Parallel morphological neural networks for hyperspectral image classification on fully heterogeneous and homogeneous networks of workstations

Javier Plaza, Antonio Plaza, Rosa Pérez, and Pablo Martínez

Department of Technology of Computers and Communications,
University of Extremadura, Avda. de la Universidad s/n, E-10071 Cáceres, Spain
E-mail: {jplaza, aplaza, rosapere, pablomar}@unex.es

Hyperspectral imaging is a new technique in remote sensing which allows an airborne/satellite sensor to collect hundreds of images (at different wavelength channels) for the same area on the surface of the Earth. Most hyperspectral imaging applications require that a response is provided quickly enough for practical use. In this paper, we develop a new parallel morphological/neural algorithm for thematic classification of hyperspectral images which has been specifically designed to be efficiently executed on fully heterogeneous computing platforms. The algorithm integrates spatial and spectral information by making use of a special kind of parallel morphological perceptrons specifically developed for this purpose. The performance of the different implementation strategies adopted for the two main modules (morphological and neural) is tested in this work by using a collection of hyperspectral image data sets obtained from real, application-oriented remote sensing missions. Performance tests are conducted in various homogeneous/heterogeneous computing platforms, including two networks of workstations at University of Maryland and a Beowulf cluster at NASA's Goddard Space Flight Center.

1 Introduction

Hyperspectral imaging is an emerging research area concerned with the measurement, analysis, and interpretation of spectra acquired from a given scene (or specific object) at a short, medium or long distance by an airborne or satellite Earth observation sensor[1]. The concept of hyperspectral imaging was first introduced when NASA's Jet Propulsion Laboratory (JPL) developed the Airborne Visible-Infrared Imaging Spectrometer (AVIRIS)[2]. This instrument covers the wavelength region from 0.4 to 2.5 μm using 224 narrow spectral channels, at a nominal spectral resolution of 10 nm. As a result, a hyperspectral *data 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 underlying objects within the pixel. The resulting data volume typically exceeds several gigabytes per flightline.

Most currently available techniques for hyperspectral image processing treat the data cubes not as images, but as unordered listings of spectral measurements with no spatial arrangement. In thematic classification applications, however, the exploitation of both spatial and spectral information can be greatly beneficial and represents a highly innovative contribution. While such integrated spatial/spectral developments hold great promise for Earth science image analysis, they also introduce new processing challenges[3], in particular, in the context of time-critical applications such as environmental modeling and assessment, target recognition for military and defense/security deployment, urban planning studies, wild land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination.

To address the computational needs introduced by advanced processing techniques in such relevant applications, several efforts have been recently directed towards the incorporation of high performance computing in remote sensing missions, especially with the advent of relatively cheap Beowulf clusters[4]. Although most parallel techniques and systems for image information processing employed by institutions such as NASA or the European Space Agency during the last decade have been chiefly 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 and distributed parallel platforms.

In this paper, we develop a new morphological/neural parallel algorithm for spatial/spectral thematic classification of hyperspectral images which has been specifically developed to be executed on fully heterogeneous networks of workstations. The algorithm integrates spatial and spectral information by making use of a special kind of morphological perceptrons. The remainder of the paper is structured as follows. Sec. 2 describes related work in morphological/neural processing. Sec. 3 develops the proposed parallel algorithm. Sec. 4 provides experimental comparisons with other parallel algorithms for thematic classification of hyperspectral images. Experimental results are given from the viewpoint of classification accuracy and parallel performance on a variety of homogeneous and heterogeneous platforms. Finally, Sec. 5 concludes with some remarks.

