Distributed Computing for Efficient Hyperspectral Imaging Using Fully Heterogeneous Networks of Workstations

Antonio Plaza, Javier Plaza and David Valencia
Department of Computer Science, University of Extremadura
Avda. de la Universidad s/n, E-10071 Cáceres, SPAIN
aplaza@unex.es

Abstract

Hyperspectral imaging is a new technique which has become increasingly important in many remote sensing applications, including automatic target recognition for military and defense/security deployment, risk/hazard prevention and response including wild land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination, etc. Hyperspectral imaging applications generate massive volumes of data and require timely responses for swift decisions which depend upon high computing performance of algorithm analysis. Although most currently available parallel processing strategies for hyperspectral image analysis assume homogeneity in the computing platform, heterogeneous networks of workstations represent a very promising cost-effective solution expected to play a major role in the design of high-performance computing platforms for many on-going and planned remote sensing missions. This paper explores innovative techniques for mapping hyperspectral analysis algorithms onto heterogeneous networks of workstations available at NASA's Goddard Space Flight Center and University of Maryland. Experimental results reveal that heterogeneous networks of workstations represent a source of computational power that is both accessible and applicable in hyperspectral imaging studies.

1. Introduction

The incorporation of hyperspectral sensors to airborne and satellite platforms is producing a nearly continual stream of high-dimensional data, and this explosion in the amount of collected information has rapidly introduced new processing challenges. The concept of hyperspectral imaging [1] was first introduced when NASA's Jet Propulsion Laboratory Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) [2] was developed. This imager covers the wavelength region from 0.4 to 2.5 μm using 224 spectral channels, at a nominal spectral resolution of 10 nm (see Fig. 1). As a result, each pixel is given by a vector of values and called "pixel vector." The automation of techniques for transforming collected data into scientific understanding is critical for space-based Earth science and planetary exploration. For instance, NASA is continuously gathering imagery data with Earth-observing sensors [3], with more than 850 GB of hyperspectral data collected and sent to Earth on a daily basis, and this expected high data volume would demand fast and efficient means for storage, transmission, and analysis.

To address the computational need introduced by hyperspectral imaging applications, several efforts have been directed towards the incorporation of high-performance computing models in remote sensing missions [4-7], especially with the advent of relatively cheap Beowulf clusters [3]. The new processing power offered by such commodity systems has been employed in information extraction and mining from very large data archives [8, 9]. However, the homogeneous nature of systems for image information processing employed by NASA and other institutions during the last decade is soon to be replaced by large-scale, heterogeneous computing resources. Heterogeneous networks of workstations [10] can realize a very high level of aggregate performance [3, 11], and it is expected that these systems will soon represent a tool of choice for the scientific community devoted to high-dimensional image analysis in remote sensing and other fields. Due to the homogeneous nature of currently available techniques and systems for parallel and distributed computing in hyperspectral imaging studies, significant opportunities to exploit heterogeneous computing practices are available in this emerging new area.

