Parallel Morphological Processing of Hyperspectral Image Data on Heterogeneous Networks of Computers

Antonio J. Plaza

Computer Architecture and Technology Section
Computer Science Department, University of Extremadura
Avda. de la Universidad s/n, E-10071 Caceres, SPAIN
Phone: +34 (927) 257195; Fax: +34 (927) 257203
E-mail: aplaza@unex.es

Abstract

Recent advances in space and computer technologies are revolutionizing the way remotely sensed data is collected, managed and interpreted. The development of efficient techniques for transforming the massive amount of collected data into scientific understanding is critical for space-based Earth science and planetary exploration. Although most currently available parallel processing strategies for hyperspectral image analysis assume homogeneity in the computing platform, heterogeneous networks of computers represent a very promising cost-effective solution expected to play a major role in the design of high-performance computing platforms for many on-going and planned remote sensing missions. This paper explores techniques for mapping morphological hyperspectral analysis algorithms, characterized by their scalability and sub-pixel accuracy, onto heterogeneous parallel computers. Important aspects in algorithm design are illustrated by using both homogeneous and heterogeneous parallel computing facilities available at NASA's Goddard Space Flight Center and University of Maryland. Experiments reveal that heterogeneous networks of workstations represent a source of computational power that is both accessible and applicable in many remote sensing studies.

1. Introduction

The incorporation of last-generation sensors to airborne and satellite platforms is currently producing a nearly continual stream of high-dimensional data, and this explosion in the amount of collected information has rapidly introduced new processing challenges [1]. In particular, NASA is continuously gathering imagery data with Earth-observing sensors. Recent advances in sensor technology have led to the development of so-called hyperspectral instruments, capable of collecting hundreds of images, corresponding to different wavelength channels for the same area on the surface of the Earth. The concept of hyperspectral imaging (see Fig. 1) was introduced when NASA's Jet Propulsion Laboratory Airborne Visible-Infrared Imaging Spectrometer (AVIRIS) [2] was developed. This imager currently covers the wavelength region from 0.4 to 2.5 μm using 224 spectral channels, at a nominal spectral resolution of 10 nm. On other hand, the Hyperion hyperspectral imager aboard NASA's Earth Observing-1 (EO-1) spacecraft has been NASA's first hyperspectral imager to become operational on-orbit. It routinely collects images hundreds of kilometers long with 220 spectral bands from 0.4 to 2.5 μm. In the near future, the use of hyperspectral sensors on satellite platforms will produce a nearly continual stream of multidimensional data, and this expected high data volume would demand fast and efficient means for storage, transmission and analysis. The automation of techniques for transforming collected data into scientific understanding is critical for space-based Earth science and planetary exploration with onboard scientific data analysis.

While integrated spatial/spectral developments hold great promise for Earth science image analysis, they also introduce new processing challenges [3]. In particular, the price paid for the wealth of spatial and spectral information available from hyperspectral sensors is the enormous amounts of data that they generate [4]. Several applications, however, require that a response is provided quickly enough for practical use. Relevant examples include environmental modeling and assessment, target detection for military and defense/security purposes, urban planning and management studies, risk/hazard prevention and response including wild land fire tracking, biological threat detection, monitoring of oil spills and other types of

1-4244-0054-6/06/$20.00

chemical contamination. To address the computational need introduced by such relevant applications, several efforts have been recently directed towards the incorporation of high-performance computing models in remote sensing missions [3], especially with the advent of relatively cheap Beowulf clusters [5]. The new processing power offered by such commodity systems can be employed in data mining applications from massively large data archives (it is estimated that NASA collects and sends to Earth more than 850 GB of hyperspectral data every day). Further, real-time systems for onboard data analysis and compression still need to be fully incorporated to remote sensing missions. Although most parallel techniques and systems for image information processing employed by NASA and other institutions during the last decade have chiefly been homogeneous in nature, a current trend in the design of systems for analysis and interpretation of the massive volumes of data provided by space-based Earth science and planetary exploration missions is to utilize heterogeneous resources. This heterogeneity is seldom planned, arising mainly as a result of technology evolution over time and computer market sales and trends. Commodity off-the-shelf heterogeneous clusters of computers can realize a very high level of aggregate performance [6], and it is expected that these clusters will represent a tool of choice for the scientific community devoted to high-dimensional image analysis in remote sensing and other fields [7-9]. It is also worth noting that significant opportunities to exploit heterogeneous computing techniques are still available in the analysis of high-dimensional image data sets [10].

In this paper, we explore techniques for mapping hyperspectral image analysis algorithms onto heterogeneous networks of computers. Section 2 describes a hyperspectral analysis methodology that will serve as our case study throughout the paper. Section 3 develops parallel versions of the considered approach, specifically designed for heterogeneous platforms. In Section 4, we assess the parallel performance of the considered parallel algorithms by drawing comparisons between their efficiency on a heterogeneous cluster of workstations with the efficiency evidenced by their homogeneous counterparts on a homogeneous cluster with the same aggregate performance as the heterogeneous one. This evaluation strategy is adopted from recent studies by Lastovetsky and Reddy. Performance data on Thunderhead, a (homogeneous) massively parallel Beowulf cluster at NASA's Goddard Space Flight Center are also given. Section 5 concludes with some remarks.

