Comparative Analysis of Different Implementations of a Parallel Algorithm for Automatic Target Detection and Classification of Hyperspectral Images

Abel Paz[a], Antonio Plaza[a] and Javier Plaza[a]

[a]Department of Technology of Computers and Communications, University of Extremadura, Avda. de la Universidad s/n CP. 10071 Caceres, Spain

ABSTRACT

Automatic target detection in hyperspectral images is a task that has attracted a lot of attention recently. In the last few years, several algoritms have been developed for this purpose, including the well-known RX algorithm for anomaly detection, or the automatic target detection and classification algorithm (ATDCA), which uses an orthogonal subspace projection (OSP) approach to extract a set of spectrally distinct targets automatically from the input hyperspectral data. Depending on the complexity and dimensionality of the analyzed image scene, the target/anomaly detection process may be computationally very expensive, a fact that limits the possibility of utilizing this process in time-critical applications. In this paper, we develop computationally efficient parallel versions of both the RX and ATDCA algorithms for near real-time exploitation of these algorithms. In the case of ATGP, we use several distance metrics in addition to the OSP approach. The parallel versions are quantitatively compared in terms of target detection accuracy, using hyperspectral data collected by NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) over the World Trade Center in New York, five days after the terrorist attack of September 11th, 2001, and also in terms of parallel performance, using a massively Beowulf cluster available at NASA's Goddard Space Flight Center in Maryland.

Keywords: Hyperspectral data, anomaly detection, target detection, parallel processing, spectral distances.

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. Hyperspectral imaging instruments such as the NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS)[2] 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 2 to 12 kilometers wide and several kilometers long using 224 spectral bands.[1] The resulting "image cube" (see 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. The resulting data volume typically comprises several GBs per flight.

Automatic target and anomaly detection are important tasks for hyperspectral data exploitation. During the last few years, several algorithms have been developed for the aforementioned purposes, including the automatic target detection and classification (ATDCA) algorithm,[3] an unsupervised fully-constrained least squares (UFCLS) algorithm,[4] an iterative error analysis (IEA) algorithm,[5] or the well-known RX algorithm developed by Reed and Xiaoli for anomaly detection.[6] The ATDCA algorithm finds a set of spectrally distinct target pixels vectors using the concept of orthogonal subspace projection (OSP)[7] in the spectral domain. On the other hand, the UFCLS algorithm generates a set of distinct targets using the concept of least square-based error minimization. The IEA uses a similar approach, but with a different initialization condition. The RX algorithm is based on the application of a so-called RXD filter, given by the well-known Mahalanobis distance. Many other target/anomaly detection algorithms have also been proposed in the recent literature,[8].[9]

Depending on the complexity and dimensionality of the input scene, the aforementioned algorithms may be computationally very expensive, a fact that limits the possibility of utilizing those algorithms in time-critical

Send correspondence to Antonio J. Plaza:
E-mail: aplaza@unex.es; Telephone: +34 927 257000 (Ext. 51662); URL: http://www.umbc.edu/rssipl/people/aplaza

Satellite Data Compression, Communication, and Processing V, edited by Bormin Huang, Antonio J. Plaza, Raffaele Vitulli,
Proc. of SPIE Vol. 7455, 74550X · © 2009 SPIE · CCC code: 0277-786X/09/$18 · doi: 10.1117/12.825458

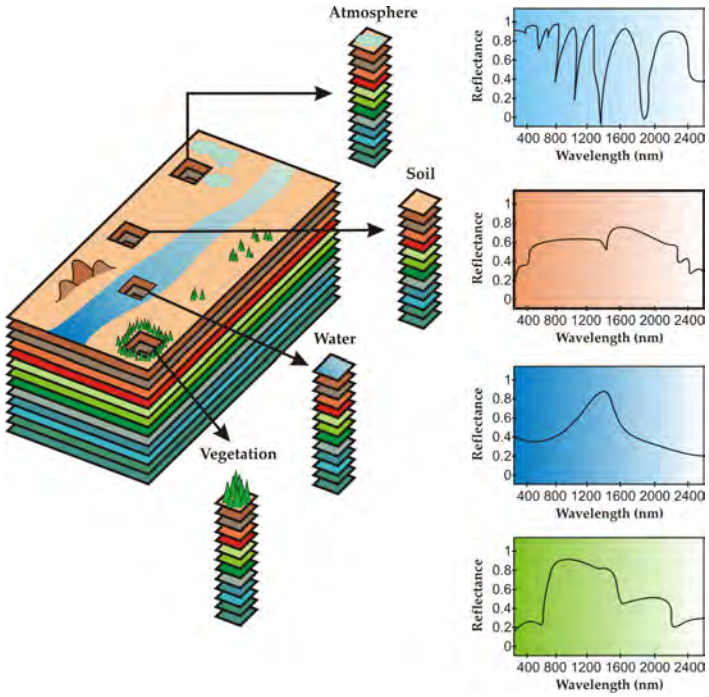

Figure 1. Concept of hyperspectral imaging.

