Cluster-Based Implementation of a Morphological Watershed Algorithm for Parallel Classification of Multichannel Images

Antonio J. Plaza
Department of Technology of Computers and Communications
Polytechnic School of Caceres, University of Extremadura
Avda. de la Universidad s/n, E-10071 Caceres, SPAIN
E-mail: aplaza@unex.es

Abstract

Due to the large data volumes often associated with multichannel imagery in many application domains, including satellite imaging and aerial reconnaissance, there is a clear need for parallel algorithms able to process these high-dimensional data sets quickly enough for practical use. This paper describes a parallel implementation of a multichannel classification algorithm that naturally combines the information provided by all dimensions. The parallel algorithm first defines multichannel morphological operations, and then uses these operations to extend the morphological watershed trans-formation to multidimensional images. The proposed technique has been implemented on Thunderhead, a Beowulf cluster at NASA's Goddard Space Flight Center, and tested using remotely sensed data collected by the Jet Propul-sion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS).

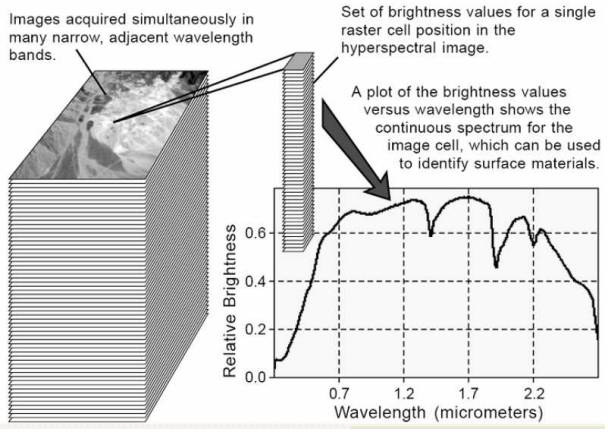

Figure 1. Graphical representation of a hyperspectral image

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. These images are characteristic in satellite imaging and aerial reconnaissance applications, where the number of spectral channels can be extremely high (in the order of hundreds or thousands of bands) as in the case of hyperspectral images [2, 4], often used by agencies such as NASA to analyze the surface of the Earth (see Fig. 1).

One of the most successful techniques for image information extraction in the recent literature has been mathematical morphology [13], which relies on the definition of a structuring element (SE) which is translated over the image, thus acting as a probe for extracting or suppressing specific structures of the image objects. An important approach for morphological analysis in the literature is the morphological watershed transformation [5], which 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 several applications [6],

0-7695-3078-8/08 $25.00 © 2008 IEEE
DOI 10.1109/SYNASC.2007.78

the watershed transformation has not been fully exploited in multichannel image analysis, mainly due to the enormous amount of data to be processed in applications such as remote sensing, which results in high computational requirements.

Parallel processing and the new processing power offered by commodity cluster-based systems can help to tackle large multidimensional image data sets and to get reasonable response times in complex image analysis scenarios [11]. This paper describes a new parallel algorithm that allows for efficient exploitation of multichannel image data via an extended watershed algorithm. The paper is organized as follows. Section 2 briefly describes our proposed strategy to extend mathematical morphology to multichannel images. Section 3 develops a parallel implementation of the proposed algorithm. In Section 4, the parallel algorithm is evaluated in the context of a precision agriculture application, supported by the utilization of remotely sensed hyperspectral imagery collected by the Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS). Important parallel properties such speedup, load balance and parallel efficiency are investigated via experiments on Thunderhead, a Beowulf cluster at NASA's Goddard Space Flight Center. Finally, Section 5 concludes with some remarks and hints at plausible research.