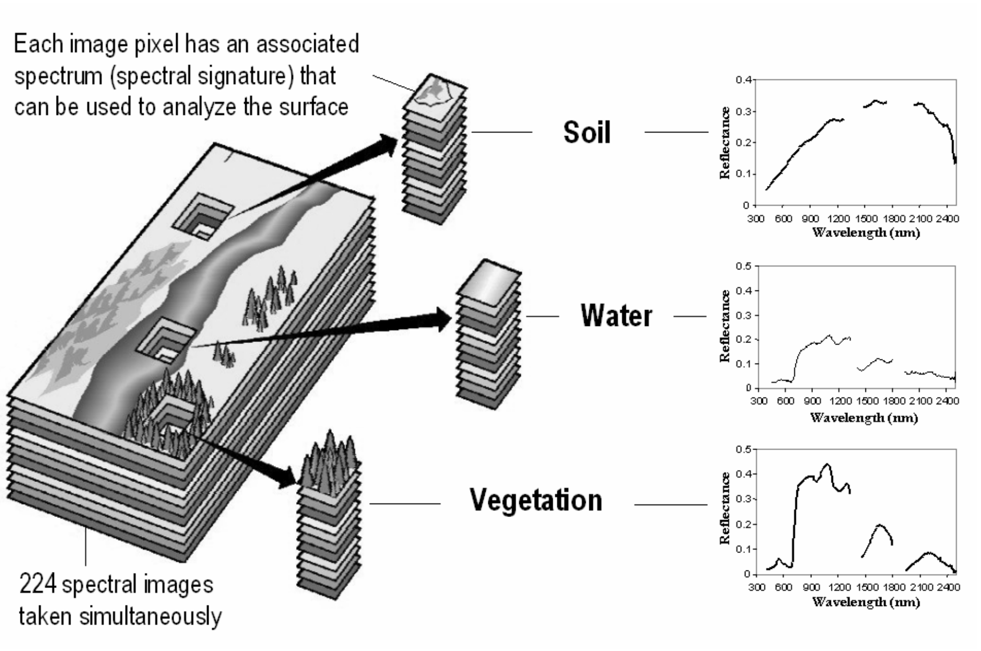
Each image pixel has an associated spectrum (spectral signature) that can be used to analyze the surface
Soil
Water
Vegetation
224 spectral images taken simultaneously
Reflectance
Wavelength (nm)

Figure 1. Concept of hyperspectral imaging using NASA Jet Propulsion Laboratory's AVIRIS sensor.

In this paper, we explore techniques for mapping hyperspectral image analysis algorithms onto heterogeneous networks of computers. The paper is structured as follows. Section 2 briefly describes a hyperspectral data processing chain that will serve as our case study throughout the paper. Section 3 develops a parallel version of the considered approach, specifically designed to be run on distributed, heterogeneous platforms. In Section 4, we assess the parallel performance of the considered parallel methodology by drawing comparisons between its efficiency on a heterogeneous network of workstations with the efficiency evidenced by its homogeneous version on a homogeneous network with the same aggregate performance as the heterogeneous one. Performance data on Thunderhead, a (homogeneous) massively parallel Beowulf cluster at NASA's Goddard Space Flight Center are also given for comparison. Finally, Section 5 highlights the main conclusions of this research.

2. Hyperspectral data processing chain

This section describes a commonly accepted hyperspectral data processing chain that will be used as a case study for the development of parallel algorithms. It consists of the following stages [2]. Firstly, the dimensionality of the input data is reduced prior to data processing. The principal component transform (PCT) is often used to summarize and decorrelate the images by reducing redundancy and packing the residual information into a small set of images, termed principal components. PCT is a highly compute-intensive algorithm amenable to parallel implementation [12, 13]. Secondly, pure spectral signatures (often called endmembers in hyperspectral analysis) are extracted from the dimensionally reduced data set. The goal of using endmembers is 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. For instance, it is very likely that the pixel vector labeled as "vegetation" in Fig. 1 would actually comprise a mixture of vegetation and soil, or different types of soil and vegetation canopies. To deal with this problem, linear spectral unmixing has been used to decompose the measured spectrum of a mixed pixel into a linear combination of endmembers weighted by a set of abundance fractions that indicate the proportion of each endmember present in the mixed pixel [14]. One of the most successful algorithms for endmember extraction in the literature has been the N-FINDR method [15]. After a PCT-based dimensional reduction, the method selects a random set of pixel vectors from the input data and calculates their corresponding volume. In order to refine the initial volume estimate, a trial volume is calculated for every pixel vector in each endmember position by replacing that endmember and recalculating the volume. If the replacement results in a volume increase, the pixel vector replaces the endmember. This procedure is repeated until there are no replacements of endmembers left. Both the identification of image endmembers and the subsequent unmixing process are computationally demanding problems. However, very few research efforts devoted to the design of parallel implementations exist in the open literature. This paper takes a necessary first step toward the comparison of strategies for parallel hyperspectral image analysis.