applications. In turn, the wealth of spectral information available in hyperspectral imaging data opens groundbreaking perspectives in many applications, including target detection for military and defense/security deployment. In particular, algorithms for detecting (moving or static) targets often require timely responses for swift decisions, which depend upon high computing performance of algorithm analysis. Therefore, it is of great importance that automatic target and anomaly detection algorithms can complete their analysis tasks quickly enough for practical use. Despite the growing interest in parallel hyperspectral imaging research[10, 11] only a few parallel implementations of automatic target and anomaly detection algorithms for hyperspectral data exist in the open literature.[12] However, with the recent explosion in the amount and dimensionality of hyperspectral imagery, parallel processing is expected to become a requirement in most remote sensing missions, including those related with the detection of anomalous and/or concealed targets. Of particular importance is the design of parallel algorithms able to detect target and anomalies at sub-pixel levels, thus overcoming the limitations imposed by the spatial resolution of the imaging instrument (often inversely related to its spectral resolution).

In this paper, we develop computationally efficient parallel versions of two highly representative algorithms for target (ATDCA) and anomaly detection (RX) in hyperspectral scenes. In the case of ATDCA, we use several distance metrics in addition to the OSP approach. The considered metrics include the spectral angle distance (SAD) and the spectral information divergence (SID), which introduce an innovation with regards to the originally proposed algorithm. The parallel versions of RX and ATDCA are quantitatively assessed in terms of target detection accuracy and parallel performance in a case study focused on identifying sub-pixel thermal hot spots (which can be seen as targets and/or anomalies) in a complex urban background, using AVIRIS hyperspectral data collected over the World Trade Center in New York, five days after the terrorist attack of September 11th, 2001. The parallel performance of the proposed implementations is assessed using a Beowulf cluster with 256 processors available at NASA's Goddard Space Flight Center in Maryland.

The remainder of the paper is organized as follows. Section 2 describes the considered anomaly (RX) and target detection (ATDCA) algorithms. In the latter case, we also describe the different metrics considered for implementing this algorithm. Section 3 develops the proposed parallel implementations, called P-RX and P-ATDCA, respectively. Section 4 first describes the hyperspectral data set used for experiments, and then (in section 5) presents experimental results in terms of both target/anomaly detection accuracy and parallel performance on a Beowulf cluster. Finally, section 6 concludes with some remarks and hints at future research.

2. METHODS

In this section we briefly describe the target detection algorithms that will be efficiently implemented in parallel in this work. These algorithms are the RX for anomaly detection and the ATDCA for automatic target detection and classification. In the latter case, several distance measures are described for implementation of the algorithm.

2.1 RX algorithm

The RX algorithm has been widely used in signal and image processing.[6] The filter implemented by this algorithm is referred to as RX filter (RXF) and defined by the following expression:

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

where $\mathbf{x} = \left[x^{(0)}, x^{(1)}, \cdots, x^{(n)}\right]$ is a sample, n-dimensional hyperspectral pixel (vector), μ is the sample mean and $\mathbf{K}$ is the sample data covariance matrix. As we can see, the form of δ^{RXF} is actually the well-known Mahalanobis distance. It is important to note that the images generated by the RX algorithm are generally gray scale images. In this case, the number of anomalies detected by the RX algorithm is determined by a prescribed value used to threshold the detected image. This issue was previously investigated.[12] In our experiments, the threshold values were found by the automatic Otsu thresholding method.[12]

2.2 ATDCA algorithm

The ATDCA algorithm[3] was developed to find potential target pixels that can be used to generate a signature matrix used in an orthogonal subspace projection (OSP) approach.[7] Let $\mathbf{x}_0$ be an initial target signature (i.e., the pixel vector with maximum length). The ATDCA begins by an orthogonal subspace projector specified by the following expression:

$$P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T, \tag{2}$$

which is applied to all image pixels, with $\mathbf{U} = [\mathbf{x}_0]$. It then finds a target signature, denoted by $\mathbf{x}_1$, with the maximum projection in $< \mathbf{x}_0 >^{\perp}$, which is the orthogonal complement space linearly spanned by $\mathbf{x}_0$. A second target signature $\mathbf{x}_2$ can then be found by applying another orthogonal subspace projector $P_{\mathbf{U}}^{\perp}$ with $\mathbf{U} = [\mathbf{x}_0, \mathbf{x}_1]$ to the original image, where the target signature that has the maximum orthogonal projection in $< \mathbf{x}_0, \mathbf{x}_1 >^{\perp}$ is selected as $\mathbf{x}_2$. The above procedure is repeated until a set of target pixels $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_t\}$ is extracted, where t is an input parameter to the algorithm.

