Chapter 8

Parallel Wildland Fire Monitoring and Tracking Using Remotely Sensed Data

David Valencia,
University of Extremadura, Spain

Pablo Martínez,
University of Extremadura, Spain

Antonio Plaza,
University of Extremadura, Spain

Javier Plaza,
University of Extremadura, Spain

Contents

Predicting the potential behavior and effects of wildland fires using remote sensing technology is a long-awaited goal. The role of high-performance computing in this task is essential since fire phenomena often require a response in (near) real-time. Several studies have focused on the potential of hyperspectral imaging as a baseline technology to detect and monitor wildland fires by taking advantage of the rich spectral information provided by imaging spectrometers. The propagation of fires is a very complex process that calls for the integrated use of advanced processing algorithms and mathematical models in order to explain and further characterize the process. In this chapter, we describe several advanced hyperspectral data processing algorithms that are shown to be useful in the task of detecting/tracking wildland fires and further study how such algorithms can be integrated with mathematical models, with the ultimate goal of designing an integrated system for surveillance and monitoring of fires.

8.1 Introduction

Many efforts have been conducted by international organizations to deal with natural and human-induced disasters through the use of remote sensing technology. Many of them are focused on post-evaluation and management of the disaster as a way to improve future evaluation and prediction. Several missions operated by international agencies are designed to produce a great amount of image data, which can be processed to evaluate and track these disasters, but this approach introduces strong computational requirements that are always challenging in terms of budget and, in some cases, lack of knowledge of the remote sensing community on high-performance computing solutions.

For instance, wildland fires represent one of the most important sources of biodiversity loss on our planet and introduce important requirements from the viewpoint of algorithm design and high-performance implementations. This is a general

problem that causes important environmental risks. In particular, the importance of preserving forests is enormous since they are multifunctional in their outcome, from economical (resources), social (recreational), and environmental perspectives (protection against atmospheric contamination and wildfires, climate control, mitigation of climate change, and water/soil preservation).

In this chapter, we evaluate different possibilities to approach the problem of monitoring and tracking wildland fires using remotely sensed hyperspectral imagery. We also outline the design of a system to monitor and track wildland fires using image data sets produced by both airborne and satellite hyperspectral sensors. The chapter is organized as follows:

- Section 8.2 introduces the requirements for real-time response on hazards and disasters, using wildland fires as a potential case study.
- Section 8.3 describes the model adopted in this chapter for characterization of wildland fires. Ideally, this model should be integrated with advanced data processing algorithms to produce advanced fire characterization products.
- Section 8.4 describes a collection of hyperspectral data processing algorithms that may be used for detecting and monitoring wildland fires. Different approaches are evaluated in this section, including detection, classification, segmentation, and spectral unmixing.
- Section 8.5 details parallel implementations of some of the proposed algorithms, including morphological techniques and neural networks for classification and spectral mixture analysis of hyperspectral data sets.
- Section 8.6 outlines a high-performance system that integrates the above-mentioned parallel algorithms with mathematical models for potential prevention and response to wildland fires. The proposed system integrates several computer architectures, such as homogeneous and heterogeneous networks of computers including grid environments, and specialized hardware platforms such as those based on programmable hardware.
- Section 8.7 provides experimental results to evaluate the accuracy and parallel efficiency of the proposed parallel algorithms. Due to the lack of hyperspectral images of fires with reliable ground-truth, we use standard hyperspectral data sets collected in the framework of other applications to provide our experimental assessment.

8.2 Real-Time Response to Hazards and Disasters: Wildland Fires as a Case Study

Many remote sensing missions have been defined with the ultimate goal of monitoring natural disasters, e.g., detection of red tides using Envisat's Medium Resolution Imaging Spectrometer Instrument (MERIS) [16]. In the next decade, and thanks to the

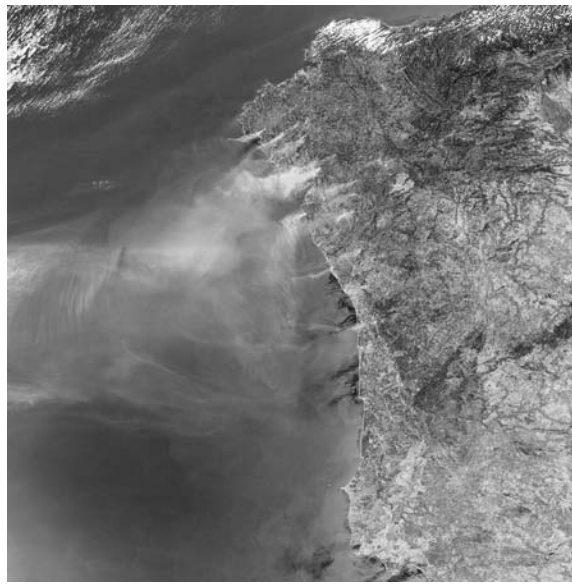

Figure 8.1 MERIS hyperspectral image of the fires that took place in the summer of 2005 in Spain and Portugal.

expected improvements in the spatial and spectral resolutions of hyperspectral imaging instruments, it is anticipated that several missions will be focused on the detection, monitoring, and tracking of hazards. For instance, Figure 8.1 shows a MERIS image obtained during the summer of 2005 over Spain and Portugal. The figure reveals the fires that occurred in the area, which caused the loss of many forest areas [17]. The main disadvantage of monitoring fires using the MERIS instrument is that the revisit time of MERIS is 3 days, thus limiting its exploitation for active fire tracking.

Fire is an important and recurrent phenomenon in all forested and non-forested regions of the Earth. In some ecosystems, fire plays an ecologically significant role in biochemical cycles and disturbance dynamics. In other ecosystems, fire may lead to the destruction of forests or to long-term site degradation. As a consequence of demographic and land use changes, and the cumulative effects of anthropogenic disturbances, many forest types adapted to fire are becoming more vulnerable to high-intensity wildfires. For several years the number of fires in European forests has been increasing [17]. Forest fires are usually a result of the simultaneous existence of several phenomena: droughts, the effect of air pollution (decline and decay of trees, the formation of loose canopy and lush growth of grasses, all resulting in large amounts of flammable material), pyromaniacs, *illegal* selective burning of areas for construction, and so on.

Decades of research to understand and quantify fire, together with revolutionary advances in computer technology and computational algorithms, have lead to the development of sophisticated mathematical models that can predict and visualize growth

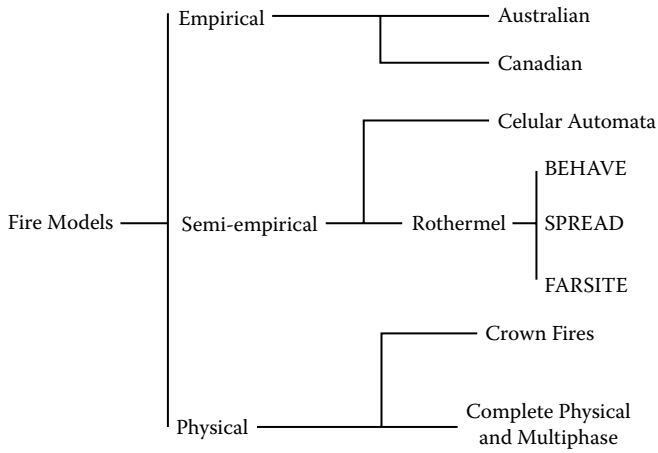

Figure 8.2 Classification of fire spread models.

and spread of fire under a variety of conditions. However, these models generally require substantial computational and data resources. Numerous fire spread models have been proposed following several methods (see Figure 8.2) that can be grouped into:

Empirical (or statistical). Statistical, stochastic, also called empirical models are predicting more probable fire behavior from average conditions and accumulated knowledge obtained from laboratory and outdoor experimental fires, or historical fires. There are two empirical models widely in use, Australian and Canadian [18, 19]. These models make no attempt to include any physical mechanisms for fire spread; they are purely statistical descriptions of test fires of such spreads. For example, the Canadian Forest Service has integrated 25 years of researching experimental and real scenario fires to develop the Canadian Forest Fire Behavior Prediction System, which is now available in book and electronic forms. It consists of 89 formulas developed empirically.

Semi-empirical (semi-physical or laboratory models). Semi-empirical models are based on a global energy balance and on the assumption that the energy transferred to the unburned fuel is proportional to the energy released by the combustion of the fuel. Several terms of the model must be fitted from laboratory fire experimental results and field campaign data [1]. The simplicity of this approach has allowed the development of operational tools.

Physical (theoretical or analytical). Models based on physical principles, have the potential to accurately predict the parameters of interest over a broader range of input variables than empirically based models. Physics based models can also provide the basic information needed for the proper description of physical processes (i.e., fluid flow, heat transfer, and chemical kinetics). But physics-based models [20] imply that the developer has an adequate understanding of the underlying physical relations sufficient to achieve the desired objectives,

that the underlying physics can be represented mathematically in a manner that permits numerical solution. Improved models are needed for increased accuracy in fire behavior prediction, fire danger rating calculations, and fuel hazard assessment. Models with the goal to predict 3-dimensional fire shapes are often referred to as crown fire models.

