Parallel Morphological/Neural Classification of Remote Sensing Images Using Fully Heterogeneous and Homogeneous Commodity Clusters

Javier Plaza, Rosa Pérez, Antonio Plaza, Pablo Martínez and David Valencia
Neural Networks & Signal Processing Group (GRNPS)
Computer Science department, University of Extremadura
Avda. de la Universidad s/n, E-10071 Cáceres, Spain
Contact e-mail: jplaza@unex.es

Abstract

The wealth spatial and spectral information available from last-generation Earth observation instruments has introduced extremely high computational requirements in many applications. Most currently available parallel techniques treat remotely sensed data not as images, but as unordered listings of spectral measurements with no spatial arrangement. In thematic classification applications, however, the integration of spatial and spectral information can be greatly beneficial. Although such integrated approaches can be efficiently mapped in homogeneous commodity clusters, low-cost heterogeneous networks of computers (HNOCs) have soon become a standard tool of choice in Earth and planetary missions. In this paper, we develop a new morphological/neural parallel algorithm for commodity cluster-based analysis of high-dimensional remotely sensed image data sets. The algorithms accuracy and parallel performance are tested (in the context of a real precision agriculture application) using two parallel platforms: a fully heterogeneous cluster made up of 16 workstations at University of Maryland, and a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center.

1. Introduction

Many international agencies and research organizations are currently devoted to the analysis and interpretation of high-dimensional image data collected over the surface of the Earth [2]. For instance, NASA is continuously gathering hyperspectral images using Jet Propulsion Laboratory's Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) [5], which measures reflected radiation in the wavelength range from 0.4 to 2.5 μm using 224 spectral channels at spectral resolution of 10 nm (see Fig. 1). The incorporation of hyperspectral instruments aboard satellite platforms is now producing a nearly continual stream of high-dimensional remotely sensed data, and cost-effective techniques for information extraction and mining from massively large hyperspectral data repositories are highly required [1]. In particular, although it is estimated that several Terabytes of hyperspectral data are collected every day, about 70% of the collected data are never processed, mainly due to extremely high computational requirements.

Several challenges still remain open in the development of efficient data processing techniques for hyperspectral imagery [2]. For instance, previous research has demonstrated that the high-dimensional data space spanned by hyperspectral data sets is usually empty, indicating that the data structure involved exists primarily in a subspace. A commonly used approach to reduce the dimensionality of the data is the principal component transform (PCT). However, this approach is characterized by its global nature and cannot preserve subtle spectral differences required to obtain a good discrimination of classes [4]. Further, this approach relies on spectral properties of the data alone, thus neglecting the information related to the spatial arrangement of the pixels in the scene. As a result, there is a need for feature extraction techniques able to integrate the spatial and spectral information available from the data simultaneously [8].

While such integrated spatial/spectral developments hold great promise in the field of remote sensing data analysis, they introduce new processing challenges [9, 15]. The concept of Beowulf cluster was developed, in part, to address such challenges [13, 3]. The goal was to create parallel computing systems from commodity components to satisfy specific requirements for the Earth and space sciences community. Although most dedicated parallel machines employed by NASA and other institutions during the last decade have been chiefly homogeneous in nature, a current trend is to utilize heterogeneous clusters of computers [6]. In particular, computing on heterogeneous networks of computers (HNOCs) is an economical alternative which

1-4244-0328-6/06/$20.00

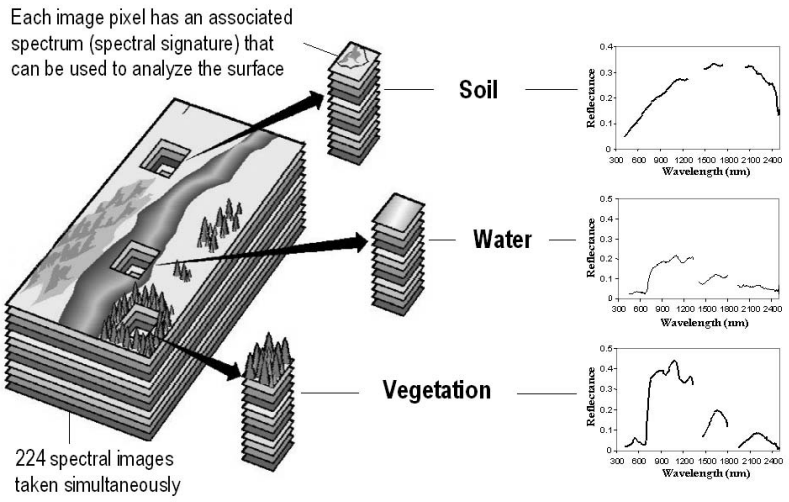


Figure 1. Concept of hyperspectral imaging using NASA Jet Propulsion Laboratory's AVIRIS system.

can benefit from local (user) computing resources while, at the same time, achieve high communication speed at lower prices. The properties above have led HNOCs to become a standard tool for high-performance computing in many ongoing and planned remote sensing missions [1, 6].