In addition to the standard OSP approach, we have explored other alternatives in the implementation of ATDCA, given by replacing the $P_{\mathbf{U}}^{\perp}$ operator used in the OSP implementation by one of the distance measures described below:

- The 1-Norm between two pixel vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, defined by $\|\mathbf{x}_i - \mathbf{x}_j\|$.
- The 2-Norm between two pixel vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, defined by $\|\mathbf{x}_i - \mathbf{x}_j\|_2$.
- The Infinity-Norm between two pixel vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, defined by $\|\mathbf{x}_i - \mathbf{x}_j\|_\infty$.
- The spectral angle distance (SAD) between two pixel vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, defined by the following expression:[1] $\mathrm{SAD}(\mathbf{x}_i, \mathbf{x}_j) = \cos^{-1}\left(\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \cdot \|\mathbf{x}_j\|_2}\right)$. As opposed to the previous metric, SAD is invariant in the presence of illumination interferers, which can provide advantages in terms of target and anomaly detection in complex backgrounds.
- Finally, we also use the spectral information divergence (SID) between two pixel vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, defined by the following expression:[1] $\mathrm{SID}(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{D}(\mathbf{x}_i||\mathbf{x}_j) + \mathrm{D}(\mathbf{x}_j||\mathbf{x}_i)$, where $\mathrm{D}(\mathbf{x}_i||\mathbf{x}_j) = \sum_{k=1}^{n} p_k \cdot \log(p_k/q_k)$. Here, we define $p_k = x_i^{(k)} / \sum_{k=1}^{n} x_i^{(k)}$ and $q_k = x_j^{(k)} / \sum_{k=1}^{n} x_j^{(k)}$.

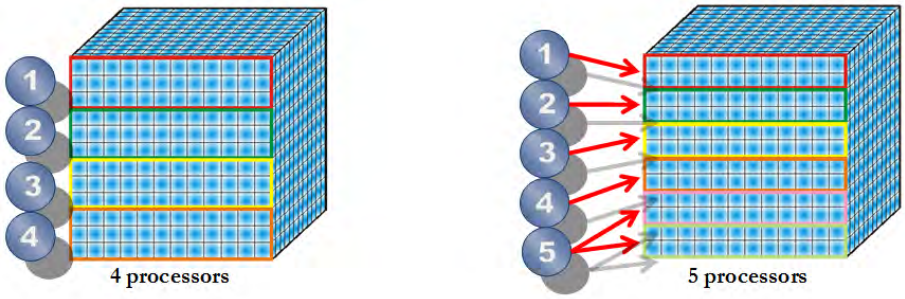

Figure 2. Spatial-domain decomposition of a hyperspectral data set.

3. PARALLEL IMPLEMENTATIONS

In all considered parallel algorithms, a data-driven partitioning strategy has been adopted as a baseline for algorithm parallelization. Specifically, two approaches for data partitioning have been tested:[13]

- *Spectral-domain partitioning*. This approach subdivides the multi-channel remotely sensed image into small cells or sub-volumes made up of contiguous spectral wavelengths for parallel processing.
- *Spatial-domain partitioning*. This approach breaks the multi-channel image into slices made up of one or several contiguous spectral bands for parallel processing. In this case, the same pixel vector is always entirely assigned to a single processor, and slabs of spatially adjacent pixel vectors are distributed among the processing nodes (CPUs) of the parallel system. Fig. 2 shows two examples of spatial-domain partitioning over 4 processors and over 5 processors, respectively.

Previous experimentation with the above-mentioned strategies indicated that spatial-domain partitioning can significantly reduce inter-processor communication, resulting from the fact that a single pixel vector is never partitioned and communications are not needed at the pixel level.[13] In the following, we assume that spatial-domain decomposition is always used when partitioning the hyperspectral data cube. The inputs to the considered parallel algorithms are a hyperspectral image cube $\mathbf{F}$ with n dimensions, where $\mathbf{x}$ denotes the pixel vector of the same scene, and a maximum number of targets to be detected, t. The output in all cases is a set of target pixel vectors $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_t\}$. Before describing the parallel algorithms, we provide additional details about the data partitioning approach adopted in their implementation.

3.1 P-RX

Our parallel version of the RX algorithm for anomaly detection adopts the spatial-domain decomposition strategy depicted in Fig. 2 for dividing the hyperspectral data cube in master-slave fashion. The parallel algorithm is given by the following steps:

1. The master processor divides the original image cube $\mathbf{F}$ into P spatial-domain partitions and distributes them among the workers.
2. The master calculates the n-dimensional mean vector $\mathbf{m}$ concurrently, where each component is the average of the pixel values of each spectral band of the unique set. This vector is formed once all the processors finish their parts. At the same time, the master also calculates the sample spectral covariance matrix $\mathbf{K}$ concurrently as the average of all the individual matrices produced by the workers using their respective portions.
3. Using the above information, each worker applies (locally) the RXF filter given by the Mahalanobis distance to all the pixel vectors in the local partition as follows: $\delta^{(\mathrm{RXF})}(\mathbf{x}) = (\mathbf{x}-\mathbf{m})^T\mathbf{K}^{-1}(\mathbf{x}-\mathbf{m})$, and returns the local result to the master.
4. The master now selects the t pixel vectors with higher associated value of $\delta^{(\mathrm{RXF})}$, and uses them to form a final set of targets $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}$.

