Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 68 (2008) 93–111
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Parallel techniques for information extraction from hyperspectral imagery using heterogeneous networks of workstations

Antonio J. Plaza*

Department of Technology of Computers and Communications, Escuela Politecnica de Caceres, University of Extremadura, Avda. de la Universidad s/n, E-10071 Caceres, Spain

Received 9 January 2006; received in revised form 11 July 2007; accepted 25 July 2007
Available online 22 August 2007

Abstract

Recent advances in space and computer technologies are revolutionizing the way remotely sensed data is collected, managed and interpreted. In particular, NASA is continuously gathering very high-dimensional imagery data from the surface of the Earth with hyperspectral sensors such as the Jet Propulsion Laboratory's airborne visible-infrared imaging spectrometer (AVIRIS) or the Hyperion imager aboard Earth Observing-1 (EO-1) satellite platform. The development of efficient techniques for extracting scientific understanding from the massive amount of collected data is critical for space-based Earth science and planetary exploration. In particular, many hyperspectral imaging applications demand real time or near real-time performance. Examples include homeland security/defense, environmental modeling and assessment, wild-land fire tracking, biological threat detection, and monitoring of oil spills and other types of chemical contamination. Only a few parallel processing strategies for hyperspectral imagery are currently available, and most of them assume homogeneity in the underlying computing platform. In turn, heterogeneous networks of workstations (NOWs) have rapidly become a very promising computing solution which is expected to play a major role in the design of high-performance systems for many on-going and planned remote sensing missions. In order to address the need for cost-effective parallel solutions in this fast growing and emerging research area, this paper develops several highly innovative parallel algorithms for unsupervised information extraction and mining from hyperspectral image data sets, which have been specifically designed to be run in heterogeneous NOWs. The considered approaches fall into three highly representative categories: clustering, classification and spectral mixture analysis. Analytical and experimental results are presented in the context of realistic applications (based on hyperspectral data sets from the AVIRIS data repository) using several homogeneous and heterogeneous parallel computing facilities available at NASA's Goddard Space Flight Center and the University of Maryland.

Keywords: Heterogeneous computing; Parallel information extraction; Hyperspectral imaging; Parallel performance evaluation

1. Introduction

Imaging spectroscopy, also known as hyperspectral remote sensing, is an imaging technique capable of identifying materials and objects in the air, land and water on the basis of the unique reflectance patterns that result from the interaction of solar energy with the molecular structure of the material [5]. Recent advances in sensor technology have led to the development of so-called hyperspectral instruments, capable of collecting hundreds of images, corresponding to different spectral channels, for the same area on the surface of the Earth [13]. As a result, each pixel (vector) in the scene has an associated spectral signature or "fingerprint" that uniquely characterizes the underlying objects, as shown by Fig. 1. The wealth of spectral information provided by last-generation sensors has opened ground-breaking perspectives in many applications, including environmental modeling and assessment, target detection for military and defense/security deployment, urban planning and management studies, risk/hazard prevention and response including wild-land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination. Many of these applications require timely responses for swift decisions which depend upon high computing performance of algorithm analysis [2,30]. However, the design of such algorithms introduces several challenges. In particular,

* Fax: +34 927 257203.
E-mail address: aplaza@unex.es.

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.07.010

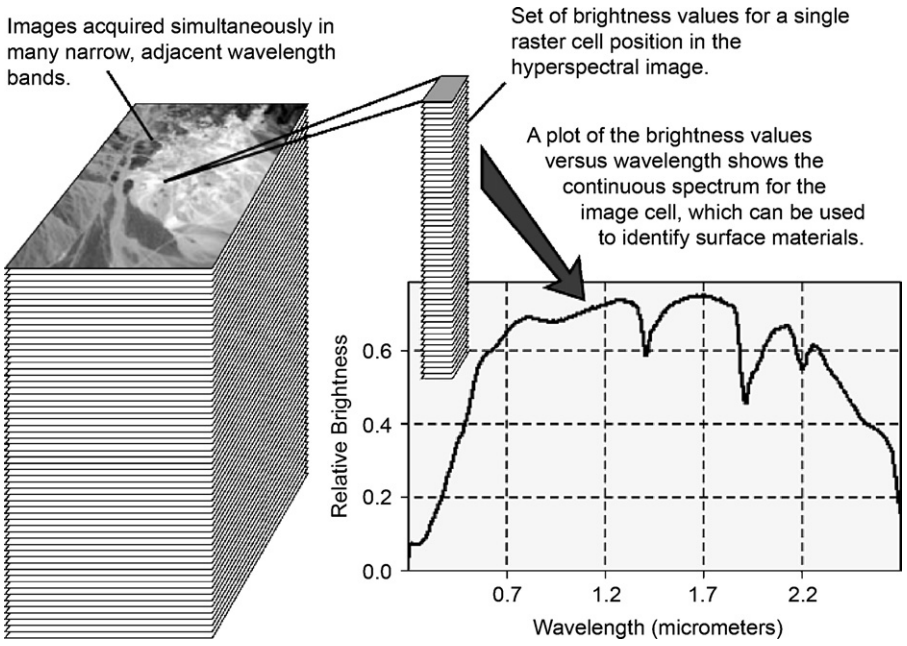

Fig. 1. The concept of hyperspectral imaging.

the price paid for the wealth of spatial and spectral information provided by hyperspectral sensors is the enormous amounts of data that they generate. For instance, NASA is continuously gathering imagery data with Earth-observing sensors such as Jet Propulsion Laboratory's airborne visible-infrared imaging spectrometer (AVIRIS) [13], or the Hyperion imager aboard Earth Observing-1 spacecraft. The incorporation of hyperspectral sensors on airborne/satellite platforms is currently producing a nearly continual stream of high-dimensional data (it is estimated that NASA collects and sends to Earth more than 950 GB of hyperspectral data every day). Since hyperspectral data archives are ever growing, the need to develop fast, unsupervised techniques for near real-time information extraction has become a highly desired goal yet to be fully accomplished.

To address the computational needs introduced by multidimensional imaging applications, a few efforts have been directed towards the incorporation of high-performance computing models and platforms in remote sensing missions [7,11,16]. With the aim of creating a cost-effective parallel computing system from commodity components to satisfy specific computational requirements for the Earth and space sciences community, the Center of Excellence in Space and Data Information Sciences (CESDIS), located at the NASA's Goddard Space Flight Center in Maryland, implemented the concept of Beowulf cluster [4,10]. Although most dedicated parallel machines for image information extraction and mining employed by NASA and other institutions during the last decade have been chiefly homogeneous in nature, a current trend in the design of systems for analysis and interpretation of the massive volumes of data provided in various scientific and engineering applications is to utilize highly heterogeneous, distributed platforms which can benefit from local (user) computing resources. In particular, computing on heterogeneous networks of workstations (NOWs) has been proven to be an economical alternative for a wide range of applications [15,19]. One of the most important features of these networks is that they enable the use of existing resources. Furthermore, such NOWs provide incremental scalability of hardware components. In other words, well-tuned parallel programs can be easily scaled to large configurations because additional workstations can always be added to a heterogeneous NOW. Also, these systems provide a high-degree of performance isolation, i.e., they allow analyzing parallel performance on a node-by-node basis. At the same time, NOWs can offer much greater communication speed at lower price using switch-based networks such as ATMs, and are able to provide distributed service and support, especially for large file systems. Finally, the technological evolution currently allows heterogeneous NOWs to support a variety of different workloads, including parallel, sequential and interactive jobs, as well as scalable computation-intensive applications.

It should be noted that, despite the growing interest in hyperspectral imaging research, only a few efforts devoted to the design of parallel techniques for information extraction from this type of data currently exist in the open literature [1,9]. This is 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 image information mining from hyperspectral imagery on distributed, highly heterogeneous computing platforms. The remainder of the paper is structured as follows. Section 2 describes standard approaches for hyperspectral image analysis in the literature.

Section 3 describes several new parallel information extraction techniques designed to be run in heterogeneous NOWs. Section 4 assesses the performance of the parallel algorithms by comparing their accuracy and parallel properties using several heterogeneous and homogeneous NOWs at University of Maryland. Our criterion for measuring optimality is to compare the efficiency achieved by a heterogeneous algorithm on a (fully or partially) heterogeneous NOW with the efficiency achieved by its equivalent homogeneous version on a fully homogeneous NOW with the same aggregate performance as the heterogeneous one. For comparative purposes, parallel performance results on Thunderhead, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center, are also provided. Section 5 concludes with some remarks and hints at plausible future research.

2. Standard techniques for information extraction from hyperspectral data

This section describes several information extraction techniques that will be considered in our study. Prior to the detailed description of such techniques, we first provide an overview of data mining related work in the context of remote sensing applications. Then, we describe three highly representative classes of algorithms which can be used for the purpose of information extraction from remotely sensed, multidimensional image data sets [18], including clustering, classification and spectral unmixing techniques. All of the algorithms addressed below are based on the analysis of the information provided by spectral signatures *as a whole*, which creates the need for data partitioning strategies able to preserve such spectral information when processing the multidimensional image in parallel. Our focus in this work is on unsupervised techniques, able to perform with little or no human supervision at all.

2.1. Data mining in remote sensing applications

Optimization and effective utilization of the full information content provided by Earth Observation sensors requires systematic archiving, data management, and means of image information mining [8,29]. The first step in the processing chain is image collection, which is generally followed by a procedure in which raw image data, generated during a flight season, are georeferenced and primary information like digital surface models is extracted automatically and made available for later use in various independent applications, some of which may be subject to near real-time constraints [28]. Each newly collected image often goes into an interim database to be used for data processing. After this step, value-added images are archived and later retrieval generally makes use of search criteria generated during the value-adding processing step, also known as *metadata* of the imagery [27]. In this work, we concentrate on the data processing aspect of image information mining. As a result, several information extraction techniques, specifically developed for hyperspectral imagery, will be presented and discussed in the following subsections. It is obvious that the large quantities of raw and processed data, collected on a daily basis in remote sensing missions, require highly automated, reliable and fast processing procedures. For that purpose, we develop several parallel implementations of the discussed information mining techniques and specifically adapt them to be efficiently run on heterogeneous parallel platforms.

2.2. Clustering

Clustering is an unsupervised data mining technique which relies on nearest neighbor information. One of the most widely used clustering algorithms for image analysis is ISODATA [26], which uses a spectral-based distance as a similarity measure to cluster data elements into different classes. In high-dimensional spaces, however, the data space becomes sparsely populated and the distances between individual data points become very similar. In order to address this issue, the principal component transform (PCT) has been adopted as a standard spectral transformation which is used to summarize and decorrelate the images prior to the clustering stage by reducing redundancy and packing the residual information into a small set of images, termed *principal components* [26]. Both the ISODATA and PCT algorithms rely on mathematical transformation and heavy linear algebra operations. Although parallel versions of such algorithms have been proposed in the literature in some form (e.g., D-ISODATA [9], S-PCT [1]), these techniques were designed to be run in homogeneous parallel computers and their performance in heterogeneous platforms is significantly degraded, as will be demonstrated by experiments. In this paper, we develop two new parallel algorithms called Hetero-ISODATA and Hetero-PCT, which are specifically tuned for execution on heterogeneous NOWs.