To address the need for cost-effective and innovative algorithms in this emerging new area, this paper develops a new parallel algorithm for classification of hyperspectral imagery. The algorithm is inspired by previous work on morphological neural networks, such as autoassociative morphological memories and morphological perceptrons [11], although it is based on different concepts. Most importantly, it can be tuned for very efficient execution on both HNOCs and massively parallel, Beowulf-type commodity clusters. The remainder of the paper is structured as follows. Section 2 describes the heterogeneous parallel algorithm, which consists of two main processing steps: 1) parallel morphological feature extraction taking into account the spatial and spectral information, and 2) robust classification using a parallel multi-layer neural network with backpropagation learning. Section 3 describes the algorithm's accuracy and parallel performance. Classification accuracy is first discussed in the context of a real precision agriculture application, based on hyperspectral data collected by NASA/JPL AVIRIS sensor over the valley of Salinas in California. Parallel performance is then assessed by comparing the efficiency achieved by the heterogeneous version of the algorithm, executed on a fully heterogeneous HNOC, with the efficiency achieved by its equivalent homogeneous version, executed on a fully homogeneous HNOC with the same aggregate performance as the heterogeneous one. For comparative purposes, performance data on Thunderhead, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center, are also given. Section 4 concludes with some remarks and hints at future research.

2. Parallel algorithm

This section describes a new parallel algorithm for analysis of remotely sensed hyperspectral images. Before describing the two main steps of the algorithm, we first formulate a general optimization problem in the context of HNOCs, composed of different-speed processors that communicate through links at different capacities [6]. This type of platform can be modeled as a complete graph $G = (P, E)$ where each node models a computing resource p_i weighted by its relative cycle-time w_i . Each edge in the graph models a communication link weighted by its relative capacity, where c_{ij} denotes the maximum capacity of the slowest link in the path of physical communication links from p_i to p_j. We also assume that the system has symmetric costs, i.e., $c_{ij} = c_{ji}$. Under the above assumptions, processor p_i will accomplish a share of $\alpha_i \times W$ of the total workload W, with $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$. With the above assumptions in mind, an abstract view of our problem can be simply stated in the form of a client-server architecture, in which the server is responsible for the efficient distribution of work among the P nodes, and the clients operate with the spatial and spectral information contained in a local partition. The partitions are then updated locally and the resulting calculations may also be exchanged between the clients, or between the server and the clients. Below, we describe the two steps of our parallel algorithm.

2.1 Parallel morphological processing

This section develops a parallel morphological feature extraction algorithm for hyperspectral image analysis. First, we briefly introduce the concept of spectral matching. Then, we describe the morphological algorithm and its parallel implementation for HNOCs.

2.1.1 Spectral matching in hyperspectral imaging

Let us first denote by f a hyperspectral image defined on an N-dimensional (*N*-D) space, where N is the number of channels or spectral bands in the image. A widely used technique to measure the similarity between spectral signatures in the input data is spectral matching [2], which evaluates the amount of correlation between the signatures. For instance, a widely used distance in hyperspectral analysis is the spectral angle mapper (SAM), which can be used to measure the spectral similarity between two pixels, $f(x,y)$ and $f(i,j)$, i.e., two *N*-D vectors at discrete spatial coordinates (x,y) and $(i,j) \in Z^2$, as follows:

$$\text{SAM}(f(x,y), f(i,j)) = cos^{-1} \frac{f(x,y) \cdot f(i,j)}{\|f(x,y)\| \cdot \|f(i,j)\|} \quad (1)$$

2.1.2 Morphological feature extraction algorithm

The proposed feature extraction method is based on mathematical morphology [12] and spectral matching concepts. The goal is to impose an ordering relation (in terms of spectral purity) in the set of pixel vectors lying within a spatial search window (called structuring element) designed by B [8]. This is done by defining a cumulative distance between a pixel vector $f(x,y)$ and all the pixel vectors in the spatial neighborhood given by B (*B*-neighborhood) as follows: $D_B[f(x,y)] = \sum_i \sum_j \text{SAM}[f(x,y), f(i,j)]$, where (x,y) refers to spatial coordinates in the *B*-neighborhood. From the above definitions, two standard morphological operations called erosion and dilation can be respectively defined as follows:

$$(f \otimes B)(x,y) = argmin_{(s,t) \in Z^2(B)} \sum_s \sum_t \text{SAM}(f(x,y), f(x+s, y+t)) \quad (2)$$

$$(f \oplus B)(x,y) = argmax_{(s,t) \in Z^2(B)} \sum_s \sum_t \text{SAM}(f(x,y), f(x-s, y-t)) \quad (3)$$

Using the above operations, the opening filter is defined as $(f \circ B)(x,y) = [(f \otimes C) \oplus B](x,y)$ (erosion followed by dilation), while the closing filter is defined as $(f \bullet B)(x,y) = [(f \oplus C) \otimes B](x,y)$ (dilation followed by erosion). The composition of opening and closing operations is called a spatial/spectral profile, which is defined as a vector which stores the relative spectral variation for every step of an increasing series. Let us denote by $\{(f \circ B)^\lambda(x,y)\}, \lambda = \{0,1,...,k\}$, the *opening series* at $f(x,y)$, meaning that several consecutive opening filters are applied using the same window B. Similarly, let us denote by $\{(f \bullet B)^\lambda(x,y)\}, \lambda = \{0,1,...,k\}$, the *closing series* at $f(x,y)$. Then, the spatial/spectral profile at $f(x,y)$ is given by the following vector:

$$p(x,y) = \{\text{SAM}((f \circ B)^\lambda(x,y), (f \circ B)^{\lambda-1}(x,y))\} \cup \{\text{SAM}((f \bullet B)^\lambda(x,y), (f \bullet B)^{\lambda-1}(x,y))\} \quad (4)$$

