ALL THE SYNTHETIC IMAGES USED IN OUR EXPERIMENTS

Scene	**Spatial location**	**Spectral composition**	**Samples** (S)	**Lines** (L)	**Spectral bands** (B)
SimHyd01	5x4 matrix (equally spaced)	Based on HYDICE signatures	128	128	169
SimHyd02	5x4 matrix (placed together)	Based on HYDICE signatures	128	128	169
SimUsgs01	4x4 matrix (equally spaced)	Based on USGS signatures	128	128	224
SimUsgs02	4x4 matrix (placed together)	Based on USGS signatures	128	128	224

can detect all the anomalies for the SimHyd01 scene, even those appearing at sub-pixel levels (third and fourth panel columns) for any noise level, while GRX was not able to detect the fourth column in any case. Thus, LRX exhibits slightly better anomaly detection accuracy. With regards to the results obtained for the SimHyd02 scene, we can observe in Fig. 7 that the results in this case vary with the spatial location of the anomalies in the image. Specifically, the local version exhibits few difficulties in detecting the smallest anomalies but, in any case, the results are still better than the results of GRX. A very similar behavior of GRX and LRX can be observed in Fig. 8 for the simulated scenes SimUsgs01 and SimUsgs02. Here, the LRX can detect up to the third column for images with low SNR values, which is a better result than the one exhibited by the GRX that can only detect up to the panels in the second column.

Fig. 9 shows the output of the $\delta^{GRX}(\mathbf{x})$ and $\delta^{LRX}(\mathbf{x})$ filters (together with a 3-D representation of the output indicating the difference between panels and background) for the real HYDICE scene. Fig. 10 provides similar results for the real WTC scene. As can be seen in Figs. 9 and 10, LRX provides better performance in the detection of small targets, such as the sub-pixel panels in the rightmost panel column of the HYDICE scene (see Fig. 9). On the other hand, GRX provides good effectiveness in the detection of anomalies with a larger size.

Overall, the accuracy of the two detectors is quite similar, as indicated by the receiver operating characteristics (ROC) curves constructed for the HYDICE experiment (see leftmost side of Fig. 11) and for the WTC experiment (see rightmost side of Fig. 11). ROC curves have not been reported for the synthetic images, due to the observed very low rate of false alarms. The evaluation of GRX and LRX for the full set of test images (synthetic and real) has been carried out by the well known area under the ROC curve (AUC). In this way, Fig. 12 provides a quantitative analysis of anomaly detection accuracy