The amount of living vegetation, and its moisture content, have a strong effect on the propagation and severity of wildland fires, and they are key components for all models types mentioned above. The direct observation of vegetation greenness is, therefore, essential for any fire-spread model (fuel bed component of the models), along with other information that can be retrieved using remote sensing. Current assessment of living vegetation moisture relies on various methods of manual sampling. While these measurements are quite accurate, they are difficult to obtain over broad areas, so they fail to portray changes in the pattern of vegetation greenness and moisture across the landscape.

The current polar orbiting satellites provide the potential for delivering information about the greenness and moisture of vegetation, along with other parameters for fire management. Some initiatives focused on the usage of constellations of satellites, as the FUEGO mission of the European Space Agency (http://http://virtual.vtt.fi/space/firealert/space_segment_detailed.html), in order to collect data about fires over a particular site in very small intervals of time. This will provide the scientists investigating forest fires with a great amount of data for their models, but again this wealth of information will be very difficult to manage and compute without the aid of high-performance computing.

In this chapter, we outline the requirements and preliminary design of a system for forest fire monitoring and tracking, combining advanced hyperspectral data processing techniques (implemented on high-performance computing platforms) with a well-known semi-empirical mathematical model for forest fire spread. Although the proposed system is aimed at wildland fire characterization, experimental results are given using data sets in other application domains for preliminary validation purposes.

8.3 Mathematical Model for Wildland Fire Characterization

In the past years, several models and systems to evaluate and predict fire risks have been made, as was pointed out before. The most interesting ones are the group of models that are based on the use of Rothermel's equations [1]. The main equations provided by the model allow the calculation of the rate of spread (ROS) and fire intensity I_b as follows:

$$\text{ROS} = \frac{I_R \xi (1 + \phi_w + \phi_s)}{\rho_b \varepsilon Q_{ig}} \tag{8.1}$$

where ROS is the heading fire steady state spread rate ($\frac{m}{min}$), I_R is the reaction intensity ($\frac{kJ}{min \cdot m^2}$), ξ is the propagating flux ratio, ϕ_w is the wind factor (dimensionless), ϕ_s is

the slope factor (dimensionless), ρ_b is the ovendry bulk density ($\frac{kg}{m^3}$), ε is the effective heating number (dimensionless), and Q_{ig} is the heat of pre-ignition ($\frac{kJ}{kg}$).

$$I_b = h \cdot w \cdot \frac{R}{60} \tag{8.2}$$

where I_b is the fire line intensity (kW/m) that describes the rate of energy release per unit length in the fire front, h is the heat yielded by the fuel (kJ/kg) or the total heat less the energy required for vaporizing moisture, w is the weight of fuel per area (kg/m^2) burned in the flaming front, and $R/60$ is the fire spread rate converted to units of m/sec.

Rothermel's model is the basis for many systems in the U.S., including the BEHAVE fire behavior prediction system, the FARSITE fire area simulator, the National Fire Danger Rating System (NFDRS), the National Fire Management Analysis System (NFMAS) for economic planning, the Rare Event Risk Assessment Process (RERAP) [22], and many more.

To facilitate the use in models and systems, fuelbed inputs have been formulated into fuel models. A fuel model is a set of fuelbed inputs needed by a particular fire behavior or fire effects model. Although a fuel model technically includes all fuel inputs to the model, several fuel inputs have never been subject to control by a user when creating a custom fuel model, leaving them static, and loosing, then, some accuracy in the predictions. The fuel models have worked well for predicting spread rate and intensity of active fires at the peak of the fire season, in part because the associated dry conditions lead to a more uniform fuel complex. However, they have deficiencies for other purposes, including prescribed fire, wildland fire use, simulating the effects of fuel treatments on potential fire behavior, and simulating the transition to crown fire using crown fire initiation models, along with comparisons of behaviors with similar species in different latitudes.

To solve the above-mentioned problems it is necessary to change the static nature of the models, allowing the change of some characteristics that can be known at the beginning of the fire season, increasing, thus, the accuracy of the predictions and simulations. To tackle this from the point of view of the field work is impossible, so it is necessary to work together with images provided by hyperspectral sensors and techniques able to obtain abundance maps, which label the vegetal species in the areas and extract further information on physical, biological, and characteristics of plants present in the image. This can be done by developing advanced hyperspectral data processing algorithms, as will be described in the following section.

8.4 Advanced Hyperspectral Data Processing Algorithms

In the present section, we introduce several advanced hyperspectral data processing algorithms intended to be used as potential building blocks for an integrated system to monitor wildland fires through the combination of field and ground-truth information, hyperspectral and additional remote sensing data (i.e. fuelbeds characteristics), and detailed mathematical models.

8.4.1 Morphological Algorithm for Endmember Extraction and Classification

The ultimate goal of endmember extraction algorithms is to select the spectrally purest constituent spectra in a scene. These endmembers are assumed to interact following a linear or non-linear spectral mixing model, and such models can be used to identify land cover within a fine spatial resolution scene. For this reason, it is very important to accurately obtain the signatures of the endmembers present in the scene. In the following, we describe a morphological algorithm for endmember extraction that integrates the spatial and spectral information in the analysis. Mathematical morphology [12] is a classic non-linear spatial processing technique that provides a remarkable framework to achieve the desired integration of spatial and spectral information. Parallelization of this technique for efficient execution in multiprocessors will be detailed in subsequent sections of this chapter.

Before describing our proposed approach, let us denote by $\mathbf{F}$ a hyperspectral data set defined on an N-dimensional (N-D) space, where N is the number of channels or spectral bands. The main idea of the algorithm is to impose an ordering relation, in terms of spectral purity, in the set of pixel vectors lying within a spatial search window or structuring element (SE) around each image pixel vector [12]. To do so, we first define a cumulative distance between one particular pixel $\mathbf{f}(x, y)$, where $\mathbf{f}(x, y)$ denotes an N-D vector at discrete spatial coordinates $(x, y) \epsilon Z^2$, and all the pixel vectors in the spatial neighborhood one given by B (B-neighborhood) as:

$$D_B[\mathbf{f}(x, y)] = \sum_i \sum_j \text{SID}[\mathbf{f}(x, y), \mathbf{f}(i, j)] \tag{8.3}$$

where (i, j) are the spatial coordinates in the B-neighborhood and SID is the spectral information divergence, a commonly used distance in remote sensing applications [15] defined as follows:

$$\text{SID}[\mathbf{f}(x, y), \mathbf{f}(i, j)] = \sum_{l=1}^{N} p_l \cdot \log\left(\frac{p_l}{q_l}\right) + \sum_{l=1}^{N} q_l \cdot \log\left(\frac{q_l}{p_l}\right) \tag{8.4}$$

where

$$p_l = \frac{f_l(x, y)}{\sum_{k=1}^{N} f_k(x, y)} \tag{8.5}$$

$$q_l = \frac{f_l(i, j)}{\sum_{k=1}^{N} f_k(i, j)} \tag{8.6}$$

Based on the distance above, we calculate the extended morphological erosion of $\mathbf{F}$ by B for each pixel in the input data scene as follows [4]:

$$(\mathbf{f} \Theta B)(x, y) = argmin_{(i,j)}\{D_B[\mathbf{f}(x + i, y + j)]\} \tag{8.7}$$

where the $argmin$ operator selects the pixel vector that is most highly similar, spectrally, to all the other pixels in the B-neighborhood. On the other hand, the extended

morphological dilation of $\mathbf{f}$ by B is calculated as follows [4]:

$$(\mathbf{f} \oplus B)(x, y) = argmax_{(i,j)}\{D_B[\mathbf{f}(x + i, y + j)]\} \tag{8.8}$$

where the $argmax$ operator selects the pixel vector that is most spectrally distinct to all the other pixels in the B-neighborhood. With the above definitions in mind, we provide below an unsupervised classification algorithm for hyperspectral imagery called the Automatical Morphological Classification (AMC), which relies on extended morphological operations. The inputs to the algorithm are a hyperspectral data cube $\mathbf{f}$, a structuring element B, and the number of classes to be detected c. The output of the algorithm is a 2-D matrix that contains a classification label for each pixel $f(x, y)$ in the input image. The algorithm can be summarized by the following steps:

1. Initialize a morphological eccentricity index score $\text{MEI}(x, y) = 0$ for each pixel.
2. Move B through all the pixels of $\mathbf{F}$, defining a local spatial search area around each $\mathbf{f}(x, y)$, and calculate the maximum and minimum pixel, at each B-neighborhood using dilation and erosion, respectively. Update the MEI at each pixel using the SID between the maximum and the minimum.
3. Select the set of c pixel vectors in $\mathbf{F}$ with a higher associated score in the resulting MEI image.
4. Estimate the sub-pixel abundance $\alpha_i(x, y)$ of the pure pixels selected in the previous stage within $\mathbf{f}(x, y)$, using the standard linear mixture model described in [10].
5. Obtain a classification label for each pixel $\mathbf{f}(x, y)$ by assigning it to the class with the highest sub-pixel fractional abundance score in that pixel. This is done by comparing all estimated abundance fractions $\{\alpha_1(x, y), \alpha_2(x, y), \ldots, \alpha_c(x, y)\}$ and finding the one with the maximum value, say $\alpha_{i^*}(x, y)$, with $i^* = arg\{max_{1 \leq i \leq c}\{\alpha_i(x, y)\}\}$.

