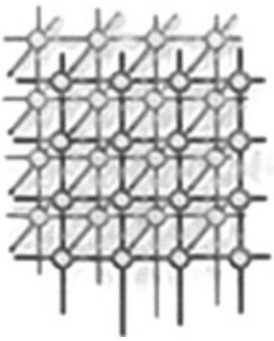
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2010; **22**:1138–1159
Published online 9 December 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1555



Parallel heterogeneous CBIR system for efficient hyperspectral image retrieval using spectral mixture analysis

Antonio J. Plaza*, †, Javier Plaza and Abel Paz

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

SUMMARY

The purpose of content-based image retrieval (CBIR) is to retrieve, from real data stored in a database, information that is relevant to a query. In remote sensing applications, the wealth of spectral information provided by latest-generation (hyperspectral) instruments has quickly introduced the need for parallel CBIR systems able to effectively retrieve features of interest from ever-growing data archives. To address this need, this paper develops a new parallel CBIR system that has been specifically designed to be run on heterogeneous networks of computers (HNOCs). These platforms have soon become a standard computing architecture in remote sensing missions due to the distributed nature of data repositories. The proposed heterogeneous system first extracts an image feature vector able to characterize image content with sub-pixel precision using spectral mixture analysis concepts, and then uses the obtained feature as a search reference. The system is validated using a complex hyperspectral image database, and implemented on several networks of workstations and a Beowulf cluster at NASA's Goddard Space Flight Center. Our experimental results indicate that the proposed parallel system can efficiently retrieve hyperspectral images from complex image databases by efficiently adapting to the underlying parallel platform on which it is run, regardless of the heterogeneity in the compute nodes and communication links that form such parallel platform. Copyright © 2009 John Wiley & Sons, Ltd.

Received 9 January 2009; Revised 30 September 2009; Accepted 19 October 2009

KEY WORDS: heterogeneous parallel computing; image processing; hyperspectral imaging

*Correspondence to: Antonio J. Plaza, Department of Technology of Computers and Communication, University of Extremadura, Avda. de la Universidad s/n, E-10071 Caceres, Spain.
†E-mail: aplaza@unex.es

Contract/grant sponsor: European Community's Marie Curie Research Training Networks Programme; contract/grant number: MRTN-CT-2006-035927
Contract/grant sponsor: Hyperspectral Imaging Network (HYPER-I-NET)
Contract/grant sponsor: Spanish Ministry of Science and Innovation; contract/grant number: AYA2008-05965-C04-02
Contract/grant sponsor: Junta de Extremadura; contract/grant number: PRI09A110

1. INTRODUCTION

Content-based image retrieval (CBIR) systems offer mechanisms for selecting the data items that resemble most a specific query among all the available information in a database [1,2]. A major challenge for the development of efficient CBIR systems in the context of remote sensing applications is how to deal with the extremely large volumes of data produced by current Earth-observing imaging spectrometers [3]. The multispectral nature [4] of those systems is crucial in applications such as environmental studies, target detection for military purposes or risk/hazard prevention and response. For instance, the NASA Jet Propulsion Laboratory's Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) [5] is able to record the visible and near infrared spectrum of the reflected light of an area several kilometers long (depending on the duration of the flight) using hundreds of spectral bands. The resulting 'image cube' is a stack of images (Figure 1), in which each pixel (vector) has an associated spectral signature or 'fingerprint' that uniquely characterizes the underlying objects. The resulting data often comprises several Gigabytes per flight.

Most available parallel systems used by institutions such as NASA or the European Space Agency during the last decade for hyperspectral data processing have been homogeneous in nature. For instance, Beowulf clusters have been used to access greatly increased computational power, but at a low cost (commensurate with falling commercial PC costs) in a number of remote sensing applications [6–10]. Despite the success of the Beowulf systems in hyperspectral imaging applications, a recent trend in the design of high-performance systems for data-intensive problems in remote sensing is the use of heterogeneous computing resources distributed among different locations [11]. It has been shown that such heterogeneous networks of computers (HNOCs) can realize a very high level of the aggregate performance in both generic [12,13] and remote sensing applications [14], and the pervasive availability of these resources has resulted in the progressive incorporation of the concept of grid computing [15] into remote sensing studies, with the ultimate goal of making distributed collections of data easy to access from different users. Although it is expected that heterogeneous and grid-based CBIR systems will soon represent a tool of choice in many scientific applications [16,17], very few efforts have been developed toward the design of computationally efficient CBIR systems for remote sensing image retrieval on fully heterogeneous platforms, i.e. in systems made up of distributed heterogeneous processors that communicate through links with different capacities.

In this paper, we describe a new parallel CBIR system for information extraction and mining from remote sensing data repositories. The system has been specifically designed to be run on HNOCs. The main objective of this paper is to analyze if HNOCs can serve as a baseline parallel system for efficient implementation of hyperspectral image retrieval algorithms from large data repositories, using spectral mixture analysis concepts when conducting the content-based image search. This strategy is expected to simplify the procedure of searching hyperspectral images by content, due to the special properties of hyperspectral imagery which can be well captured using spectral unmixing concepts. In order to address this general objective, we pursue the following specific objectives:

1. To develop a spectral mixture analysis-based CBIR system that can assist users in the task of efficiently searching hyperspectral image instances in large data repositories.

DOI: 10.1002/cpe

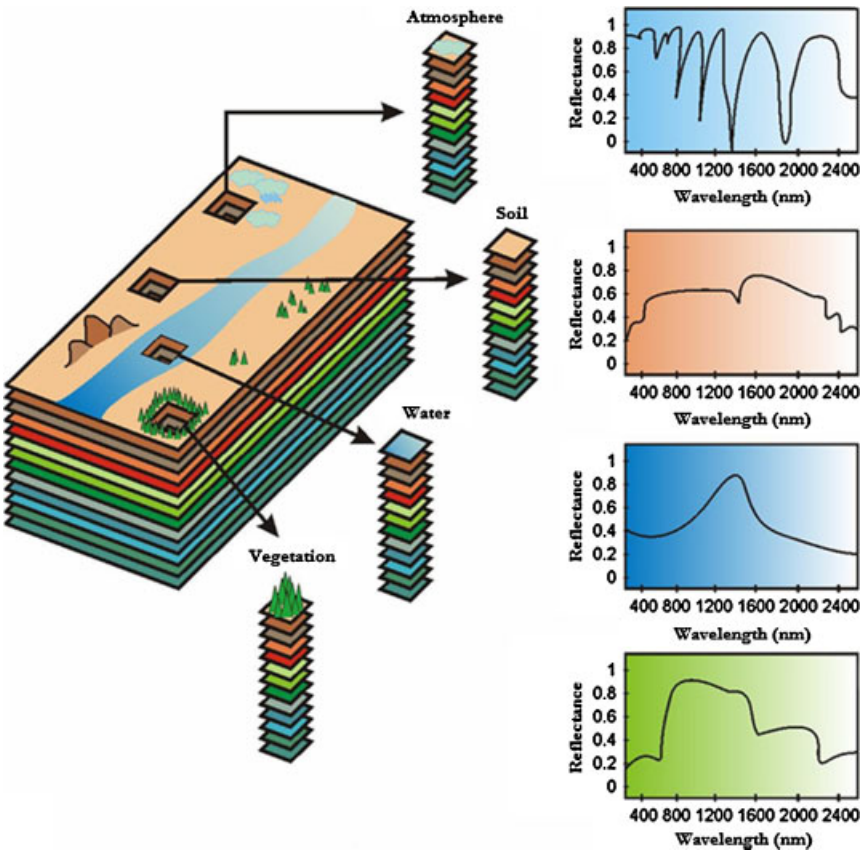


Figure 1. The concept of hyperspectral imaging.