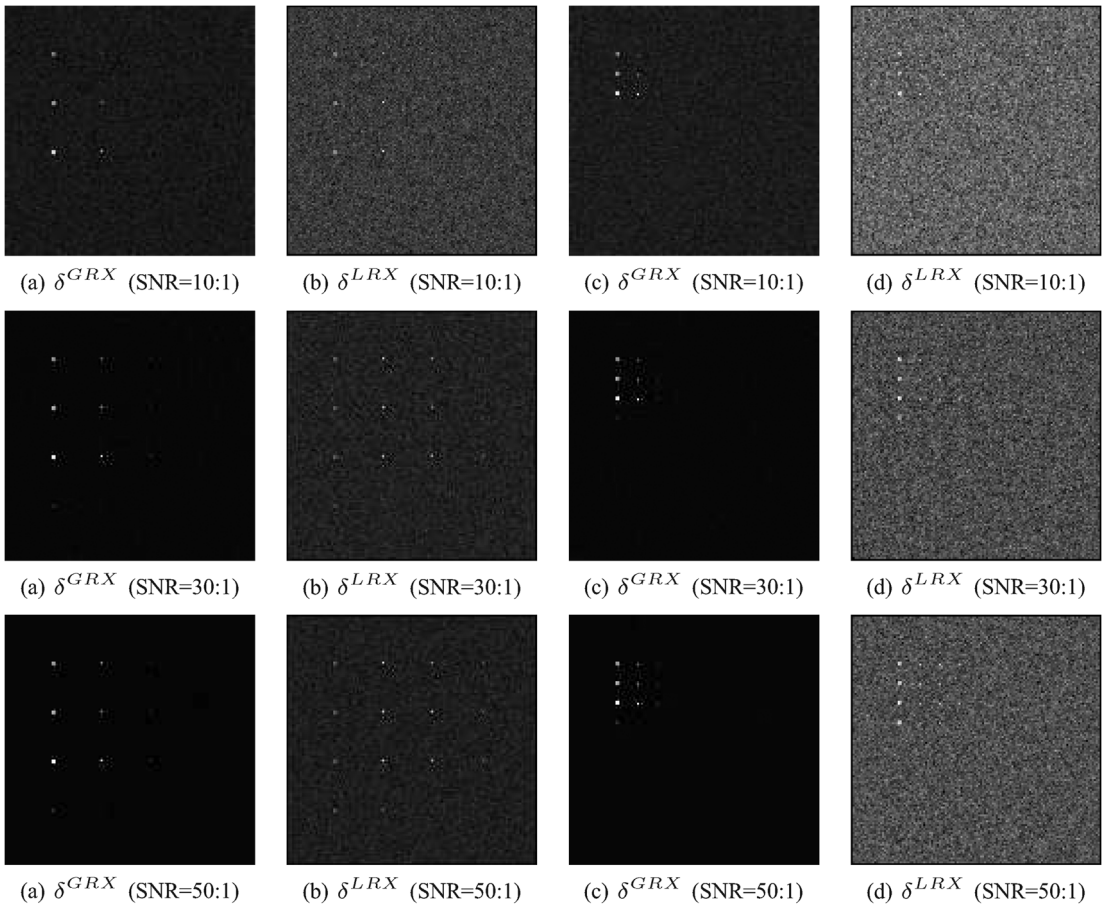


Fig. 8. Processing output of δ^{GRX} and δ^{LRX} for the simulated scenes SimUsgs01 (columns 1–2) and SimUsgs02 (columns 3–4) using different SNR values.

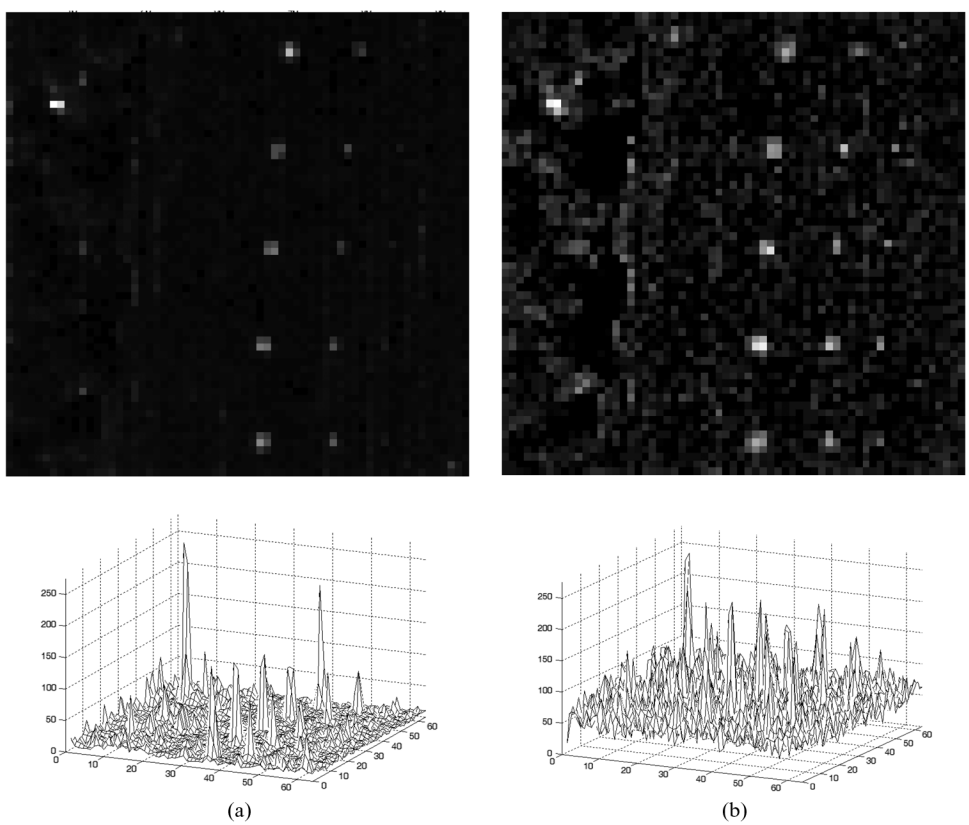


Fig. 9. Processing output of δ^{GRX} and δ^{LRX} for the HYDICE scene. (a) δ^{GRX}; (b) δ^{LRX}.