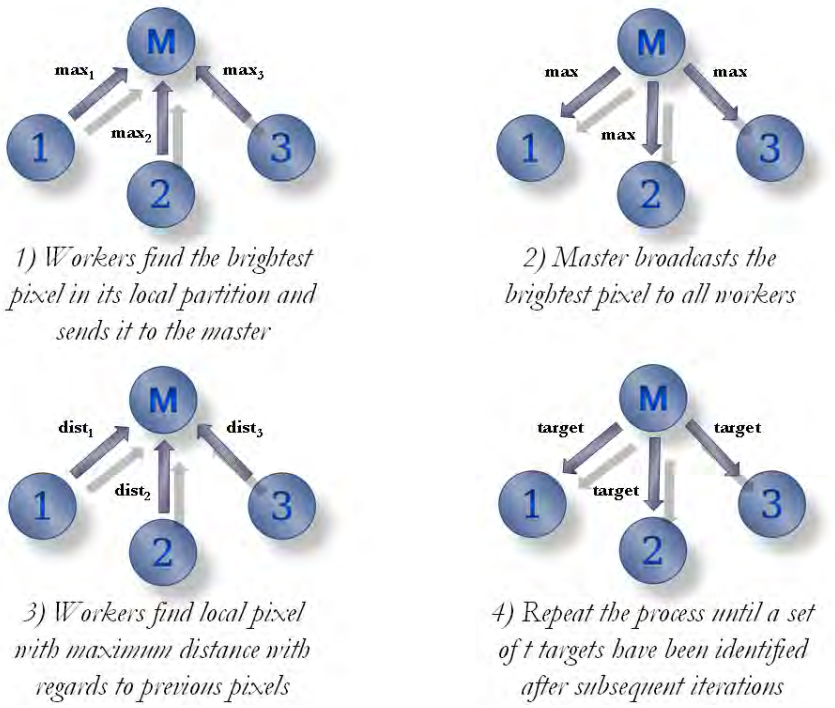

Figure 3. Graphical summary of the parallel implementation of ATDCA algorithm using 1 master processor and 3 slaves.

3.2 P-ATDCA

The parallel version of ATDCA also uses the spatial-domain decomposition strategy depicted in Fig. 2. The implementation of this algorithm is graphically summarized in Fig. 3, and consists of the following steps:

1. The master divides the original image cube $\mathbf{F}$ into P spatial-domain partitions. Then, the master sends the partitions to the workers.

2. Each worker finds the brightest pixel in its local partition using $\mathbf{x}_1 = argmax\{\mathbf{x}^T \cdot \mathbf{x}\}$, where the superscript T denotes the vector transpose operation. Each worker then sends the spatial locations of the pixel identified as the brightest one in its local partition back to the master.

3. Once all the workers have completed their parts, the master finds the brightest pixel of the input scene, $\mathbf{x}_1$, by applying the $argmax$ operator in step 2 to all the pixels at the spatial locations provided by the workers, and selecting the one that results in the maximum score. Then, the master sets $\mathbf{U} = \mathbf{x}_1$ and broadcasts this matrix to all workers.

4. Each worker finds (in parallel) the pixel in its local partition with the maximum orthogonal projection relative to the pixel vectors in $\mathbf{U}$, using a projector given by $P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$, where $\mathbf{U}$ is the identity matrix. The orthogonal space projector $P_{\mathbf{U}}^{\perp}$ is now applied to all pixel vectors in each local partition to identify the most distinct pixels (in orthogonal sense) with regards to the previously detected ones. Each worker then sends the spatial location of the resulting local pixels to the master node.

5. The master now finds a second target pixel by applying the $P_{\mathbf{U}}^{\perp}$ operator to the pixel vectors at the spatial locations provided by the workers, and selecting the one which results in the maximum score as follows $\mathbf{x}_2 = argmax\{(P_{\mathbf{U}}^{\perp}\mathbf{x})^T(P_{\mathbf{U}}^{\perp}\mathbf{x})\}$. The master sets $\mathbf{U} = \{\mathbf{x}_1, \mathbf{x}_2\}$ and broadcasts this matrix to all workers.

6. Repeat from step 4 until a set of t target pixels, $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}$, are extracted from the input data. It should be noted that the P-ATDCA algorithm has not only been implemented using the aforementioned OSP-based approach, but also the different metrics discussed in section 2.2 by simply replacing the $P_{\mathbf{U}}^{\perp}$ operator by a different distance measure.

Figure 4. False color composition of an AVIRIS hyperspectral image collected by NASA's Jet Propulsion Laboratory over lower Manhattan on Sept. 16, 2001 (left). Location of thermal hot spots in the fires observed in World Trade Center area, available online: http://pubs.usgs.gov/of/2001/ofr-01-0429/hotspot.key.tgif.gif (right).

Table 1. Properties of the thermal hot spots reported in Fig. 4(right).

