Available online at www.sciencedirect.com
SCIENCE DIRECT®
J. Parallel Distrib. Comput. 66 (2006) 345–358
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Commodity cluster-based parallel processing of hyperspectral imagery

Antonio Plaza*, David Valencia, Javier Plaza, Pablo Martinez

Department of Computer Science, Polytechnic Institute of Caceres, University of Extremadura, Avda. de la Universidad s/n, E-10071 Caceres, Spain

Received 10 July 2004; received in revised form 18 September 2005; accepted 3 October 2005
Available online 11 November 2005

Abstract

The rapid development of space and computer technologies has made possible to store a large amount of remotely sensed image data, collected from heterogeneous sources. In particular, NASA is continuously gathering imagery data with hyperspectral Earth observing sensors such as the Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) or the Hyperion imager aboard Earth Observing-1 (EO-1) spacecraft. The development of fast techniques for transforming the massive amount of collected data into scientific understanding is critical for space-based Earth science and planetary exploration. This paper describes commodity cluster-based parallel data analysis strategies for hyperspectral imagery, a new class of image data that comprises hundreds of spectral bands at different wavelength channels for the same area on the surface of the Earth. An unsupervised technique that integrates the spatial and spectral information in the image data using multi-channel morphological transformations is parallelized and compared to other available parallel algorithms. The code's portability, reusability and scalability are illustrated by using two high-performance parallel computing architectures: a distributed memory, multiple instruction multiple data (MIMD)-style multicomputer at European Center for Parallelism of Barcelona, and a Beowulf cluster at NASA's Goddard Space Flight Center. Experimental results suggest that Beowulf clusters are a source of computational power that is both accessible and applicable to obtaining results in valid response times in information extraction applications from hyperspectral imagery.

Keywords: Commodity cluster; Hyperspectral imaging; Load balance; Mathematical morphology; Parallelizable spatial/spectral partition (PSSP)

1. Introduction

Recent advances in sensor technology have led to the development of hyperspectral imaging systems capable of collecting hundreds of images corresponding to different wavelength channels for the same area on the surface of the Earth [4]. A chief hyperspectral sensor is the NASA's Jet Propulsion Laboratory Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) system [9], which covers the wavelength region from 0.4 to 2.5 μm using 224 spectral channels and nominal spectral resolution of 10 nm. Another example is the Hyperion instrument aboard Earth Observing-1 (EO-1) spacecraft, which has been NASA's first hyperspectral imager to become operational on-orbit. It routinely collects images hundreds of kilometers long with 220 spectral bands in the same spectral range as AVIRIS. With such spectral detail, the ability to detect and identify individual materials or land-cover classes is greatly enhanced with respect to other techniques available such as multispectral imaging, which typically collects only tens of images. The incorporation of hyperspectral sensors on airborne/satellite platforms is currently producing a nearly continual stream of multidimensional data, and this high data volume demands robust and efficient data analysis techniques. Specifically, both AVIRIS and Hyperion data repositories currently consist of hundreds of Terabytes of image data. Since these archives are ever growing (it is estimated that NASA collects and sends to Earth more than 850 Gb of hyperspectral data every day), the development of fast, unsupervised analysis techniques for near real-time information extraction and data mining has become a highly desired goal yet to be fully accomplished.

A diverse array of techniques has been applied to analyze hyperspectral imagery during the last decade [4,13]. They are inherently either full pixel techniques or mixed pixel techniques, where each pixel vector in a hyperspectral image records the spectral information. The underlying assumption governing full pixel techniques is that each pixel vector measures the response of one predominantly underlying material at each site in a scene.

* Corresponding author. Fax: +34 927 257203.
E-mail address: aplaza@unex.es (A. Plaza).

0743-7315/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2005.10.001

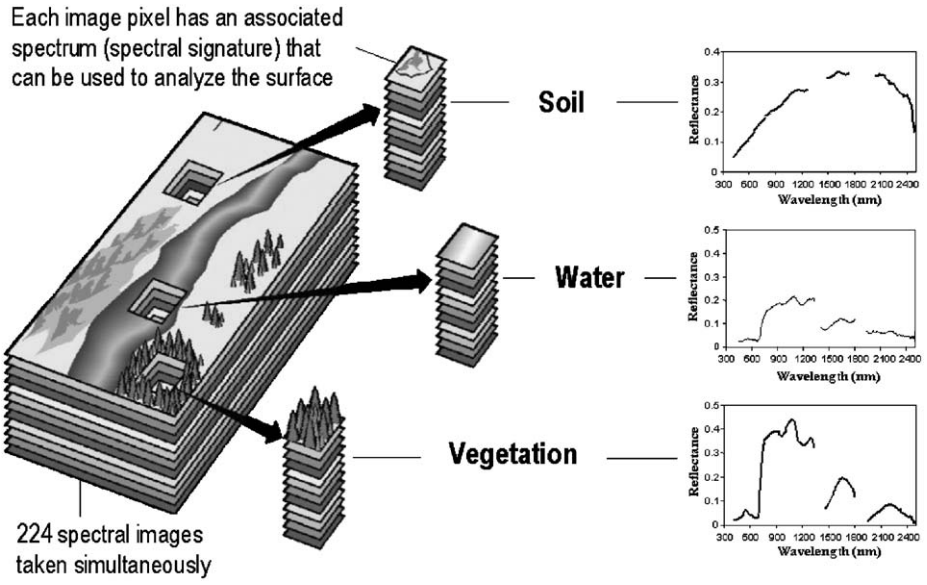

Fig. 1. The concept of hyperspectral imaging using NASA Jet Propulsion Laboratory's AVIRIS sensor.

In contrast, the underlying assumption governing mixed pixel techniques is that each pixel vector measures the response of multiple underlying materials at each site. Mixed pixels are a mixture of more than one distinct substance, and exist for one of two reasons. Firstly, if the spatial resolution of the sensor is not high enough to separate different materials, these can jointly occupy a single pixel, and the resulting spectral measurement will be a composite of the individual spectra. Secondly, mixed pixels can also result when distinct materials are combined into a homogeneous mixture. This circumstance occurs independent of the spatial resolution of the sensor. A hyperspectral image (sometimes referred to as "image cube") is often a combination of the two situations, where a few sites in a scene are pure materials, but many other are mixtures of materials (see Fig. 1).

Spectral unmixing is a commonly used procedure in which the measured spectrum of a mixed pixel is decomposed into a collection of spectrally pure constituent spectra, or *endmembers*, and a set of correspondent fractions, or *abundances*, that indicate the proportion of each endmember present in the pixel [12]. Hence, the identification of image endmembers is a crucial objective in hyperspectral image analysis applications. It is important to emphasize that most available techniques for hyperspectral data analysis focus on analyzing the data without incorporating information on the spatially adjacent data; i.e. the hyperspectral data is treated not as an image but as an unordered listing of spectral measurements where the spatial coordinates can be shuffled arbitrarily without affecting the analysis [10]. However, one of the distinguishing properties of hyperspectral data, as collected by available imaging spectrometers, is the multivariate information coupled with a two-dimensional (2-D) pictorial representation amenable to image interpretation. Subsequently, there is a need to incorporate the spatial component of the data in the development of automated techniques for hyperspectral data exploitation. Unfortunately, most available techniques for analyzing hyperspectral data are based on the spectral information alone [13]. It should also be noticed that such spectral-based techniques yield the same result for a data cube, and for the same data cube where the spatial positions have been randomly permuted. By taking into account the complementary nature of spatial and spectral information in simultaneous fashion, it is possible to alleviate the problems related to each of them taken separately [18].

While integrated spatial/spectral developments hold great promise for Earth science image analysis, they also introduce new processing challenges [14]. In particular, the price paid for the wealth of spatial and spectral information available from hyperspectral sensors is the enormous amounts of data that they generate. In addition, analysis techniques in Earth observation studies are often computationally tedious, and require lengthy durations to calculate desired quantities [5,2]. Several applications exist, however, where having the desired information calculated in near real-time is highly desirable. Such is the case of military applications, where a commonly pursued goal is detection of full or sub-pixel targets, often associated to hostile weaponry, camouflage, concealment, and decoys. Other relevant examples can be found in applications aimed at detecting and/or tracking natural disasters such as forest fires, oil spills, and other types of chemical contamination, where timely classification is highly desirable.