Here, the step of the opening/closing series iteration at which the spatial/spectral profile provides a maximum value gives an intuitive idea of both the spectral and spatial distribution in the *B*-neighborhood [8]. As a result, the profile can be used as a feature vector on which the classification is performed using a spatial/spectral criterion.

2.1.3 Parallel implementation

Two types of partitioning can be exploited in the parallelization of spatial/spectral algorithms such as the one addressed above [9]. 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. With this model, each pixel vector is split amongst several processors, which breaks the spectral identity of the data because the calculations for each pixel vector (e.g., for the SAM calculation) need to originate from several different processing units. On the other hand, spatial-domain partitioning provides data chunks in which the same pixel vector is never partitioned among several processors. In this work, we adopt a spatial-domain partitioning approach due to several reasons. First, the application of spatial-domain partitioning is a natural approach for morphological image processing, as many operations require the same function to be applied to a small set of elements around each data element present in the image data structure, as indicated in the previous subsection. A second reason has to do with the cost of inter-processor communication. In spectral-domain partitioning, the window-based calculations made for each hyperspectral pixel need to originate from several processing elements, in particular, when such elements are located at the border of the local data partitions (see Fig. 2), thus requiring intensive inter-processor communication.

However, if redundant information such as an overlap border is added to each of the adjacent partitions to avoid accesses outside the image domain, then boundary data to be communicated between neighboring processors can be

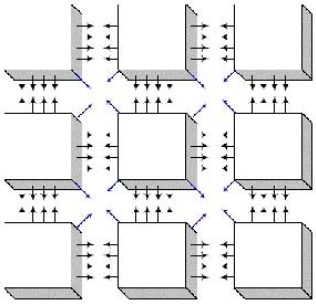


Figure 2. Communication framework for the morphological feature extraction algorithm.

greatly minimized. Such an overlapping scatter would obviously introduce redundant computations, since the intersection between partitions would be non-empty. Our implementation makes use of a constant structuring element *B* (with size of 3×3 pixels) which is repeatedly iterated to increase the spatial context, and the total amount of redundant information is minimized. To do so, we have implemented a special 'overlapping scatter' operation that also sends out the overlap border data as part of the scatter operation itself (i.e., redundant computations replace communications). Here, we make use of MPI *derived datatypes* to directly scatter hyperspectral data structures, which may be stored non-contiguously in memory, in a single communication step. A comparison between the associative costs of redundant computations in overlap with the overlapping scatter approach, versus the communications costs of accessing neighboring cell elements outside of the image domain has been presented and discussed in previous work [9].

A pseudo-code of the proposed parallel algorithm, specifically tuned for HNOCs, is given below.

HeteroMORPH algorithm:

Inputs: N-dimensional cube f, Structuring element B.

Output: Set of morphological profiles for each pixel.

1. Obtain information about the heterogeneous system, including the number of processors, P, each processors identification number, $\{p_i\}_{i=1}^{P}$, and processor cycle-times, $\{w_i\}_{i=1}^{P}$.

2. Using B and the information obtained in step 1, determine the total volume of information, R, that needs to be replicated from the original data volume, V, according to the data communication strategies outlined above, and let the total workload W to be handled by the algorithm be given by $W = V + R$.

3. Set $\alpha_i = \lfloor \frac{(P/w_i)}{\sum_{i=1}^{P}(1/w_i)} \rfloor$ for all $i \in \{1, ..., P\}$.

4. For $m = \sum_{i=1}^{P} \alpha_i$ to $(V + R)$, find $k \in \{1, .., P\}$ so that $w_k \cdot (\alpha_k + 1) = \min\{w_i \cdot (\alpha_i + 1)\}_{i=1}^{P}$ and set $\alpha_k = \alpha_k + 1$.

5. Use the resulting $\{\alpha_i\}_{i=1}^{P}$ to obtain a set of P spatial-domain heterogeneous partitions (with overlap borders) of W, and send each partition to processor p_i, along with B.

6. Calculate the morphological profiles $p(x, y)$ for the pixels in the local data partitions (in parallel) at each heterogeneous processor.

7. Collect all the individual results and merge them together to produce the final output.

A homogeneous version of the HeteroMORPH algorithm above can be simply obtained by replacing step 4 with $\alpha_i = P/w_i$ for all $i \in \{1, ..., P\}$, where w_i is the communication speed between processor pairs in the cluster, which is assumed to be homogeneous.

2.2. Parallel neural network processing

In this section, we describe a supervised parallel classifier based on a multi-layer perceptron (MLP) neural network with back-propagation learning. This approach has been shown in previous work to be very robust for classification of hyperspectral imagery [10]. However, the considered neural architecture and back-propagation-type learning algorithm introduce additional considerations for parallel implementation on HNOCs.

