Massively Parallel Processing of Remotely Sensed Hyperspectral Images

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

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

ABSTRACT

In this paper, we develop several parallel techniques for hyperspectral image processing that have been specifically designed to be run on massively parallel systems. The techniques developed cover the three relevant areas of hyperspectral image processing: 1) spectral mixture analysis, a popular approach to characterize mixed pixels in hyperspectral data addressed in this work via efficient implementation of a morphological algorithm for automatic identification of pure spectral signatures or *endmembers* from the input data; 2) supervised classification of hyperspectral data using multi-layer perceptron neural networks with back-propagation learning; and 3) automatic target detection in the hyperspectral data using orthogonal subspace projection concepts. The scalability of the proposed parallel techniques is investigated using Barcelona Supercomputing Center's MareNostrum facility, one of the most powerful supercomputers in Europe.

Keywords: Hyperspectral data, endmember extraction, spectral unmixing, supervised classification, neural networks, automatic target detection, massively parallel processing, cluster computing.

1. INTRODUCTION

The development of computationally efficient techniques for transforming massive amounts of remote sensing data into scientific understanding is critical for space-based Earth science and planetary exploration.[1] The wealth of information provided by latest-generation remote sensing instruments has opened ground-breaking perspectives in many applications, including environmental modeling and assessment for Earth-based and atmospheric studies, risk/hazard prevention and response including wild land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination, target detection for military and defense/security purposes, urban planning and management studies, etc.[2]

A relevant example of a remote sensing application in which the use of HPC technologies, such as parallel and distributed computing, are highly desirable is hyperspectral imaging,[3] in which an image spectrometer collects hundreds or even thousands of measurements (at multiple wavelength channels) for the same area on the surface of the Earth. The scenes provided by such sensors are often called 'data cubes' to denote the extremely high dimensionality of the data. For instance, the NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS)[4] is 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 from 2 to 12 kilometers wide and several kilometers long using 224 spectral bands (see Fig. 1). The resulting cube is a stack of images in which each pixel (vector) has an associated spectral signature or 'fingerprint' that uniquely characterizes the underlying objects, and the resulting data volume typically comprises several GBs per flight.

Specifically, the utilization of HPC systems in hyperspectral imaging applications has become more and more widespread in recent years. The idea developed by the computer science community of using COTS (commercial off-the-shelf) computer equipment, clustered together to work as a computational 'team,' is a very attractive solution.[5] This strategy is often referred to as Beowulf-class cluster computing, and has already offered access to greatly increased computational power, but at a low cost (commensurate with falling commercial PC costs) in a number of remote sensing applications.[1]

Send correspondence to Javier Plaza:
E-mail: jplaza@unex.es; Telephone: +34 927 257000 (Ext. 82576)

Satellite Data Compression, Communication, and Processing V, edited by Bormin Huang, Antonio J. Plaza, Raffaele Vitulli,
Proc. of SPIE Vol. 7455, 74550O · © 2009 SPIE · CCC code: 0277-786X/09/$18 · doi: 10.1117/12.825455

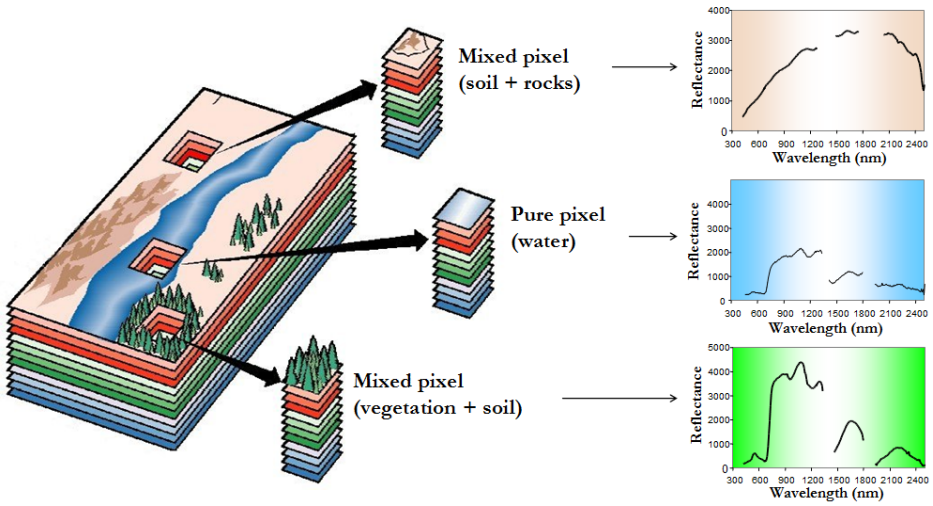

Figure 1. The Concept of Hyperspectral Imaging.

