Heterogeneous Parallel Computing in Remote Sensing Applications: Current Trends and Future Perspectives

Antonio J. Plaza
Computer Science department, University of Extremadura
Avda. de la Universidad s/n, E-10071 Cáceres, Spain
Contact e-mail: aplaza@unex.es

Abstract

Heterogeneous networks of computers have rapidly become a very promising commodity computing solution, expected to play a major role in the design of high performance computing systems for remote sensing missions. Currently, only a few parallel processing strategies are available in this research area, and most of them assume homogeneity in the underlying computing platform. This paper develops several highly innovative heterogeneous parallel algorithms for information extraction from high-dimensional remotely sensed images, with particular emphasis on target detection and land-cover mapping applications. Experimental results are presented in the context of a realistic application, using real data collected by NASA's Jet Propulsion Laboratory over the World Trade Center complex in New York City after September 11th, 2001. Parallel performance of the proposed algorithms is discussed using several (fully and partially) heterogeneous networks at University of Maryland, and a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center. Combined, these parts deliver a snapshot of the state-of-the-art in those areas, and a thoughtful perspective on the potential and challenges of applying heterogeneous computing practices to remote sensing problems.

1. Introduction

The interest in remote sensing applications and platforms has grown dramatically in recent years. Remote sensing technology has shifted from panchromatic (a wide range of wavelengths merged into a single response) to hyperspectral data [3], with hundreds or thousands of spectral bands. For example, the NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) [6] records the visible and the near-infrared spectrum (wavelength region from 0.4 to 2.5 micrometers) of the reflected light of an area 2 to 12 kilometers wide and several kilometers long –depending on the duration of the flight– using 224 spectral bands. As a result, a hyperspectral 'image cube' is typically a stack of hundreds of images collected at different wavelengths, in which each pixel (vector) has an associated spectral signature or 'fingerprint' that can be accurately used to uniquely characterize the underlying objects. The resulting data volume typically exceeds 500 MB per flight. The widespread availability of Earth observation sensors such as AVIRIS has opened groundbreaking perspectives in many applications, including target detection for military and defense/security deployment, risk/hazard prevention and response including wild-land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination.

Despite the growing interest in hyperspectral imaging, only a few efforts devoted to the design of parallel processing techniques currently exist in the open literature [7, 8, 5]. In addition, most dedicated parallel machines used for hyperspectral imaging have been homogeneous in nature [4]. However, a current trend in the design of systems for information extraction and mining from high-dimensional remote sensing data repositories is to utilize heterogeneous, distributed platforms made up of local (user) computing resources [17]. In particular, heterogeneous computing [10] greatly benefits from the considerable amount of work done in dynamic, resource-aware static and dynamic task scheduling and load balancing in distributed platforms, including large-scale distributed systems [18, 12, 2, 1]. To address the need for cost-effective heterogeneous algorithms in this emerging new area, this paper takes a necessary first step toward the comparison of different techniques and strategies for parallel hyperspectral image analysis on distributed, highly heterogeneous computing platforms.

The remainder of the paper is structured as follows. Section 2 describes several new heterogeneous parallel algorithms for target detection and land-cover mapping. Section 3 first describes the hyperspectral data sets and parallel computing platforms used in this study, and then evaluates

1-4244-0328-6/06/$20.00 ©2006 IEEE.

the parallel performance of the proposed algorithms by using several heterogeneous and homogeneous networks of workstations. For comparative purposes, a massively parallel Beowulf cluster at NASA's Goddard Space Flight Center is also used in experiments. The parallel algorithms are discussed in the context of a specific case study, focused on detecting fires and mapping debris compositions on a real hyperspectral data set collected by the AVIRIS sensor over the World Trade Center area in New York City after September 11, 2001. Section 4 concludes with some remarks and hints at plausible future research.

2. Parallel algorithm design

Before describing our parallel heterogeneous algorithms we first formulate a general optimization problem in the context of fully heterogeneous networks. This kind of platform can be modeled as a complete graph $G = (P, E)$, where each node models a computing resource p_i weighted by its relative cycle-time w_i. Each edge in the graph models a communication link weighted by its relative capacity, where c_{ij} denotes the maximum capacity of the slowest link in the path of physical communication links from p_i to p_j (we assume that the system has symmetric costs, i.e., $c_{ij} = c_{ji}$. With the above assumptions in mind [12], processor p_i should accomplish a share of $\alpha_i \cdot W$ of the total workload, denoted by W, to be performed by a certain algorithm, with $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$. Taking into account the standard optimization problem above, an abstract view of our proposed hyperspectral image processing framework can be simply given in the form of a client-server architecture, in which a server processor is responsible for the efficient distribution of work among the P nodes, and the clients operate with the spectral signatures contained in a local partition. The partitions are updated locally and, depending on the considered algorithm, the resulting calculations may also be exchanged between the clients, or between the server and the clients.

2.1 Data partitioning strategies

In a data-driven application environment such as the one described above, it is important to define efficient data partitioning strategies [15]. Two standard approaches have been traditionally considered for this purpose [14]:

- *Spectral-domain partitioning*. This approach subdivides the multi-channel remotely sensed image into small cells or sub-volumes made up of contiguous spectral wavelengths.
- *Spatial-domain partitioning*. This approach breaks the multi-channel image into slices made up of one or several contiguous spectral bands.