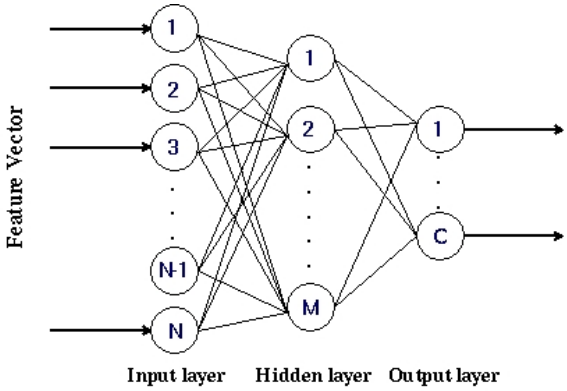


Figure 3. MLP neural network topology.

2.2.1 Network architecture and learning

The architecture adopted for the proposed MLP-based neural network classifier is shown in Fig. 3. As shown by the figure, the number of input neurons equals the number of

spectral bands acquired by the sensor. In the case of PCT-based pre-processing or morphological feature extraction commonly adopted in hyperspectral analysis, the number of neurons at the input layer equals the dimensionality of feature vectors used for classification. The second layer is the hidden layer, where the number of nodes, M, is usually estimated empirically. Finally, the number of neurons at the output layer, C, equals the number of distinct classes to be identified in the input data. With the above architecture in mind, the standard back-propagation learning algorithm can be outlined by the following steps:

1. *Forward phase*. Let the individual components of an input pattern be denoted by $f_j(x,y)$, with $j = 1,2,...,N$. The output of the neurons at the hidden layer are obtained as: $H_i = \varphi(\sum_{j=1}^{N} \omega_{ij} \cdot f_j(x,y))$ with $i = 1,2,...,M$, where $\varphi(\cdot)$ is the activation function and ω_{ij} is the weight associated to the connection between the *i*-th input node and the *j*-th hidden node. The outputs of the MLP are obtained using $O_k = \varphi(\sum_{i=1}^{M} \omega_{ki} \cdot H_i)$, with $k = 1,2,...,C$. Here, ω_{ki} is the weight associated to the connection between the *i*-th hidden node and the *k*-th output node.

2. *Error back-propagation*. In this stage, the differences between the desired and obtained network outputs are calculated and back-propagated. The *delta* terms for every node in the output layer are calculated using $\delta_k^o = (O_k - d_k) \cdot \varphi^{'}(\cdot)$, with $i = 1,2,...,C$. Here, $\varphi^{'}(\cdot)$ is the first derivative of the activation function. Similarly, *delta* terms for the hidden nodes are obtained using $\delta_i^h = \sum_{k=1}^{C} (\omega_{ki} \cdot \delta_i^o) \cdot \varphi(\cdot))$, with $i = 1,2,...,M$.

3. *Weight update*. After the back-propagation step, all the weights of the network need to be updated according to the *delta* terms and to η, a learning rate parameter. This is done using $\omega_{ij} = \omega_{ij} + \eta \cdot \delta_i^h \cdot f_j(x,y)$ and $\omega_{ki} = \omega_{ki} + \eta \cdot \delta_k^o \cdot H_i$. Once this stage is accomplished, another training pattern is presented to the network and the procedure is repeated for all incoming training patterns.

Once the back-propagation learning algorithm is finalized, a classification stage follows, in which each input pixel vector is classified using the weights is obtained by the network during the training stage [10].

2.2.2 Parallel multi-layer perceptron classifier

The parallel classifier presented in this work is based on a hybrid partitioning scheme, in which the hidden layer is partitioned using neuronal level parallelism and weight connections are partitioned on the basis of synaptic level parallelism [14]. As a result, the input and output neurons are common to all processors, while the hidden layer is partitioned so that each heterogeneous processor receives a number of hidden neurons which depends on its relative speed. Each processor stores the weight connections between the neurons local to the processor. Since the fully connected MLP network is partitioned into P partitions and then mapped onto P heterogeneous processors using the above framework, each processor is required to communicate with every other processor to simulate the complete network. For this purpose, each of the processors in the network executes the three phases of the back-propagation learning algorithm described above.

HeteroNEURAL algorithm:

Inputs: N-dimensional cube f, Training patterns $f_j(x,y)$.

Output: Set of classification labels for each image pixel.

1. Use steps 1-4 of HeteroMORPH algorithm to obtain a set of values $(\alpha_i)_{i=1}^{P}$ which will determine the share of the workload to be accomplished by each heterogeneous processor.

2. Use the resulting $(\alpha_i)_{i=1}^{P}$ to obtain a set of *P* heterogeneous partitions of the hidden layer and map the resulting partitions among the *P* heterogeneous processors (which also store the full input and output layers along with all connections involving local neurons).

