J Supercomput (2011) 58:411–419
DOI 10.1007/s11227-011-0598-0

Fast anomaly detection in hyperspectral images with RX method on heterogeneous clusters

J.M. Molero · A. Paz · E.M. Garzón · J.A. Martínez · A. Plaza · I. García

Published online: 30 March 2011

Abstract Remotely sensed hyperspectral sensors provide image data containing rich information in both the spatial and the spectral domain, and this information can be used to address detection tasks in many applications. One of the most widely used and successful algorithms for anomaly detection in hyperspectral images is the RX algorithm. Despite its wide acceptance and high computational complexity when applied to real hyperspectral scenes, few approaches have been developed for parallel implementation of this algorithm. In this paper, we evaluate the suitability of using a hybrid parallel implementation with a high-dimensional hyperspectral scene. A general strategy to automatically map parallel hybrid anomaly detection algorithms for hyperspectral image analysis has been developed. Parallel RX has been tested on

J.M. Molero (✉) · E.M. Garzón · J.A. Martínez
Dpt. of Computer Architecture and Electronics, University of Almería, Ctra Sacramento s/n, 04120 Almería, Spain
e-mail: jmp384@ual.es

E.M. Garzón
e-mail: gmartin@ual.es

J.A. Martínez
e-mail: jamartine@ual.es

A. Paz · A. Plaza
Dpt. of Technology of Computers and Communications, University of Extremadura, Avda. de la Universidad S/N, 10071 Cáceres, Spain

A. Paz
e-mail: apazgal@unex.es

A. Plaza
e-mail: aplaza@unex.es

I. García
Dpt. Computer Architecture, Malaga University, Escuela de Ingenierias, Campus de Teatinos, 29071 Málaga, Spain
e-mail: igarciaf@uma.es

an heterogeneous cluster using this routine. The considered approach is quantitatively evaluated using hyperspectral data collected by the NASA's Airborne Visible Infra-Red Imaging Spectrometer system over the World Trade Center in New York, 5 days after the terrorist attacks. The numerical effectiveness of the algorithms is evaluated by means of their capacity to automatically detect the thermal hot spot of fires (anomalies). The speedups achieved show that a cluster of multi-core nodes can highly accelerate the RX algorithm.

Keywords Hyperspectral imaging · Anomaly detection · Heterogeneous clusters

1 Introduction

Hyperspectral imaging [4] 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 [11]. Hyperspectral imaging instruments such as the NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) [5] are now able to record the visible and near-infrared spectrum of the reflected light of an area 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 [3]. The resulting data volume typically comprises several GBs per flight [9].

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 without prior knowledge. In general, such anomalous signatures are compared to the image background, and only occur in the image with low probabilities. A well-known approach for anomaly detection was developed by Reed and Yu, and is referred to as the RX algorithm, which has shown success in anomaly detection hyperspectral images [3].

Despite its wide acceptance and high computational complexity when applied to real hyperspectral scenes, few approaches have been developed for parallel implementation of this algorithm due to the complexity of calculating the sample covariance matrix (and its inverse) in parallel [7, 8]. A standard data partitioning framework for parallel implementation may provide different results if the sample covariance matrices are calculated independently for relatively small sub-images rather than computing the sample covariance matrix for the entire hyperspectral image, which from a parallel computing point of view would require additional inter-processor communications that may reduce parallel performance. This aspect is crucial for the RX implementation since the consideration of a local or global strategy for the computation of the sample covariance matrix is expected to significantly affect the scalability of the parallel solution. However, there is a trade-off between improving the performance of the parallel implementation and the quality of the final solution in terms of anomaly detection accuracy.

In this paper, we investigate the use of a local approach for the calculation of the inverse of the sample covariance matrix, in which each process calculates the covariance matrix of its local partition. From a parallel perspective, this reduces the

inter-processor communications. Bearing in mind this approach, a hybrid parallel implementation exploiting two levels of parallelism has been developed and tested on an heterogeneous cluster of multi-processors. We have designed a routine that evaluates the computational resources of the heterogeneous platform and automatically determines the computational load to be assigned to each processing unit.

The remainder of the paper is structured as follows. Section 2 briefly describes the classic RX algorithm. Section 3 describes the parallel implementation adopted in this work. Section 4 validates the implementation and conducts a detailed experimental assessment of the numerical accuracy and computational performance (in term of the scalability) of the proposed parallel implementations using a hyperspectral scene as a relevant case study. Finally, Sect. 5 concludes with some remarks and hints at plausible future research.

