Parallel Processing of High-Dimensional Remote Sensing Images Using Cluster Computer Architectures

David Valencia*, Antonio Plaza*, Pablo Martínez*, Javier Plaza*
University of Extremadura, 10071 Caceres, SPAIN

Abstract

Hyperspectral sensors represent the most advanced instruments currently available for remote sensing of the Earth. The high spatial and spectral resolution of the images supplied by systems like the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS), developed by NASA Jet Propulsion Laboratory, allows their exploitation in diverse applications, such as detection and control of wildland fires and hazardous agents in water and atmosphere, detection of military targets and management of natural resources. Even though the above applications generally require a response in near real time, few solutions are currently available to provide fast and efficient processing of such high-dimensional image data sets. This is mainly due to the extremely high volume of data collected by hyperspectral sensors, which often limits their exploitation in analysis scenarios where the spatial and temporal requirements are very high. In this paper, we describe new parallel processing methodologies for hyperspectral image processing, based on neural architectures and morphological concepts. The computational performance of the proposed methods is demonstrated using real analysis scenarios based on the exploitation of AVIRIS data using two parallel computer systems, an SGI Origin 2000 multicomputer located at the Barcelona Supercomputing Center (BSC), and the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center (NASA/GSFC).

Key Words: Hyperspectral analysis, spectral classification, parallel computing, clusters of computers.

1 Introduction

The development of advanced instruments for remote observation of the Earth has created a growing interest in the design of efficient techniques for the interpretation of the images provided by these sensors. In particular, hyperspectral sensors represent the most advanced generation of remote sensing instruments for Earth observation and planetary exploration, and are characterized by their high resolution in both spatial and spectral domains [3]. For instance, the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS), developed by NASA Jet Propulsion Laboratory [5] covers the range of wavelengths from 0.4 to 2.5 μm using 224 spectral channels, with a spatial resolution of 20 meters per pixel and a nominal spectral resolution of 10 nm. As shown by Figure 1, the analytic capability of AVIRIS allows for the collection of a detailed spectral signature for each pixel in the image, where each spectral signature comprises a set of reflectance values measured by the sensor at different wavelengths. Such fingerprints can be used to accurately characterize the composition of each site in the scene. The exploitation of the data sets provided by this emerging type of sensors has been quite notorious in the recent years, especially since their incorporation to spatial satellite type platforms like NASA's Earth Observing (EO-1) or European Space Agency's ENVISAT, which offer almost global covering of the planet.

Despite the significant technological evolution of hyperspectral instruments, the developments in techniques for analysis of the data provided by these sensors have not been so notorious. In particular, the design of analysis techniques able to naturally integrate both the spatial and the spectral information contained in the data is still a challenge for the scientific community [15]. Many studies reveal that is it indeed possible to obtain thematic maps where each pixel is labeled as belonging to a single land-cover class by taking advantage of the high spectral resolution provided by hyperspectral sensors. However, it should be noted that the spatial resolution of hyperspectral sensors is usually in the order of several meters (e.g., 20-meter pixels for the AVIRIS instrument). As a result, pure pixel-based classification techniques may suffer from sub-pixel estimation errors. For demonstration purposes, we can use the following toy example: let us assume that a low resolution pixel, made up of a mixture of water and sand, is classified as either water or sand. In this scenario, the estimation error introduced by such a pure pixel-based (*hard*) classification approach could be as high as 50 percent depending on the dominant sub-pixel constituent. In order to overcome this limitation, a current trend in hyperspectral analysis is to resort to mixed pixel (*soft*) classification techniques, where a single pixel may be classified into several pure classes with different land-cover proportions.

In recent work, morphological approaches have been successfully applied to mixed pixel decomposition in

* Computer Architecture and Technology Section, Department of Computer Science, Avda. de la Universidad s/n. E-mail: {davaleco, aplaza, pablomar, jplaza}@unex.es.

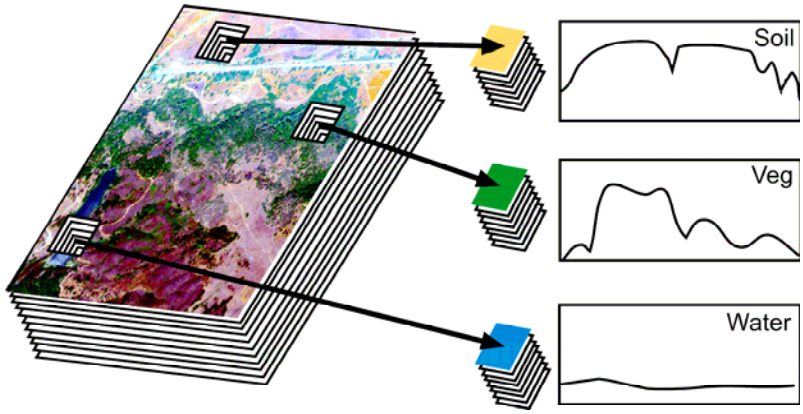


Figure 1: The concept of hyperspectral imaging

hyperspectral imaging. One of such approaches has been the Automated Morphological Endmember Extraction algorithm (AMEE) method, which automatically extracts a set of pure spectral signatures corresponding to non-contaminated macroscopic components such as water, soil, vegetation, etc. These components, often called spectral "endmembers" in hyperspectral analysis terminology, can be used to "unmix" a given pixel by expressing its associated spectrum as a linear/nonlinear combination of pure components. In some cases, spectral endmembers are ,also suitable to be used as input information for other applications. For instance, there are many situations where a detailed knowledge of image endmembers is not enough to extract a detailed land-cover classification map. In this context, artificial neural networks (ANNs) have demonstrated to be a powerful tool for hyperspectral imaging because the information provided by ANNs can not only be used to provide a *hard* classification, but also to obtain a *soft* classification, e.g., by taking into account the degree of membership (or similarity) of a certain input pattern (pixel vector) to a certain output class (endmember). In the field of ANN-based hyperspectral imaging, self-organizing maps (SOMs) have been recognized as a very powerful tool to perform both hard and soft classification. This model is based on an unsupervised learning strategy that does not require any previous test samples [8, 9]. Again, one of the main restrictions of SOM-based analysis is the computation time involved.

While integrated spatial/spectral developments hold great promise for Earth science image analysis, they create new processing challenges. In particular, the price paid for the wealth spatial and spectral information available from hyperspectral sensors is the enormous amounts of data that they generate. In addition, analysis techniques in Earth observation studies are often computationally tedious, and require lengthy durations to calculate desired quantities. Several applications exist, however, where having the desired information calculated in near real-time is highly desirable. For instance, detection and/or tracking of natural disasters such as forest fires, oil spills, and other types of chemical contamination demands timely processing output.

It is worth noting that, although parallel computing techniques have been widely used in general-purpose image processing applications [14], the use of large-scale computing facilities in hyperspectral imaging has been traditionally limited to a few institutions only. However, nowadays it is possible to design low cost "commodity" high-performance systems by resorting to personal computers or workstations, connected through high performance communications networks. In particular, Beowulf clusters were originally conceived at NASA's Goddard Space Flight Center (NASA/GSFC) to create a cost-effective parallel computing system to satisfy specific computational requirements for applications such as those present in the Earth and space sciences community [10].