3. *Parallel training.* For each considered training pattern, the following three parallel steps are executed:

   (a) *Parallel forward phase.* In this phase, the activation value of the hidden neurons local to the processors are calculated. For each input pattern, the activation value for the hidden neurons is calculated using $H_i^P = \varphi(\sum_{j=1}^{N} \omega_{ij} \cdot f_j(x,y))$. Here, the activation values and weight connections of neurons present in other processors are required to calculate the activation values of output neurons according to $O_k^P = \varphi(\sum_{i=1}^{M/P} \omega_{ki}^P \cdot H_i^P)$, with $k = 1,2,...,C$. In our implementation, broadcasting the weights and activation values is circumvented by calculating the partial sum of the activation values of the output neurons.

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

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

4. *Classification.* For each pixel vector in the input data cube f, calculate (in parallel) $\sum_{j=1}^P O_k^j$, with $k = 1, 2, ..., C$. A classification label for each pixel can be obtained using the winner-take-all criterion commonly used in neural networks by finding the cumulative sum with maximum value, say $\sum_{j=1}^P O_{k^*}^j$, with $k^* = \arg\{\max_{1 \leq k \leq C} \sum_{j=1}^P O_k^j\}$.

3. Experimental results

This section provides an assessment of the effectiveness of the parallel algorithms described in section 2. The section is organized as follows. First, we describe a framework for assessment of heterogeneous algorithms and provide an overview of the heterogeneous and homogeneous clusters 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 subsection.

3.1. Network description

Following a recent study [7], we assess the proposed heterogeneous algorithms using the basic postulate that they cannot be executed on a heterogeneous cluster faster than its homogeneous prototype on the equivalent homogeneous cluster. Let us assume that a heterogeneous cluster consists of $\{p_i\}_i^P$ 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 $t = 1, ..., p^{(j)}$. Finally, let $c^{(j,k)}$ be the speed of the communication link between segments s_j and s_k, with $j, k = 1, ..., m$. According to [7], the above cluster can be considered equivalent to a homogeneous one made up of $\{q_i\}_{i=1}^P$processors with constant cycle-time and interconnected through a homogeneous communication network with speed c if, and only if the following expressions are satisfied:

$$c = \frac{\sum_{j=1}^m c^{(j)} \cdot [\frac{p^{(j)}(p^{(j)}-1)}{2}]}{\frac{P(P-1)}{2}} + \cdots \\ \cdots + \frac{\sum_{j=1}^m \sum_{k=j+1}^m p^{(j)} \cdot p^{(k)} \cdot c^{(j,k)}}{\frac{P(P-1)}{2}} \quad (5)$$

Table 2. Capacity of communication links (in time in milliseconds to transfer a one-megabit 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

and

$$w = \frac{\sum_{j=1}^m \sum_{t=1}^{p^{(j)}} w_t^{(j)}}{P} \quad (6)$$

where the first expression states that the average speed of point-to-point communications between processors $\{p_i\}_{i=1}^P$ in the heterogeneous cluster should be equal to the speed of point-to-point communications between processors $\{q_i\}_{i=1}^P$ in the homogeneous cluster, with both clusters 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 the proposed heterogeneous hyperspectral imaging algorithm:

- *Heterogeneous network.* Consists of 16 different workstations, and four communication segments. Table 1 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 2 also shows the capacity of all point-to-point communications in the heterogeneous network, expressed as the time in milliseconds to transfer a one-megabit message between each processor pair (p_i, p_j) in the heterogeneous system. As noted, the communication network of the fully heterogeneous network consists of four relatively fast homogeneous communication segments, interconnected by three slower communication links with capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$, $c^{(3,4)} = 58.14$ in milliseconds, respectively. Although this is a simple architecture, it is also a quite typical and realistic one as well.

- *Homogeneous network.* Consists of 16 identical Linux workstations $\{q_i\}_{i=1}^1 6$ with processor cycle-time of $w = 0.0131$ seconds per megaflop, interconnected via

Table 1. Specifications of heterogeneous processors.

Processor	Architecture	Cycle-time (secs/megaflop)	Main 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

a homogeneous communication network where the capacity of links is $c = 26.64$ milliseconds.

Finally, in order to test the proposed algorithm on a large-scale parallel platform, we have also experimented with Thunderhead, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center. It is composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 GB of main memory and 80 GB of disk space and interconnected via 2 GHz optical fibre Myrinet (see http://thunderhead.gsfc.nasa.gov for additional details). The total peak performance of the system is 2457.6 Gflops. In all considered platforms, the operating system used at the time of experiments was Linux RedHat 8.0, and MPICH was the message-passing library used.

3.2. Hyperspectral data

Before empirically investigating the performance of parallel hyperspectral imaging algorithms, we first describe a hyperspectral image scene that will be used in experiments. The scene was collected by the 224-band AVIRIS sensor over Salinas Valley, California, and is characterized by high spatial resolution (3.7-meter pixels). The relatively large area covered (512 lines by 217 samples) results in a total image size of more than 1 GB. Fig. 4(a) shows the spectral band at 587 nm wavelength and a sub-scene (called hereinafter Salinas A), which comprises 83x86 pixels and is dominated by directional features. Fig. 4(b) shows the ground-truth map, in the form of a class assignment for each labeled pixel with 15 mutually exclusive ground-truth classes. As shown by Fig. 4(b), ground truth is available for nearly half of Salinas scene.