In both cases, each pixel (vector) may be split amongst several processors, and the calculations made for each spectral signature would need to originate from several processors, thus increasing the cost of communications (which can be significant in fully heterogeneous platforms made up of communication links with different capacities). In this work, we adopt a hybrid strategy, in which the data is partitioned into blocks made up of spatially adjacent pixel vectors which retain the full spectral content associated to them. This approach has several advantages [14]. First and foremost, the application of a hybrid partitioning provides a *natural* approach for low-level image processing, as it generally involves a kernel which is repeatedly applied to small set of neighboring pixels within the image data structure [15]. A second major reason has to do with the cost of interprocessor communication, as stated above.

In order to balance the load of the processors in the heterogeneous environment, each processor should execute an amount of work that is proportional to its speed. Therefore, two major goals of our partitioning algorithm are: i) to obtain an appropriate set of workload fractions $\{\alpha_i\}_{i=1}^{P}$ that best fit the heterogeneous environment, and ii) to translate the chosen set of values into a suitable decomposition of the input data, taking into account the properties of the heterogeneous system. To accomplish the above goals, we use a workload estimation algorithm (WEA) that assumes that the workload of each processor p_i must be directly proportional to its local memory and inversely proportional to its cycle-time w_i.

Algorithm 1. Workload estimation algorithm (WEA)

```
Input: Hyperspectral image cube F.
Output: P spatial-domain partitions of the input data.
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. In order to obtain the value of $\alpha_i$ for each processor $p_i$, calculate $\alpha_i = \lfloor \frac{(1/w_i)}{\sum_{i=1}^{P}(1/w_i)} \rfloor$ for all $i \in \{1, \cdots, P\}$.
3. Use the calculated values of $\{\alpha_i\}_{i=1}^{P}$ to produce $P$ partitions of the input hyperspectral data set F as follows:
    (a) Obtain a first partitioning of F so that the number of rows in each partition is proportional to the values of $\{\alpha_i\}_{i=1}^{P}$ and assuming that no upper bound exists on the number of pixel vectors that can be stored in the local memory associated to the processor. If the number of pixel vectors in each partition assigned to each $p_i$ is less than the upper bound, we have an optimal distribution and the partitioning algorithm is terminated.
```

(b) For each p_i, check if the number of pixel vectors assigned to it is greater than the upper bound on the number of elements that it can store in its local memory. For all the processors whose upper bounds are exceeded, we recursively apply this procedure until all the elements have been assigned. In the end, the number of pixel elements stored at each processor is proportional to its cycle-time and memory capacity.

It should be noted that a homogeneous version of the WEA algorithm can be obtained by simply replacing Step 3 with $\alpha_i = 1/w_i$ for all $i \in \{1, \cdots, P\}$, where w_i is assumed to be a constant cycle-time for all processors in the homogeneous system.

2.2 Algorithm descriptions

In this section, we provide algorithmic descriptions of parallel heterogeneous algorithms for target detection and classification of remotely sensed hyperspectral image scenes.

2.2.1 Target detection algorithms

Two heterogeneous parallel target detection algorithms are described in this section: automated target detection and classification algorithm (Hetero-ATDCA) and unsupervised fully constrained least squares (Hetero-UFCLS) algorithm. The HeteroATDCA algorithm [**?**] finds a set of spectrally distinct target pixels vectors using orthogonal subspace projections in the spectral domain. Below, we provide a step-by-step description of our parallel version of this algorithm, specifically adapted to heterogeneous environments:

Algorithm 2. Hetero-ATDCA

Input: Hyperspectral image $\mathbf{F}$, number of targets t.

Output: Set of t target pixels.

1. Using the WEA algorithm, the master divides the original image cube $\mathbf{F}$ into P heterogeneous partitions. Then, the master sends the partitions to the workers.
2. Each worker finds the brightest pixel in its local partition using $\mathbf{t}_i^{(1)} = arg\{max_{(x,y)}\mathbf{F}(x,y)^T \cdot \mathbf{F}(x,y)\}$, where the superscript T denotes the vector transpose operation and $i = 1, 2, \cdots, P$. Each worker then sends the spatial locations of the pixel identified as the brightest ones in its local partition back to the master.
3. Once all the workers have completed their parts, the master finds the brightest pixel of the input scene, $\mathbf{t}^{(1)}$, by applying the $argmax$ operator in step 2 to all the pixels at the spatial locations provided by the workers, and selecting the one that results in the maximum score. Then, the master sets $\mathbf{U} = \mathbf{t}^{(1)}$ and broadcasts this matrix (with $1 \times N$ dimensions) to all workers.
4. Each worker finds (in parallel) the pixel in its local partition with the maximum orthogonal projection relative to the pixel vectors in $\mathbf{U}$, using a projector given by $P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$, where $\mathbf{I}$ is the identity matrix. The orthogonal space projector $P_{\mathbf{U}}^{\perp}$ is now applied to all pixel vectors in each local partition to obtain $t_i^{(2)} = argmax_{(x,y)}\{(P_{\mathbf{U}}^{\perp}\mathbf{F}(x,y))^T(P_{\mathbf{U}}^{\perp}\mathbf{F}(x,y))\}$. Each worker then sends the spatial location of the resulting local pixels to the master node.
5. The master now finds a second target pixel, $\mathbf{t}^{(2)}$, by applying $P_{\mathbf{U}}^{\perp}$ to the pixel vectors at the spatial locations provided by the workers, and selecting the one which results in the maximum score. The master now sets $\mathbf{U} = \{\mathbf{t}^{(1)}, \mathbf{t}^{(2)}\}$ and broadcasts this matrix (now with $2 \times N$ dimensions) to all workers.
6. Repeat from step 4 until a set of t target pixels $\{\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, \cdots, \mathbf{t}^{(t)}\}$ (where the value of t given as an input parameter), are extracted from a matrix $\mathbf{U}$ made up of $t \times N$ dimensions.