2 Related work

Parallelization of low-level image processing algorithms, including morphological operations[4], a special kind of nonlinear image filters which provide an excellent framework for the integration of spatial and spectral information, has been the subject of several research studies[3]. However, the development of parallel approaches dealing with multi-channel imagery in the context of remote sensing applications is limited to only a few studies in the area of cluster-based and distributed computing[5]. In the latter case, the general approach for the design of parallel implementations is based on the development of effective heuristics for scheduling iterative task computations on the distributed platform. Quite opposite, the development of parallel neural network-oriented methods in general remote sensing applications has been quite popular. Previous research in this area has addressed the impact of neural network partitioning and mapping onto specific parallel architectures such as clusters of computers[6,7]. However, only preliminary results have been reported in the area of parallel implementations of neural network architectures for hyperspectral image processing[8]. In the following section, we describe a novel parallel morphological/neural supervised classification technique, which performs supervised classification of hyperspectral images taking into account both the spectral and the spatial information contained in the input hyperspectral data.

3 Parallel algorithm

This section describes a new parallel algorithm for analysis of remotely sensed hyperspectral images. First, we formulate a general optimization problem in the context of fully heterogeneous networks, composed of different-speed processors that communicate through

links at different capacities[9]. This type of platform can be modeled as a complete graph $G = (P, E)$ where each heterogeneous processor p_i is weighted by its relative speed w_i. Each edge in the graph models a communication link (e.g., from p_i to p_j) weighted by its relative capacity $c^{(i,j)}$. With the above definitions in mind, processor p_i will accomplish a share of $\alpha_i \times W$ of the total workload W, with $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$. An abstract view of our problem can be stated in the form of a client-server architecture, in which the server is responsible for the distribution of work among the P nodes, and the clients operate with the spatial and spectral information contained in a local partition.

3.1 Parallel morphological feature extraction

The parallel algorithm first performs parallel morphological feature extraction taking into account the spatial and spectral information in simultaneous fashion. This algorithm consists of moving a kernel (called *structuring element* in mathematical morphology terminology) around each pixel vector, defining a spatial context around each spectral-based elements and constructing a so-called morphological profile which can then be used for classification purposes[4].

Two types of partitioning can be adopted for the parallelization of spatial/spectral algorithms such as the one addressed above[4]. 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. In this work, we adopt a spatial-domain partitioning approach, in which the same pixel vector is never partitioned among several processors to avoid additional inter-processor communications. In this case, the window-based calculations made for each hyperspectral pixel may need to originate from several processing elements when such elements are located at the border of the local data partitions. However, if redundant information such as an overlap border is added to each of the adjacent partitions to avoid accesses outside the image domain, then boundary data to be communicated between neighboring processors can be greatly minimized. Our implementation makes use of a constant structuring element with size of 3×3 pixels, which is repeatedly iterated to increase the spatial context[4], and therefore the total amount of redundant information is minimized. Three different approaches have been tested in the implementation of the overlapping scatter operation:

- The first one (called MP-1) implements a standard non-overlapping scatter operation followed by overlap communication for every hyperspectral pixel vector, thus communicating small sets of pixels very often.
- The second one (called MP-2) implements a standard non-overlapping scatter operation followed by a special overlap communication which sends all border data beforehand, but only once.
- The third one (called MP-3) implements a special overlapping scatter operation that also sends out the overlap border data as part of the scatter operation itself.

A pseudo-code of the proposed parallel algorithm, specifically tuned for heterogeneous platforms, is given below. The inputs to the algorithm are an N-dimensional cube f, where N is the number of spectral bands, and a structuring element with 3×3 pixel size. The output is a set of morphological profiles for each pixel $f(x, y)$.

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

2. Using B and the information obtained in the previous step, determine the total volume of information, R, that needs to be replicated from the original data volume, V in accordance with the selected overlapping scatter communication strategy (MP-1, MP-2 or MP-3), and let $W = V + R$.

3. Set $\alpha_i = (1/w_i) \cdot (1/\sum_{j=1}^{p}(1/w_i))$ for $i \in \{1 \cdots P\}$.

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

5. Calculate (in parallel) the morphological profile, denoted by MP(x, y), for all the pixels in the local data partitions at each heterogeneous processor.