In recent years, high-performance computing systems have become more and more widespread, especially with the advent of relatively cheap Beowulf clusters [3,7]. The new processing power offered by such commodity cluster-based systems can be employed to tackle large remotely sensed data sets and to get reasonable response times in complex image analysis scenarios [1,6,8,22]. Thanks to the geographic local organization of the pixels of an image as a 2-D mesh [11], and to the regularity of most low-level computations, mesh-based parallel architectures have become quite popular for image analysis applications since they allow for efficient

implementations of basic neighbor-based primitives [24]. However, the development of appropriate parallel analysis techniques for joint spatial/spectral analysis of multi-component data such as hyperspectral imagery has not been fully explored yet in the literature, and represents both a challenge and a novel contribution. In order to take advantage of available parallel computing architectures, designing and implementing well-optimized spatial/spectral analysis techniques is essential in order to reduce the total research time to complete these studies [27].

This paper describes a parallel morphological technique that allows for efficient spatial/spectral exploitation of hyperspectral image data. The algorithm was specifically designed to be run on similar computing nodes employed in a stand-alone manner. In the following section, we describe standard approaches for parallel hyperspectral imaging in the literature. Next, the morphological analysis method is described in light of available approaches, and a detailed description of its parallel implementation is provided. The algorithm is evaluated in Section 4, which also includes a detailed survey on the parallel code's portability, scalability and performance in comparison with other parallel hyperspectral analysis algorithms. Two high-performance parallel computers at European Center for Parallelism of Barcelona (CEPBA) and NASA's Goddard Space Flight Center (NASA/GSFC) are used to investigate variables such as impact of inter-processor communication and coordination, load balance, and speedup ratios. Finally, Section 5 concludes with some remarks and future research lines.

2. Available parallel hyperspectral imaging algorithms

Despite the growing interest in hyperspectral imaging, only a few research efforts devoted to the design of parallel implementations exist in the open literature. It should be noted that several existing parallel techniques are subject to non-disclosure restrictions, mainly due to their use in military and defense applications. However, with the recent explosion in the amount of hyperspectral imagery, parallel processing is expected to become a requirement in virtually every remote sensing application. As a result, this paper takes a necessary first step toward the comparison of different techniques and strategies for parallel hyperspectral image analysis. In the following, we describe two standard unsupervised parallel hyperspectral image classification techniques. These methods will be used for comparative evaluation and assessment in this work.

2.1. Distributed spectral-screening principal component transform algorithm (S-PCT)

The principal component transform (PCT) is one of the most widely used spectral transformations in hyperspectral analysis [13]. It is used to summarize and decorrelate the images by reducing redundancy and packing the residual information into a small set of images, termed *principal components*. PCT is a highly compute-intensive algorithm amenable to parallel implementation. In order to speed performance up, the S-PCT algorithm [1] uses a standard master–slave decomposition technique, where the master coordinates the actions of the workers, gathers the partial results from them and provides the final result. To reduce communication overhead, the worker overlaps the request for its next subproblem with the calculation associated with the current subproblem. Using this approach, the S-PCT algorithm can be divided into eight steps as shown below.

S-PCT algorithm

Inputs: N-Dimensional (N-D) image cube $\boldsymbol{f}$, number of unique spectra p.

Outputs: 2-D image which contains a classification label for each pixel $\boldsymbol{f}(x, y)$ in the original image.

1. Divide the original image cube $\boldsymbol{f}$ into K partitions such that there is no overlapping among different partitions, where K is the number of workers in the system. Send each partition, consisting of a set of pixel vectors, to a worker. Each worker proceeds concurrently to form a unique spectral set by calculating the spectral angle for all vector pairs as:
$$\mathrm{SAD}\left[\boldsymbol{f}(x, y), \boldsymbol{f}(x', y')\right] = \cos^{-1}\left(\frac{\boldsymbol{f}(x, y) \cdot \boldsymbol{f}(x', y')}{\|\boldsymbol{f}(x, y)\| \, \|\boldsymbol{f}(x', y')\|}\right).$$
2. The K unique sets are sent back to the master and combined, one pair at a time. Upon completion, there will be only one unique set left with p unique pixel vectors.
3. Calculate the N-D mean vector $\boldsymbol{m}$ 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.
4. All the pixel vectors in the unique set are divided into K parts and sent to K workers. Each worker then computes the covariance component and forms a covariance sum.
5. Calculate the covariance matrix sequentially as the average of all the matrices calculated in step 4.
6. Obtain a transformation matrix $\boldsymbol{T}$ by calculating and sorting the eigenvectors of the covariance matrix according to their corresponding eigenvalues, which provide a measure of their variances. As a result, the spectral content is forced into the front components. Since the degree of data dependency of the calculation is high and its complexity is related to the number of spectral bands rather than the image size, this step is also done sequentially at the master.
7. Transform each pixel vector in the original hyperspectral image independently using $\boldsymbol{T} \cdot \left[\boldsymbol{f}(x, y) - \boldsymbol{m}\right]$. This step is done in parallel, where all workers transform their respective portions of data concurrently.
8. Finally, a parallel post-processing step is applied to perform classification at a pixel level in the PCT-transformed space. First, K partitions of a reduced, p-dimensional data cube given by the first PCT components are sent to the workers, along with the spatial locations of the p unique pixel vectors resulting from step 2. Each worker then labels each pixel in its corresponding partition with a class label given

by the most spectrally similar unique pixel vector in the PCT-reduced space, and sends back the result to the master, which composes a final 2-D classification image.

It should be noted that step 8 was not included in the original S-PCT algorithm, which only generates a color composite using the first PCT bands for visual interpretation of the data, but is included in our implementation to allow a comparison with other parallel classification algorithms for hyperspectral imagery.

2.2. Distributed ISODATA algorithm (D-ISODATA)

One of the main drawbacks of PCT-based techniques is that they are based on the statistical significance of the spectra, rather than the uniqueness of the spectra. As a result, small objects and ground features (which may be crucial in certain applications) would likely manifest themselves in the last principal components, thus being discarded prior to classification. In order to resolve this issue, the D-ISODATA algorithm [6] was designed as the first parallel approach able to deal with the entire high-dimensional volume directly, thereby preserving all the spectral information in the data. Since the ISODATA classification procedure is regarded as the benchmark for all unsupervised classification algorithms, it will be of great interest to include its parallel implementation in our comparative assessment. The algorithm uses the Euclidean distance as a similarity measure to cluster data elements into different classes [13]. A master–slave parallel implementation of this algorithm can be summarized as follows.

D-ISODATA algorithm

Inputs: N-D image cube $\boldsymbol{f}$, number of clusters p, convergence threshold t.

Outputs: 2-D image which contains a classification label for each pixel $\boldsymbol{f}(x, y)$ in the original image.

1. Divide the original image cube $\boldsymbol{f}$ into K equally sized blocks such that there is no overlapping among different blocks. Send each partition, consisting of a set of pixel vectors, to a worker, along with a set of p randomly selected pixel vectors assumed to be initial centroids.
2. Each worker labels each pixel $\boldsymbol{f}(x, y)$ in the corresponding partition. Let us denote by n_{ij} the number of pixels belonging to the jth cluster of the ith worker, and by $\boldsymbol{f}_k^j(x, y)$ the kth pixel of the jth cluster. Then, to minimize inter-processor communication, each worker sends the number of pixels belonging to each cluster and the summation of feature vectors of all pixels belonging to each cluster, that is,
$$\boldsymbol{Y}_{ij} = \left[\sum_{k=1}^{n_{ij}} \boldsymbol{f}_k^1(x, y), \ldots, \sum_{k=1}^{n_{ij}} \boldsymbol{f}_k^p(x, y)\right].$$
3. The master collects all the information provided by the workers and combines it to obtain a new centroid for each cluster j using $\boldsymbol{c}_j = \left(\sum_{i=1}^{p_j} \boldsymbol{Y}_{ij} / \sum_{i=1}^{p_j} \boldsymbol{Y}_{ij} \sum_{i=1}^{p_j} n_{ij}\right)$, where p_j is the number of pixels in the cluster.
4. The master compares the current centroids and the new centroids. If the difference between them is less than the convergence threshold t, then convergence occurs and the master informs all workers about the current status of convergence. Otherwise, steps 2–4 are repeated until the convergence status is true.
5. Following convergence, each worker i computes the summation of the Euclidean distance of all pixels within a cluster j from its centroid $\boldsymbol{c}_j$. Each worker then sends this information to the master, which combines it to obtain the deviation of pixels within each cluster j. It should be noted that this information is only exchanged when convergence has occurred (instead of doing so every time new centroids are calculated) in order to minimize inter-processor communication.
6. The master now decides about split or merge of the resulting clusters, based on parameters such as the intercluster distances (separation), the intracluster distances (compactness), the ratio of standard deviations along different axes for each cluster, and the number of pixels per cluster [6]. The above procedure is repeated (following the same steps for the previous convergence) until no cluster is further eligible for split and merge.
7. When the stopping criterion is satisfied, each worker sends the label (cluster number) associated with each local pixel to the master, which combines all the individual results and forms the final 2-D image.