The data set above represents a very challenging classification problem (due to the spectral similarity of most classes, discriminating among them is very difficult). This fact has made the scene a universal and widely used benchmark to validate classification accuracy of hyperspectral algorithms [8]. In order to test the accuracy of the proposed parallel morphological/neural classifier, a random sample of less than 2% of the pixels was chosen from the known ground truth of the 15 land-cover classes in Fig. 4(b). Morphological profiles were then constructed in parallel for the selected training samples using 10 iterations, which resulted in feature vectors with dimensionality of 20 (i.e., 10 structuring element iterations for the *opening series* and 10 iterations for the *closing series*). The resulting features were then used to train the parallel back-propagation neural network classifier with one hidden layer, where the number of hidden neurons was selected empirically as the square root of the product of the number of input features and information classes (several configurations of the hidden layer were tested and the one that gave the highest overall accuracies was reported). The trained classifier was then applied to the remaining 98% of labeled pixels in the scene, yielding the classification accuracies shown in Table 3.

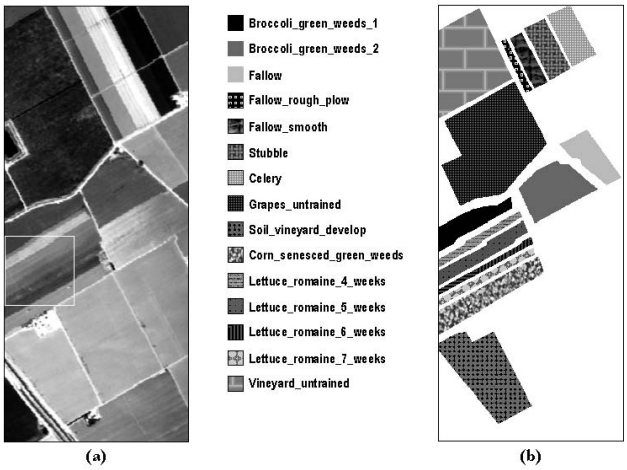


Figure 4. (a) AVIRIS scene collected over Salinas Valley, California, and (b) Land-cover ground classes.

For comparative purposes, the accuracies obtained using the full spectral information and PCT-reduced features as input to the neural classifier are also reported in Table 3. As shown by the table, morphological input features substantially improve individual and overall classification accuracies with regards to PCT-based features and the full spectral information (e.g., for the directional "lettuce" classes contained in the Salinas A subscene). This is not surprising since morphological operations use both spatial and spectral information as opposed to the other methods which rely on spectral information alone. For illustrative purposes, Table 3 also includes (in the parentheses) the algorithm processing times in seconds for the different approaches tested,

Class	Spectral information (2981)	PCT-based features (3256)	Morphological features (3679)
Fallow rough plow	96.51	91.90	96.78
Fallow smooth	93.72	93.21	97.63
Stubble	94.71	95.43	98.96
Celery	89.34	94.28	98.03
Grapes untrained	88.02	86.38	95.34
Soil vineyard develop	88.55	84.21	90.45
Corn senesced green weeds	82.46	75.33	87.54
Lettuce romaine 4 weeks	78.86	76.34	83.21
Lettuce romaine 5 weeks	82.14	77.80	91.35
Lettuce romaine 6 weeks	84.53	78.03	88.56
Lettuce romaine 7 weeks	84.85	81.54	86.57
Vineyard untrained	87.14	84.63	92.93
Overall accuracy	87.25	86.21	95.08

Table 3. Classification accuracies in percentage achieved by the parallel neural classifier using morphological features, PCT-based features and the original spectral information (processing times in a single Thunderhead node are given in the parentheses).

measured on a single processor in the Thunderhead system. Experiments were performed using the GNU-C/C++ compiler in its 4.0 version. As shown by Table 3, the computational cost was slightly higher when morphological feature extraction was used.

3.3. Assessment of the parallel algorithm

To investigate the properties of the parallel algorithm, its performance was first tested by timing the program using the heterogeneous network and its equivalent homogeneous one. As expected, the execution times reported on Table 4 indicate that the heterogeneous implementations of morphological feature extraction (HeteroMORPH) and neural network-based classification (HeteroNEURAL) algorithms were both able to adapt much better to the heterogeneous computing environment than their respective homogeneous versions, which were only able to perform satisfactorily on the homogeneous cluster. For the sake of comparison, Table 4 also shows the performance ratio between the heterogeneous algorithms and their respective homogeneous versions (referred to as Homo/Hetero ratio in the table and 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 algorithms were always several times faster than their homogeneous counterparts in the heterogeneous cluster, while the homogeneous algorithms only slightly outperformed their heterogeneous counterparts in the homogeneous cluster. The Homo/Hetero ratios reported in the table for the homogeneous algorithms executed on the homogeneous cluster were indeed very close to 1, a fact that 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 environment. The above result demonstrates the flexibility of the proposed heterogeneous

	Homogeneus cluster		Heterogeneus cluster	
Algorithm	Time	Homo/Hetero	Time	Homo/Hetero
HeteroMORPH	221	1.11	206	10.98
HomoMORPH	198		2261	
HeteroNEURAL	141	1.12	130	9.70
HomoNEURAL	125		1261	

Table 4. Execution times (in seconds) and performance ratios reported for the homogeneous algorithms versus the heterogeneous ones on the two considered clusters.

algorithms, which were able to adapt efficiently to the two considered clusters. Interestingly, Table 4 also reveals that the performance of the heterogeneous algorithms on the heterogeneous cluster was almost the same as that evidenced by the equivalent homogeneous algorithms on the homogeneous cluster (i.e., the algorithms achieved essentially the same speed, but each on its network). This seems to indicate that the heterogeneous algorithms are very close to the optimal heterogeneous modification of the basic homogeneous ones.