2. Hyperspectral Analysis Methodology

This section develops a sequential morphological processing algorithm for analysis and classification of hyperspectral image data [11]. The algorithm will be used as a case study throughout the paper, as a representative algorithm of integrated spatial/spectral approaches, i.e., algorithms that take into account both the spatial and spectral information of the data in simultaneous fashion. Such hybrid techniques represent the most advanced generation of algorithms for analyzing hyperspectral imagery [1]. Before describing our proposed approach, let us denote by $\boldsymbol{f}$ a hyperspectral data set defined on an N-dimensional (N-D) space, where N is the number of channels or spectral bands. Using extended morphological operations [12], we impose an ordering relation in terms of spectral purity in the set of pixel vectors lying within a spatial search window (structuring element), designed by B, and defined in advance. In order to do so, we first define a cumulative distance between one particular pixel $\boldsymbol{f}(x,y)$, where $\boldsymbol{f}(x,y)$ denotes an N-D vector at discrete spatial coordinates $(x,y) \in Z^2$, and all the pixel vectors in the spatial neighborhood given by B (B-neighborhood) as:

$$D_B[\boldsymbol{f}(x,y)] = \sum_i \sum_j SAM[\boldsymbol{f}(x,y), \boldsymbol{f}(i,j)],$$

where (i,j) refers to spatial coordinates in the B-neighborhood and SAM is the spectral angle mapper [1], defined by :

$$\mathrm{SAM}(\boldsymbol{f}(x,y), \boldsymbol{f}(i,j)) = \cos^{-1}\left(\boldsymbol{f}(x,y) \cdot \boldsymbol{f}(i,j) / \|\boldsymbol{f}(x,y)\| \|\boldsymbol{f}(i,j)\|\right)$$

As a result, $D_B[\boldsymbol{f}(x,y)]$ is ultimately given by the sum of SAM scores between $\boldsymbol{f}(x,y)$ and every other pixel vector in the B-neighborhood. Based on the distance above, we calculate the extended erosion of $\boldsymbol{f}$ by B [13] for each pixel in the input data scene, i.e., for each possible B-neighborhood in the input scene, we select the pixel that produces the minimum value for D_B :

$$(\boldsymbol{f} \Theta B)(x,y) = \left\{\boldsymbol{f}(x+i', y+j'), (i', j') = \arg\min_{(i,j)} \{D_B[\boldsymbol{f}(x+i, y+j)]\}\right\}$$

where the arg min operator selects the pixel vector is most highly similar, spectrally, to all the other pixels in the B-neighborhood. Similarly, we apply an extended dilation of $\boldsymbol{f}$ by B [13] to select (for each possible B-neighborhood in the input scene) the pixel vector that produces the maximum value for D_B :

$$(\boldsymbol{f} \oplus B)(x,y) = \left\{\boldsymbol{f}(x-i', y-j'), (i', j') = \arg\max_{(i,j)} \{D_B[\boldsymbol{f}(x+i, y+j)]\}\right\}$$

where the argmax operator selects the pixel vector that is most spectrally distinct to all the other pixels in the B-neighborhood. Finally, we calculate a morphological eccentricity index (MEI) [11] at each pixel as follows:

$$\mathrm{MEI}(x,y) = \mathrm{SAM}[(\boldsymbol{f} \oplus B)(x,y), (\boldsymbol{f} \Theta B)(x,y)]$$

The resulting MEI scores (at a pixel level) can be then used for a variety of applications in hyperspectral imaging [12], most notably, to select the "purest" pixels in the

image data, which can then be used to express "mixed" pixels in terms of linear/nonlinear combinations of pure pixels. Mixed pixel characterization is crucial in hyperspectral imaging, where the spatial resolution of the sensor is often not fine enough to separate different pure materials at a sub-pixel level and these can jointly occupy a single pixel, with the resulting spectral measurement at the pixel given by composite of the individual spectra. Although spectral unmixing procedures based on pure pixel identification have become quite popular in recent years, their exploitation in real applications is often limited by their high computational complexity.

3. Parallel Implementation

This section describes a heterogeneous parallel processing framework for hyperspectral image analysis, which makes use of the morphological algorithm outlined in section 2 as a case study. Before providing an overview of the proposed parallel processing approach, we fist discuss volume partitioning and data communications.

3.1. Volume Partitioning

A major requirement for efficient parallel algorithms on distributed memory systems is finding a decomposition that minimizes the communication between the processors. For that purpose, domain decomposition techniques provide flexibility and scalability in parallel image processing. Two types of partitioning can be exploited in hyperspectral image analysis algorithms: spectral-domain partitioning and spatial-domain partitioning [5]. Spectral-domain partitioning subdivides the volume into small cells or sub-volumes made up of contiguous spectral bands, and assigns one or more sub-volumes to each processor. With this model, each pixel vector is split amongst several processors, which breaks the spectral identity of the data because the calculations for each pixel vector (e.g., for the SAM calculation) need to originate from several different processing units [14].

On the other hand, spatial-partitioning preserves the entire spectral signature of each hyperspectral image pixel [5]. In this work, we adopt a spatial-domain partitioning approach due to several reasons. First, the application of spatial-domain partitioning is a natural approach for low-level image processing [10], as many operations require the same function to be applied to a small set of elements around each data element present in the image data structure. A second reason has to do with the cost of inter-processor communication. In spectral-domain parallel, the structuring element-based calculations made for each hyperspectral pixel need to originate from several processing elements, and thus require intensive inter-processor communication.