3. Parallel algorithms

This section describes the parallel algorithms that will be compared in this study. Before introducing the algorithm descriptions, which are based on the techniques introduced in the previous section, we must first discuss strategies for data partitioning in the considered application. In the considered application, two types of partitioning can be exploited: spectral-domain partitioning and spatial-domain partitioning. Spectral-domain partitioning subdivides the volume into small cells or sub-volumes made up of contiguous spectral bands, and assigns one or more sub-volumes to each processor. Quite opposite, spatial-domain partitioning keeps the spectral identity of each pixel vector and assigns groups of spatially correlated full pixel vectors to each processor. With the former model, each pixel vector may be split amongst several processors and the communication cost for the computations based on spectral signatures would be increased. This is due to the fact that the hyperspectral data processing chain described in the previous section utilizes the information provided by each pixel vector as a whole. This has a significant impact on the design of data partitioning strategies for parallelization. In order to exploit parallelism as much as possible, a spatial-domain partitioning approach was adopted in our framework, i.e., the data is always partitioned in a way that the same pixel vector is never split among different processors. As a result, all the considered parallel algorithms are designed under the assumption that each pixel vector is uniquely represented by its associated spectral signature. Next, we provide a pseudo-code description of the three parallel algorithms considering in this study, which fall into the categories of dimensionality reduction, endmember extraction and linear spectral unmixing.

3.1. Parallel dimensionality reduction

Inputs: N-D data cube $\mathbf{F}$, Number of endmembers, E.
Output: E-D data cube $\mathbf{G}$.

1. Let p_i denote a processor in the heterogeneous network, and let w_i denote its relative cycle-time. Similarly, let V be the total volume of data in $\mathbf{F}$. Processor p_i will be assigned a certain share $\alpha_i \cdot V$ of the input volume, where $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$. In order to obtain the value of α_i for processor p_i, calculate $\alpha_i = (1/w_i) \cdot \left(1 / \sum_{j=1}^{P} (1/w_j)\right)$ and use the calculated values of α_i to generate a set of P spatial-domain heterogeneous partitions of $\mathbf{F}$.
2. Calculate the N-D mean vector $\bar{\mathbf{f}}$ concurrently, where each component is the average of the pixel values of each spectral band of the input data. This vector is formed at the master once all the processors have finish their parts.
3. Broadcast vector $\bar{\mathbf{f}}$ to all workers, so that each worker computes the covariance component using its local partition and forms a covariance sum, which is sent to the master.
4. Calculate the covariance matrix sequentially at the master as the average of all the matrices calculated in step 3.
5. Obtain a transformation matrix $\mathbf{T}$ by calculating and sorting the eigenvectors of the covariance matrix according to their 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.
6. Transform each N-D pixel vector in the original image by $\mathbf{g}(x,y) = \mathbf{T} \cdot [\mathbf{f}(x,y) - \bar{\mathbf{f}}]$. This step is done in parallel, where all workers transform their respective data partitions. The results are sent to the master, which retains the first E components of the resulting data cube $\mathbf{G}$.

3.2. Parallel endmember extraction

Input: E-D cube $\mathbf{G}$.

Output: set of E final endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$.

1. The master selects a random set of E initial pixel vectors $\{\mathbf{e}_e^{(0)}\}_{e=1}^{E}$ randomly, and then finds $V(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \cdots, \mathbf{e}_E^{(0)})$, i.e., the volume of the simplex defined by $\{\mathbf{e}_e^{(0)}\}_{e=1}^{E}$, as follows:

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

2. The workers calculate the volume of E simplexes, $V(\mathbf{g}(x,y),\mathbf{e}_2^{(0)},\cdots,\mathbf{e}_E^{(0)}),\ldots,V(\mathbf{e}_1^{(0)},\mathbf{e}_2^{(0)},\cdots,\mathbf{g}(x,y))$ in parallel, each of which is formed by replacing one endmember $\mathbf{e}_e^{(0)}$ with the sample vector $\mathbf{g}(x,y)$. Each worker performs replacements using pixels in its local partition, obtained using step 1 of the algorithm in section 3.1.

