Journal of
Applied Remote Sensing

Anomaly detection based on a parallel kernel RX algorithm for multicore platforms

José M. Molero
Ester M. Garzón
Inmaculada García
Antonio Plaza

Anomaly detection based on a parallel kernel RX algorithm for multicore platforms

José M. Molero,[a] Ester M. Garzón,[a] Inmaculada García,[b] and Antonio Plaza[c]

[a]University of Almería, Department of Computer Architecture and Electronics, Agrifood Campus of International Excellence (CEIA3), Ctra Sacramento s/n. 04120 Almería, Spain
jmp384@ual.es

[b]University of Málaga, Department of Computer Architecture, Escuela de Ingenierías, Campus de Teatinos, 29071 Málaga, Spain

[c]University of Extremadura, Hyperspectral Computing Laboratory, Avda. de la Universidad s/n, E-10071 Cáceres, Spain

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 Yu, also called RX algorithm. It implements the Mahalanobis distance, which has been widely used in hyperspectral imaging applications. A variation of this algorithm, known as kernel RX (KRX), consists of applying the same concept to a sliding window centered around each image pixel. KRX is computationally very expensive because, for every image pixel, a covariance matrix and its inverse has to be calculated. We develop an efficient implementation of the kernel RX algorithm. Our proposed approach makes use of linear algebra libraries and further develops a parallel implementation optimized for multi-core platforms, which is a well known, inexpensive and widely available high performance computing technology. Experimental results for two hyperspectral data sets are provided. The first one was 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 the second one was collected by the hyperspectral digital image collection experiment (HYDICE). Our anomaly detection accuracy, evaluated using receiver operating characteristics (ROC) curves, indicates that KRX can significantly outperform the classic RX while achieving close to linear speedup in state-of-the-art multi-core platforms. [DOI: 10.1117/1.JRS.6.061503]

Keywords: hyperspectral image processing; anomaly detection; kernel RX algorithm; multi-core processors.

Paper 11209SS received Sep. 30, 2011; revised manuscript received Feb. 15, 2012; accepted for publication Mar. 7, 2012; published online May 10, 2012.

1 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" (Fig. 1) 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 gigabytes (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

0091-3286/2012/$25.00 © 2012 SPIE

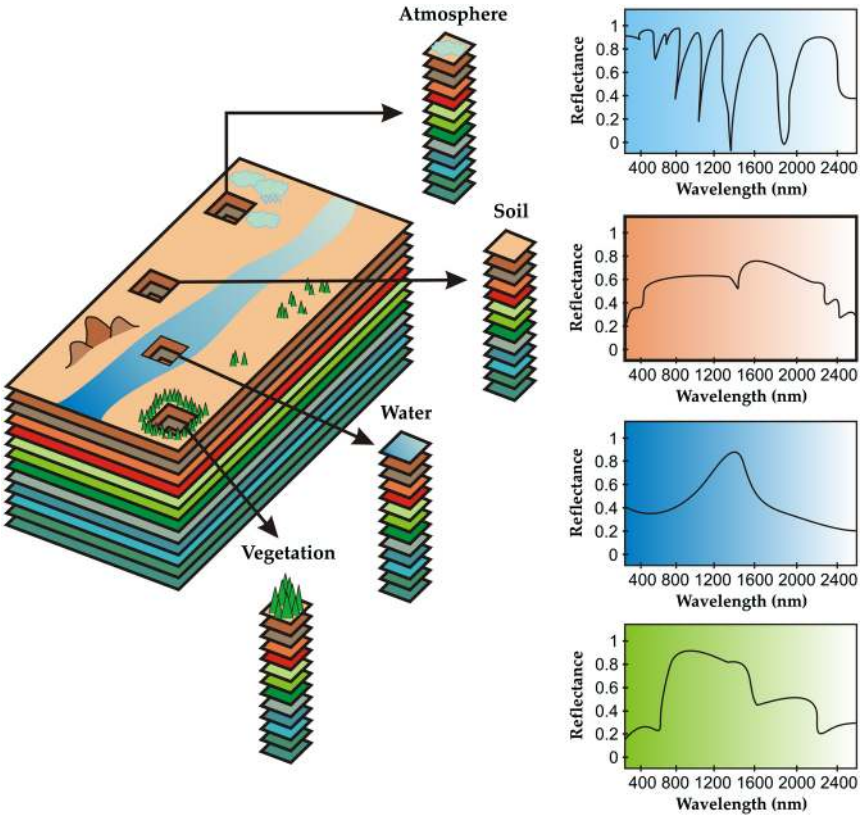

Fig. 1 Concept of hyperspectral imaging.

