On the use of parallel computing to process multichannel imagery via extended morphological operations

Antonio Plaza, Pablo Martínez, David Valencia, Isabel García, Javier Plaza
Dept. de Arquitectura y Tecnología de Computadores
Escuela Politécnica de Cáceres, Universidad de Extremadura
Avda. de la Universidad s/n
10071 Cáceres
aplaza@unex.es

Abstract

Multichannel images are characteristic of certain applications, such as medical imaging or remotely sensed data analysis. In such images, each pixel is given by a vector of values. Due to the large data volumes often associated with multichannel imagery, there is a need for parallel algorithms able to process those data quickly enough for practical use. This paper describes a parallel implementation of a multichannel image processing algorithm that naturally combines spatial and spectral/temporal information. The algorithm combines a vector-preserving approach to extend morphological operations to multichannel images with the watershed transformation used in morphological processing, It is evaluated using remotely sensed hyperspectral data collected by the NASA's Jet Propulsion Laboratory Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS).

1. Introduction

Multichannel images are defined over the common spatial definition domain. It follows that, in multichannel image data, a vector of values rather than a single value is associated with each spatial location.

Many types of multichannel images exist depending on the type of information collected for each image pixel. For instance, color images are multichannel images with three channels, one for each primary color in the RGB space. Images optically acquired in more than one spectral or wavelength interval are called multispectral [1]. These images are characteristic in satellite imaging and aerial reconnaissance applications. The number of spectral channels can be extremely high, as in the case of hyperspectral images produced by imaging spectrometers. Finally, all image types above can be extended to the class of multitemporal images or image sequences, which consist of series of images defined over the same definition domain, but collected at more than a single time. Examples include magnetic resonance (MR) images in medical applications and video sequences.

One of the most successful techniques for image analysis in the recent literature has been mathematical morphology [2], which relies on the definition of a structuring element (SE) which is translated over the image, and acts as a probe for extracting or suppressing specific structures of the image objects. Based on this idea, two fundamental operators are defined in mathematical morphology: erosion and dilation. The application of the erosion operator to an image yields an output image, which shows where the SE *fits* the objects in the image. On the other hand, the application of the dilation operator to an image produces an output image, which shows where the SE *hits* the objects in the image. All other morphological operations are expressed in terms of erosion and dilation.

For instance, the most important morphological approach for segmenting an image is the *watershed* transformation [3], which consists of a combination of seeded region growing and edge detection. It relies on a marker-controlled approach that considers the image data as imaginary topographic relief; the brighter the intensity, the higher the corresponding elevation. Let us assume that a drop of water falls on such a topographic surface. The drop will flow down along the steepest slope path until it reaches a minimum. The set of points of the surface whose steepest slope path reach a given minimum constitutes the catchment basin associated with that minimum, while the watersheds are the zones dividing adjacent catchment basins.

Despite its encouraging results in, among others, industrial applications, object identification, security control, and image coding, mathematical morphology-based techniques such as the watershed transform have not been fully exploited in multichannel image analysis, mainly due to the fact that the price paid for the wealth of spatial and spectral/temporal information in certain applications is an enormous amount of data to be processed [4].

Parallel processing and the new processing power offered by commodity cluster-based systems [5] can help to tackle large multidimensional image data sets and to get reasonable response times in complex image analysis scenarios [6]. Thanks to the geographic local organization of the pixels of an image as a 2-D mesh, and to the regularity of most low-level computations, mesh-based parallel architectures have become quite popular for image analysis applications since they allow for efficient implementations of basic neighbor-based primitives [7]. However, the development of appropriate parallel analysis techniques for joint spatial and spectral/temporal analysis has not been fully explored yet in the literature.

This paper describes a new parallel algorithm that allows for efficient exploitation of multichannel image data via an extended watershed transform. The algorithm was specifically designed to be run on similar computing nodes employed in a stand-alone manner. A quantitative cost-performance evaluation of the algorithm is provided, using Thunderhead, a 256-processor commodity cluster at NASA's Goddard Space Flight Center.