2. Extended Mathematical Morphology

Our attention in this section focuses on the development of a mechanism to extend basic morphological operations (erosion and dilation) to multichannel imagery [8]. Both operations rely on the assumption that there exists an ordering relation among the pixels of a given scene. 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 pixel vectors in the multichannel image data [7]. To accomplish the above goal, we adopt a distance-based technique which utilizes a cumulative distance between one particular pixel vector $\mathbf{f}(x, y)$, where (x, y) indicates the spatial coordinates, and all the pixel vectors in the spatial neighborhood given by a SE B as follows:

$$C_B(\mathbf{f}(x,y)) = \sum_{(s,t)} SAD(\mathbf{f}(x,y), \mathbf{f}(s,t)) \qquad (1)$$

where *SAD* is the spectral angle distance. As a result,$C_B(\mathbf{f}(x, y))$ is given by the sum of *SAD* scores between $\mathbf{f}(x, y)$ and every other pixel vector in the B-neighborhood. At this point, we need to be able to define a maximum and an minimum given an arbitrary set of vectors $\mathbf{S} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_p\}$, where p is the number of vectors in the set. This can be done by computing $C_B(\mathbf{S}) = \{C_B(\mathbf{v}_1), C_B(\mathbf{v}_2), \cdots, C_B(\mathbf{v}_p)\}$ and selecting $\mathbf{v}_j$ such that $C_B(\mathbf{v}_j)$ is the minimum of $C_B(\mathbf{S})$, with $1 \leq j \leq p$. In similar fashion, we can select $\mathbf{v}_k$ such that $C_B(\mathbf{v}_k)$ is the maximum of $C_B(\mathbf{S})$, with $1 \leq k \leq p$. Based on the definitions above, the extended erosion $\mathbf{f} \ominus B$ consists of selecting the B-neighborhood pixel vector that produces the minimum C_B value. On the other hand, the extended dilation $\mathbf{f} \oplus B$ selects the B-neighborhood pixel that produces the maximum value for C_B. As a result, the morphological gradient at $\mathbf{f}(x, y)$ using B can be defined as:

$$G_B(\mathbf{f}(x,y)) = SAD((\mathbf{f} \oplus B)(x,y), (\mathbf{f} \ominus B)(x,y)) \qquad (2)$$

3. Multichannel Watershed Classification

Based on the concepts introduced the previous section, we provide below a multichannel watershed classification algorithm which consists of three main stages:

1. *Minima selection*. In order to select 'markers' or minima from which the watershed transform is started, we hierarchically order all minima according to their deepness, and then select only those above a threshold. The deepness of a basin is 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 class of morphological transformation that does not introduce discontinuities). Given the multichannel gradient $G_B(\mathbf{f})$ of an n-dimensional image, the morphological reconstruction of $G_B(\mathbf{f})$ from $G_B(\mathbf{f}) \ominus B$ has a watershed transform in which the regions with deepness lower than a certain value have been joined to the neighbor region with closer spectral properties.

2. *Flooding*. Let the set $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_k\}$ denote the set of k minimum pixel vectors resulting from minima selection. Let the catchment basin associated with a minimum pixel $\mathbf{p}_i$ be denoted by $CB(\mathbf{p}_i)$. The catchment basins are 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 $D_B(\mathbf{p}_l)$, where $\mathbf{p}_l$ is the deepest pixel in $\mathbf{P} - \{\mathbf{p}_j\}$. This operation is repeated until $\mathbf{P} = \emptyset$.

3. *Region merging*. To obtain the final classification, some of the regions $\{CB(\mathbf{p}_l)\}_{l=1}^{k}$ 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 $e(\mathbf{p}_i, \mathbf{p}_j)$ is the value of $SAD(\mathbf{p}_i, \mathbf{p}_j)$. Regions $CB(\mathbf{p}_i), CB(\mathbf{p}_j)$ can be merged attending to spatial properties in the case of adjacent regions, and also according to pixel vector similarity criteria for non-adjacent regions. The output of the algorithm above is a 2-D classification based on the set of $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_k\}$ class prototypes.

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. In this section, our goal is to develop an efficient and scalable parallel implementation of the algorithm provided in Section 3. Before describing our implementation, design features such as partitioning, task replication and communication are discussed.