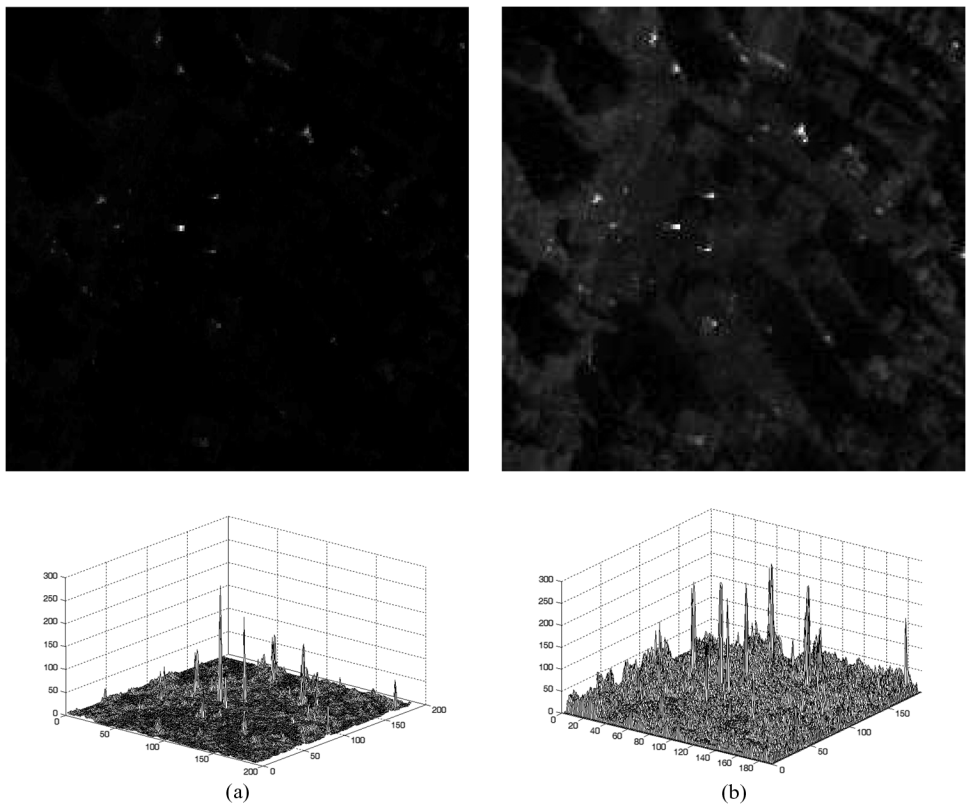


Fig. 10. Processing output of δ^{GRX} and δ^{LRX} for the WTC scene. (a) δ^{GRX}; (b) δ^{LRX}.

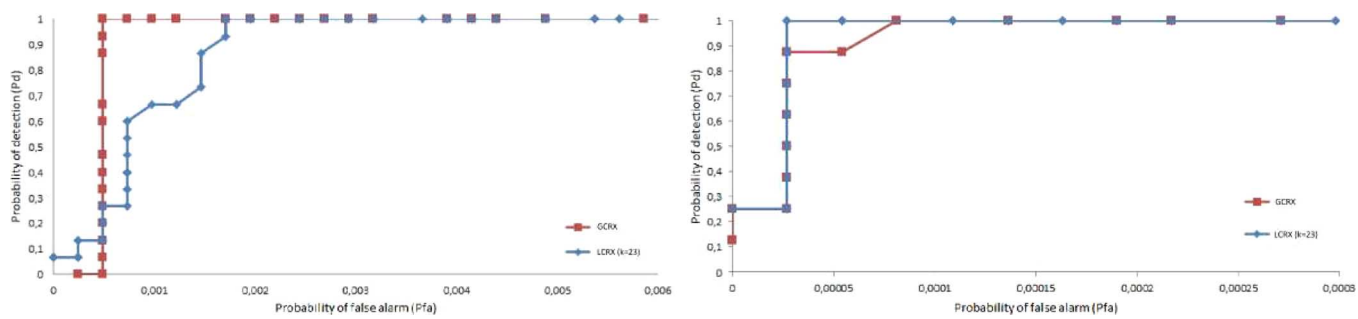


Fig. 11. ROC curves for δ^{GRX} and δ^{LRX} in the experiment with the HYDICE and WTC hyperspectral scene.

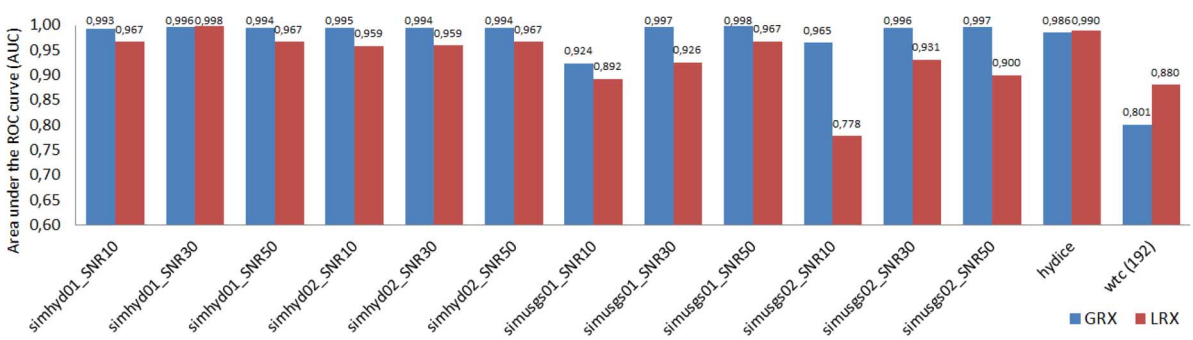


Fig. 12. Quantitative analysis of anomaly detection accuracy, indicated by the area under the ROC curve (AUC), for all the simulated and real scenes considered in this study.