As can be seen from the algorithmic description given above, the Hetero-ATDCA assumes that all pixel vectors in the input data are represented by a single underlying substance. In contrast, the Hetero-UFCLS [3] generates a set of t targets using the concept of least square-based error minimization.

Algorithm 3. Hetero-UFCLS

Input: Hyperspectral image $\mathbf{F}$, number of targets t.

Output: Set of t target pixels.

1. Execute steps 1-3 of the Hetero-ATDCA algorithm to obtain $\mathbf{t}^{(1)}$, the brightest pixel of the input scene, and broadcast $\mathbf{U} = \mathbf{t}^{(1)}$ to all the workers.
2. Each worker forms a local error image $E^{(i)}$ by calculating the least squares-based error for each pixel vector in the input data represented in terms of a fully constrained linear mixture of all the spectra in $\mathbf{U}$, as shown in [3].
3. Each worker then finds the pixel $\mathbf{F}(x, y)$ in the local partition with the largest associated score in the error image $E^{(i)}$. The spatial coordinates of this pixel (and its associated error score) are sent back to the master.

4. The master obtains a second target $\mathbf{t}^{(2)}$ by selecting the pixel vector with largest associated error score from the pixel vectors at the spatial locations provided by the workers and broadcasts $\mathbf{U} = \{\mathbf{t}^{(1)}, \mathbf{t}^{(2)}\}$ to the workers.

5. Repeat from step 4 to incorporate a new target pixel $\mathbf{t}^{(3)}, \mathbf{t}^{(4)}, \cdots, \mathbf{t}^{(t)}$ to $\mathbf{U}$ until a set of t target pixels have been extracted.

2.3 Classification algorithms

In this section, we first develop a heterogeneous classification algorithm based on the principal component transform (PCT). This technique has been widely used as a standard spectral transformation which is used to summarize and decorrelate the information present in the data by reducing redundancy and packing the residual information into a small set of images, termed principal components [9]. In order to implement this algorithm, we have used the spectral angle distance (SAD) as a baseline. SAD is a widely used metric in hyperspectral analysis [3] which can be used to measure the spectral similarity between two pixel vectors at different discrete spatial locations, $\mathbf{F}(x, y)$ and $\mathbf{F}(i, j)$, where (x, y) and $(i, j) \in Z^2$, as follows:

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

Algorithm 4. Hetero-PCT

Input: Hyperspectral image $\mathbf{F}$, number of classes c.

Output: Classification label for each pixel $\mathbf{F}(x, y)$.

1. Using the WEA algorithm, the master divides the original image cube $\mathbf{F}$ into P heterogeneous partitions. Then, the master sends the partitions to the workers.

2. Each worker forms a unique spectral set by calculating the SAD distance for all vector pairs.

3. The P unique sets are sent back to the master and combined, one pair at a time. Upon completion, there will be only one unique set left made up of c pixel vectors.

4. An N-dimensional mean vector $\mathbf{m}$ is calculated concurrently, where each component is the average of the pixel values of each spectral band of the unique set. This vector is formed once all the processors finish processing their respective parts.

5. All the pixel vectors in the unique set are divided into P parts and sent to the workers. Each worker then computes the covariance component and forms a covariance sum.

6. The covariance matrix is calculated sequentially as the average of the matrices calculated in Step 5.

7. A transformation matrix $\mathbf{T}$ is obtained by calculating and sorting the eigenvectors of the covariance matrix according to their corresponding eigenvalues, which provide a measure of their variances. Since the degree of data dependency of the calculation is high and its complexity is related to the number of spectral bands rather than the image size, this step is also done sequentially at the master.

8. Each pixel in the original hyperspectral image us transformed independently using $\mathbf{T} \cdot (\mathbf{F}(x, y) - \mathbf{m})$. This step is done in parallel, where all workers transform their respective portions of data concurrently.

9. Finally, a parallel post-processing step is applied to perform classification in the PCT-transformed space. First, P partitions of a reduced data cube given by the first c components of the PCT-transformed image are sent to the workers, along with the spatial locations of the c unique pixel vectors resulting from Step 2. Each worker then labels each pixel in its corresponding partition using a classification label given by the most spectrally similar unique pixel vector in the PCT-reduced space, and sends back the result to the master, which puts together the final label image.