One of the main features of the algorithm is the regularity of its computations. As shown in previous work [4], its computational complexity is $O(p_f \times p_B \times N)$, where $p_{\mathbf{F}}$ is the number of pixels in the hyperspectral image $\mathbf{F}$ and p_B is the number of pixels in the structuring element B. This results in high computational cost in real applications.

8.4.2 Orthogonal Subspace Projection Algorithm for Target Detection

As opposed to classification algorithms, which aim at assigning appropriate class labels to each pixel, target detection algorithms aim at finding a collection of significant pixel vectors in accordance with different criteria.

One of the most effective target detection algorithms for hyperspectral image analysis is the automatic target generation process (ATGP), developed to find potential target pixels that can be used to generate a signature matrix using the concept of

orthogonal subspace projection (OSP) [10]. The algorithm makes use of an OSP projector defined defined by

$$P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^{\mathbf{T}} \tag{8.9}$$

The ATGP repeatedly makes use of equation 8.9 to find target pixel vectors of interest from the data without prior knowledge, regardless of the types of pixels that the targets one. It can be briefly described as follows. Let's assume that $\mathbf{t}_0$ is an initial target pixel vector. The algorithm begins with the initial target pixel vector $\mathbf{t}_0$ by applying an orthogonal subspace projector $P_{\mathbf{t}_0}^{\perp}$ specified by equation 8.9 with $\mathbf{U} = \mathbf{t}_0$ to all image pixel vectors. It then finds a target pixel vector, denoted by $\mathbf{t}_1$, with the maximum orthogonal projection in the orthogonal complement space, denoted by $< \mathbf{t}_0 >^{\perp}$, that is orthogonal to the space $< \mathbf{t}_0 >$ linearly spanned by $\mathbf{t}_0$. The reason for this selection is that the selected t_1 generally has the most distinct features from $\mathbf{t}_0$ in the sense of orthogonal projection, because $\mathbf{t}_1$ has the largest magnitude of the projection in $< \mathbf{t}_0 >^{\perp}$ produced by $P_{\mathbf{t}_0}^{\perp}$. A second target pixel vector $\mathbf{t}_2$ can be found by applying an orthogonal subspace projector $P_{[\mathbf{t}_0\mathbf{t}_1]}^{\perp}$ with $\mathbf{U} = [\mathbf{t}_0\mathbf{t}_1]$ to the original image, and the target pixel vector that has the maximum orthogonal projection in $< \mathbf{t}_0, \mathbf{t}_1 >^{\perp}$ is selected as $\mathbf{t}_2$. The above procedure is repeated until a certain stopping rule is satisfied, usually determined by an estimated number of target pixel vectors required to generate, using several different procedures for the determination of the number of target pixel vectors. If we consider p as the number of target pixel vectors to generate the stopping criterion, the ATGP can be summarized by the following steps:

1. *Initial condition*. Select an initial target pixel vector of interest denoted by $\mathbf{t}_0$. In order to initialize the ATGP without knowing $\mathbf{t}_0$, we select a target pixel vector with the maximum length as the initial target $\mathbf{t}_0$, namely, $\mathbf{t}_0 = arg\{max\{\mathbf{f}(x, y) \cdot \mathbf{f}(x, y)^T\}\}$, which has the highest intensity, i.e., the brightest pixel vector in the image scene. Set $k = 1$ and $\mathbf{U} = [\mathbf{t}_0]$. It is worth noting that this selection may not necessarily be the best selection. However, according to the experiments, it was found that the brightest pixel vector was always extracted later on, if it was not used as an initial target pixel vector in the initialization.
2. *Iterative target detection*. At the k-th iteration, apply $P_{\mathbf{t}_0}^{\perp}$ via equation 8.9 to all image pixels $\mathbf{f}(x, y)$ in the image and find the k-th target, $\mathbf{t}_k$, generated at the k-th stage, which has the maximum orthogonal projection.
3. *Stopping rule*. If $k < p - 1$, let $\mathbf{U}_k = [\mathbf{U}_{k-1}t_k] = [t_1, t_2, \cdots, t_k]$ be the k-th target matrix and go to step 2. Otherwise, continue. At this point, the ATGP is terminated. The resulting target matrix is $\mathbf{U}_{p-1}$, which contains $p - 1$ target pixel vectors as its column vectors, which do not include the initial target pixel vector $\mathbf{t}_0$. The final set of target pixel vectors produced comprises p target pixel vectors, $\{\mathbf{t}_0, \mathbf{t}_1, \cdots, \mathbf{t}_{p-1}\} = \{\mathbf{t}_0\} \bigcup \{\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_{p-1}\}$, that where found by repeatedly using equation 8.9. These target pixel vectors are the resulting targets for the hyperspectral image $\mathbf{F}$.

8.4.3 Self-Organizing Map for Neural Network Classification

A further approach for advanced classification of remotely sensed data consists of using a self-organizing map (SOM) [5]. This neural network architecture has been demonstrated in previous work to be very useful for analyzing hyperspectral images [8]. The main objective of the SOM model is to transform a given N-D signal or input pattern into a multidimensional map and the adaptive refining of such transformation using different topological criteria.

The SOM-based classification technique proposed in this chapter consists of N input neurons and M output neurons, where N is the dimensionality of input pixel vectors and M is the number of endmembers obtained using an endmember extraction algorithm (for instance, the morphological endmember extraction procedure described in this chapter). Therefore, our SOM neural network is composed of two layers, with forward connections from the input layer towards the output layer and the set of weights associated distributed in a matrix, named in the following $W_{M\times N}$. The analysis process performed by the network is divided in two phases: training and classification:

- In the training phase, the different patterns are presented to the neural network in a way that makes the forward connections change to adapt to the information contained in the training data.
- In the classification phase, the forward connections project the input patterns, i.e., the pixel vectors are classified in the feature space using the Euclidean distance to identify the winning neuron.

The whole procedure can be summarized in the following steps:

1. *Initialization of weights*. Random values are normalized to initialize the weight vectors: $w_i^{(0)}$, with $i = 1, 2, \cdots, M$.
2. *Training*. This step makes use of the endmembers obtained using some endmember extraction algorithm, e.g., the morphological endmember extraction procedure described in this chapter, which is used to provide input training patterns for the neural network.
3. *Clustering*. For each input pattern x, a winning neuron i^* is obtained in time t using a similarity criteria based on the Euclidean distance, i.e., $i^*[x] = \min_{1<=j<=M} ||x - w_j||^2$.
4. *Adjustment of weights*. The winning neuron and those in its neighborhood change their weights following the expression $w_i^{(t+1)} = w_i^t + \sum_{t'=t_0}^{t_{max}} \sum \alpha(t') \sum \sigma(t') \sum \left(x - w_i^{(t)}\right)$, where $\alpha(t)$ and $\sigma(t)$ are learning and neighborhood functions, respectively. One must take into account that the weights associated to i^* are modified proportionally to the learning frequency.
5. *Stopping condition*. The SOM algorithm ends as soon as a predetermined number of iterations, t_{max}, is completed.

From the steps indicated, it is clear that the SOM algorithm is naturally sequential. As a result, the parallelization strategies for this algorithm must particularly address the problem of data dependencies.

8.4.4 Spectral Mixture Analysis Algorithms

To conclude this section on advanced data processing techniques, we address the important issue of mixed pixels in hyperspectral image analysis. In several works, it has been demonstrated that the knowledge on the endmembers present in the image is not sufficient to characterize the different properties of plants, biomass, etc. In other words, once a set of endmembers has been extracted, it is important to be able to express the mixed pixels as a decomposition of a collection of 'pure' spectral signatures and a set of 'abundances' that indicate the individual proportion or contribution of each of the pure spectra to the mixed pixel [14].

The model used to properly describe the above situation is denominated as a 'mixture model,' which often assumes that the scene is composed of a limited set of endmembers with unique spectral features and a majority of mixed pixels with different endmembers participating in different proportions. In the models, two possibilities are observed:

- *Linear mixture model*. It considers that each incident beam of solar radiation only interacts with a single component or endmember, so that the total radiation reflected by a pixel can be decomposed proportionally to the abundance of each of the endmembers in the pixel.
- *Non-linear mixture model*. It supposes that the endmembers interact following a non-linear model. The non-linear effects appearing in this case are due, fundamentally, to multiple scattering effects in the light reflected by different materials.