Beowulf clusters were originally developed with the purpose of creating a cost-effective parallel computing system able to satisfy specific computational requirements in the Earth and space sciences community. Initially, the need for large amounts of computation was identified for processing multispectral imagery with only a few bands.[6] As sensor instruments incorporated hyperspectral capabilities, it was soon recognized that computer mainframes and mini-computers could not provide sufficient power for processing this kind of data.

One of the most representative NASA clusters for efficient remote sensing data processing has been the Thunderhead system*, a 512-processor homogeneous Beowulf cluster 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. NASA is currently supporting other massively parallel clusters for remote sensing applications, such as the Columbia supercomputer at NASA Ames Research Center, a 10,240-CPU SGI Altix supercluster, with Intel Itanium 2 processors, 20 terabytes total memory and heterogeneous interconnects including InfiniBand network and 10 gigabit Ethernet. Other massively parallel systems that heve been used in the context remote sensing applications include Jaws, a Dell PowerEdge cluster with 3 GHz Infiniband connectivity, 5,200 GB of main memory and 5,200 processors available at Maui High-Performance Computing Center (MHPCC) in Hawaii, or NEC's Earth Simulator Center, a 5,120-processor system developed by Japan's Aerospace Exploration Agency and the Agency for Marine-Earth Science and Technology. It is highly anticipated that many new supercomputer systems will be specifically developed in forthcoming years to support remote sensing applications.

In this paper, we make use of MareNostrum†, an IBM cluster with 10,240 processors, 2.3 GHz Myrinet connectivity and 20,480 GB of main memory (available at Barcelona Supercomputing Center) in order to demonstrate several new parallel versions of different hyperspectral image processing algorithms. The remainder of the paper is organized as follows. Section 2 describes the hyperspectral processing techniques which are efficiently implemented in this work, covering three relevant areas of hyperspectral image analysis that will be developed in this section (i.e., spectral mixture analysis, supervised classification, and automatic target detection). Section 3 develops computationally efficient versions of representative algorithms and techniques in each considered category. Section 4 first describes the architecture of the MareNostrum supercomputer, and then provides an experimental assessment of the considered parallel algorithms in terms of both analysis accuracy and parallel performance, using highly representative hyperspectral data sets (with reference information) collected by the AVIRIS instrument. Section 5 summarizes our study and provides hints at plausible future research.

*http://thunderhead.gsfc.nasa.gov
†http://www.bsc.es

2. METHODS

2.1 Spectral mixture analysis

Spectral unmixing has been an alluring exploitation goal since the earliest days of imaging spectroscopy.[7] No matter the spatial resolution in natural environments, spectral signatures in hyperspectral data are invariably a mixture of the signatures of the various materials found within the spatial extent of the ground instantaneous field view (see Fig. 1). Due to the high spectral dimensionality of the data, the number of spectral bands usually exceeds the number of spectral mixture components, and the unmixing problem is cast in terms of an over-determined system of equations in which, given a correct set of pure spectral signatures called *endmembers*,[8] the objective is to estimate fractional abundances for those endmembers. Since each observed spectral signal is the result of an actual mixing process, the driving abundances must obey two rather common-sense constraints. First, all abundances must be non-negative. Second, the sum of abundances for a given pixel must be unity. However, it is the derivation and validation of the correct suite of endmembers that has remained a challenging and elusive goal for the past twenty years. Over the last years, several algorithms have been developed for automatic or semi-automatic extraction of spectral endmembers directly from an input hyperspectral data set. Some classic techniques for this purpose include the pixel purity index (PPI)[9] or the N-FINDR algorithm,[10] both focused on analyzing the data without incorporating information on the spatially adjacent data. However, one of the distinguishing properties of hyperspectral data is the multivariate information coupled with a two-dimensional (pictorial) representation amenable to image interpretation. In this work, we address the computational requirements introduced by spectral unmixing applications by addressing a specific case study focused on efficient implementation of the automatic morphological endmember extraction (AMEE) algorithm,[11] a technique that integrates both the spatial and the spectral information when searching for image endmembers in hyperspectral data sets.

2.2 Supervised classification

The utilization of fully or partially unsupervised approaches, such as the spectral unmixing techniques discussed in the previous subsection, are of great interest for extracting relevant information from hyperspectral scenes. However, several machine learning and image processing techniques have also been applied for the same purpose when *a priori* knowledge (often available in the form of labeled data or ground-truth) is available for the scenes to be analyzed.[12] In this case, the labeled data usually consists of training samples cataloged by assuming that each pixel vector measures the response of one single underlying material. Perhaps the most relevant examples of supervised machine learning techniques used for remote sensing data classification are support vector machines (SVMs)[13] and artificial neural networks.[14] Although many neural network architectures exist, for hyperspectral data classification mostly feed-forward networks of various layers, such as the multi-layer perceptron (MLP), have been used.[15] In this work, the computational requirements addressed by supervised classifiers are illustrated by a case study focused on efficient implementation of an MLP-based classifier for hyperspectral image data.