As shown by the above algorithm, the PCT considers the hyperspectral data not as an image, but as an unordered listing of spectral measurements where the spatial coordinates can be shuffled arbitrarily without affecting the analysis. Below, we introduce a spatial/spectral classification algorithm which is based on the definition of a cumulative distance between each pixel vector, $\mathbf{F}(x, y)$, and all pixel vectors in the spatial neighborhood of $\mathbf{F}(x, y)$ given by a spatial kernel or structuring element B as follows [**?**]:

$$D_B(\mathbf{F}(x, y)) = \sum_{(i,j) \in Z^2(B)} SAM(\mathbf{F}(x, y), \mathbf{F}(i, j)) \quad (2)$$

where (i, j) are the spatial coordinates in the B-neighborhood discrete domain, represented by $Z^2(B)$. Based on the cumulative distance metric above, two basic morphological operations (called erosion and dilation [16, 13]) are defined as follows:

$$(\mathbf{F} \ominus B)(x, y) = argmin_{(i,j) \in Z^2(B)} \{D_B(\mathbf{F}(x + i, y + j))\} \quad (3)$$

$$(\mathbf{F} \oplus B)(x, y) = argmax_{(i,j) \in Z^2(B)} \{D_B(\mathbf{F}(x + i, y + j))\} \quad (4)$$

These operations can be respectively used to extract the most highly mixed and the most highly pure pixel in the B-given spatial neighborhood. Using the two standard operations above, we provide below a new parallel heterogeneous morphological algorithm for unsupervised classification of hyperspectral imagery. This algorithm introduces redundant computations (in the form of overlap borders) to reduce inter-processor communication when the kernel computation is split among two heterogeneous processors.

Algorithm 5. Hetero-MORPH

Inputs: Hyperspectral image $\mathbf{F}$, structuring element B, Number of iterations I_{max}, Number of classes c.

Output: Classification label for each pixel $\mathbf{F}(x, y)$.

1. Using the WEA algorithm, the server divides the original image cube $\mathbf{F}$ into P heterogeneous partitions with overlap borders to avoid accesses outside the local image domain at each workstation [14].

2. Using parameters I_{max}, B and c, each worker executes the following steps:

   (a) Set $j = 1$ and initialize a morphological eccentricity index score MEI$(x, y) = 0$ for each pixel.

   (b) Update the MEI score at each pixel by calculating the SAD between the pixel vectors selected as 'maximum' and 'minimum' by morphological dilation and erosion operations as follows:

$$MEI(x, y) = SAD[(\mathbf{F} \ominus B)(x, y), (\mathbf{F} \oplus B)(x, y)] \tag{5}$$

   (c) Set $j = j + 1$. If $j = I_{max}$, then go to step (d). Otherwise, set $\mathbf{F} = \mathbf{F} \oplus B$ and go to step (b).

   (d) Select the set of c pixel vectors in the local partition with the highest associated MEI scores.

3. The master gathers all the individual pixel vectors provided by the workers and forms a unique spectral set of $p \leq c$ pixel vectors by calculating the SAD for all vector pairs in parallel.

4. The set of p unique pixel vectors are broadcast to all the workers, which obtain a classification label for each pixel $\mathbf{F}(x, y)$ in the local partition by assigning it to a class given by the unique pixel vector which is most highly similar to the pixel in terms of the SAD distance.

5. Each worker then sends the label associated with each local pixel to the master, which gathers all the individual results and forms the final 2-dimensional classification matrix.

3. Experimental results

Before describing our experimental results, we first provide an overview of the parallel (heterogeneous and homogeneous) computing architectures and hyperspectral data sets that will be used for evaluation purposes. Then, we thoroughly assess the parallel performance and accuracy of the proposed algorithms.

3.1 Parallel computing architectures

The parallel computing architectures used in this study comprise four networks of workstations distributed among different locations at University of Maryland and the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center (see http://thunderhead.gsfc.nasa.gov). The networks were custom-designed in order to approximate a recently proposed framework for evaluation of heterogeneous parallel algorithms [11], which relies on the assumption that a heterogeneous algorithm cannot be executed on a heterogeneous network faster than its homogeneous version on the equivalent homogeneous network. In [11], a homogeneous computing environment is considered equivalent to the heterogeneous one in light of the three following principles:

1. Both environments should have exactly the same number of processors.

2. The speed of each processor in the homogeneous environment should be equal to the average speed of processors in the heterogeneous environment.

3. The aggregate communication characteristics of the homogeneous environment should be the same as those of the heterogeneous environment.

With the above three principles in mind, a heterogeneous algorithm may be considered optimal if its efficiency on a heterogeneous network is the same as that evidenced by its homogeneous version on the equivalent homogeneous network. This allows using the parallel performance achieved by the homogeneous version as a benchmark for assessing the parallel efficiency of the heterogeneous algorithm. The four networks are considered approximately equivalent under the above framework. Their description follows:

- *Fully heterogeneous network.* Consists of 16 different workstations, and four communication segments. Table 1 shows the properties of the 16 heterogeneous workstations, where processors $\{p_i\}_{i=1}^{4}$ are attached to communication segment s_1, processors $\{p_i\}_{i=5}^{8}$ communicate through s_2, processors $\{p_i\}_{i=9}^{10}$ are interconnected via s_3, and processors $\{p_i\}_{i=11}^{16}$ share the communication segment s_4. The communication links between the different segments $\{s_j\}_{j=1}^{4}$ only support serial communication. For illustrative purposes, Table 2

Table 1. Specifications of heterogeneous processors.