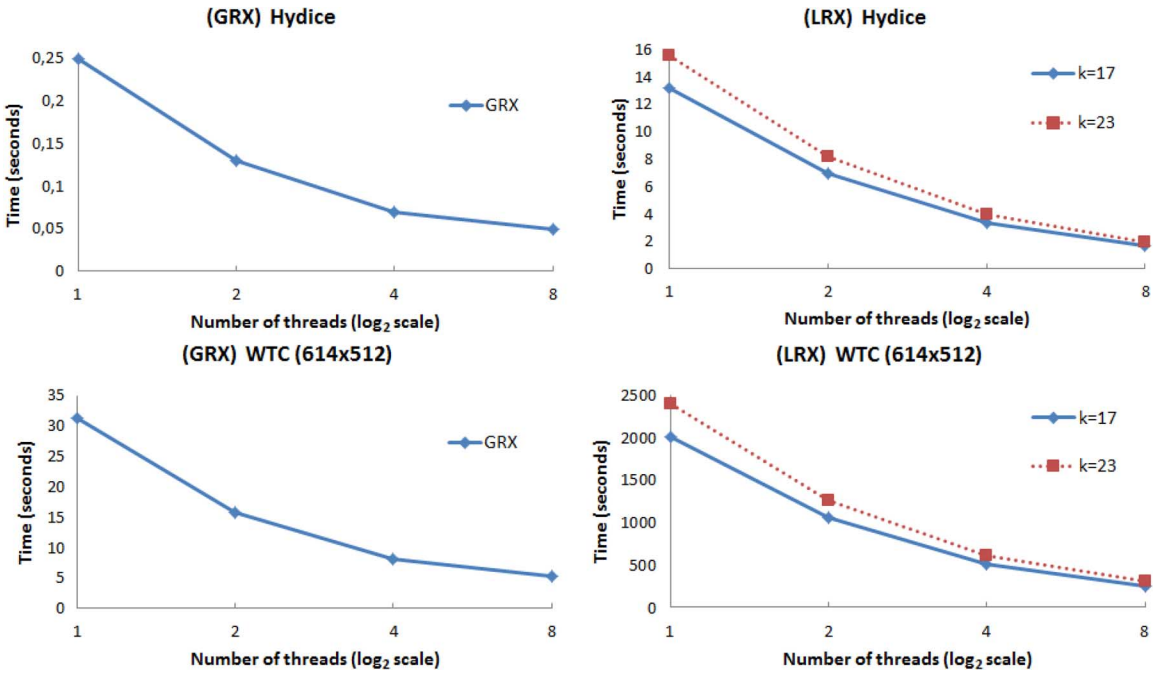


Fig. 13. Graphical representation of the execution time (in seconds) of the parallel GRX and LRX algorihtms for the hyperspectral scenes HYDICE and WTC.

TABLE II
EXECUTION TIME (IN SECONDS) WITHOUT AND WITH THE PROPOSED SOFTWARE OPTIMIZATIONS FOR THE CALCULATION OF DIFFERENT STEPS OF THE LRX ALGORITHM FOR THE SET OF HYPERSPECTRAL IMAGES CONSIDERED IN THIS STUDY

Hyperspectral image	$\mathbf{R}_{\kappa\times\kappa}$ (optimized)	$\mathbf{R}_{\kappa\times\kappa}$ (no optimization)	$\mathbf{y}(\mathbf{x})$ (based on linear systems)	$\mathbf{y}(\mathbf{x})$ (based on inverse)
SimHyd	39.58	426.62	27.52	64.14
SimUsgs	75.88	750.03	48.91	125.44
HYDICE	8.67	98.29	6.89	16.02
WTC (192×192)	223.89	1764.07	110.32	281.34
WTC (512×614)	1455.95	15043.59	938.46	2406.87

for all tested methods and scenes based on the AUC, where the values of AUC for the synthetic images with several noise levels ($\mathrm{SNR} = 10:1$, $\mathrm{SNR} = 30:1$ and $\mathrm{SNR} = 50:1$), and for the real images have been represented. Globally, Fig. 12 shows that both GRX and LRX methods exhibit similar values of AUC, which in most of the cases is larger than 0.9 (here, we assume that 1.0 is the optimal value, i.e. 100% AUC). However, LRX outperforms GRX for both real images, and particularly for WTC. These numerical results are coherent with the output filters presented in Figs. 9 and 10 where more anomalies of small size can be visually identified for the LRX filters.