2.3 Automatic target detection

Another technique that has attracted a lot of attention in hyperspectral image analysis in recent years is automatic target detection,[16] which is particularly crucial in military-oriented applications. During the last few years, several algorithms have been developed for the aforementioned purpose, including the automatic target detection and classification (ATDCA) algorithm,[17] an unsupervised fully-constrained least squares (UFCLS) algorithm,[18] an iterative error analysis (IEA) algorithm,[19] or the well-known RX algorithm developed by Reed and Xiaoli for anomaly detection.[20] In this work, we illustrate the computational requirements of target detection applications by addressing a case study focused on efficient implementation of the ATDCA algorithm.

In the following section, we describe parallel versions of ATDCA for target detection, AMEE for endmember extraction, and MLP for supervised classification of hyperspectral data. We believe that, although additional implementation case studies would be certainly relevant, the spectrum of parallel techniques included in this paper provides sufficient coverage of different strategies and approaches for efficient information extraction from hyperspectral data sets.

3. PARALLEL IMPLEMENTATIONS

In all considered parallel implementations, a data-driven partitioning strategy has been adopted as a baseline for algorithm parallelization. Specifically, two approaches for data partitioning have been tested:[5]

- *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.

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.[5] In the following, we assume that spatial-domain decomposition is always used when partitioning the hyperspectral data cube.

3.1 Parallel Automatic Morphological Endmember Extraction

In this subsection, we describe a parallel implementation of the AMEE algorithm[11] for endmember extraction. The parallel algorithm adopts a spatial-domain partitioning framework in which border data are replicated in order to avoid communicating data during the kernel-based (local) processing, which relies on the definition of a spatial window or structuring element that is used to define spatial context around each pixel vector in the scene. The inputs to the parallel algorithm are a hyperspectral image cube $\mathbf{F}$ with n dimensions, where $\mathbf{f}(x, y)$ denotes the pixel vector at spatial coordinates (x, y) of the scene, a spatial kernel B (called morphological structuring element) of fixed size (3×3 pixels in our implementation), a maximum number of iterations I_{max}, and a maximum number of endmembers to be detected, e. The output in all cases is a set of endmembers $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_e\}$. The parallel algorithm consists of the following steps:

1. The master processor partitions the data into p spatial-domain partitions (with their scratch borders to avoid inter-processor communications), and distributes the partitions among the workers.
2. Using parameters I_{max} (maximum number of iterations) and e (maximum number of endmembers to be extracted), each worker executes (in parallel) the following steps:
   (a) Set $i = 1$ and initialize a morphological eccentricity index $\text{MEI}(x, y) = 0$ for each pixel $\boldsymbol{f}(x, y)$.
   (b) Move the structuring element through all the pixels of the input data, defining a local spatial search area around each pixel $\boldsymbol{f}(x, y)$, and calculate the maximum and minimum pixel vectors at each B-neighborhood using extended morphological erosion and dilation, respectively defined as follows:

$$(\boldsymbol{f} \ominus B)(x, y) = argmin_{(i,j) \in Z^2(B)} \{D_B[\boldsymbol{f}(x + i, y + j)]\} \tag{1}$$

$$(\boldsymbol{f} \oplus B)(x, y) = argmax_{(i,j) \in Z^2(B)} \{D_B[\boldsymbol{f}(x + i, y + j)]\} \tag{2}$$

   (c) Update the MEI at each spatial location (x, y) using:

$$\text{MEI}(x, y) = Dist[(\boldsymbol{f} \ominus B)(x, y), (\boldsymbol{f} \oplus B)(x, y)] \tag{3}$$

   (d) Set $i = i + 1$. If $i = I_{max}$, then go to step (e). Otherwise, replace the original image with its dilation using B as follows: $\boldsymbol{f} = \boldsymbol{f} \oplus B$. This propagates only the purest pixels at the local neighborhood to the following algorithm iteration. Then, go to step (b).
   (e) Select the set of pixel vectors with higher associated MEI scores (called endmember candidates).
3. The master gathers all the local endmember sets provided by the workers and forms a *global* set of final endmembers $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_e\}$.