2. To implement the proposed system in HNOCs so that it can be run in distributed parallel platforms made up of different processing nodes connected by heterogeneous communication links.
3. To validate the proposed system in terms of both hyperspectral image retrieval accuracy and parallel performance, with particular attention to the scalability and load balancing achieved in heterogeneous platforms.

The paper is structured as follows. Section 2 describes the spectral unmixing-based hyperspectral image retrieval methodology used to implement the core of our CBIR system. Section 3 describes its parallel heterogeneous implementation. Section 4 assesses the performance of the system by comparing its accuracy and parallel properties using several heterogeneous and homogeneous networks, and a massively parallel Beowulf cluster. Finally, Section 5 concludes with some remarks and hints at the plausible future research.

DOI: 10.1002/cpe

2. HYPERSPECTRAL IMAGE RETRIEVAL METHODOLOGY

2.1. Problem formulation

Let us assume that a remotely sensed hyperspectral scene with N bands is denoted by $\mathbf{F}$, in which a pixel of the scene is represented by a vector $\mathbf{f}_i = [f_{i1}, f_{i2}, \ldots, f_{in}] \in \Re^N$, where $\Re$ denotes the set of real numbers in which the pixel's spectral response f_{ik} at sensor channels $k = 1, \ldots, N$ is included. Under the linear mixture model assumption, each pixel vector in the original scene can be modeled using the following expression:

$$\mathbf{f}_i = \sum_{e=1}^{E} \mathbf{e}_e \cdot a_{\mathbf{e}_e} + \mathbf{n} \tag{1}$$

where $\mathbf{e}_e$ denotes the spectral response of a pure spectral signature (endmember in hyperspectral imaging terminology), $a_{\mathbf{e}_e}$ is a scalar value designating the fractional abundance of the endmember $\mathbf{e}_e$, E is the total number of endmembers, and $\mathbf{n}$ is a noise vector. The use of spectral endmembers allows one to deal with the problem of mixed pixels, which arise when the spatial resolution of the sensor is not high enough to separate different materials [18,19]. For instance, it is likely that the pixel labeled as 'vegetation' in Figure 1 actually comprises a mixture of vegetation and soil. In this case, the measured spectrum can be decomposed into a linear combination of pure spectral endmembers of soil and vegetation, weighted by abundance fractions that indicate the proportion of each endmember in the mixed pixel.

The solution of the linear spectral mixture problem described in (1) relies on a successful estimation of how many endmembers, E, are present in the input hyperspectral scene $\mathbf{F}$, and also on the correct determination of a set $\{\mathbf{e}_e\}_{e=1}^{E}$ of endmembers and their correspondent abundance fractions $\{a_{\mathbf{e}_e}\}_{e=1}^{E}$ at each pixel $\mathbf{f}_i$. Two physical constraints are generally imposed into the model described in (1), these are the abundance non-negativity constraint (ANC), i.e. $a_{\mathbf{e}_e} \geq 0$, and the abundance sum-to-one constraint (ASC), i.e. $\sum_{e=1}^{E} a_{\mathbf{e}_e} = 1$ [20].

2.2. Proposed methodology

The image content retrieval methodology used in this work to describe hyperspectral data sets consists of two main stages:

1. *Endmember extraction*: First, a set of pure spectral signatures (endmembers) are extracted from the input data set. For this purpose, we have considered a standard algorithm in the literature: Boardman's pixel purity index (PPI) [21]. This algorithm has been widely used in hyperspectral image analysis for endmember extraction due to its publicity and availability in the environment for visualizing images software[‡] originally developed by Analytical Imaging and Geophysics. The algorithm searches for a set of vertices of a convex hull in a given data set, which are supposed to be pure signatures present in the data. The concept of the algorithm is illustrated by a toy example in Figure 2.

[‡]http://www.ittvis.com.

DOI: 10.1002/cpe

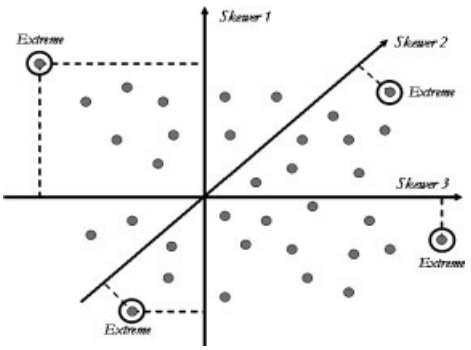


Figure 2. Toy example illustrating the performance of the PPI algorithm used for endmember extraction.

2. *Fully constrained linear spectral unmixing* (*FCLSU*): Using the set of endmembers extracted in the previous stage, we estimate the fractional abundances of such endmembers in each pixel of the scene using a fully constrained linear spectral unmixing (FCLSU) algorithm that incorporates both the ANC and the ASC constraints. The algorithm adopted in this work was proposed by Heinz and Chang [20,22].
3. *Spectral signature matching*: Once a set of image endmembers have been extracted, along with their correspondent abundances, the similarity of a pair of images can be determined by a *spectral matching* procedure in which the spectral endmembers of an input (test) image are sorted and then compared with the ordered endmembers of each of a set of reference images stored in a database [19]. After the endmember spectra of the two images have been matched by means of a spectral similarity metric, their corresponding abundance fractions are compared and used to produce a feature vector representing the fractional proportions of endmembers in the compared images.

In the following, we describe the endmember extraction algorithm (EEA) used in this work and also the spectral similarity matching algorithm used to perform signature comparison and sorting. For convenience, and in order to save space, from now on we assume that spectral unmixing is the last step of the EEA, although it is important to emphasize that different unmixing algorithms can use the endmembers provided by the PPI (or by a different algorithm) to produce final fractional abundance estimates.

2.3. Endmember extraction algorithm (EEA)

The inputs to the algorithm are a hyperspectral image cube $\mathbf{F}$ with N spectral bands; a maximum number of projections, K; a cutoff threshold value, v_c, used to select as final endmembers only those pixels that have been selected as extreme pixels at least v_c times throughout the process; and a threshold angle, v_a, used to discard redundant endmembers during the process. The output of the algorithm is a set of E final endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$. The algorithm can be summarized by the

DOI: 10.1002/cpe

following steps [21]:

1. *Skewer generation*: Produce a set of K randomly generated unit vectors, denoted by $\{skewer_j\}_{j=1}^{K}$.
2. *Extreme projections*: For each $skewer_j$, all sample pixel vectors $\mathbf{f}_i$ in the original data set $\mathbf{F}$ are projected onto $skewer_j$ via products of $|\mathbf{f}_i \cdot skewer_j|$ to find sample vectors at its extreme (maximum and minimum) projections, forming an extrema set for $skewer_j$, which is denoted by $S_{extrema}(skewer_j)$.
3. *Calculation of pixel purity scores*: Define an indicator function of a set S, denoted by $I_S(\mathbf{x})$, to denote membership of an element $\mathbf{x}$ to that particular set as $I_S(\mathbf{f}_i) = 1$ if $\mathbf{x} \in S$. Using the indicator function above, calculate the number of times that given pixel has been selected as extreme using the following equation:

$$N_{times}(\mathbf{f}_i) = \sum_{j=1}^{K} I_{S_{extrema}(skewer_j)}(\mathbf{f}_i) \tag{2}$$

4. *Endmember selection*: Use the virtual dimensionality concept in [22,23] to estimate the number of endmembers E in the input image. Then find the pixels with the value of $N_{times}(\mathbf{f}_i)$ above v_c and form a unique set of endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$ by calculating the spectral angle distance (SAD) for all possible endmember pairs and discarding those that result in an angle value below v_a. SAD is a standard similarity metric for remote sensing operations, which is invariant to unknown multiplicative scalings that may arise due to differences in the illumination and sensor observation angle [3]. The SAD between endmember $\mathbf{e}_i$ and endmember $\mathbf{e}_j$ is defined as follows:

$$\mathrm{SAD}(\mathbf{e}_i, \mathbf{e}_j) = cos^{-1} \frac{\mathbf{e}_i \cdot \mathbf{e}_j}{\|\mathbf{e}_i\| \cdot \|\mathbf{e}_j\|} \tag{3}$$