2.3. Classification

A number of classification algorithms for hyperspectral image data have been developed in recent years. One of the most successful ones has been the automatic target detection and classification algorithm (ATDCA) [5], a computation-intensive approach which finds a set of unique pixel vectors using of orthogonal subspace projections in the spectral domain, and then uses this set to characterize other pixels in the data. An important research thrust yet to be fully adopted in the design of unsupervised classifiers is the incorporation of spatial information. Available techniques such as ATDCA treat the hyperspectral data not as an image, but as an unordered listing of spectral measurements where the spatial coordinates can be shuffled arbitrarily without affecting the final analysis. However, one of the distinguishing properties of hyperspectral data is the multivariate information coupled with a two-dimensional (pictorial) representation amenable to image interpretation. Subsequently, there is a need to incorporate the spatial arrangement of the data. In this paper, we develop a new unsupervised method for hyperspectral data classification on heterogeneous NOWs. This method, called heterogeneous morphological classification (Hetero-MORPH), uses

mathematical morphology concepts to categorize pixel vectors in terms of both their spectral purity and spatial relevance in the scene [23].

2.4. Spectral mixture analysis

The underlying assumption governing clustering and classification techniques above is that each pixel vector measures the response of a single underlying material. However, 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 "mixed pixel," i.e., a composite of the individual pure spectra [17]. To deal with this problem, spectral mixture analysis techniques first identify a collection of spectrally pure constituent spectra, called *endmembers*, and then express the measured spectrum of each mixed pixel as a combination of *endmembers* weighted by fractions or *abundances* that indicate the proportion of each endmember present in the pixel [24]. One of the most successful algorithms for endmember extraction has been the pixel purity index (PPI) [3]. The algorithm generates a large number of random unit vectors, called "skewers," and projects every pixel vector in the data set onto each skewer. The data points that correspond to extrema after many random projections are identified as endmembers. Another standard technique is the N-FINDR algorithm [31], which finds the set of pixels which define the simplex with the maximum volume potentially inscribed within the data set. Both the identification of image endmembers and the subsequent spectral unmixing process are computationally demanding problems. However, very few research efforts devoted to the design of parallel algorithms for endmember extraction and spectral mixture analysis exist in the open literature. In this work, we develop three highly innovative algorithms calleded Hetero-PPI, Hetero-FINDR and Hetero-LSU (for linear spectral unmixing [5]), which have been specifically tuned for execution on heterogeneous NOWs.

3. Parallel implementations

This section describes several parallel algorithms that will be compared in this study. Before introducing their algorithmic descriptions, we first formulate a general optimization problem in the context of fully heterogeneous NOWs, composed of different-speed processors that communicate through links at different capacities. This type of platform can be modeled as a complete graph $G = (P, E)$, where each node models a computing resource p_i weighted by its relative cycle-time w_i. Each edge in the graph models a communication link weighted by its relative capacity, where c_{ij} denotes the maximum capacity of the slowest link in the path of physical communication links from p_i to p_j. We also assume that the system has symmetric costs, i.e., $c_{ij} = c_{ji}$. Processor p_i will accomplish a share of $\alpha_i \cdot W$ of the total workload W to be performed by a certain algorithm, with $\alpha_i \geqslant 0$ for $1 \leqslant i \leqslant P$ and $\sum_{i=1}^{P} \alpha_i = 1$. Since most processing algorithms in hyperspectral imaging applications involve repeated vector product operations, we can measure the workload involved by each considered algorithm in terms of elementary multiplication/accumulation (MAC) operations. With the above assumptions in mind, an abstract view of our problem can be simply stated in the form of a client–server architecture, in which the server is responsible for the efficient distribution of work among the P nodes, and the clients operate with the spectral signatures contained in a local partition. The partitions are updated locally and, depending on the considered algorithm, the resulting calculations may also be exchanged between the clients, or between the server and the clients. Before describing our parallel algorithms, we provide an overview of the adopted data partitioning strategy.

3.1. Hyperspectral data partitioning

Two types of partitioning can be exploited in the considered application: spectral-domain partitioning and spatial-domain partitioning. Spectral-domain partitioning subdivides the data volume into small cells or sub-volumes made up of contiguous spectral bands, and assigns one or more sub-volumes to each processor. It should be noted that most information extraction techniques for hyperspectral imaging focus on analyzing the data based on the properties of spectral signatures, i.e., they utilize with the information provided by each pixel vector *as a whole*. With a spectral-domain partitioning model, each pixel vector may be split amongst several processors and the calculations made for each spectral signature would need to originate from several processors. Although such spectral-domain partitioning strategy has been used in very low-dimensional remote sensing applications (e.g., those based on color images), it was soon discarded in our framework due to several reasons [25]. First, spatial-domain partitioning is a more natural approach for neighbor-based image processing, as many image-processing operations require the same function to be applied to a small set of elements around each pixel vector in the input data volume. A second major reason has to do with the overhead introduced by inter-processor communications. Since most hyperspectral algorithms extract information based on spectral signatures *as a whole*, partitioning the data in the spectral domain would linearly increase communications with the increase in the number of processing elements, thus complicating the design of parallel algorithms (in particular, in heterogeneous environments). A final major issue is code reusability; to reduce code redundancy and enhance portability, it is desirable to reuse much of the code for the sequential algorithm in the design of its correspondent parallel version. In order to balance the load of the processors in the heterogeneous environment, each processor should execute an amount of work that is proportional to its speed. Therefore, two major goals of our partitioning algorithm are: (1) to obtain an appropriate set of workload fractions $\{\alpha_i\}_{i=1}^{P}$ that best fit the heterogeneous environment, and (2) to translate the chosen set of values into a suitable spatial-domain decomposition of the input data. In order to accomplish the first goal, we use a simple workload estimation algorithm (WEA).

Algorithm 1. Workload estimation algorithm (WEA).
Input: Workload W.
Output: Set of values $\{\alpha_i\}_{i=1}^{P}$.

1. Obtain necessary system information, including the number of available processors P, each processor's identification number $\{p_i\}_{i=1}^{P}$, and processor cycle-times $\{w_i\}_{i=1}^{P}$.
2. Set $\alpha_i = \left\lfloor \frac{(P/w_i)}{\sum_{i=1}^{P}(1/w_i)} \right\rfloor$ for all $i \in \{1, \ldots, P\}$.
3. For $m = \sum_{i=1}^{P} \alpha_i$ to W, find $k \in \{1, \ldots, P\}$ such that $w_k \cdot (\alpha_k + 1) = \min\{w_i \cdot (\alpha_i + 1)\}_{i=1}^{P}$, and then set $\alpha_k = \alpha_k + 1$.

The WEA algorithm above assumes that the workload of each processor p_i must be inversely proportional to its cycle-time w_i. Specifically, step 2 first approximates the $\{\alpha_i\}_{i=1}^{P}$ so that $\alpha_i \cdot w_i \approx const$ and $\sum_{i=1}^{p} \alpha_i \leqslant W$. Then, step 3 iteratively increments some α_i until $\sum_{i=1}^{p} \alpha_i = W$. It should be noted that a homogeneous version of the WEA algorithm can be obtained by simply replacing step 3 with $\alpha_i = P/w_i$ for all $i \in \{1, \ldots, P\}$, where w_i is a constant cycle-time for all processors in a homogeneous system. Once the set $\{\alpha_i\}_{i=1}^{P}$ has been obtained, a further objective is to produce P spatial-domain partitions of the input hyperspectral data set. In this work, we adopt two different strategies to accomplish the latter goal. In the so-called *without overlap* strategy, the data are partitioned in the spatial domain so that the size of partitions is proportional to the values of $\{\alpha_i\}_{i=1}^{P}$. Here, a communication overhead may be introduced for algorithms which are based on the use of the spatial context around each pixel vector (e.g., mathematical morphology-based methods). In such cases, a workstation may need an extra communication overhead for the calculations involving pixel vectors which are close to the boundary of a partition. To avoid such macro-communication overhead, we use an *overlap mapping* strategy based on replicating the pixel vectors at the border of a partition so that each workstation can perform all the calculations independently, with no need to know which other workstations have pixels close to the boundary of the partition. It is important to note that the amount of redundant information introduced by the overlapping scatter depends on the size of the structuring element (only 3×3 pixels in our Hetero-MORPH algorithm). It is also important to note that such overlap borders are not required for clustering and spectral mixture analysis algorithms, which work on a pixel-by-pixel basis and do not incorporate the spatial information. In order to perform the final data partitioning in both cases, we adopt a simple hybrid methodology which consists of two main steps:

1. Partition the hyperspectral data set so that the number of rows in each partition is proportional to the values of $\{\alpha_i\}_{i=1}^{P}$ and assuming that no upper bound exists on the number of pixel vectors that can be stored by the processor (at the same time, replicate necessary information for the overlap partitioning strategy when required). If the number of pixel vectors in each partition assigned to each processor is less than the upper bound on the number of elements that can be stored by the processor, we have an optimal distribution.
2. For each processor p_i, check if the number of pixel vectors assigned to it is greater than the upper bound on the number of elements that it can store. For all the processors whose upper bounds are exceeded, we assign them a number of pixel vectors equal to their upper bounds. Now, we solve the partitioning problem of a set with remaining pixel vectors (including those resulting from data replication in the overlap strategy) over the remaining processors. We recursively apply this procedure until all the elements have been assigned.

3.2. Parallel hyperspectral algorithms

In this section, we provide algorithmic descriptions of parallel heterogeneous algorithms for hyperspectral information extraction based on the concepts of clustering, classification and spectral mixture analysis.

3.2.1. Clustering

The algorithm selected in this study as a representative of spectral clustering techniques for hyperspectral imaging is the ISODATA [26]. We have implemented the algorithm using the spectral angle distance (SAD). If we denote by $\mathbf{F}(x, y)$ and $\mathbf{F}(x', y')$ the pixel vectors at two different spatial locations of an input hyperspectral image, denoted by $\mathbf{F}$, then the SAD between these two pixel vectors is given by the following expression [5]:

$$\mathrm{SAD}[\mathbf{F}(x, y), \mathbf{F}(x', y')] = \cos^{-1}[\mathbf{F}(x, y) \cdot \mathbf{F}(x', y')/\|F(x, y)\| \cdot \|F(x', y')\|].$$

Below, we provide a master–slave parallel implementation of ISODATA which is based on the above-mentioned SAD distance measure.

Algorithm 2. Heterogeneous ISODATA (Hetero-ISODATA).
Input: Hyperspectral image $\mathbf{F}$ with N dimensions, number of clusters c, convergence threshold t.
Output: 2-dimensional matrix which contains a cluster label for each pixel $\mathbf{F}(x, y)$.