surroundings with no *a priori* knowledge. In general, such anomalous signatures are relatively small compared to the image background, and only occur in the image with low probabilities. The RX algorithm, developed by Reed and Yu, is a well-know approach for anomaly detection which has shown success for multispectral and hyperspectral images.[4] The RX algorithm uses the pixel currently being processed as the matched signal. Since RX uses the sample covariance matrix to take into account the sample spectral correlation, it performs the same task as the Mahalanobis distance, which has been widely used in hyperspectral imaging applications.[6]

A variation of this algorithm, known as the kernel RX algorithm (KRX), consists of applying the same concept to a sliding window centered around each image pixel.[7] Kernel RX is very interesting due to its high accuracy to detect anomalies. However, it is computationally very expensive because involves the computation of a covariance matrix and its inverse for every image pixel. Consequently, the literature describes this kind of methods to detect anomalies as unachievable for real-time applications.[8,9] Nevertheless, current advances in the high performance computing field (HPC) offer resources which can strongly reduce the runtimes for the kernel RX algorithms. Thus, the modern multicore architectures[10,11] represent an inexpensive, widely available and well-known technology in the HPC field and, moreover, a wide set of linear algebra libraries has been developed to reduce the time for solving algebra problems by means of the multi-core systems.[12,13] In this way, HPC allows that kernel RX methods can be revised as practical algorithms in the anomaly detection.

Parallel implementations of the RX algorithm and some of their variations can be found in Refs. 14–16. The large computational burden of the KRX algorithm prevents its sequential executions on single core systems. So, we describe and analyze a parallel approach for accelerating the KRX algorithm on multicore systems.

The goal of this work is to explore the accuracy and performance of the kernel RX algorithm exploiting state-of-the-art multicore architectures and using real test hyperspectral images.

Our aim is to optimize the exploitation of multicore architectures by a parallel version of the kernel RX algorithm. A comparative evaluation of the accuracy for the kernel and the classical RX algorithms is also carried out. Their capacity to detect anomalies is analyzed by means of

receiver operating characteristics curves (ROC curves).[17] The analysis is carried out using images taken from two different real scenes. The first is a data set collected by the 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; in this example, anomalies are thermal hot spots. The second is a data set collected by the hyperspectral digital Image collection experiment (HYDICE); this image is interesting for anomaly detection because it includes a set of 15 panels (anomalies in this context) with several different spectral signatures and several sizes.

The remainder of the paper is structured as follows. Section 2 briefly describes the classic and kernel versions of the RX algorithm. Section 3 defines the hyperspectral data used in experiments, and a ROC analysis of anomaly detection accuracy is carried out. Section 4 is focused on the parallel implementation of the kernel RX algorithm on a multicore architecture, where experimental results are analyzed in terms of scalability. Finally, Sec. 5 concludes with some remarks and hints at plausible future research.

2 Anomaly Detection Based on RX Algorithm

2.1 *RX Algorithm*

The RX algorithm has been widely used in signal and image processing.[18] The result of this algorithm is referred to as RX filter and defined by the following expression:

$$\delta^{\mathrm{RX}}(\mathbf{x}) = (\mathbf{x}-\mu)^T\mathbf{K}^{-1}(\mathbf{x}-\mu), \tag{1}$$

where $\mathbf{x} = [x^{(0)}, x^{(1)}, \cdots, x^{(n)}]$ is a sample, n-dimensional hyperspectral pixel (vector), μ is the sample mean of the hyperspectral image and $\mathbf{K}$ is the sample data covariance matrix. As we can see, the form of δ^{RX} is actually the well-known Mahalanobis distance.[6] Replacing $\mathbf{K}$ with the sample correlation matrix $\mathbf{R}$ and replacing $\mathbf{x}-\mu$ with the pixel vector, results in a sample correlation matrix-based RX filter given by

$$\delta^{\mathrm{RX}}(\mathbf{x}) = \mathbf{x}^T\mathbf{R}^{-1}\mathbf{x}, \tag{2}$$

where the sample correlation matrix $\mathbf{R}$ is used in the filter design.[4]