Our introspection is that the selection of GRX or LRX is strongly application-dependent although LRX seems to be more robust than GRX for the cases of very small and sub-pixel anomalies in scenes with anomalies of different sizes (as it is often the case in real scenes). It should be noted that sub-pixel anomalies are of great interest in hyperspectral data exploitation given the highly mixed nature of most pixels collected by imaging spectrometers. From these results, we also conclude that the combination of both filters (GRX and LRX) in a single pass, in an adaptive fashion depending on the size of the considered anomalies, is a topic deserving future research.

B. Assessment of the Parallel Performance of GRX and LRX on Multicore Architectures

Our proposed optimizations of GRX and LRX have been evaluated in terms of parallel performance on a multicore architecture composed by an octo-core 1.87 GHz Intel Xeon L7555 (8 cores) with 16 GB of main memory. The experiments were carried out using the Linux Debian (version 2.6.32) operating system, the C compiler `icc` (version 12.0.4) and the parallel interface `POSIX Threads NTPL` (version 2.7).[5]

Table II summarizes the acceleration obtained by the proposed software optimizations for the calculation of the LRX algorithm. At this point, it is important to emphasize that the LRX algorithm is computationally much more expensive than GRX as explained in Section II. As shown in Table II, which reports the results for all the hyperspectral images considered in this study, the adopted software optimizations lead to significant improvements in all cases. On the one hand, the computation time of $\mathbf{R}_{\kappa\times\kappa}$ based on the recurrence relation is almost 10 times lower. On the other hand, if the computation of $\mathbf{y}(\mathbf{x})$ is based on the solution of linear systems, the time for this stage is reduced in a factor of $F = 2.5$ (on average) with respect to

[5]https://computing.llnl.gov/tutorials/pthreads.

TABLE III
EXECUTION TIME (IN SECONDS) AND SPEEDUP OBTAINED AFTER APPLYING THE PROPOSED PARALLEL OPTIMIZATIONS FOR THE GRX AND LRX ALGORITHMS TO ALL THE HYPERSPECTRAL IMAGES CONSIDERED IN THIS STUDY

SimHyd $L = 128$ $S = 128$ $B = 169$						
	GRX		LRX ($\kappa = 17$)		LRX ($\kappa = 23$)	
Number of threads	Time	Speedup	Time	Speedup	Time	Speedup
1	1.11	–	56.65	–	67.1	–
2	0.60	1.85	29.81	1.9	35.31	1.9
4	0.33	3.36	14.52	3.9	17.2	3.9
8	0.23	4.83	7.26	7.8	8.6	7.9
SimUsgs $L = 128$ $S = 128$ $B = 224$						
	GRX		LRX ($\kappa = 17$)		LRX ($\kappa = 23$)	
Number of threads	Time	Speedup	Time	Speedup	Time	Speedup
1	1.55	–	104.16	–	124.79	–
2	0.96	1.61	54.82	1.9	65.68	1.9
4	0.54	2.87	26.71	3.9	32.1	3.9
8	0.34	4.56	13.34	7.8	16.3	7.8
HYDICE $L = 64$ $S = 64$ $B = 169$						
	GRX		LRX ($\kappa = 17$)		LRX ($\kappa = 23$)	
Number of threads	Time	Speedup	Time	Speedup	Time	Speedup
1	0.25	–	13.18	–	15.56	–
2	0.13	1.92	6.94	1.9	8.19	1.9
4	0.07	3.57	3.38	3.9	3.99	3.9
8	0.05	5.01	1.69	7.8	1.99	7.8
WTC (192×192) $L = 192$ $S = 192$ $B = 224$						
	GRX		LRX ($\kappa = 17$)		LRX ($\kappa = 23$)	
Number of threads	Time	Speedup	Time	Speedup	Time	Speedup
1	3.65	–	266.39	–	334.11	–
2	1.89	1.93	140.21	1.9	175.85	1.9
4	1.28	2.85	68.31	3.9	85.67	3.9
8	0.68	5.36	33.72	7.9	42.29	7.9
WTC (614×512) $L = 614$ $S = 512$ $B = 224$						
	GRX		LRX ($\kappa = 17$)		LRX ($\kappa = 23$)	
Number of threads	Time	Speedup	Time	Speedup	Time	Speedup
1	31.30	–	1999.16	–	2394.42	–
2	15.82	1.97	1052.19	1.9	1260.22	1.9
4	8.17	3.83	512.6	3.9	613.95	3.9
8	5.28	5.93	256.3	7.9	306.98	7.9

the approach based on the computation of the inverse matrix. Notice that the value of F is related to the ratio between the computational complexity to obtain $\mathbf{y}(\mathbf{x})$ using the inverse and the solution of the linear system. The complexity to compute the inverse matrix of dimension B is $O((5/3)B^3)$ and the solution of linear system based on the QR factorization is $O((2/3)B^3)$. Globally, the optimized version of LRX is 7 times faster than the standard implementation. The sequential LRX algorithm has been optimized and taken as reference to develop the multicore parallel implementation.