2. Extended mathematical morphology

Our attention in this section focuses primarily on the development of a mechanism to extend basic morphological operations to multichannel imagery [8].

A simple approach consists in applying monochannel techniques to each image channel separately, an approach usually referred to as "marginal" morphology in the literature. This approach is unacceptable in many applications because, when morphological techniques are applied independently to each image channel, there is a possibility for loss or corruption of information of the image due to the probable fact that new pixel vectors (i.e., not present in the original image domain) may be created as a result of processing the channels separately. In addition, no correlation between spectral/temporal components is taken into account.

An alternative way to approach the problem of multidimensional morphology is to treat the data at each pixel as a vector [9]. Since there is no unambiguous means of defining the minimum and maximum values between two vectors of more than one dimension, it is important to define an appropriate arrangement of vectors in vector space. In this paper, we adopt a distance-based technique based on a cumulative distance between one particular pixel vector $\boldsymbol{f}(\mathrm{x},\mathrm{y})$, where (x,y) indicates the spatial coordinates, and all the pixel vectors in the spatial neighborhood given by structuring element B (B-neighborhood) as:

$$\mathrm{C}_B\left(\boldsymbol{f}(\mathrm{x},\mathrm{y})\right)=\sum_{(\mathrm{s},\mathrm{t})}\mathrm{Dist}\left(\boldsymbol{f}(\mathrm{x},\mathrm{y}),\boldsymbol{f}(\mathrm{s},\mathrm{t})\right)$$

where Dist is the spectral angle distance. As a result, $\mathrm{C}_B\left(\boldsymbol{f}(\mathrm{x},\mathrm{y})\right)$ is given by the sum of Dist scores between $\boldsymbol{f}(\mathrm{x},\mathrm{y})$ and every other pixel vector in the B-neighborhood. To be able to define the standard morphological operators in a complete lattice framework, we need to be able to define a *supremum* and an *infimum* given an arbitrary set of vectors $\mathrm{S}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,...,\boldsymbol{v}_p\}$, where p is the number of vectors in the set. This can be done by computing $\mathrm{C}_B(\mathrm{S})=\{\mathrm{C}_B(\boldsymbol{v}_1),\mathrm{C}_B(\boldsymbol{v}_2),\cdots,\mathrm{C}_B(\boldsymbol{v}_p)\}$ and selecting $\boldsymbol{v}_j$ such that $\mathrm{C}_B(\boldsymbol{v}_j)$ is the minimum of $\mathrm{C}_B(\mathrm{S})$, with $1\le j\le p$. In similar fashion, we can select $\boldsymbol{v}_k$ such that $\mathrm{C}_B(\boldsymbol{v}_k)$ is the maximum of $\mathrm{C}_B(\mathrm{S})$, with $1\le k\le p$. Based on the simple definitions above, the flat extended erosion of $\boldsymbol{f}$ by B, denoted by $\boldsymbol{f}\ominus B$, consists of selecting of the B-neighborhood pixel vector that produces the minimum C_B value. On other hand, the flat extended dilation of $\boldsymbol{f}$ by B, i.e., $\boldsymbol{f}\oplus B$, selects the B-neighborhood pixel that produces the maximum value for C_B. The multichannel morphological gradient at $\boldsymbol{f}(\mathrm{x},\mathrm{y})$ using B can be simply defined as follows [10]:

$$\mathrm{G}_B\left(\boldsymbol{f}(\mathrm{x},\mathrm{y})\right)=\mathrm{Dist}\left((\boldsymbol{f}\oplus B)(\mathrm{x},\mathrm{y}),(\boldsymbol{f}\otimes B)(\mathrm{x},\mathrm{y})\right)$$

3. Multichannel watershed algorithm

Our multichannel watershed segmentation algorithm consists of three stages. First, multidimensional morphological operations are used to collect a set of minima according to some measure of minima importance. Starting from the selected minima and using the multidimensional morphological gradient as a reference, a multichannel watershed transformation by flooding is then applied. Finally, watershed regions are iteratively merged, according to a similarity criterion, to obtain the final segmentation.