In order to measure load balance, Table 5 shows the imbalance scores achieved by the parallel algorithms on the two considered clusters. The imbalance is defined as $D = R_{max}/R_{min}$, where R_{max} and R_{min} are the maxima and minima processor run times, respectively. Therefore, perfect balance is achieved when $D = 1$. In the table, we display the imbalance considering all processors, D_{All}, and also considering all processors but the root, D_{Minus}. As we can see from Table 5, both the HeteroMORPH and HeteroNEURAL algorithms were able to provide values of D_{All} close to 1 in the two considered clusters. Further, the above algorithms provided almost the same results for both D_{All} and D_{Minus} while, for the homogeneous versions, load balance was much better when the root processor was not included. While the homogeneous algorithms executed on the heterogeneous cluster provided the highest values of

Algorithm	Homogeneus cluster		Heterogeneus cluster	
	D_{All}	D_{Minus}	D_{All}	D_{Minus}
HeteroMORPH	1.03	1.02	1.05	1.01
HomoMORPH	1.05	1.01	1.59	1.21
HeteroNEURAL	1.02	1.01	1.03	1.01
HomoNEURAL	1.03	1.01	1.39	1.19

Table 5. Load-balancing rates for the parallel algorithms on the homogeneous and heterogeneous cluster.

D_{All} and D_{Minus} (and hence the highest imbalance), the heterogeneous algorithms executed on the homogeneous cluster resulted in values of D_{Minus} which were close to optimal. Despite the fact that conventional feature extraction algorithms (such as those based on PCT) do not take into account the spatial information explicitly into the computations –a fact that has traditionally been perceived as an advantage for the development of parallel implementations– and taking into account that both HeteroMORPH and HeteroNEURAL introduce redundant information expected to slow down the computation a priori, results in Table 5 indicate that the two heterogeneous algorithms are effective in finding an appropriate workload distribution among the heterogeneous processors.

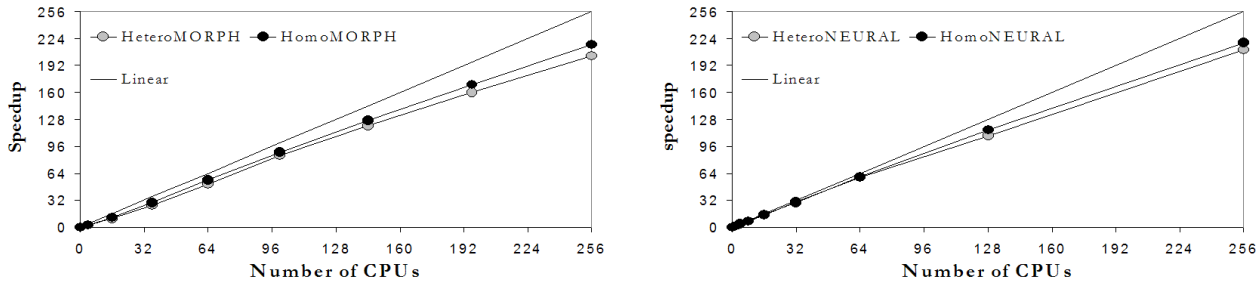
Taking into account the results presented above, and with the ultimate goal of exploring issues of scalability (considered to be a highly desirable property in the design of heterogeneous parallel algorithms), we have also compared the performance of the heterogeneous algorithms and their homogeneous versions on the Thunderhead Beowulf cluster. Fig. 5 plots the speedups achieved by multi-processor runs of the heterogeneous algorithms over the corresponding single-processor runs of each considered algorithm on Thunderhead. For the sake of comparison, Fig. 5 also plots the speedups achieved by multi-processor runs of the homogeneous versions on Thunderhead. As Fig. 5 shows, the scalability of heterogeneous algorithms was essentially the same as that evidenced by their homogeneous versions, with both HeteroNEURAL and HeteroMORPH showing scalability results close to linear in spite of the fact that the two algorithms introduce redundant computations expected to slow down the computation a priori.

For the sake of quantitative comparison, Table 6 reports the measured execution times achieved by all tested algorithms on Thunderhead, using different numbers of processors. The times reported on Table 6 reveal that the combination of HeteroMORPH for spatial/spectral feature extraction, followed by HeteroNEURAL for robust classification was able to obtain highly accurate hyperspectral analysis results (in light of Table 3), but also quickly enough for practical use. For instance, using 256 Thunderhead processors, the proposed classifier was able to provide a highly accurate classification for the Salinas AVIRIS scene in less than 20 seconds. In this regard, the measured processing times represent a significant improvement over commonly used processing strategies for this kind of high-dimensional data sets, which can take up to more than one hour of computation for the considered problem size, as indicated by the single-processor execution times reported on Table 3.