In any case, one should note that the linear mixture model is not intended to fully describe the mixture problem since it is mainly a simplification. The real situation is dominated by secondary effects due to light multiply scattered by several covers, absorption and diffusion effects, shadows, etc. These effects pose an interesting question on which one of the mixture models is predominant in the mixture pixel spectra obtained by the sensors.

In the particular case of the wildland fires, the main goal is to estimate the temperature and quantity of fire appearing in each pixel of the image in order to produce maps of fires and hot spots that initiated the fires. For this purpose, several investigations have been carried out on the use of linear unmixing for the detection of the fraction of a fire endmember inside a pixel. The model proposed in [3] is the one used as a baseline in our investigations, and its basic idea is to use the linear spectral mixing model as an initial estimation, and then use the best fit linear spectral mixing model to identify fire temperature and land cover within a fine spatial resolution image scene. To accomplish this goal, we propose the use of artificial neural networks for linear unmixing of hyperspectral data.

8.5 Parallel Implementations

In the present section, we explore the application of parallel techniques in order to implement the different methods described in Section 8.4. First, we provide a parallel version of the AMC classification algorithm for both homogeneous and heterogeneous platforms, and then we describe a parallel version of the proposed SOM-based classification algorithm.

8.5.1 Parallelization of Automated Morphological Classification (AMC) Algorithm for Homogeneous Clusters

The parallel implementation of AMC is based on the definition of an efficient data partitioning scheme for hyperspectral imagery. We have considered two different approaches to the problem: partitioning in the spatial domain and partitioning in the spectral domain:

- The spatial-domain partitioning option divides the hyperspectral image in multiple blocks, in a way that the pixels for each block preserve its entire spectral identity.
- The spectral-domain partitioning 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, in spatial-domain partitioning the information of a single pixel in the image would be scattered across several different processing units. 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 [7]. Besides, the overhead introduced by the communication increases with the number of processors, thus introducing problems in the load-balancing accomplished by the designed algorithms [25].

At this point, we introduce the concept of a parallelizable spatial/spectral pattern (PSSP), 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 [24]. Such patterns are automatically generated by a spatial domain partitioning (SDP) module.

Figure 8.3 describes the partitioning framework using two computing units. In the example, the SDP divides the image into two PSSPs. 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 3×3 pixels. Such values are then updated in a local 2-D image. At the end of the process, the SDP integrates the various local images, obtaining a resulting 2-D image that is used as a baseline to extract a final set of endmembers, which lead to a final classification result. An issue of major importance in the design of SE-based parallel image processing applications is the possibility of

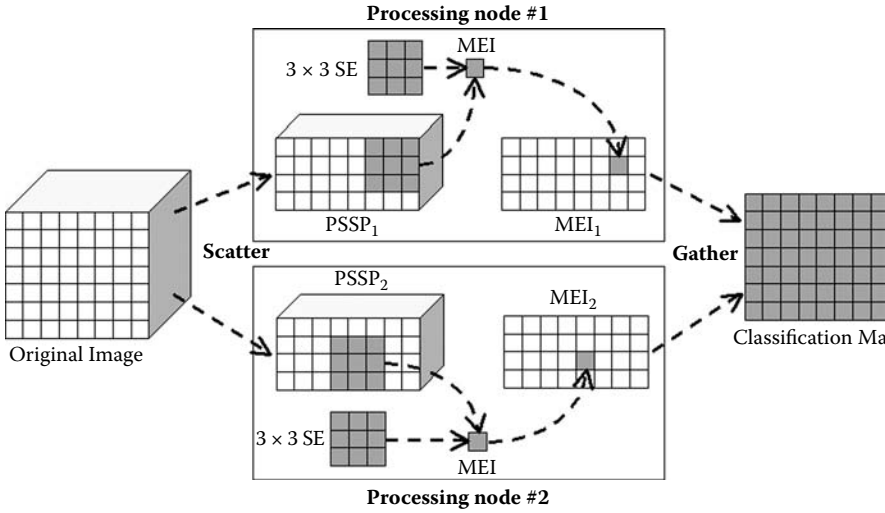

Figure 8.3 Concept of parallelizable spatial/spectral pattern (PSSP) and proposed partitioning scheme.

accessing pixels out of the spatial domain of the partition available in the processor. This is normally managed by a determined border-handling strategy. In our parallel implementation, two such strategies have been implemented, as described below:

- *Border-handling strategy relative to the pixels out of the domain of the original image*. This strategy is necessary in the situation illustrated in Figure 8.4. In this case, only the pixels of the SE that fall inside the image domain are used for the MEI computation. This strategy is similar to the mirroring technique commonly used in kernel-based image processing applications.

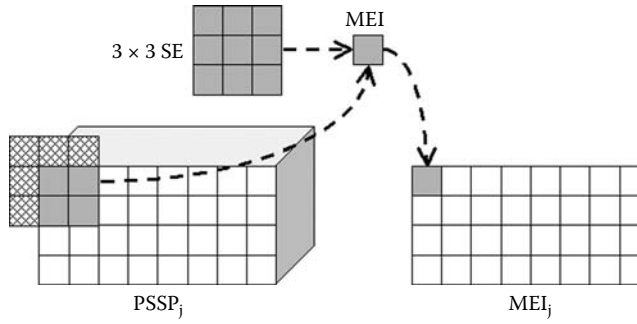

Figure 8.4 Problem of accessing pixels outside the image domain.

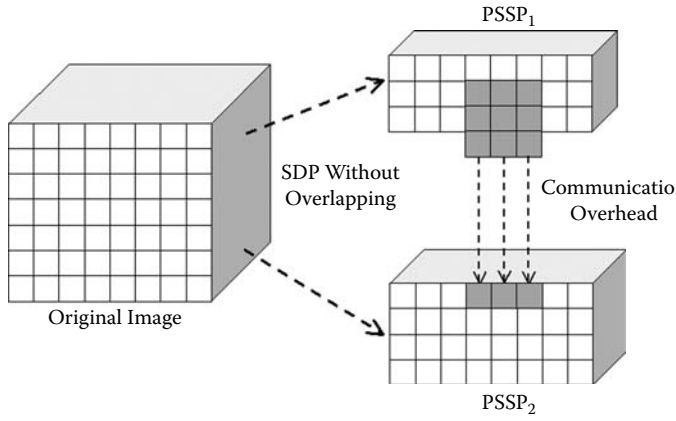

Figure 8.5 Additional communications required when the SE is located around a pixel in the border of a PSSP.

- *Border-handling strategy relative to the pixels out of the domain of the PSSP*. 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 8.5). To resolve this issue, we introduce redundant computations to minimize the communication/coordination between processors. In other words, we replicate the information necessary to avoid border effects between different processors, as shown in Figure 8.6.

According to our experiments [11], the cost of processing the information resulting from the procedure above is sensibly inferior to dealing with the overhead

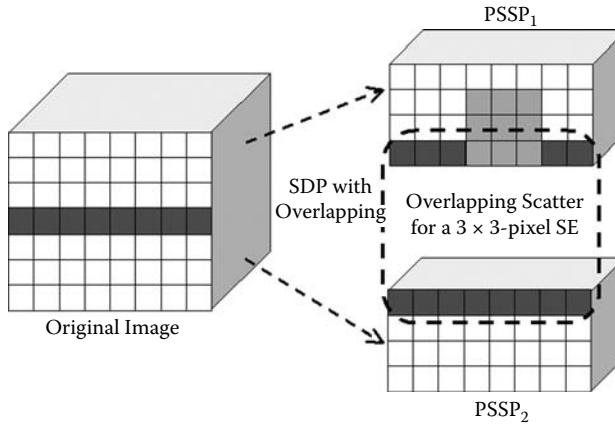

Figure 8.6 Border-handling strategy relative to pixels in the border of a PSSP.

introduced by communication among different processors if no redundant information is introduced in the system. Given the characteristics of the AMC algorithm, which relies on the utilization of an SE of 3×3 pixels iteratively, the number of redundant pixels p_R introduced in the processing of a hyperspectral image is given by $p_R = 2 \times [(2^{\lceil \frac{\log_2 N}{2} \rceil} t) - 1] \times I_R + 2 \times [(2^{\lfloor \frac{\log_2 N}{2} \rfloor}) - 1] \times I_C$, where N is the number of processors, I_R is the number of rows in the original image, and I_C is the number of columns in the original image.

All the algorithms in this section have been implemented in the C++ programming language, using calls to a message passing interface (MPI). Specifically, we used the MPICH-1.2.5 version due the demonstrated flexibility of this version in order to migrate the code to different parallel architectures, which increases code reusability and portability.

8.5.2 Parallelization of Automated Morphological Classification (AMC) Algorithm for Heterogeneous Clusters

As has been mentioned before, spatial/spectral morphological algorithms are particularly suitable for being implemented on heterogeneous architectures. Our efforts in this area have been directed toward the minimization of the execution time of the algorithms in order to provide (near) real-time responses. To describe and calculate the optimal data partitioning and the best data communication scheme for heterogeneous networks of computers, we resort to the HeteroMPI [2] library.