3.2 Parallel Multi-Layer Perceptron

In this subsection, we describe a supervised parallel classifier based on a MLP neural network with back-propagation learning.[15] The inputs to the parallel algorithm are a hyperspectral image cube $\mathbf{F}$ with n dimensions, where $\mathbf{f}(x,y)$ denotes the pixel vector at spatial coordinates (x,y) of the scene. The number of input neurons of the considered neural network equals n, the number of spectral bands acquired by the sensor. The number of hidden neurons, m, is adjusted empirically, and the number of output neurons, c, equals the number of distinct classes to be identified in the input data.

The parallel classifier considered in this work is based on an exemplar partitioning scheme, also called training example parallelism, which explores data level parallelism and can be easily obtained by simply partitioning the training pattern data set. Each process determines the weight changes for a disjoint subset of the training population and then changes are combined and applied to the neural network at the end of each epoch. This scheme requires a suitable large number of training patterns to take an advantage. Each processor implements a complete neural network which is trained with a disjoint subset of training patterns $\mathbf{f}_j^p(x,y))$, executing the following three phases of the back-propagation learning algorithm for each training pattern:

1. *Parallel forward phase.* In this phase, the activation value of the hidden neurons local to the processors are calculated. For each input pattern –meaning that labeled pixel vectors are used as training patterns– the activation value for the hidden neurons is calculated at each processor using the expression $H_i^p = \varphi(\sum_{j=1}^{n} \omega_{ij} \cdot \mathbf{f}_j^p(x,y))$. Here, the activation values and weight connections of neurons present in other processors are required to calculate the activation values of output neurons according to the expression $O_k^P = \varphi(\sum_{i=1}^{m} \omega_{ki}^p \cdot H_i^p)$, with $k = 1, 2, \cdots, C$.

2. *Parallel error back-propagation.* In this phase, each processor calculates the error terms for the local hidden neurons. To do so, *delta* terms for the output neurons are first calculated using $(\delta_k^o)^P = (O_k - d_k)^p \cdot \varphi^{'}(\cdot)$, with $i = 1, 2, \cdots, c$. Then, error terms for the hidden layer are computed using the following expression: $(\delta_i^h)^p = \sum_{k=1}^{p} (\omega_{ki}^p \cdot (\delta_k^o)^p) \cdot \varphi^{'}(\cdot)$, with $i = 1, 2, \cdots, n$.

3. *Parallel weight update.* In this phase, the weight connections between the input and hidden layers are updated by $\omega_{ij} = \omega_{ij} + \eta^p \cdot (\delta_i^h)^p \cdot \mathbf{f}_j^p(x,y))$. Similarly, the weight connections between the hidden and output layers are updated using the expression: $\omega_{ki}^p = \omega_{ki}^P + \eta^p \cdot (\delta_k^o)^p \cdot H_i^p$.

4. *Broadcast and initialization of weight matrices.* In this phase, each node sends its partial weight matrices to its neighbour node, which sums it to its partial matrix and proceed to send it again to the neighbour. Once all nodes have added their local matrices the resulting total weight matrices are broadcast to be used by all processors in the next iteration.

Once the MLP neural network has been trained with a set of labeled patterns, a parallel classification step follows. For each pixel vector $\mathbf{f}(x,y)$ in the input data cube, the classification step calculates (in parallel) $\sum_{j=1}^{p} O_k^j$, with $k = 1, 2 \cdots, c$. A classification label for each pixel can be obtained using the winner-take-all criterion commonly used in neural networks by finding the cumulative sum with maximum value, say $\sum_{j=1}^{p} O_{k^*}^j$, with $k^* = \arg\{\max_{1 \le k \le c} \sum_{j=1}^{p} O_k^j\}$.

3.3 Parallel Automatic Target Detection and Classification Algorithm

In this subsection, we describe a parallel implementation of the ATDCA algorithm[17] for automatic target detection. The inputs to the parallel algorithm are a hyperspectral image cube $\mathbf{F}$ with n dimensions, where $\mathbf{f}$ denotes a pixel vector at spatial coordinates (x,y) of the scene, and a maximum number of targets to be detected, t. The output in all cases is a set of target pixel vectors $\{\mathbf{t}_0, \mathbf{t}_1, \cdots, \mathbf{t}_t\}$. The parallel algorithm 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.