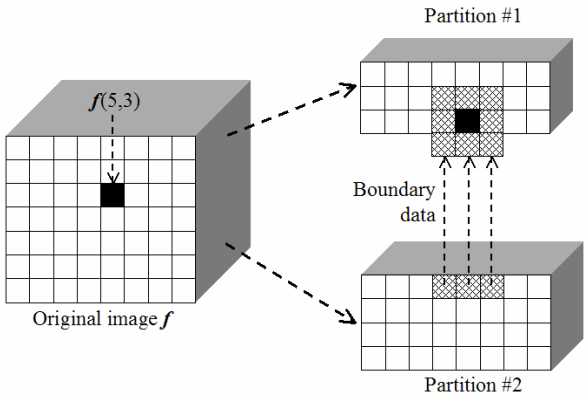

Figure 1. 3x3-pixel structuring element computation split between two processing nodes.

Finally, we believe that volume partitioning in the spatial domain is easier to handle in systems for hyperspectral imaging due to the fact that most available algorithms rely on spatial domain decomposition, while spectral domain-based partitioning would require careful re-design and re-programming of standard techniques.

3.2. Handling Data Communications

Before describing our adopted parallel algorithm, we should point out that an important issue in neighborhood-based image processing applications such as mathematical morphology is that additional inter-processor communications are required when the structuring element computation needs to be split amongst several different processing nodes due to boundary effects, as illustrated in Fig. 2 for a 3x3-pixel structuring element. In the example, the computations for a certain pixel need to originate from two heterogeneous processors, and a communication overhead involving three high-dimensional pixel vectors is introduced.

However, if redundant information such as an overlap border is added to one of the adjacent partitions to avoid accesses outside image domain, as illustrated in Fig. 3, then boundary data to be exchanged between neighboring processors can be greatly minimized [15]. It is clear at this point that an overlapping scatter would introduce redundant computations, since the intersection between the two involved partitions would be non-empty. It is also worth noting that the solution above may be prohibitive for large structuring element sizes. Subsequently, there is an application-dependent threshold to decide whether a redundant information-based or data exchange-based strategy should be adopted.

In order to explore the above relevant issue, in this work we will compare three different overlap communication strategies. It should be noted that the strategies addressed below have never been tested in the context of hyperspectral imaging applications:

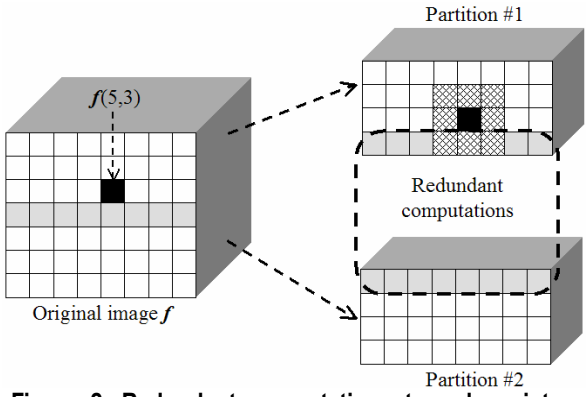

Figure 2. Redundant computations to reduce inter-processor communication.

a) Standard non-overlapping scatter, followed by overlap communication in the structuring element-based filtering operation for *every* pixel (as shown in Fig. 2), thus sending very small sets of pixels very often.
b) Standard non-overlapping scatter, followed by overlap communication *before* the morphological filtering, to have all data available in the overlap border areas (thus sending all border data beforehand, but only once). This is the strategy that is used in [16].
c) A special "overlapping scatter" operation, that also sends out the overlap border data as part of the scatter operation itself.

3.2. Implementation Details

The main purpose of the parallel algorithm in this section is to provide a mechanism to slice the available data into chunks (according to the spatial-domain volume partitioning framework described in subsection 3.1) so that the total execution time is minimized. For this purpose, there is a need to load-balance the workloads of p participating heterogeneous resources so that each processor P_i will accomplish a share α_i of the total workload W, where $\alpha_i \geq 0$ for $1 \leq i \leq p$ and $\sum_{i=1}^{p} \alpha_i = 1$. Therefore, a desired goal is to find a set of optimal values for the set $\{\alpha_i\}_{i=1}^{p}$, taking into account the three data communication strategies addressed in subsection 3.2. We provide below a step-by-step description of the proposed parallel algorithm, where the input parameters are a hyperspectral image f with N spectral channels, and a structuring element, B, that will be used for the construction of morphological operations. Based on our previous results with the proposed morphological algorithm, we will assume that the structuring elements used in this work are square-shaped, although structuring elements with different shapes may also be considered.