1. The server divides the original image cube $\mathbf{F}$ into P heterogeneous partitions using the WEA algorithm.
2. Each partition is sent to a worker along with a set of c randomly selected pixel vectors (initial centroids).
3. Let us denote by n_{ij} the number of pixels belonging to the jth cluster of the ith worker, and by $\mathbf{F}_k^j(x, y)$ the kth pixel of the jth cluster. Each worker sends to the master the number of pixels belonging to each cluster and the sum of all pixel vectors of the pixels belonging to each cluster, which involves a workload (expressed in MACs) of $n_{ij} \times N$ as can be derived from the expression below:

$$Y_{ij} = \left[\sum_{k=1}^{n_{ij}} \mathbf{F}_k^1(x, y), \ldots, \sum_{k=1}^{n_{ij}} \mathbf{F}_k^c(x, y) \right].$$

4. The master collects all the information provided by the workers and combines it to obtain a new centroid for each cluster j as $\mathbf{m}_j = (\sum_{i=1}^{p_j} \mathbf{Y}_{ij} / \sum_{i=1}^{p_j} n_{ij})$, where p_j is the number of pixels in the cluster. This operation involves a workload of $p_j \times N$ for each cluster.
5. The master compares the current centroids and the new ones. If the SAD between them is below 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.
6. Following convergence, each worker i computes the summation of the SAD distances of all pixels within a cluster j from its centroid $\mathbf{m}_j$, which involves a workload of $n_{ij} \times N$ to calculate the centroid plus an additional workload of $3 \times n_{ij} \times N$ to compute the SAD distances (each SAD computation involves a dot product of the pixel considered with the centroid and two additional dot products to calculate the vector norms of the pixel and the centroid). To reduce communication, this information is only exchanged when convergence has occurred (instead of doing so every time new centroids are calculated).
7. The master now decides about splitting or merging of the resulting clusters, based on parameters 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 [26]. The procedure is repeated (following the same steps for the previous convergence) until no cluster is further eligible.
8. When the stopping criterion is satisfied, each worker sends the label (cluster number) associated with each local pixel to the master, which combines the individual results and forms the final label image.

The second clustering technique proposed in this work first preprocesses the data via a PCT-based transformation to increase the data separability in spectral space [26]. Then, it makes use of a SAD-based distance criterion to provide a final classification, as described below.

Algorithm 3. Heterogeneous PCT-based clustering (Hetero-PCT).

Input: Hyperspectral image $\mathbf{F}$ with N dimensions, number of clusters c.

Output: 2-dimensional matrix which contains a cluster label for each pixel $\mathbf{F}(x, y)$.

1. The server divides the original image cube $\mathbf{F}$ into P heterogeneous partitions using the WEA algorithm.
2. Each partition is sent to a worker, which forms a unique spectral set by calculating the SAD distance for all vector pairs. The total workload involved by this step is $3 \times M^2 \times N$, where M is the number of pixels in the hyperspectral image and N is the number of spectral bands (each SAD-based computation involves three vector dot products).
3. The P 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 made up of c pixel vectors.
4. An N-dimensional mean vector $\mathbf{m}$ is calculated 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 processing their respective parts. The total workload involved by this step is $M \times N$.
5. All the pixel vectors in the unique set are divided into P parts and sent to the workers. Each worker then computes the covariance component and forms a covariance sum.
6. The covariance matrix is calculated sequentially as the average of all the matrices calculated in step 5. It should be noted that steps 5 and 6 involve M matrix multiplications and a total workload of $M \times N^2$.
7. A transformation matrix $\mathbf{T}$ is obtained 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 done sequentially at the master (the eigenvectors calculation has a workload of N^3 [11]).
8. Each pixel vector in the original hyperspectral image is transformed independently using $\mathbf{T} \cdot [\mathbf{F}(x, y) - \mathbf{m}]$. This step is done in parallel, where all workers transform their respective portions of data concurrently. The linear transformation process is performed over all the pixels in $\mathbf{F}$ and each computation on a pixel involves a matrix multiplication. Thus, the workload of this step is $M^2 \times N^2$.
9. Finally, a parallel post-processing step is applied to perform clustering in the PCT-transformed space. First, P partitions of a reduced, c-dimensional data cube given by the first PCT components are sent to the workers, along with the spatial locations of the c unique pixel vectors resulting from step 2. Each worker then labels each pixel in its corresponding partition using a cluster label given by the most spectrally similar unique pixel vector in the PCT-reduced space, and sends back the result to the master, which puts together the final label image.

3.2.2. Classification

This section first describes a parallel heterogeneous version of the ATDCA algorithm which first identifies a unique set of target pixel vectors, and then performs hyperspectral data classification based on this unique set.

Algorithm 4. Heterogeneous ATDCA algorithm (Hetero-ATDCA).

Inputs: Hyperspectral image $\mathbf{F}$ with N dimensions, number of classes c.

Output: 2-dimensional matrix which contains a classification label for each pixel $\mathbf{F}(x, y)$.

1. Using the WEA algorithm, the server divides the original image cube $\mathbf{F}$ into P heterogeneous partitions.
2. Each worker finds in parallel the brightest pixel at the local partition $\mathbf{t}_i^{(1)} = \arg\{\max_{(x,y)} F(x,y)^{\mathrm{T}} \cdot \mathbf{F}(x,y)\}$, where $i = 1,\ldots,P$ and the superscript "T" denotes the vector transpose operation. The workers send the spatial locations of the pixel identified as the brightest ones in each partition back to the master. The total workload involved in this operation is $M^2 \times N$, where M is the number of pixels in the hyperspectral image and N is the number of spectral bands.
3. The master finds the brightest pixel $\mathbf{t}^{(1)}$ by applying the arg max operator in step 2 to the pixels at that the spatial locations provided by the workers and selecting the one which results in the maximum score, and then sets $\mathbf{U} = \mathbf{t}^{(1)}$ to initialize the algorithm, where the superscript "(1)" indicates that $\mathbf{t}^{(1)}$ is the first pixel selected throughout the process. It is worth noting that the brightest pixel is not the only possible selection for initialization of ATDCA. However, according to previous experiments in the literature [5], the brightest pixel is always extracted later on by the algorithm if it is not used as the initial pixel vector.
4. The master broadcasts matrix $\mathbf{U}$ to all the workers, each of which works in parallel to find the pixel in the local partition with the maximum orthogonal projection relative to the pixel vectors in $\mathbf{U}$, using an orthogonal space projector $P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}(\mathbf{U}^{\mathrm{T}}\mathbf{U})^{-1}\mathbf{U}^{\mathrm{T}}$, where $\mathbf{I}$ is the identity matrix. The projector is applied to all pixel vectors in each local partition to obtain $\mathbf{t}_i^{(2)} = \arg\{\max_{(x,y)}[(P_{\mathbf{U}}^{\perp}\mathbf{F}(x,y))^{\mathrm{T}}(P_{\mathbf{U}}^{\perp}\mathbf{F}(x,y))]\}$. Each worker then sends the spatial location of the resulting local pixels to the master node.
5. The master finds a second pixel $\mathbf{t}^{(2)}$ by applying the orthogonal space projector $P_{\mathbf{U}}^{\perp}$ to the pixel vectors at the spatial locations provided by the workers, and selecting the one which results in the maximum score. The master then sets $\mathbf{U} = \{\mathbf{t}^{(1)}\mathbf{t}^{(2)}\}$.
6. Repeat from step 4 until a set of c unique pixel vectors, $\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, \ldots, \mathbf{t}^{(c)}$, have been extracted. It should be noted that the selection of each new pixel involves a vector-matrix multiplication with workload of $N^2 \times c$ and four matrix multiplications. As the value of c grows, the complexity of the algorithm increases in exponential fashion.
7. The resulting set of c unique pixel vectors is broadcast to all the workers, which obtain a classification label for each pixel $\mathbf{F}(x,y)$ in the local partition by assigning it to a class given by the unique pixel vector which is most highly similar to the pixel in terms of the SAD distance. This involves an additional set of $3 \times N \times c$ MACs.
8. Each worker then sends the labels associated with each local pixel to the master, which gathers all the individual results and forms a final, 2-dimensional classification matrix.

Since the Hetero-ATDCA algorithm is composed of several sequential steps, which are executed in parallel, an appropriate workload distribution is essential. From the algorithm description above, it is clear that the algorithm relies on the spectral information alone, without taking into account the spatial arrangement of pixels.

A second morphological classification algorithm is introduced in this section to naturally integrate the spatial and spectral information present in the input data. The proposed algorithm is based on the definition of a cumulative distance between each pixel vector, $\mathbf{F}(x,y)$, and all pixel vectors in the spatial neighborhood of $\mathbf{F}(x,y)$, given by a spatial kernel or structuring element B as follows: $D_B[\mathbf{F}(x,y)] = \sum_s \sum_t \mathrm{SAD}[\mathbf{F}(x,y), \mathbf{F}(s,t)]$, where $(s,t) \in Z^2(B)$. Based on the distance metric above, two basic morphological operations (erosion and dilation) can be defined and used, respectively, to extract the most highly pure and the most highly mixed pixel in the B-given spatial neighborhood as follows: $(\mathbf{F}\Theta B)(x,y) = \arg\min_{(s,t)\in Z^2(B)}\{D_B[\mathbf{F}(x+s,y+t)]\}$ and $(\mathbf{F} \oplus B)(x,y) = \arg\max_{(s,t)\in Z^2(B)}\{D_B[\mathbf{F}(x+s,y+t)]\}$. Using the two extended operations above, we provide below a new parallel heterogeneous algorithm for unsupervised classification of hyperspectral imagery. This algorithm adopts our proposed *overlap mapping* strategy for data partitioning to reduce inter-processor communication.

Algorithm 5. Heterogeneous morphological classification (Hetero-MORPH).

Inputs: Hyperspectral image $\mathbf{F}$ with N dimensions, Number of iterations $I_{\max}$, 3×3-pixel structuring element B, Number of classes c.

Output: 2-dimensional matrix which contains a classification label for each pixel $\mathbf{F}(x,y)$.

1. Using the WEA algorithm, the server divides the original image cube $\mathbf{F}$ into P heterogeneous partitions with overlap borders to avoid accesses outside the local image domain at each workstation.
2. Using parameters $I_{\max}$, B and c, each worker executes the following steps:
   2.1. Set $j = 1$ and initialize a morphological eccentricity index score $MEI(x,y) = 0$ for each local pixel.
   2.2. Move B through all the pixels of the local partition, and calculate the "maximum" and "minimum" pixel vectors using erosion and dilation operations, respectively. The MEI score at each pixel is then updated by calculating the SAD between the pixel vectors selected as "maximum" and "minimum."
   2.3. Set $j = j + 1$. If $j = I_{\max}$ then go to step 2.4. Otherwise, replace the local partition by its dilation using B, i.e., $\mathbf{F} = \mathbf{F} \oplus B$, and go to step 2.2.
   2.4. Select the set of c pixel vectors in the local partition with the highest associated MEI scores. The workload involved by the first two steps of the algorithm can be approximated by $M \times B \times I_{\max} \times N$, where M is the number of pixel vectors in the input image, B is the size in pixels of the structuring element (set to a

constant $B = 3 \times 3$ in this work), $I_{\max}$ is the number of iterations, and N is the number of spectral bands.

3. The master gathers all the individual pixel vectors provided by the workers and forms a unique spectral set of $p \leqslant c$ pixel vectors by calculating the SAD for all vector pairs in parallel. The total workload involved by this step is $3 \times M^2 \times N$ (each SAD-based computation involves three vector dot products).
4. The set of p unique pixel vectors are broadcast to all the workers, which obtain a classification label for each pixel $\mathbf{F}(x, y)$ in the local partition by assigning it to a class given by the unique pixel vector which is most highly similar to the pixel in terms of the SAD distance. This step involves an additional set of $3 \times M \times N$ MACs across all the workers.
5. Each worker then sends the label associated with each local pixel to the master, which gathers all individual results and forms the final 2-dimensional classification matrix.

3.2.3. Spectral mixture analysis

In this section, we first provide parallel heterogeneous versions of two standard algorithms for endmember extraction. Our heterogeneous version of the PPI algorithm [3] is based on the generation of a large number of "skewer" vectors which are then projected onto input pixel vectors until extreme pixels are identified.