To conclude this section, we must emphasize that the two parallel algorithms above rely exclusively on the spectral information of the data, without considering the spatial correlation. In the following subsection, we describe a parallel algorithm that combines spectral and spatial information in the original data through the use of multi-channel morphological operations.

3. Parallel morphological hyperspectral imaging algorithm

Mathematical morphology [26] is a classic non-linear spatial processing technique that provides a remarkable framework to achieve the desired integration of spatial and spectral responses in hyperspectral data analysis. In this section, we first briefly describe a sequential morphological algorithm for endmember extraction and mixed pixel classification. We will refer to the algorithm hereinafter as Automated Morphological Endmember Extraction (AMEE). In order to render this algorithm computationally, it will be embedded in a parallel implementation (AMEEPAR).

3.1. Automated morphological endmember extraction

In order to define morphological operations in hyperspectral imaging, we first impose an ordering relation in terms of spectral purity in a set of neighboring pixel vectors lying within a kernel neighborhood, known as structuring element (SE) in mathematical morphology terminology. Let us consider a flat SE designed by B. Then, the cumulative distance between one particular pixel $\boldsymbol{f}(x, y)$ and all the pixel vectors in the spatial neighborhood given by B (B-neighborhood) can be defined

as follows:

$$D_B\left[\boldsymbol{f}(x,y)\right] = \sum_s \sum_t \mathrm{SAD}\left[\boldsymbol{f}(x,y), \boldsymbol{f}(s,t)\right], \quad \forall (s,t) \in Z^2(B), \tag{1}$$

where SAD is the spectral angle between two N-D vectors. As a result, $D_B\left[\boldsymbol{f}(x,y)\right]$ is given by the sum of SAD scores between $\boldsymbol{f}(x,y)$ and every other pixel vector in the B-neighborhood. Based on the cumulative distance above, the flat extended erosion of $\boldsymbol{f}$ by B is based on the selection of the B-neighborhood pixel vector that produces the minimum value for D_B:

$$(\boldsymbol{f} \Theta B)(x,y) = \left\{\boldsymbol{f}\left(x+s', y+t'\right), \left(s', t'\right) = \arg\min_{(s,t)\in Z^2(B)} \left\{D_B\left[\boldsymbol{f}(x+s, y+t)\right]\right\}\right\}, \quad (x,y) \in Z^2, \tag{2}$$

where the arg min operator selects the pixel vector is most highly similar, spectrally, to all the other pixels in the B-neighborhood. On other hand, the flat extended dilation of $\boldsymbol{f}$ by B selects the B-neighborhood pixel vector that produces the maximum value for D_B:

$$(\boldsymbol{f} \oplus B)(x,y) = \left\{\boldsymbol{f}\left(x-s', y-t'\right), \left(s', t'\right) = \arg\max_{(s,t)\in Z^2(B)} \left\{D_B\left[\boldsymbol{f}(x-s, y-t)\right]\right\}\right\}, \quad (x,y) \in Z^2, \tag{3}$$

where the arg max operator selects the pixel vector that is most spectrally distinct to all the other pixels in the B-neighborhood. With the above definitions in mind, we provide below a morphological approach to endmember extraction that focuses on the analysis of spatial and spectral patterns simultaneously [18,17]. The full image data cube, with no previous dimensionality reduction or pre-processing, is input to the method. The algorithm can be summarized by the following steps.

AMEE algorithm

Inputs: N-D image cube $\boldsymbol{f}$, structuring element B, number of iterations I_{MAX}, number of endmembers p.

Output: 2-D image which contains a classification label for each pixel $\boldsymbol{f}(x,y)$ in the original image.

1. Set $i = 1$ and initialize a morphological eccentricity index score $\mathrm{MEI}(x,y) = 0$ for each pixel.
2. Move B through all the pixels of $\boldsymbol{f}$, defining a local spatial search area around each $\boldsymbol{f}(x,y)$ and calculate the maximum and the minimum pixel at each B-neighborhood using dilation and erosion, respectively. Update the MEI at each pixel using the SAD between the maximum and the minimum.
3. Set $i = i + 1$. If $i = I_{\max}$ then go to step 4. Otherwise, replace $\boldsymbol{f}$ by its dilation using B , and go to step 2.
4. Select the set of p pixel vectors in $\boldsymbol{f}$ with higher associated score in the resulting MEI image and form a unique spectral set of $q \leqslant p$ endmembers by calculating the spectral angle for all vector pairs. Estimate the fractional abundance, $\alpha_i(x,y)$, of those signatures at $\boldsymbol{f}(x,y)$ using the linear mixture model in [12].
5. Obtain a classification label for each pixel $\boldsymbol{f}(x,y)$ by assigning it to a class given by the endmember with the highest fractional abundance score in that pixel. This is done by comparing all estimated abundance fractions $\left\{\alpha_1(x,y), \alpha_2(x,y), \ldots, \alpha_q(x,y)\right\}$ and finding the one with the maximum value, say

$$\alpha_{i^*}(x,y), \quad \text{with } i^* = \arg\left\{\max_{1\leqslant i\leqslant q}\left\{\alpha_i(x,y)\right\}\right\}.$$

The AMEE algorithm has been described in [17,19,20] and we will not expand on its detailed implementation here. One of the main features of the algorithm is regularity in the computations. Its computational complexity is $O\left(p_f \times p_B \times I_{\mathrm{MAX}} \times N\right)$, where p_f is the number of pixels in $\boldsymbol{f}$ and p_B is the number of pixels in B. This results in high computational cost as shown in [17]. However, an adequate parallelization strategy can greatly enhance the computational performance of the algorithm, as shown in the following subsection.

3.2. Parallel implementation

In this subsection, we describe AMEEPAR, a parallel version of the AMEE algorithm. We should point out that the proposed parallel algorithm has been implemented as an extension of classic mesh-based parallel techniques [21]. To reduce code redundancy and enhance reusability, our goal was to reuse much of the code for the sequential algorithm in the parallel implementation. For that purpose, we adopted a domain decomposition approach which is similar to the one provided in a user transparent manner by the software architecture introduced by Seinstra et al. [25]. The window SE-based nature of the algorithm introduces some border-handling and overlapping issues that are carefully addressed in the algorithm description given below. This subsection ends with an overview of the operations performed by the parallel algorithm, and a summary of its main contributions over existing parallel hyperspectral imaging approaches.

3.2.1. Partitioning scheme

Two types of parallelism can be exploited in hyperspectral image analysis algorithms [16]: spatial-domain parallelism and spectral-domain parallelism. Spatial-domain parallelism subdivides the image into multiple blocks made up of entire pixel vectors, and assigns one or more blocks to each processing element (PE). On other hand, the spectral-domain parallel paradigm subdivides the whole multi-band data into blocks made up of contiguous spectral bands (sub-volumes), and assigns one or more sub-volumes to each PE. The latter approach breaks the spectral identity of the data because each pixel vector is split amongst several PEs. A spatial-domain decomposition partitioner (SDP) was developed in our application. As a result, no partitioning of data was accomplished in the

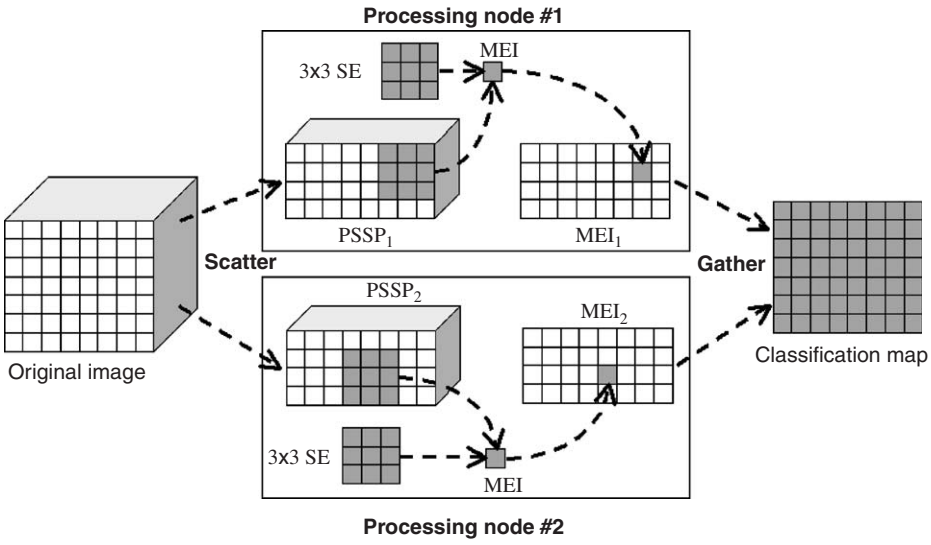

Fig. 2. Example SE-based parallel morphological operation performed using two processing units.

spectral domain, thus preserving the entire spectral signature of each hyperspectral image pixel. There are several reasons that justify the above decision. First, the application of spatial-domain partitioning is a *natural approach* for low level image processing, as many operations require the same function to be applied to a small set of elements around each data element present in the image data structure. A second reason has to do with the cost of inter-processor communication. In spectral-domain parallelism, the SE-based calculations made for each hyperspectral pixel need to originate from several PEs, and thus require intensive inter-processor communication.