Processor	Architecture	Cycle-time (secs/megaflop)	Main memory (MB)	Cache (KB)
p_1	Free BSD – i386 Intel Pentium 4	0.0058	2048	1024
p_2, p_5, p_8	Linux – Intel Xeon	0.0102	1024	512
p_3	Linux – AMD Athlon	0.0026	7748	512
p_4, p_6, p_7, p_9	Linux – Intel Xeon	0.0072	1024	1024
p_{10}	SunOS – SUNW UltraSparc-5	0.0451	512	2048
$p_{11} - p_{16}$	Linux – AMD Athlon	0.0131	2048	1024

also shows the capacity of all point-to-point communications in the heterogeneous network, expressed as the time in milliseconds to transfer a one-megabit message between each processor pair (p_i, p_j) in the heterogeneous system. As noted, the communication network of the fully heterogeneous network consists of four relatively fast homogeneous communication segments, interconnected by three slower communication links. Although this is a simple architecture, it is also a quite typical and realistic one as well.

- *Fully homogeneous network.* Consists of 16 identical Linux workstations with processor cycle-time of $w = 0.0131$ seconds per megaflop, interconnected via a homogeneous communication network where the capacity of links is 26.64 milliseconds.
- *Partially heterogeneous network.* Formed by the set of 16 heterogeneous workstations in Table 1 but interconnected using the same homogeneous communication network with capacity of 26.64 milliseconds.
- *Partially homogeneous network.* Formed by 16 identical Linux workstations with cycle-time of $w = 0.0131$ seconds per megaflop, but interconnected using the heterogeneous network shown in Table 2.

In order to test the proposed algorithms on a larger-scale parallel platform, we have also experimented with Thunderhead, a 256-node Beowulf cluster at NASA's Goddard Space Flight Center. This system is composed of 256 dual 2.4 GHz Intel Xeon nodes, each with 1 GB of main memory, 80 GB of disk space and 512 KB of cache, interconnected via 2 GHz optical fibre Myrinet. The total peak performance of the system is 2457.6 Gflops. The operating system used at the time of measurements was Linux RedHat 8.0, and MPICH was the message-passing library used.

3.2 Hyperspectral data

The image scene used for experiments in this work was collected by the AVIRIS instrument, which was flown by NASA's Jet Propulsion Laboratory over the World Trade Center (WTC) area in New York City on September 16, 2001, just five days after the terrorist attacks that collapsed the two main towers and other buildings in the WTC complex. The full data set selected for experiments consists of 2133x512 pixels, 224 spectral bands and a total size of approximately 1 GB. The spatial resolution is 1.7 meters per pixel. Fig. 1(left) shows a false color composite of the data set selected for experiments using the 1682, 1107 and 655 nm channels, displayed as red, green and blue, respectively, with a detail of the WTC area shown in a red rectangle. Vegetated areas appear green in Fig. 1(left), while burned areas appear dark gray. Smoke coming from the WTC area and going down to Battery Park at the south of Manhattan appears bright blue due to high spectral reflectance in the 655 nm channel.

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

Processor	$p_1 - p_4$	$p_5 - p_8$	$p_9 - p_{10}$	$p_{11} - p_{16}$
$p_1 - p_4$	19.26	48.31	96.62	154.76
$p_5 - p_8$	48.31	17.65	48.31	106.45
$p_9 - p_{10}$	96.62	48.31	16.38	58.14
$p_{11} - p_{16}$	154.76	106.45	58.14	14.05

At the same time of data collection, a small U.S. Geological Survey (USGS) field crew visited lower Manhattan to collect spectral samples of dust and airfall debris deposits from several outdoor locations around the WTC area (see http://speclab.cr.usgs.gov/wtc). These spectral samples were then mapped into the AVIRIS data using reflectance spectroscopy and chemical analyses in specialized USGS laboratories. For illustrative purposes, Fig. 1(right) shows a thermal map centered at the region where the buildings collapsed. The map shows the target locations of the thermal hot spots, shown as bright red, orange and yellow spots on Fig. 1(right). The temperatures range from 700F (marked as 'F') to 1300F (marked as 'G'). This thermal map, along with a USGS dust/debris map (available online from http://pubs.usgs.gov/of/2001/ofr-01-0429) are used in this work as ground-truth to validate the detection/classification accuracy of the parallel algorithms.

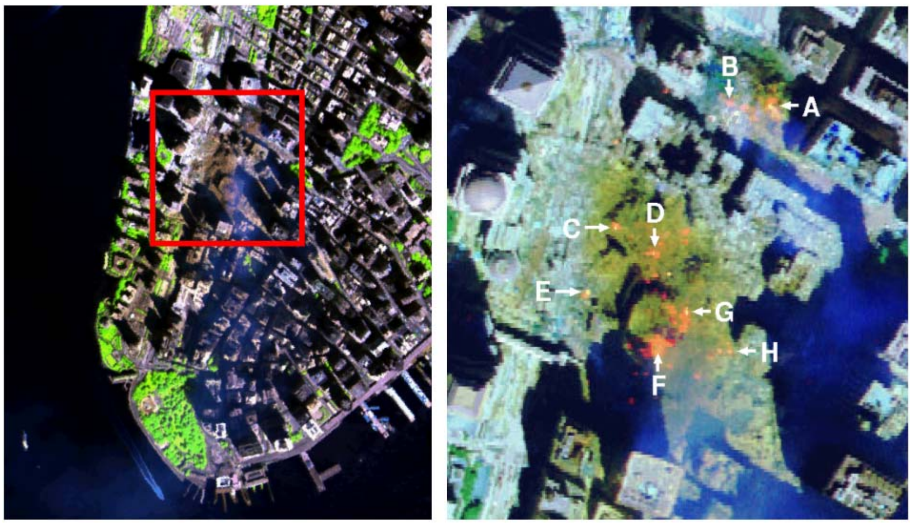

Figure 1. False color composite of an AVIRIS hyperspectral image collected over lower Manhattan on Sept. 16, 2001 (left) and location of thermal hot spots in the World Trade Center fires (right).