If none of these E recalculated volumes is greater than $V(\mathbf{e}_1^{(0)},\mathbf{e}_2^{(0)},\cdots,\mathbf{e}_E^{(0)})$, then no endmember in $\{\mathbf{e}_e^{(0)}\}_{e=1}^{E}$ 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 sequentially at the master 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 2 until no replacements occur.

3.3. Parallel spectral unmixing

Input: N-D data cube $\mathbf{F}$, Set of endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$.

Output: Set of fractional abundances $\{a_e(x,y)\}_{e=1}^{E}$ for each pixel vector $\mathbf{f}(x,y)$.

1. Divide the original data cube $\mathbf{F}$ into P heterogeneous partitions using step 1 of the parallel dimensionality reduction algorithm in section 3.1, where P is the number of workers.
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, obtain a set of abundance fractions specified by $a_1(x,y), a_2(x,y),\ldots,a_E(x,y)$ using $\{\mathbf{e}_e\}_{e=1}^{E}$, so that $\mathbf{f}(x,y)=\mathbf{e}_1\cdot a_1(x,y)+\mathbf{e}_2\cdot a_2(x,y)+\cdots+\mathbf{e}_E\cdot a_E(x,y)$ [2, 14].
4. The master collects all the individual sets of fractional abundances $\{a_e^{(i)}(x,y)\}_{e=1}^{E}$ calculated for the pixels at every individual partition i, with $i=1,\cdots,P$, and forms a final set of fractional abundances designated by $\{a_e(x,y)\}_{e=1}^{E}=\bigcup_{i=1}^{P}\{a_e^{(i)}(x,y)\}_{e=1}^{E}$.

As a final note, we emphasize that the proposed parallel hyperspectral analysis framework consists of a sequence of three steps, i.e., dimensionality reduction, endmember extraction and spectral unmixing, each of which is implemented in parallel. Performance data for the three considered parallel algorithms are given in the following section.

4. Experimental results

This section provides an assessment of the effectiveness of the parallel algorithms described in section 3. The section is organized as follows. First, we describe a framework for assessment of heterogeneous algorithms introduced by Lastovetsky and Reddy [16], and provide an overview of the heterogeneous and homogeneous networks used in this work for evaluation purposes. Second, we briefly describe the hyperspectral data set used in experiments. Performance data are given in the last sub-section.

4.1. Network description

Following a recent study [16], we assess the proposed heterogeneous algorithms using the basic postulate that they cannot be executed on a heterogeneous network faster than its homogeneous prototype on the equivalent homogeneous network. Let us assume that a heterogeneous network consists of $\{p_i\}_{i=1}^{P}$ heterogeneous workstations with different cycle-times w_i, which span m communication segments $\{s_j\}_{j=1}^{m}$, where $c^{(j)}$ denotes the communication speed of segment s_{j}. Similarly, let $p^{(j)}$ be the number of processors that belong to s_{j}, and let $w_t^{(j)}$ be the speed of the t-th processor connected to s_{j}, where $t=1,\cdots,p^{(j)}$. Finally, let $c^{(j,k)}$ be the speed of the communication link between segments s_{j} and s_{k}, with $j,k=1,\cdots,m$. According to [16], the above network can be considered equivalent to a homogeneous network made up of $\{q_i\}_{i=1}^{P}$ processors with constant cycle-time w and interconnected through a homogeneous network with communication speed c if and only if the following expressions are satisfied:

$$c=\frac{\sum_{j=1}^{m}c^{(j)}\cdot\left[p^{(j)}\left(p^{(j)}-1\right)/2\right]+\sum_{j=1}^{m}\sum_{k=j+1}^{m}p^{(j)}\cdot p^{(k)}\cdot c^{(j,k)}}{p(p-1)/2}$$