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

2.2 *Kernel RX Algorithm*

The classic RX algorithm can be considered as a global anomaly detector because the covariance (correlation) matrix is computed using all the pixels of the image. The kernel RX algorithm[7] can be considered as a local anomaly detector because each pixel of the image has its own covariance (correlation) matrix, which is computed just considering a small set of neighborhood pixels of the pixel under test.

The kernel RX algorithm developed in this work (KRX algorithm), uses the concept of a sliding local window (kernel) for every pixel in the image. For each pixel $\mathbf{x}$ of the image, the RX filter is computed using this local square window of size $\kappa \times \kappa$ centered at pixel $\mathbf{x}$. So, matrix $\mathbf{K}_\kappa$ ($\mathbf{R}_\kappa$) and vector μ_κ are calculated for every pixel $\mathbf{x}$ based on its own local window. Consequently, the KRX filter is defined by:

$$\delta_\kappa^{\mathrm{RX}}(\mathbf{x}) = (\mathbf{x}-\mu_\kappa)^T\mathbf{K}_\kappa^{-1}(\mathbf{x}-\mu_\kappa), \tag{3}$$

$$\delta_\kappa^{RX}(\mathbf{x}) = \mathbf{x}^T\mathbf{R}_\kappa^{-1}\mathbf{x}. \tag{4}$$

The covariance (correlation) matrix is estimated from a small number of high dimensional data samples, which could involve rank-deficient matrices (non invertible matrices). An almost

rank-deficient matrix presents a high condition number and, therefore, inversion is numerically difficult.[19] To overcome numerical instabilities, the pseudo-inverse based on the QR method allows us to compute the KRX filter.[19,20]

Our introspection is that this approach is more suitable for parallel implementation, but it can also suffer from several problems, including the fact that pixel vectors located in small windows are almost never statistically independent (thus introducing conditioning problems in the calculated matrices), or the fact that outliers (i.e., the anomalies we are looking for) can compromise the integrity of the local statistics. Since our results obtained in terms of anomaly detection accuracy show that the variation caused using covariance or correlation is not relevant, hereinafter we assume that the correlation matrix is used.

3 Comparative Validation of Methods

3.1 *Hyperspectral Data Used in Experiments*

In this work, two real hyperspectral images have been used as test data for a comparative analysis of the RX and KRX in terms of the accuracy for anomaly detection.

One of the image scenes used for the experiments was collected by the AVIRIS instrument, which was flown by NASA's Jet Propulsion Laboratory over the World Trade Center (WTC) area 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 full data set selected for experiments consists of 614×512 pixels, 224 spectral bands, and a total size of (approximately) 140 MB. The spatial resolution is 1.7 meters per pixel. The left side part of Fig. 2 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 the left side part of Fig. 2, while burned areas appear dark gray. Smoke coming from the WTC area (in the red rectangle) and going down to south Manhattan appears bright blue due to high spectral reflectance in the 655 nm channel. Extensive reference information, collected by U.S. Geological Survey (USGS), is available for the WTC scene. In this work, we use a U.S. Geological Survey thermal map as a ground truth which shows the locations of the eight thermal hot spots (which can be considered as anomalies) at the WTC area, displayed as bright red, orange and yellow spots on the right side of Fig. 2. This figure is centered at the region where the towers collapsed, and temperatures of the thermal hot spots ranges from 700 to 1300°F.