In this paper, we develop a new parallel morphologyical/neural approach for hyperspectral image classification, and specifically discuss implementation aspects using several commodity cluster-based architectures. Although several parallel algorithms for remote sensing image analysis already exist in the open literature [1, 2, 4, 6, 7, 16], our parallel algorithm is one of the few available methods that considers both the spatial and the spectral information in a natural way. It relies on domain decomposition techniques aimed at minimizing inter-processor communication and maximizing load balance. The remainder of the paper is organized as follows. Section 2 describes the fundamentals of the proposed methodology, which rely on multi-channel mathematical morphology and ANNs. Section 3 provides a detailed description of the parallel implementation, which is based on C++ and the MPI message passing library. Section 4 conducts a detailed study of the computational performance of the parallel implementation using two parallel computers: an SGI Origin 2000 located at Barcelona Supercomputer Center (BSC) and the Thunderhead massively parallel supercomputer at NASA's Goddard Space Flight Center (NASA/GSFC). The paper concludes with some remarks and hints at plausible future research.

2 Methodology

The proposed methods for hyperspectral analysis can be included in the category of spectral unmixing and

classification approaches, respectively [11]. In the following subsection we provide a detailed description of the (soft) classification problem of spectral mixing, and then introduce a set of morphological operations oriented to solve this problem using and endmember extraction-based approach. The section concludes with the description a (hard) SOM-based classification technique which takes advantage of morphological endmember extraction to provide accurate class labels. The latter approach, although subject to potential inaccuracies at a sub-pixel level, is particularly useful for the purpose of developing thematic maps in land-cover and land-use applications. The two types of algorithms addressed above will be parallelized in the following section.

2.1 Spectral Unmixing

Mixed pixels are predominant in hyperspectral images and result as mixtures of more than one distinct substance. Mixed pixels exist for one of two reasons. Firstly, if the spatial resolution of the sensor is not high enough to separate different materials, these can jointly occupy a single pixel, and the resulting spectral measurement will be a composite of the individual spectra. Secondly, mixed pixels can also result when distinct materials are combined into a homogeneous mixture. This circumstance occurs independent of the spatial resolution of the sensor. A hyperspectral image is often a combination of the two situations, where a few sites in a scene are pure materials, but many other are mixtures of materials. Spectral unmixing is a commonly used procedure in which the measured spectrum of a mixed pixel is decomposed into a collection of spectrally pure constituent spectra, or endmembers [12, 13], and a set of correspondent fractions, or abundances, that indicate the proportion of each endmember in the pixel.

Identification of image endmembers is a crucial objective in hyperspectral image analysis applications. It is important to emphasize that most available techniques for endmember selection focus on analyzing the data without incorporating information on the spatially adjacent data; i.e., the hyperspectral data is treated not as an image but as an unordered listing of spectral measurements where the spatial coordinates can be shuffled arbitrarily without affecting the analysis. However, one of the distinguishing properties of hyperspectral data, as collected by available imaging spectrometers, is the multivariate information coupled with a two-dimensional (2-D) pictorial representation amenable to image interpretation. Subsequently, there is a need to incorporate the image representation of the data in the development of automated techniques for endmember selection and hyperspectral data exploitation. The main contribution of the endmember extraction algorithm described in this work is simultaneous consideration of both spatial and spectral information. By taking into account the complementary nature of spatial and spectral information in simultaneous fashion, it is possible to alleviate the problems related to each of them taken separately. The proposed method is based on mathematical morphology [13], a classic image analysis technique that is generalized to the case of multidimensional data in the following subsection.

2.2 Morphological Endmember Extraction Algorithm

Two basic operations articulate classic MM theory: erosion and dilation. They are respectively based on the selection of the maximum and minimum value of a neighborhood or spatial region around each pixel of the image, where the shape and size of the considered region are determined by the spatial properties of a neighborhood function called structuring element (SE). The main challenge in order to extend these operations to the case of hyperspectral image data is the lack of an ordering relation between the pixels of the image, which can be seen as L-dimensional (L-D) vectors where L is the number of spectral channels (see Figure 1). Following a usual notation, let $\boldsymbol{f}$ be an image defined on an L-D space and let B a so-called SE. We impose an ordering relation in terms of spectral purity in the set of pixel vectors lying within a flat SE, designed by B, by defining a cumulative distance between one particular pixel $\boldsymbol{f}(\mathrm{x},\mathrm{y})$, where $\boldsymbol{f}(\mathrm{x},\mathrm{y})$ denotes an L-D vector at discrete spatial coordinates $(\mathrm{x},\mathrm{y})\in Z^2$, and all the pixel vectors in the spatial neighborhood given by B (B-neighborhood) as follows:

$$D_B[\boldsymbol{f}(x,y)]=\sum_{\mathrm{s}}\sum_{\mathrm{t}}\mathrm{Dist}[\boldsymbol{f}(x,y),\,\boldsymbol{f}(\mathrm{s},\mathrm{t})] \quad \forall(s,t)\in Z^2(B), \qquad (1)$$

where Dist is a linear point-wise distance measure between two L-D vectors. As a result, $\mathrm{D}_B[\boldsymbol{f}(\mathrm{x},\mathrm{y})]$ is given by the sum of Dist scores between $\boldsymbol{f}(\mathrm{x},\mathrm{y})$ and every pixel vector in the B-neighborhood. Based on the cumulative distance above, the extended erosion of $\boldsymbol{f}$ by B is based on the selection of the B-neighborhood pixel vector that produces the minimum value for D_B:

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

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

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

where the `arg max` operator selects the pixel vector that is most highly different, according to Dist, to all the other pixels

in the B-neighborhood. In this work, our choice for Dist is a widely used distance metric in remote sensing applications: the spectral angle distance (SAD) [3]. If we consider our definition of an endmember as a spectrally pure element that can be used to describe mixed pixels in the image, it is clear that morphological operations exhibit a great potential in the task of detecting endmembers using the spatial and spectral information contained in the original image.

Based on the morphological concepts introduced above, we develop a methodology for endmember extraction which incorporates both spatial and spectral information. The proposed method is called Automated Endmember Extraction Algorithm (AMEE), and allows for soft classification of hyperspectral images in fully automated fashion. The algorithm consists of a sequence of steps which are outlined below.

AMEE algorithm

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

Outputs: Set of endmembers $\{\boldsymbol{e}_j\}_{j=1}^{p}$; Set of fractional abundances $\{\alpha_i(x,y)\}_{i=1}^{p}$ for each pixel $\boldsymbol{f}(x,y)$.