$$w=\frac{\sum_{j=1}^{m}\sum_{t=1}^{p^{(j)}}w_t^{(j)}}{p},$$

where the first expression states that the average speed of point-to-point communications between processors $\{p_i\}_{i=1}^{P}$ in the heterogeneous network should be equal to the speed of point-to-point communications between processors $\{q_i\}_{i=1}^{P}$ in the homogeneous network, with both networks having the same number of processors. On the other hand, the second expression simply states that the aggregate performance of processors $\{p_i\}_{i=1}^{P}$ should be equal to the aggregate performance of processors $\{q_i\}_{i=1}^{P}$. We have configured two networks of workstations (one homogeneous and the other one heterogeneous) that satisfy the above constraints to serve as sample networks for testing the performance of parallel heterogeneous hyperspectral imaging algorithms. The first network is composed of 16 identical Linux workstations with processor cycle-time $w = 0.0131$ seconds per megaflop, interconnected via a homogeneous network with capacity $c = 26.64$ milliseconds. The second network consists of 16 different SGI, Solaris and Linux workstations, and four communication segments. Table 1 shows the cycle-times of the heterogeneous processors in seconds per megaflop. It can be seen from the table that processors $\{p_i\}_{i=1}^{4}$ are attached to segment s_1, processors $\{p_i\}_{i=5}^{8}$ are attached to s_2, processors $\{p_i\}_{i=9}^{10}$ are attached to s_3, and processors $\{p_i\}_{i=11}^{16}$ are attached to s_4. The communication links between the different segments only support serial communication. For illustrative purposes, Table 2 shows the capacity of all point-to-point communications in milliseconds to transfer a one-megabit message between each processor pair (p_i, p_j) in the heterogeneous system. As it can be deducted from Table 2, the communication network of the heterogeneous platform consists of four relatively fast homogeneous communication segments interconnected by three slower communication links with capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$ and $c^{(3,4)} = 58.14$ milliseconds, respectively. Although this is a simple architecture, it is also a quite typical and realistic one as well.

4.2. Hyperspectral data description

The parallel algorithm in section 3 was applied to a hyperspectral scene collected by the AVIRIS hyperspectral image over the Jasper Ridge Biological Preserve (JRBP) in California (see Fig. 2). This scene was selected for experiments due to the availability of ground-truth image endmembers for this scene. The dataset, acquired on April 1998, consists of 512x614 pixels and 224 spectral bands, with a nominal ground resolution of 20 m, spectral resolution of 10 nm, and 16-bit radiometric resolution. The total size of the image data set is 137 MB. In a previous study of surface materials over JRBP, image endmembers and their corresponding abundance fractions were derived from the scene above based on extensive ground studies [17]. A library of spectral signatures associated to the main constituent materials at JRBP was used in experiments (see Fig. 3). These signatures, corresponding to soil, forest, grass, chaparral vegetation and water, along with their abundance fractions available from previous studies [17], will be used as reference in order to validate the accuracy of the proposed hyperspectral data processing chain.

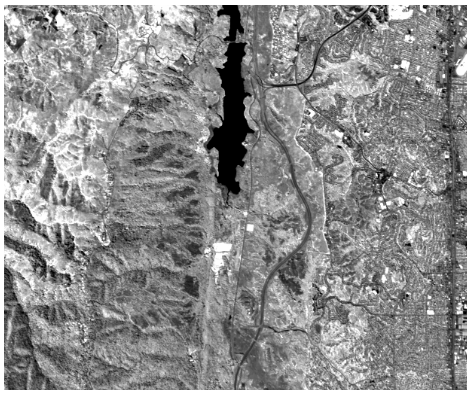
Figure 2. Spectral band at 903 nm of an AVIRIS image collected over Jasper Ridge, California.

4.3. Performance data

Before analyzing the parallel properties of the considered algorithms, implemented using MPI, we discuss their accuracy in the context of hyperspectral imaging applications. Table 3 tabulates the spectral similarity scores obtained after comparing the five reference spectra in Fig. 3 with the corresponding endmembers extracted by the proposed parallel endmember extraction algorithm in section 3.2. The closer these values are to zero, the better the results. This table also reports (in bold typeface) the root mean square error (RMSE) between the abundances in percentage estimated by using the parallel spectral unmixing algorithm in section 3.3 in combination with the endmembers provided by the parallel endmember extraction method. As shown in the table, the error scores were low for the five materials considered, with an average error in abundance estimation of less than 5% and a set of final endmembers which are very similar, spectrally, to the reference signatures in Fig. 3.