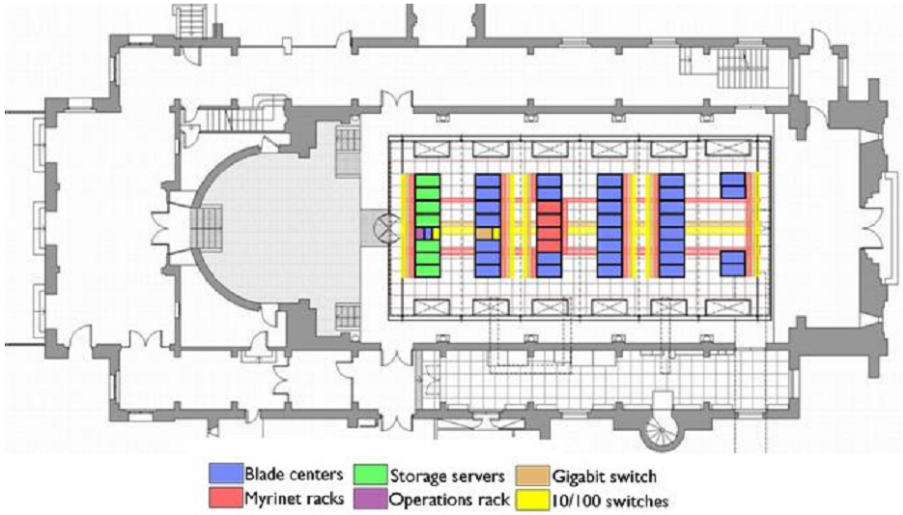

Figure 2. The MareNostrum supercomputer at Barcelona Supercomputing Center (this figure, along with further details on the system are available online: http://www.bsc.es.

2. Each worker finds the brightest pixel in its local partition using $\mathbf{t}_1 = argmax\{\mathbf{f}(x,y)^T \cdot \mathbf{f}(x,y)\}$, where the superscript T denotes the vector transpose operation. Each worker then sends the spatial locations (x,y) 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, $\mathbf{t}_1$, 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 $\mathbf{U} = \mathbf{t}_1$ 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 $\mathbf{f}(x,y)$ 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 (x,y) provided by the workers, and selecting the one which results in the maximum score as follows $\mathbf{t}_2 = argmax\{(P_{\mathbf{U}}^{\perp}\mathbf{f}(x,y))^T(P_{\mathbf{U}}^{\perp}\mathbf{f}(x,y))\}$. The master sets $\mathbf{U} = \{\mathbf{t}_1, \mathbf{t}_2\}$ and broadcasts this matrix to all workers.

6. Repeat from step 4 until a set of t target pixels, $\{\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_t\}$, are extracted from the input data.

4. EXPERIMENTAL RESULTS

4.1 MareNostrum supercomputer

MareNostrum is one of the most powerful supercomputers in Europe[‡]. Although only a reduced number of resources were needed in our specific application and we only scaled up to 512 processors, the MareNostrum system has recently increased its calculation capacity until reaching 94.21 Teraflops (94.21 trillions of operations per second), doubling its previous capacity (42.35 Teraflops). It had 4.812 processors and has now 10.240 processors with a final calculation capacity of 94.21 Teraflops. The system has 44 racks and takes up a space of 120 square meters. Fig. 2 depicts the floor where the supercomputer is located.

[‡]http://www.top500.org/list/2008/11/100

4.2 Hyperspectral images

Three hyperspectral scenes collected by the AVIRIS instrument have been used in experiments:

- The first one was gathered over the Indian Pines test site in Northwestern Indiana, a mixed agricultural/forested area, early in the growing season, and consists of 1939×677 pixels and 224 spectral bands in the wavelength range $0.2 - 2.5\mu$m (574 MB in size). This scene, used to illustrate endmember extraction using the parallel AMEE algorithm, represents a very challenging analysis dominated by similar spectral classes and mixed pixels. In particular, the primary crops of the area, mainly corn and soybeans, were very early in their growth cycle with only about 5% canopy cover. Hence the importance of applying endmember extraction and spectral unmixing techniques. Fig. 3(a) shows the spectral band at 587 nm of the original scene and Fig. 3(b) shows the corresponding ground-truth map, displayed in the form of a class assignment for each labeled pixel, with 30 mutually exclusive ground-truth classes.

- The second AVIRIS data set used in experiments was collected over the Valley of Salinas in Southern California. The full scene consists of 512×217 samples with 224 spectral bands in the same range as the previous scene (48 MB in size). It was taken at low altitude with a pixel size of 3.7 meters. The data include vegetables, bare soils and vineyard fields. Fig. 4(a) shows the entire scene. Fig. 4(b) shows the available ground-truth regions. As shown in Fig. 4(b), ground-truth is available for about two thirds of the entire Salinas scene. As a result, it is a good test site for supervised classification using the proposed parallel MLP-based algorithm.