3.1. Minima selection

The key of an accurate segmentation resides in the first step, that is, the selection of "markers" or minima from which the transform is started. Following a recent work [3], we hierarchically order all minima according to their deepness, and then select only those above a threshold [11]. The deepness of a basin would be the level the water would reach, coming in through the minimum of the basin, before the water would overflow into a neighbor basin.

Deepness can be computed using morphological reconstruction applied to the multichannel gradient. Reconstruction is a special class of morphological transformation that does not introduce discontinuities. Given a "flat" SE designed by B, and the multichannel gradient $\mathrm{G}_B(\boldsymbol{f})$ of an n-dimensional image $\boldsymbol{f}$, it can be proven that the morphological reconstruction of $\mathrm{G}_B(\boldsymbol{f})$ from $\mathrm{G}_B(\boldsymbol{f})\otimes B$ will have a watershed transform in which the regions with deepness lower than a certain value v have been joined to the neighbor region with closer spectral properties. Subsequently, the parameter v can serve as a minima selection threshold.

3.2. Flooding

Let the set $\mathrm{P}=\{\boldsymbol{p}_1,\boldsymbol{p}_2,...,\boldsymbol{p}_k\}$ denote the set of k minimum pixel vectors selected after multidimensional minima selection. Similarly, let the catchment basin associated with a minimum pixel $\boldsymbol{p}_i$ be denoted by $\mathrm{CB}(\boldsymbol{p}_i)$. All the points in this catchment basin which have an altitude less than or equal to a certain deepness score. The catchment basins is progressively created by simulating the flooding process. The first pixel vectors reached by water are the points of highest deepness score. From now on, the water either expands the region of the catchment basin already reached by water, or starts to flood the catchment basin whose minima have a deepness equal to $D_B(\boldsymbol{p}_l)$, where $\boldsymbol{p}_l$ is the deepest pixel in the set of $\mathrm{P}-\{\boldsymbol{p}_j\}$. This operation is repeated until $\mathrm{P}=\varnothing$.

3.3. Region merging

To obtain the final segmentation, some of the regions $\{\mathrm{CB}(\boldsymbol{p}_i)\}_{i=1}^{k}$ resulting from the watershed can be merged to reduce the number of regions. First, all regions are ordered into a region adjacency graph (RAG). Each edge in the RAG is assigned a weight, so that the weight of an edge $\mathrm{e}(\boldsymbol{p}_i,\boldsymbol{p}_j)$ is the value of $\mathrm{Dist}(\boldsymbol{p}_i,\boldsymbol{p}_j)$. Regions $\mathrm{CB}(\boldsymbol{p}_i),\mathrm{CB}(\boldsymbol{p}_j)$ can be merged attending to spatial properties in the case of adjacent regions, and also according to pixel vector similarity criteria in the case of non-adjacent regions.

4. Parallel implementation

Parallelization of watershed algorithms that simulate flooding is not a straightforward task. The watershed process has a very volatile behavior, starting with a high degree of parallelism that very rapidly diminishes to a much lower degree of parallelism [12]. In this section, our goal is to develop an efficient and scalable parallel implementation of the algorithm proposed in the previous section. Design features such as partitioning, task replication and communication are discussed.

4.1. Partitioning

Two types of partitioning strategies can be applied to our problem. One may decide whether the computation associated with the given problem should be split into pieces (functional decomposition), or the data on which the

computation is applied (domain decomposition). Functional decomposition is inappropriate for segmentation with watersheds, where a sequence of operators are applied in a chain to the entire image. Our parallel algorithm uses domain decomposition, that is, the original multichannel image $\boldsymbol{f}$ is decomposed into subimages, where each subimage is made up of entire pixel vectors, i.e. a single pixel vector is never split amongst several processing elements (PEs). There are several reasons for the above decision. First, this is a natural approach for low-level image processing, 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. If the computations for each pixel vector need to originate from several PEs, then they would require intensive inter-processor message passing.