Table III reports our experimental results in terms of execution time (in seconds) and speedup obtained after processing all the hyperspectral images considered in the present study. The speedup represent how much faster the optimized multicore implementation is with respect to the optimized sequential version. This means that the values of the execution time given in Table III correspond to a multithreaded version of the best available (optimized) serial code. As shown by Table III, linear speedups have been obtained for the LRX since the computational burden of this algorithm has been mapped on independent and balanced tasks, with light redundancy, which are efficiently computed by the threads on the multicore. This is an important observation, since the reported speedups are obtained using as a reference the (already optimized) sequential

implementations obtained after the proposed improvements described in this paper. Combined, the proposed sequential and parallel optimizations lead to highly improved computational performance of both GRX and LRX. It should be noted that, for the LRX experiments reported in Table III, we provide results for two different values of parameter κ (i.e., $\kappa = 17$ and $\kappa = 23$, used to generate the anomaly detection results reported in this paper) in order to illustrate that the speedup results do not depend on the values of κ. Further, for real images with different sizes and number of spectral channels, Fig. 13 shows graphs of the total runtime versus the number of threads of the parallel version and the size of the window (κ). It is worthy to point out that the execution time of LRX linearly decreases as the number of threads increases.

An important observation is that LRX is much better than GRX in terms of scalability; i.e. while the speedup of LRX is almost linear, for the GRX algorithm the speedup is around 4 when using 8 cores. Moreover, the values of the execution time achieved by the optimized version of LRX are sufficiently small to be processed at the same time as the data is collected, since the cross-track line scan time in AVIRIS, a push-broom instrument, is quite fast (8.3 msec to collect 512 full pixel vectors). This introduces the need to process the WTC scene (with 614×512 pixels, which represents the standard chunk size collected by AVIRIS before writing the data to disk onboard) in approximately 5.09 seconds to achieve similar time for processing and data acquisition. In this regard, the reported processing time of 5.28 seconds for the optimized version of GRX represents a good processing result reported for this algorithm using a small number of cores (8, in our experiments). Despite the fact that the proposed optimizations of LRX algorithm are still far from processing at the same time as the data is collected, the obtained linear speedup in this case suggests that our proposed parallel version of LRX has the potential to scale efficiently in multicore architectures with a large number of cores. In this regard, a line of research deserving future attention consists of adapting the proposed implementation of LRX to architectures with a higher number of cores, including graphics processing units (GPUs).

VI. Conclusions and Future Lines

In this paper, we have proposed optimizations of the global RX (GRX) algorithm for anomaly detection and a local version of this algorithm (called LRX) in the context of hyperspectral image processing. The proposed optimizations are twofold: on the one side, they exploit sequential optimization techniques and linear algebra libraries in order to accelerate the calculation of the correlation matrix and the least square systems, which are crucial for both algorithms. Second, we have developed new parallel implementations optimized for multi-core platforms, a well known, inexpensive and widely available high performance computing technology. A detailed quantitative and comparative assessment of the proposed algorithms has been carried out using both simulated and real hyperspectral images, reporting processing times which are on the order of the time employed to collect the data for the GRX algorithm and perfect scalability for the LRX algorithm in an inexpensive multicore system. The results obtained also suggest that GRX and LRX are complementary in nature: while LRX is often more effective in the task of detecting very small (sub-pixel) anomalies, GRX generally provides a good characterization of larger anomalies. Both approaches exhibit adequate performance in the presence of noise.

Overall, the experimental results reported in this paper suggest that the combination of linear algebra optimizations and multithreading for parallel exploitation of standard multicore architectures represents a source of computational power which can help in the important task of obtaining processing times similar to the data collection times in hyperspectral anomaly detection applications. To the best of our knowledge, the performance results reported in this paper with a very small number of cores and an inexpensive parallel platform represent a significant step forward towards the ultimate goal of being able to exploit hyperspectral data at the same time as the data is collected onboard the image acquisition platform. In this regard, our future research will be focused on adapting the proposed implementations to multicore architectures with a higher number of cores, including commodity GPUs and field programmable gate arrays (FPGAs) for onboard exploitation in airborne and spaceborne platforms for Earth observation. Another topic deserving future research is the development of an integrated approach able to exploit the advantages of both the GRX and LRX in adaptive and synergistic fashion.

Acknowledgment

The authors would like to take this opportunity to gratefully thank the editors and the anonymous reviewers for their comments and suggestions to improve the technical content and presentation of this manuscript.

References

[1] A. Goetz, G. Vane, J. Solomon, and B. Rock, "Imaging spectrometry for earth remote sensing," *Science*, vol. 228, pp. 1147–1153, 1985.

[2] A. Plaza, J. Benediktsson, J. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, J. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, pp. 110–122, 2009.

[3] R. Green, M. Eastwood, C. Sarture, T. Chrien, M. Aronsson, B. Chippendale, J. Faust, B. Pavri, C. Chovit, and M. Solis *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, no. 3, pp. 227–248, 1998.