Hot spot	Latitude (North)	Longiture (West)	Temperature (Kelvin)	Area (Square meters)
'A'	40°42'47.18"	74°00'41.43"	1000	0.56
'B'	40°42'47.14"	74°00'43.53"	830	0.08
'C'	40°42'42.89"	74°00'48.88"	900	0.80
'D'	40°42'41.99"	74°00'46.94"	790	0.80
'E'	40°42'40.58"	74°00'50.15"	710	0.40
'F'	40°42'38.74"	74°00'46.70"	700	0.40
'G'	40°42'39.94"	74°00'45.37"	1020	0.04
'H'	40°42'38.60"	74°00'43.51"	820	0.08

4. EXPERIMENTAL RESULTS

4.1 Data description

The image scene used for experiments in this work 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. Fig. 4(left) 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 4(left), 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.

In this work, we use a U.S. Geological Survey thermal map* which shows the target locations of the thermal hot spots at the WTC area, displayed as bright red, orange and yellow spots in Fig. 4(right). The map is centered at the region where the towers collapsed, and the temperatures of the targets range from 700F to 1300F. Further information about the targets (including location, size and temperature) is reported on Table 1. As shown by Table 1, all the targets are sub-pixel in size since the spatial resolution of a single pixel is 1.7 square meters. The thermal map in Fig. 4(right) will be used in this work as ground-truth to validate the target detection accuracy of the proposed parallel algorithms.

*Available online: http://pubs.usgs.gov/of/2001/ofr-01-0429/hotspot.key.tgif.gif

	A	B	C	D	E	F	G	H
P-ATDCA (OSP)	0,16232	0,24413	0,00000	0,00000	0,35271	0,49844	0,37602	0,38034
P-ATDCA (1-Norm)	0,16232	0,42486	0,00000	0,28923	0,66099	0,75851	0,67635	0,62394
P-ATDCA (2-Norm)	0,16232	0,42486	0,00000	0,28923	0,66099	0,75851	0,67635	0,45081
P-ATDCA (∞-Norm)	0,15979	0,39329	0,00000	0,24984	0,48334	0,54754	0,37919	0,46425
P-ATDCA (SAD)	0,16232	0,39744	0,00000	0,25746	0,67016	0,56650	0,44422	0,52727
P-ATDCA (SID)	0,16232	0,42486	0,00000	0,28923	0,69115	0,56143	0,40070	0,35451
P-RX	0,00000	0,21537	0,00000	0,00000	0,33051	0,49773	0,59596	0,71031

Table 2. Spectral similarity between target pixels and known ground targets for P-RX and P-ATDCA.

4.2 Parallel computing platform

The parallel computing architecture used in experiments is the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center (NASA/GSFC). From the early nineties, the overwhelming computational needs of Earth and space scientists have driven NASA/GSFC to be one of the leaders in the application of low cost high-performance computing.[14] Up until 1997, the commodity clusters at NASA/GSFC were in essence engineering prototypes, that is, they were built by those who were going to use them.

In spring of 1997 the Highly Parallel Virtual Environment (HIVE) project was started to build a commodity cluster intended to be exploited by different users in a wide range of scientific applications. The idea was to have workstations distributed among many offices and a large number of compute nodes (the compute core) concentrated in one area. The workstations would share the compute core as though it was a part of each. The HIVE was the first commodity cluster to exceed a sustained 10 Gflops on an algorithm. The Thunderhead system can be seen as an evolution of the HIVE project. It is composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 Gb of memory and 80 Gb of main memory. The total peak performance of the system is 2457.6 Gflops. Along with the 512-processor computer core, Thunderhead has several nodes attached to the core with 2 GHz optical fibre Myrinet. The parallel algorithms tested in this work were run from one of such nodes, called thunder1. The operating system used at the time of experiments was Linux RedHat 8.0, and MPICH was the message-passing library used.

4.3 Quantitative and comparative assessment in terms of target detection accuracy

For illustrative purposes, Table 2 shows the spectral angle distance (SAD) between the most similar target pixels detected by P-RX and P-ATDCA (implemented using different distance metrics) and the pixel vectors at the known target positions, labeled from 'A' to 'H' in Fig. 4(right). The lower the SAD score, the more similar the pixel vectors are. In all cases, the number of target pixels to be detected was set to $t = 30$ after calculating the virtual dimensionality of the data.[1] As shown by Table 2, both the P-ATDCA and P-RX extracted targets were very similar, spectrally, to the known ground-truth targets. The P-RX was able to perfectly detect the targets labeled as 'A', 'C' and 'D' (all of them relatively large in size and with high temperature), while the P-ATDCA implemented using OSP was able to perfectly detect the targets labeled as 'C' and 'D'. Both the P-RX and P-ATDCA had more difficulties in detecting very small targets

In the case of the P-ATDCA algorithm implemented with a distance measure other than OSP we realized that, in many cases, some of the target pixels obtained were repeated. To solve this issue, we have developed a method called relaxed pixel method (RPM) which simply removes a detected target pixel from the scene so that it cannot be selected in subsequent iterations. Table 2 shows the SAD between the most similar target pixels detected by P-ATDCA (implemented using the aforementioned RPM strategy) and the pixel vectors at the known target positions. As shown by Table 2, most measured SAD-based scores are lower when the RPM strategy is used, in particular, for targets of moderate size such as 'A', 'E' or 'F'. The detection results were also slightly improved for the target with highest temperature, i.e. the one labeled as 'G'. This indicated that the proposed RPM strategy can improve the detection results despite its simplicity.