In the following, we will refer to each partial data structure produced by the proposed SDP module as a parallelizable spatial/spectral partition (PSSP). This definition should be distinguished from the one given in [25], where the distinct, but related term "parallelizable pattern" is used to describe a parameterized sequential algorithm—hence, a piece of code. Thus, a PSSP can be defined as a hyperspectral data partition that can be processed independently, without communication. In other words, all data accesses in a PSSP must refer to the data local to the processing unit executing the operation. In order to achieve this goal, it may be necessary to handle border effects, and also to introduce an overlapping scatter at each partition that will be included in our definition of PSSP as explained in the following subsection. For illustrative purposes, a pictorial view of a PSSP-based parallel operation to calculate MEI scores (using a square-shaped 3×3 SE and two partitions) is given in Fig. 2. In the example, a final MEI image is formed after combining the individual results produced at each PE. The generalized description of a PSSP given above allows us to maximize code reusability since each PSSP can be analyzed independently at each processing unit. At the same time, communication overhead is minimized, while—for the given parallelization granularity—the available parallelism is fully exploited.

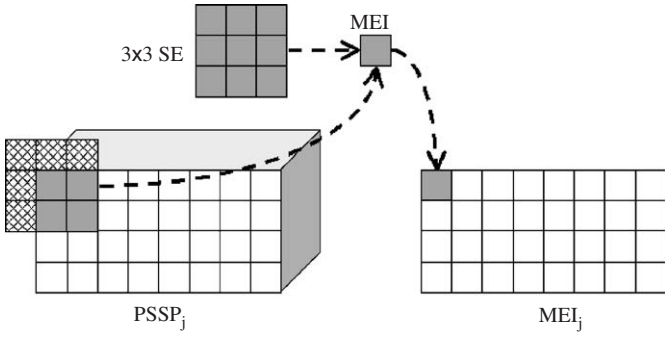

Fig. 3. Border-handling strategy implemented on a PSSP when pixels lying outside the input image domain are required for the SE-based morphological operation.

3.2.2. Border-handling and overlapping function

An important issue in SE-based morphological image processing operations is that accesses to pixels outside the spatial domain of the input image are possible. This is particularly so when the SE is centered on a pixel located in the border of the original image. In sequential implementations, it is common practice to redirect such accesses according to a predefined border handling strategy. In our application, a border handling strategy is adopted when the location of the SE is such that some of the pixel positions in the SE are outside the input image domain (see Fig. 3). In this situation, only those pixels inside the image domain are read for the MEI calculation. This strategy is equivalent to the common mirroring technique used in digital image processing applications, but slightly faster since less pixels are involved in the SE calculation.

Apart from the border handling strategy above, a function to update overlapping parts of partial data structures has been implemented in order to avoid inter-processor communication when the SE computation is split amongst several different processing nodes (see Fig. 4). It should be noted that communi-

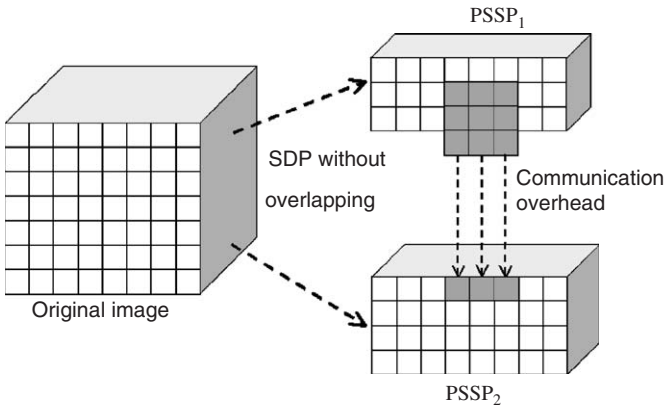

Fig. 4. Communication overhead introduced when the SE-based operation is split among processing nodes.

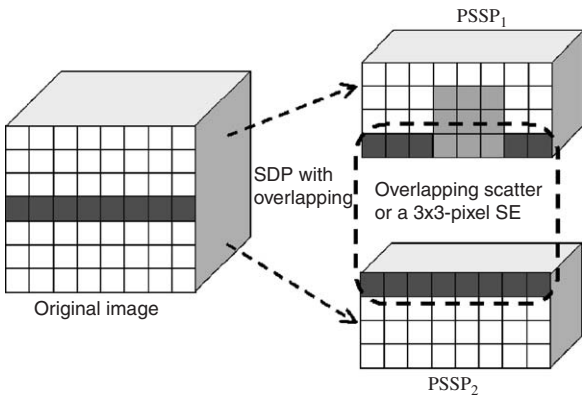

Fig. 5. Overlapping function implemented on adjacent PSSPs to avoid inter-processor communication.

cation overhead in Fig. 4 prevents adequate exploitation of the concept of PSSP in the previous subsection, since processing of a PSSP cannot be entirely accomplished at one single PE without communication. In order to eliminate such overhead, the proposed SDP module has been designed to allow overlapping between adjacent PSSPs, as shown in Fig. 5. A so-called overlap border is added to each of the adjacent PSSPs to avoid accesses outside their domain. The line is filled with pixel values obtained from neighboring nodes in the logical CPU grid. It should be noted that Fig. 5 gives a simplified view, as some steps of the operation are not shown. For example, depending on how many adjacent PSSPs are involved in the parallel computation of a kernel, it may be necessary to place an overlap border around each PSSP to completely avoid inter-processor communication. In this regard, it is important to emphasize that the amount of redundant information introduced by the overlapping scatter depends on the size of B, the SE used in the morphological operations. However, our implementation of the AMEE algorithm always uses a constant 3×3-pixel SE through the different iterations (see Section 3.2). Instead of increasing the size of the SE to consider a larger spatial neighborhood, we replace the original image cube $\boldsymbol{f}$ or, equivalently, the local PSSP in parallel processing, by the resulting cube after applying a dilation operation using B (see step 3 of the AMEE algorithm). This allows us to perform multi-scale analysis of the data without increasing the overlap border size between subsequent iterations [21]. As will be shown in experiments, the fraction of redundant information introduced in the system by a constant 3×3-pixel structuring element is insignificant when compared with the total volume of information to be processed. As a result, the computational time to process redundant information is insignificant compared to the total processing time. A more detailed discussion on the above issues will be presented in Section 4.

3.2.3. Sequence of operations

The parallel implementation of the AMEE algorithm (AMEEPAR) relies entirely on the spatial-domain decomposition scheme implemented by the SDP, which also acts as a root node in charge of all I/O operations. The partitioner has been implemented so that it scatters hyperspectral data structures without creating partial structures at the root. Therefore, although we will refer to individual PSSPs in our description, it is important to note that such structures are not buffered before transmission in order to avoid wasting memory resources and compute power. The proposed parallel algorithm has been implemented in the C++ programming language using calls to message passing interface (MPI). We make use of MPI *derived datatypes* to directly scatter hyperspectral data structures, which may be stored non-contiguously in memory, in a single communication step [23]. We describe next the operations executed by the SDP module in the proposed parallel implementation.

AMEEPAR algorithm

Inputs: N-D image cube $\boldsymbol{f}$, SE B, number of iterations I_{MAX}, number of endmembers p.

Outputs: 2-D image which contains a classification label for each pixel $\boldsymbol{f}(x, y)$ in the original image.

1. Scatter K partial data structures $\{\mathrm{PSSP}_j\}_{j=1}^{K}$ of the original image $\boldsymbol{f}$ (with their corresponding overlap borders) by indicating all partial data structure elements which are to be accessed and sent to each of the K workers. A standard non-overlapping scatter, followed by overlap communication *before* the filtering, is adopted to have all data available in the overlap border areas (thus sending all border data beforehand, but only once).
2. Using parameters B, I_{MAX} and p, each worker executes the sequential AMEE algorithm locally at each processor for the corresponding PSSP_j, obtaining a classification label for each pixel in the PSSP_j.
3. Each worker sends the label associated with each local pixel to the master, which gathers all the individual results and forms the final 2-D classification image.