1) Set $i=1$ and initialize a morphological eccentricity index $MEI(x,y)=0$ for each pixel $\boldsymbol{f}(x,y)$.
2) Move B through all the pixels of $\boldsymbol{f}$, defining a local spatial search area around each $\boldsymbol{f}(x,y)$ and calculate the maximum pixel $(\boldsymbol{f}\oplus B)(x,y)$ and the minimum pixel $(\boldsymbol{f}\ominus B)(x,y)$ at each B-neighborhood. Update the resulting MEI score at each pixel selected as a local maximum, $\boldsymbol{f}(x',y')=(\boldsymbol{f}\oplus B)(x,y)$, using the following expression:

$$MEI(x',y')=MEI(x',y')+SAD[(\boldsymbol{f}\oplus B)(x,y),(\boldsymbol{f}\ominus B)(x,y)] \quad (4)$$

1) Set $i=i+1$. If $i=I_{\max}$ then go to step 4. Otherwise, set $\boldsymbol{f}=\boldsymbol{f}\oplus B$ and go to step 2.
2) Select the set of p pixels $\{\boldsymbol{e}_j\}_{j=1}^{p}$ in $\boldsymbol{f}$ with higher score in the resulting MEI image. These pixels form the final endmember set.

Once a final set of endmembers has been extracted, these endmembers are generally coupled with a linear or nonlinear model to expressed each mixed pixel as a combination of endmembers. In this work, and for illustrative purposes, we resort to a simple linear mixture model. There are two main reasons for our choice of this model: i) it is the most standardized approach in the remote sensing community, and ii) it is very easy to implement, resulting in an "embarrassingly parallel" implementation which can work on a pixel-by-pixel basis, thus allowing a simple distribution of the workload among a set of parallel processors. The linear model is extensively described in the literature, but the specific implementation used in this paper is described in detail in [3].

2.3 Self-Organizing Map (SOM)

In this section, we describe a SOM neural architecture which can use a set of input endmembers to produce a thematic map with a classification label for each pixel. The neural model proposed in this work consists of N input neurons and M output neurons, where N is the dimensionality of the input vectors and M is the number of endmembers provided by AMEE algorithm. The network consists of two layers, with feedforward connections from the input to the output layer and a set of associated weights arranged in a matrix that will be denoted hereinafter as $\boldsymbol{W}_{MxN}$. The network procedure is given by two different stages: training and clustering. In the former step, different training patterns are presented to the network so that feedforward connections change to adapt to the information provided by training data. In the clustering step, feedforward connections project input patterns (i.e., pixel vectors to be classified) onto the feature space and the Euclidean distance is used to identify a winning neuron. The entire procedure can be summarized by the following steps:

1) *Weight initialization*. Normalized random values are used to initialize the weigth vectors: $\boldsymbol{w}_i^{(0)}$, with $i=1,2,\cdots,M$.
2) *Training*. In this work, this step is accomplished by using AMEE-generated endmember signatures.
3) *Clustering*. For each input pattern $\boldsymbol{x}$ (i.e., a spectral endmember), a winning neuron i^* is obtained at time t by using an Euclidean distance-based similarity criterion, i.e., $i^*[\boldsymbol{x}]=\min_{1\le j\le M}\|\boldsymbol{x}-\boldsymbol{w}_j\|^2$.
4) *Weight adjustment.* The winning neuron (and those neurons in the neighborhood of the winning one) adapt their weights using the following expression, where $\alpha(t)$ and $\sigma(t)$ are the learning and neighbouring functions, respectively. It should be noted that the weigths associated to i^* are modified proportionally to the learning rate.

$$\boldsymbol{w}_i^{(t+1)}=\boldsymbol{w}_i^{(t)}+\sum_{t'=t_0}^{t_{\max}}\alpha(t')\cdot\sigma(t')\cdot\left(\boldsymbol{x}-\boldsymbol{w}_i^{(t')}\right) \quad (4)$$

5) *Stopping rule.* The SOM algorithm terminates as soon as a pre-determined number of iterations, $t_{\max}$, has been accomplished.

From the above description, it is clear that the SOM algorithm is sequential in nature. As a result, parallelization strategies for this algorithm must cope with data dependencies. In the following section, we discuss parallelization strategies for both

the AMEE and SOM algorithms.

3 Parallel Implementation

The combined characteristics of the proposed morphological/neural algorithm described in the previous section introduces new considerations that need to be taken into account in order to exploit parallelism through well-defined strategies. In particular, two types of data parallelism can be exploited to optimize the proposed algorithm: spatial-domain parallelism and spectral-domain parallelism. Spatial-domain parallelism subdivides the input image into multiple blocks made up of entire pixel vectors, and assigns one or more blocks to each processing element (PE). On other hand, the spectral-domain parallel paradigm subdivides the whole multi-band data into blocks made up of contiguous spectral bands (sub-volumes), and assigns one or more sub-volumes to each PE. The latter approach breaks the spectral identity of the data because each pixel vector is split amongst several PEs. In the following, we provide a discussion on the two types of parallelism above and their impact on the individual steps (morphological/neural) of the proposed method.

3.1 Parallelization of the Morphological Algorithm

In order to describe the partitioning scheme for the morphological operations described in Section 2, we have considered two different approaches to the problem: partitioning in the spatial domain and partitioning in the spectral domain. The first option divides the hyperspectral image in multiple blocks, in a way that the pixels for each block preserve its entire spectral identity. The second option divides the original image in blocks constituted by several bands, in a way that we can preserve the spatial identity for each band but all the pixels in each block lose their spectral identity. In other words, if the partitioning scheme adopted were in the spatial domain, the information of a single pixel in the image would be scattered across several different processing units.

If we take in account the fundamental characteristics of our method, which works with all of the spectral information associated to each pixel, the selection of a partitioning scheme in the spectral domain is critical and could substantially increase the costs of communication and/or coordination between processors [5]. Besides, the overhead introduced by the communication increases with the number of processors, thus introducing problem in the load balance accomplished by the designed algorithms [5]. On other hand, the spatial information is particularly relevant in the local neighborhood around each pixel [13]. This is a reason why a partitioning scheme in the spatial domain is able to preserve most of the information required for our morphological processing. A final major point is that selection of a spatial partitioning scheme enhances load balance between different processors.

At this point, we can introduce the concept of spatial/spectral parallelizable pattern (PEEP), which is defined as the maximum amount of information that the parallel system can process without the need for additional communication and/or coordination between processors [5]. Such patterns are automatically generated by a partitioning module, as Figure 2 describes using two computing units. In the example, the partition module divides the image into two PEEPs. The values of the MEI index for two pixels of the original hyperspectral image are calculated in parallel by each of the processors, using a square-shaped SE of 3x3 pixels. Such values are then updated in a local 2-D image. At the end of the process, the PM fuses the various local images obtaining a resulting 2-D image used as a baseline to extract a final set of endmembers.