5. *Spectral unmixing*: For each sample pixel vector $\mathbf{f}_i$ in $\mathbf{F}$, a set of abundance fractions specified by $\{a_{\mathbf{e}_1}, a_{\mathbf{e}_2}, \ldots, a_{\mathbf{e}_E}\}$ and satisfying the ASC and ANC constraints are obtained using the set of endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$, so that each $\mathbf{f}_i$ can be expressed as a linear combination of endmembers as follows, using the FCLSU algorithm proposed by Heinz and Chang [20,22]:

$$\mathbf{f}_i = \mathbf{e}_1 \cdot a_{\mathbf{e}_1} + \mathbf{e}_2 \cdot a_{\mathbf{e}_2} + \cdots + \mathbf{e}_E \cdot a_{\mathbf{e}_E} \tag{4}$$

2.4. Spectral signature matching algorithm (SSMA)

Let $\{\mathbf{e}_e\}_{e=1}^{E}$ be a set of E endmembers extracted from a test image, and let $\{\mathbf{r}_r\}_{r=1}^{R}$ be a set of R endmembers extracted from a reference image in the database. It should be noted that the reference endmembers are stored for each image data set cataloged in the system as part of the image header file, which also contains information about the image dimensions, application domain, etc. In order to match endmembers in the test set to endmembers in the reference set, a SAD-based spectral similarity criterion is implemented using the following steps:

1. *Initial labeling*: Label all endmembers in the test set $\{\mathbf{e}_e\}_{e=1}^{E}$ as 'unmatched.'
2. *Matching*: For each unmatched endmember in the test set $\{\mathbf{e}_e\}_{e=1}^{E}$, calculate the spectral angle between the test endmember and all endmembers in the reference set $\{\mathbf{r}_r\}_{r=1}^{R}$. If the pair

DOI: 10.1002/cpe

$(\mathbf{e}_k, \mathbf{r}_j)$, with $1 \leq k \leq E$ and $1 \leq j \leq R$, results in the minimum obtained value of $\mathrm{SAD}(\mathbf{e}_k, \mathbf{r}_j)$, and the value is below the threshold angle v_a, then label the associated endmembers, $\mathbf{e}_k$ and $\mathbf{r}_j$ as 'matched.'

3. *Relative difference calculation*: For each matched endmember $\mathbf{e}_k$ resulting from the previous step, calculate $|a_{\mathbf{e}_k} - a_{\mathbf{r}_j}|$, i.e. the relative difference between the abundance fraction associated with endmember $\mathbf{e}_k$ in the test image and the abundance associated with its matched endmember $\mathbf{r}_j$ in the reference image. The resulting values are used as a feature vector for signature comparison when searching the database.

3. HETEROGENEOUS CBIR SYSTEM

In this section we describe the proposed parallel CBIR system. First, we describe the optimization problem in the context of fully heterogeneous networks. Then, we briefly discuss data partitioning strategies and further provide a heterogeneous parallel implementation aimed at maximizing the load balance.

3.1. Optimization problem

A fully heterogeneous network can be modeled as a complete graph, where each node models a computing resource p_i weighted by its relative cycle-time t_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 assume that the system has symmetric costs, i.e. $c_{ij} = c_{ji}$). With the above assumptions in mind [24], processor p_i should accomplish a share of $\alpha_i \cdot W$ of the total workload, denoted by W, to be performed by a certain algorithm, with $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$, being P the total number of processors in the system.

In order to take into account the communication time to distribute the workload among the processing nodes, *diffusion* algorithms have been shown to be able to improve the load balancing of the system in some situations [25]. These algorithms migrate a load fraction (flow) over the topology's communication links, depending on the workload difference to its neighbors. An important aspect in this context is the local calculation of the flow. The calculation has to be fast, in order not to increase the overall computation time. Thus, a global gathering of the load information and the distribution of the calculated flow over the whole network results in a high communication overhead. In this work, we adopt a local iterative load balancing scheme in which each processor exchanges information only with its neighbors when seeking for an appropriate division of the original workload, as reported in [25,26].

In this context, it should also be taken into account that the workload in real-world applications usually cannot be divided arbitrarily, but only to some extent. The unit-size token model [27] assumes a smallest load entity, called indivisible token, so that the workload is always represented by a multiple of this smallest entity. Given the characteristics of the hyperspectral processing algorithms described in Section 2, which work on a pixel-by-pixel basis, in our specific application domain each pixel vector will be regarded as a unit-size token.

DOI: 10.1002/cpe

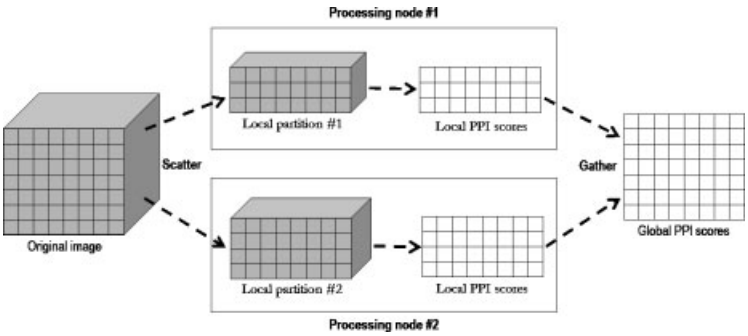


Figure 3. Spatial-domain decomposition approach adopted for the partitioning of hyperspectral data.

Taking into account the optimization problem above, an abstract view of our proposed CBIR implementation can be simply stated in the form of a client–server architecture, in which a server processor receives an input image and distributes work according to the processing speeds of P heterogeneous nodes, and the capacities of the heterogeneous communication links that connect such nodes, so that image features are obtained in parallel. Once the image features have been obtained at the master, this node performs a parallel search by comparing the features with those for other images in the database. In the course of this process, some communications between the master and the workers also take place [10].

3.2. Hyperspectral data partitioning

The proposed system adopts a partitioning strategy in which the data are always partitioned into blocks made up of spatially adjacent pixel vectors that retain the full spectral content associated with them (Figure 3).

The main advantage of the spatial-domain decomposition approach in Figure 3 is that the cost of inter-processor communication is reduced, as shown in the previous work [10,18,24]. At this point, it is important to emphasize that spatial-domain partitioning should be used with extra care when parallelizing data processing techniques that include sample spectral correlation and/or covariance calculations, such as the principal component transform [28] or the RX detector developed by Reed and Yu [29], both of them widely used in hyperspectral image analysis. In those cases, important aspects such as the size of the spatial-domain partitions or the need to overlap adjacent partitions arise. These aspects have been addressed in the previous work [30,31].