Heterogeneous Morphological Processing (HMP)
Inputs: N-dimensional image f, Structuring element B
Output: 2-dimensional image MEI

1. Obtain necessary information about the heterogeneous system, including the number of available processors, p, each processor's identification number, $\{P_i\}_{i=1}^{p}$, and processor cycle-times, $\{w_i\}_{i=1}^{p}$.
2. Using B and the information obtained in step 1, determine the total volume of information, R, that needs to be replicated from the original data volume, V according to the three data communication strategies described in subsection 3.2.
3. Let the total workload W to be handled by the algorithm be given by $W = V + R$.
4. Set $\alpha_i = \left\lfloor \frac{(p/w_i)}{\sum_{i=1}^{p}(1/w_i)} \right\rfloor$ for all $i \in \{1, \cdots, p\}$.
5. For $m = \sum_{i=1}^{p} \alpha_i$ to $(V+R)$, find $k \in \{1, \cdots, p\}$ so that $w_k \cdot (\alpha_k + 1) = \min\{w_i \cdot (\alpha_i + 1)\}_{i=1}^{p}$ and set $\alpha_k = \alpha_k + 1$.
6. Use the resulting $\{\alpha_i\}_{i=1}^{p}$ to obtain a set of p spatial-domain heterogeneous partitions of $(V+R)$, and send its corresponding partition to each processor P_i along with B following the data communication strategies described in subsection 3.2.
7. Execute the sequential algorithm in section 2 in parallel at each heterogeneous processor.
8. Collect all the individual results $\{\text{MEI}_i\}_{i=1}^{p}$ provided by each processor P_i, and merge them together to form a final image $\text{MEI} = \bigcup_{i=1}^{p} \{\text{MEI}_i\}$.

In order to perform spatial-domain data partitioning in step 6, we adopt a hybrid methodology with two steps:

a) Partition the hyperspectral data set so that the number of rows in each partition is proportional to the values of $\{\alpha_i\}_{i=1}^{p}$ assuming that no upper bound exists on the number of pixels that can be stored by the processor.
b) For each processor, check if the number of pixel vectors assigned to it is greater than the upper bound. For all the processors whose upper bounds are exceeded, assign them a number of pixels equal to their upper bounds. Now, we solve the partitioning problem of a set with remaining pixel vectors over the remaining processors. We recursively apply this procedure until all the elements have been assigned.

It should be noted that a homogeneous version of the algorithm above can be obtained by replacing step 4, so that $\alpha_i = p/w_i$ for all $i \in \{1, \cdots, p\}$, where w_i is a constant communication speed between each processor pair.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}
0.0072	0.0102	0.0206	0.0072	0.0102	0.0058	0.0072	0.0102	0.0072	0.0451	0.0131	0.0131	0.0131	0.0131	0.0131	0.0131

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

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}
P_1		19.26	19.26	19.26	48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
P_2	19.26		19.26	19.26	48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
P_3	19.26	19.26		19.26	48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
P_4	19.26	19.26	19.26		48.31	48.31	48.31	48.31	96.62	96.62	154.76	154.76	154.76	154.76	154.76	154.76
P_5	48.31	48.31	48.31	48.31		17.65	17.65	17.65	48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
P_6	48.31	48.31	48.31	48.31	17.65		17.65	17.65	48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
P_7	48.31	48.31	48.31	48.31	17.65	17.65		17.65	48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
P_8	48.31	48.31	48.31	48.31	17.65	17.65	17.65		48.31	48.31	106.45	106.45	106.45	106.45	106.45	106.45
P_9	96.62	96.62	96.62	96.62	48.31	48.31	48.31	48.31		16.38	58.14	58.14	58.14	58.14	58.14	58.14
P_{10}	96.62	96.62	96.62	96.62	48.31	48.31	48.31	48.31	16.38		58.14	58.14	58.14	58.14	58.14	58.14
P_{11}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14		14.05	14.05	14.05	14.05	14.05
P_{12}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05		14.05	14.05	14.05	14.05
P_{13}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05		14.05	14.05	14.05
P_{14}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05	14.05		14.05	14.05
P_{15}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05	14.05	14.05		14.05
P_{16}	154.76	154.76	154.76	154.76	106.45	106.45	106.45	106.45	58.14	58.14	14.05	14.05	14.05	14.05	14.05	

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

4. Experimental Results

This section provides an assessment of the effectiveness of the proposed parallel algorithm using different data communication and redundant computation handling strategies, as described in subsection 3.2. In order to assess the algorithm's performance, we resort to a framework for assessment of heterogeneous algorithms recently proposed by Lastovetsky and Reddy [17], who stated that a heterogeneous algorithm cannot be executed on a heterogeneous environment faster than its homogeneous prototype on the equivalent homogeneous environment. In [17], a homogeneous computing environment was considered equivalent to the heterogeneous one if: 1) both environments have the same number of processors; 2) the speed of each processor in the homogeneous environment is equal to the average speed of processors in the heterogeneous environment; and 3) the aggregate communication characteristics of the homogeneous environment are the same as those of the heterogeneous environment. As a result, the heterogeneous algorithm may be considered optimal if its efficiency is the same as that of the homogeneous prototype. This strategy is appropriate in morphological hyperspectral imaging, where the proposed heterogeneous algorithm is a modification of some homogeneous one.