An issue of major importance in the design of SE-based

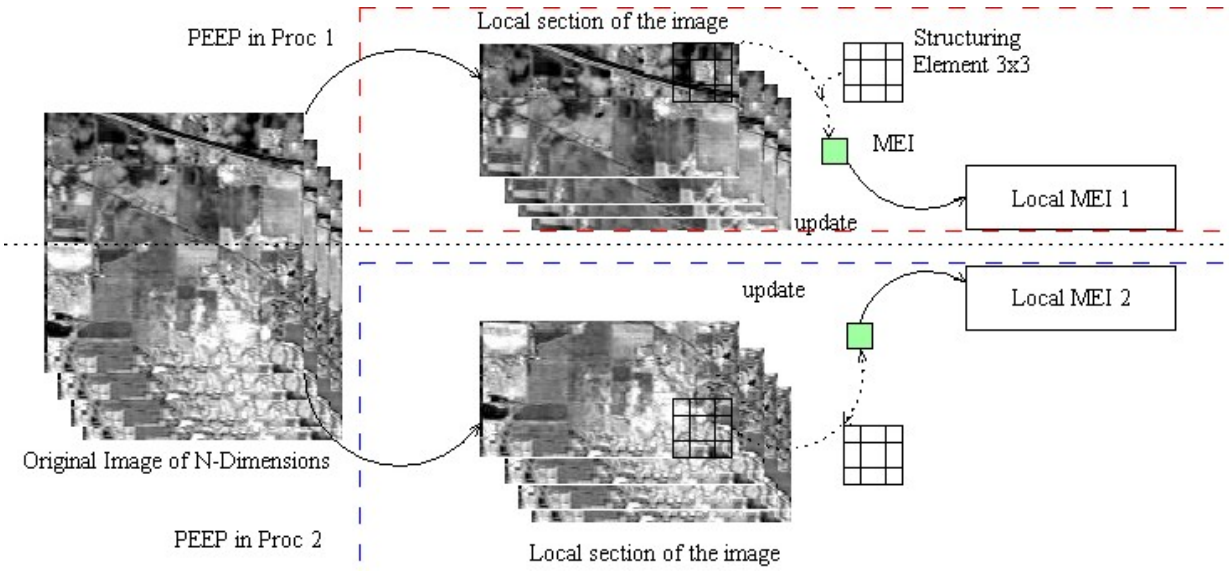


Figure 2: Concept of spatial/spectral parallelizable pattern (PEEP) and proposed partitioning scheme.

parallel image processing applications is the possibility to access pixels out of the spatial domain of the partition available in the processor. This is normally managed by a determined border-handling strategy (BHS). In our parallel implementation, two BHSs have been implemented, both of which are briefly addressed next:

1) BHS relative to the pixels out of the domain of the original image. This strategy is necessary in situations in which the SE is centered around a pixel located in the border of the input image. In this case, the BHS adopted only uses the pixels of the SE which fall inside the image domain. In our application, this strategy is similar to the mirroring technique commonly used in kernel-based image processing applications.
2) BHS relative to the pixels out of the domain of the SSPP. This strategy is applied when the pixel located in a remote processor is required in the calculation of the MEI index associated with another pixel in a given processor (see Figure 3). To resolve this issue, we aim at minimizing the communication/coordination between processors.

It should be noted that the BHS adopted in the latter situation is based on the replication of the information necessary to avoid border effects between different processors, as shown in Figure 3. According to our preliminary experiments, the cost of processing the information resulting from the policy above is sensibly inferior to dealing with the overhead introduced by communication among different processors if no redundant information is introduced in the system.

Given the characteristics of the implementation proposed in Section 2, which relies on the utilization of an SE of 3x3 pixels iteratively, the number of redundant pixels R introduced in the processing of a hyperspectral image is given by

$$R = 2 \times \left[\left(2^{\lceil \frac{\log_2 N}{2} \rceil} \right) - 1 \right] \times I_F + 2 \times \left[\left(2^{\lfloor \frac{\log_2 N}{2} \rfloor} \right) - 1 \right] \times I_C, \quad (5)$$

where N is the number of processors, I_F is the number of rows in the original image and I_C is the number of columns in the original image. For example, in order to process an AVIRIS image of 512x512 pixels using 16 processors, the total number of redundant pixels is $R = 2 \times [(2^2) - 1] \times 512 + 2 \times [(2^2) - 1] \times 512 = 6144$. If we assume that each pixel has 224 spectral values, each of them coded using two bytes, the total amount if redundant information introduced in the system is 2,625 Mb (6144 × 224 × 2) which, compared with the total size of the original image in bytes (about 114 Mb), can be considered insignificant. As noted above, the amount of redundant information is below 2.5 percent of the total information present in the original image. It is important to point out that the amount of redundant information grows as the number of processors increases, a fact that introduces a limit to the performance of the parallel code which is directly related to the problem of having more redundant information than pixels to process inside a certain SE.

3.2. Parallelization of the Neural Algorithm

A straightforward approach to parallelization of the neural algorithm is to simply replicate the whole neural network architecture, which is a feasible approach due to the random nature of the initial weights of the network. However, this option results in the need for very complex rules of reduction, and integrity hazards. Taking into account our previous studies [9] and considering the relatively small size of the training set, we can state that the overhead of the neural network is mainly located in the training process (in the form of Euclidean distance calculations and adjustment of weight factors). This fact makes partitioning of the neural network (weight factors matrix) an appealing solution in order to reduce the processing load and time. Again, two main alternatives can be adopted to carry out such partitioning: (1) Division by input neurons (endmembers/training patterns); or (2) Division by output neurons (class prototypes). The two options are graphically illustrated in Figure 4.

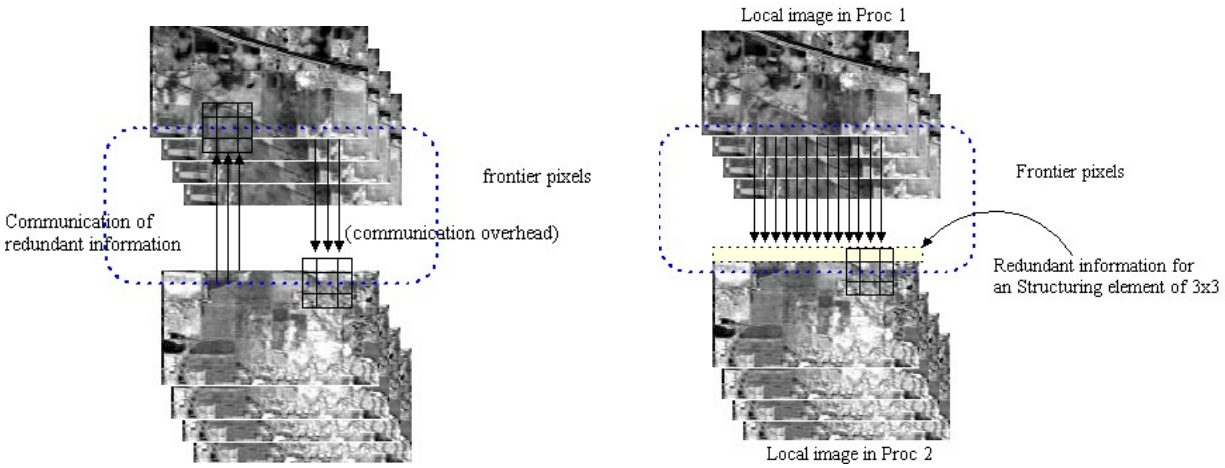


Figure 3: Problem of accessing pixels out of the SSPP domain (left) and BHS relative to the pixels out of the PEEP domain (right).