[4] C.-I Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Norwell, MA: Kluwer, 2003.

[5] A. Plaza and C.-I Chang, *High Performance Computing in Remote Sensing*. Boca Raton, FL: CRC Press, 2007.

[6] P. Gurram, H. Kwon, and T. Han, "Sparse kernel-based hyperspectral anomaly detection," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 5, pp. 943–947, 2012.

[7] B. Du and L. Zhang, "Random selection based anomaly detector for hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 5, pp. 1578–1589, 2011.

[8] I. S. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distrihution," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 38, pp. 1760–1770, 1990.

[9] J. Richards and X. Jia, *Remote Sensing Digital Image Analysis: An Introduction*. New York: Springer, 2006.

[10] Y. P. Taitano, B. A. Geier, and K. W. B. Jr, "A locally adaptable iterative rx detector," *EURASIP J. Advances in Signal Processing*, p. 10, 2010, 10.1155/2010/341908.

[11] L. Ma, M. M. Crawford, and J. Tian, "Anomaly detection for hyperspectral images based on robust locally linear embedding," *J. Infrared Millimeter and Terahertz Waves*, no. 31, pp. 753–762, 2010, 10.1007/s10762-010-9630-3.

[12] J. M. Molero, E. M. Garzón, I. García, and A. Plaza, "Anomaly detection based on a parallel kernel rx algorithm for multicore platforms," *J. Appl. Remote Sens.*, vol. 6, p. 12, 2012, 0091-3286/2012.
[13] G. Shaw and D. Manolakis, "Signal processing for hyperspectral image exploitation," *IEEE Signal Process. Mag.*, vol. 19, pp. 12–16, 2002.
[14] C.-I Chang and S.-S. Chiang, "Anomaly detection and classification for hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 6, pp. 1314–1325, 2002.
[15] D. W. J. Stein, S. G. Beaven, L. E. Hoff, E. M. Winter, A. P. Schaum, and A. D. Stocker, "Anomaly detection from hyperspectral imagery," *IEEE Signal Process. Mag.*, vol. 19, pp. 58–69, 2002.
[16] D. Manolakis, D. Marden, and G. A. Shaw, "Hyperspectral image processing for automatic target detection applications," *MIT Lincoln Lab. J.*, vol. 14, pp. 79–116, 2003.
[17] S. Matteoli, M. Diani, and G. Corsini, "A tutorial overview of anomaly detection in hyperspectral images," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 25, no. 7, pp. 5–28, Jul. 2010.
[18] Y. Tarabalka, T. V. Haavardsholm, I. Kasen, and T. Skauli, "Real-time anomaly detection in hyperspectral images using multivariate normal mixturemodels and gpu processing," *J. Real-Time Image Process.*, vol. 4, pp. 1–14, 2009.
[19] C.-I Chang, H. Ren, and S.-S. Chiang, "Real-time processing algorithms for target detection and classification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 4, pp. 760–768, 2001.
[20] C. Lee, S. Gasster, A. Plaza, C.-I Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 508–527, 2011.
[21] A. Plaza, D. Valencia, J. Plaza, and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *J. Parallel Distrib. Comput.*, vol. 66, pp. 345–358, 2006.
[22] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*. San Diego, CA: Morgan Kaufmann, 2006.
[23] A. Remon, S. Sanchez, A. Paz, E. S. Quintana-Ortí, and A. Plaza, "Real-time endmember extraction on multi-core processors," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, pp. 924–928, 2011.
[24] M. Marqués, G. Quintana-Ortí, E. Quintana-Ortí, and R. van de Geijn, "Using desktop computers to solve large-scale dense linear algebra problems," *J. Supercomput.*, vol. 58, pp. 145–150, 2011.
[25] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. V. D. Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1990.
[26] J. M. Molero, A. Paz, E. M. Garzón, J. A. Martínez, A. Plaza, and I. García, "Fast anomaly detection in hyperspectral images with Rx method on heterogeneous clusters," *J. Supercomput.*, vol. 58, no. 3, pp. 411–419, 2011, 10.1007/s11227-011-0598-0.
[27] A. Paz, A. Plaza, and S. Blazquez, "Parallel implementation of target and anomaly detection algorithms for hyperspectral imagery," in *Proc. IEEE Geosci. Remote Sens. Symp.*, 2008, vol. 2, pp. 589–592.
[28] A. Paz, A. Plaza, and J. Plaza, "Comparative analysis of different implementations of a parallel algorithm for automatic target detection and classification of hyperspectral images," in *Proc. SPIE*, 2009, vol. 7455, pp. 1–12.
[29] LAPACK Linear Algebra PACKage User Guide. LAPACK 3.3.1. 2011 [Online]. Available: http://www.netlib.org/lapack/lapack-3.3.1.html
[30] Intel Math Kernel Library—Documentation. 2011 [Online]. Available: http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/
[31] H. Kwon and N. M. Nasrabadi, "Hyperspectral anomaly detection using Kernel Rx-algorithm," in *Int. Conf. Image Processing (ICIP)*, 2004.
[32] D. Borghys, I. Kåsen, V. Achard, and C. Perneel, "Comparative evaluation of hyperspectral anomaly detectors in different types of background," in *Proc. SPIE 8390, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVIII*, 2012, pp. 83 902J–83 902J-12 [Online]. Available: http://dx.doi.org/10.1117/12.920387
[33] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins University Press, 1996.
[34] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. New York: Cambridge University Press, 1992.
[35] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1999.
[36] G. E. Suh, L. Rudolph, and S. Devadas, "Dynamic partitioning of shared cache memory," *J. Supercomput.*, vol. 28, pp. 7–36, 2004.
[37] D. Andrade, M. Arenaz, B. B. Fraguela, J. Tourino, and R. Doallo, "Automated and accurate cache behavior analysis for codes with irregular access patterns," *Concurrency and Computation: Practice and Experience*, no. 19, pp. 2407–2433, 2002.
[38] J. C. Harsanyi and C.-I Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, pp. 779–785, 1994.