2 RX algorithm

The RX algorithm has been widely used in signal and image processing [12]. The filter implemented by 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 $\mathbf{x} = [x^{(0)}, x^{(1)}, \ldots, x^{(n)}]$ 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 [13]. It is important to note that the images generated by the RX algorithm are generally gray scale images. In this case, the anomalies can be categorized in terms of the value returned by RX, so that the pixel with higher value of $\delta^{\mathrm{RX}}(\mathbf{x})$ can be considered the first anomaly, and so on. So, from the computational point of view, the RX algorithm produces the following output in four steps:

1. Calculate the sample data covariance matrix $\mathbf{K}$ $[O(rows \cdot columns \cdot bands^2)]$
2. Calculate $\mathbf{K}^{-1}$ by the Gauss Method $[O(bands^3)]$
3. Calculate the Mahalanobis distance as δ^{RX} $[O(rows \cdot columns \cdot bands^2)]$
4. Locate the maximum values of δ^{RX} or anomalies $[O(rows \cdot columns)]$

The computational complexity of each step is described between brackets, where *rows* and *columns* are the spatial dimensions of the input hyperspectral image and *bands* is the spectral dimension. This is a relevant information for subsequent analysis of parallel RX algorithm.

3 Parallel implementation of the RX algorithm

Previous studies in the field of parallel hyperspectral image processing based on spatial domain partitioning and MPI parallel interface have reached a high performance on clusters of mono-processors [6, 7, 10]. For these implementations, a pixel vector is always entirely assigned to a single processor, and slabs of spatially adjacent pixel

vectors are distributed among the processing nodes of the parallel system. The inter-processor communication is reduced, resulting from the fact that a single pixel vector is never partitioned and communications are not needed at the pixel level. However, none of the aforementioned implementations have been designed for multi-core systems, which represent a low-power, inexpensive, widely available, and well-known technology. Modern clusters based on heterogeneous multi-core nodes can significantly reduce the run-time of hyperspectral image analysis algorithms in general and the RX anomaly detection algorithm in particular. The idea is that several nodes with different number of cores and connected using an external network cooperate together (in a balanced way) to solve the problem. However, in order to fully exploit the computational power of these clusters, additional efforts are needed to develop a new parallel implementation, which takes into account the particular characteristics of the RX algorithm. The capability and accuracy of the parallel RX algorithm to detect anomalies depend on the number of sub-images in which the full image is decomposed. The value of this parameter is chosen by the user and is not related to the number of processors of the cluster where the algorithm is run. Thus, in order to exploit the modern clusters of multi-core nodes, we propose a parallel implementation of the RX algorithm which combines the distributed and shared memory parallel programming mode, using the parallel interfaces MPI and PThreads. So, every multi-core node (hereinafter referred as node) of the cluster can execute several MPI processes which expands a set of threads.

Our parallel version of the RX algorithm for anomaly detection is a variation of the proposal in [7, 8]. We use a combination of global/local approaches for the calculation of the covariance matrix where each MPI process computes the covariance matrix of its local partition based on a multi-threaded computing model.

We can consider two levels of parallelism according to the characteristics of (1) the RX algorithm and (2) the architecture of modern clusters. The first level is based on spatial decomposition of the image cube, using the MPI parallel interface, so every MPI process computes the RX algorithm on its sub-image without communications, however, the maximum number of MPI processes is bounded because the accuracy of the RX algorithm decreases as the size of the sub-image decreases. The second level lets us exploit the shared-memory multi-processor architecture. So, each MPI process expands a set of threads by means of the POSIX Threads parallel interface. Taking into account the data dependencies of the RX algorithm and the computational load of the operations, specific loops have been selected to be parallelized in order to improve the performance.

Let N_{mpi} be the number of MPI processes and $Threads[i]$ the number of POSIX Threads that the i-th MPI process executes. Thus, the computational complexity of the hybrid parallel algorithm at each step is:

1. Compute the covariance matrix in parallel, each MPI process computes the matrix K for its local sub-image exploiting the multi-thread parallelism, then the parallel complexity for this step is $O(\frac{rows \cdot columns \cdot bands^2}{N_{mpi} \cdot Threads[i]})$.
2. Compute $\mathbf{K}^{-1}$ by the Gauss Method, using the Automatically Tuned Linear Algebra Software (ATLAS) library that exploits the multi-thread parallelism [1]. So, every MPI process computes its inverse matrix without communication between processes. The parallel complexity is $O(\frac{bands^3}{Threads[i]})$.