Figure 3 shows the other real test hyperspectral image, which was collected by the hyperspectral digital image collection experiment (HYDICE) described in Ref. 4. It is an image scene with a size of 64×64 pixels with 15 panels and a ground-truth map indicating the location of the panels (real targets). The size of the panels in the first, second, and third columns is (in meters) 3×3, 2×2 and 1×1, respectively. This image was acquired by 210 spectral

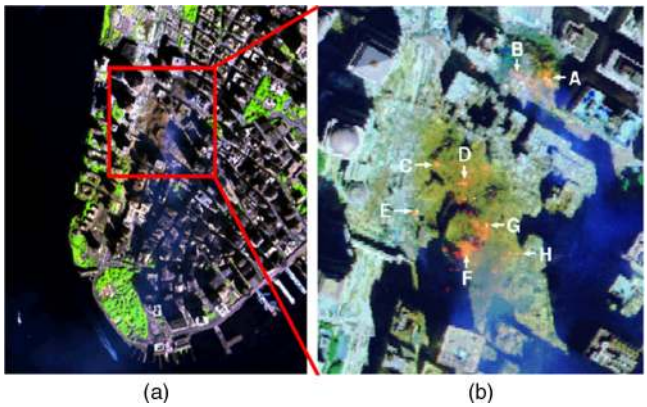

Fig. 2 (a) False color composition of an AVIRIS hyperspectral image collected by NASA's Jet Propulsion Laboratory over lower Manhattan on Sept. 16, 2001. (b) Location of thermal hot spots in the fires observed in World Trade Center area, available online: http://pubs.usgs.gov/of/2001/ofr-01 to 0429/hotspot.key.tgif.gif.

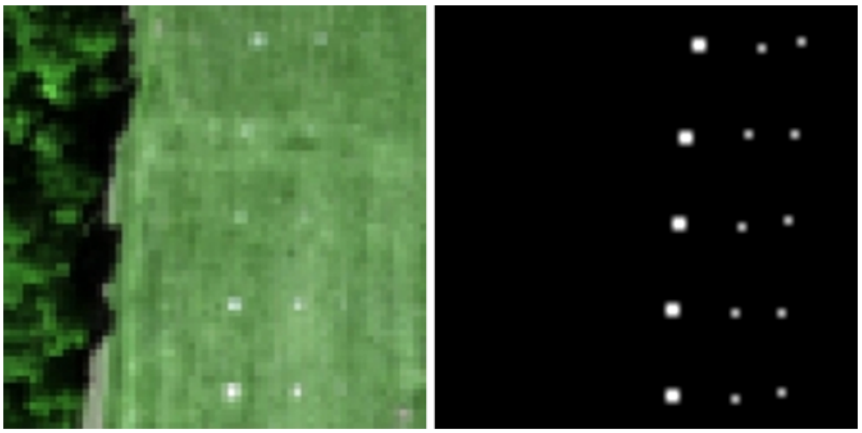

Fig. 3 (a) False color composition of an hyperspectral image collected by HYDICE. (b) Location of panels in the scene as a ground truth map.

bands with a spectral coverage from 0.4 to 2.5 microns. Low signal/high noise bands, 1 to 3 and 202 to 210, and water vapor absorption bands, 101 to 112 and 137 to 153, were removed prior to experiments, so a total of 169 bands were used. The spatial resolution is 1.56 meters and the spectral resolution is 10 nanometers.

3.2 *Analysis of Anomaly Detection Accuracy*

The receiver operating characteristics (ROC) curves represent the probability of detection (P_D) versus the probability of false-alarm (P_{FA}).[8,17] It is the standard validation metric for anomaly detection methods. The evaluation of the accuracy of the RX and the KRX algorithms is based on this metric. P_D and P_{FA} are defined as:

$$P_D = \frac{N_{\mathrm{HIT}}}{N_T}; \qquad P_{\mathrm{FA}} = \frac{N_{\mathrm{MISS}}}{N_{\mathrm{TOT}}}, \tag{5}$$

where N_{HIT} represents the number of target pixels detected given a certain threshold; N_T denotes the total number of real target pixels in the image, N_{MISS} is the number of background pixels classified as target incorrectly; and N_{TOT} represents the total number of pixels in the image. P_D becomes one only when all the individual real target pixels within a target are detected.

In order to generate ROC curves, the coordinates of all the real target positions has been obtained from the ground truth information of the hyperspectral images. Every pixel inside real target regions is considered as a target candidate to be detected.