With the above issues in mind, the two major goals of our partitioning algorithm are: (i) to obtain an appropriate set of workload fractions $\{\alpha_i\}_{i=1}^{P}$ that best fit the heterogeneous environment and (ii) to translate the chosen set of values into a suitable decomposition of the input data, taking into account the properties of the heterogeneous system. To accomplish such goals, we have developed a workload estimation algorithm (WEA) that assumes that the workload of each processor p_i must be directly proportional to its local memory and inversely proportional to its cycle-time t_i. The algorithm also takes into account the communication time to distribute the workload among

DOI: 10.1002/cpe

the nodes. It performs the following operations:

1. Obtain necessary information about the heterogeneous system, including the number of available processing nodes P, each processor's identification number $\{p_i\}_{i=1}^{P}$, and processor cycle-times $\{t_i\}_{i=1}^{P}$.
2. Set

$$\alpha_i = \left\lfloor \frac{(1/t_i)}{\sum_{i=1}^{P}(1/t_i)} \right\rfloor$$

for all $i \in \{1, \ldots, P\}$. In other words, this step first approximates the $\{\alpha_i\}_{i=1}^{P}$ so that the amount of work assigned to each processing node is proportional to its speed and $\alpha_i \cdot t_i \approx$ const for all processors.
3. Iteratively increment some α_i until the set of $\{\alpha_i\}_{i=1}^{P}$ best approximates the total workload to be completed, i.e. for $m = \sum_{i=1}^{P}\alpha_i$ to W, find $k \in \{1, \ldots, P\}$ so that $t_k \cdot (\alpha_k + 1) = min\{t_i \cdot (\alpha_i + 1)\}_{i=1}^{P}$, and then set $\alpha_k = \alpha_k + 1$.
4. Produce P partitions of the input hyperspectral data set, so that the spectral channels corresponding to the same pixel vector (unit-size token) are never stored in different partitions. In order to achieve this goal, we have adopted a methodology that consists of three main steps:

   (a) The hyperspectral data set is first partitioned, using spatial-domain decomposition, into a set of vertical slabs that retain the full spectral information at the same partition (Figure 3). The number of rows in each slab is set to be proportional to the estimated values of $\{\alpha_i\}_{i=1}^{P}$, and assuming that no upper bound exist on the number of pixel vectors that can be stored by the local memory at the considered node.
   (b) For each processor p_i, check if the number of pixel vectors assigned to it is greater than the upper bound. For all the processors whose upper bounds are exceeded, assign them a number of pixels equal to their upper bounds. Now, we solve the partitioning problem of a set with remaining pixel vectors over the remaining processors. We recursively apply this procedure until all the pixel vectors in the input data have been assigned, thus obtaining an initial workload distribution for each p_i. It should be noted that, with the proposed algorithm description, it is possible that all processors exceed their upper bounds. This situation was never observed in our experiments. However, if the considered network includes processing units with low memory capacity, this situation could be handled by allocating an amount of data equal to the upper bound to those processors, and then processing the remaining data as an offset in a second algorithm iteration.
   (c) Iteratively recalculate the workload assigned to each processor using the following expression:

$$W_i^k = W_i^{k-1} - \sum_{j \in N(i)} c_{ij}\left(\frac{W_i^{k-1}}{t_i} - \frac{W_j^{k-1}}{t_j}\right) \tag{5}$$

where $N(i)$ denotes the set of neighbors of processing node p_i, and W_i^k denotes the workload of p_i (i.e. the number of unit-size tokens or pixel vectors assigned to this

DOI: 10.1002/cpe

processor) after the kth iteration. This scheme has been demonstrated in the previous work to converge to an average workload [25]

$$\overline{W_i} := \frac{\sum_{j=1}^{P} W_j}{\sum_{j=1}^{P} t_j} t_i$$

To conclude this subsection, we would like to emphasize that we have also tested a simplified version of the WEA algorithm above, which does not take into account the communication capacity of heterogeneous communication links in the workload estimation process. The simplified version, referred to hereinafter as SWEA, simply does not execute step 4(c) of the WEA algorithm.

3.3. Parallel EEA

To reduce the code redundancy and enhance reusability, our goal, when designing the parallel EEA, was to reuse much of the code for the sequential one, as indicated by the master–slave parallel implementation given below:

1. *Data partitioning*: Produce a set of L spatial-domain heterogeneous partitions of $\mathbf{F}$ using the WEA algorithm.
2. *Skewer generation*: Generate K random unit vectors $\{skewer_j\}_{j=1}^{K}$ in parallel, and broadcast the entire set of skewers to all the workers.
3. *Extreme projections*: For each $skewer_j$, project all the sample pixel vectors at each local partition l onto $skewer_j$ to find sample vectors at its extreme projections, and form an extrema set for $skewer_j$, which is denoted by $S^{(l)}_{extrema}(skewer_j)$. Now calculate the number of times each pixel vector $\mathbf{f}_i^{(l)}$ in the local partition is selected as extreme using the following expression:

$$N^{(l)}_{times}(\mathbf{f}_i^{(l)}) = \sum_{j=1}^{K} I_{S^{(l)}_{extrema}(skewer_j)}(\mathbf{f}_i^{(l)}) \tag{6}$$

4. *Candidate selection*: Select those pixel vectors with $N^{(l)}_{times}(\mathbf{f}_i^{(l)}) > v_c$ and send them to the master node.
5. *Endmember selection*: The master calculates the total number of endmembers in the input image using the virtual dimensionality concept in [22,23] and forms a unique set $\{\mathbf{e}_e\}_{e=1}^{E}$ by calculating the SAD for all possible pixel vector pairs provided by the workers in parallel, and discarding those pixels that result in angle values below v_a.
6. *Spectral unmixing*: The master broadcasts the set of endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$ to all workers. Each worker then obtains a set of fractional abundances $\{a^{(l)}_{\mathbf{e}_1}, a^{(l)}_{\mathbf{e}_2}, \ldots, a^{(l)}_{\mathbf{e}_E}\}$ for each pixel vector $\mathbf{f}_i^{(l)}$ in its local partition l, using the set $\{\mathbf{e}_e\}_{e=1}^{E}$ so that the following expression is satisfied:

$$\mathbf{f}_i^{(l)} = \mathbf{e}_1 \cdot a^{(l)}_{\mathbf{e}_1} + \mathbf{e}_2 \cdot a^{(l)}_{\mathbf{e}_2} + \cdots + \mathbf{e}_E \cdot a^{(l)}_{\mathbf{e}_E} \tag{7}$$

DOI: 10.1002/cpe

3.4. Parallel SSMA

As in our design of the parallel EEA, our goal is to reuse the code for the sequential SSMA. The master–slave implementation of this algorithm is shown as follows:

1. *Data partitioning and initial labeling*: The master processor produces a set of L spatial-domain heterogeneous partitions of $\mathbf{F}$ using the WEA algorithm and labels all endmembers in the test set $\{\mathbf{e}_e\}_{e=1}^{E}$ as 'unmatched.'
2. *Matching*: For each unmatched endmember in the test set $\{\mathbf{e}_e\}_{e=1}^{E}$, calculate (in parallel) the spectral angle between the test endmember and all endmembers in the reference set $\{\mathbf{r}_r\}_{r=1}^{R}$. If the pair $(\mathbf{e}_k, \mathbf{r}_j)$, with $1 \leq k \leq E$ and $1 \leq j \leq R$, results in the minimum obtained value of $\mathrm{SAD}(\mathbf{e}_k, \mathbf{r}_j)$, and the value is below the threshold angle v_a, then label the associated endmembers, $\mathbf{e}_k$ and $\mathbf{r}_j$ as 'matched.'
3. *Relative difference calculation*: For each matched endmember $\mathbf{e}_k$ resulting from the previous step, calculate (in parallel) $|a_{\mathbf{e}_k} - a_{\mathbf{r}_j}|$, i.e. the relative difference between the abundance fraction associated with endmember $\mathbf{e}_k$ in the test image and the abundance associated with its matched endmember $\mathbf{r}_j$ in the local partition. The resulting values are used as a feature vector for signature comparison when searching the database.

The proposed parallel algorithms for endmember extraction and spectral signature matching have been implemented in the C++ programming language, using calls to message passing interface (MPI) [32]. We resorted to MPI-*derived data types* to directly scatter the hyperspectral data structures, which may be stored non-contiguously in memory, in a single communication step. As a result, we avoid creating all partial data structures on the master node (thus making a better use of memory resources and compute power).