4.1. Parallel Computing Architectures

This subsection provides an overview of the heterogeneous and homogeneous parallel computing architectures used for evaluation purposes in this work. For the design of experiments, we have considered three clusters of workstations. The first one is a small-scale heterogeneous network of 16 different SGI, Solaris and Linux workstations, and four communication segments at University of Maryland. Table 1 shows the cycle-times of the heterogeneous processors, where processors $\{P_i\}_{i=1}^{4}$ are attached to communication segment s_1, processors $\{P_i\}_{i=5}^{8}$ communicate through s_2, processors $\{P_i\}_{i=9}^{10}$ are interconnected via s_3, and processors $\{P_i\}_{i=11}^{16}$ share communication segment s_4. The communication links between the different segments $\{s_j\}_{j=1}^{4}$ only support serial communication. For illustrative purposes, Table 2 shows the capacity of all point-to-point communications, expressed as the time in milliseconds to transfer a one-megabit message between each processor pair (P_i, P_j) in the heterogeneous system. As it can be deduced from Table 2, the communication network of the heterogeneous

platform consists of four relatively fast homogeneous communication segments interconnected by three slower communication links with capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$, $c^{(3,4)} = 58.14$ in milliseconds, respectively. Although this is a simple architecture, it is also a quite typical and realistic one as well.

The second parallel computing architecture used in experiments is a homogeneous cluster of 16 identical Linux workstations which is considered to be equivalent to the heterogeneous one. The processor cycle-time of the 16 processors in this architecture is $w = 0.0131$ seconds per megaflop, and they are interconnected via a homogeneous network with capacity $c = 26.64$ milliseconds. It should also be noted that the same processors $\{P_i\}_{i=11}^{16}$ in the heterogeneous cluster were also used to construct the homogeneous cluster, which allowed us to better control the accuracy of experiments by ensuring that these processors have the same speed in the homogeneous cluster running an homogeneous prototype and in the heterogeneous cluster running its corresponding heterogeneous algorithm. It is also important to emphasize that the configuration of the two platforms above was custom-designed to make sure that the two architectures are approximately equivalent in the context of our specific heterogeneous application, as detailed in [17].

Finally, In order to test the heterogeneous algorithm on a larger-scale parallel platform, we have also experimented with Thunderhead, a Beowulf cluster located at NASA's Goddard Space Flight Center (GSFC). Beginning in the early nineties, the overwhelming computational needs of Earth and space scientists have driven GSFC to be one of the leaders in the application of low cost high-performance computing to remote sensing problems. In 1997, the HIVE (Highly Parallel Virtual Environment) project was started to build a homogeneous commodity cluster intended to be exploited by different users in a wide range of scientific applications. The Thunderhead system can be seen as an evolution of the HIVE project. It is currently composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 GB of memory and 80 GB of main memory. The total peak performance of the system is 2457.6 GFlops. Despite the computational power offered by Thunderhead, a current trend at GSFC and other NASA centers is to exploit highly heterogeneous, massively parallel computing platforms able to operate in large-scale distributed environments.

4.2. Performance Analysis

The parallel algorithm in section 3 was applied to a hyperspectral scene collected by the AVIRIS hyperspectral imager, using seven different square-shaped structuring elements, i.e., B_{3x3}, B_{5x5}, B_{7x7}, B_{9x9}, B_{11x11}, B_{13x13} and B_{15x15}. The full data set, with dimensions of 1024 samples by 614 lines and 224 spectral bands (around 280 MB) was acquired over the well-known Indian Pines region, a mixed forest/agricultural test site, and represents a very challenging classification problem due to the fact that most of the classes are dominated by mixed pixels. Extensive ground-truth information is available for the area along with ground-truth information, as shown by Fig. 4. This map was preliminary used to validate the accuracy of our proposed morphological algorithm combined with a spectral mixture analysis approach, in which the procedure described in section 2 was first used to identify the most spectrally pure pixels in the data set according to the resulting MEI scores. As a result, a set of 30 representative pure pixels (one per ground-truth class) was identified, and each pixel was labeled as belonging to the class given by the most abundant sub-pixel component (this strategy is known as winner-take-all in the literature [1]). For illustrative purposes, Table 3 shows the classification accuracy scores (in terms of the percentage of correctly classified pixels) obtained using the procedure above with the seven considered structuring elements, where the most appropriate structuring element seemed to be B_{13x13}.

B_{3x3}	B_{5x5}	B_{7x7}	B_{9x9}	B_{11x11}	B_{13x13}	B_{15x15}
65.34	73.48	80.29	84.05	90.13	90.55	90.96

Table 3. Percentages of correctly classified pixels in the AVIRIS scene by a winner-take-all strategy based on the proposed morphological processing using different structuring elements.

Fig. 5(a) plots the speedup of the heterogeneous algorithms over their corresponding homogeneous prototypes on the heterogeneous platform as a function of R/W, where the three considered data communication strategies are respectively labelled as HMP-A, i.e., overlap communication for every single pixel; HMP-B, i.e., overlap communication to have all data available before the morphological filtering; and HMP-C, i.e., sending the overlap border data as part of the scatter operation itself. Results in Fig. 5(a) show that heterogeneous algorithms were several times faster than their homogeneous versions, in particular, those labelled as HMP-B and HMP-C. The speedup was simply calculated as the execution time of the homogeneous algorithm divided by the execution time of the heterogeneous algorithm for the same R/W ratio. The main reason why the HMP-A algorithm performed less effectively is due to its very expensive communication strategy, while HMP-B implemented a better communication strategy, and HMP-C was usually about as good as strategy B.