6. The master collects all the individual results produced by the workers and merges them together to produce the final output.

3.2 Parallel neural algorithm

The second stage of the parallel algorithm is based on a multi-layer perceptron (MLP) neural network with back-propagation learning. The network is trained with selected features from the previous morphological feature extraction stage. This supervised approach has been shown in previous work to be very robust for classification of hyperspectral imagery[8]. Several mapping schemes for implementing the parallel MLP classifier on heterogeneous networks have been tested:

- The first one (called *exemplar partitioning*) partitions the training pattern data set so that each processor determines the weight changes for a disjoint subset of the training population and then changes are combined and applied to the neural network at the end of each epoch.

- The second strategy allows us to develop several combinations of neuronal-level as well as synaptic-level parallelism which have been shown in previous work to significantly reduce the amount of inter-processor communications at each iteration[8]:

  - In the case of *neuronal-level parallelism* (also called *vertical partitioning*), all the incoming weights to the neurons local to each processor are computed by a single processor.
  - In *synaptic-level parallelism*, each workstation computes only the outgoing weight connections of the neurons local to the processor.
  - Finally, in the *mixed neuronal/synaptic-level parallelism*, the hidden layer of the neural network is partitioned using neuronal-level parallelism, while parallelization of the weight connections adopts synaptic-level parallelism[7].

The two parallel classifiers tested in this work are the *exemplar partitioning*-based and the one with *mixed neuronal/synaptic-level parallelism*, in which the hidden layer is partitioned using neuronal level parallelism and weight connections are partitioned on the basis

of synaptic level parallelism. In the former, the same parallel neural architecture is simply applied in parallel to different subsets of the training set, where the size in pixels of each training set depends on the relative speed of the allocated processor. On the other hand, in the latter approach the input and output layers of the same neural network are common to all processors, while the hidden layer is partitioned so that each heterogeneous processor receives a number of hidden neurons which depends on its relative speed. Since the fully connected MLP network is partitioned into P partitions and then mapped onto P heterogeneous processors using the above framework, each processor is required to communicate with every other processor to simulate the complete network.

A pseudo-code of the parallel back-propagation learning algorithm is given below. The inputs to the algorithm are an N-dimensional data cube f, a number of classes to be detected C, and a set of pixel vectors $f_j(x, y)$ used as training patterns. The output is a set of per-pixel classification labels.

1. Apply steps 1-3 of the parallel morphological feature extraction algorithm to obtain a set of values $\{\alpha_i\}_{i=1}^{P}$ and use them to obtain a set of *P* heterogeneous partitions of the hidden layer and map the resulting partitions among the *P* heterogeneous processors.

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

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

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

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

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

Table 1. Specifications of processors in a heterogeneous network at University of Maryland.

Processor	Architecture	Speed (MHz)	Main memory (MB)	Cache (KB)
p_1	Free BSD – i386 Intel Pentium 4	2867	2048	1024
p_2, p_5, p_8	Linux – Intel Xeon	1977	1024	512
p_3	Linux – AMD Athlon	2457	7748	512
p_4, p_6, p_7, p_9	Linux – Intel Xeon	2783	1024	1024
p_{10}	SunOS – SUNW UltraSparc-5	440	512	2048
$p_{11} - p_{16}$	Linux – AMD Athlon	2457	2048	1024

4 Experimental results

4.1 Parallel computing platforms

The parallel platforms considered for evaluation purposes comprise two networks of workstations at University of Maryland and a Beowulf cluster at NASA's Goddard Space Flight Center, also in Maryland. Their detailed descriptions follow:

- *Heterogeneous network*. Consists of 16 heterogeneous workstations and four communication segments. Table 1 shows the properties of the 16 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. These segments are interconnected by three slower communication links with capacities $c^{(1,2)} = 29.05$, $c^{(2,3)} = 48.31$, $c^{(3,4)} = 58.14$ in milliseconds, respectively. Although this is a simple architecture, it is also a quite typical and realistic one as well.