3.5. Search procedure

To conclude this section devoted to the algorithmic description of our system, we briefly list the stages involved in a standard search procedure using the proposed CBIR system from a user's point of view:

1. *Input query*: The user first selects a sample portion or a full hyperspectral scene to be used as an input image $\mathbf{F}$. Then, the system computes (in parallel) the feature vector associated with that portion/image using the parallel EEA algorithm developed in the previous section.
2. *Signature comparison and sorting*: The feature vector obtained in the previous stage is stored in a header file associated with $\mathbf{F}$ and compared (in parallel) with the pre-computed feature vectors of all the images in the database, using the parallel SSMA algorithm in the previous section. At this point, it is important to emphasize that, in the proposed implementation, the database is centralized at the master node. In this regard, a distributed database implementation (to be targeted in the future work) may provide advantages in terms of increased availability, quality of service, and ease of expansion. After this process, the identifiers of the M images that are most similar to the test image are extracted and ranked in the descending order of similarity.

DOI: 10.1002/cpe

3. *Display of results*: A mosaic made up of the first M images selected is assembled and then presented to the user as the search result.
4. *Query update*: If the user considers the search result to be unsatisfactory, he may select one of the displayed images (or a different portion of the original image) as a new input, and then return to the first stage. The system keeps track of successful and unsuccessful queries as identified by the user.

4. EXPERIMENTAL RESULTS

4.1. Parallel computing architectures

The parallel computing architectures used in this study comprise the Thunderhead Beowulf cluster at NASA (see http://thunderhead.gsfc.nasa.gov for details) and two HNOCs distributed among different locations at the University of Maryland. Thunderhead is a 512-processor homogeneous Beowulf cluster composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 GB of memory and 80 GB of main memory, interconnected with 2 GHz optical fiber Myrinet [33]. The two HNOCs were custom designed in order to approximate a recently proposed framework for the evaluation of heterogeneous parallel algorithms [34], which relies on the assumption that a heterogeneous algorithm cannot be executed on a heterogeneous network faster than its homogeneous version on the equivalent homogeneous network. In [34], a homogeneous computing environment is considered equivalent to the heterogeneous one if the following three principles are satisfied:

1. Both environments should have exactly the same number of processors.
2. The speed of each processor in the homogeneous environment should be equal to the average speed of processors in the heterogeneous environment.
3. The aggregate communication characteristics of the homogeneous environment should be 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 network is the same as that evidenced by its homogeneous version on the equivalent homogeneous network. This allows using the parallel performance achieved by the homogeneous version as a benchmark for assessing the parallel efficiency of the heterogeneous implementation.

The fully heterogeneous network considered in our study consists of 16 different workstations, and four communication segments. Table I shows the properties 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 s_4. The communication links between the different segments $\{s_j\}_{j=1}^{4}$ only support serial communication. For illustrative purposes, Table II also shows the capacity of all point-to-point communications in the heterogeneous network, expressed as the time in milliseconds to transfer a 1 MB message between each processor pair (p_i, p_j) in the heterogeneous system. As noted, the communication network of the fully heterogeneous network consists of four relatively fast homogeneous communication segments, interconnected by three slower communication links with

DOI: 10.1002/cpe

Table I. Specifications of heterogeneous processors.

Processor number	Architecture specification	Cycle-time (seconds per megaflop)	Memory (MB)	Cache (kB)
p_1	Free BSD—i386 Intel Pentium 4	0.0058	2048	1024
p_2, p_5, p_8	Linux—Intel Xeon	0.0102	1024	512
p_3	Linux—AMD Athlon	0.0026	7748	512
p_4, p_6, p_7, p_9	Linux—Intel Xeon	0.0072	1024	1024
p_{10}	SunOS—SUNW UltraSparc-5	0.0451	512	2048
p_{11}–p_{16}	Linux—AMD Athlon	0.0131	2048	1024

Table II. Capacity of communication links (in time in milliseconds to transfer a 1 MB message).

Processor	p_1–p_4	p_5–p_8	p_9–p_{10}	p_{11}–p_{16}
p_1–p_4	19.26	48.31	96.62	154.76
p_5–p_8	48.31	17.65	48.31	106.45
p_9–p_{10}	96.62	48.31	16.38	58.14
p_{11}–p_{16}	154.76	106.45	58.14	14.05

capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$, $c^{(3,4)} = 58.14$ in milliseconds, respectively. On the other hand, the fully homogeneous network consists of 16 identical Linux—AMD Athlon workstations with cycle-time of $w = 0.0131$ seconds per megaflop and 2 GB of main memory, interconnected via a homogeneous communication network in which the capacity of links is $c = 26.64$ ms.

4.2. Hyperspectral data

In order to illustrate the performance of our parallel CBIR system, we specifically address a case study of urban monitoring and assessment, using a collection of 154 high-resolution hyperspectral data sets (comprising a total space of more than 20 Terabytes) gathered by NASA over the World Trade Center (WTC) area in New York City during the last two weeks of September, 2001, only a few days after the terrorist attacks that collapsed the two main towers and other buildings in the WTC complex. In all cases, the spatial resolution is of 3.7 m pixel^{-1}, and the spectral resolution is of 224 narrow spectral bands between 0.4 and 2.5 μm. Figure 4 shows a false color composite of one of such images, with 614×512 pixels and 224 bands. The false color composition has been formed using the 1682, 1107, and 655 nm channels. The vegetated areas are located at the south-west of Manhattan, whereas the burned areas can be seen further north in mid-town. Smoke coming from the WTC area is also visible in the scene. The area used as input query in our experiment is shown in a red rectangle, and is centered at the region where the towers collapsed. This area contains spectral signatures of thermal hot spots corresponding to fires in the area. The fact that our search area contains such spectral signatures was expected to assist in the detection of other images containing fires across the entire database, which is a useful task in order to assist in the detection of such fires at sub-pixel levels, thus contributing to the extinction efforts conducted in the area.

DOI: 10.1002/cpe

Figure 4. AVIRIS hyperspectral image collected over the World Trade Center (left) and detail of the area used as input query (right).

Using the search area in the rightmost part of Figure 4 as input query, the proposed parallel CBIR system successfully retrieved all the image instances ($M = 7$) containing the WTC complex across the database, with no false positive detections, using the Virtual Dimensionality concept in [22,23] to automatically estimate the number of endmembers in each scene and a total of $K = 10^4$ skewers for the endmember extraction stage. Specifically, it was observed that the PPI algorithm produced the same final set of endmembers as the number of skewers was above this value (experiments with $K = 10^5$ and $K = 10^6$ were conducted). Based on the above experiments, the cutoff threshold parameter v_c was set to the mean of PPI scores obtained after $K = 10^4$ iterations. Finally, the threshold angle value used to implement the SAD-based similarity criterion of SSMA was set to $v_a = 0.1$, a reasonable limit of tolerance for this metric. These parameter values are in agreement with those used before in the literature [19].

For illustrative purposes, Figure 5 shows the seven full image flightlines in the considered AVIRIS database, which contain the searched area centered at the WTC complex. On the other hand, Figure 6 shows some of the full image flightlines in the considered database, which do not contained the searched area. Typically, each flightline comprises from 5 to 7 individual images, and a total of 24 full flightlines were considered in our experiments[§]. As shown by Figures 5 and 6, the complexity of the scenes is very high due to smoke and urban interferers in the scene, which hinder the identification of areas with hot spot thermal fires that are used as a search criterion in this experiment. In this regard, the proposed parallel CBIR system performed very accurately in this

[§] See http://aviris.jpl.nasa.gov/ql/listg01.html.