It should be noted that the parallel implementation strategy adopted in the design of AMEEPAR is not new in itself. However, the application of this parallelization strategy to hyperspectral imaging opens new perspectives which have not been yet explored in the design of efficient hyperspectral algorithms, as revealed by our experimental assessment of techniques in Section 4.

Before concluding this section, we summarize the main contributions of the proposed parallel algorithm with regards to conventional approaches in Section 2. Firstly, AMEEPAR is the only parallel hyperspectral algorithm available that integrates spatial and spectral information simultaneously, which is essential to obtain accurate classification results as will be demonstrated by experiments in this paper. Both the S-PCT and D-ISODATA algorithms rely on spectral clustering of the data alone. As opposed to the two algorithms above, which treat all the pixel vectors in the data as pure signatures under a full-pixel classification assumption, AMEEPAR was specifically designed to optimize a morphological endmember extraction procedure that allows characterization of mixed pixels (which are predominant in hyperspectral imagery). Most importantly, the AMEEPAR algorithm falls into the category of "pleasingly parallel'' problems, because there is no dependence between the calculations made at each PSSP_j and only minimal communication is required for the entire calculation. Specifically, communication only takes place at the beginning (scatter) and at the end (gather) of the process. Although conventional algorithms also use a master–slave decomposition, they show a much higher degree of data dependency, with several synchronization steps as described in Section 2. As a result, it may not always be possible to have the master and the slaves work simultaneously, in particular, when the workers wait for calculations where the complexity depends on the specific properties of the image data (e.g., step 6 in the D-ISODATA algorithm). On the contrary, all the computations in the AMEEPAR algorithm are characterized by their regularity and locality within each PSSP_j. These properties are accomplished at the expense of introducing a small fraction of redundant information which, on the other hand, enhances code reusability and minimizes inter-processor communication. A quantitative and comparative performance analysis of the proposed algorithm with regards to conventional parallel techniques is given in the following section.

4. Parallel performance evaluation

This section provides an assessment of parallel hyperspectral algorithms in providing significant performance gains without loss of accuracy in the analysis of real high-dimensional data. The section is organized as follows. First, we provide an overview of the parallel computing architectures used for evaluation purposes. Second, a quantitative assessment of the proposed parallel approach, in comparison with other parallel hyperspectral image analysis approaches, is provided. Third, comparisons between doing border exchange between processors and not doing this (i.e., computing the border locally) are also provided to objectify the impact of redundant information introduced by the proposed parallel approach. Finally, other important aspects of the parallel implementations are analyzed, including impact of inter-processor communication, coordination and load balance. In all cases, speedup characteristics are discussed in light of the accuracy of classification results, thus providing a study of computational cost versus classification performance that may help image analysts in selection of parallel hyperspectral algorithms for specific applications.

4.1. Parallel computing architectures

The parallel algorithms described in this work have been implemented and tested on two different high-performance parallel computers. The first one is the SGI Origin 2000 multicomputer located at European Center for Parallelism of Barcelona (CEPBA). The system used is composed of 64 MIPS R10000 processors at 250 MHz (each one with 4 MB of cache) and 12 Gb of main memory, interconnected via 1.2 Gbps communication network. The theoretical peak performance of the system is 32 Gflops. At the time of measurement, the nodes ran the Irix 6.5 operating system, with single-kernel architecture. The software was compiled using version 7.3.1.2 of the MIPSpro compiler suite. The second 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 [7]. 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.

4.2. Quantitative and comparative analysis of parallel hyperspectral imaging algorithms

Various code performance tests were carried out on the parallel computing systems above. Before empirically investigating the performance of parallel hyperspectral imaging algorithms,

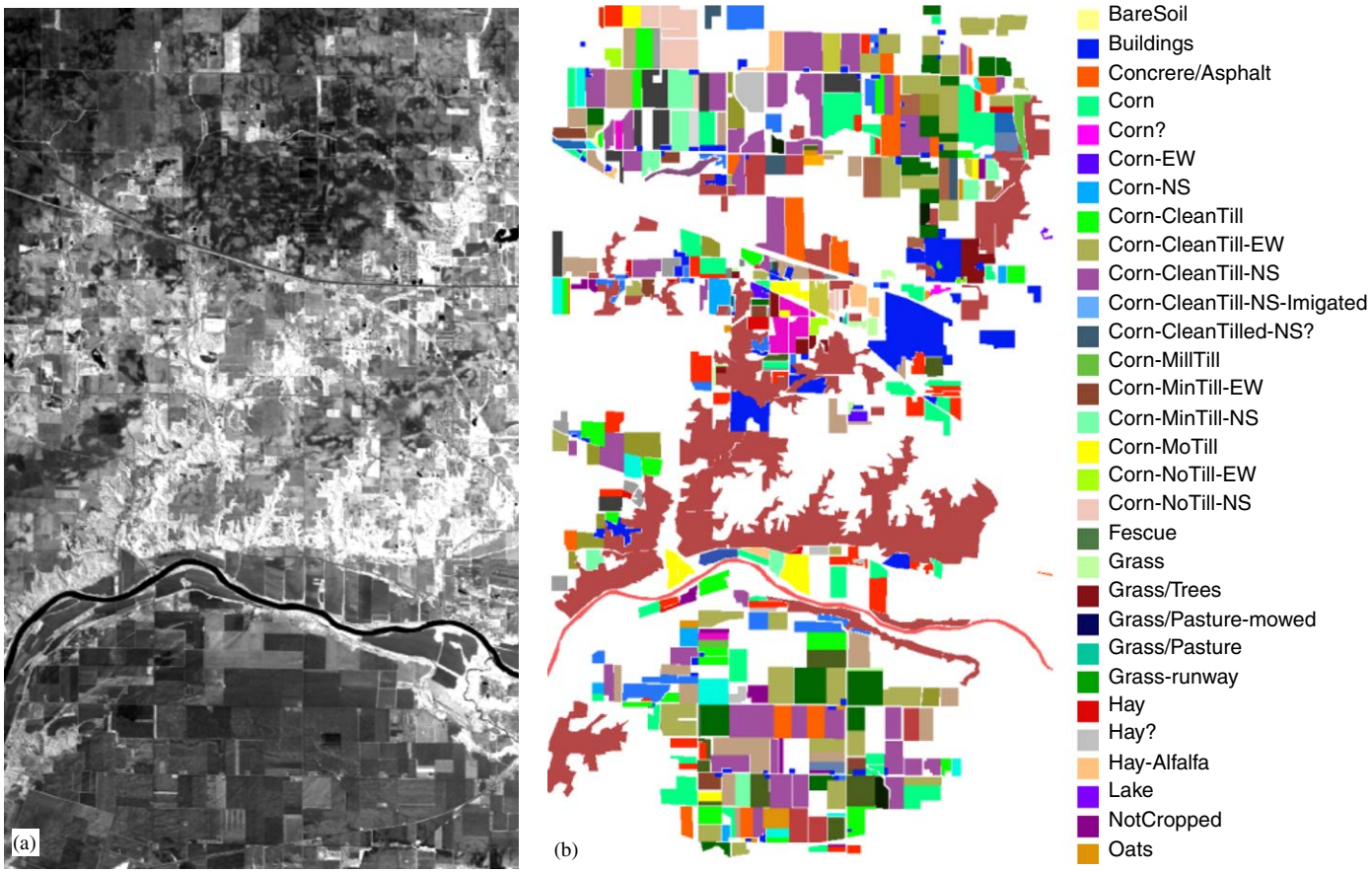

Fig. 6. (a) Spectral band at 587 nm wavelength of an AVIRIS scene comprising agricultural and forest features at Indian Pines test site, Indiana. (b) Ground-truth map with thirty mutually-exclusive land-cover classes.

we briefly describe a real hyperspectral image scene that will be used in experiments. The scene was collected by the AVIRIS sensor, and is characterized by very high spectral resolution (224 narrow spectral bands in the range 0.4–2.5 μm and moderate spatial resolution 20-m pixels). It was gathered over the Indian Pines test site in Northwestern Indiana, a mixed agricultural/forested area, early in the growing season. The data set represents a very challenging classification problem. The primary crops of the area, mainly corn and soyabeans, were very early in their growth cycle with only about 5% canopy cover. This fact makes most of the scene pixels highly mixed in nature. Discriminating among the major crops under this circumstances can be very difficult, a fact that has made this scene a universal and extensively used benchmark to validate classification accuracy of hyperspectral imaging algorithms [13]. Fig. 6(a) shows the spectral band at 587 nm of the original scene and Fig. 6(b) shows the ground-truth map, in the form of a class assignment for each labeled pixel with 30 mutually exclusive ground-truth classes. Part of these data, including ground-truth, are available online (from http://dynamo.ecn.purdue.edu/~biehl/MultiSpec). As a result, people interested in the proposed algorithms can reproduce our results and conduct their experiments to exploit various algorithms.