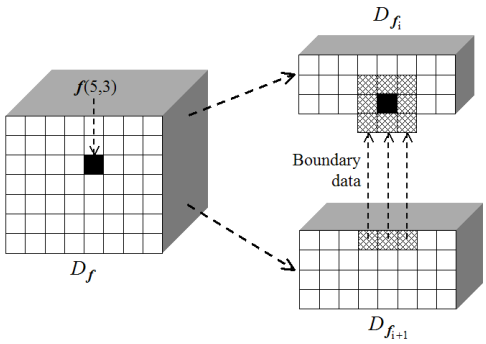

Figure 1. Morphological structuring element computation split between two processors.

4.2. Task replication

An important issue in SE or kernel-based image processing operations is that accesses to pixels outside the domain D_f of the input image are possible. For instance, when the SE is centered on a pixel located in the border of the original image, a simple border-handling strategy can be applied to indicate that only pixels inside the input image domain will be taken into account in the SE-based computation. However, when such a border effect happens in a local partition D_{f_i}, then additional inter-processor communications may be required when the structuring element computation needs to be split amongst several different processing nodes, as shown by Fig. 1. In the example, the computations for the pixel vector at spatial location $(5,3)$ need to originate from two processors, and a communication overhead involving three pixel vectors is introduced. In order to avoid such an overhead, edge/corner pixels are replicated in the neighboring processors whose subdomains are thus enlarged with a so-called extension area. It should be noted that the task-replication strategy above enhances code reusability, which is highly recommended in order to build a robust parallel algorithm.

4.3. Implementation

Our implementation of the parallel multichannel watershed algorithm uses a simple master-slave model. The master processor reads the whole multichannel image $\boldsymbol{f}$ and divides it into a set of multichannel subimages $\boldsymbol{f}_i$ which are sent to different processors. The slave processors run the segmentation algorithm on the respective subimages and also exchange data among themselves for uniform segmentation. After the segmented regions become stable, the slaves send the output to the master, which combines all of them in a proper way and provides the final segmentation. If we assume that the parallel system has p processors available, then one of the processors is reserved to act as the master, while each of the remaining $p-1$ processors create a local queue Q_i with $1 \le i \le p-1$. The minima selection algorithm is run locally at each processor to obtain a set of minima pixels surrounded by non-minima, which are then used to initialize each queue Q_i. Flooding is then performed locally in each processor as in the serial algorithm.

It should be noted that, due to the image division, flooding is confined only to the local subdomain. There may exist parts of the subimage that cannot be reached by flooding since they are contained in other subimages. Our approach to deal with this problem is to first flood locally at every deepness score in the subimage. Once the local flooding is finished, each processor exchanges segmentation labels of pixels in the boundary with appropriate neighboring

processors. Subsequently, a processor can receive segmentation labels corresponding to pixels in the extended subdomain. The processor must now "reflood" the local subdomain from those pixels, a procedure that may introduce changes in segmentation labels of the local subdomain. Communication and reflooding are again repeated until stabilization (i.e. no more changes occur). Performance data for the parallel algorithm are given in the following subsection.

5. Experimental results

This section reports on the effectiveness of the proposed parallel segmentation algorithm in a specific application, where remotely sensed hyperspectral imagery data with hundreds of spectral channels as collected by the by the NASA's Jet Propulsion Laboratory Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) are used to test the performance of the algorithm in a complex remote sensing classification scenario. The section is organized as follows. First, we provide an overview of the parallel computing architectures used for evaluation purposes. Second, a detailed computational cost-performance analysis of our parallel algorithm is given.

5.1. Parallel computing architectures

The parallel computing architecture used in experiments is the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center (NASA/GSFC) (see Fig. 2). The Thunderhead system can be seen as an evolution of the HIVE (Highly Parallel Virtual Environment) project, started in 1997 to build a commodity cluster intended to be exploited by different users in a wide range of scientific applications. The idea was to have workstations distributed among many offices and a large number of compute nodes (the compute core) concentrated in one area.