- *Homogeneous network*. Consists of 16 identical Linux workstations $\{p_i\}_{i=1}^{16}$ with processor cycle-time of $w = 0.0131$ seconds per megaflop, interconnected via a communication network where the capacity of all links is $c = 26.64$ milliseconds.

- *Beowulf cluster*. Consists of 268 dual Xeons resulting in a total peak performance of 2.5728 Tflops (http://thunderhead.gsfc.nasa.gov). Each of the nodes has 1 GB of main memory and 80 GB of local disk space.

4.2 Performance results

The hyperspectral data set used in experiments was collected by the ROSIS airbone imaging spectrometer of the German Aerospace Agency (DLR). It was collected by a flight at 1500 meters altitude over the city of Pavia, Italy. The resulting scene has spatial resolution of 1.3 meters per pixel and 102 spectral bands. Fig. 1(a) shows the band at 639 nm, which reveals a dense residential area on one side of the river, as well as open areas and meadows on the other side. Fig. 1(b) shows ground-truth is available for several areas of the scene. As reported in previous work, the proposed morphological/neural classifier was able to provide classification results above 90% using only 578 out of 14655 ground-truth samples for the training stage, thus outperforming other methodologies able to use only a limited number of training samples to accurately classify the input hyperspectral image[3].

The parallel performance of the proposed algorithm was first evaluated by timing the different implementations tested for its two main modules (morphological and neural) using the two networks of workstations at University of Maryland. The execution times

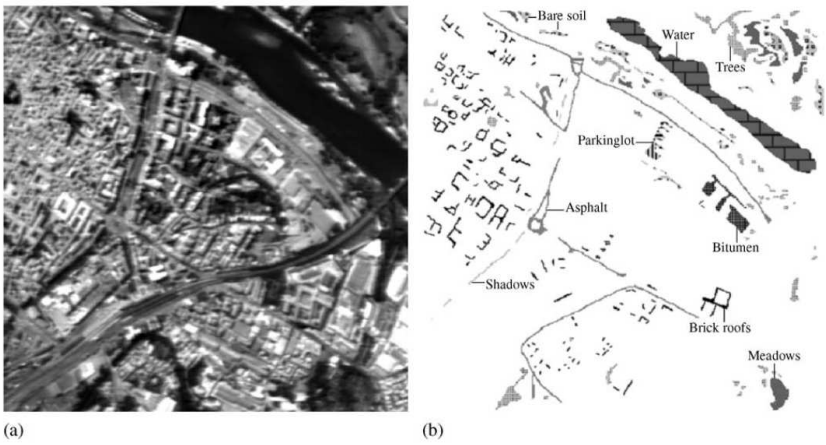

Figure 1. (a) Hyperspectral scene of Pavia city, Italy, and (b) Land-cover ground classes.

Table 2. Execution times and load balancing rates (in the parentheses) for the different alternatives tested in the implementation of the morphological and neural stages of the proposed parallel classifier.

	Morphological stage			**Neural stage**	
Parallel platform	MP-1	MP-2	MP-3	Exemplar	Hybrid
Heterogeneous network	267 (1.13)	211 (1.02)	214 (1.03)	156 (1.04)	125 (1.02)
Homogeneous network	279 (1.15)	216 (1.03)	221 (1.04)	178 (1.03)	141 (1.01)

reported on Table 2 for the considered heterogeneous algorithms indicate that the heterogeneous implementations were able to adapt to both heterogeneous and homogeneous computing environments. In order to measure load balance, Table 2 also shows (in the parentheses) the imbalance scores achieved by the different stages of the parallel algorithm on the two considered networks of workstations. The imbalance is simply defined as $D = R_{max}/R_{min}$, where R_{max} and R_{min} are the maxima and minima processor run times across all processors, respectively. Therefore, perfect balance is achieved when $D = 1$. As we can see from Table 2, the proposed implementations were effective in terms of load balance in all cases except for MP-1 which communicates pixels too often.