DOI: 10.1002/cpe

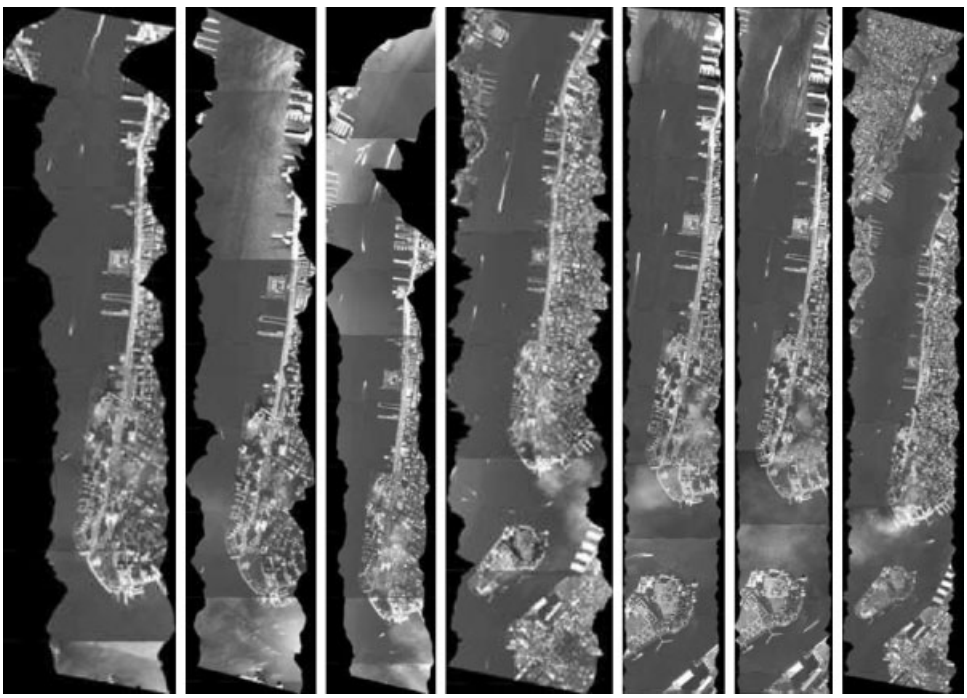


Figure 5. Full flightlines collected by the AVIRIS sensor over the World Trade Center area, which contain the search area in Figure 4. Typically, each flightline contains 5–7 hyperspectral images (each with 224 spectral bands).

task, thus serving as a relevant tool for content-based retrieval of hyperspectral images based on a complex search criterion: the presence of fires, which in many cases can only be detected at sub-pixel levels.

In addition, as will be shown in the following subsection, the signature comparison and sorting times achieved by queries in the proposed parallel CBIR algorithm were deemed suitable for (near) real-time exploitation of the system, including the appealing possibility of rapidly providing emergency response teams with information about the presence of fires and the evolution in the distribution of debris and other materials in the dusts deposited around the WTC area in this particular case study.

4.3. Parallel performance evaluation

To investigate its parallel properties, the proposed CBIR system was first tested on the two considered HNOCs. Table III shows the measured execution times for the heterogeneous EEA (referred to hereinafter as Hetero-EEA) and the heterogeneous SSMA (referred to hereinafter as Hetero-SSMA), and their respective homogeneous versions, in the task of performing a new input query using a reference hyperspectral scene not yet cataloged in the system. In the following, we will refer to

DOI: 10.1002/cpe

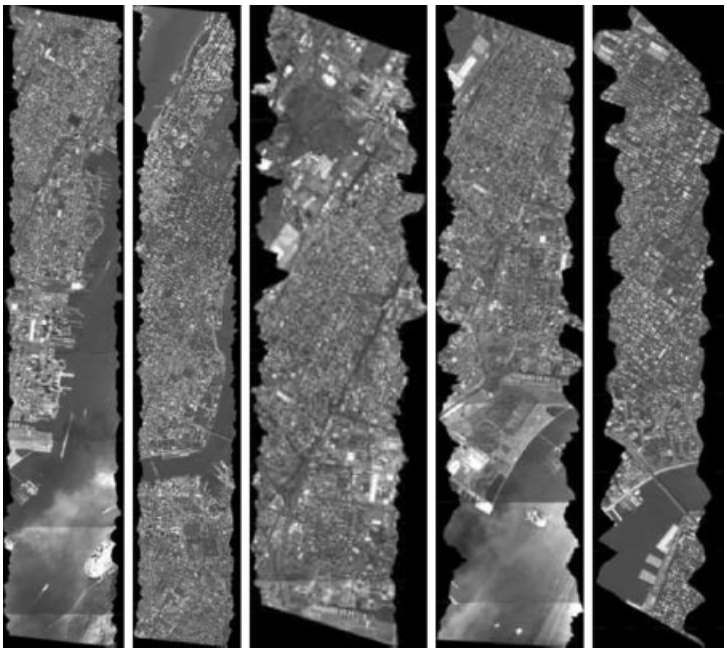


Figure 6. Full flightlines collected by the AVIRIS sensor over the World Trade Center area, which do not contain the search area in Figure 4. Typically, each flightline contains 5–7 hyperspectral images (each with 224 spectral bands).

Table III. Execution times (seconds) of the parallel versions of EEA and SSMA on the two networks.

Algorithm	Heterogeneous	Homogeneous
Hetero-EEA (implemented using WEA)	82.07	87.24
Hetero-EEA (implemented using SWEA)	84.40	89.06
Homo-EEA	667.35	81.33
Hetero-SSMA (implemented using WEA)	16.02	17.36
Hetero-SSMA (implemented using SWEA)	16.94	18.91
Homo-SSMA	158.23	15.93

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.

It is important to emphasize that the workload estimation process in the homogeneous versions was conducted by simply setting

$$\alpha_i = \left\lfloor \frac{(1/t_i)}{\sum_{i=1}^{P}(1/t_i)} \right\rfloor$$

DOI: 10.1002/cpe

for all $i \in \{1, 2, \ldots, P\}$, followed by execution of steps 4(a) and (b) of the WEA algorithm. For the heterogeneous algorithms, we used both the WEA algorithm and its simplified version (SWEA) to measure the impact of introducing (or not) the communication properties in the workload estimation process. The total time spent by this process (less than 2 s in all cases) was negligible in comparison with the total execution time of each algorithm. This is due to the high volume of computations involved in hyperspectral imaging applications and also to the regularity of such computations, which simplifies the load predictions.

As expected, the execution times reported in Table III show that the heterogeneous algorithms were able to adapt much better to the heterogeneous environment than the homogeneous versions, which only performed satisfactorily on the homogeneous network. One can see that the heterogeneous algorithms were always several times faster than their homogeneous counterparts in the heterogeneous system. On the other hand, the homogeneous algorithms only slightly outperformed their heterogeneous counterparts in the homogeneous network. Table III also indicates that the performance of the heterogeneous algorithms on the heterogeneous platform was almost the same as that evidenced by the equivalent homogeneous algorithms on the homogeneous one. This indicated that the proposed heterogeneous algorithms were close to the optimal heterogeneous modifications of the basic homogeneous ones [34]. Table III also reveals that using WEA for workload estimation in the heterogeneous algorithms resulted in slightly lower execution times than those obtained using SWEA, a fact that suggests that including the capacities of the communication links in the workload estimation can improve the overall performance of the heterogeneous algorithms.

In order to fully substantiate the above remark, Figure 7(a) plots the speedups achieved by the heterogeneous algorithms (implemented using both WEA and SWEA) over their respective homogeneous versions on the heterogeneous network. Here, the speedup was simply calculated as the execution time of the homogeneous algorithm divided by the execution time of the heterogeneous algorithm. One can see that, for both algorithms, the speedups were slightly higher when WEA was used to partition the initial workload. This revealed that including the capacity of the communication links in the initial workload estimation can be beneficial. On the other hand, Figure 7(b) plots the speedup of the homogeneous algorithms over their heterogeneous counterparts (implemented using both WEA and SWEA) in the homogeneous network. In this case, the speedup was calculated as the execution time of the heterogeneous algorithm divided by the execution time of the homogeneous algorithm. As can be seen in Figure 7(b), the homogeneous versions only slightly outperformed the heterogeneous ones in this platform. The speedup factors in Figure 7(b) were very close to one (in particular, when WEA was used), 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 homogeneous network. The reported speedups suggest the flexibility of the proposed heterogeneous algorithms in adapting to the properties of different computing environments, regardless of their inherent heterogeneity or homogeneity.