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}
0.0072	0.0102	0.0206	0.0072	0.0102	0.0058	0.0072	0.0102	0.0072	0.0451	0.0131	0.0131	0.0131	0.0131	0.0131	0.0131

Table 1. Processor cycle-times (in seconds per megaflop) for the heterogeneous cluster.

	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	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	

Table 2. Capacity of links (measured in terms of the time in milliseconds to transfer a one-megabit message) for the heterogeneous cluster.

Parallel algorithm	SAM-Endmember extraction	RMSE-Spectral unmixing
Soil (s_1)	0.007	5%
Forest (s_2)	0.005	3%
Dry grass (s_3)	0.009	4%
Chaparral (s_4)	0.007	6%
Lake water (s_5)	0.006	2%

Table 3. SAM-based similarity scores for the parallel endmember extraction algorithm and RMSE-based abundance estimation errors (in percentage) for the parallel spectral unmixing algorithm.

It should be noted that the full data analysis process took several hours of computation in a last-generation desktop computer with AMD processor at 2 GHz and 1 GB of RAM memory. To investigate the parallel properties of the proposed parallel heterogeneous algorithms using the AVIRIS scene in Fig. 2, their performance was tested by timing the programs using the heterogeneous network of workstations and its equivalent homogeneous network, where Table 4 shows the measured execution times. As expected, the times reported show that heterogeneous algorithms were able to adapt better to the heterogeneous environment than their homogeneous versions.

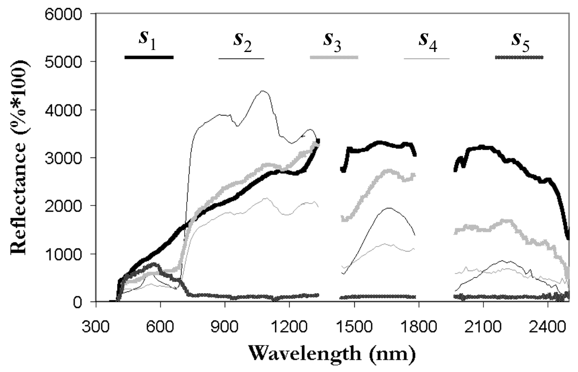

Figure 3. Reference spectral signatures: soil (s_1), forest (s_2), grass (s_1), chaparral vegetation (s_4) and water (s_5).

The homogeneous versions of the parallel algorithms were simply obtained by replacing the heterogeneous data partitioning operation in the three considered versions by a much more simple operation in which spatial-domain partitions are obtained by using to constant values of $\alpha_i = 1/P$ for all $i \in \{1, \cdots, P\}$, where w is the cycle-time for all processors in the homogeneous network.

	Heterogeneous network		Homogeneous network	
Algorithm	Execution time	Speedup	Execution time	Speedup
Dimensionality reduction (homogeneous)	614	5.29	116	1.06
Dimensionality reduction (heterogeneous)	116		124	
Endmember extraction (homogeneous)	209	4.86	41	1.07
Endmember extraction (heterogeneous)	43		44	
Spectral unmixing (homogeneous)	1123	9.28	114	1.07
Spectral unmixing (heterogeneous)	121		122	

Table 4. Execution times and speedups achieved by the parallel algorithms executed on the heterogeneous cluster over and their homogeneous versions executed on the homogeneous cluster.