Finally, Table 4 shows a summary of the detection results obtained by the P-RX and P-ATDCA (with and without RPM strategy). It should be noted that the RPM strategy was not necessary when the P-ATDCA

	A	B	C	D	E	F	G	H
P-ATDCA (1-Norm)	0,00000	0,21537	0,00000	0,19497	0,39928	0,55798	0,60994	0,54323
P-ATDCA (2-Norm)	0,00000	0,26788	0,00000	0,19497	0,48334	0,55798	0,61466	0,45081
P-ATDCA (∞-Norm)	0,15979	0,26788	0,00000	0,24984	0,45223	0,54647	0,35078	0,46425
P-ATDCA (SAD)	0,00000	0,26788	0,00000	0,20132	0,52119	0,49773	0,40676	0,52727
P-ATDCA (SID)	0,00000	0,30477	0,00000	0,19497	0,54579	0,49773	0,39520	0,38821

Table 3. Spectral similarity between target pixels and known ground targets for P-ATDCA (implemented using RPM)

	Detected	Similar	Detected (RPM)	Similar (RPM)
P-ATDCA (OSP)	C, D	B	–	–
P-ATDCA (1-Norm)	C	A, D	A, C	B, D
P-ATDCA (2-Norm)	C	A, D	A, C	B, D
P-ATDCA (∞-Norm)	C	A, D	C	A, B, D
P-ATDCA (SAD)	C	A, D	A, C	B, D
P-ATDCA (SID)	C	A, D	A, C	D
P-RX	A, C, D	B	–	–

Table 4. Summary of detection results by the P-RX and P-ATDCA algorithms (using different distances), with and without the RPM strategy.

was implemented using OSP. In the table, the column 'detected' lists those targets that were exactly identified (at the same spatial coordinates) with regards to the ground-truth. On the other hand, the column 'similar' lists those targets that were identified with a SAD value below 0.3 (a reasonable spectral similarity threshold according to previous work). As shown by Table 4, the RPM strategy generally improved the results provided by the P-ATDCA algorithm, regardless of the distance measure used. The RPM was not applied to the P-RX algorithm, which generally provided the best detection results in this example (highest number of detected plus similar targets).

4.4 Quantitative and comparative assessment in terms of parallel performance

Table 5 gives processing times in seconds for several multi-processor versions of P-RX and P-ATDCA using different numbers of processors (CPUs) on the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center. In previous work, we have reported the processing times of the parallel version of P-ATDCA using the OSP distance measure,[12] so in this work we only report the times obtained by the parallel algorithm implemented using the remaining distance measures considered in this work. As shown by Table 5, when 32 processors were used the P-ATDCA (implemented using SAD) was able to finalize in about 19 seconds, thus clearly outperforming the sequential version which takes 4 minutes of computation in one Thunderhead processor. In the case of P-RX, we only applied the algorithm to a small portion of the considered data due to communication issues, and two versions were implemented: using communications when needed (communicative) and using independent computations (independent), obtaining similar results in both cases. Here, the processing time using 32 CPUs was only about 4 seconds, while the sequential time measured in one CPU was above one minute. Although these results are encouraging, the proposed parallel implementation of the RX algorithm can still be enhanced by optimizing memory swapping issues in the parallel architecture.

Table 6 reports the speedups (number of times that the parallel version was faster than the sequential one as the number of processors was increased) achieved by multi-processor runs of the P-ATDCA algorithm (implemented using different distances) and P-RX. It can be seen that P-ATDCA scaled better than the two considered versions of P-RX. This has to do with the high number of sequential computations involved in P-RX. For illustrative purposes, the speedups achieved by the different implementations of P-ATDCA and P-RX are graphically illustrated in Figs. 5 and 6, respectively. The speedup plot in Fig. 5 reveals that P-ATDCA scaled better when SID was used as a baseline distance for implementation, resulting in speedups close to linear (although this distance measure resulted in significantly higher processing times, as indicated by Table 5. On the other hand, Fig. 6 reveals that both versions of P-RX resulted in speedup plots that started to flatten from

	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs	32 CPUs
P-ATDCA (1-Norm)	260,43	191,33	97,284	50,002	27,725	19,807
P-ATDCA (2-Norm)	235,78	182,74	94,383	49,42	25,465	19,283
P-ATDCA (∞-Norm)	268,95	187,92	99,288	50,96	27,755	22,001
P-ATDCA (SAD)	241,93	187,83	96,145	49,242	25,352	19,008
P-ATDCA (SID)	2267,6	1148,8	579,51	305,32	165,46	99,375
P-RX (Communicative)	68,868	32,463	16,883	9,1441	5,6728	4,6785
P-RX (Independent)	68,868	32,707	16,826	8,9874	5,4679	4,4264

Table 5. Processing times of P-ATDCA (using different distance measures) and P-RX using different numbers of CPUs on the Thunderhead Beowulf cluster.