With the ultimate goal of exploring issues of scalability, Table 3 plots the processing times (in seconds) and speedups achieved by multi-processor runs with regards to single-processor runs of each considered algorithm on Thunderhead. The table reveals that MP-2 and hybrid neural parallelism respectively provided the best results for each stage. The exemplar neural parallelism involved convergence problems resulting from the execution of several neural classifiers in parallel. This introduced unstability in the speedups and superlinear effects which were not present in the parallel hybrid approach (only one neural classifier). Using 256 Thunderhead processors, the combined classifier (based on MP-2 feature extraction and hybrid neural parallelism) was able to provide a highly accurate classification of the considered hyperspectral scene in only 17 seconds. This represents a significant improvement over commonly used processing strategies for this kind of high-dimensional data sets, which can take up several minutes for the considered problem size.

Table 3. Processing times in seconds and speedups (in the parentheses) achieved by multi-processor runs of the considered parallel algorithms on the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center.

	4	**16**	**36**	**64**	**100**	**144**	**196**	**256**
MP-1	1177 (1.8)	339 (6.5)	146 (15.0)	81 (27.2)	53 (41.5)	42 (52.4)	37 (59.5)	36 (61.2)
MP-2	797 (2.5)	203 (10.0)	79 (25.8)	39 (52.3)	23 (88.73)	17 (120.0)	13 (157.0)	10 (204.1)
MP-3	826 (2.4)	215 (9.5)	88 (23.3)	45 (45.7)	27 (76.2)	20 ()	16 (102.9)	12 (171.5)
	2	**4**	**8**	**16**	**32**	**64**	**128**	**256**
Exmp.	1041 (1.9)	414 (4.8)	248 (8.1)	174 (11.5)	142 (14.1)	99 (20.2)	120 (16.7)	120 (16.7)
Hyb.	973 (1.6)	458 (3.5)	222 (7.2)	114 (14.0)	55 (29.2)	27 (59.5)	15 (107.1)	7 (229.5)

5 Conclusions

In this paper, we have developed a heterogeneous parallel method for classification of remotely sensed hyperspectral images using a combined morphological/neural methodology. Different approaches have been evaluated for the implementation of the morphological and neural stages, and performance results have been reported in the context of a complex urban classification scenario, using both homogeneous and heterogeneous computing platforms for validation. Experimental results anticipate that the combination of heterogeneous parallel architectures with parallel information extraction algorithm design may introduce substantial changes in the systems currently used in remote sensing missions.

References

1. A. F. H. Goetz, G. Vane, J. E. Solomon and B. N. Rock, *Imaging spectrometry for Earth remote sensing*, Science **228**, 1147–1153 (1985).
2. R. O. Green, *Imaging spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)*, Remote Sensing of Environment **65**, 227–248 (1998).
3. A. Plaza and C.-I Chang, *High performance computing in remote sensing*, Chapman & Hall/CRC Press: Boca Raton, FL (2007).
4. A. Plaza, J. Plaza and D. Valencia, *Impact of platform heterogeneity on the design of parallel algorithms for morphological processing of high-dimensional image data*, Journal of Supercomputing **40**, 81–107 (2007).
5. 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).
6. V. Kumar, S. Shekhar and M. B. Amin, *A scalable parallel formulation of the back-propagation algorithm for hypercubes and related architectures*, IEEE Transactions on Parallel and Distributed Systems **5**, 1073–1090 (1994).
7. S. Suresh, S. N. Omkar, and V. Mani, *Parallel implementation of back-propagation algorithm in networks of workstations*, IEEE Transactions on Parallel and Distributed Systems **16**, 24–34 (2005).
8. 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*, Proc. IEEE International Conference on Cluster Computing, 475–482 (2006).
9. A. Lastovetsky, *Parallel computing on heterogeneous networks*, Wiley-Interscience: Hoboken, NJ (2003).