For the sake of comparison, Table 4 also shows the speedup of the heterogeneous algorithms over their respective homogeneous versions on the same heterogeneous platform. 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 the heterogeneous dimensionality reduction and endmember extraction algorithms were about five times faster than their respective homogeneous versions, while the heterogeneous spectral unmixing algorithm was more than nine times faster than its homogeneous version in the heterogeneous cluster. Similarly, Table 4 shows a comparison of the execution times of the heterogeneous algorithms and their homogeneous versions on the homogeneous platform, along with the speedup of the homogeneous algorithms over the heterogeneous ones on the same homogeneous platform. As can be seen in Table 4, the homogeneous versions only slightly outperformed the heterogeneous algorithms in the homogeneous network. The speedup factors reported in the table were low and very similar for all tested methods, which reveals that the performance of heterogeneous algorithms was almost the same as that evidenced by homogeneous algorithms when they were run in the same homogeneous network. This demonstrates the flexibility of the proposed heterogeneous algorithms, which were able to adapt efficiently to both the homogeneous and heterogeneous network.

Interestingly, after comparing the execution times of heterogeneous algorithms performed on the heterogeneous network with those achieved by their homogeneous versions on the homogeneous network (see Table 4), we noticed that the heterogeneous algorithms achieved essentially the same speed as their homogeneous versions, but each on its network. This also indicated that the proposed heterogeneous algorithms were very close to the optimal heterogeneous modifications of the basic homogeneous ones. To fully validate the above remark, we have also compared the performance of the proposed heterogeneous algorithms (and their homogeneous counterparts) on Thunderhead, a (homogeneous) Beowulf cluster at NASA's Goddard Space Flight Center to explore code scalability issues [18].

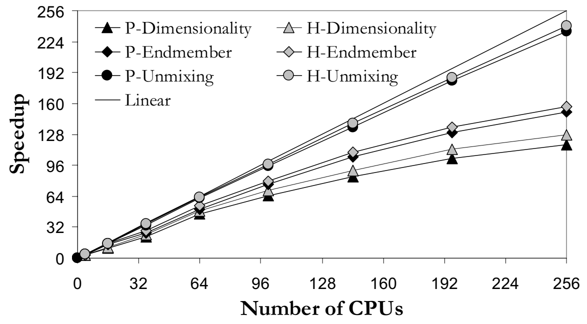

Figure 4. Scalability of the parallel algorithms on NASA's Thunderhead system.

Thunderhead is currently composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 GB of memory and 80 GB of main memory. The total peak performance of the system is 2457.6 GFlops. Fig. 4 plots the speedups achieved by multi-processor runs of the considered algorithms over their corresponding single-processor runs on the Thunderhead system. As Fig. 4 shows, the scalability of all heterogeneous algorithms (denoted by "P-") was almost the same as that evidenced by their homogeneous versions (denoted by "H-"), with the parallel spectral unmixing algorithm showing almost perfect scalability. This is not surprising, given their very straightforward parallelization strategy as compared to the adopted framework to implement the dimensionality reduction and endmember extraction algorithms in parallel, which introduce additional data dependencies. The total processing time of the full heterogeneous data processing chain implemented in parallel and applied to the AVIRIS data set in Fig. 2 was below 10 seconds when 256 processors were used, and below 50 seconds when only 36 processors were used. This indicates that the heterogeneous algorithms were able to obtain highly accurate hyperspectral analysis results (in light of Table 3), but also quickly enough for practical use. To conclude this paper, we must emphasize that, despite the computational power offered by Thunderhead, the current trend in remote sensing studies is to exploit highly heterogeneous, massively parallel computing platforms able to operate in large-scale distributed environments. As evidenced by experimental results in this work, standard homogeneous parallel algorithms often cannot efficiently adapt to such systems, while carefully designed heterogeneous algorithms offer a relatively simple, platform-independent and scalable solution. We feel that the applicability of the proposed approach extends beyond remote sensing applications. This is particularly true for the domains of signal processing and linear algebra applications, which include similar patterns of communication and calculation.