- The third scene was obtained after an AVIRIS flight 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 selected data set comprises 614×512 pixels and 224 narrow spectral bands in the same range as the previous scenes (140 MB in size). The spatial resolution is very fine as a consequence of the low altitude of the AVIRIS flight; with 1.7 meters per pixel (most other AVIRIS data sets exhibit spatial resolutions of 20m per pixel). The fine spatial resolution available in this application case study allowed us to use this scene as a benchmark for target detection studies, which often require very fine spatial and spectral detail. Hence, we use this scene to evaluate the performance of the parallel ATDCA algorithm. Fig. 5(a) 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. Fig. 5(b) shows a thermal map centered at the region where the buildings collapsed. The map, generated by U.S. Geological Survey (USGS) shows the target locations of the thermal hot spots, shown as bright red, orange, and yellow spots.

4.3 Parallel Performance

This subsection provides performance results for the parallel algorithms described in section 3. It should be noted that, in this work, we are mainly interested in validating the parallel performance of our proposed implementations on the MareNostrum system and not on evaluating the accuracy of the considered algorithms with the aforementioned hyperspectral scenes. Detailed validation results can be found in previous work for the AMEE (using the AVIRIS Indian Pines scene),[5] for MLP (using the AVIRIS Salinas scene),[21] and ATDCA (using the AVIRIS World Trade Center scene).[22] In the following, we analyze the scalability of parallel implementations of the algorithms above with the considered scenes, providing only experimental results for algorithm configurations that have been optimized after testing different input parameters. We would also like to emphasize that, in all cases, the parallel versions provide exactly the same results as the corresponding sequential implementations, which were used as a reference to calculate the processing times as the number of processors increases).

Table 1 shows the processing times measured for the parallel implementation of AMEE using a number of processors ranging from 2 to 512 CPUs on MareNostrum. The algorithm was run on the AVIRIS Indian Pines scene using 3×3 structuring element and $I_{max} = 5$ iterations. As it can be seen in the table, the algorithm scales well up to 128 processors. In order to illustrate this effect, we also measured the load balancing scores for the parallel algorithm. Load balance is defined as $D = Max/Min$, where Max and Min are the maxima and minima

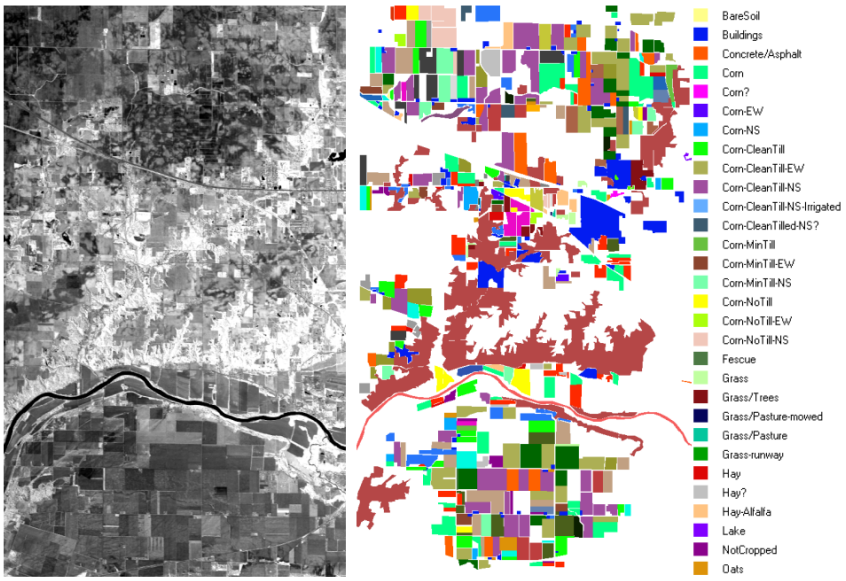

Figure 3. AVIRIS Indian Pines scene. (a) Spectral band at 439 nm. (b) Ground-truth map.

Figure 4. AVIRIS Salinas scene. (a) Spectral band at 439 nm. (b) Ground-truth map. (c) MLP classification result.

processor run times, respectively. Therefore, perfect balance is achieved when $D = 1$. In our experiments, we observed that load balance was almost perfect (around 0.99) until 128 processors, and started to decrease for 256 processors (0.84) and 512 processors (0.61). This is due to the fact that the spatial-domain partitions become smaller, and the ratio of communications versus computations increases. Since our experiments are confined to a single AVIRIS scene, we anticipate that this effect will not occur if a larger number of scenes is used as input to the parallel algorithm (in which case the ratio of computations to communications is expected to be very high). In any event, and although the algorithm has been shown to scale well for large volumes of data, our future work will be directed towards improving the scalability of the parallel AMEE algorithm on the MareNostrum system.