Algorithm 6. Heterogeneous PPI (Hetero-PPI).

Input: Hyperspectral image $\mathbf{F}$ with N dimensions, Number of endmembers E, Number of skewers k, Threshold t.

Output: Set of E endmembers $\{\mathbf{e}_e^{(0)}\}_{e=1}^{E}$.

1. Divide the original image cube $\mathbf{F}$ into P heterogeneous partitions using the WEA algorithm.
2. Generate a set of k randomly generated unit, N-dimensional vectors called "skewers," denoted by $\{\mathbf{skewer}_j\}_{j=1}^{k}$, and broadcast the entire set to all the workers.
3. For each $\mathbf{skewer}_j$, project all the pixel vectors at each local partition i, with $i = 1, \ldots, P$, onto $\mathbf{skewer}_j$ using the vector dot product $|\mathbf{skewer}_j \cdot \mathbf{F}(x, y)|$ to find sample vectors at extreme positions, and form an extrema set for $\mathbf{skewer}_j$, denoted by $S^{(i)}(\mathbf{skewer}_j)$. Define an indicator function of a set S as

$$I_S(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in S, \\ 0 & \text{if } \mathbf{x} \notin S, \end{cases}$$

and calculate $N_{PPI}^{(i)}[\mathbf{F}(x, y)] = \sum_j I_{S^{(i)}(\mathbf{skewer}_j)}[\mathbf{F}(x, y)]$ for each $\mathbf{F}(x, y)$ in the local partition.

4. Select those pixels with $N_{PPI}^{(i)}[\mathbf{F}(x, y)] > t$ and send their spatial locations to the master, which collects the partial results and forms a set of spectrally pure pixels (endmembers) $\{\mathbf{e}_e^{(0)}\}_{e=1}^{E}$ using the SAD distance. The total workload involved in the PPI can be approximated by $M \times k \times E$, where M is the total number of pixel vectors in the input scene, k is the number of skewers and E is the number of endmembers to be extracted by the algorithm.

In order to implement the N-FINDR algorithm, we first need to reduce the dimensionality of the input data from N to E [31]. For that purpose, we make use of PCT transform as indicated by the following algorithm.

Algorithm 7. Heterogeneous N-FINDR (Hetero-FINDR).

Inputs: Hyperspectral image $\mathbf{F}$ with N dimensions, Number of endmembers E.

Output: Set of E endmembers $\left\{\mathbf{e}_e^{(0)}\right\}_{e=1}^{E}$.

1. Execute steps 1–8 of the Hetero-PCT algorithm on $\mathbf{F}$ using $c = E$ to obtain an E-dimensional data cube $\mathbf{G}$.
2. The master selects a random set of E initial pixels $\left\{\mathbf{e}_e^{(0)}\right\}_{e=1}^{E}$ from the reduced data set, and then finds $V\left(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_E^{(0)}\right)$, i.e., the volume of the simplex defined by $\left\{\mathbf{e}_e^{(0)}\right\}_{e=1}^{E}$ as follows:

$$V\left(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_E^{(0)}\right) = \frac{\left|\det\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \mathbf{e}_1^{(0)} & \mathbf{e}_2^{(0)} & \ldots & \mathbf{e}_E^{(0)} \end{bmatrix}\right|}{(E-1)!}.$$

3. Calculate the volume of E simplexes, $V\left(\mathbf{G}(x, y), \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_E^{(0)}\right), \ldots, V\left(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{G}(x, y)\right)$ in parallel, each of which is formed by replacing one endmember $\mathbf{e}_e^{(0)}$ with each sample vector $\mathbf{G}(x, y)$. Each worker performs replacements using pixels in its local partition, obtained using the WEA algorithm.
4. If none of these E recalculated volumes is greater than $V\left(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_E^{(0)}\right)$, then no endmember in $\left\{\mathbf{e}_e^{(0)}\right\}_{e=1}^{E}$ is replaced. Otherwise, the master replaces the endmember which is absent in the largest volume among the E simplexes with the vector $\mathbf{G}(x, y)$. Let such endmember be denoted by $\mathbf{e}_l^{(0)}$. A new set of endmembers is produced by letting $\mathbf{e}_l^{(1)} = \mathbf{g}(x, y)$ and $\mathbf{e}_e^{(1)} = \mathbf{e}_e^{(0)}$ for $e \neq l$. Repeat from step 3 until no replacements take place. As described in [6], the total workload involved by the N-FINDR algorithm can be approximated by taking into account that the algorithm has to compute the determinant of a taking into account that the algorithm has to compute the determinant of an $E \times E$ matrix $E \times N$ times. The naive method of implementing an algorithm to compute the determinant is to use Laplace's formula for development by minors [32]. However, this approach is inefficient as it is of order $E!$ for an $E \times E$ matrix. In this work, an improvement to E^3 was achieved by using the LU decomposition for the computation of determinants [12].

Once a set of spectral endmembers has been identified, an inversion model is required to estimate the fractional abundances of the endmembers provided by methods such as Hetero-PPI or Hetero-FINDR in each of the pixels in the scene. For that purpose, we use a commonly adopted technique in the hyperspectral analysis literature called linear spectral unmixing (LSU), which can be easily implemented in parallel as follows.

Algorithm 8. Heterogeneous LSU (Hetero-LSU).
Inputs: Hyperspectral image $\mathbf{F}$ with N dimensions, set of E endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$.
Output: Set of fractional abundances $\{a_e(x, y)\}_{e=1}^{E}$ for each pixel $\mathbf{F}(x, y)$.

1. Divide the original image cube $\mathbf{F}$ into P heterogeneous partitions using the WEA algorithm.
2. Broadcast the set $\{\mathbf{e}_e\}_{e=1}^{E}$ to all the workers.
3. For each pixel $\mathbf{F}(x, y)$ in the local partition, each worker obtains a set of abundance fractions specified by $a_1(x, y), a_2(x, y), \ldots, a_E(x, y)$, so that $\mathbf{F}(x, y) = \mathbf{e}_1 \cdot a_1(x, y) + \mathbf{e}_2 \cdot a_2(x, y) + \cdots + \mathbf{e}_E \cdot a_E(x, y)$, taking into account the sum-to-one and non-negativity constraints: $\sum_{e=1}^{E} a_e = 1$ and $a_e \geqslant 0$ for $1 \leqslant e \leqslant E$. This is done by multiplying each pixel $\mathbf{F}(x, y)$ by $(\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}^{\mathrm{T}}$ [5], where $\mathbf{M} = \{\mathbf{e}_e\}_{e=1}^{E}$. This step involves a vector-matrix multiplication with workload of $N^2 \times E$, and three matrix multiplications, each with E^3.
4. The master collects all the individual sets of fractional abundances $\{a_e^{(i)}(x, y)\}_{e=1}^{E}$ calculated by each of the workers for the pixels at every individual partition i, with $i = 1, \ldots, P$, and forms a final set of fractional abundances designated by $a_e(x, y)\}_{e=1}^{E} = \bigcup_{i=1}^{P}\{a_e^{(i)}(x, y)\}_{e=1}^{E}$ for each píxel $\mathbf{F}(x, y)$ in the input scene.

As a final note, we emphasize that the proposed spectral mixture analysis framework consists of a sequence of three steps, i.e., dimensionality reduction (optional), endmember extraction, and spectral unmixing, each of which is implemented in parallel.

To conclude this section, we emphasize that all the parallel algorithms described in this section have been implemented in the C++ programming language using calls to message passing interface (MPI). Specifically, 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. It should be noted that the parallel codes generally described above could take the form of either MPMD (*multiple programs*, *multiple data*) or SPMD (*single program*, *multiple data*) when implemented using MPI. In the MPMD programming style, one distinct program is run per processor, while in SPMD the same program is executed on different processors, over distinct data sets [22]. In this work, we have tested the two above-mentioned strategies when implementing our parallel codes and compared the two approaches via experimental results, as will be shown in the following section.

4. Experimental results

This section provides an assessment of the effectiveness of the parallel algorithms described in Section 3. In order to quantitatively compare the performance of the different algorithms, we resort to a recently proposed framework for evaluation of heterogeneous parallel algorithms [20], which relies on the assumption that a heterogeneous algorithm cannot be executed on a heterogeneous NOW faster than its homogeneous version on the equivalent homogeneous NOW. In [20], a homogeneous computing environment was considered equivalent to the heterogeneous one if the following three principles were satisfied: (1) both environments have the same number of processors; (2) the speed of each processor in the homogeneous environment is equal to the average speed of processors in the heterogeneous environment; and (3) the aggregate communication characteristics of the homogeneous environment are the same as those of the heterogeneous environment. With the above three principles in mind, a heterogeneous algorithm may be considered optimal if its efficiency on a heterogeneous NOW is the same as that evidenced by its homogeneous version on the equivalent homogeneous NOW. This allows using the parallel performance achieved by the homogeneous version as a benchmark for assessing the parallel efficiency of the heterogeneous parallel algorithm. Before describing our detailed study on algorithm performance, we first introduce the parallel computing architectures used in this work and the hyperspectral data sets selected for evaluation purposes.

4.1. Parallel computing architectures

The parallel computing architectures used in this study comprise four NOWs distributed among several locations at University of Maryland, and a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center. The configuration of the four considered NOWs was carefully custom-designed in order to make sure that the three design principles above were satisfied. This offered an unprecedented opportunity to assess the performance of the proposed heterogeneous parallel algorithms.

- *Fully heterogeneous network*: It consists of 16 different SGI, Solaris and Linux workstations, and four communication segments. Table 1 shows the processor cycle-times of the 16 heterogeneous workstations, where processors $\{p_i\}_{i=1}^{4}$ are attached to communication segment s_1, processors $\{p_i\}_{i=5}^{8}$ communicate through s_2, processors $\{p_i\}_{i=9}^{10}$ are interconnected via s_3, and processors $\{p_i\}_{i=11}^{16}$ share the communication segment denoted by s_4. The communication links between the different segments $\{s_j\}_{j=1}^{4}$ only support serial communication. For illustrative purposes, Table 2 also shows the capacity of all point-to-point communications in the heterogeneous network, expressed as the time in milliseconds to transfer a one-megabit message between each processor pair (p_i, p_j) in the heterogeneous system. As noted, the communication network of the fully heterogeneous NOW consists of four relatively fast homogeneous communication

Table 1
Processor cycle-times (in seconds per megaflop) for the 16 heterogeneous workstations

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}
0.0058	0.0102	0.0206	0.0072	0.0102	0.0072	0.0072	0.0102	0.0072	0.0451	0.0131	0.0131	0.0131	0.0131	0.0131	0.0131

Table 2
Link capacity (in time in milliseconds to transfer a one-megabit message) for the heterogeneous network

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}
p_1	–	19.26	19.26	19.26	48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
p_2	19.26	–	19.26	19.26	48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
p_3	19.26	19.26	–	19.26	48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
p_4	19.26	19.26	19.26	–	48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
p_5	48.31	48.31	48.31	48.31	–	17.65	17.65	17.65	48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
p_6	48.31	48.31	48.31	48.31	17.65	–	17.65	17.65	48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
p_7	48.31	48.31	48.31	48.31	17.65	17.65	–	17.65	48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
p_8	48.31	48.31	48.31	48.31	17.65	17.65	17.65	–	48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
p_9	96.62	96.62	96.62	96.62	48.31	48.31	48.31	48.31	–	16.38	58.14	58.14	58.14	58.14	58.14	58.14
p_{10}	96.62	96.62	96.62	96.62	48.31	48.31	48.31	48.31	16.38	–	58.14	58.14	58.14	58.14	58.14	58.14
p_{11}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	–	–	14.05	14.05	14.05	14.05	14.05
p_{12}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	–	14.05	14.05	14.05	14.05
p_{13}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05	–	14.05	14.05	14.05
p_{14}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05	14.05	–	14.05	14.05
p_{15}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05	14.05	14.05	–	14.05
p_{16}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05	14.05	14.05	14.05	–

segments, interconnected by three slower communication links with capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$, $c^{(3,4)} = 58.14$ in ms, respectively. Although this is a simple architecture, it is also a quite typical and realistic one as well.