Thunderhead 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, the system cost was relatively cheap (about 220K in 2001). It has been recently reported that the cost of an SGI Origin multicomputer with exactly the same specifications as the Thunderhead Beowulf cluster is more than 20 times greater than that of the cluster. Along with the 512-processor computer core, Thunderhead has several nodes attached to the core with 2 Ghz optical fibre Myrinet. Our parallel algorithm was run from one of such nodes, called thunder1. The operating system used at the time of experiments was Linux RedHat 8.0, and MPICH was the message-passing library used.

Figure 2. Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center.

5.2. Computational cost-performance analysis

The image data set used in experiments is a 224-band AVIRIS scene taken at a low altitude with a 3.7 meter-pixel size. The data set was collected over an agricultural test site located in Salinas Valley, California, and represents a challenging segmentation problem. Fortunately, extensive ground-truth information is available for the area, allowing a quantitative assessment in terms of segmentation accuracy. Fig. 3(a) shows the entire scene and a sub-scene of the dataset (called hereinafter Salinas A), dominated by directional regions. Fig. 3(b) shows the 15 available ground-truth regions, which comprise nearly half of the entire scene. The imaged data consists of a relatively large area (512 lines by 217 samples), where the total size of the image exceeds 100 Mbytes.

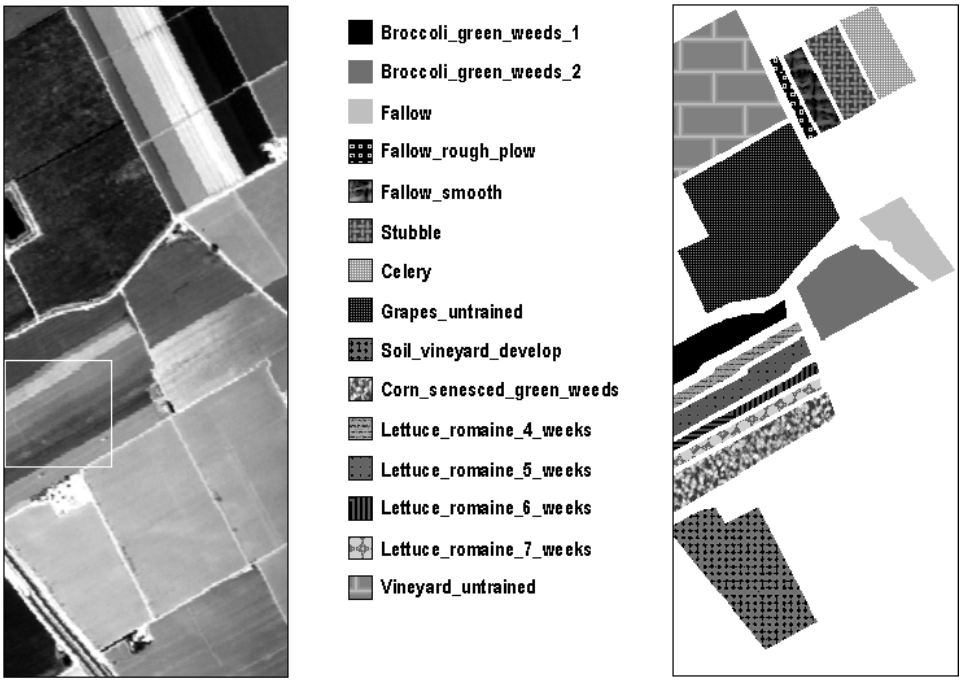

Figure 3. (a) Spectral band at 488 nm of an AVIRIS hyperspectral image comprising several agricultural fields in Salinas Valley, California, and a sub-scene (Salinas A), outlined by a white rectangle. (b) Land-cover ground classes.