The first step to accomplish the HeteroMPI-based implementation is to define a performance model that is able to capture the data partitioning and communication framework for the heterogeneous platform [24]. Listing 1 shows the most important fragments of the code that describes the adopted performance model, which has six input parameters. Specifically, parameter *m* specifies the number samples of the data cube, while parameter *n* specifies the number of lines. Parameters *se_size* and *iter* respectively denote the size of the SE and the number of iterations executed by the algorithm. Finally, parameters *p* and *q* indicate the dimensions of the computational processor grid (in columns and rows, respectively) used to map the spatial coordinates of the individual processors within the processor grid layout. Finally, parameter *partition_size* is an array that indicates the size of the local PSSPs (calculated automatically using the computing power of the heterogeneous processors).

It should be noted that some of the definitions have been removed from Listing 1 for simplicity. However, some of the most representative sections are included. The *coord* section defines the mapping of individual abstract processors performing the algorithm onto the grid layout using variables I and J. The node primitive defines the amount of computations that will be made by every processor, which depends on its spatial coordinates in the grid as indicated by I and J and the computing power of the individual processors as indicated by *partition_size*. Finally, the parent directive indicates the spatial localization of the master processor in the grid. An additional link section is used to define the individual communications that every processor carries out based on its position in the grid. Further information on performance model definition is available in [2].

Once a performance model for the parallel algorithm has been defined, implementation using the standard HeteroMPI is quite straightforward [26]. Listing 2 shows the most interesting fragments of the HeteroMPI-based code of our parallel implementation. The HeteroMPI runtime system is initialized using operation HeteroMPI_Init. Then, operation HeteroMPI_Recon updates the estimation of performances of processors. This is followed by the creation of a group of processes using operation HeteroMPI_Group_create. The members of this group then perform the computations of the heterogeneous parallel algorithm using standard MPI mechanisms. This is followed by freeing the group using operation HeteroMPI_Group_free and the finalization of the HeteroMPI run-time system using operation HeteroMPI_Finalize. In the code described above, the benchmark function used to measure the processing power of the processors in HeteroMPI_Recon is essential, mainly because a poor estimation of the power and memory capacity of processors may result in load unbalancing problems. In our implementation, the benchmark function is based on a simple 3×3 morphological computation to calculate the MEI index, as described in Figure 8.3.

Listing 1 Typical performance model for hyperspectral analysis algorithms.

```
algorithm hhaa_ rend(int m, int n, int iter, int p, int q, int partition_size[p * q]) {
  coordI = p, J = q;
  node {I >= 0 && J >= 0: benchmark*(partition_size[I * q + J]*iter);};
  parent[0, 0];
}
```

Listing 2 Parallel implementation of AMC algorithm with HeteroMPI directives to calculate the load balance based on a benchmark function containing the core computations of the method.

```
main(int argc, char *argv[]){
  HeteroMPI_Init(&argc,&argv);
  If(HeteroMPI_Is_member(HMPI_COMM_WORLD_GROUP)){
    HeteroMPI_Recon(benchmark_function,dims,15,&output_p);
  }
  HeteroMPI_Group_create(&gid,&MPC_NetType_hhaa_rend,modelp,num_param);
  If(HeteroMPI_Is_free()){
    HeteroMPI_Group_create(&gid,&MPC_NetType_hhaa_rend,NULL,0);
  }
  if(HeteroMPI_Is_free()){
    HeteroMPI_Finalize(0);
  }
  If(HeteroMPI_Is_member(&gid)){
    HeteroMPI_Group_performances(&gid,speeds);
    //Calculations of the size and communications based on the obtained performance
    //Definition of the best size and distribution of communications
    Read_image(name,image,lin,col,bands,data_type,init);
    for( i = I_max ; i > 1 ; i - - ){
      AMC_algorithm(image,lin,col,bands,sizeofB,res);
    }
    if(HeteroMPI_Is_member(&gid)){
      free(image);
    }
    HeteroMPI_Group_free(&gid);
    HeteroMPI_Finalize(0);
  }
}
```

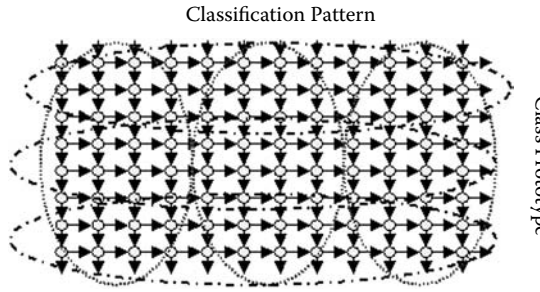

Figure 8.7 Partitioning options for the considered neural algorithm.

8.5.3 Parallelization of the SOM-based Classification Algorithm for Homogeneous Clusters

In order to parallelize the SOM algorithm, we face similar problems to those already addressed for the AMC algorithm in previous subsections. 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 [27] and considering the relatively small size of the training set [28], 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 simply illustrated in Figure 8.7.

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 result in worse results than those found by the sequential version of the algorithm. On the other hand, the partitioning scheme based on dividing by class prototype 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 $\log_2 N$ synchronizing 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 us to directly obtain 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, as was the case with the AMC algorithm. 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., $\sigma(t) = e^{\frac{\|i^* - i\|}{t}}$, other neighborhood functions may also be considered [8].

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 [6]. 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 matrices of weights $W_{M \times N}$.

8.6 Architectural Outline of an Advanced System for Management of Wildland Fires Using Hyperspectral Imagery

In the present section, we outline the system proposed for the surveillance and management of wildland fires. This system is built on the algorithms and models described in Sections 8.4 and 8.5, and also on the mathematical model for characterization of wildland fires described in Section 8.3. Figure 8.8 provides a flowchart of the proposed design, in which each processing component and the interconnections between them are displayed. It can be seen that the first stage of the flowchart is the extraction of the endmembers of the image that are to be used, first, as inputs to the linear unmixing process; second, as relevant spectra of the classes available on the scene; and, finally, as input of the classification stage that will provide the distribution of fuels, along with any other elements present in the image. The spectra obtained can be compared with information about laboratory and field measurements of specimens (bulk density, humidity, etc.) available in the database. The database gives relevant characteristics

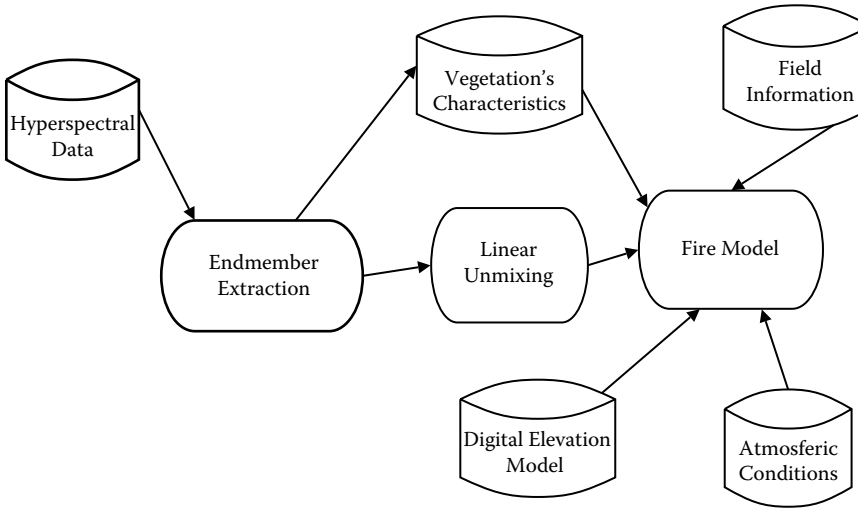

Figure 8.8 Functional diagram of the system design model.

of the vegetation that allows one to change the statical nature of the fuel models, thus providing the mathematical model with more precise data, and increasing its accuracy.

After this stage, the set of endmembers obtained used as a learning pattern for the neural network-based linear unmixing algorithm. In this particular case, we propose the use of a Hopfield neural network to perform the unmixing [6]. This stage will provide the abundance fractions of each endmember at each pixel in the image. In the system, these abundances indicate the precise distribution of fuels inside each pixel in the scene (i.e., mixture of fuelbeds, different stages of senescence for a plant, etc.). These abundances, combined with the experimental laboratory data adjusted to the actual conditions in the studied area, provide us with knowledge to define a particular fire risk index. These abundances may also be combined with distribution maps (such as those produced by the proposed SOM-based classifier), giving important information for the mathematical models to operate.

The model can be improved for realistic modeling of the behavior of the fire by additional information sources; i.e., the slope factor can be calculated using a digital elevation model (DEM) of the area, the wind factor can be obtained from meteorological measures, etc. Further, we emphasize that the proposed system adopts a component-based approach, in which each module can be replaced with different algorithms or enhanced versions, thus providing greater flexibility and accuracy on the particular results obtained.

In the following, we introduce the computational resources in which the proposed system can be implemented, including *homogeneous platforms, heterogeneous platforms*, and *specialized programmable hardware*, listing the advantages and disadvantages of each considered architecture from the viewpoint of their incorporation to the system.