5. Conclusions

Distributed computing on heterogeneous networks is a paradigm which is soon to be adopted to satisfy the extreme computational requirements of most Earth-observing and planetary applications. The incorporation of such heterogeneous systems requires carefully design and implementation of new parallel techniques and algorithms for efficient information extraction from imagery data, in particular, taking into account that latest-generation sensor systems are currently producing a nearly continual stream of very high-dimensional data. This paper has explored the impact of platform heterogeneity on the design of parallel algorithms for hyperspectral analysis, designed to be run on fully heterogeneous networks of workstations. The strategy adopted in this work was to experimentally assess heterogeneous algorithms by comparing their efficiency on a fully heterogeneous network of workstations with the efficiency achieved by their homogeneous versions on an equally powerful homogeneous network. Our experimental results revealed important algorithmic aspects that may be of great importance for designing and adapting existing high-performance hyperspectral imaging applications (developed in the context of homogeneous computing platforms) to highly heterogeneous environments, which are currently the tool of choice in many remote sensing and Earth exploration missions.

References

[1] R. O. Green et al., "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer," *Remote Sensing of Environment*, vol. 65 pp. 227–248, 1998.

[2] C.-I Chang, *Hyperspectral imaging: Techniques for spectral detection and classification*, Kluwer: NY, 2003.

[3] J. Dorband, J. Palencia and U. Ranawake, "Commodity computing clusters at Goddard Space Flight Center," *Journal of Space Communication*, vol. 1, 2003.

[4] G. Aloisio, M. Cafaro, "Dynamic earth observation system," *Parallel Comp.*, vol. 29, pp. 1357–1362, 2003.

[5] L. Chen, I. Fujishiro and K. Nakajima, "Optimizing parallel performance of unstructured volume rendering for the Earth Simulator," *Parallel Comp*, vol. 29, pp. 355–371, 2003.

[6] P. Wang, K. Y. Liu, T. Cwik and R.O. Green, "MODTRAN on supercomputers and parallel computers," *Parallel Comp.*, vol. 28, pp. 53–64, 2002.

[7] K. A. Hawick, P. D. Coddington and H. A. James, "Distributed frameworks and parallel algorithms for processing large-scale geographic data," *Parallel Comp.*, vol. 29, pp. 1297–1333, 2003.

[8] J. Le Moigne, W. J. Campbell and R. F. Cromp, "An automated parallel image registration technique of multiple source remote sensing data," *IEEE Trans, Geosci. Remote Sensing*, vol. 40, pp. 1849-1864, 2004.

[9] J. C. Tilton, "Method for implementation of recursive hierarchical segmentation on parallel computers," U.S. Patent no. 09/839147 (pending published: http://www.fuentek.com/technologies/rhseg.htm).

[10] A. Lastovetsky, *Parallel computing on heterogeneous networks*, Wiley-Interscience: Hoboken, NJ, 2003.

[11] R. Brightwell, L. A. Fisk, D. S. Greenberg, T. Hudson, M. Levenhagen, A. B. Maccabe and R. Riesen, "Massively parallel computing using commodity components," *Parallel Comp.*, vol. 26, pp. 243–266, 2000.

[12] T. El-Ghazawi, S. Kaewpijit and J. Le Moigne, "Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality," *Proc. 3rd IEEE Intl. Conf. on Cluster Computing (Cluster'01)*, pp. 102-112, 2001.

[13] T. Achalakul and S. Taylor, "A distributed spectral-screening PCT algorithm," *Journal of Parallel and Distributed Computing*, vol. 63, pp. 373–384, 2003.

[14] A. Plaza, P. Martínez, R. Pérez and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sensing*, vol. 42, pp. 650–663, 2004.

[15] M.E. Winter, "N-FINDR: An algorithm for fast autonomous spectral endmember determination in hyperspectral data," *Proc. of SPIE Imaging Spectrometry Conference*, vol. 3753, pp. 266–277, 1999.

[16] A. Lastovetsky and R. Reddy, "On performance analysis of heterogeneous parallel algorithms," *Parallel Comp.*, vol. 30, pp. 1195–1216, 2004.

[17] M. Garcia and S. L. Ustin, "Detection of interannual vegetation responses to climatic variability using AVIRIS data in a coastal savanna in California," *IEEE Trans. Geosci. Remote Sensing*, vol. 39, pp. 1480–1490, 2001.

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