For perspective, Table 1 reports the overall classification accuracy scores produced by the considered parallel algorithms on the AVIRIS scene. It should be noted that a previous discussion on classification accuracy (separately from parallel processing) is important in order to perform a preliminary validation of the considered algorithms, where input parameters were set as follows. For the D-ISODATA algorithm, a tolerance threshold $t = 0.05$ was used after experimental results in [6]. For the AMEEPAR, a constant 3×3-pixel SE was adopted, and different values for the maximum number of iterations were tested. In all cases, we assumed that the maximum number of clusters/endmembers to be detected by the algorithms was set to $p = 30$ after calculating the intrinsic dimensionality of the data using the Harsanyi–Farrand–Chang (HFC) method in [4]. As shown by Table 1, the AMEEPAR algorithm produced higher classification scores than those found by S-PCT and D-ISODATA, in particular, when parameter I_{MAX} was set to 7 iterations (an overall accuracy of more than 90%. was achieved). This table provides an objective confirmation of our introspection: that the incorporation of spatial and spectral information simultaneously can greatly enhance classification scores achieved using the spectral information only.

To empirically investigate the scaling properties of the considered parallel algorithms, Fig. 7 plots the speedup factors as a function of the number of available processors at both the SGI Origin 2000 and Thunderhead computers. Results in Fig. 7 reveal that the performance drop from linear speedup in both

Table 1
Overall classification accuracies (in percentage) achieved by the parallel algorithms

S-PCT	D-ISODATA	AMEEPAR			
		$I_{MAX} = 1$	$I_{MAX} = 3$	$I_{MAX} = 5$	$I_{MAX} = 7$
82.25	69.84	75.23	81.94	87.95	90.02

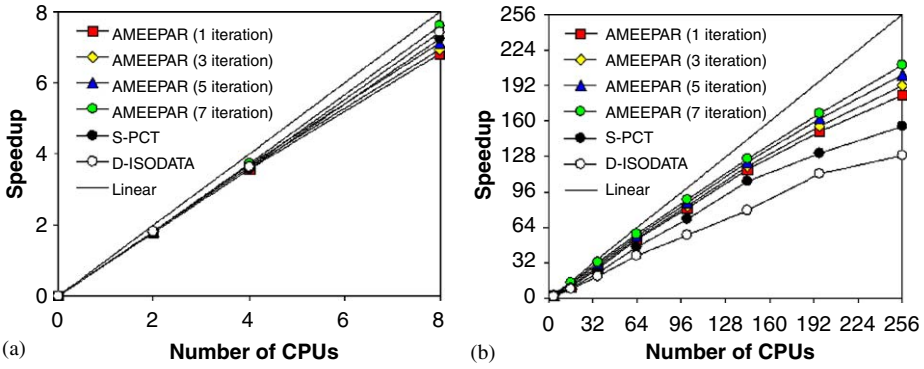

Fig. 7. Parallel performance of hyperspectral algorithms on SGI Origin 2000 (a) and Thunderhead (b).

S-PCT and D-ISODATA algorithms increases significantly as the number of processors increase. This is due to the data dependencies and sequential calculations present in the algorithms, i.e., steps 5 and 6 (covariance matrix and eigenvector calculations) in the S-PCT, and step 6 (split and merge) in the D-ISODATA. It should be noted that the complexity of the eigenvector calculations is related to the number of spectra used in the problem, while the split and merge is highly dependent on the inherent complexity of the input data. Since the AVIRIS scene represents a large-sized problem dominated by mixed pixels, there is sufficient computation to gain an impact from a large number of processors in parallel computations. However, the complexity of sequential steps seems to dominate that of parallel computations when the number of processors is high (in many cases, parallel computations cannot start until sequential calculations are completed). As a result, the performance gain achieved by both S-PCT and D-ISODATA begins to drop off significantly as the number of processors increase.

On the other hand, although the proposed AMEEPAR algorithm introduces redundant calculations that may slow down the computation, we can observe in Fig. 7 that the measured speedups tend to be higher for large values of I_{MAX}, a fact that reveals that the proposed scheme scales better as the size of the problem increases. As a result, the dominant issue in AMEEPAR is problem size, which makes the algorithm very appealing for high-dimensional imaging applications. Results in Fig. 7 reveal that the algorithm obtains good scalability in both the SGI Origin 2000 and Thunderhead parallel computers. For a maximum number of 7 iterations, the algorithm achieved a speedup of 7.62 for 8 processors in the SGI Origin 2000 system (see Fig. 7(a)). On other hand, AMEEPAR performed within 10% of linear speedup using 64 processors (see Fig. 7(b)). Although for a high number of processors graphs flatten out a little, due to the relatively short execution times, they still outperform those found for the other tested algorithms. For illustrative purposes, Table 3 reports the algorithm processing times in both the SGI Origin 2000 and Thunderhead. Interestingly, the table shows that the utilization of a moderate number of processors on Thunderhead allowed near real-time processing of the AVIRIS scene—less than 4 minutes were required to produce a classification map with over 90% accuracy using 64 processors.

In order to relate parallel performance to classification accuracy, we now compare results in Tables 1 and 2 to discuss the cost of parallel computation in light of the overall classification scores achieved by each algorithm. As shown by Table 2, processing times obtained for both S-PCT and D-ISODATA generally prevented near real-time exploitation of the AVIRIS data using a reduced number of processors. For instance, S-PCT required as many as 100 processors to classify the scene in less than ten minutes, while D-ISODATA required 144 processors to finish in about the same time. In both cases, classification scores were only moderate as shown by Table 1. With 256 processors, S-PCT required about 4 min to complete the calculations, while D-ISODATA required more than 6 min. On the contrary, the utilization of 256 processors in the Thunderhead system allowed AMEEPAR to obtain a better classification score in just 1 min, which is a good response time given the high dimensionality and complexity of the data. To conclude this subsection, it is also interesting to compare the results produced by the parallel algorithms on the two considered architectures. In general, it was observed that the algorithms performed better on Thunderhead, where moderately accurate results could be accomplished in around 10 min using only 16 processors. As reported in Table 2, running the parallel algorithms on the SGI

Table 2
Execution times in seconds achieved by the parallel algorithms for different numbers of processors

Parallel computer	# CPUs	S-PCT	D-ISODATA	AMEEPAR			
				$I_{MAX} = 1$	$I_{MAX} = 3$	$I_{MAX} = 5$	$I_{MAX} = 7$
SGI Origin 2000	1	49781	60570	5867	8934	13567	19030
	2	27656	33464	3278	4936	7454	10399
	4	13867	16595	1639	2461	3677	5088
	8	6885	8152	862	1285	1905	2497
Thunderhead	1	41239	49912	3867	6423	9456	13188
	4	13521	21330	1427	2177	3031	4083
	16	4314	5907	354	571	716	927
	36	1759	2428	140	220	303	394
	64	884	1299	73	118	168	226
	100	572	865	48	77	109	148
	144	392	630	33	54	77	105
	196	314	444	26	41	58	79
	256	265	386	21	33	47	63

Origin system always resulted in higher execution times than those found in the commodity cluster for the same number of processors (e.g., when 4 processors were used).

4.3. Quantifying the impact of redundant computations in the parallel morphological algorithm

Although experiments in the previous subsection revealed that the proposed algorithm can achieve significant speedups, an important issue to fully understand the parallel properties of the algorithm is to objectify the impact of using an overlap border to reduce inter-processor communications and *not* using such strategy, but using a standard non-overlapping scatter followed by overlap communication in the SE-based operation for *every* pixel, as shown in Fig. 4. The exchange of border data is performed in four steps as explained in [23], i.e., each node first transmits a subset of its local partial data to the neighboring node on the right within a logical CPU grid organization. After a node has received a full block of data, it transmits a subset of its local partial data to the neighboring node on the left (in both steps, non-contiguous blocks of data are involved). Finally, the border data is exchanged in upward and downward direction, in both cases involving contiguous data blocks only.