3.3 Performance evaluation

Before empirically investigating the parallel performance of the proposed heterogeneous parallel algorithms, we first evaluate their target detection and classification accuracy in the context of the considered application. Table 3 shows the SAD between the most similar target pixels detected by Hetero-ATDCA and Hetero-UFCLS and the pixel vectors at the known target positions given in Fig. 1(right). In both cases, the number of target pixels to be detected, t, was set to 18 after calculating the intrinsic dimensionality of the data [3]. As shown by Table 3, the Hetero-ATDCA was always able to extract pixel vectors which were identical (or very close, spectrally) to the known targets. In Table 3, the closer the SAD score is to zero, the more similar the two pixel vectors are. Quite opposite, the Hetero-UFCLS could not accurately detect several target pixels, including the one labeled as ‘F’ which corresponds to the thermal hot spot with 700F temperature. For illustrative purposes, Table 3 also gives processing times in seconds (between parenthesis) for the sequential versions, respective to ATDCA and UFCS (implemented using the GNU-C/C++ compiler in its 4.0 version) executed on a single processor of the Thunderhead Beowulf cluster. Here, the tested versions were really sequential (not parallel versions running on one processor).

On the other hand, Table 4 shows the classification accuracies obtained by Hetero-PCT and Hetero-MORPH for all the classes available in a reference dust/debris map available from USGS (additional information on the classes displayed in Table 4 is available online from http://pubs.usgs.gov/of/2001/ofr-01-0429/dustplume.html). The number of classes to be detected by Hetero-PCT and Hetero-MORPH, c, was empirically set to seven in both cases after observing the number of classes in the online USGS dust/debris map. For the Hetero-MORPH algorithm, the number of iterations I_{max} was set to five after previous studies [14]. As shown by Table 4, Hetero-MORPH substantially improved the classification accuracies obtained by Hetero-PCT. This is because morphological processing uses spatial and spectral information as opposed to PCT-based processing, which only uses spectral information. Table 4 also includes (between parentheses) the processing times obtained after executing the sequential versions in a single Thunderhead processor. As shown by Table 4, the computational cost was higher when Hetero-MORPH was used, but this algorithm consistently produced much better classification results for all the considered classes.

Table 3. Spectral similarity between target pixels detected by heterogeneous algorithms and the known ground targets. Single-processor times given in the parentheses.

Hot spot	Hetero-ATDCA (1263)	Hetero-UFCLS (916)
‘A’	0.002	0.123
‘B’	0.001	0.005
‘C’	0.005	0.012
‘D’	0.003	0.002
‘E’	0.008	0.026
‘F’	0.001	0.169
‘G’	0.001	0.001

Table 4. Classification accuracies (percentage) obtained by heterogeneous algorithms for the USGS dust/debris classes. Single-processor times given in the parentheses.

Dust/debris	Hetero-PCT (1884)	Hetero-MORPH (2334)
Concrete (WTC01-37B)	93.56	0.123
Concrete (WTC01-37Am)	90.23	0.005
Cement (WTC01-37A)	81.64	0.012
Dust (WTC01-15)	79.23	0.002
Dust (WTC01-28)	76.67	0.026
Dust (WTC01-36)	85.02	0.169
Gypsum wall board	82.99	0.001
Overall	80.45	0.001

In order to improve computational performance, we tested the parallel heterogeneous versions above on the five considered parallel platforms. The algorithms were implemented using C++ with calls to message passing interface (MPI). We made use of MPI *derived datatypes* to directly scatter hyperspectral data structures, which may be stored non-contiguously in memory, in a single communication step. To investigate their parallel properties, the algorithms were first tested by timing the parallel versions on the four equivalent networks of computers. Table 5 shows the measured execution times for the proposed parallel heterogeneous algorithms and their respective homogeneous versions.

As expected, the execution times reported on Table 5 show that the heterogeneous algorithms were able to adapt much better to fully (or partially) heterogeneous environments than the homogeneous versions, which only performed satisfactorily on the fully homogeneous network. Table 5 also reveals that the performance of the heterogeneous algorithms on the fully heterogeneous platform was almost the same as that evidenced by the equivalent homogeneous algorithms on the fully homogeneous network. This indicated that the proposed heterogeneous algorithms were always close to the optimal heterogeneous modification of the basic homogeneous ones. One can also see from Table 5 that processor heterogeneity has a more significant impact on algorithm performance than network heterogeneity, which is not surprising the type of algorithms being executed, in which the amount of communication is much less than the amount of computation required. Finally, Table 5 also shows that the homogeneous versions only slightly outperformed the heterogeneous algorithms in the fully homogeneous network. This demonstrates the flexibility of heterogeneous algorithms, which were able to adapt efficiently to the four considered environments.