- *Fully homogeneous network*: Consists of 16 identical Linux workstations with processor cycle-time of $w = 0.0131$ s per megaflop, interconnected via a homogeneous communication network in which all communication links have the same capacity of $c = 26.64$ ms.
- *Partially heterogeneous network*: Formed by the set of 16 heterogeneous workstations in Table 1 but interconnected using the same homogeneous communication network with capacity $c = 26.64$ ms.
- *Partially homogeneous network*: Formed by 16 identical Linux workstations with processor cycle-time of $w = 0.0131$ s per megaflop, but interconnected using the heterogeneous communication network shown in Table 2.

In order to test the proposed algorithms on a larger-scale parallel platform, we have also experimented with Thunderhead, a 256-node Beowulf cluster located at NASA's Goddard Space Flight Center (GSFC) in Maryland. The Thunderhead system is composed of 256 Linux nodes at 2.4 GHz, interconnected via 2 GHz optical fibre Myrinet.

4.2. Hyperspectral image data sets

Two standard AVIRIS hyperspectral image data sets have been selected for experiments. The first one was gathered over the Indian Pines test site in Northwestern Indiana, a mixed agricultural/forest area, early in the growing season. It comprises 715 × 2149 pixels with 224 spectral bands, where each reflectance value is stored in integer format (the total size of the image is 672 MB). The data set represents a very challenging clustering/classification problem since the major crops of the area, mainly corn and soybeans, were very early in their growth cycle with only about 5% canopy cover. Discriminating among them is very difficult, a fact that has made this scene a universal and extensively used benchmark to validate hyperspectral imaging algorithms due to the availability of ground-truth information. Fig. 2(a) shows the spectral band at 587 nm of the original scene, and Fig. 2(b) shows a ground-truth map, in the form of a class assignment for each labeled pixel with 30 mutually exclusive ground-truth classes. These classes will be used to evaluate the accuracy of the proposed Hetero-ISODATA, Hetero-PCT, Hetero-ATDCA and Hetero-MORPH algorithms. Part of these data, including ground-truth, is available online (from http://dynamo.ecn.purdue.edu/~biehl/MultiSpec).

The second data set used in experiments is an AVIRIS scene with exactly the same dimensions as the previous one. It was collected over the Cuprite mining district in Nevada. The site is well understood mineralogically, and has several exposed minerals of interest. Fig. 3(a) shows a small portion of the image centered at the area with highest mineral variability, while Fig. 3(b) plots the spectra of five minerals of interest measured in the field by U.S. Geological Survey (USGS). These signatures will be used to substantiate end-member extraction accuracy of the Hetero-PPI and Hetero-FINDR methods. Fractional abundance maps derived by the USGS Tetracorder method will also be used to assess abundance estimation accuracy of the Hetero-LSU method. Since both the AVIRIS data and ground-truth are available online (from http://aviris.jpl.nasa.gov/html/aviris.freedata.html and

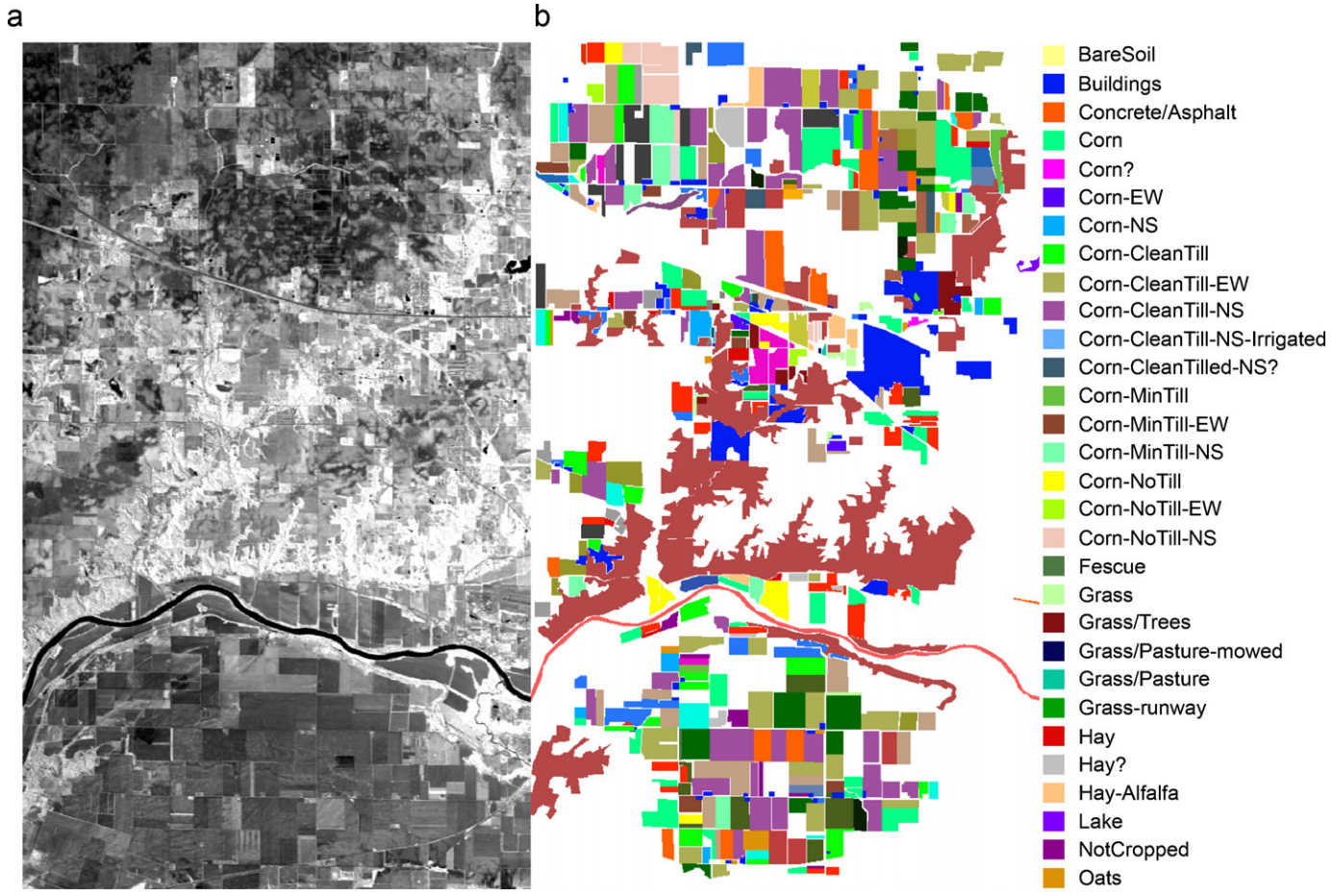

Fig. 2. (a) Spectral band at 587 nm wavelength of an AVIRIS scene comprising agricultural and forest features at Indian Pines test site, Indiana. (b) Ground-truth map with 30 mutually exclusive land-cover classes.

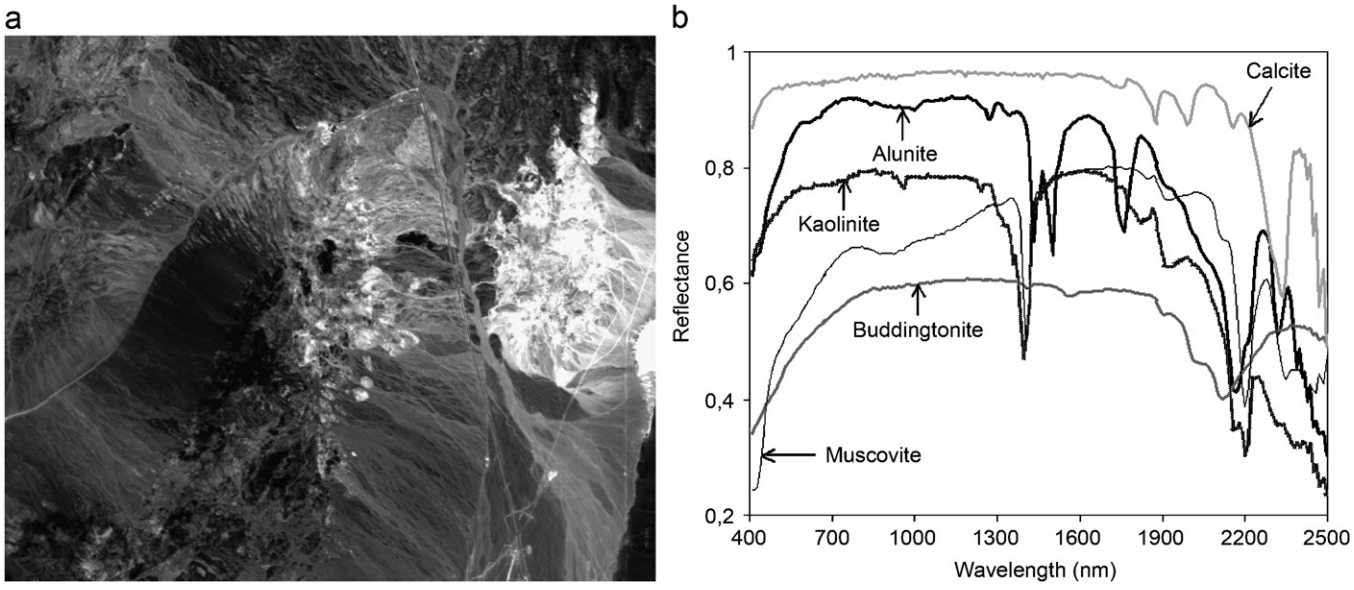

Fig. 3. (a) A small portion of the spectral band at 587 nm wavelength of an AVIRIS scene comprising mineral features at the Cuprite mining district, Nevada. (b) USGS ground-truth mineral spectra.

http://speclab.cr.usgs.gov, respectively) people interested in the proposed parallel algorithms can reproduce our results and conduct their own experiments.

4.3. Performance evaluation

Before describing the parallel properties of the considered algorithms, we first assess the quality of the information they extract. Table 3 reports the overall clustering/classification accuracy scores produced by the Hetero-ISODATA, Hetero-PCT, Hetero-ATDCA and Hetero-MORPH algorithms on the AVIRIS Indian Pines scene in Fig. 2(a). In the four cases, the number of clusters/classes was set to $c = 30$ after calculating the intrinsic dimensionality of the data using the virtual dimensionality concept in [5]. For the Hetero-ISODATA algorithm, a tolerance threshold $t = 0.05$ was used after experimental results in [9]. The Hetero-MORPH algorithm used a 3×3-pixel structuring element, and the maximum number of iterations was set to $I_{\max} = 7$ after previous results in [23]. As shown by Table 3, the Hetero-MORPH algorithm produced higher accuracy scores than those found by the other tested algorithms. The above results provide an objective confirmation of our introspection: that the incorporation of spatial information can enhance analysis results achieved using spectral information only.