Table 3 shows the total time spent by AMEEPAR in communications and computations in the two considered implementations, for a case study of 7 algorithm iterations. It can be seen that the cost of communications tends to dominate that of computations as the number of processors increases. On other hand, the table reveals that the increase in processing times when the border is computed locally is not significant. This is because the volume of computations involved in hyperspectral imaging is very high. It should be noted that the relationship between the number of iterations and the amount of redundant information introduced by the overlap border is linear, because the algorithm always uses a constant 3×3-pixel SE throughout the different iterations. We have experimentally tested that the fraction of redundant pixels that had to be introduced in the parallel computations involving the entire AVIRIS scene in one algorithm iteration was 0.35% of the size of the original image (275 Mb). Subsequently, only a fraction of 2.45% of the data was replicated for seven algorithm iterations. From results in Table 3, we can conclude that communication times in the proposed redundant computation-based implementation are always small compared to computation times in both parallel architectures. As a result, the granularity of problem decomposition (defined as the ratio of computation to communication) is much better when an overlap border is introduced. This result is not surprising, especially in light of the proposed decomposition technique, which aims at minimizing inter-processor communication during morphological processing. We must also note that communication overheads in Table 3 are very high, in particular, for the SGI Origin 2000 experiments (even for the "without border exchange" case). This is due to several reasons, most notably, the large amount of data involved in each overlap communication. Also, although we carefully optimized our parallel code, it might still contain redundant operations that could be avoided in future versions. Finally, we emphasize that the execution of jobs on the SGI Origin 2000 facility is based on a queue system.

Although we requested a time-window to run our experiments in the Origin system, there is a possibility that other jobs may have been active at the time of measurements, thus affecting the measured communication times.

To conclude this subsection, Fig. 8 examines the scalability of the implementation based on data communication between processors, using different numbers of algorithm iterations. One can clearly see that, while the curves for the implementation based on redundant computations were closer to linear performance (see Fig. 7), the implementation based on border data exchange did not scale as well. In particular, nonlinear changes in performance (which may be due to interruptions in data transfers, packetization, etc.) were observed. This seems to indicate that the proposed overlapping scatter-based

Table 3
Computation/communication times in seconds achieved by AMEEPAR (with and without border exchange) for a case study of 7 algorithm iterations

Parallel computer	# CPUs	Without border exchange		With border exchange	
		Computation	Communication	Computation	Communication
SGI Origin 2000	2	10365	24	10112	215
	4	5047	31	4924	233
	8	2460	27	2399	251
Thunderhead	4	4074	9	3975	116
	16	914	13	892	133
	36	383	11	374	168
	64	212	14	207	198
	100	131	17	128	215
	144	90	15	86	234
	196	67	12	64	241
	256	52	11	50	265

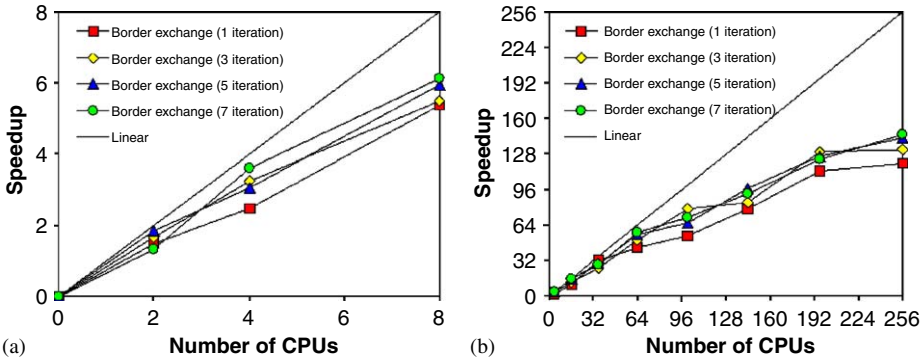

Fig. 8. Parallel performance of a border exchange implementation on the SGI Origin 2000 (a) and Thunderhead (b).

implementation offers a more reliable solution in hyperspectral imaging applications.

4.4. Study of load balance

To conclude our analytical study of parallel performance, we evaluate the tested algorithms considering an important new topic: the load balance. For that purpose, Table 4 shows the imbalance scores [15] achieved by the different algorithms, where the imbalance is defined as $D = T_{MAX}/T_{MIN}$, with T_{MAX} and T_{MIN}, respectively, denoting the maxima and minima processor run times. Therefore, perfect balance is achieved when $D = 1$. In the table, we display the imbalance considering all processors, D_{All}, and also considering all processors but the root, D_{Minus}. It is clear from Table 4 that AMEEPAR provided almost the same results for both D_{All} and D_{Minus}. Most importantly, the above remark seems to be true regardless of the number of processors used in the computation and the number of iterations executed by the algorithm. For the S-PCT and D-ISODATA, however, load balance was much better when the root processor was not included. This means that the master node has high load, which may be due to the sequential computations included in both algorithms. Despite the fact that conventional hyperspectral imaging algorithms do not take into account the spatial information explicitly into the computations (which has traditionally been perceived as an advantage for the development of parallel implementations), the adopted PSSP-based parallel framework can be considered as more "pleasingly parallel" for hyperspectral imaging because it reduces sequential computations at the root node, and only involves minimal communication between the parallel tasks, namely, at the beginning and ending of such tasks.

5. Conclusions and future work

The aim of this paper has been the examination of different parallel strategies for hyperspectral image analysis on high performance computers, with the purpose of evaluating the possibility of obtaining results in valid response times and with adequate reliability for the remote sensing environment where these techniques are intended to be applied. It has been shown and proven that parallel computing at the massively parallelism level, supported by message passing, provides a unique framework to accomplish the above goals. For this purpose, computing systems made up of arrays of commercial off-the-shelf computing hardware are a cost-effective way of exploiting this sort of parallelism in remote sensing applications. In particular, a redundant computation-based implementation (derived

Table 4
Maxima and minima processor run times and load balancing rates for the parallel algorithms

Parallel computer	# CPUs	S-PCT		D-ISODATA		AMEEPAR	
		D_{All}	D_{Minus}	D_{All}	D_{Minus}	D_{All}	D_{Minus}
SGI Origin 2000	2	1.31	1.05	1.43	1.09	1.11	1.00
	4	1.26	1.04	1.39	1.08	1.15	1.02
	8	1.22	1.03	1.36	1.05	1.12	1.01
Thunderhead	4	1.47	1.08	1.74	1.13	1.15	1.04
	16	1.35	1.05	1.62	1.08	1.10	1.02
	36	1.32	1.05	1.57	1.10	1.09	1.04
	64	1.30	1.04	1.42	1.04	1.11	1.03
	100	1.23	1.03	1.39	1.07	1.07	1.01
	144	1.24	1.04	1.28	1.05	1.10	1.02
	196	1.21	1.04	1.31	1.07	1.05	1.03
	256	1.19	1.03	1.27	1.06	1.04	1.01

as an extension of the classic mesh-based parallel processing paradigm) allowed us to parallelize a morphological approach for hyperspectral imaging that successfully integrates the spatial and spectral information in the data. This implementation, aimed at minimizing inter-processor communication and maximizing load balance, outperforms other parallel approaches used for hyperspectral image classification. The proposed implementation can be ported to any type of distributed memory system, in particular, to a Beowulf cluster of PCs, an architecture that has gained popularity in the last few years due to the chance of building a "high performance system" at a reasonable cost. Experimental results in this paper suggest that the parallel morphological algorithm provides adequate results in both the quality of the solutions and the time to obtain them, in particular, when it is implemented on a Beowulf cluster. Further, the proposed PSSP-based parallel framework offers an unprecedented opportunity to explore methodologies in other fields (e.g., data mining) that previously looked to be too computationally intensive for practical applications due to the immense files common to remote sensing problems. Combining this readily available computational power with the new sensor instruments may introduce major changes in the systems currently used by NASA and other agencies for exploiting Earth and planetary remotely sensed data. As future work, we plan to implement parallel hyperspectral imaging algorithms on other high-performance parallel computing architectures, such as the Medusa Beowulf cluster at NASA/GSFC or the Bull NovaScale 5160 multicomputer at CEPBA. We are also working toward field programmable gate array (FPGA)-based implementations, which may allow us to fully accomplish the goal of near real-time processing of hyperspectral image data, with potential applications in exploitation-based on-board hyperspectral image compression and analysis.