On the other hand, Table 2 shows the processing times measured for the parallel implementation of MLP using a number of processors ranging from 4 to 512 CPUs on MareNostrum. The algorithm was run on the AVIRIS Salinas scene using $m = 40$ hidden nodes and a total of 2500 training samples extracted from the ground truth (the remaining samples were used for testing), where parameter c was set to the number of classes in Fig.

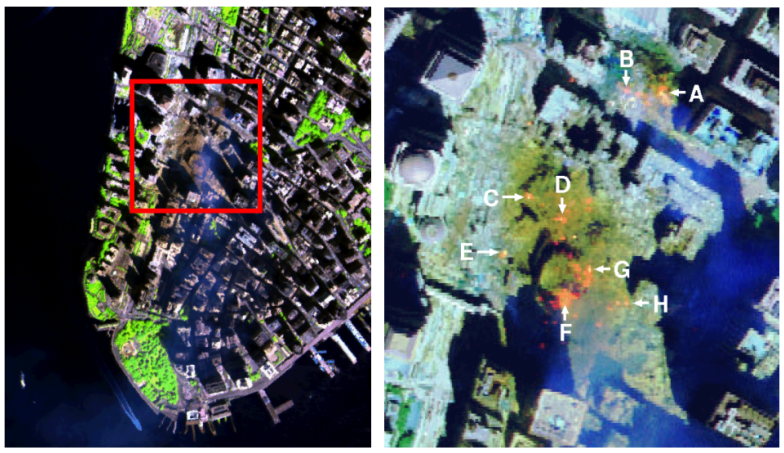

Figure 5. (Left) AVIRIS World Trade Center scene. (Right) USGS map of thermal hot spots.

Number of CPUs	2	4	8	16	32	64	128	256	512
Processing time	4833	2418	1218	625	325	173	103	95	113

Table 1. Processing times (seconds) achieved by our parallel version of AMEE endmember extraction algorithm using different numbers of CPUs on the MareNostrum supercomputer.

4(b). With this configuration, the parallel algorithm provided classification results with high accuracy (above 90%) as indicated in Fig. 4(c). As shown by Table 2, the parallel algorithm scaled relatively well and allowed a significant reduction of the processing time, although the parallel backpropagation implementation based on exemplar parallelism is essentially an irregular algorithm. Most of the irregularities arise due to the fact that, when the number of processors is increased, the distribution and number of training patterns on each processor is modified, thus affecting the convergence procedure and the final number of iterations needed. Table 3 shows the ratio between iteration number and executing time for different number of processors used (in the form of averaged iteration time). As it can be seen in the table, using this approach, the algorithm scales well up to 128 processors. Further analysis and tests should be conducted in order to analyze the scalability to a higher number of processors and to improve load balancing.

Finally, Table 4 shows the processing times measured for the parallel implementation of ATDCA using a number of processors ranging from 2 to 256 CPUs on MareNostrum. The algorithm was run on the AVIRIS World Trade Center scene using $t = 200$ targets. Although this number appears high, it should be noted that the complexity of an urban data analysis scenario, indicated by Fig. 5(left), requires that a high number of targets are extracted in order to be able to identify all targets of interest such as the thermal hot spots shown in Fig. 5(right). From Table 4, it can also be seen that the parallel ATDCA scales well up to 32 processors. For 64 processors and beyond, the ratio of communications to computations severely affects the scalability of the algorithm. Further work should be conducted in order to better adapt the parallel version of this algorithm to the MareNostrum architecture by increasing the ratio of computations to communications.

5. CONCLUSIONS AND FUTURE LINES

In this paper, we have evaluated the performance of several parallel techniques for hyperspectral image processing that have been specifically designed to be run on the MareNostrum supercomputer at Barcelona Supercomput-

Number of CPUs	4	8	16	32	64	128	256	512
Processing time	61545	44021	4539	4829	4394	2033	754	675

Table 2. Processing times (seconds) achieved by our parallel version of MLP supervised classification algorithm using different numbers of CPUs on the MareNostrum supercomputer.

Number of CPUs	4	8	16	32	64	128	256	512
Avg. iteration time	0.218	0.105	0.050	0.025	0.013	0.007	0.004	0.003

Table 3. Averaged iteration times (in seconds) achieved by our parallel version of MLP supervised classification algorithm using different numbers of CPUs on the MareNostrum supercomputer.