On the other hand, Table 4 tabulates the SAD scores obtained after comparing the five USGS library spectra in Fig. 3(b) with the corresponding endmembers extracted by the Hetero-PPI and Hetero-FINDR parallel algorithms. The smaller the SAD values across the five minerals considered in Table 4, the better the results (a similarity score of 0.1 and below is generally considered as highly accurate in most hyperspectral applications). Table 4 also reports (in bold typeface) the root mean square error (RMSE) between the abundances estimated by using the Hetero-LSU algorithm (in combination with the endmembers provided by Hetero-PPI and Hetero-FINDR), estimated for each endmember using $\mathrm{RMSE}(\mathbf{e}_e) = ((1/X \cdot Y) \sum_{x=1}^{X} \sum_{y=1}^{Y} [a_e(x, y) - \hat{a}_e(x, y)]^2)^{1/2}$, where $\hat{a}_e(x, y)$ denotes the abundance estimated by Hetero-LSU for endmember $\mathbf{e}_e$ at $\mathbf{F}(x, y)$, $a_e(x, y)$ denotes the abundance measured by USGS for that endmember at the same pixel, and $X \cdot Y$ is the total number of pixels in the scene (in our example, $X = 715$ and $Y = 2149$). It should be noted that RMSE scores below 5% are acceptable in most applications. For the Hetero-PPI, parameter t was set to the mean of N_{PPI} scores after $k = 10^3$ skewer iterations (these parameter settings above are in agreement with those used before in the literature [24]).

Table 3
Clustering/classification accuracies (in percentage of correctly labeled pixels) for the Indian Pines scene

Hetero-ISODATA	Hetero-PCT	Hetero-ATDCA	Hetero-MORPH
71.45	83.58	84.79	91.24

To investigate the parallel properties of the tested algorithms, their performance was first tested by timing the programs using the four equivalent NOWs. Table 5 shows the measured execution times for the proposed parallel heterogeneous algorithms and their respective homogeneous versions (hereinafter, we refer to the homogeneous version of a certain heterogeneous algorithm by replacing the "Hetero-" in the name of the heterogeneous algorithm by "Homo-" to indicate that the algorithm is the equivalent, homogeneous version of the same heterogeneous one). The table reports the times obtained after adopting both MPMD and SPMD programming styles in the design of the parallel codes. In the MPMD approach, the final parallel C++ code took the form of a set of distinct main functions (one per processor), each containing direct calls to application-specific functions for hyperspectral image processing, interleaved with communications. By contrast, the latter approach was carried out by a single program or *kernel*, written in C++ and run in SPMD mode on all processors, thus ensuring a dynamic distribution and scheduling of processes and communications.

As expected, the execution times reported on Table 5 show that the heterogeneous algorithms were able to adapt much better to fully (or partially) heterogeneous environments than the homogeneous versions, which only performed satisfactorily on the fully homogeneous network. One can see that the heterogeneous algorithms were always several times faster than their homogeneous counterparts in the fully heterogeneous NOW, and also in both the partially homogeneous and the partially heterogeneous networks. On the other hand, the homogeneous algorithms only slightly outperformed their heterogeneous counterparts in the fully homogeneous NOW. Table 5 also reveals that the performance of the heterogeneous algorithms on the fully heterogeneous platform was almost the same as that evidenced by the equivalent homogeneous algorithms on the fully homogeneous NOW. This indicates that the proposed heterogeneous algorithms were always very close to the optimal heterogeneous modification of the basic homogeneous one. Finally, it is also worth noting that the use of an MPMD- or SMPD-oriented strategy in the development of the

Table 4
SAD-based similarity scores and RMSE-based abundance estimation errors (in percentage) obtained by the parallel endmember extraction algorithms (combined with Hetero-LSU for abundance estimation) for the Cuprite scene

Algorithm	Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite
Hetero-PPI	0.074	0.075	0.081	0.066	0.068
	3.03%	**2.12%**	**1.21%**	**0.43%**	**1.08%**
Hetero-FINDR	0.083	0.091	0.094	0.074	0.077
	4.12%	**3.05%**	**2.36%**	**1.94%**	**1.85%**

Table 5
Execution times (seconds) of the heterogeneous algorithms and their homogeneous versions on the four NOWs using both MPMD and SPMD parallel implementations

Parallel algorithm	Fully heterogeneous		Fully homogeneous		Partially heterogeneous		Partially homogeneous	
	MPMD	SPMD	MPMD	SPMD	MPMD	SPMD	MPMD	SPMD
Hetero-ISODATA	139	145	144	157	140	148	142	154
Homo-ISODATA	489	503	136	146	477	495	301	322
Hetero-PCT	132	138	136	149	133	141	135	145
Homo-PCT	562	574	129	138	547	561	330	342
Hetero-ATDCA	84	86	89	94	87	91	88	93
Homo-ATDCA	667	693	81	86	638	655	374	385
Hetero-MORPH	171	209	177	206	172	195	174	199
Homo-MORPH	2216	2395	168	188	2203	2346	925	986
Hetero-PPI	258	267	264	275	261	271	263	273
Homo-PPI	2719	2786	255	261	2698	2740	1217	1267
Hetero-FINDR	51	53	56	60	55	59	56	61
Homo-FINDR	506	525	50	58	497	519	253	264
Hetero-LSU	115	117	121	129	117	124	119	127
Homo-LSU	1595	1609	113	121	1563	1581	673	702

parallel codes provided similar results from the viewpoint of computational performance, with the SMPD approach resulting in slightly higher execution times in most cases, as reported by Table 5. Although the SMPD strategy provides several interesting features, such as the simplicity of the data distribution phase among the different computation processes, it has been experimentally tested (in the context of our hyperspectral imaging application) that the *kernel*-based approach adopted by SPMD generally requires more communications in exchanging data between the master and the slaves when the computation performed by the considered algorithm is not completely local (i.e., when the ratio of computations to communications is decreased). With the above issues in mind, in the following we only report results for MPMD-based implementations of our parallel algorithms.

For the sake of comparison, Figs. 4(a–d) plot the performance ratio between each heterogeneous algorithm and its respective homogeneous version (referred to as Homo/Hetero ratio in the table) on the four considered parallel platforms. The ratio was simply calculated as the execution time of the homogeneous algorithm divided by the execution time of the heterogeneous algorithm. One can see that, although Hetero-ISODATA and Hetero-PCT showed acceptable execution times in Table 3, their Homo/Hetero ratios in the fully heterogeneous NOW were clearly the lowest [see Fig. 4(a)]. This is due to the fact that the two algorithms involve operations that need to be completed sequentially at the master *before* distributing the results to the workers (e.g., split and merge in Hetero-ISODATA or covariance matrix calculations in Hetero-PCT). Some of the computations involved by the two algorithms are also highly irregular in nature, which results in the fact that both Hetero-PCT and Hetero-ISODATA were only approximately four times faster than their respective homogeneous versions in the fully heterogeneous NOW. Quite opposite, the other tested algorithms showed much better Homo/Hetero ratios in the same platform. For instance, the ratio measured for the Hetero-ATDCA algorithm was close to eight, which means that the heterogeneous algorithm was about eight times faster than its homogeneous version in the fully heterogeneous NOW, while the two parallel endmember extraction algorithms (Hetero-PPI and Hetero-FINDR)R) were about 10 times faster than their homogeneous versions in the same computing platform. These algorithms are dominated by regular computations, although they also still involve sequential processing steps at the master. The highest Homo/Hetero ratios in Fig. 4(a) were reported for both Hetero-LSU and Hetero-MORPH. While the former is a straightforward parallel technique with almost no data dependencies or sequential computations, the high ratio achieved by Hetero-MORPH is of great importance, given the spatial/spectral nature of the algorithm. It results from the carefully adopted algorithm design, which is based on the incorporation of intelligent data replication to enhance regularity in the computations. As noted, the small amount of redundant computations introduced by the *overlap mapping* strategy does not prevent Hetero-MORPH from achieving comparable results to those found by the best-reported method (Hetero-LSU).

While the Homo/Hetero ratios in the partially heterogeneous NOW were very similar to those reported above [see Fig. 4(b)], they were significantly decreased for the partially homogeneous NOW [see Fig. 4(c)]. This was because the homogeneous algorithms performed much better on the partially homogeneous network (made up of processors with the same cycle-times as opposed to those in the partially heterogeneous network). This fact reveals that processor heterogeneity has a more significant impact on algorithm performance than communication network heterogeneity, which is not surprising given our data partitioning-driven strategy for the design of parallel heterogeneous algorithms. Finally, Fig. 4(d) plots the speedup of the

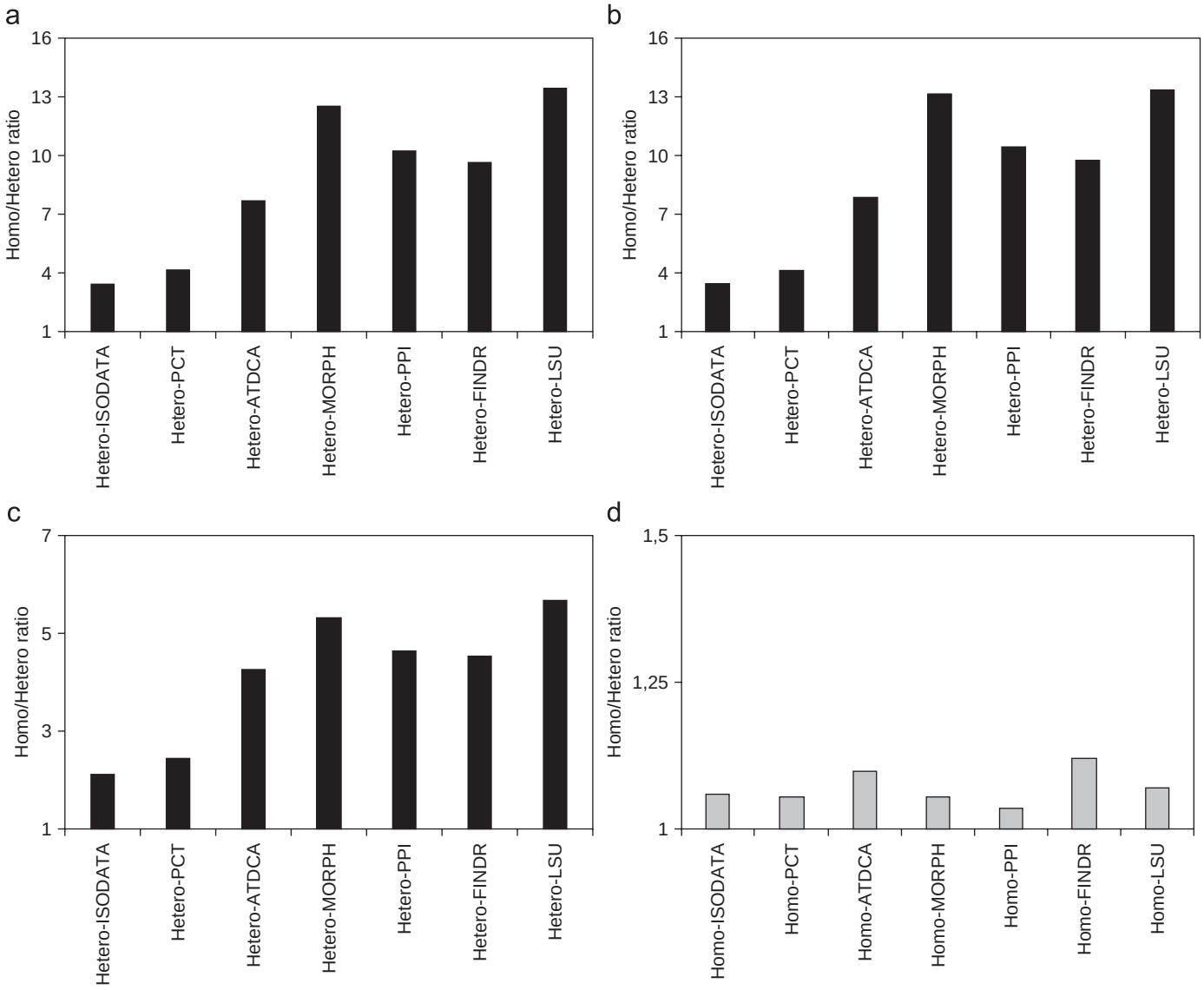

Fig. 4. Homo/Hetero ratios measured on the four considered parallel platforms. (a) Fully heterogeneous NOW. (b) Partially heterogeneous NOW. (c) Partially homogeneous NOW. (d) Fully homogeneous NOW.