Acknowledgements

This research was supported by the European Commission through the project entitled "Generation of test images to validate the performance of endmember extraction and hyperspectral unmixing algorithms," part of the DLR project "HySens—DAIS/ROSIS Imaging Spectrometers at DLR" (contract number HPRI-1999-00057). Additional funding from Junta de Extremadura (local government) through 2PR03A026 project is also gratefully acknowledged. The authors would like to thank Anthony Gualtieri, James Tilton and John Dorband for many helpful discussions, and also for their collaboration in the development of experiments using the Thunderhead Beowulf cluster. They would also like to state their appreciation for Professors Mateo Valero and Francisco Tirado; without their collaboration and support this research effort could not have been possible. Troubleshooting guidelines by Glen Gardner (Thunderhead system administrator) and Javier Bartolome (SGI Origin 2000 administrator) are also greatly appreciated. Finally, we thank David Landgrebe and Jacqueline Le Moigne for providing the full AVIRIS data over the Indian Pines region. The first author would like to thank support received from the Spanish Ministry of Education and Science through Fellowship PR2003-0360, which allowed him to conduct research as postdoctoral associate at NASA's Goddard Space Flight Center. Last but not least, the authors would like to thank the three anonymous reviewers for their helpful comments and suggestions, which greatly helped us to improve the implementation of techniques and the presentation of the manuscript.

References

[1] T. Achalakul, S. Taylor, A distributed spectral-screening PCT algorithm, J. Parallel Distrib. Comput. 63 (2003) 373–384.

[2] G. Aloisio, M. Cafaro, A dynamic earth observation system, Parallel Comput. 29 (2003) 1357–1362.

[3] R. Brightwell, L.A. Fisk, D.S. Greenberg, T. Hudson, M. Levenhagen, A.B. Maccabe, R. Riesen, Massively parallel computing using commodity components, Parallel Comput. 26 (2000) 243–266.

[4] C.-I. Chang, Hyperspectral Imaging: Techniques for Spectral Detection and Classification, Kluwer Academic Publishers, Dordrecht, 2003.

[5] L. Chen, I. Fujishiro, K. Nakajima, Optimizing parallel performance of unstructured volume rendering for the Earth Simulator, Parallel Comput. 29 (2003) 355–371.

[6] M.K. Dhodhi, J.A. Saghri, I. Ahmad, R. Ul-Mustafa, D-ISODATA: a distributed algorithm for unsupervised classification of remotely sensed data on network of workstations, J. Parallel Distrib. Comput. 59 (1999) 280–301.
[7] J. Dorband, J. Palencia, U. Ranawake, Commodity computing clusters at Goddard Space Flight Center, J. Space Commun. 1 (3) (2003) (Available online: http://satjournal.tcom.ohiou.edu/pdf/Dorband.pdf).
[8] M. Gil, C. Gil, I. Garcia, The load unbalancing problem for region growing image segmentation algorithms, J. Parallel Distrib. Comput. 63 (2003) 387–395.
[9] R.O. Green, et al., Imaging spectroscopy and the airborne visible/infrared imaging spectrometer AVIRIS, Remote Sens. Environ. 65 (1998) 227–248.
[10] J.A. Gualtieri, J.C. Tilton, Hierarchical segmentation of hyperspectral data, XI NASA/Jet Propulsion Laboratory Airborne Earth Science Workshop, Pasadena, CA, USA, 2002.
[11] K.A. Hawick, P.D. Coddington, H.A. James, Distributed frameworks and parallel algorithms for processing large-scale geographic data, Parallel Comput. 29 (2003) 1297–1333.
[12] N. Keshava, J.F. Mustard, Spectral unmixing, IEEE Signal Process. Mag. 19 (2002) 44–57.
[13] D.A. Landgrebe, Signal Theory Methods in Multispectral Remote Sensing, Wiley, Hoboken, NJ, 2003.
[14] J. Le Moigne, J.C. Tilton, Refining image segmentation by integration of edge and region data, IEEE Trans. Geosci. Remote Sensing 33 (1995) 605–615.
[15] M.J. Martín, D.E. Singh, J.C. Mouriño, F.F. Rivera, R. Doallo, J.D. Bruguera, High performance air pollution modeling for a power plan environment, Parallel Comput. 29 (2003) 1763–1790.
[16] C. Nicolescu, P. Jonker, A data and task parallel image processing environment, Parallel Comput. 28 (2002) 945–965.
[17] A. Plaza, Development, validation and testing of a new morphological method for hyperspectral image analysis that integrates spatial and spectral information, Ph.D. Dissertation, Computer Science Department, University of Extremadura, Spain, 2002.
[18] A. Plaza, P. Martinez, R.M. Perez, J. Plaza, Spatial/spectral endmember extraction by multidimensional morphological operations, IEEE Trans. Geosci. Remote Sensing 40 (9) (2002) 2025–2041.
[19] A. Plaza, P. Martinez, R.M. Perez, J. Plaza, A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data, IEEE Trans. Geosci. Remote Sensing 42 (3) (2004) 650–663.
[20] A. Plaza, P. Martinez, R.M. Perez, J. Plaza, A new approach to mixed pixel classification of hyperspectral imagery based on extended morphological profiles, Pattern Recog. 37 (2004) 1097–1116.
[21] A. Plaza, D. Valencia, P. Martínez, J. Plaza, R.M. Perez, Parallel implementation of algorithms for endmember extraction from AVIRIS hyperspectral imagery, Proceedings of the XIII NASA/Jet Propulsion Laboratory Airborne Earth Science Workshop, Pasadena, CA, USA, 2004.
[22] K. Sano, Y. Kobayashi, T. Nakamura, Differential coding scheme for efficient parallel image composition on a PC cluster system, Parallel Comput. 30 (2004) 285–299.
[23] F.J. Seinstra, D. Koelma, P-3PC: a point-to-point communication model for automatic and optimal decomposition of regular domain problems, IEEE Trans. Parallel Distrib. Systems 13 (2002) 758–768.
[24] F.J. Seinstra, D. Koelma, User transparency: a fully sequential programming model for efficient data parallel image processing, Concurrency and Computation: Practice and Experience 16 (2004) 611–644.
[25] F.J. Seinstra, D. Koelma, J.M. Geusebroek, A software architecture for user transparent parallel image processing, Parallel Comput. 28 (2002) 967–993.
[26] P. Soille, Morphological Image Analysis: Principles and Applications, second ed., Springer, Berlin, 2003.
[27] P. Wang, K.Y. Liu, T. Cwik, R.O. Green, MODTRAN on supercomputers and parallel computers, Parallel Comput. 28 (2002) 53–64.

Antonio Plaza received his Ph.D. degree in Computer Science from the University of Extremadura, Spain, in 2002, where he is currently an Associate Professor with the Computer Science Department. He has also been Visiting Researcher with the University of Maryland, NASA Goddard Space Flight Center and Jet Propulsion Laboratory. His main research interests include the development and efficient implementation of high-dimensional data algorithms on parallel homogeneous and heterogeneous computing systems and hardware-based computer architectures. He has authored or co-authored more than one hundred publications including journal papers, book chapters and peer-reviewed conference proceedings, and currently serves as regular manuscript reviewer for more than 15 highly cited journals in the areas of parallel and distributed computing, computer architectures, pattern recognition, image processing and remote sensing. He is editing a book on "High-Performance Computing in Remote Sensing" (with Prof. Chein-I Chang) for Chapman & Hall/CRC Press.

David Valencia is a Resarch Associate at the University of Extremadura, Spain, where he is currently pursuing his M.Sc. degree in Computer Science. His research interests are in the development of parallel implementations of algorithms for high-dimensional data analysis, with particular emphashis on commodity cluster-based (homogeneous and heterogeneous) systems and hardware-based architectures, including systolic arrays and field programmable gate arrays. He is also involved in the design and testing of large-scale distributed heterogeneous computing platforms.

Javier Plaza received his M.Sc. degree in Computer Science from the University of Extremadura, Spain, in 2002, where he is currently an Assistant Professor. His current research work is focused on the development of efficient implementations of neural network-based algorithms for analysis and classification of hyperspectral scenes. He is also involved in the design and configuration of homogeneous and fully heterogeneous parallel computing architectures for high-performance scientific applications. Other major research interests include telecommunications, networking and configuration and training of neural network architectures for specific applications.

Pablo Martínez is a Professor of Computer Science at the University of Extremadura, Spain, since 1985. He is the Head Scientist of the Neural Networks and Signal Processing Group (GRNPS). He has held Visiting Researcher positions at the NASA Goddard Space Flight Center and the Department of Electrical Engineering, University of Maryland, College Park, MD. His research interests are in remote sensing, digital image analysis, parallel and distributed computing, hardware-based architectures, operating systems management and configuration, and neural network-based pattern recognition.