In ROC analysis, the area under the curve (AUC) provides a reasonable estimate of anomaly detection accuracy. The larger the value of AUC the better the anomaly detection accuracy. Figures 4 and 5 show the ROC curves for algorithms RX and KRX, for two different sizes of the kernel (23 and 25), applied to test images in Figs. 2 and 3, respectively. As we can see in Figs. 4 and 5, there are differences in the ROC curves obtained for HYDICE and WTC datasets as the scale considered for the probability of false alarm is different in both cases. There are some reasons that explain this behavior. The size of the scenes is different with the AVIRIS scene being much larger than the HYDICE one. The distribution of the anomalies is different (in the HYDICE image the anomalies are closer to each other), and this represents an advantage for the local version. Finally the size of the anomalies, since the local version is expected to better detect small anomalies that are close to large anomalies. For the WTC image, the value $P_D = 1$ is achieved for very small values of P_{FA} (0.0001), for both kernel sizes and global RX. For the HYDICE image, the best accuracy results are obtained by KRX.

Results show that the KRX algorithm has the advantage of producing potentially less false detections than RX and a better probability of detection. So, according to these validation results, it can be concluded that the KRX algorithm improves the accuracy for anomaly detection when an appropriate size of the kernel is selected.

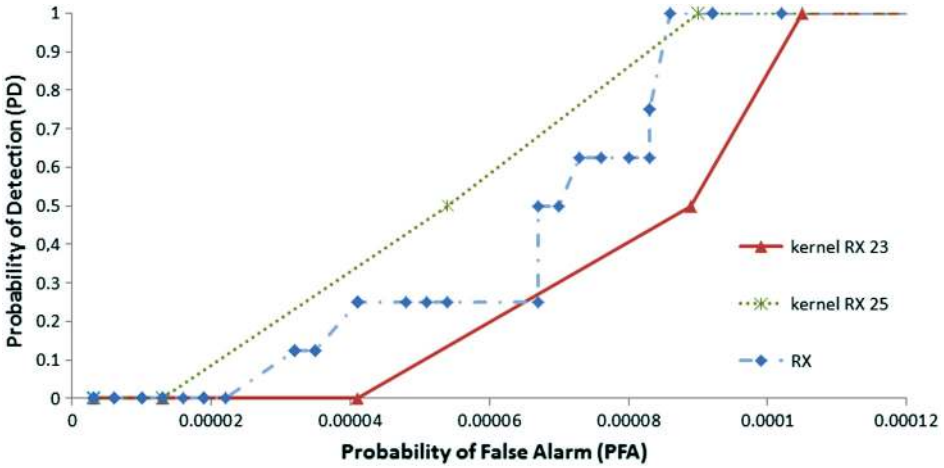

Fig. 4 ROC curves of WTC image using the RX algorithm and the KRX algorithm ($\kappa = 23$ and $\kappa = 25$).

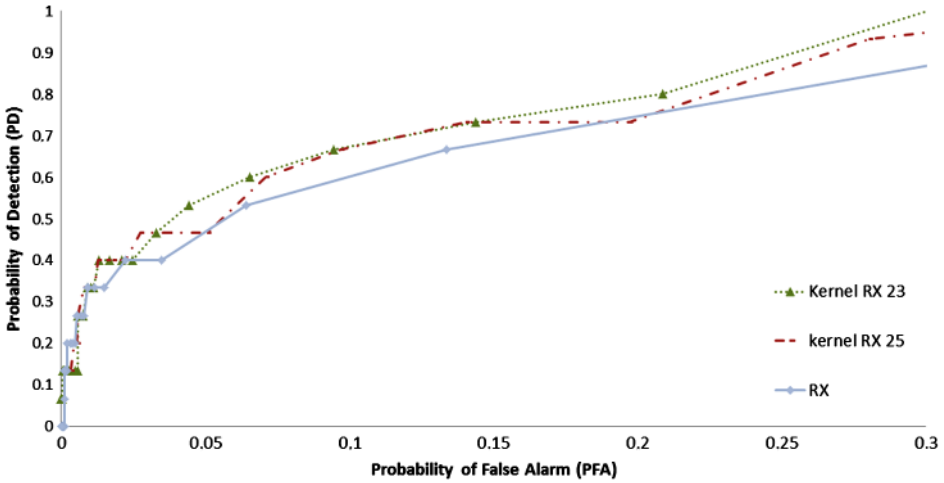

Fig. 5 ROC curves of HYDICE image using the RX algorithm and the KRX algorithm ($\kappa = 23$ and $\kappa = 25$).