In order to further explore the parallel properties of the considered algorithms in more detail, an in-depth analysis of computation and communication times achieved by the different methods is also highly desirable. Table 6 shows the total time spent by the tested algorithms in communications and computations in the four networks, where two types of computation times were analyzed, namely, sequential (those performed by the root node with no other parallel tasks active in the system, labeled as SEQ in the table) and parallel (the rest of computations, i.e., those performed by the root node and/or the workers in parallel, labeled as PAR in the table). The latter includes the times in which the workers remain idle. It can be seen from Table 6 that, among all considered heterogeneous parallel algorithms, SEQ scores were particularly significant for the Hetero-PCT algorithm. This is mainly due to the fact that this algorithm involves several steps based on sequential computations. SEQ scores were also relevant for the Hetero-ATDCA. It should be noted that Hetero-ATDCA involves several gather/scatter operations followed by compute-intensive orthogonal space projections at the master, which need to be completed in sequential fashion before a new parallel operation can be accomplished by the workers. Similar issues are also present in Hetero-UFCLS (see Step 3 of this algorithm). Quite opposite, although the Hetero-MORPH is the only technique that introduces redundant information (expected to slow down computations a priori), Table 6 reveals that the SEQ scores measured for this algorithm were much lower than those reported by the other tested algorithms, in particular, in the fully heterogeneous network. This comes at no surprise, since the Hetero-MORPH algorithm is a windowing-type approach with very few data dependencies. Finally, the cost of parallel (PAR) computations clearly dominated that of communications (COM) in all the considered parallel algorithms. For instance, the PAR scores achieved by the homogeneous algorithms executed on the (fully or partially) heterogeneous network were extremely high, due to a less efficient workload distribution among the heterogeneous workers.

To analyze the issue of load balance in more detail, Table 7 shows the *imbalance* scores achieved by the parallel algorithms on the four networks. The imbalance is defined as $D = R_{max}/R_{min}$, where R_{max} and R_{min} are the maxima and minima processor run times, respectively. Therefore, perfect balance is achieved when $D = 1$. In the table, we display the imbalance considering all processors, D_{all}, and also considering all processors but the root, D_{minus}. As we can see from Table 7, only the Hetero-MORPH algorithm was able to provide values of D_{All} close to 1 in all considered networks, with Hetero-ATDCA being able to produce results which are also relatively close to 1. Further, the Hetero-MORPH provided almost the same results for both D_{All} and D_{Minus} while, for the other tested methods, load balance was generally better when the root processor was not included. Interestingly enough, despite the fact that conventional hyperspectral imaging algorithms do not take into account the spatial information explicitly into the computa-

Table 5. Execution times (seconds) of heterogeneous algorithms and their homogeneous versions.

Algorithm	Fully heterogeneous	Fully homogeneous	Partially heterogeneous	Partially homogeneous
Hetero-ATDCA	84	89	87	88
Homo-ATDCA	667	81	638	374
Hetero-UFCLS	51	56	55	56
Homo-UFCLS	506	50	497	253
Hetero-PCT	132	136	133	135
Homo-PCT	562	129	547	330
Hetero-MORPH	171	177	172	174
Homo-MORPH	2216	168	2203	925

Table 6. Communication (COM), sequential computation (SEQ) and parallel computation (PAR) times (in seconds) measured for the heterogeneous algorithms and their homogeneous versions.

	Fully heterogeneous			Fully homogeneous			Partially heterogeneous			Partially homogeneous		
	COM	SEQ	PAR	COM	SEQ	PAR	COM	SEQ	PAR	COM	SEQ	PAR
Hetero-ATDCA	7	19	58	11	16	62	8	18	61	8	20	60
Homo-ATDCA	14	19	634	6	16	59	9	18	611	12	20	342
Hetero-UFCLS	4	17	30	7	14	35	6	17	32	8	16	32
Homo-UFCLS	9	17	480	3	14	33	5	17	475	13	16	224
Hetero-PCT	6	27	99	9	28	99	8	26	99	8	27	100
Homo-PCT	12	27	523	5	28	96	7	26	514	9	27	294
Hetero-MORPH	9	6	156	13	8	156	10	7	155	10	8	156
Homo-MORPH	17	6	2201	7	8	153	9	7	2187	11	8	906

tions (which has traditionally been perceived as an advantage for the development of parallel implementations) and taking into account that Hetero-MORPH introduces redundant information expected to slow down the computation, results in Table 7 indicate that this heterogeneous algorithm is indeed effective in terms of workload distribution among heterogeneous processors.

Taking into account the results presented above, and with the ultimate goal of exploring the scalability of heterogeneous algorithms, we have also compared their performance on NASA's Thunderhead Beowulf cluster. Fig. 2 plots the speedups achieved by multi-processor runs of the heterogeneous parallel algorithms over their corresponding single-processor runs on Thunderhead. It can be seen that Hetero-MORPH scaled much better than Hetero-PCT. This has to do with the high number of sequential computations involved in Hetero-PCT (see Table 6). On the other hand, the Hetero-ATDCA scaled slightly better than Hetero-UFCLS.

For the sake of quantitative comparison, Table 8 also reports the execution times achieved by the heterogeneous parallel algorithms on Thunderhead. For instance, using 256 processors, the Hetero-MORPH algorithm provided a high-quality (more than 93% accuracy) debris/dust map of the full AVIRIS scene in only 11 seconds (with only 2 seconds spent in SEQ computations), while the Hetero-ATDCA algorithm was able to accurately detect the spatial location of the most significant thermal hot spots in the WTC area in only 7 seconds (with 1 second in SEQ computations). The above results represent significant improvements over the single-processor versions of the same algorithms, which can take up to more than one hour of computation for the considered problem size, as indicated by tables 3 and 4. The processing times above were also deemed suitable for rapidly providing emergency response teams with information on the presence of fires and the distribution of debris and other materials in the dusts deposited around the WTC area in this particular case study.