8.6.1 Homogeneous Parallel Platforms

Homogeneous resources are the most widely used high-performance computer architectures. In this category, we can include architectures with shared memory multiprocessors (SMPs), vector processors, and homogeneous clusters.

- *Vector Processors*. The vector processor provides a single control flow with serially executed instructions operating on both vector and scalar operands. The parallelism of this architecture is at the instruction level [2].
- *Shared Memory Multiprocessors* (SMPs). The shared memory multiprocessor architecture consists of a number of identical processors sharing a global main memory. In general, the processors of an SMP computer framework are of the vector or superscalar architecture [2].
- *Homogeneous Clusters*. A computer system of the distributed memory multiprocessor architecture consists of a number of identical processors not sharing global main memory and interconnected via a communication network. The architecture is also called the massively parallel processors (MPP) architecture [2].

In our opinion, the main advantages of homogeneous resources from the viewpoint of their incorporation to a remote sensing data processing system are

- Regularity of computations in the processing and data access leads to a better understanding for new users.
- In the case of vector processors, if the size of the vector register is greater than or equal to the number of bands of the spectral signature, one can achive a very good speedup.
- Due to computer market sales and trends, it is common to buy a large number of computers, making homogeneous clusters a reasonable solution for high-performance computing on a large scale.
- Homogeneous platforms often allow abstractions from more complex questions (most of them at a hardware level), and allow the programmer to concentrate only on the development of applications.

On the other hand, we believe that the main drawbacks of homogeneous platforms for their incorporation to a remote sensing data processing development are

- The SMP and vector processors architectures tend to be very expensive and closed solutions for high-performance computing.
- Due to the price of supercomputers, they are usually only available in large computing facilities.
- The large size of hyperspectral data forces the allocation of great amounts of data, and a bad partitioning policy may result in too many cache misses and inefficient memory usage.

8.6.2 Heterogenous Parallel Platforms

New research trends in high-performance computing tend to the use of heterogeneous computing resources, such as heterogeneous networks and Grid architectures, covered extensively in subsequent chapters of this volume. In the following, we outline the platforms that could be integrated in our proposed system.

- *Heterogeneous networks of computers* (HNOCs). With the commercial availability of networking hardware, it soon became obvious that networked groups of machines distributed among different locations could be used together by one single parallel remote sensing code as a distributed-memory machine. As a result, HNOCs have now become a very popular tool for distributed computing with essentially unbounded sets of machines, in which the number and location of machines may not be explicitly known.
- *Grid architectures*. The grid is a new class of infrastructure. By providing scalable, secure, high-performance mechanisms for discovering and negotiating access to remote resources, the grid promises to make it possible for scientific collaborations to share resources on an unprecedented scale, and for geographically distributed groups to work together in ways that were previously impossible.

In our opinion, the main advantages of heterogeneous platforms for their incorporation to a remote sensing data processing system are

- Distribution of data based on availability processing power, which produce a better utilization of the resources in terms of usage and accessibility for different users.
- Greater supercomputing structures with different functionalities can be created with the use of grid infrastructure, allowing the interconnection of systems geographically distributed and the sharing of functionalities, databases, etc.
- Due to the inclusion of some homogeneous resources, even in the case of having them distributed in several centers or laboratories, one can still make use of the advantages indicated for homogeneous resources locally.

The main disadvantages of this platforms in the context of our considered application are, in our opinion

- The cost is still an issue, even though reusability of components allows integration of existing resources.
- There is a deep need for knowledge of compilers and libraries, although access to large computing infrastructures is often provided at a high level.
- The control of load balance and correct communication between several configurations is, in some situations, not fully defined and correctly exploited by general-purpose libraries, which forces the developers to explicitly expose the parallelism through compiler directives.
- The programming paradigms are generally more complex.

These architectures are especially indicated for embarrassingly parallel applications, which need great processing power and small communication, allowing data scattering in terms of availability of processing resources. This is indeed the case for the AMC, ATGP, and SOM algorithms described in Section 8.5 and also for the mathematical model in Section 8.6.

8.6.3 Programmable Hardware

To conclude this section, we emphasize the importance of specialized hardware for onboard processing as part of any integrated system for remote sensing data analysis. By means of onboard processing, one can drastically reduce the size of the data to be communicated by means of compression (e.g., for real-time surveillance).

In order to consider the design of a specialized electronic system for remote sensing data processing, there are several possibilities for implementation, each with their own advantages and disadvantages in terms of cost, flexibility, performance, and complexity. On the one hand, the best performance is usually given by full-custom designs. On the other hand, semi-custom designs based on programmable hardware provide a very good compromise in terms of cost-performance ratio. Examples include field-programmable gate arrays (FPGAs) and graphics processing units (GPUs). The application of these specialized hardware platforms for remote sensing data analysis are extensively covered in several chapters of the present volume and therefore are out of the scope of this chapter.

8.7 Experimental Results

In the present section, we illustrate the classification accuracy and parallel performance of the parallel algorithms described in Section 8.5. First, a brief overview of the parallel architectures used in this study is provided. Then, performance data for the parallel algorithm are given and discussed in light of realistic hyperspectral imaging applications.

8.7.1 Parallel Computer Architectures

Two parallel computers have been used to evaluate the computational performance of the morphological algorithm proposed. The first parallel computer used in the study is a Beowulf-type cluster called Thunderhead, located at NASA's Goddard Space Flight Center in Maryland (a detailed system description is available at http://thunderhead.gsfc.nasa.gov). The second architecture used for experiments is available at the Heterogeneous Computing Laboratory (HCL), University College Dublin (UCD). This heterogeneous cluster is composed of 16 heterogeneous nodes of two types (Xeon and CSUltra), running Fedora Core Linux and Sun Os operating systems. The MPI implementation is MPICH-1.2.5 with the HeteroMPI library installed. The nodes are interconnected via a 100 Mbit Ethernet communication network with a switch

enabling parallel communication among the processors. Although this is a simple configuration, it is also a quite typical and realistic heterogeneous platform as well.

8.7.2 Hyperspectral Data Sets

The hyperspectral scene used for the experiments in this chapter was collected by the NASA Jet Propulsion Laboratory Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) sensor and is characterized by very high spectral resolution (1939×614 pixels with 224 narrow spectral bands in the range $0.4\mu\text{m} - 2.5\mu\text{m}$ and moderate spatial resolution 20-m pixels). It was 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. The primary crops of the area, mainly corn and soybeans, were very early in their growth cycle with only about 5% canopy cover. This fact makes most of the scene pixels highly mixed in nature. Discriminating among the major crops under these circumstances can be very difficult, a fact that has made this scene a universal and extensively used benchmark to validate classification accuracy of hyperspectral imaging algorithms [29]. Figure 8.9 shows the spectral band at 587 nm of the original scene. Part of these data, including ground-truth, are available online (from http://dynamo.ecn.purdue.edu/~biehl/MultiSpec). Apart from the availability of detailed ground-truth information, we have particularly selected this scene because it contains several agricultural classes at different stages of growth and senescence and

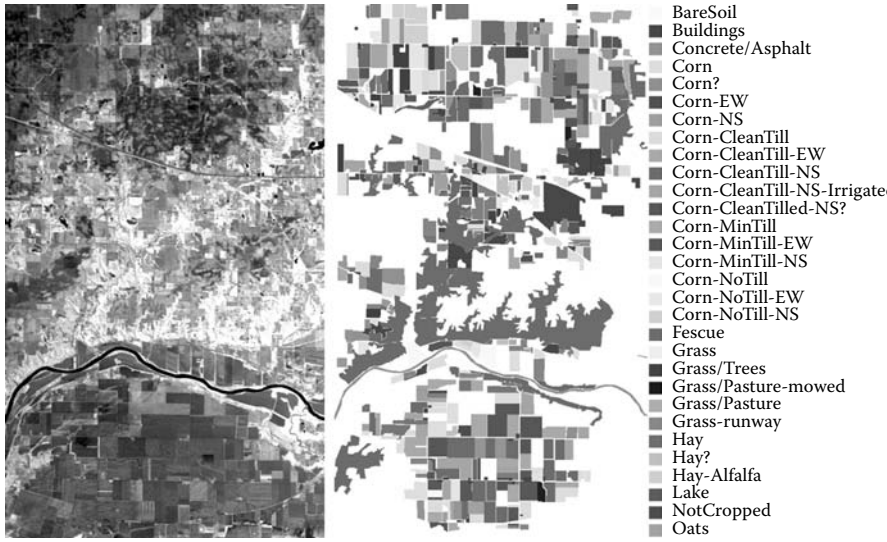

Figure 8.9 (Left) Spectral band at 587 nm wavelength of an AVIRIS scene comprising agricultural and forest features at Indian Pines, Indiana. (Right) Ground-truth map with 30 mutually exclusive land-cover classes.

with variable water content, which makes it a perfect scenario for testing the ability of the algorithms presented in this chapter in the task of generating accurate biomass fuel maps.