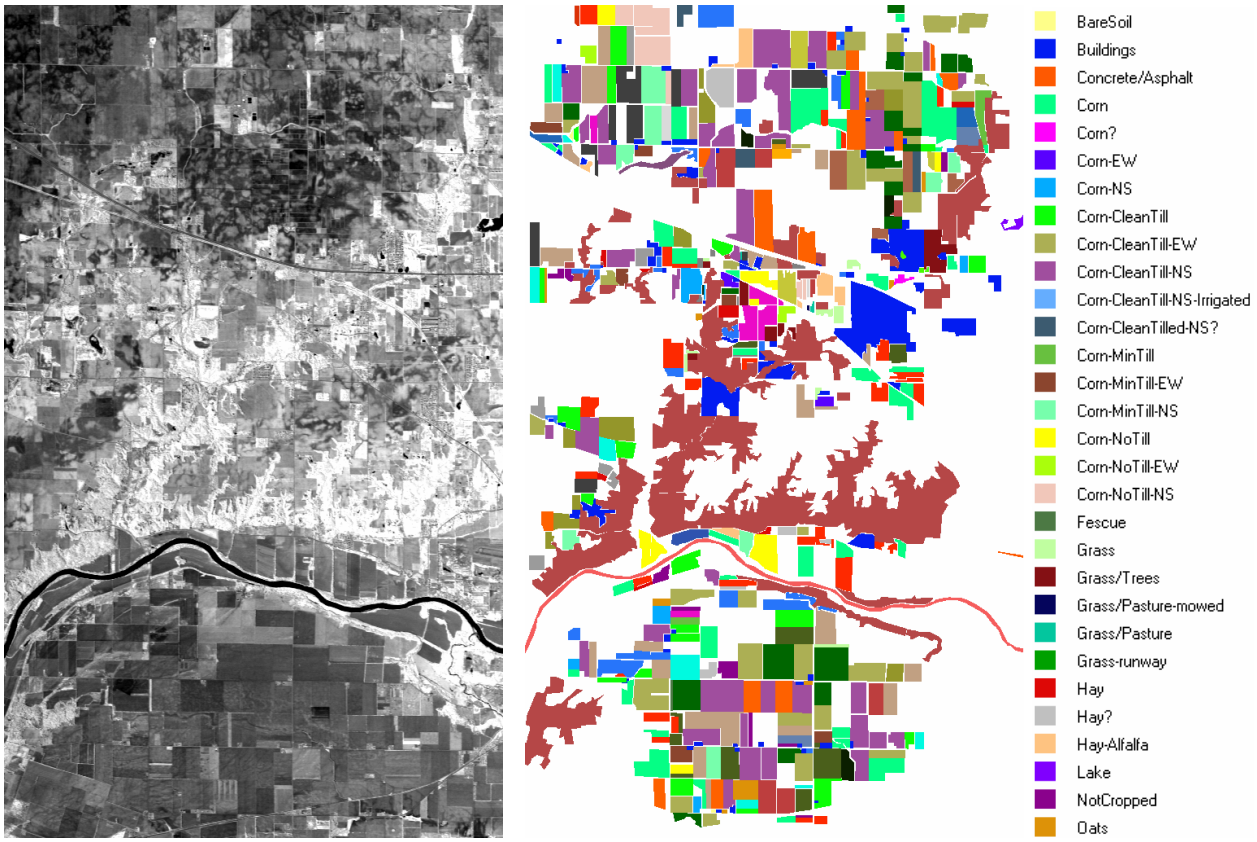

Figure 4. (Left) Spectral band at 587 nm wavelength of an AVIRIS scene comprising agricultural and forest features at Indian Pines, Indiana. (Right) Ground-truth map with 30 mutually exclusive land-cover classes.

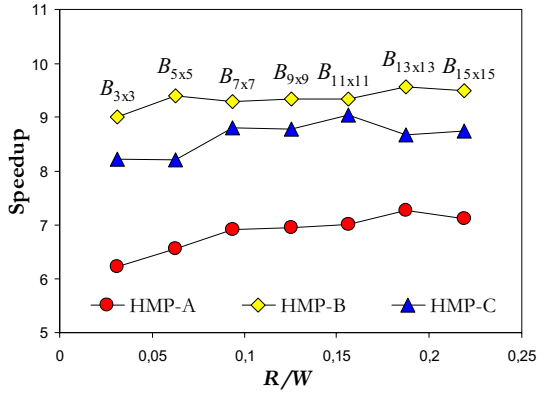

Figure 5. Speedup achieved by the heterogeneous algorithms over their corresponding homogeneous counterparts on a heterogeneous cluster at University of Maryland.

Fig. 5 plots the speedup of the heterogeneous algorithms over their corresponding homogeneous prototypes on the heterogeneous platform as a function of R/W, where the three considered data communication strategies are respectively labeled as HMP-A, i.e., overlap communication for every single pixel; HMP-B, i.e., overlap communication to have all data available before the morphological filtering; and HMP-C, i.e., sending the overlap border data as part of the scatter operation itself. Results in Fig. 5 show that heterogeneous algorithms were several times faster than their homogeneous versions, in particular, those labeled as HMP-B and HMP-C. The speedup was simply calculated as the execution time of the homogeneous algorithm divided by the execution time of the heterogeneous algorithm for the same R/W ratio. The reason why the HMP-A algorithm performed less effectively is due to its expensive communication strategy, while HMP-B implemented a better communication strategy, and HMP-C implemented a strategy that was usually about as good as strategy B. Similarly, Fig. 6 shows the speedup of the homogeneous algorithms over the heterogeneous ones on the homogeneous platform as a function of R/W. Results in Fig. 6 demonstrate that the homogeneous algorithms only slightly outperformed the heterogeneous ones for small structuring elements when the computing platform was also homogeneous. However, as the volume of computation increased (which is often a requirement in light of results in Table 3), heterogeneous algorithms achieved very similar performance to their respective homogeneous counterparts.