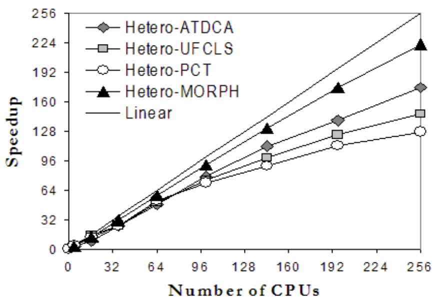

Figure 2. Scalability of heterogeneous parallel algorithms on NASA's Thunderhead.

4. Conclusions

In this paper, we have discussed the role of heterogeneous parallel computing in remote sensing applications. Specifically, we have presented several highly innovative parallel techniques for target detection and classification of hyperspectral imagery, and implemented them on various homogeneous, heterogeneous and massively parallel platforms. In order to address several important issues involved

Table 7. Load balancing rates for the heterogeneous algorithms and their homogeneous versions.

	Fully heterogeneous		Fully homogeneous		Partially heterogeneous		Partially homogeneous	
	D_{all}	D_{minus}	D_{all}	D_{minus}	D_{all}	D_{minus}	D_{all}	D_{minus}
Hetero-ATDCA	1.19	1.05	1.16	1.03	1.24	1.06	1.22	1.03
Homo-ATDCA	1.62	1.23	1.20	1.06	1.67	1.26	1.41	1.05
Hetero-UFCLS	1.49	1.06	1.51	1.05	1.69	1.06	1.54	1.08
Homo-UFCLS	1.68	1.25	1.54	1.11	1.75	1.34	1.77	1.09
Hetero-PCT	1.69	1.06	1.58	1.03	1.72	1.05	1.68	1.07
Homo-PCT	1.81	1.28	1.56	1.05	1.82	1.39	1.83	1.08
Hetero-MORPH	1.05	1.01	1.03	1.02	1.06	1.02	1.06	1.04
Homo-MORPH	1.59	1.21	1.05	1.01	1.62	1.24	1.28	1.13

Table 8. Execution times (seconds) for the heterogeneous versions of the parallel algorithms using different numbers of processors on Thunderhead.

CPUs	ATDCA	UFCLS	PCT	MORPH
1	1263	916	1884	2334
4	493	286	460	741
16	141	63	154	191
36	49	36	73	74
64	26	18	36	40
100	16	12	26	26
144	11	9	21	18
196	9	7	17	13
256	7	6	15	11

in the exploitation of such heterogeneous algorithms in real applications, we have also provided a detailed discussion on the effects that platform heterogeneity has on degrading parallel performance of hyperspectral imaging algorithms. The evaluation strategy conducted in this work was based on experimentally assessing heterogeneous algorithms by comparing their efficiency on (fully or partially) heterogeneous networks of workstations with the efficiency achieved by their homogeneous versions on equally powerful homogeneous networks. Our study reveals that the combination of the (readily available) computational power offered by heterogeneous platforms with the recent advances in parallel algorithm design is likely to introduce new perspectives in the systems currently used by NASA and other agencies for exploiting sheer volumes of Earth and planetary remotely sensed data, now being collected on a daily basis.

References

[1] O. Beaumont, V. Boudet, F. Rastello, and Y. Robert. Matrix multiplication on heterogeneous platforms. *IEEE Trans. Parallel and Distributed Systems*, 12:1033–1055, 2001.

[2] H. Casanova, M. Thomason, and J. Dongarra. Stochastic performance prediction for iterative algorithms in distributed environments. *Journal of Parallel and Distributed Computing*, 58:68–91, 1999.

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

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

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

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

[7] K. Itoh. Massively parallel fourier-transform spectral imaging and hyperspectral image processing. *Optics and Laser Technology*, 25:202, 1993.

[8] S. Kalluri, Z. Zhang, J. JaJa, S. Liang, and J. Townshend. Characterizing land surface anisotropy from AVHRR data at a global scale using high performance computing. *International Journal of Remote Sensing*, 22:2171–2191, 2001.

[9] D. A. Landgrebe. *Signal theory methods in multispectral remote sensing*. Wiley: Hoboken, NJ, 2003.

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

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

[12] A. Legrand, H. Renard, Y. Robert, and F. Vivien. Mapping and load-balancing iterative computations on heterogeneous clusters with shared links. *IEEE Trans. Parallel and Distributed Systems*, 15:549–558, 2004.

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

[14] A. Plaza, D. Valencia, J. Plaza, and P. Martinez. Commodity cluster-based parallel processing of hyperspectral imagery. *Journal of Parallel and Distributed Computing*, 66(3):345–358, 2006.

[15] F. J. Seinstra, D. Koelma, and J. M. Geusebroek. A software architecture for user transparent parallel image processing. *Parallel Computing*, 28:967–993, 2002.

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

[17] S. Tehranian, Y. Zhao, T. Harvey, A. Swaroop, and K. Mckenzie. A robust framework for real-time distributed processing of satellite data. *Journal of Parallel and Distributed Computing*, 66:403–418, 2006.

[18] T. Yang and C. Fu. Heuristic algorithms for scheduling iterative task computations on distributed memory machines. *IEEE Trans. Parallel and Distributed Systems*, 8:608–622, 1997.