8.7.3 Performance Evaluation

Before analyzing its parallel performance, we first briefly discuss the classification accuracy obtained by the AMC algorithm with the different ground-truth classes available for the AVIRIS Indian Pines scene. For this purpose, Table 8.1 shows the

TABLE 8.1 Classification Accuracy Obtained by the Proposed Parallel AMC Algorithm for Each Ground-Truth Class in the AVIRIS Indian Pines Data

Class	**Classification Accuracy (%)**
BareSoil	98.05
Buildings	30.43
Concrete/Asphalt	96.24
Corn	99.37
Corn?	86.77
Corn-EW	37.01
Corn-NS	91.50
Corn-CleanTill	65.39
Corn-CleanTill-EW	69.88
Corn-CleanTill-NS	71.64
Corn-CleanTill-NS-Irrigated	60.91
Corn-CleanTilled-NS	70.27
Corn-MinTill	79.71
Corn-MinTill-EW	65.51
Corn-MinTill-NS	69.57
Corn-NoTill	87.20
Corn-NoTill-EW	91.25
Corn-NoTill-NS	44.64
Fescue	42.37
Grass	70.15
Grass/Trees	51.30
Grass/Pasture-mowed	79.87
Grass/Pasture	66.40
Grass-runway	60.53
Hay	62.13
Hay?	61.98
Hay-Alfalfa	83.35
Lake	83.41
NotCropped	99.20
Oats	78.04
Road	86.60
Woods	88.89
Overall:	72.35

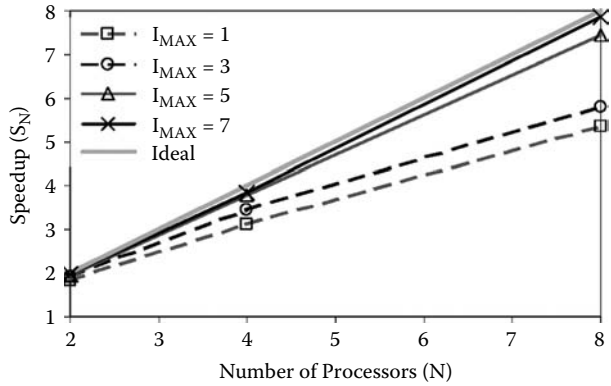

Figure 8.10 Speedups achieved by the parallel AMC algorithm using a limited number of processors on Thunderhead.

overall and individual classification accuracies (in percentages) achieved by the proposed parallel AMC algorithm, using a 3×3 structuring element for the construction of morphological operations and $I_{MAX} = 7$ algorithm iterations (the classification accuracies obtained for less iterations were lower).

As shown by Table 8.1, the lowest classification accuracies were reported for the buildings class, due to the presence of mixed pixels in this area as a result of the coarse spatial resolution of the scene, and for some of the corn classes due to the early growth stage of the crops in these areas, which also results in heavily mixed pixels. Quite opposite, very high classification accuracies were reported for macroscopically pure classes such as BareSoil, Concrete/Asphalt, and Woods. The measured overall accuracy of 72.35% is a very good classification result given the extremely high complexity of the data set, as reported in previous studies [15]. In addition, Table 8.1 also reveals that the results obtained are sufficient to meet the requirements of the system proposed in terms of accuracy, for the generation of biomass maps, and the capacity to discriminate between several vegetation canopies at different stages of growth and senescence.

Figure 8.10 shows the speedup achieved by the parallel AMC algorithm using a limited number of processors on Thunderhead. As we can see in the figure, the speedups achieved were better when the number of iterations (parameter I_{MAX}) were increased, thus revealing that the algorithm scales better when the amount of data to be processed is higher. On the other hand, Figure 8.11(a)–(b) reports the speedup and parallel efficiency achieved by the first three steps of the AMC algorithm (which extract the most relevant endmembers from the Indian Pines scene), using much larger numbers of processors on the Thunderhead system. As we can see, the speedup and parallel efficiency were better when the number of iterations (parameter I_{MAX}) was increased, thus revealing that the algorithm scales better when the amount of data to be processed is higher. On the other hand, Figure 8.11(c) and (d) illustrates the performance of the

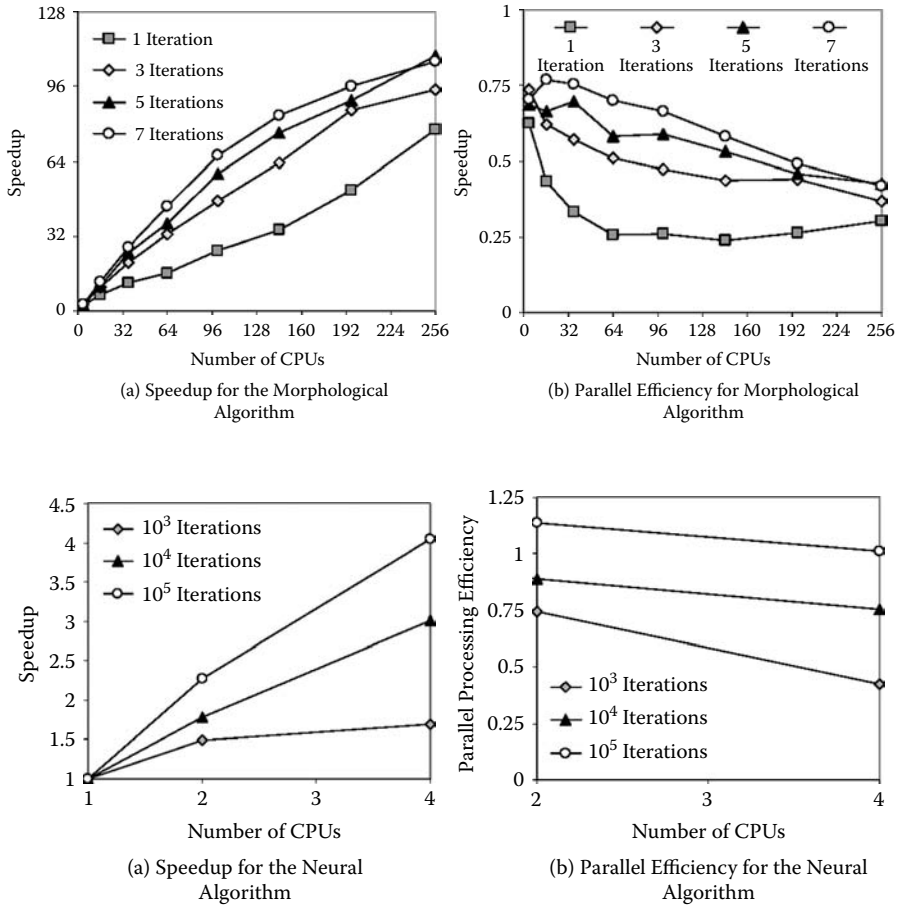

Figure 8.11 Speedups achieved by the parallel SOM-based classification algorithm (using endmembers produced by the first three steps of the AMC algorithm) using a large number of processors on Thunderhead.

parallel SOM-based classification stage (only a maximum of four Thunderhead processors were required to complete all calculations), which also scales well and even results in super-linear scalability. Finally, Figure 8.12 shows performance results for a straightforward master-slave parallelization of the ATGP algorithm, in which the data one partitioned and processed locally at each worker and the targets identified locally at each processor are gathered at the master processor. This implementation suffers from unbalanced communications, which lead to low speedups. Here, the lack of an appropriate benchmark function for dynamic distributions may also be an issue affecting parallel performance.

In order to illustrate the performance of our proposed HeteroMPI-based implementation of the AMC algorithm on the HCL heterogeneous cluster at UCD, Table 8.2

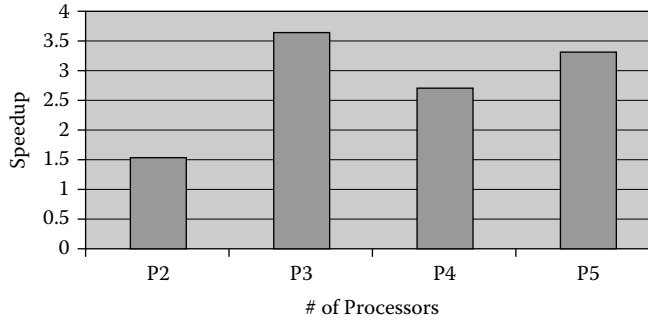

Figure 8.12 Speedups achieved by the parallel ATGP algorithm using a limited number of processors on Thunderhead.

shows the execution times obtained for each heterogeneous processor. The processing times reported in Table 8.2 are well balanced as indicated by Table 8.3, which shows the minima (t_{min}) and maxima (t_{max}) processing times (in seconds) and the load imbalance (defined as $R = \frac{t_{max}}{t_{min}}$).