From a computational point of view, the KRX burden is extremely high, due to the computation of a large number of correlation matrices and their pseudo-inverses. For this reason, the use of high performance computing techniques is essential for fast execution of the KRX algorithm. The next section describes an efficient parallel implementation of the KRX algorithm on multicore processors.

4 Tuning KRX on Multi-Core Architectures

The KRX algorithm has been improved at the sequential level and at the thread level parallelism.

The sequential version of the KRX algorithm has been optimized as much as possible and several optimizations have been implemented. Our interest has been focused on tuning the most computationally expensive stages such as the computation of correlation matrix and its pseudo-inverse.

In this line, several optimizations have been developed. The computation associated with the correlation matrix has been reduced taking advantage of the symmetric properties of this kind of matrices. This allows us to compute only half of a matrix, and to reduce the computation of the correlation matrix by nearly half. The runtime of the pseudo-inverse has been reduced using an optimized algebraic library (Intel MKL 10.3). This library is able to appropriately exploit the resources of every superescalar core. Our approach to compute the pseudoinverse (defined in Sec. 2.2) is based on the routine dgels, which allows to solve a least square problem defined by

Algorithm 1 KRX algorithm

1. load HI (Hyperspectral Image)
2. **for** $i = 0 \rightarrow \text{lines}(L)$ **do**
3. **for** $j = 0 \rightarrow \text{samples}(S)$ (cyclically distributed among T threads) **do**
4. Compute R_κ matrix
5. Compute R_κ^{pinv} {pseudoinverse using dgels MKL routine}
6. Compute $\delta_\kappa^{\text{RX}}(x)$
7. **end for**
8. **end for**
9. write δ^{RX}

$AX = B$, where A and B are known matrices and X is an unknown matrix. If $B = I$, matrix X computed by dgels is the pseudoinverse based on the QR factorization, according to the MKL library specifications.[13] Appropriate data structures have been defined in order to obtain a regular code, reducing the conditional branches in the program. This is achieved by adding an halo of redundant pixels around the image with a reduction in the KRX runtime of 30%.

KRX algorithm consists in evaluating for every pixel of the image, with a computational complexity which strongly depends on the number of bands, the size of the image, and the kernel size: $O[S \cdot L \cdot B^2 \cdot (B + \kappa^2)]$, where $S =$ samples, $L =$ lines, $B =$ bands. and $\kappa =$ kernel size. Its parallel implementation is carried out distributing the set of pixels among the cores of the system as described in Algorithm 1. KRX algorithm is composed of a set of independent procedures or tasks, where each task computes the kernel associated with the pixel under test. For each core, a thread is created to compute the lines 4, 5, and 6 of Algorithm 1 for the subset of pixels associated with it. The original image pixels have been mapped onto the set of T threads following a cyclic distribution, as shown in Fig. 6. It is important to emphasize that other mapping approaches have been tested and that the cyclic distribution has achieved the best results in terms of memory management (i.e., number of cache misses) for the shared memory architecture used.

In our parallel KRX implementation, every line of the image is processed in parallel using threads. Each thread will work on pixels that are separated T positions in the same line. In this way, this thread mapping strategy exploits and takes advantage of the memory hierarchy of a multicore. Moreover, this approach maintains the local features of a sequential version of the