3. Each MPI process applies the RX filter based on the Mahalanobis distance (δ^{RX}) on its sub-image by means of a multi-threaded approach. Then the parallel complexity is $O(\frac{rows \cdot columns \cdot bands^2}{N_{mpi} \cdot Threads[i]})$.
4. In order to locate the targets, a master processor selects the pixels with higher associated value of $\delta^{(\mathrm{RX})}$, and these are used to define a final set of targets or anomalies. So, in this step, there are some communications and synchronization points among processes. Its complexity is about $O(\frac{rows \cdot columns}{N_{mpi}})$.

3.1 Automatic configuration for heterogeneous clusters of multi-processors

We consider hybrid clusters composed of N nodes with different number of processors defined by the array $Cores[i]$ with $0 \leq i < N$. Moreover, for the parallel RX algorithm, the number of MPI processes (N_{mpi}) determines the number of local sub-images, then in general $N_{mpi} \geq N$. We have developed a external routine called `init_mapping`, which is used as a preprocessing stage to evaluate the resources of a specific cluster of multi-core nodes, and to establish the mapping of the MPI processes. Then the parallel RX can be executed automatically with an efficient configuration for the cluster. Thus, `init_mapping` takes the following input parameters: (1) the total number of MPI processes, N_{mpi}; (2) the number of cores of the ith node provided by the operating system [2], which is denoted by $Cores[i]$. The output of `init_mapping` is: (1) the number of MPI processes to be mapped on ith node, called $MPI[i]$; and (2) the number of threads for every MPI process on the ith node, based on the number of cores of each node. This parameter is denoted by $Threads[i]$.

Let N_{pmi} be the total number of MPI processes, each node executes its initialization step as a MPI process, with the following algorithm:

`init_mapping` (N_{mpi}, i = *identifier node*)
1 *Initialize* $Cores[i]$ *provided by operative system*
2 *Interchange* $Cores[i]$ *between all nodes and compute* $TCores = \sum Cores[i]$
3 $Th_{av} = \frac{TCores}{N_{mpi}}$ *initial hypothesis* $Threads[i] = Th_{av}$ *for all MPI processes*
4 *Assign one MPI process to every node*
5 *Cyclic distribution of the remainder MPI processes*
(*bearing in mind the values of* Th_{av} *and the no assigned cores*)
6 $Threads[i] = \lfloor \frac{Cores[i]}{MPI[i]} \rfloor$ *final tuning of* $Threads[i]$
7 *Return* $MPI[i]$, $Threads[i]$

The information generated by `init_mapping` is the key to fit a parallel program on a specific heterogeneous cluster. Then the RX program can use the output of `init_mapping` to setup the parallel execution that will use as many cores as possible, according to the restrictions of the cluster architecture and the value of N_{mpi}.

4 Validation and evaluation of parallel RX

In order to evaluate the numerical effectiveness of our parallel implementation of the RX algorithm, we consider a hyperspectral data set, that was collected by the AVIRIS

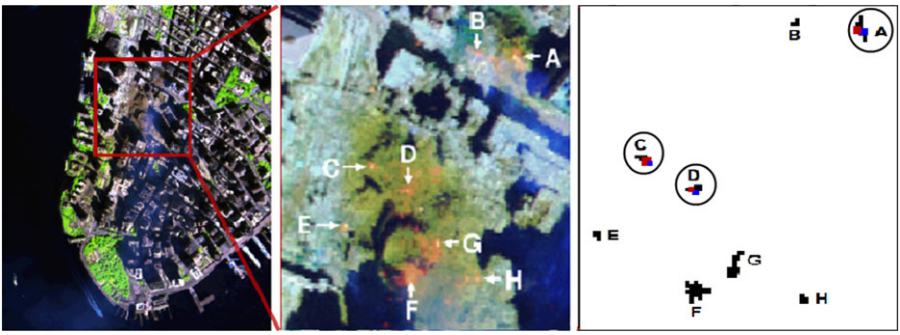

Fig. 1 False color composite of a hyperspectral image over the WTC in New York (*left*). Location of thermal hot spots (*middle*). Detection results with *red* (*blue*) pixels indicating targets detected by the serial (*parallel*) version (*right*)

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. The full data set selected for experiments consists of 614 × 512 pixels, 224 spectral bands and a total size of (approximately) 538 MB. The leftmost part of Fig. 1 shows a false color composite of the data set selected for experiments. Extensive reference information, collected by US Geological Survey (USGS), is available for the WTC scene.[1] In this work, we use a US Geological Survey thermal map[2] which shows the locations of the thermal hot spots (which can be seen as anomalies) at the WTC area, displayed as bright red, orange, and yellow spots at the center part of Fig. 1; the map is centered at the region where the towers collapsed. The thermal map displayed in the center part of Fig. 1 will be used in this work as ground-truth to validate the target detection accuracy of the proposed parallel algorithms and their respective serial versions.