To conclude this chapter, we provide a preliminary evaluation (in terms of elapsed time) of the different stages in order to evaluate the global response time that we expect for our proposed fire tracking/surveillance system. We emphasize that the fire model used in our experimentation is the one available in the Firelib software package [22],

TABLE 8.2 Execution Times (Seconds) of the HeteroMPI-Based Parallel Version of AMC Algorithm on the Different Heterogeneous Processors of the HCL Cluster

I_{MAX}	1	2	3	4	5	6	7
0	45.37	92.81	135.15	180.23	226.41	266.37	311.95
1	43.31	92.63	135.42	180.15	225.79	279.03	315.19
2	46.84	92.15	134.23	178.81	226.79	272.02	314.95
3	45.73	92.16	139.06	185.82	226.41	271.77	329.92
4	45.55	92.79	137.15	184.96	221.29	265.24	318.17
5	44.93	93.65	135.11	185.14	224.56	274.07	323.19
6	46.86	90.55	135.79	180.23	228.80	278.85	315.95
7	46.38	91.82	137.82	187.15	229.43	274.45	325.80
8	54.57	107.41	161.06	204.08	268.01	314.47	357.69
9	54.90	108.67	158.50	201.71	266.86	315.72	350.59
10	54.43	105.76	158.10	201.46	262.69	315.14	356.83
11	53.28	105.94	158.48	199.51	262.71	315.30	349.34
12	52.49	106.81	157.97	198.88	259.77	311.86	346.56
13	49.98	102.27	158.63	193.45	257.12	308.38	347.03
14	50.10	87.64	158.68	196.05	250.73	308.77	328.95

TABLE 8.3 Minima (t_{min}) and Maxima (t_{max}) Processor Run-Times (In Seconds) and Load Imbalance (R) of the Heterompi-Based Implementation of AMC Algorithm on the HCL Cluster

	I_{MAX}	**2**	**3**	**4**	**5**	**6**	**7**
t_{max}	54.90	108.67	161.06	204.08	268.01	315.72	357.69
t_{min}	43.31	87.64	134.23	178.81	221.29	265.24	311.95
$R = \frac{t_{max}}{t_{min}}$	1.26	1.23	1.19	1.14	1.21	1.19	1.13

with the inclusion of the results from previous hyperspectral analysis stages in order to refine and complement the static nature of the method.

Specifically, the execution of the Firelib software [22] (software based on the Rothermel model described in Section 8.6) resulted in an average elapsed time of 60 seconds for an image grid of a size similar to the hyperspectral scene considered in Figure 8.9. The execution time for the parallel AMC method is 18 seconds when 256 processors are used on Thunderhead (it should be noted that the HeteroMPI-based version of AMC took 358 seconds to be executed on the HCL heterogeneous cluster with 16 processors). On the other hand, the proposed parallel SOM-based classification algorithm resulted in an execution time of 560 seconds on Thunderhead. This algorithm employs the endmembers provided by the parallel AMC algorithm as input. Therefore, the estimated execution time of the whole system (including the parallel version of Firelib and the combined AMC/SOM classification), which would be able to accurately characterize the fire risk, is around 660 seconds. This response time can be further reduced by incorporating specialized hardware implementations of AMC and SOM on GPUs and reconfigurable hardware architectures, as indicated by subsequent chapters.

8.8 Conclusions

In this chapter, we have outlined the preliminary design of a fire tracking/surveillance system based on the integration of remotely sensed hyperspectral imagery and advanced processing algorithms implemented on a parallel computing infrastructure. The system is inspired by semi-empirical fire models such as the one proposed by Rothermel. In particular, this chapter conducts a preliminary evaluation on how the proposed parallel processing algorithms can be used to transform the (generally) static nature of fire models into a more dynamic framework, able to incorporate features that are highly relevant to characterize the evolution of wildland fires, such as automatic characterization of water content and distribution of biomass using spectral mixture analysis techniques. Further, we evaluated the performance of the proposed parallel algorithms using high-performance architectures such as homogeneous and heterogeneous supercomputers, and also studied their scalability and parallel efficiency from

a computational standpoint. From our experimental assessment, we conclude that the strong real-time restrictions required for a complex system of this kind are still subject to future developments and investigations. Our future research will be directed toward the optimization of the parallel algorithms proposed in this chapter for more efficient execution on distributed platforms and specialized hardware architectures.

8.9 Acknowledgments

The authors thank Drs. Robert Green, Anthony Gualtieri, James Tilton, John Dorband, and Alexey Lastovetsky for many helpful discussions, and also for their collaboration in the development of experiments on the Thunderhead Beowulf cluster and the HCL cluster at University College Dublin.

References

[1] R. C. Rothermel, A mathematical model for Predicting fire spread in wildland fuels, Research Paper INT-115. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station, 1972.

[2] A. Lastovetsky, *Parallel Computing on Heterogeneous Networks*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2003.

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

[4] A. Plaza, P. Martinez, R. Perez and J. Plaza, Spatial/spectral endmember extraction by multidimensional morphological operations, *IEEE Transactions on Geoscience and Remote Sensing,* vol. 40, no. 9, pp. 2025–2041, Sept. 2002.

[5] T. Kohonen, *Self-Organizing Maps,* Springer, Berlin, Heidelberg, 1995. (Second Edition 1997).

[6] P. Martinez, P. L. Aguilar, R. Perez and A. Plaza, Systolic SOM neural network for hyperspectral image classification. *Neural Networks and Systolic Array Design,* edited by D. Zhang and S. K. Pal. World Scientific, pp. 26–43, 2002.

[7] A. Plaza, Extended morphological profiles for hyperspectral image analysis on parallel computers, *Neural Information Processing Systems Conference,* Vancouver, 2003.

[8] P. L. Aguilar, A. Plaza, R. M. Perez, P. Martinez, Morphological endmember identification and its systolic array design. In *Neural Networks and Systolic Array Design,* pp. 47–69, 2002.

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

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

[11] A. Plaza, D. Valencia, J. Plaza and C.-I Chang, Parallel implementation of endmember extraction algorithms from hyperspectral data. *IEEE Geoscience and Remote Sensing Letters,* vol. 3, no. 3, pp. 334–338, 2006.

[12] P. Soille, *Morphological Image Analysis, Principles and Applications (2nd ed.),* Springer, Berlin, 2003.

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

[14] N. Keshava and J. F. Mustard, Spectral unmixing, *IEEE Signal Processing Magazine,* vol. 19, pp. 44–57, 2002.

[15] A. Plaza, *Development, validation and testing of a new morphological method for hyperspectral image analysis that integrates spatial and spectral information,* Ph.D. Dissertation, Computer Science Department, University of Extremadura, Spain, 2002.

[16] ENVISAT's MERIS webpage: http://envisat.esa.int/instruments/meris/.

[17] Spanish National Association of Forest Entreprises webpage: http://www.asemfo.org.

[18] GISCA webpage: http://www.gisca.adelaide.edu.au/.

[19] Canadian Fire model webpage: http://www.firegrowthmodel.com/index.cfm.

[20] D. D. Evans, R. G. Rehm, E. S. Baker, E. G. McPherson and J. B. Wallace, Physics-based modeling of community fires. *International Interflam Conference, 10th Proceedings,* vol. 2, 2004.

[21] C.-I. Chang and A. Plaza, A fast iterative implementation of the pixel purity index algorithm. *IEEE Geoscience and Remote Sensing Letters,* vol. 3, pp. 63–67, 2006.

[22] Public Domain Sotfware for the Wildland Fire Community webpage: http://www.fire.org.

[23] J. Setoain, C. Tenllado, M. Prieto, D. Valencia, A. Plaza and J. Plaza, Parallel hyperspectral image processing on commodity grahics hardware, *Proceedings of International Conference on Parallel Processing (ICPP'2006),* Columbus, OH, 2006.

[24] D. Valencia and A. Plaza, *FPGA-Based Compression of Hyperspectral Imagery Using Spectral Unmixing and the Pixel Purity Index Algorithm,* vol. 3993, Lecture Notes in Computer Science, Springer, New York, pp. 24–31, 2006.

[25] 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 Commuications (ISCC),* Cartagena, Spain, pp. 995–1000, 2005.

[26] D. Valencia, A. Lastovetsky and A. Plaza, Design and implementation of a parallel heterogeneous algorithm for hyperspectral image analysis using HeteroMPI, *Proceedings of 5th International Symposium on Parallel and Distributed Computing (ISPDC),* Timisoara, Romania, 2006.

[27] J. Plaza, R. Perez, A. Plaza, P. Martinez and D. Valencia. Parallel morphological/neural classification of remote sensing images using fully heterogeneous and homogeneous commodity clusters, *Proceedings of IEEE International Conference on Cluster Computing (Cluster'2006),* Barcelona, Spain, 2006.

[28] T. D. Hamalainen, Parallel implementation of self-organizing maps. In *Self-Organizing Neural Networks: Recent Advances and Applications,* U. Seiffert and L. C. Jain, Eds. Springer, Berlin, pp. 245–278, 2002.

[29] J. A. Gualtieri, Hyperspectral analysis, the support vector machine, and land and benthic habitats, *IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data,* pp. 354–364, 2003.