The large data volume in the AVIRIS image described in Fig. 3 creates the need for parallel watershed-based analysis strategies, able to produce segmentation results quickly enough for practical use. In experiments, the best segmentation results were obtained when a disk-shaped structuring element (SE) of 15-pixel radius, denoted by $B_{15}^{(\mathrm{disk})}$, was used. This is mainly due to the relation between the SE and the spatial properties of regions of interest in the scene. Specifically, the usage of the SE above resulted in a classification accuracy above 92%.

Interestingly, classification scores were even higher for the Salinas A subscene, which contains several lettuce fields at different weeks since planting (4, 5, 6 and 7 weeks, respectively), and covering the soil in different proportions. These classes represent a very complex analysis scenario due to the presence of different proportions of mixing between soil and lettuce at the different classes. Overall, the classification results above revealed that the proposed algorithm could achieve very accurate segmentation results in a complex analysis scenario given by agricultural classes with very similar spectral features. However, the algorithm required several hours of computation in a single-processor PC with AMD Athlon 2.6 GHz processor and 512 Megabytes of RAM. This created the need for a parallel implementation.

In order to investigate the efficiency of our parallel multichannel watershed algorithm, we implemented the algorithm using the C++ programming language with calls to message passing interface (MPI). The parallel code was tested on the Thunderhead massively parallel cluster. Table 1 shows execution times in seconds of the parallel algorithm with the AVIRIS scene for $B_{15}^{(\mathrm{disk})}$ and different number of processors. As shown by Table 1, processing times were considerably reduced as the number of processors was increased, although it was not required to use a very large number of processors to obtain results in moderate processing times. For instance, 36 processors were required to produce an accurate segmentation result in less than ten minutes.

In order to further analyze the scalability of the parallel code, Fig. 4 plots the speed-up factors achieved by the parallel algorithm over a single-processor run of the algorithm as a function of the number of processors used in the parallel computation. Results in Fig. 4 reveal that the parallel algorithm obtained reasonably good scalability on Thunderhead. In particular, the achieved speed-up factors were better for large SEs, a fact that reveals that the proposed parallel implementation is more effective as the volume of computations increases. It should be noted that classification results using $B_3^{(\text{disk})}$, $B_7^{(\text{disk})}$ and $B_{11}^{(\text{disk})}$ were always below 84% accuracy.

Processors	Execution time	Speed-up
1	16234	1.00
4	5026	3.23
16	1445	11.23
36	551	29.45
64	424	39.34
100	282	57.45
144	221	73.28
196	181	89.34
256	165	98.23

Table 1. Execution time (in seconds) and speed-up achieved by the parallel watershed algorithm for different numbers of processors.

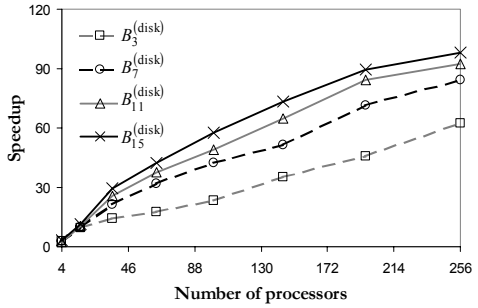

Figure 4. Speed-up factors achieved by the parallel algorithm as a function of the number of processors.

Finally, in order to investigate load balance [12], Fig. 5 shows the execution times for of the parallel algorithm for each of the processors on Thunderhead for a case study where 64 processors were used in the parallel computation, where one processor presents a slightly higher computational load as compared to the other processors. This comes at no surprise because the root processor is in charge of data partitioning, and also combines the partial results provided by every processor. It can be seen, however, that load balance is much better among the rest of the processors.

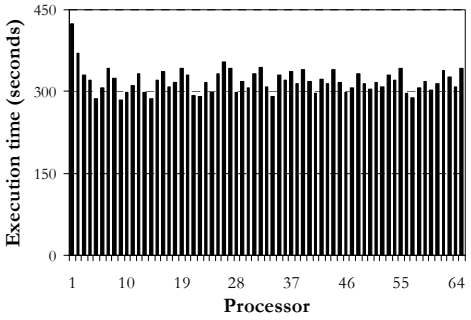

Figure 5. Execution times in seconds achieved for each of the processors for a case study with 64 processors.