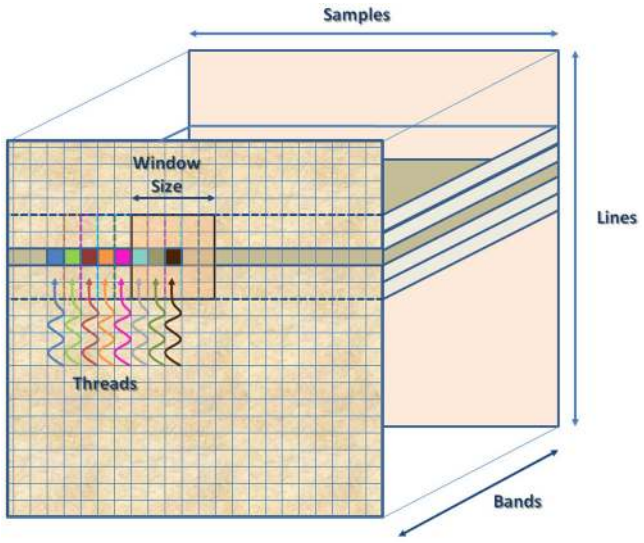

Fig. 6 Mapping the workload of each sliding window (kernel) among threads.

Table 2 Runtime (**RTime**) (in sec) and speedup (**SpUp**) values obtained from the execution of KRX for one single line and the full image of HYDICE (64 lines × 64 samples × 169 bands) and WTC (614 lines × 512 samples × 224 bands). The number of threads (PThreads) $T = 1, 2, 4$ and 8. The kernel size was 15×15 and 23×23.

	One single line							
	Hydice				WTC			
	$\kappa = 15$		$\kappa = 23$		$\kappa = 15$		$\kappa = 23$	
T	RTime	SpUp	RTime	SpUp	RTime	SpUp	RTime	SpUp
1	1.01		1.52		20.02		20.92	
2	0.53	1.9	0.80	1.9	10.43	1.9	11.01	1.9
4	0.26	3.9	0.38	3.9	5.03	3.9	5.24	3.9
8	0.13	7.6	0.20	7.6	2.52	7.6	2.75	7.6
	Full image							
	Hydice				WTC			
	$\kappa = 15$		$\kappa = 23$		$\kappa = 15$		$\kappa = 23$	
T	RTime	SpUp	RTime	SpUp	RTime	SpUp	RTime	SpUp
1	64.20		77.66		12597.41		12647.51	
2	33.79	1.9	40.87	1.9	6629.19	1.9	6656.10	1.9
4	16.46	3.9	19.46	3.9	3215.23	3.9	3242.95	3.9
8	8.45	7.5	10.35	7.6	1640.19	7.6	1664.30	7.6

KRX algorithm; that is, as a new line in the image is obtained, KRX can compute the RX filter without requiring the full image. This feature is interesting for real time systems but not provided by the classic RX algorithm.

A parallel implementation of the KRX has been evaluated in terms of performance on a multicore platform: a Dell PowerEdge R810, composed by 1 octo-core 1.87 GHz Intel Xeon L7555 (8 cores), with 16 Gb of main memory. The experiments were carried out using Linux Debian 2.6.26, the C compiler icc 12.0.4 and the parallel interface POSIX Threads NTPL 2.7 [PThreads (https://computing.llnl.gov/tutorials/pthreads)].

Table 2 summarizes the experimental results of the parallel executions of KRX in terms of the run-time (RTime) and speedup (SpUp). This table includes data for HYDICE and WTC, considering one single line of each image and the full images. Two kernel sizes (15×15 and 23×23) have been tested. These results show that: (1) linear speedup is obtained and high scalability is achieved for all tests; the run-time not only depends on the pixels per line, but also on the number of spectral bands; the run-time related to the computation of the covariance/correlation matrices strongly increases with the size of the kernel, as was predictable, however the speedup does not depend on the kernel size.

5 Conclusions and Future Work

In this paper, we have developed a new parallel implementation of the kernel version of the RX algorithm (KRX) for anomaly detection in hyperspectral imagery. The KRX algorithm, which is shown to outperform the classic RX in terms of anomaly detection accuracy, has

been implemented in parallel using a cyclic mapping strategy on a multicore architecture. This algorithm has shown to be able to exploit the memory hierarchy of a modern multi-core architecture. Moreover, the allocation of the computational workload between threads has been optimized. As a result of these optimizations, our implementation achieves almost optimum performance on multi-core systems (experimental results show linear speedup). Despite the run-time being strongly reduced when all the resources of the multicore system are used, this approach is not already an alternative for being used in real time systems, since an AVIRIS sensor can obtain a line of the image in less than 0.01 sec, which is less than the time used to compute a single kernel (approximately 0.04 sec).