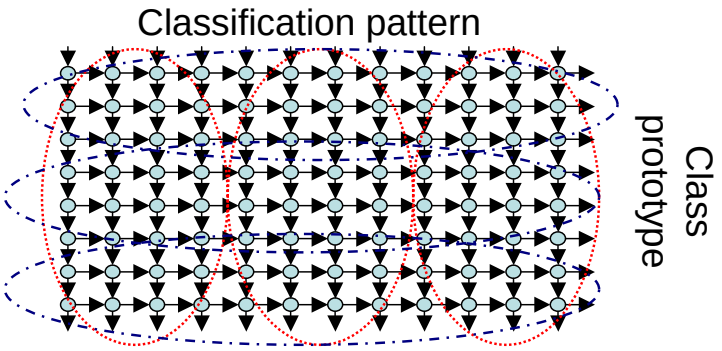


Figure 4: Partitioning options for the considered neural algorithm

It should be noted that, in the latter case, the parallelization strategy is very simple. Quite opposite, when the former approach is adopted, there is a need to communicate both calculations and intermediate results among different processors. This introduces an overhead in communications that may significantly slow down the algorithm: according to our preliminary experiments, this option could even give worst results than those found by the sequential version of the algorithm. On the other hand, the partitioning scheme based on dividing by class prototypes only introduces a minor communication overhead, i.e., that created by the need to obtain the winner class. To do so, a protocol similar to logarithmic synchronization barriers is adopted. Also, there is a need to introduce a broadcast/all-reduce protocol to obtain the class prototype through local minimum calculations in a batch SOM processing way. The winner neuron for each pattern needs to be tailored, and subsequent modifications for the weighting factor need to be stored for later addition/subtraction. This approach also allows directly obtaining of the winner neuron at each iteration without the need for any further calculations. It also facilitates a more pleasingly parallel solution, aimed at taking full advantage of the processing power available in the considered parallel architecture while minimizing the communication overhead.

At this point, we must emphasize that the proposed scheme still introduces the need to replicate calculations in order to reduce communications. However, the amount of replicated data is limited to the presence of the complete training pattern set at each processor, along with administrative information, i.e., which processor holds the winner neuron, which processor holds the neurons in the neighborhood of the winner neuron, etc. Such administrative information can be used to reduce the communication overhead even further. For instance, using the above information we consider two implementations of the neighborhood modification function $\sigma(t')$, where the first one is applied when a node is in the neighborhood of the winner neuron and the second is considered when the node is outside the domain of that processor. To assess the integrity of the considered neighborhood function, a look-up table is locally created at each processor so that the value of $\sigma(t)$ is stored for every pair of neurons. While in the present work the function selected is gaussian, i.e., $\sigma(t) = e^{-\left(\left|i^* - i\right|/t\right)}$, other neighborhood functions may also be considered [9]. In any regard, we emphasize that when the neighborhood function is applied to the processor that holds the winner neuron, it is used in a traditional way. On the contrary, when the function is applied to other processors, a modified version is implemented to average the distances with all possible winners. There are two main reasons for this decision: (1) First and foremost, this approach significantly reduces the amount of communications; and (2) It represents a more meaningful and robust neighborhood function. As a final major remark, we must point out that our MPI-based implementation makes use of blocking primities, thus ensuring that all processors are synchronized and preventing integrity problems in the calculations with the matrix of weights $\boldsymbol{W}_{\mathrm{MxN}}$.

3.3 Summary of Operations

The parallel implementation described in the above subsections is based on a partitioning scheme in the spatial domain in which one of the processors acts as the master node in charge of the I/O operations. The master node implements a spatial partitioning policy that enhances load balance between processors. The partitioner has been implemented so that it automatically determines the optimum size for the PEEPs to be distributed between the different processors. In the parallel implementation, the master node sends to each processor a portion of the original image, or weight matrix depending on the stage, using the `MPI_Send` primitive. Each processor works locally with its corresponding portion. Once it has finished the local processing, each processor sends the results back to the master (which receives the portions using `MPI_rcvd` primitive) to get the local results. Finally, the master compounds the partial results and carries out the process of selecting the final endmembers using the information provided by each of the processors. By distributing data evenly among the processors, load balance is achieved. Also, the utilization of the concept of PEEP allows us to greatly minimize interprocessor communication overhead. To conclude this section, we emphasize that the proposed parallel algorithm fully exploits the underlying parallelism inherent in image processing methods [14], i.e. it minimizes the communication between processors. The parallel code described in this section is portable to any distributed system, provided that the memory available to each processing is large enough to store the respective PEEP or partial weight matrix. Performance data for the parallel algorithm are given in section 4.

4 Experimental Results

This section describes the performance of the parallel implementations outlined in Section 3 in terms of their computational efficiency (speedup) compared with the serial version of the code, the scalability of the parallel code, and also in terms of its accuracy in the context of automated

classification of hyperspectral images. In a first subsection we describe the parallel computers used in the study. Then, we discuss the obtained results in the analysis of a well-known AVIRIS image.

4.1 Parallel Computers

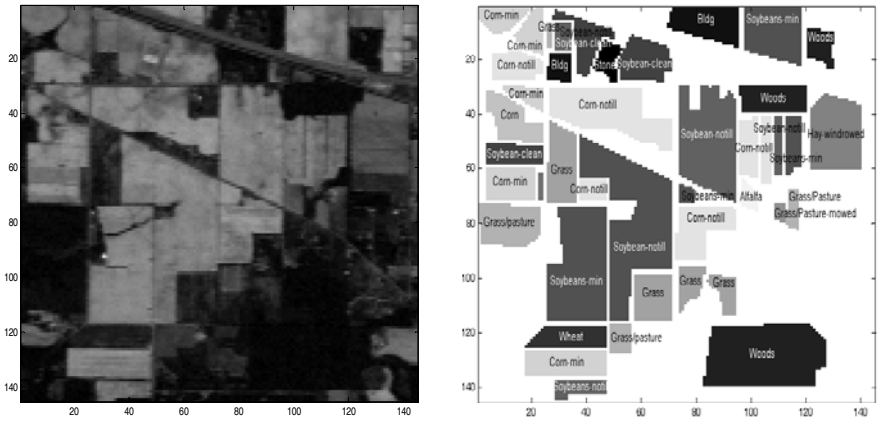
Two parallel computers have been used to evaluate the computational performance of the morphological algorithm proposed. The first is an SGI Origin 2000 multicomputer located in the European Center of Parallelism of Barcelona. It is a system with a MIMD cc-NUMA distributed memory composed of 64 MIPS R10000 processors (each of them with 4 Mb of cache and 12 Gb of main memory) connected through an intercommunication network of 1.2 Gbps. The theoretical peak performance of the system is 32 Gflops. The operating system used during our experiments was Irix 5.6, and the software was compiled using mpicc available from MIPSpro 7.3.1.2 suite. This system is no longer available, and was mainly used in this work to provide preliminary tests of the parallel algorithms as separate modules. The second parallel computer used in the study is a Beowulf type cluster named Thunderhead located at the Applied Information Sciences Branch of NASA/GSFC. This system consists of 256 nodes, each of them with two 2.4 GHz Intel Xeon processors. Each node has 1 Gb of local memory. The full system has a total of 80 Gb of distributed memory. The communication network is Myrinet at 2 GHz (optical fibre). The maximum theorical performance of the system is 2457.6 Gflops. The operating system in Thunderhead is Linux Red Hat 8.0, and MPICH is the communication library. Thunderhead was used in this work to test the combined morphological/neural classification method developed in Section 3.