The accuracy (numerical effectiveness) of the algorithm is evaluated by its capability to automatically detect the thermal hot spot of fires (anomalies) in the WTC area. The evaluation of RX is based on the results shown on the rightmost part of Fig. 1. For illustrative purposes, Fig. 1 shows the detection results obtained by the serial and the parallel implementation of RX using different number of MPI processes ($N_{mpi} = 4$ and 16). In all cases, the number of target pixels to be detected was set to 30 after calculating the virtual dimensionality of the data [3]. As shown by Fig. 1, the same targets were detected by the parallel version regardless of the number of partitions. In this case, a local approach is used for calculating the covariance matrix, while the serial version uses a global approach. However, the local strategy exhibits results which are almost identical to those obtained by the global strategy, reducing the communication between MPI processes. In all cases, only three out of eight targets (i.e. those labeled as 'A', 'C', and 'D') were detected.

The parallel computing platform used for the performance evaluation of the parallel RX algorithm is a dedicated cluster composed by: (1) Node A: DELL PowerEdge R810, based on Intel Xeon L7555 1.87 GHz (8 cores), with 16 GB of main memory;

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

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

Table 1 Run-time (seconds) for Hybrid Parallel RX Algorithm on the test heterogeneous cluster

Node	N_{mpi}	*MPI*	*NThs*	*Step*1	*Step*3	*Total*	*C*	*TCores*	*I_InSp*	*InSp*
A	2	1	8	3.5	1.9	5.7	8	16	8	9.4
B		1	8	3.1	1.6		8			
A		*1*	*1*	*38.8*	*14.5*	*53.4*	1	*2*		
B		*1*	*1*	*11.4*	*9.8*		1			
A	4	2	4	3.7	1.9	6.2	8	16	4	4.2
B		2	4	3.2	1.5		8			
A		*2*	*1*	*18.4*	*7.1*	*26.2*	2	*4*		
B		*2*	*1*	*6.5*	*4.9*		2			
A	8	4	2	4.2	1.9	6.3	8	16	2	2.3
B		4	2	3.4	1.4		8			
A		*4*	*1*	*9.1*	*3.8*	*14.6*	4	*8*		
B		*4*	*1*	*5.3*	*2.7*		4			
A	16	8	1	4.4	1.8	7.9	8	16	1	1
B		8	1	3.6	2.1		8			
A	3	2	4	4.7	2.6	7.4	8	12	4	4.6
B′		1	4	2.4	1.7		4			
A		*2*	*1*	*23.6*	*9.9*	*33.8*	2	*3*		
B′		*1*	*1*	*8.5*	*7.1*		1			
A	6	4	2	5.6	2.5	8.4	8	12	2	2.1
B′		2	2	2.2	1.7		4			
A		*4*	*1*	*12.2*	*4.9*	*17.6*	4	*6*		
B′		*2*	*1*	*3.8*	*3.3*		2			

and (2) Node B: based on 2 Intel Xeon E5640 2.66 GHz (8 cores), with 12 GB of main memory. The operating system used at the time of experiments was Linux Debian in both nodes, and OpenMPI and POSIX Threads were used as parallel interface programming. Additionally, we have considered the case that the Node B has only 4 available cores (denoted by B′).

Table 1 summarizes the main measurements experimentally obtained in order to evaluate the performance of our parallel algorithm. So, for different configurations of the test cluster, we can analyze the following details about the parallel RX algorithm: (1) the profile, that is, the run-time of every step, in the columns labeled *Step*∗; (2) the total run-time (column labeled as *Total*) for two MPI implementations: the pure MPI version, one thread per MPI process, and the hybrid version with maximum number of threads per MPI process; (3) the ratio of run-times for pure and hybrid MPI versions, denoted as $InSp = Total_{MPI}/Total_H$ (hereinafter the labels *MPI* and *H* are refereed to the both kinds of configurations evaluated, pure MPI and hybrid, respectively). This parameter is introduced because the speed-up is not an appropriate parameter to evaluate the scalability of parallel RX, since the sequential RX is penalized

by a hard memory management. If the performance of the cores of the nodes were similar, then the ideal value of *InSp* would be near to the ratio $TCores_H/TCores_{MPI}$, its value is shown in the column *I_InSp*. Additionally, Table 1 shows the number of MPI processes and Threads of every node (columns *MPI* and *NThs*), the effective number of cores used per node (column $C = Cores[*]$) and the total cores used (column *TCores*).