José Manuel Molero was born in Almería, Spain, in 1983. He received the M.Sc. degree in computer science from the University of Almería in 2009. He is currently pursuing the Ph.D. degree at the Supercomputing-Algorithms research group, University of Almería.

His research interest include high performance computing and remote sensing, especially in hyperspectral image processing.

Ester M. Garzón received the B.Sc. degree in physics at the University of Granada, Spain, in 1985, and the Ph.D. degree in computer science from the University of Almería, Spain, in 2000.

Currently, she is an Associate Professor at the Department of Informatics at University of Almería and she is head of the research group Supercomputing Algorithms. Her research interest focuses on high performance computing, matrix computation and imaging processing. References on her works in these lines can be found in http://www.hpca.ual.es/gmartin/.

Inmaculada García received the B.Sc. degree in physics from the Complutense University of Madrid, Spain, in 1977, and the Ph.D. degree from the University of Santiago de Compostela, Spain, in 1986.

From 1977 to 1987, she was an Assistant Professor, Associate professor during 1987–1997, between 1997 and 2010, Full Professor at the University of Almeria and, since 2010, Full Professor at the University of Málaga. She was head of the Department of Computer Architecture and Electronics at the University of Almería for more than 12 years. During 1994–1995 she was a visiting researcher with the University of Pennsylvania, Philadelphia. Her research interest lies in the field of parallel algorithms for irregular problems related to image processing, global optimization, and matrix computation.

Antonio Plaza (M'05–SM'07) was born in Cáceres, Spain, in 1975. He received the M.S. and Ph.D. degrees in computer engineering from the University of Extremadura, Spain.

He has been a Visiting Researcher with the Remote Sensing Signal and Image Processing Laboratory (RSSIPL), University of Maryland Baltimore County; with the Applied Information Sciences Branch, NASA Goddard Space Flight Center (GSFC), Greenbelt, MD; with the Airborne Visible Infrared Imaging Spectrometer Data Facility,

NASA Jet Propulsion Laboratory (JPL), Pasadena, CA; with the Telecommunications and Remote Sensing Laboratory, University of Pavia, Italy; and with the GIPSA-lab, Grenoble Images Parole Signal Automatique, France. He is currently an Associate Professor (with accreditation for Full Professor) with the Department of Technology of Computers and Communications, University of Extremadura, where he is the Head of the Hyperspectral Computing Laboratory (HyperComp). He was the Coordinator of the Hyperspectral Imaging Network, a European project designed to build an interdisciplinary research community focused on hyperspectral imaging activities. He is the author or coauthor of more than 350 publications on remotely sensed hyperspectral imaging, including more than 90 journal citation report papers (45 since January 2011), around 20 book chapters, and over 230 conference proceeding papers. His research interests include remotely sensed hyperspectral imaging, pattern recognition, signal and image processing, and efficient implementation of large-scale scientific problems on parallel and distributed computer architectures.

Dr. Plaza has guest edited seven special issues on scientific journals on the topic of remotely sensed hyperspectral imaging. He has been a Chair for the IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (Whispers 2011). He has also been serving as a Chair for the SPIE Conference on Satellite Data Compression, Communications, and Processing, since 2009, and for the SPIE Europe Conference on High-Performance Computing in Remote Sensing, since 2011. He has been 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, a journal for which he has served as Associate Editor since 2007 and for which he has reviewed more than 260 manuscripts. He served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) in 2011–2012, and is currently serving as President of the Spanish Chapter of IEEE GRSS. As of January 2013, he starts a three-year term as Editor of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING journal.