To further explore the parallel properties of the considered algorithms in more detail, an in-depth analysis of the computation and communication times achieved by the different methods is also highly desirable. For that purpose, Table IV shows the total time spent by the tested algorithms in communications and computations in the two considered networks, 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).

DOI: 10.1002/cpe

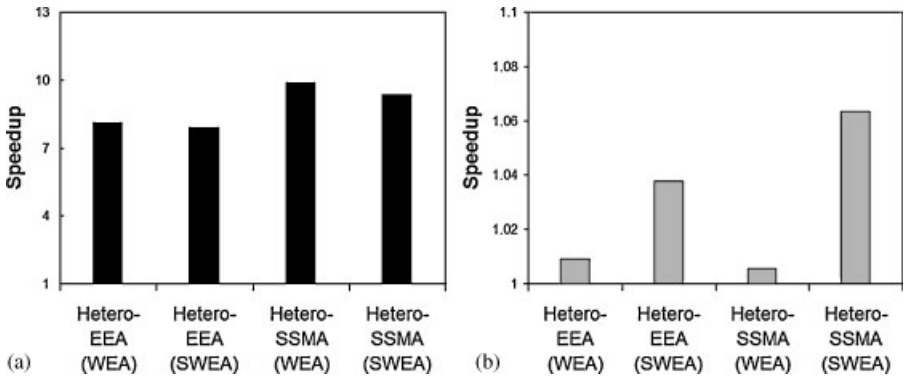


Figure 7. (a) Speedup of the heterogeneous algorithms over their homogeneous versions on the heterogeneous network and (b) speedup of the homogeneous algorithms over their heterogeneous versions on the homogeneous network.

Table IV. Communication (COM), sequential computation (SEQ), and parallel computation (PAR) times for the parallel versions of EEA and SSMA on the two networks.

	Heterogeneous			Homogeneous		
Algorithm	COM	SEQ	PAR	COM	SEQ	PAR
Hetero-EEA (implemented using WEA)	6.88	19.06	56.13	9.63	15.75	61.86
Hetero-EEA (implemented using SWEA)	7.11	19.25	58.04	10.55	16.40	62.11
Homo-EEA	14.12	19.03	634.20	5.95	16.20	59.18
Hetero-SSMA (implemented using WEA)	1.91	2.35	11.76	2.60	2.02	12.74
Hetero-SSMA (implemented using SWEA)	2.21	2.41	12.32	3.31	2.54	13.06
Homo-SSMA	1.96	2.05	154.22	2.13	1.25	12.55

The latter includes the times in which the workers remain idle. It can be seen from Table IV that SEQ scores were significant for the EEA algorithm as a result of the *endmember selection* step, which is performed in sequential fashion at the master. Table IV also reveals that the proposed SSMA implementation is almost *embarrassingly parallel* since PAR scores clearly dominated the SEQ scores. Finally, it can also be seen from Table IV that the cost of PAR computations dominated that of communications (COM) in all the considered parallel algorithms. In particular, the values of PAR scores achieved by the homogeneous algorithms executed on the heterogeneous network were very high, but this is mainly due to a less efficient workload distribution among the heterogeneous workers. This aspect was significantly improved when the WEA or SWEA algorithms were used to estimate the workload to be assigned to each heterogeneous processor, resulting in more efficient balancing of the load. Therefore, a more detailed study of load balance is highly desirable to fully substantiate the parallel properties of the considered algorithms.

To analyze the issue of load balance in more detail, Table V shows the *imbalance* scores achieved by the parallel algorithms on the two considered networks. The imbalance is defined

DOI: 10.1002/cpe

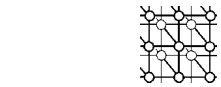


Table V. Load balancing rates for the parallel versions of EEA and SSMA on the two networks.

Algorithm	Heterogeneous		Homogeneous	
	D_{all}	D_{minus}	D_{all}	D_{minus}
Hetero-EEA (implemented using WEA)	1.03	1.02	1.05	1.03
Hetero-EEA (implemented using SWEA)	1.07	1.05	1.06	1.03
Homo-EEA	1.82	1.23	1.10	1.06
Hetero-SSMA (implemented using WEA)	1.01	1.01	1.03	1.02
Hetero-SSMA (implemented using SWEA)	1.02	1.01	1.03	1.02
Homo-SSMA	1.16	1.11	1.03	1.01

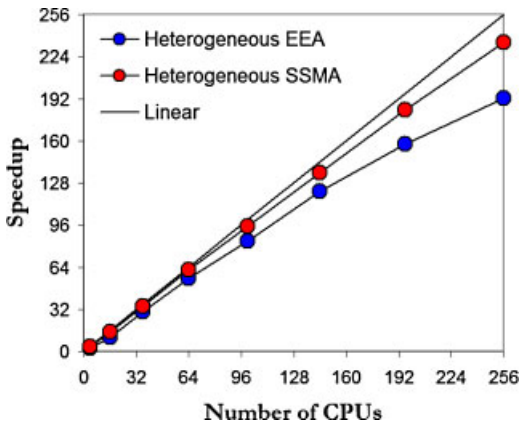


Figure 8. Scalability of the heterogeneous versions of EEA and SSMA on Thunderhead.

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 V, only the heterogeneous algorithms were able to provide the values of D_{all} close to 1 (the optimal case) in the two considered networks, with both the heterogeneous SSMA and EEA algorithms producing results close to optimal when the WEA algorithm was used to estimate the workload in the heterogeneous network. In this case, the heterogeneous versions provided almost the same results for both D_{all} and D_{minus}. While the homogeneous algorithms executed on the heterogeneous network provided the highest values of D_{all} and D_{minus} (and hence the highest imbalance), the heterogeneous algorithms executed on the homogeneous network resulted in values of D_{minus}, which were also close to optimal.

Taking into account the results presented above, and with the ultimate goal of exploring issues of scalability, we have also compared the performance of the parallel CBIR system on NASA's Thunderhead Beowulf cluster. For that purpose, Figure 8 plots the speedups achieved by multiprocessor runs of the heterogeneous EEA and SSMA algorithms on Thunderhead. In this case, we only used SWEA to perform the workload estimation due to the fact that the communication links in

DOI: 10.1002/cpe

the Thunderhead system are homogeneous. As the speedup factors achieved by the homogeneous versions were almost identical to those obtained by the heterogeneous algorithms, the plots for the homogeneous algorithms are omitted for simplicity. It can be seen from Figure 8 that the heterogeneous SSMA scaled slightly better than the heterogeneous EEA. This has to do with the higher number of sequential computations involved in the parallel EEA algorithm. However, the combined performance was satisfactory. For instance, using 256 processors on Thunderhead, the heterogeneous EEA was able to extract feature vectors for the considered AVIRIS scene (140 MB in size) in only 6 s, whereas the SSMA efficiently searched the most similar scenes across the full database of 154 images (with pre-computed signatures) in only 4 s, resulting in a total processing of approximately 10 s to catalog and fully describe a new entry in the database. The above result represents a significant improvement over the implementation of the same CBIR process on a single Thunderhead processor, which took over 1 h of computation for the same operation.

5. CONCLUSIONS AND FUTURE LINES

This paper described an innovative parallel CBIR system for hyperspectral image retrieval from heterogeneous platforms. As a case study of the specific issues involved in the development of data mining systems in remote sensing applications, we provided a detailed discussion on the effects that platform heterogeneity has on degrading the parallel performance of an information extraction algorithm which first extracts the spectral endmembers and then uses their relative abundance fractions as a feature vector to perform a query based on the sub-pixel image content. The evaluation strategy conducted in this work was based on experimentally assessing the proposed heterogeneous implementation by comparing its efficiency on a fully HNOCs with the efficiency achieved by the equivalent homogeneous version on an equally powerful homogeneous network of computers. Performance results have also been provided for a Beowulf cluster, thus covering the two most widely used types of parallel platforms in parallel CBIR applications.