homogeneous algorithms over their heterogeneous counterparts in the fully homogeneous NOW. As can be seen in Fig. 4(d), the homogeneous versions only slightly outperformed the heterogeneous algorithms in this platform. As a result, the ratios in Fig. 4(d) were all very close to one, a fact that reveals that the performance of the heterogeneous algorithms was almost the same as that evidenced by their respective homogeneous versions in the fully homogeneous NOW. The above results clearly demonstrate the flexibility of the proposed heterogeneous algorithms, which were able to adapt efficiently to all considered NOWs. Interestingly, after comparing the execution times of heterogeneous algorithms on the fully heterogeneous NOW with those achieved by their homogeneous counterparts on the fully homogeneous NOW (see Table 3), we noticed that the heterogeneous algorithms achieved essentially the same speed as their homogeneous versions, but each on its network. This indicated that the heterogeneous algorithms were very close to the optimal heterogeneous modifications of the basic homogeneous ones.

In order to further explore the parallel properties of the considered algorithms in more detail, an in-depth analysis of computation and communication times achieved by the different methods is also highly desirable. For that purpose, Table 6 shows the total time spent by the tested algorithms in communications and computations in the four considered NOWs, where two types of computation times were analyzed, namely, sequential (those performed by the root node with no other parallel tasks active in the system, labeled as SEQ in the table) and parallel (the rest of computations, i.e., those performed by the root node and/or the workers in parallel, labeled as PAR in the table). The latter includes the times in which the workers remain idle. It can be seen from Table 6 that, among all considered heterogeneous parallel algorithms, SEQ scores were particularly significant for the Hetero-ISODATA and Hetero-PCT

Table 6
Communication (COM), sequential computation (SEQ) and parallel computation (PAR) times in seconds measured for the parallel heterogeneous algorithms and their homogeneous versions on the four considered NOWs

Algorithm	Fully heterogeneous			Fully homogeneous			Partially heterogeneous			Partially homogeneous		
	COM	SEQ	PAR	COM	SEQ	PAR	COM	SEQ	PAR	COM	SEQ	PAR
Hetero-ISODATA	11	36	92	14	34	96	13	35	92	11	33	98
Homo-ISODATA	18		435	9		93	10		432	12		256
Hetero-PCT	6	27	99	9	28	99	8	26	99	8	27	100
Homo-PCT	12		523	5		96	7		514	9		294
Hetero-ATDCA	7	19	58	11	16	62	8	18	61	8	20	60
Homo-ATDCA	14		634	6		59	9		611	12		342
Hetero-MORPH	9	6	156	13	8	156	10	7	155	10	8	156
Homo-MORPH	17		2201	7		153	9		2187	11		906
Hetero-PPI	8	15	235	12	16	236	10	14	237	9	13	241
Homo-PPI	17		2687	5		234	6		2678	12		1192
Hetero-FINDR	4	17	30	7	14	35	6	17	32	8	16	32
Homo-FINDR	9		480	3		33	5		475	13		224
Hetero-LSU	2	2	111	6	3	112	3	2	1121	3	2	114
Homo-LSU	6		1587	2		108	4		1557	4		667

algorithms. As mentioned above, this is mainly due to the fact that these algorithms involve several steps based on sequential computations. SEQ scores were also relevant for the Hetero-ATDCA and Hetero-FINDR. It should be noted that Hetero-ATDCA involves several gather/scatter operations followed by compute-intensive orthogonal space projections at the master, which need to be completed in sequential fashion before a new parallel operation can be accomplished by the workers. Similar issues are also present in Hetero-FINDR (see steps 3 and 4 of this algorithm). Finally, the relatively high SEQ scores for the Hetero-PPI result from the last step of the algorithm, in which endmember selection is mainly accomplished at the master once all the workers have completed processing their respective data portions.

Quite opposite, although the Hetero-MORPH is the only technique that introduces redundant information (expected to slow down computations *a priori*), Table 6 reveals that the SEQ scores measured for this algorithm were comparable to those introduced by Hetero-LSU, and much lower than those reported by the other tested algorithms, in particular, in the fully heterogeneous NOW. As a result, the ratio of computations to communications for this method is much higher. This comes at no surprise, since the Hetero-MORPH algorithm is a windowing-type approach with very few data dependencies and, therefore, it is expected to scale better. In fact, the times reported by this algorithm were similar to those achieved by Hetero-LSU, but it should be noted that the former algorithm needs to be combined with a parallel endmember extraction technique to produce useful results from the viewpoint of spectral mixture analysis. Finally, it can also be seen from Table 6 that the cost of parallel (PAR) computations clearly dominated that of communications (COM) in all the considered parallel algorithms. For instance, the PAR scores achieved by the homogeneous algorithms executed on the (fully or partially) heterogeneous NOWs were extremely high, due to a less efficient workload distribution among the heterogeneous workers. To analyze this relevant issue in more detail, a study of load balance is highly required to fully substantiate the parallel properties of the considered algorithms.

In order to measure load balance, Table 7 shows the imbalance scores [21] achieved by the considered parallel algorithms on the four considered NOWs. The imbalance is defined as $D = R_{max}/R_{min}$, where R_{max} and R_{min} are the maxima and minima processor run times, respectively. 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}. As we can see from Table 7, only the Hetero-LSU and Hetero-MORPH were able to provide values of D_{All} close to 1 in all considered NOWs. Further, the above algorithms provided almost the same results for both D_{All} and D_{Minus} while, for the other tested methods, load balance was generally better when the root processor was not included. While the homogeneous algorithms executed on the (fully or partially) heterogeneous NOWs provided the highest values of D_{All} and D_{Minus} (and hence the highest imbalance), the heterogeneous algorithms executed on the homogeneous NOWs

Table 7
Load balancing rates for the parallel algorithms and their homogeneous versions executed on the four NOWs

Algorithm	Fully heterogeneous		Fully homogeneous		Partially heterogeneous		Partially homogeneous	
	D_{All}	D_{Minus}	D_{All}	D_{Minus}	D_{All}	D_{Minus}	D_{All}	D_{Minus}
Hetero-ISODATA	1.73	1.09	1.65	1.12	1.77	1.09	1.75	1.08
Homo-ISODATA	1.93	1.36	1.68	1.03	1.95	1.41	1.94	1.09
Hetero-PCT	1.69	1.06	1.58	1.03	1.72	1.05	1.68	1.07
Homo-PCT	1.81	1.28	1.56	1.05	1.82	1.39	1.83	1.08
Hetero-ATDCA	1.48	1.07	1.45	1.04	1.67	1.07	1.54	1.06
Homo-ATDCA	1.72	1.22	1.46	1.07	1.71	1.31	1.74	1.06
Hetero-MORPH	1.05	1.01	1.03	1.02	1.06	1.02	1.06	1.04
Homo-MORPH	1.59	1.21	1.05	1.01	1.62	1.24	1.28	1.13
Hetero-PPI	1.52	1.08	1.56	1.08	1.72	1.08	1.59	1.07
Homo-PPI	1.65	1.28	1.54	1.12	1.73	1.34	1.71	1.07
Hetero-FINDR	1.49	1.06	1.51	1.05	1.69	1.06	1.54	1.08
Homo-FINDR	1.68	1.25	1.54	1.11	1.75	1.34	1.77	1.09
Hetero-LSU	1.03	1.01	1.03	1.01	1.04	1.03	1.03	1.01
Homo-LSU	1.39	1.19	1.03	1.01	1.41	1.23	1.18	1.12

resulted in values of D_{Minus} which were close to 1. 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) and taking into account that Hetero-MORPH introduces redundant information expected to slow down the computation, results in Table 7 indicate that this algorithm seems to be more effective in terms of workload distribution than most other tested methods.

Taking into account the results presented above, and with the ultimate goal of exploring issues of scalability, considered to be a highly desirable property in heterogeneous parallel algorithms [14], we have also compared the performance of the parallel heterogeneous information extraction techniques on NASA's GSFC Thunderhead Beowulf cluster. For that purpose, Fig. 5 plots the speedups achieved by multi-processor runs of the heterogeneous parallel algorithms over their corresponding single-processor runs on Thunderhead. We experimentally tested that the scalability (and final processing times) provided by the homogeneous versions was essentially the same as that evidenced by the heterogeneous algorithms in Fig. 5. Therefore, only results for the proposed parallel algorithms are reported. It can be seen that Hetero-MORPH scaled better than the other clustering/classification algorithms tested, while the Hetero-PPI endmember extraction algorithm scaled significantly better than Hetero-FINDR. It is also clear that Hetero-LSU was the algorithm that reached the highest speedup factors, a fact that was previously observed in our experiments with the four NOWs.