	2 CPUs	4 CPUs	8 CPUs	16 CPUs	32 CPUs
P-ATDCA (1-Norm)	1,38651973	2,67700752	5,20839166	9,39332732	13,1483819
P-ATDCA (2-Norm)	1,29024844	2,49811936	4,77094294	9,25898292	12,2273505
P-ATDCA (∞-Norm)	1,43119413	2,70878656	5,27766876	9,69014592	12,2244443
P-ATDCA (SAD)	1,288	2,5163035	4,91308233	9,54283686	12,7277988
P-ATDCA (SID)	1,97388579	3,91296095	7,42696188	13,7048229	22,8186164
P-RX (Communicative)	2,12143055	4,07913286	7,5314137	12,1400367	14,7201026
P-RX (Independent)	2,1056043	4,09295138	7,66272782	12,5949633	15,5584674

Table 6. Speedups for the P-ATDCA algorithm (using different distance measures) and P-RX using different numbers of CPUs on the Thunderhead Beowulf cluster.

linear speedup from 16 CPUs in advance. Again, this is probably due to the fact that memory considerations were not included in the design of this parallel algorithm.

Finally, Table 7 shows the load balancing scores for all considered parallel algorithms. Load balance is defined as $D = Max/Min$, where Max and Min are the maxima and minima processor run times, respectively. Therefore, perfect balance is achieved when $D = 1$. As we can see from Table 7, all the considered parallel algorithms were able to provide values of D very close to optimal in the considered cluster. It is our belief that the (slightly higher) unbalance measured for the P-RX algorithms, in particular, when these are executed on a low number of processors, are due to memory considerations. Our future research will include considerations about including the local memory hierarchy of the individual processors in the data partitioning algorithm.

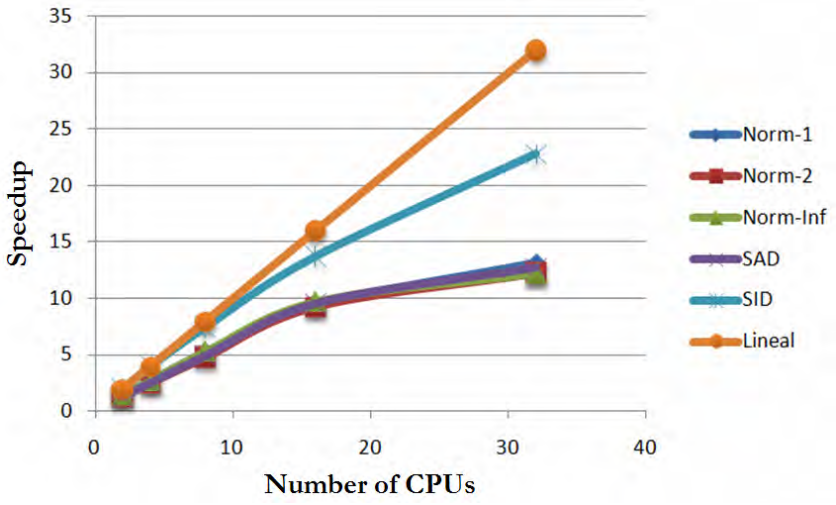

Figure 5. Speedups for the P-ATDCA algorithm (using different distance measures).

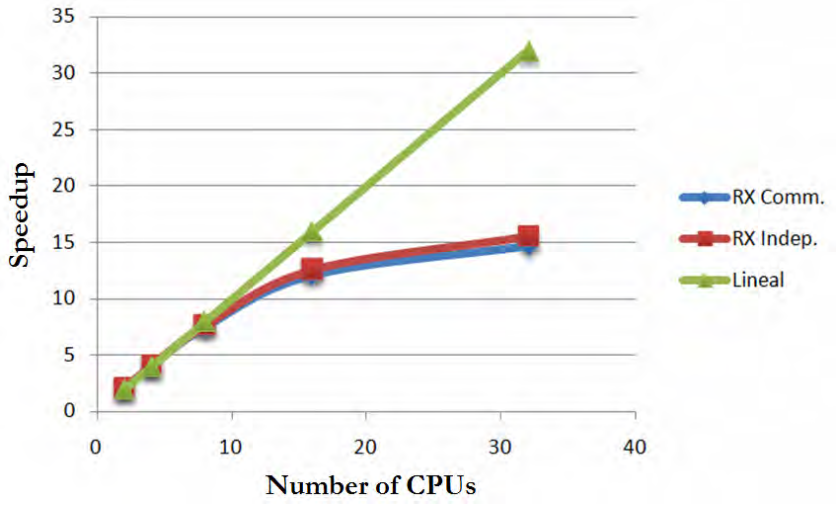

Figure 6. Speedups for the P-RX algorithm (communicative and independent versions).