Our experimental results indicate that the proposed parallel CBIR system can accurately extract hyperspectral image instances from a complex image database with sub-pixel precision and quickly enough for practical use. As a result, we believe that the proposed system can adequately exploit the source of the computational power currently offered by Beowulf clusters and heterogeneous networks of workstations, thus making the proposed tool accessible and applicable to obtaining results quickly enough and with high reliability in many on-going and planned Earth-observing missions. As a future extension of the system, we plan to develop a distributed database implementation that may provide competitive advantages in terms of increased availability, quality of service, and ease of expansion.

ACKNOWLEDGEMENTS

This work has been partially supported by the European Community's Marie Curie Research Training Networks Programme under reference MRTN-CT-2006-035927, Hyperspectral Imaging Network (HYPER-I-NET). Funding from the Spanish Ministry of Science and Innovation (HYPERCOMP/EODIX project, reference AYA2008-05965-C04-02), and from Junta de Extremadura (PRI09A110 project) is also gratefully acknowledged. The authors gratefully thank Dr Robert O. Green and the AVIRIS team at NASA/JPL for providing the

DOI: 10.1002/cpe

hyperspectral data sets used in our experiments. Last but not the least, the authors gratefully thank the three anonymous reviewers for their fruitful comments and suggestions, which greatly helped us to improve the techniques discussed in this work, as well as the technical quality and presentation of our manuscript.

REFERENCES

1. Smeulders AWM, Worring M, Santini S, Gupta A, Jain R. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2000; **22**:1349–1380.
2. Vogel J, Schiele B. Performance evaluation and optimization for content-based image retrieval. *Pattern Recognition* 2006; **39**:897–909.
3. Chang C-I. *Hyperspectral Data Exploitation*: *Theory and Applications*. Wiley: New York, 2007.
4. Yoo H-W, Park H-S, Jang D-S. Expert system for color image retrieval. *Expert Systems with Applications* 2005; **28**:347–357.
5. Green RO, Eastwood ML, Sarture CM, Chrien TG, Aronsson M, Chippendale BJ, Faust JA, Pavri BE, Chovit CJ, Solis M, Olah MR, Williams O. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sensing of Environment* 1999; **65**:227–248.
6. Wang P, Liu KY, Cwik T, Green RO. MODTRAN on supercomputers and parallel computers. *Parallel Computing* 2002; **28**:53–64.
7. Kalluri S, Zhang Z, JaJa J, Liang S, Townshend J. Characterizing land surface anisotropy from AVHRR data at a global scale using high performance computing. *International Journal of Remote Sensing* 2001; **22**:2171–2191.
8. Dhodhi MK, Saghri JA, Ahmad I, Ul-Mustafa R. D-ISODATA: A distributed algorithm for unsupervised classification of remotely sensed data on network of workstations. *Journal of Parallel and Distributed Computing* 1999; **59**:280–301.
9. Achalakul T, Taylor S. A distributed spectral-screening PCT algorithm. *Journal of Parallel and Distributed Computing* 2003; **63**:373–384.
10. Plaza A, Valencia D, Plaza J, Martinez P. Commodity cluster-based parallel processing of hyperspectral imagery. *Journal of Parallel and Distributed Computing* 2006; **66**:345–358.
11. Lastovetsky A. *Parallel Computing on Heterogeneous Networks*. Wiley: Hoboken, 2003.
12. Banino C, Beaumont O, Carter L, Ferrante J, Legrand A, Robert Y. Scheduling strategies for master–slave tasking on heterogeneous processor platforms. *IEEE Transactions on Parallel and Distributed Systems* 2004; **15**:319–330.
13. Berten V, Goosens J, Jeannot E. On the distribution of sequential jobs in random brokering for heterogeneous computational grids. *IEEE Transactions on Parallel and Distributed Systems* 2006; **17**:113–124.
14. Hawick K, James H, Silis A, Grove D, Pattern C, Mathew J, Coddington P, Kerry K, Hercus J, Vaughan F. DISCWorld: An environment for service-based meta-computing. *Future Generation Computer Systems* 1999; **15**:623–635.
15. Foster I, Kesselman C. *The Grid*: *Blueprint for a New Computing Infrastructure*. Morgan Kaufman: San Francisco, 1999.
16. Bosque JL, Robles O, Pastor L, Rodriguez A. Parallel CBIR implementations with load balancing algorithms. *International Journal of Parallel and Distributed Systems* 2006; **66**:1062–1075.
17. Robles O, Bosque JL, Pastor L, Rodriguez A. *CBIR on Grids* (*Lecture Notes in Computer Science*, vol. 66). Springer: New York, 2006; 1412–1421.
18. Plaza A, Valencia D, Plaza J, Chang C-I. Parallel implementation of end member extraction algorithms from hyperspectral data. *IEEE Geoscience and Remote Sensing Letters* 2006; **3**:63–67.
19. Plaza A, Martinez P, Perez R, Plaza J. A quantitative and comparative analysis of end member extraction algorithms from hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing* 2004; **42**:650–663.
20. Heinz D, Chang C-I. Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing* 2001; **39**:529–545.
21. Boardman J, Kruse FA, Green RO. Mapping target signatures via partial unmixing of AVIRIS data. *Summaries of the NASA/JPL Airborne Earth Science Workshop*, Pasadena, CA, vol. 8, 1995; 110–123.
22. Chang C-I. *Hyperspectral Imaging*: *Techniques for Spectral Detection and Classification*. Kluwer Academic Publisher: New York, 2003.
23. Chang C-I, Du Q. Estimation of number of spectrally distinct signal sources in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing* 2004; **42**:608–619.
24. Plaza A, Valencia D, Plaza J. An experimental comparison of parallel algorithms for hyperspectral analysis using homogeneous and heterogeneous networks of workstations. *Parallel Computing* 2008; **34**:92–114.
25. Elsasser R, Monien B, Preis R. Diffusion schemes for load balancing on heterogeneous networks. *Theory of Computing Systems* 2002; **35**:305–320.
26. Legrand A, Renard H, Robert Y, Vivien F. Mapping and load-balancing iterative computations. *IEEE Transactions on Parallel and Distributed Systems* 2004; **15**:546–558.

DOI: 10.1002/cpe

27. Elsasser R, Monien B, Schamberger S. Distributing unit size workload packages in heterogeneous networks. *Journal of Graph Algorithms and Applications* 2006; **10**:51–68.
28. Richards JA, Jia X. *Remote Sensing Digital Image Analysis*: *An Introduction* (4th edn). Springer: Berlin, Germany, 2006.
29. Reed I, Yu X. Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution. *IEEE Transactions on Acoustics* 1990; **10**:1760–1770.
30. Plaza A, Valencia D, Blazquez S, Plaza J. Parallel detection of targets in hyperspectral images using heterogeneous networks of workstations. *Proceedings of the 15th Euromicro International Conference on Parallel*, *Distributed and Network-Based Processing*, Naples, Italy, 2007; 333–340.
31. Plaza A. Parallel processing of remotely sensed hyperspectral imagery: Full-pixel versus mixed-pixel classification. *Concurrency and Computation*: *Practice and Experience* 2008; **20**:1539–1572.
32. Gropp W, Huss-Lederman S, Lumsdaine A, Lusk E. MPI—The complete reference. *The MPI Extensions*, vol. 2. MIT Press: Cambridge, MA, 1999.
33. Dorband J, Palencia J, Ranawake U. Commodity computing clusters at goddard space flight center. *Journal of Space Communication* 2003; **1**:1–13.
34. Lastovetsky A, Reddy R. On performance analysis of heterogeneous parallel algorithms. *Parallel Computing* 2004; **30**:1195–1216.

DOI: 10.1002/cpe