Overall, experimental results in our study reveal that the proposed heterogeneous parallel algorithms offer a relatively platform-independent and highly scalable solution in the context of realistic hyperspectral image analysis applications. Contrary to common perception that spatial/spectral feature extraction and back-propagation learning algorithms are too computationally demanding for practical use and/or (near) real time exploitation in hyperspectral imaging, results in this paper demonstrate that such approaches are indeed appealing for parallel implementation, not only due to the regularity of the computations involved in such algorithms, but also because they can greatly benefit from the incorporation of redundant information to reduce sequential computations at the master node and involve minimal communication between the parallel tasks, namely, at the beginning and ending of such tasks.

4. Conclusions and future research

In this paper, we have presented several innovative parallel algorithms for hyperspectral image analysis and implemented them on both heterogeneous and homogeneous commodity clusters. As a case study of specific issues involved in the exploitation of heterogeneous algorithms for hyperspectral image information extraction, this paper provided a detailed discussion on the effects that platform heterogeneity has on degrading parallel performance of a highly innovative morphological/neural classification algorithm, able to exploit the spatial and spectral information in simultaneous fashion. The parallel performance evaluation strategy conducted in this work was based on experimentally assessing the heterogeneous algorithm by comparing its efficiency on a fully heterogeneous cluster (made up of processing units with different speeds and highly heterogeneous communication links) with the efficiency achieved by its equivalent homogeneous version on an equally powerful homogeneous cluster. Scalability results on massively parallel commodity clusters are also provided. Experimental results in this work anticipate that the combination of the (readily available) computational power offered by heterogeneous architectures and the recent advances in the design of last-generation Earth observation sensors and new parallel algorithms (such as those presented in this work) may introduce substantial changes in the systems currently used for exploiting the sheer volume of hyperspectral data which is now being collected worldwide, on a daily basis.



(a) (b)

Figure 5. Scalability of morphological feature extraction (a) and neural network-based (b) parallel algorithms on Thunderhead.

Processors:	1	4	16	36	64	100	144	196	256
HeteroMORPH	2041	797	203	79	39	23	17	13	10
HomoMORPH		753	170	70	36	22	16	12	9
Processors:	1	2	4	8	16	32	64	128	256
HeteroNEURAL	1638	985	468	239	122	61	30	18	9
HomoNEURAL		973	458	222	114	55	27	15	7

Table 6. Processing times (in seconds) achieved by multi-processor runs of the considered parallel algorithms on Thunderhead.

5 Acknowledgement

The authors would like to thank Drs. John Dorband, James C. Tilton and Anthony Gualtieri for their with experiments on NASA's Thunderhead system and also for many helpful discussions. They also state their appreciation for Profs. Mateo Valero and Francisco Tirado.

References

[1] G. Aloisio and M. Cafaro. A dynamic earth observation system. *Parallel Computing*, 29:1357–1362, 2003.

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

[3] J. Dorband, J. Palencia, and U. Ranawake. Commodity clusters at Goddard Space Flight Center. *Journal of Space Communication*, 3:227–248, 2003.

[4] T. El-Ghazawi, S. Kaewpijit, and J. L. Moigne. Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality. *Cluster Computing*, pages 102–110, 2001.

[5] R. O. Green. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sensing of Environment*, 65:227–248, 1998.

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

[7] A. Lastovetsky and R. Reddy. On performance analysis of heterogeneous parallel algorithms. *Parallel Computing*, 30:1195–1216, 2004.

[8] A. Plaza, P. Martinez, J. Plaza, and R. Perez. Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 43:466–479, 2005.

[9] A. Plaza, D. Valencia, J. Plaza, and P. Martinez. Commodity cluster-based parallel processing of hyperspectral imagery. *Journal of Parallel and Distributed Computing*, 66:345–358, 2006.

[10] J. Plaza, A. Plaza, R. Perez, and P. Martinez. Automated generation of semi-labeled training samples for nonlinear neural network-based abundance estimation in hyperspectral data. *Proceedings IEEE International Geoscience and Remote Sensing Symposium*, pages 345–350, 2005.

[11] G. X. Ritter, P. Sussner, and J. L. Diaz. Morphological associative memories. *IEEE Transactions on Neural Networks*, 9:281–293, 2004.

[12] P. Soille. *Morphological image analysis: Principles and applications*. Springer: Berlin, 2003.

[13] T. Sterling. Cluster computing. *Encyclopedia of Physical Science and Technology*, 3, 2002.

[14] S. Suresh, S. N. Omkar, and V. Mani. Parallel implementation of back-propagation algorithm in networks of workstations. *IEEE Transactions on Parallel and Distributed Systems*, 16:24–34, 2005.

[15] P. Wang, K. Y. Liu, T. Cwik, and R. Green. MODTRAN on supercomputers and parallel computers. *Parallel Computing*, 28:53–64, 2002.