For the sake of quantitative comparison, Table 8 reports the execution times achieved by the tested parallel heterogeneous algorithms on Thunderhead, using different numbers of processors. Results in Table 8 reveal that the tested algorithms were able to obtain relevant information from the considered

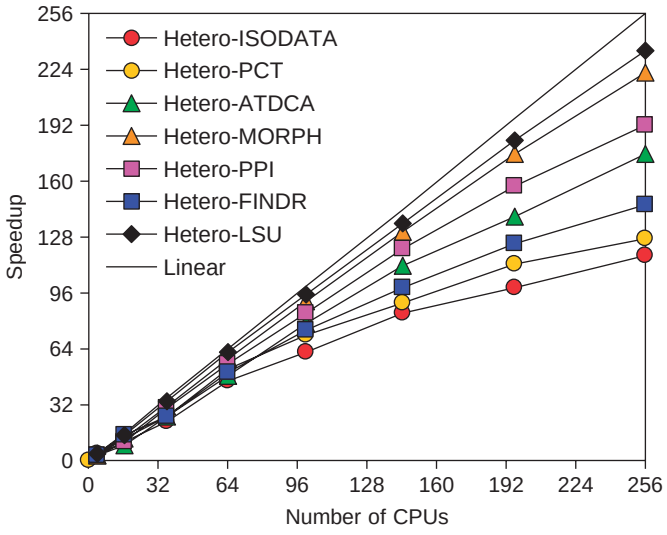

Fig. 5. Scalability of the parallel heterogeneous algorithms on NASA's GSFC Thunderhead Beowulf cluster.

hyperspectral data sets (in light of results in Tables 3 and 4), but also quickly enough for practical use. For instance, using 256 processors, the Hetero-MORPH algorithm provided a highly accurate classification result of the Indian Pines scene in 11 s, only 4 s more than ATDCA, which was the fastest algorithm. Since the single-processor run of Hetero-ATDCA was twice as fast as that of Hetero-MORPH, we can see that the latter algorithm scaled much better. On the other hand, the combination of Hetero-FINDR for endmember extraction followed by Hetero-LSU for spectral unmixing was able to provide an accurate estimation of fractional abundances at subpixel level in only 13 s. The above results indicate significant improvements over commonly used processing strategies for this kind of high-dimensional data sets, which can take up to

Table 8
Execution times (in seconds) for the parallel heterogeneous algorithms and their homogeneous versions on Thunderhead

Algorithm	Number of processors								
	1	4	16	36	64	100	144	196	256
Hetero-ISODATA	2072	648	202	93	45	33	24	21	18
Hetero-PCT	1884	460	154	73	36	26	21	17	15
Hetero-ATDCA	1263	493	141	49	26	16	11	9	7
Hetero-MORPH	2334	741	191	74	40	26	18	13	11
Hetero-PPI	4012	1638	388	135	72	48	33	26	21
Hetero-FINDR	916	286	63	36	18	12	9	7	6
Hetero-LSU	1724	470	117	51	28	18	13	9	7

more than one hour of computation for the considered problem size, as indicated by the single-processor execution times in Table 8.

Summarizing, experimental results in this paper revealed that heterogeneous parallel algorithms offer a relatively platform-independent and highly scalable solution to the problem of extracting useful information and knowledge from hyperspectral image data sets. Our quantitative assessment of different parallel strategies for image information mining represents a first attempt to provide a rigorous comparison of techniques which, in most cases, represent completely novel contributions (either from the perspective of designing computationally efficient versions of available sequential algorithms, or from the viewpoint of developing completely new parallel algorithms and techniques for this kind of high-dimensional image data). Contrary to common perception that spatial/spectral information extraction algorithms are too computationally demanding for practical use, our results demonstrate that such combined approaches may indeed be very appealing for parallel design and implementation, not only due to the window-based nature of such algorithms, but also because they can greatly reduce sequential computations at the master node and involve only minimal communication between the parallel tasks, namely, at the beginning and ending of such tasks. Our experimental results also revealed several important algorithmic aspects that may be of great importance for adapting existing hyperspectral imaging techniques to highly heterogeneous NOWs, which represent a cost-effective parallel computing platform for many scientific and engineering applications. In particular, we feel that the applicability of the proposed parallel heterogeneous methods may extend beyond the domain of hyperspectral image analysis. This is particularly true for the domains of signal processing and linear algebra applications, which include similar patterns of communication and calculation.

5. Conclusions and future work

In this paper, we have described some of the problems that need to be addressed in order to translate the tremendous advances in our ability to gather and store high-dimensional remotely sensed data into fundamental, application-oriented scientific advances through data-driven information extraction. We have presented and thoroughly analyzed several innovative parallel techniques for information mining from hyperspectral data sets, and implemented them on several heterogeneous and homogeneous computing platforms with the purpose of evaluating the possibility of obtaining results in valid response times and with adequate reliability in highly heterogeneous computing environments where these techniques are intended to be applied. The considered approaches include: parallel *clustering* approaches, able to associate together similar signature patterns in spectral space; parallel *classification* techniques, able to label each image pixel vector as member of a representative class in terms of its spectral, and spatial/spectral information; and parallel *spectral mixture analysis* methods, able to extract information at sub-pixel levels by decomposing mixed pixels into a collection of spectrally pure constituents with their corresponding fractional abundances. These methods offer a highly representative sample of available and new techniques in current hyperspectral analysis research which, despite the enormous computational demands and potential societal impact, has not yet developed standardized parallel-solution algorithms able to tackle high-dimensional data sets in low-cost parallel computing facilities such as distributed networks of workstations.

Heterogeneous systems are rapidly becoming a tool of choice in remote sensing missions, due to their low-cost, availability (workstations are everywhere, and it is not difficult to put together a network and/or a cluster, given the raw materials) and scalability. To address specific issues involved in the exploitation of heterogeneous computing systems for image information extraction, this paper provided a detailed discussion on the effects that platform heterogeneity has on degrading parallel performance of hyperspectral analysis techniques. An interesting finding by experiments is that spatial/spectral heterogeneous approaches offer a surprisingly simple, yet effective and highly scalable approach. The parallel performance evaluation strategy conducted in this work was based on experimentally assessing heterogeneous algorithms by comparing their efficiency on (fully or partially) heterogeneous networks of workstations with the efficiency achieved by their homogeneous versions on equally powerful homogeneous networks. Combining the readily available computational power offered by heterogeneous architectures with last-generation sensor and parallel processing technology may introduce substantial changes in the systems currently used by NASA and other agencies for exploiting the sheer volume of Earth and planetary remotely sensed data collected on a daily basis.

As future work, we plan to implement the proposed parallel techniques on other massively parallel computing architectures, such as NASA's Project Columbia or CEPBA's Mare Nostrum supercomputer. We are also developing field programmable gate array implementations, which may allow us to fully accomplish the goal of real-time, onboard information extraction from hyperspectral data sets. We also envision closer multidisciplinary collaborations with environmental scientists to address global monitoring land services and security issues which are the main concern of several on-going and planned remote sensing missions.

Acknowledgments

This research was supported by the European Commission through the Marie Curie project "Hyperspectral Imaging Network," (contract number MRTN-CT-2006-035927). Additional funding from Junta de Extremadura (local government) under project 2PR03A026 is gratefully acknowledged. The author would like to thank Drs. John E. Dorband, James C. Tilton and J. Anthony Gualtieri for many helpful discussions. He would also like to acknowledge support received from the Spanish Ministry of Education and Science (Fellowship PR2003-0360), which allowed him to conduct postdoctoral research at NASA's Goddard Space Flight Center and University of Maryland in 2004.

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] J.W. Boardman, F.A. Kruse, R.O. Green, Mapping target signatures via partial unmixing of AVIRIS data, in: Summaries of JPL Airborne Earth Science Workshop, Pasadena, CA, 1995.
[4] 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.
[5] C.-I. Chang, Hyperspectral Imaging: Techniques for Spectral Detection and Classification, Kluwer Academic Publishers, Dordrecht, 2003.
[6] C.-I. Chang, C.-C. Wu, W.-M. Liu, Y.-C. Ouyang, A new growing method for simplex-based endmember extraction algorithm, IEEE Trans. Geosci. Remote Sensing 44 (2006) 2804–2819.
[7] L. Chen, I. Fujishiro, K. Nakajima, Optimizing parallel performance of unstructured volume rendering for the Earth Simulator, Parallel Comput. 29 (2003) 355–371.
[8] M. Datcu, K. Seidel, Human centered concepts for exploration and understanding of Earth Observation images, IEEE Trans. Geosci. Remote Sensing 43 (2005) 601–609.
[9] 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.
[10] J. Dorband, J. Palencia, U. Ranawake, Commodity computing clusters at Goddard Space Flight Center, J. Space Commun. 1 (3) (2003).
[11] T. El-Ghazawi, S. Kaewpijit, J. Le Moigne, Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality, in: Proceedings of the 3rd International Conference on Cluster Computing, 2001, p. 102.
[12] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[13] R.O. Green, et al., Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS), Remote Sensing Environ. 65 (1998) 227–248.
[14] A. Kalinov, Scalability analysis of matrix-matrix multiplication on heterogeneous clusters, in: Proceedings of ISPDC'2004/HeteroPar'04, IEEE Computer Society, Silver Spring, MD, 2004.
[15] A. Kalinov, A. Lastovetsky, Y. Robert, Heterogeneous computing, Parallel Comput. 31 (2005) 649–652.
[16] S. Kalluri, Z. Zhang, J. JaJa, S. Liang, J. Townshend, Characterizing land surface anisotropy from AVHRR data at a global scale using high performance computing, Int. J. Remote Sensing 22 (2001) 2171–2191.
[17] N. Keshava, J.F. Mustard, Spectral unmixing, IEEE Signal Process. Mag. 19 (2002) 44–57.
[18] D.A. Landgrebe, Signal Theory Methods in Multispectral Remote Sensing, Wiley, Hoboken, NJ, 2003.
[19] A. Lastovetsky, Parallel Computing on Heterogeneous Networks, Wiley-Interscience, Hoboken, NJ, 2003.
[20] A. Lastovetsky, R. Reddy, On performance analysis of heterogeneous parallel algorithms, Parallel Comput. 30 (2004) 1195–1216.
[21] 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.
[22] A. Plastino, C.C. Ribeiro, N. Rodriguez, Developing SPMD applications with load balancing, Parallel Comput. 29 (2003) 743–766.
[23] 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.
[24] 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.
[25] A. Plaza, D. Valencia, J. Plaza, P. Martinez, Commodity cluster-based parallel processing of hyperspectral imagery, J. Parallel Distrib. Comput. 66 (3) (2006) 345–358.
[26] J. Richards, X. Jia, Remote Sensing Digital Image Analysis, fourth ed., Springer, Berlin, 2005.
[27] C.-R. Shyu, M. Klaric, G. Scott, A. Barb, C. Davis, K. Palaniappan, GeoIRIS: geospatial information retrieval and indexing system—content mining, semantics modeling, and complex queries, IEEE Trans. Geosci. Remote Sensing 45 (2007) 839–852.
[28] S. Tehranian, Y. Zhao, T. Harvey, A. Swaroop, K. McKenzie, A robust framework for real-time distributed processing of satellite data, J. Parallel Distrib. Comput. 66 (2006) 403–418.
[29] J.C. Tilton, G. Marchisio, K. Koperski, M. Datcu, Image information mining utilizing hierarchical segmentation, IEEE International Geoscience and Remote Sensing Symposium, vol. 2, 2002, pp. 1029–1031.
[30] P. Wang, K.Y. Liu, T. Cwik, R.O. Green, MODTRAN on supercomputers and parallel computers, Parallel Comput. 28 (2002) 53–64.
[31] M.E. Winter, N-FINDR: an algorithm for fast autonomous spectral endmember determination in hyperspectral data, in: Proceedings of SPIE Imaging Spectrometry Conference, vol. 3753, 1999, pp. 266–277.
[32] M.E. Winter, A proof of the N-FINDR algorithm for the automated detection of endmembers in a hyperspectral image, in: Proceedings of SPIE, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery X, vol. 5425, 2004, pp. 31–41.

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 a 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 150 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 and a Special Issue on "High Performance Computing for Hyperspectral Imaging" for the International Journal of High Performance Computing Applications. Dr. Plaza is the project coordinator of Hyperspectral Imaging Network, a four-year Marie Curie Research Training Network designed to build an interdisciplinary European research community focused on hyperspectral imaging activities. Dr. Plaza has served has served as an expert evaluator for the European Commission (Engineering panel, FP7), the European Space Agency, the Belgium Science Foundation, and the Spanish Ministry of Science and Education. He is a Senior Member of IEEE and an Associate Editor for the IEEE Transactions on Geoscience and Remote Sensing journal in the areas of Hyperspectral Image Analysis and Signal Processing.