6. Conclusions and future lines

The aim of this paper has been the development of a parallel implementation on massively parallel computers of an innovative technique for unsupervised classification of multichannel image data sets, with the purpose of obtaining results in valid response times and with adequate reliability. For this purpose, computing systems made up of arrays of commercial off-the-shelf computing hardware are a cost-effective way of exploiting this sort of parallelism in remote sensing applications. The proposed MPI-based parallel implementation of a morphological watershed algorithm, tuned for multiple dimensions, minimizes inter-processor communication overhead, thus enhancing processor load balance. Besides, it is independent of the number of processors, and can be ported to any type of distributed memory system. Experimental results reveal that the parallel algorithm achieved good results in terms of segmentation accuracy, speed-up, scalability and load balance in the context of a high-dimensional image analysis application, dominated by large data volumes and complex patterns of communication and calculation.

As future work, we plan to implement the parallel algorithm in other high performance parallel computing architectures, such as massively parallel computers available at the European Center for Parallelism of Barcelona (CEPBA) and CSC (Centro de Supercomputación Complutense) of Madrid, Spain, and also to fully heterogeneous networks and Grids. Our major future goal is to integrate the algorithm in applications with near real-time requirements, such as detection and/or tracking of natural disasters. Specifically, the proposed algorithm may be of great utility to detect and monitor forest fires, such as those that recently happened in the Extremadura region in SW Spain. In the near future, we will work towards the integration of the proposed parallel watershed algorithm onto an automated forest fire tracking system in conjunction with Junta de Extremadura (local government).

Acknowledgement

This research was supported by the European Commission through the project entitled "Generation of test images to validate the performance of endmember extraction and hyperspectral unmixing algorithms." (contract no. HPRI–1999–00057). Additional funding from Junta de Extremadura (local government) through 2PR03A026 project is also gratefully acknowledged. The authors would like to state their appreciation for Professors Mateo Valero and Francisco Tirado; without their collaboration, the present research effort could not have been possible. The first author would also like to acknowledge support provided by the Spanish Ministry in order to conduct research as postdoctoral associate at NASA's Goddard Space Flight Center in Maryland.

References

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

[2] Soille, P. Morphological image analysis, principles and applications, second edition. Berlin: Springer, 2003.

[3] Malpica, N., Ortuño, J. E., Santos, A. A multichannel watershed-based algorithm for supervised texture segmentation. Pattern Recognition Letters, 24, 1545-1554, 2003.

[4] Valencia, D., Plaza, A., Martinez, P. On the use of cluster computing architectures for implementation of hyperspectral analysis algorithms. Tenth IEEE Symp. on Computers & Communications, Cartagena, 2005.

[5] Dorband, J., Palencia, J., Ranawake, U. Commodity computing clusters at Goddard Space Flight Center. Journal of Space Communication, vol. 1, no. 3, 2003.

[6] Tilton, J.C. A recursive PVM implementation of an image segmentation algorithm with performance results comparing the HIVE and the Cray T3E. Proc. Symposium on Massively Parallel Computation, Maryland, 1999.

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

[8] Plaza, A., Martinez, P., Perez, R., Plaza, J. Spatial/spectral endmember extraction by multidimensional morphological operations. IEEE Transactions on Geoscience and Remote Sensing, 40(9), 2025-2041, 2002.

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

[10] Plaza, A., Martinez, P., Plaza, J., Perez, R. Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. IEEE Transactions on Geoscience and Remote Sensing, vol. 43, no. 3, pp. 466-479, 2005.

[11] Plaza, A., Martinez, P., Perez, R., Plaza. J. A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data, IEEE Transactions on Geoscience and Remote Sensing, vol. 42, no. 3, pp. 650-663, 2004.

[12] Gil, M., Gil, C., Garcia, I. The load unbalancing problem for region growing image segmentation algorithms. Journal of Parallel and Distributed Computing, vol. 63, pp. 387-395, 2003.