	Factor	2 CPUs	4 CPUs	8 CPUs	16 CPUs	32 CPUs
P-ATDCA (1-Norm)	Max	191,33	97,284	50,002	27,745	19,818
	Min	191,32	97,274	49,987	27,725	19,807
	D	1,00005227	1,0001028	1,00030008	1,00072137	1,00055536
P-ATDCA (2-Norm)	Max	182,75	94,384	49,429	25,477	19,297
	Min	182,74	94,378	49,419	25,465	19,283
	D	1,00005472	1,00006357	1,00020235	1,00047124	1,00072603
P-ATDCA (∞-Norm)	Max	187,93	99,295	50,97	27,772	22,016
	Min	187,92	99,288	50,96	27,755	22,001
	D	1,00005321	1,0000705	1,00019623	1,0006125	1,00068179
P-ATDCA (SAD)	Max	187,83	96,145	49,242	25,352	19,018
	Min	187,83	96,137	49,237	25,337	19,004
	D	1	1,00008321	1,00010155	1,00059202	1,00073669
P-ATDCA (SID)	Max	1148,8	579,52	305,33	165,47	99,391
	Min	1148,8	579,51	305,32	165,46	99,375
	D	1	1,00001726	1,00003275	1,00006044	1,00016101
P-RX (Communicative)	Max	32,463	16,883	9,1441	5,6728	4,6785
	Min	32,463	16,79	8,9215	5,5006	4,5264
	D	1	1,00553901	1,02495096	1,03130568	1,03360286
P-RX (Independent)	Max	32,707	16,826	8,9874	5,4679	4,4264
	Min	32,479	16,681	8,9574	5,4607	4,4178
	D	1,00701992	1,00869252	1,00334919	1,00131851	1,00194667

Table 7. Load balancing rates for the P-ATDCA (implemented using different distance measures) and P-RX (communicative and independent versions).

5. CONCLUSIONS AND FUTURE RESEARCH

This paper described several innovative parallel algorithms for target detection in hyperspectral data sets. As a case study of specific issues involved in the exploitation of an automatic algorithm for target detection and classification (ATDCA), we have investigated the impact of including several distance measures in the design of the parallel version of that algorithm and compared it with the parallel version of a consolidated anomaly detection algorithm (RX). Our experimental results indicate that commodity cluster computers represent a source of computational power that is both accessible and applicable to obtaining results quickly enough and with high reliability in target/anomaly detection applications. Although the results reported in this work are encouraging, further experiments should be conducted in order to increase the scalability of the proposed parallel algorithms to a higher number of processors by resolving memory issues and optimizing the parallel design of such algorithms. Experiments with additional scenes under different target/anomaly detection scenarios are also highly desirable.

6. ACKNOWLEDGEMENT

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). Funding from the Spanish Ministry of Science and Innovation (HYPERCOMP/EODIX project, reference AYA2008-05965-C04-02) is gratefully acknowledged.

REFERENCES

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

[2] R. O. Green, "Imaging spectroscopy and the airborne visible infrared imaging spectrometer (aviris)," *Remote Sensing of Environment* **65**, pp. 227–248, 1998.

[3] H. Ren and C.-I. Chang, "Automatic spectral target recognition in hyperspectral imagery," *IEEE Trans. Aerosp. Electron. Syst.* **39**, pp. 1232–1249, 2003.

[4] D. Heinz and C.-I. Chang, "Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.* **39**, pp. 529–545, 2001.

[5] R. A. Neville, K. Staenz, T. Szeredi, J. Lefebvre, and P. Hauff, "Automatic endmember extraction from hyperspectral data for mineral exploration," in *21st Canadian Symposium on Remote Sensing*, pp. 401–415, 1999.

[6] I. Reed and X.Yu, "Adaptive multiple-band cfar detection of an optical pattern with unknown spectral distrihution.," *IEEE Trans. Acoustics, Speech and Signal Processing* **38**, pp. 1760–1770, 1990.

[7] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Trans. Geosci. Remote Sens.* **32**, pp. 779–785, 1994.

[8] C.-I. Chang and H. Ren, "An experiment-based quantitative and comparative analysis of hyperspectral target detection and image classification algorithms for hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.* **2**, p. 1044.

[9] D. Manolakis, D. Marden, and G. A. Shaw, "Hyperspectral image processing for automatic target detection applications," *MIT Lincoln Laboratory Journal* **14**, pp. 79–116, 2003.

[10] K. Itoh, "Massively parallel fourier-transform spectral imaging and hyperspectral image processing," *Optics and Laser Technology* **25**, p. 202, 1993.

[11] T. El-Ghazawi, S. Kaewpijit, and J. L. Moigne., "Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality," *Cluster Computing* **1**, pp. 102–110, 2001.

[12] A. Paz, A. Plaza, and S. Blazquez, "Parallel implementation of target and anomaly detection algorithms for hyperspectral imagery," *IEEE Geosci. Remote Sens. Symp.* **2**, pp. 589–592, 2008.

[13] A. Plaza, D. Valencia, J. Plaza, and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *Journal of Parallel and Distributed Computing* **66**, pp. 345–358, 2006.

[14] U. R. J. Dorband, J. Palencia, "Commodity computing clusters at goddard space flight center," *J. Space Commun.* **3**, p. 1, 2003.