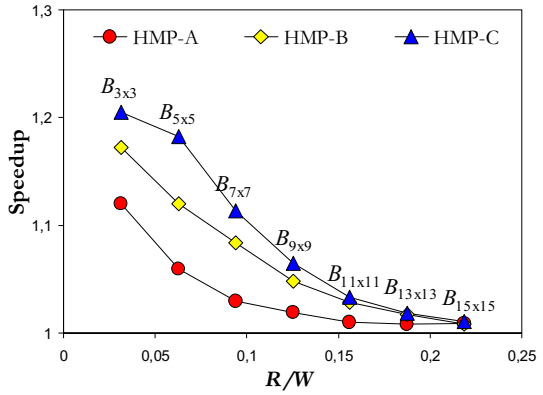

Figure 6. Speedup achieved by the homogeneous algorithms over their corresponding heterogeneous algorithms on a homogeneous cluster which is considered equivalent to the heterogeneous one.

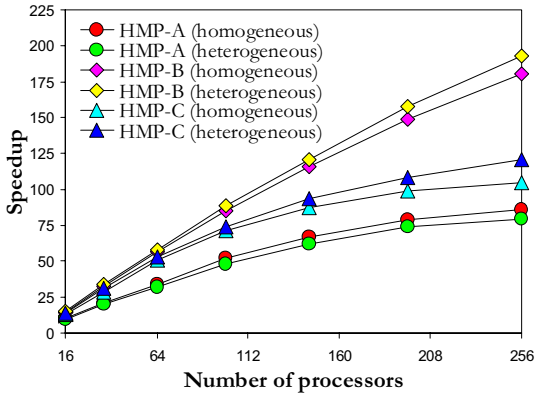

Figure 7. Scalability of the heterogeneous and homogeneous algorithms on Thunderhead, using a square-shaped structuring element of 13x13 pixels as a case study for demonstration.

This result demonstrates the flexibility of the proposed heterogeneous algorithms, which were able to adapt much better to homogeneous computing environments, in particular, when the volume of computations involved is very large. Quite opposite, homogeneous algorithms could not effectively adapt to a heterogeneous computing scenario, as demonstrated by results in Fig. 5. This is mainly due to a less efficient workload distribution among the heterogeneous workers. To analyze this relevant issue in more detail, a study of load balance is highly required to fully substantiate the parallel properties of the considered algorithms.

In order to explore load balance, Table 4 shows the imbalance scores achieved by the different algorithms (implemented with I_{max} set to 5 iterations). The imbalance is defined as $D = R_{max} / R_{min}$, where R_{max} and R_{min} are the maxima and minima processor run times, respectively. Therefore, perfect balance is achieved when $D = 1$. In the table, we report the imbalance considering all processors, D_{All}, and also considering all processors but the root, D_{Minus}. In all cases, load balance was similar when the root processor was not included, which means that the master node does not have high computation load. It is also clear from Table 4 that the homogeneous algorithms executed on the heterogeneous network provided the highest values of D_{All} and D_{Minus} (and hence the highest imbalance), while the heterogeneous algorithms always resulted in values of D_{All} and D_{Minus} which were closer to 1, regardless of the platform where they were run. More specifically, it can be seen from Table 4 that the HMP-B implementation was the only heterogeneous algorithm which was able to provide values of D_{All} and D_{Minus} which were always very similar, while HMP-C provided slightly less similar scores.

Taking in mind the results described above, and with the ultimate goal of exploring issues of scalability and portability of the proposed heterogeneous algorithms to existing massively parallel computing platforms at NASA (which are mainly homogeneous in nature), we have also compared the performance of the proposed heterogeneous algorithms (and their homogeneous counterparts) on Thunderhead. Fig. 7 shows the speedups achieved by the heterogeneous algorithms and their homogeneous versions over a single-processor run of the sequential morphological algorithm on Thunderhead, as a function of the number of processors, using a structuring element size of 13x13 pixels (measured processing times are also reported on Table 5). As Fig. 7 shows, the scalability of the heterogeneous algorithms was similar to that achieved by their homogeneous prototypes, in particular, for both HMP-B and HMP-C algorithms. It should also be noted that a processing time of 40 seconds was measured for HMP-B when all available processors on Thunderhead were used. This is a relevant achievement given the extremely high dimensionality of the considered scene, in particular, if we take into account that more than two hours of computation (7267 seconds) were required to process the full hyperspectral data set using a single Thunderhead processor. Overall, experimental results in this section revealed that parallel algorithms based on heterogeneous computing paradigms offer a simple, relatively platform-independent and scalable solution in the context of high-dimensional imaging applications.