As future work, we have in mind to exploit other potential types of parallelism for the computation of the correlation matrix. Other types of high performance computing platforms such as graphic processing units (GPUs) may help to exploit even more deeply the inherent parallelism of the KRX algorithm.

Acknowledgments

This work has been funded by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117 and AYA2008-05965-C04-02), Junta de Andalucia (P08-TIC-3518 and P10-TIC-6002), and Junta de Extremadura (PRI09A110 and GR10035), in part financed by the European Regional Development Fund (ERDF). The work has also been supported by the European Community's Marie Curie Research Training Networks Programme under reference MRTN-CT-2006-035927 (HYPER-I-NET).

References

1. A. Goetz et al., "Imaging spectrometry for earth remote sensing," *Science* **228**(4704), 1147–1153 (1985).http://dx.doi.org/10.1126/science.228.4704.1147.
2. A. Plaza et al., "Recent advances in techniques for hyperspectral image processing," *Rem. Sens. Environ.* **113**(1), S110–S122 (2009).http://dx.doi.org/10.1016/j.rse.2007.07.028
3. R. Green et al., "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Rem. Sens. Environ.* **65**(3), 227–248 (1998).http://dx.doi.org/10.1016/S0034-4257(98)00064-9
4. C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, Kluwer, Norwell, MA (2003).
5. A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*, CRC Press, Boca Raton (2007).
6. J. Richards and X. Jia, *Remote Sensing Digital Image Analysis: An Introduction*, Springer, Heidelberg, Germany (2006).
7. Y. P. Taitano, B. A. Geier, and K. W. Bauer, "A locally adaptable iterative rx detector," *EURASIP J. Adv. Signal Process.* **2010** 1–10 (2010).http://dx.doi.org/10.1155/2010/341908
8. D. Manolakis, D. Marden, and G. A. Shaw, "Hyperspectral image processing for automatic target detection applications," *MIT Lincoln Lab. J.* **14**(1), 79–116 (2003).
9. S. Matteoli, M. Diani, and G. Corsini, "A tutorial overview of anomaly detection in hyperspectral images," *IEEE Aero. Electron. Syst. Mag.* **25**(7), 5–28 (2010).http://dx.doi.org/10.1109/MAES.2010.5546306
10. A. Plaza et al., "Commodity cluster-based parallel processing of hyperspectral imagery," *J. Parallel Distr. Comput.* **66**(3), 345–358 (2006).http://dx.doi.org/10.1016/j.jpdc.2005.10.001
11. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publications, San Francisco, CA (2006).
12. "LAPACK linear algebra PACKage user guide. LAPACK 3.3.1," (2011).http://www.netlib.org/lapack/.
13. "Intel math kernel library? documentation," (2011).http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/.

14. J. M. Molero et al., "Fast anomaly detection in hyperspectral images with rx method on heterogeneous clusters," *J. Supercomput.* **58**(3), 411–419 (2011).http://dx.doi.org/10.1007/s11227-011-0598-0
15. A. Paz, A. Plaza, and S. Blazquez, "Parallel implementation of target and anomaly detection algorithms for hyperspectral imagery," *Proc. IEEE Geosci. Rem. Sens. Symp.* **2**, II589–II592 (7–11 July 2008).http://dx.doi.org/10.1109/IGARSS.2008.4779061
16. 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," *Proc. SPIE* **7455**, 1–12 (2009).http://dx.doi.org/10.1117/12.825458
17. H. Kwon and N. M. Nasrabadi, "Hyperspectral anomaly detection using kernel RX-algorithm," in *International Conference on Image Processing*, IEEE, Singapore, Vol. **5**, pp. 3331–3334 (24–27 October 2004).
18. I. S. Reed and X. Yu, "Adaptive multiple-band cfar detection of an optical pattern with unknown spectral distribution," *IEEE Trans. Acoust. Speech Signal Process.* **38**(10), 1760–1770 (1990).
19. A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics, Texts in Applied Mathematics*, Springer, New York (2000).
20. G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD (1996).

Biographies and photographs of the authors not available.