4.2 Results and Discussion

To empirically investigate the scaling properties of the parallel algorithm, we have used a hyperspectral image obtained by the AVIRIS sensor in June 1992 over a small area (145 lines by 145 samples and 220 spectral bands) gathered over the Indian Pines test site in Northwestern Indiana, a mixed agricultural/forested area, early in the growing season. The data set represents a very challenging classification problem with due to the presence of mixed pixels. Figure 5(a) shows the spectral band at 587 nm of the original scene, and Figure 5(b) shows the ground-truth map. The image is available, along with ground truth information, from http://dynamo.ecn.purdue.edu/~biehl/Multispec/.

Using the information provided by ground truth, we have analyzed the cost-performance accuracy of the proposed morphological. Our classification scheme consisted of the following steps: 1) Endmember extraction via morphological operations, 2) Fully constrained linear unmixing using the extracted endmembers [1], and 3) Classification of each pixel as belonging to a class given by the most abundant endmember in the pixel. The following parameters were considered: I_{MAX} was set to 1, 3, 5 and 7 iterations, respectively. B is a 3x3-pixel structuring element of fixed size, and p, the maximum number of endmembers to be detected that was set to $p = 16$ after removing the image background and calculating the intrinsic dimensionality of the data using the Harsanyi-Farrand-Chang (HFC) method in [3]. Table 1 quantitatively shows overall and individual classification accuracy scores produced by the parallel algorithm with different values of I_{MAX}. As shown in Table 1, setting $I_{MAX} = 7$ results in an overall accuracy of more than 90 percent and very high classification scores for all the individual ground-truth classes. We have experimentally tested that classification accuracies do not significantly improve when $I_{MAX} > 7$.

In order to illustrate the efficiency of the parallel AMEE algorithm, Table 2 shows execution times in seconds of the proposed algorithm with the AVIP92 scene for several combinations of number of iterations and number of processors, along with speedup factors on the SGI Origin 2000



(a) (b)

Figure 5: (a) Spectral band at 587 nm wavelength of an AVIRIS scene comprising agricultural features at Indian Pines test site, Indiana. (b) Ground-truth map with 16 mutually-exclusive land-cover classes

multicomputer. On the other hand, Table 3 summarizes similar experiments conducted on the Thunderhead Beowulf cluster. It should be noted that a maximum number of 8 and 256 processors were respectively utilized in the SGI Origin 2000 and Thunderhead, respectively, due to system availability at the time of the experiments. From results in Table 2, we can conclude that the proposed parallel version of the AMEE algorithm achieves significant speedups when compared to the serial implementation in the two parallel computers. Also, the measured speedups tend to be higher for large values of I_{MAX}, a fact that reveals that the proposed scheme scales better when the number of morphological operations to be accomplished is very high. In this case, the proposed algorithm is able to obtain high classification accuracies in near real-time. In order to analyze the scalability of the parallel code, Figure 6 plots the speedup factors as a function of the number of available processors N at the SGI Origin 2000 computer.

Table 1: Overall and individual classification accuracies of the proposed algorithm using different number of iterations, I_{MAX}

Class (number of pixels)	$I_{MAX}=1$	$I_{MAX}=3$	$I_{MAX}=5$	$I_{MAX}=7$
Alfalfa (54)	41.23	55.55	61.11	75.92
Corn-notill (1434)	84.65	79.91	83.82	89.47
Corn-min (834)	66.54	69.06	75.42	84.53
Corn (234)	40.29	64.10	70.51	82.05
Grass/Pasture (497)	65.99	73.64	79.67	85.71
Grass (747)	67.20	94.11	95.58	97.45
Grass/pasture-mowed (26)	45.21	84.61	84.61	96.15
Hay-windrowed (489)	42.32	99.59	99.59	99.79
Oats (20)	40.78	75.00	85.00	80.00
Soybeans-notill (968)	63.43	71.17	77.06	85.95
Soybeans-min (2468)	75.77	77.39	81.56	89.30
Soybean-clean (614)	70.24	72.80	80.13	87.78
Wheat (212)	52.36	99.52	99.52	100.00
Woods (1294)	87.17	88.79	91.19	95.13
Bldg (380)	81.05	82.63	86.05	91.31
Stone-steel towers (95)	70.52	71.57	75.78	90.52
Overall (10366)	66.16	79.92	83.98	90.24

Table 2: Execution times in seconds of the AMEE algorithm at the SGI Origin 2000 multi-computer for several combinations of number of iterations, I_{MAX}, and number of processors, N

N	$I_{MAX}=1$	$I_{MAX}=3$	$I_{MAX}=5$	$I_{MAX}=7$
1	372	1066	1809	2476
2	182	522	864	1178
4	89	252	429	569
8	64	143	338	293

Table 3: Execution times in seconds of the AMEE algorithm at the Thunderhead Beowulf cluster for several combinations of number of iterations, I_{MAX}, and number of processors, N.

N	$I_{MAX}=1$	$I_{MAX}=3$	$I_{MAX}=5$	$I_{MAX}=7$
1	311	947	1528	1925
4	124	321	557	685
16	45	95	144	156
36	26	46	61	71
64	19	29	41	43
100	12	20	26	29
144	9	15	20	23
196	6	11	17	20
256	4	10	14	18

It should be noted that the speedup factors in Figure 6 were calculated as follows: if we approximate the real time required to complete a task on N parallel processors, $T(N)$, by

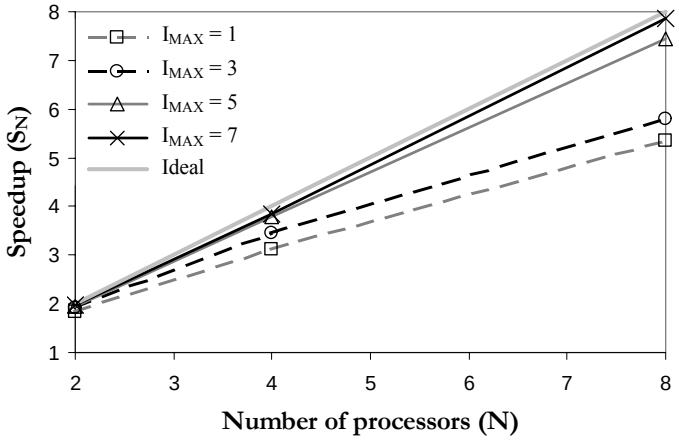


Figure 6: Parallel performance of the parallel AMEE algorithm in the SGI Origin 2000 computer

$T(N) = A_N + \frac{B_N}{K}$, where A_N is the sequential (non-parallelizable) portion of the computation and B_N is the parallel portion. In the parallel code, A_N corresponds to the sequence of operations implemented by the partitioning module in the case of the AMEE algorithm (in the case of the parallel SOM, the sequential time corresponds to the generation of random weight values). On the other hand, B_N corresponds to the selection of endmembers (AMEE) and the training process (SOM). Then, we can define the speedup for N processors, S_N, as follows: $S_N = \frac{T(1)}{T(N)} \approx \frac{A_N + B_N}{A_N + (B_N / N)}$, where $T(1)$ denotes single processor time. The relationship above is generally known as Amdahl's Law. It is obvious from this expression that the speedup of a parallel algorithm does not continue to increase with increasing the number of processors. The reason is that the sequential portion A_N is proportionally more important as the number of processors increase and, thus, the performance of the parallelization is degraded for a large number of processors. Since only the parallel portion B_N scales with the time required to complete the calculation and the serial component remains constant, there is a theoretical limit for the maximum parallel speedup achievable for N processors, which is given by: $S_\infty^N = \lim_{N\to\infty} S_N = \frac{A_N + B_N}{A_N} = 1 + \frac{B_N}{A_N}$. Once the speedup values are obtained, we may also calculate the parallel efficiency as $E_F = \frac{T(N)}{N}$.