5. Conclusions and Future Research

This paper provided an investigation of parallel techniques to extract relevant information from hyperspectral image data sets in highly heterogeneous computing environments.

	HMP-A		HMP-B		HMP-C	
Algorithm	D_{All}	D_{Minus}	D_{All}	D_{Minus}	D_{All}	D_{Minus}
Homogeneous versions	1.31	1.15	1.07	1.03	1.12	1.04
Heterogeneous algorithms	1.28	1.13	1.06	1.02	1.10	1.03

Table 4. Load-balancing rates for heterogeneous algorithms (and homogeneous versions) in the heterogeneous cluster.

	HMP-A		HMP-B		HMP-C	
Number of processors	Execution time	Speedup	Execution time	Speedup	Execution time	Speedup
4	3460	2.01	1990	3.65	2344	3.12
16	741	9.18	490	14.81	549	13.23
36	357	20.03	219	33.12	256	28.34
64	224	32.24	128	56.67	141	51.23
100	151	48.35	85	85.23	102	71.23
144	117	62.34	62	116.23	83	87.34
196	98	74.23	48	149.56	73	99.86
256	90	80.32	40	180.28	69	105.23

Table 5. Execution time in seconds and speedup factors for heterogeneous algorithms on Thunderhead.

Experimental results demonstrated that heterogeneous networks of workstations represent a cost-effective way of exploiting parallelism in spatial/spectral algorithms such as those based on mathematical morphology, which represent a most advanced generation of algorithms in hyperspectral imaging. It has been shown that parallel computing at the massively parallelism level provides a unique framework to extract information in near real-time and with adequate reliability in an environment dominated by heterogeneous processing components. The proposed parallel framework is particularly suitable for data mining applications that previously looked to be too computationally intensive in practice, due to immense files and data archives common to remote sensing problems. Combining this readily available computational power with the new sensor instruments may introduce major changes in the systems used by NASA and other agencies for exploiting Earth and planetary remotely sensed data. We feel that the applicability of the techniques described in this paper extend beyond the domain of high-dimensional image processing. This is particularly true for the domains of signal processing and linear algebra applications, which include similar patterns of communication and calculation.

Acknowledgement

This research was supported by the Spanish Ministry of Education and Science through Fellowship PR2003-0360, which allowed the author to conduct research on parallel computing as postdoctoral research associate at NASA's Goddard Space Flight Center and University of Maryland. Funding from the European Commission through the project "Performance analysis of hyperspectral analysis algorithms" (contract no. HPRI–1999–00057) and additional funding from Junta de Extremadura (local government) through project 2PR03A026 are also acknowledged. The author would like to thank Drs. J. E. Dorband, J. C. Tilton and J. A. Gualtieri for their collaboration in experiments on the Thunderhead cluster, and Profs. M. Valero and F. Tirado for their support.

References

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

[2] 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.

[3] A. Plaza and C.-I Chang, *High-Performance Computing in Remote Sensing*, Chapman & Hall/CRC Press, to appear in 2006.

[4] J. A. Gualtieri and J. C. Tilton, "Hierarchical segmentation of hyperspectral data," *XI NASA/Jet Propulsion Laboratory Airborne Earth Science Workshop*, Pasadena, CA, USA, 2002.

[5] A. Plaza, D. Valencia, J. Plaza and P. Martínez, "Commodity cluster-based parallel processing of hyperspectral imagery," *Journal of Parallel and Distributed Computing*, in press, to appear in 2006.

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

[7] V. E. Bazterra, M. Cuma, M. B. Ferraro, J. C. Facelli, "A general framework to understand parallel performance in heterogeneous clusters: analysis of a new adaptive parallel genetic algorithm," *Journal of Parallel and Distributed Computing*, vol. 65, pp. 48-57, 2005.

[8] O. Beaumont, V. Boudet, F. Rastello and Y. Robert, "Matrix multiplication on heterogeneous platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, pp. 1033-1051, 2001.

[9] P. Boulet, J. Dongarra, F. Rastello, Y. Robert and F. Vivien, "Algorithmic issues on heterogeneous computing platforms," *Parallel Processing Letters*, vol. 9, pp. 197-213, 1999.

[10] F. J. Seinstra, D. Koelma and J. M. Geusebroek, "A software architecture for user transparent parallel image processing," *Parallel Computing*, vol. 28, pp. 967-993, 2002.

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

[12] A. Plaza, P. Martinez, R.M. Perez, J. Plaza, "A new approach to mixed pixel classification of hyperspectral imagery based on extended morphological profiles," *Pattern Recognition*, vol. 37, pp. 1097-1116, 2004.

[13] P. Soille, *Morphological Image Analysis: Principles and Applications*, *2nd ed.* Springer: Berlin, 2003.

[14] 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*, pp. 995-1000, 2005.

[15] M. Prieto, I. M. Llorente and F. Tirado, "Data locality exploitation in the decomposition of regular domain problems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, pp. 1141-1149, 2000.

[16] F. J. Seinstra and D. Koelma, "P-3PC: A point-to-point communication model for automatic and optimal decomposition of regular domain problems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 758-768, 2002.

[17] A. Lastovetsky and R. Reddy, "On performance analysis of heterogeneous algorithms," Parallel Computing, vol. 30, pp. 1195-1216, 2004.