*Step*2 related to the computation of K^{-1} matrix is not shown in Table 1 because its run-time is less than 1 second, since, the multi-threaded ATLAS library used in this step achieves a very high performance. Run-times for *Step*4 are negligible compared to *Step*1 and *Step*3.

As we can see on Table 1, *Step*1 is the slowest. It computes the covariance matrix which realizes a lot of readings of hyperspectral image. So, the memory management in this algorithm has a strong impact on the performance of *Step*1. Moreover, despite the nodes are composed of cores with similar architecture, the memory hierarchy is different for each node and the performances are different when the size of the data are larger, that is, for smaller values of N_{mpi}. *Step*3 (with the same computational complexity that *Step*1) achieves similar run-times on both nodes, probably because it includes less readings of the hyperspectral image from main memory.

Moreover, Table 1 (column labeled as *Total*) shows the parallel RX total run-times for several configurations of the heterogeneous cluster. These results show the performance improvements achieved by our hybrid parallel RX. So, our parallel approach allows us to exploit better two levels of parallelism on the current clusters, taking into a count the restrictions in the number of MPI processes for the RX algorithm. Finally, Table 1 (right) shows the parameter *InSp* which measures the acceleration obtained by our hybrid parallel version over the pure MPI execution. The experimental values of *InSp* are lightly over the ideal values *I_InSp*. This illustrates the good performance of our implementation and the improvements of memory management achieved by our hybrid parallel RX.

5 Conclusions and future research lines

In this paper, our goal has been to accelerate the RX algorithm to detect anomalies in the context of a real hyperspectral imaging application. We have proposed a new parallel implementation of the RX algorithm exploiting the current clusters of multicore nodes. Additionally, a routine to configure automatically parallel hybrid MPI programs for modern clusters has been devised. The proposed parallel RX is based on: (1) spatial domain partitioning at level of MPI interface reducing the interprocessor communications; and (2) the selection of particular loops to be multi-threaded. The parallel version has been validated in the context of a real hyperspectral imaging application. Our experimental results indicate that the high performance of the hybrid parallel RX is based on the exploitation of both levels of parallelism supplied by the current clusters with nodes composed by cores with similar performance. Two lines are in mind for the future works: (1) the development of hybrid parallel RX for nodes based on cores with different architectures and (2) the extension of the conducted evaluation to additional hyperspectral scenes in the context of different analysis scenarios.

Acknowledgements This work has been supported by the European Community's Marie Curie Research Training Networks Programme under reference MRTN-CT-2006-035927 (HYPER-I-NET), by Spanish Ministry of Science and Innovation TIN2008-01117, (HYPERCOMP/EODIX project, reference AYA2008-05965-C04-02), and by Junta de Andalucía (P08-TIC-3518, P10-TIC-6002).

References

1. Automatically tuned linear algebra software (ATLAS) (2011) http://math-atlas.sourceforge.net
2. Bovet D, Cesati M (2002) Understanding the Linux kernel, 2nd edn. O'Reilly & Associates, Sebastopol
3. Chang C-I (2003) Hyperspectral imaging: techniques for spectral detection and classification. Kluwer, Norwell
4. Goetz AFH, Vane G, Solomon JE, Rock BN (1985) Imaging spectrometry for Earth remote sensing. Science 228:1147–1153
5. Green RO et al (1998) Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). Remote Sens Environ 65(3):227–248
6. Paz A, Plaza A (2010) GPU implementation of target and anomaly detection algorithms for remotely sensed hyperspectral image analysis. In: Satellite data compression, communications, and processing, vol 7810. SPIE Press, Bellingham
7. Paz A, Plaza A, Blazquez S (2008) Parallel implementation of target and anomaly detection algorithms for hyperspectral imagery. In: Proc IEEE geosci remote sens symp, vol 2, pp 589–592
8. Paz A, Plaza A, Plaza J (2009) Comparative analysis of different implementations of a parallel algorithm for automatic target detection and classification of hyperspectral images. Proc SPIE 7455:1–12
9. Plaza A, Chang C-I (2007) High performance computing in remote sensing. CRC Press, Boca Raton
10. Plaza A, Valencia D, Plaza J, Martinez P (2006) Commodity cluster-based parallel processing of hyperspectral imagery. J Parallel Distrib Comput 66:345–358
11. Plaza A et al (2009) Recent advances in techniques for hyperspectral image processing. Remote Sens Environ 113:110–122
12. Reed IS, Yu X (1990) Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution. IEEE Trans Acoust Speech Signal Process 38:1760–1770
13. Richards JA, Jia X (2006) Remote sensing digital image analysis: an introduction. Springer, Berlin