With the above performance metrics in mind, Figure 7 shows the speedups and parallel efficiencies obtained by the parallel implementations of both AMEE and SOM algorithms in Thunderhead cluster (reported in separate fashion to better evaluate the specific aspects of each parallel algorithm). As shown by Figure 7(a), the morphological algorithm scales reasonably well on Thunderhead. This is because it takes andvantage of some of the intrinsic characteristics of window-moving image processing algorithms, such as spatial and temporal data locality that result in cache reuse. The best speedup compromise for $I = 7$ algorithm iterations was achieved for $N = 16$ processors, with $S_{16} = 12.33$ and $E_{16} = 0.77$. The degradation in parallel efficiency as the number of processors is increased [see Figure 7(b)] is likely due to the effect of redundant computations. On the other hand, Figure 7(c) reveals that, although parallelization of the

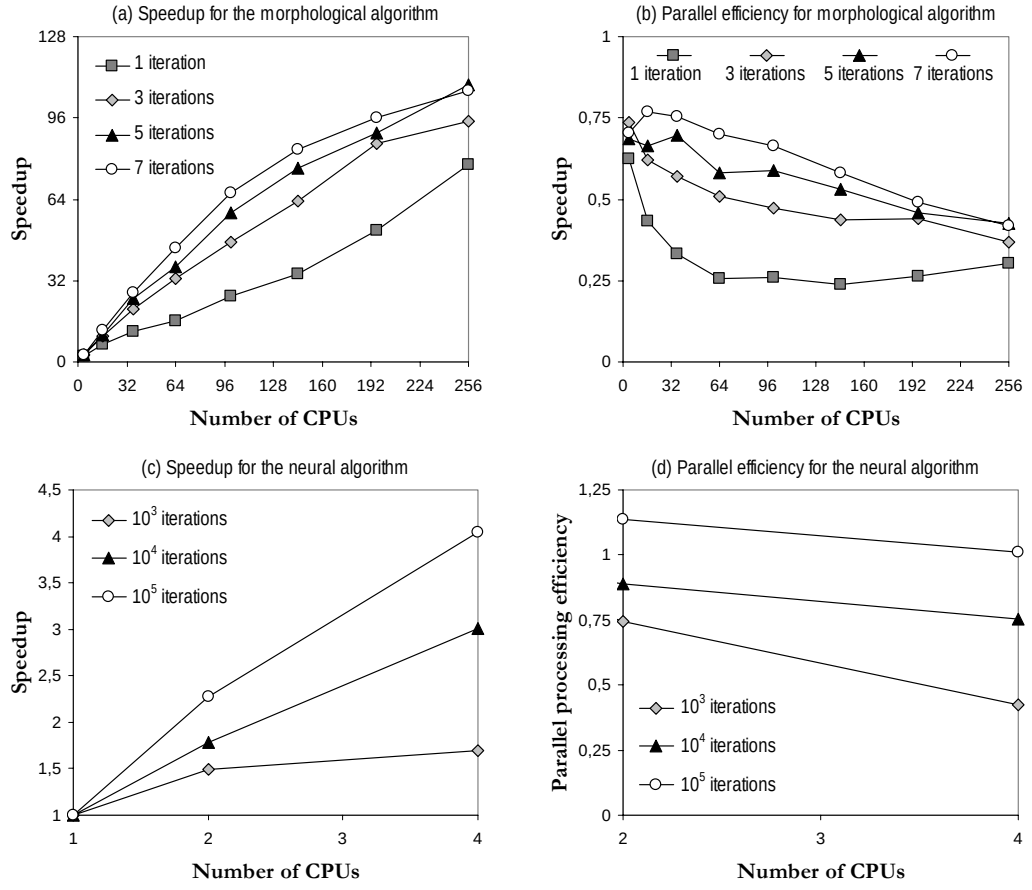


Figure 7: Speedup/parallel efficiency achieved by morphological/neural algorithms on Thunderhead

neural algorithm is more complicated *a priori* due the expected impact of communications, the parallel neural code also scales relatively well (for a reduced number of processors). It should be noted that the cost of communications in the parallel neural algorithm cannot be reduced by introducing redundant computations, as it was the case in the morphological algorithm. Even though the amount of data to be exchanged is minimized by the proposed parallel neural strategy, we still had to deal with the size of the minimum transfer unit (MTU) of the communication network, a parameter that is not easily adjustable in the Thunderhead system. In future developments, we are planning on incorporating techniques able to automatically adjust the size of the MTU according to the properties of the input data. For instance, the domain of a single batch-mode iteration could be expanded to several network epochs (with all training patterns involved at each one) instead of just one epoch as in the current implementation. This could lead to much better data compaction inside the considered MTU. Also, results in Figure 7(d) reveal that the parallel efficiency achieved for large training sets is significantly higher than that found for smaller training sets. This is because computations clearly dominate communications in this case, thus greatly enhancing the granularity of the parallel computation. As one would expect, the use of large training sets also results in much higher classification accuracies by the SOM neural network. Although only results with 4 processors are reported in this work for the combined morphological/neural method, we also observed that increasing the number of processors introduced fluctuations in the achieved speedups with significant drops in parallel efficiency. This is due in part to the scheduling policies implemented in the Thunderhead cluster, which tend to assign high priority to jobs that require a very large number of processors. Even in spite of the above limitations, our measured speedups reveal slight superlinear scaling effects in some cases, probably due to cache reuse (e.g., when 10^5 training iterations were considered, values of $E_2 = 1.135$ and $E_4 = 1.01$ were measured). This reveals that cache spatial and temporal locality could be partially used to overcome the limitations imposed by excessive communications. The above results also lead us to believe that the best configuration for the parallel SOM algorithm is likely to be achieved when most neural network partitions fit completely in the local processor caches. Further experimentation, however, is highly desirable in order to adapt the parallel properties of the neural algorithm to those observed in the morphological algorithm. In particular, there is a need to balance the combined computing power achieved by the pool of processors employed by the morphological algorithm and those used by the neural algorithm in the same algorithm run. This feature brings out new exciting future perspectives, such as the possibility to launch multiple neural-based classifiers in parallel. Such multiple classifier-based processing framework represents a completely novel data analysis paradigm in hyperspectral imaging, which previously looked too computationally complex to be developed in practical applications.