Number of CPUs	2	4	8	16	32	64	128	256
Processing time	521	263	221	129	44	86	206	333

Table 4. Processing times (seconds) achieved by our parallel version of ATDCA target detection algorithm using different numbers of CPUs on the MareNostrum supercomputer.

ing Center. The techniques developed cover the three following areas: 1) spectral mixture analysis, a popular approach to characterize mixed pixels in hyperspectral data (addressed by efficient implementation of a spatial-spectral endmember extraction algorithm); 2) supervised classification using MLP-based neural networks; and 3) target detection using an automatic algorithm based on orthogonal subspace projection concepts. Our preliminary results reveal some interesting properties of the developed parallel implementations that should be enhanced in future work to increase scalability, such as the ratio of computations to communications or the sustained performance of the parallel algorithms independently of the volume of data to be processed. In our experiments we have observed that, although the algorithms scale well when processing large image volumes (which is appealing for information extraction from large data archives), further work is still needed in order to effectively process smaller data volumes. Future work will also comprise a more detailed investigation of algorithm parameters including computation and communication patterns.

6. ACKNOWLEDGEMENT

This work has been supported by the project AECT-2008-2-0012 entitled High Performance Computing for Earth Observation-Based Hyperspectral Imaging Applications provided by the Spanish Supercomputing Network (RES). Funding from the Spanish Ministry of Science and Innovation (HYPERCOMP/EODIX project, reference AYA2008-05965-C04-02) is also gratefully acknowledged.

REFERENCES

1. A. Plaza and C.-I. Chang, *High performance computing in remote sensing*, CRC Press, Boca Raton, 2006.
2. C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification*, Kluwer Academic and Plenum Publishers, New York, 2003.
3. A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for earth remote sensing," *Science* **228**, pp. 1147–1153, 1985.
4. R. O. Green, "Imaging spectroscopy and the airborne visible-infrared imaging spectrometer (AVIRIS)," *Remote Sensing of Environment* **65**, pp. 227–248, 1998.
5. A. Plaza, D. Valencia, J. Plaza, and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *Journal of Parallel and Distributed Computing* **66**(3), pp. 345–358, 2006.
6. D. A. Landgrebe, *Signal theory methods in multispectral remote sensing*, John Wiley and Sons, Hoboken, NJ, 2003.
7. J. B. Adams, M. O. Smith, and P. E. Johnson, "Spectral mixture modeling: a new analysis of rock and soil types at the viking lander 1 site," *Journal of Geophysical Research* **91**, pp. 8098–8112, 1986.
8. A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing* **42**, pp. 650–663, 2004.
9. J. W. Boardman, "Automating spectral unmixing of aviris data using convex geometry concepts," in *Summaries of Airborne Earth Science Workshop*, R. O. Green, ed., *JPL Publication* **93-26**, pp. 111–114, 1993.

10. M. E. Winter, "Algorithm for fast autonomous spectral endmember determination in hyperspectral data," in *Imaging Spectrometry V*, M. R. Descour and S. S. Shen, eds., *Proceedings of SPIE* **3753**, pp. 266–275, 1999.
11. A. Plaza, P. Martinez, R. Perez, and J. Plaza, "Spatial/spectral endmember extraction by multidimensional morphological operations," *IEEE Transactions on Geoscience and Remote Sensing* **40**, pp. 2025–2041, 2002.
12. G. M. Foody and A. Mathur, "Toward intelligent training of supervised image classifications: directing training data acquisition for svm classification," *Remote Sensing of Environment* **93**, pp. 107–117, 2004.
13. L. Bruzzone, M. Chi, and M. Marconcini, "A novel transductive svm for the semisupervised classification of remote sensing images," *IEEE Trans. Geoscience and Remote Sensing* **44**, pp. 3363–3373, 2006.
14. C. Lee and D. A. Landgrebe, "Decision boundary feature extraction for neural networks," *IEEE Trans. Neural Networks* **8**, pp. 75–83, 1997.
15. J. Plaza, A. Plaza, R. Perez, and P. Martinez, "On the use of small training sets for neural network-based characterization of mixed pixels in remotely sensed hyperspectral images," *Pattern Recognition* **42**, pp. 3032–3045, 2009.
16. 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.
17. H. Ren and C.-I. Chang, "Automatic spectral target recognition in hyperspectral imagery," *IEEE Trans. Aerosp. Electron. Syst.* **39**, pp. 1232–1249, 2003.
18. 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.
19. 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.
20. 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.
21. J. Plaza, R. Perez, A. Plaza, P. Martinez, and D. Valencia, "Parallel morphological/neural processing of hyperspectral images using heterogeneous and homogeneous platforms," *Cluster Computing* **11**, pp. 17–32, 2008.
22. A. Paz, A. Plaza, and S. Blazquez, "Parallel implementation of target detection algorithms for hyperspectral imagery," *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium* **1**, pp. 234–239–32, 2008.