5 Conclusions and Future Work

The aim of this paper has been the parallel implementation on high performance computers of an innovative morphological/neural technique for unsupervised classification of high-dimensional remotely sensed data sets, with the purpose of obtaining processing results in valid response times and with adequate reliability for the remote sensing environment where it is intended to be applied. It has been shown and proven that parallel computing at the massively parallelism level, supported by message passing, provides a unique framework to accomplish the above goals. For this purpose, computing systems made up of arrays of commercial off-the-shelf computing hardware are a cost-effective way of exploiting this sort of parallelism in remote sensing applications. Specifically, the proposed MPI-based parallel implementation minimizes inter-processor communication overhead. and can be ported to any type of distributed memory system. In particular, it can be easily ported to a Beowulf cluster of PCs, an architecture that has gained popularity in the last few years due to the chance of building a "high performance system" at a reasonable cost. Experimental results in this paper suggest that our parallel algorithm provides adequate results in both the quality of the solutions and the time to obtain them, in particular, when it is implemented on low-cost Beowulf clusters. Further, the proposed parallel framework offers an unprecedented opportunity to explore methodologies in other fields that previously looked to be too computationally intensive for practical applications due to the inmense files common to remote sensing problems. Combining this readily available computational power with the new sensor instruments may introduce major changes in the systems currently used by NASA and other agencies for exploiting Earth and planetary remotely sensed data.

References

[1] T. Achalakul and S. Taylor, "A Distributed Spectral-Screening PCT Algorithm," *Journal of Parallel and Distributed Computing*, 63:373-384, 2003.

[2] G. Aloisio and M. Cafaro, "A Dynamic Earth Observation System," *Parallel Computing*, 29:1357-1362, 2003.

[3] C.-I Chang, *Hyperspectral Imaging: Spectral Detection and Classification*, Kluwer Academic/Plenum Publishers, New York, 2003.

[4] M. K. Dhodhi, J. A. Saghri, I. Ahmad, and R. Ul-Mustafa, "D-ISODATA: A Distributed Algorithm for Unsupervised Classification of Remotely Sensed Data on Network of Workstations," *Journal of Parallel and Distributed Computing*, 59:280-301, 1999.

[5] R. O. Green M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis, M. R. Olah, and O. Wiliams, "Imaging Spectroscopy and the Airborne Vsible/Infrared Imaging Spectrometer (AVIRIS)," *Remote Sensing of Environment*, 65:227–248, 1998.

[6] K. A. Hawick, P. D. Coddington, and H. A. James, "Distributed Frameworks and Parallel Algorithms for Processing Large-Scale Geographic Data," *Parallel Computing*, 29:1297-1333, 2003.

[7] K. Itoh, "Massively-Parallel Fourier-Transform Spectral Imaging and Hyperspectral Image Processing," *Optics & Laser Technology*, 25:202, 1993.

[8] T. Kohonen, *Self-Organizing Map, 2nd Ed*, Springer-Verlag, Berlin, Heidelberg, 1997.

[9] P. Martinez, P. L. Aguilar, R. Perez and A. Plaza, "Systolic SOM Neural Network for Hyperspectral Image Classification," D. Zhang and S. K. Pal. (Eds.), *Neural Networks and Systolic Array Design*, World Scientific: Singapore, pp. 26-43, 2002.

[10] J. Le Moigne, W. J. Campbell, and R. F. Cromp, "Automated Parallel Image Registration Based on Correlation of Wavelet Features," *IEEE Trans. Geosci. Remote Sensing*, 40:1849–1864, 2002.

[11] A. Plaza, P. Martínez, R. Pérez, and J. Plaza, "A New Approach for Mixed Pixel Classification in Hyperspectral Imagery Based on Extended Morphological Profiles," *Pattern Recognition*, 37:1097-1116, 2004.

[13] A. Plaza, P. Martinez, R. Pérez and J. Plaza, "Spatial/Spectral Endmember Extraction by Multidimensional Morphological Operations," *IEEE Transactions on Geoscience and Remote Sensing*, 40:2025-2041, 2002.

[12] A. Plaza, P. Martínez, R. Pérez, and J. Plaza, "A Quantitative and Comparative Analysis of Endmember Extraction Algorithms from Hyperspectral Data," *IEEE Transactions on Geoscience and Remote Sensing*, 42:650-663, 2004.

[14] F. J. Seinstra, D. Koelma, and J. M. Geusebroek, "A Software Architecture for Transparent Parallel Image Processing," *Parallel computing*, 28:967-923, 2002.

[15] D. Valencia, A. Plaza, P. Martinez and J. Plaza, "On the use of Cluster Computing Architectures for Implementation of Hyperspectral Analysis Algorithms," *Proceedings of the 10th IEEE Symposium on Computers and Communications* (ISCC), Cartagena, Spain, pp. 995-1000, 2005.

[16] P. Wang, K. Y. Liu, T. Cwik, and R.O. Green, "MODTRAN on Supercomputers and Parallel Computers," *Parallel Computing*, 28:53–64, 2002.

David Valencia (photo not available) is a Research Associate at the University of Extremadura, Spain, where he is currently pursuing his M.Sc. degree in Computer Science. His research interests are in the development of parallel implementations of algorithms for high-dimensional data analysis, with particular emphasis on commodity cluster-based (homogeneous and heterogeneous) systems and hardware-based architectures, including systolic arrays, field programmable gate arrays and graphic processing units. He is also involved in the design, testing and implementation of large-scale distributed heterogeneous computing platforms.

Antonio Plaza (photo not available) received his Ph.D. degree in Computer Science from the University of Extremadura, Spain, in 2002, where he is currently an Associate Professor with the Computer Science Department. He has also been Visiting Researcher with the University of Maryland, NASA Goddard Space Flight Center and Jet Propulsion Laboratory. His main research interests include the development and efficient implementation of high-dimensional data algorithms on parallel homogeneous and heterogeneous computing systems and hardware-based computer architectures. He has authored or co-authored more than one hundred publications including journal papers, book chapters and peer-reviewed conference proceedings, and currently serves as regular manuscript reviewer for more than 15 highly cited journals in the areas of parallel and distributed computing, computer architectures, pattern recognition, image processing and remote sensing. He is editing a book on "High- Performance Computing in Remote Sensing" (with Prof. Chein-I Chang) for Chapman & Hall/CRC Press.

Pablo Martínez (photo not available) is a Professor of Computer Science at the University of Extremadura, Spain, since 1985. He is the Head Scientist of the Neural Networks and Signal Processing Group (GRNPS). He has held Visiting Researcher positions at the NASA Goddard Space Flight Center and the Department of Electrical Engineering, University of Maryland, College Park, MD. His research interests are in remote sensing, digital image analysis, parallel and distributed computing, hardware-based architectures, operating systems management and configuration, and neural network-based pattern recognition.

Javier Plaza (photo not available) received his M.Sc. degree in Computer Science from the University of Extremadura, Spain, in 2002, where he is currently an Assistant Professor. His current research work is focused on the development of efficient implementations of neural network-based algorithms for analysis and classification of hyperspectral scenes. He is also involved in the design and configuration of homogeneous and fully heterogeneous parallel computing architectures for high-performance scientific applications. Other major research interests include telecommunications, networking and configuration and training of neural network architectures for specific applications.