4.1. Partitioning

Two types of data partitioning can be applied for multichannel images [11]: spectral-domain partitioning [see Fig. 2(left)] and spatial-domain partitioning [see Fig. 2(right)]. Our parallel algorithm uses spatial-domain partitioning, that is, the original multichannel image $\mathbf{f}$ is decomposed into subimages using a standard scatter operation, 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 decision of partitioning the data in the spatial domain instead of the spectral domain. 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 since the proposed algorithm uses the spectral information *as a whole*.

4.2. Task Replication

An important issue in SE or kernel-based image processing operations is that accesses to pixels outside the domain $D_{\mathbf{f}}$ 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.

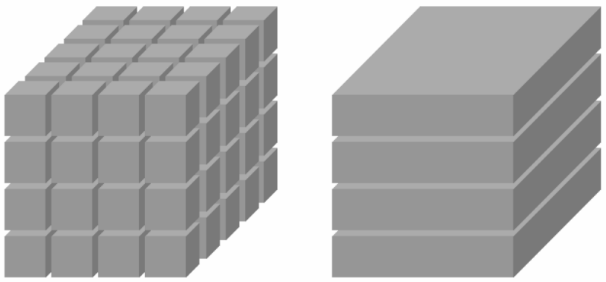

Figure 2. Spectral (left) and spatial (right) hyperspectral image partioning. This figure originally appeared in [10]

However, when such a border effect happens in a local partition $D_{\mathbf{f}_i}$, then additional inter-processor communications may be required when the SE-based (or flooding) computation needs to be split amongst several different processing nodes. In this case, the computations for the pixel vector at a certain spatial location need to originate from several processors, and a communication overhead involving several pixel vectors is introduced. In order to avoid such an overhead, edge/corner pixels may be replicated in the neighboring processors whose subdomains are thus enlarged with a so-called extension or overlap 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. In order to deal with the flooding process in parallel, further operations are required as will be described in the following subsection.

4.3. Parallel Implementation

Our implementation of the parallel multichannel watershed algorithm uses a simple master-slave model. The master processor reads the whole multichannel image $\mathbf{f}$ and divides it into a set of multichannel subimages $\mathbf{f}_i$ which are sent to different processors. The slave processors run the watershed algorithm on the respective subimages and also exchange data among themselves for uniform classification. After the classified regions become stable, the slaves send the output to the master, which combines all of them in a proper way and provides the final classification. 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 $p-1$ remaining processors create a local queue Q_i with $1 \leq i \leq p-1$. 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, however, that due to the im-

age 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 classification labels of pixels in the boundary with appropriate neighboring processors. Subsequently, a processor may receive classification 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 classification labels at the local subdomain. Communication and reflooding are again repeated until stabilization (i.e. no more changes occur).

5. Experimental Results

This section reports on the effectiveness of the proposed parallel classification algorithm in the context of a realistic precision agriculture application, where remotely sensed hyperspectral data collected by the by the hyperspectral AVIRIS sensor are used to test the performance of the algorithm. First, we provide an overview of the parallel computing architecture used for evaluation purposes. Then, a computational cost-performance analysis of the algorithm is presented and discussed.

The parallel computing architecture used for experiments is the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center (see Fig. 3. The system is currently composed of 256 2.4 GHz Intel Xeon nodes, each with 1 GB of memory and 80 GB of main memory (see http://newton.gsfc.nasa.gov/thunderhead). The total peak performance of the system is 2457.6 GFlops. Along with the 256-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.

The image data set used in experiments was a 224-band AVIRIS scene taken at a low altitude with a 3.7 meter-pixel size. It was collected over an agricultural test site located in Salinas Valley, California, and represents a challenging classification problem. Fortunately, extensive ground-truth information is available for the area, allowing a quantitative assessment in terms of classification accuracy. Fig. 4(left) shows a spectral band of the scene, and Fig. 4(right) shows the 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), and the total size of the image is 137 MB.

Table 1 shows the classification accuracies obtained by the proposed parallel algorithm (using different square-shaped SEs), compared to those achieved by other two widely used parallel classification techniques for hyperspectral imagery: the D-ISODATA [3] algorithm (a parallel, multichannel version of the ISODATA method - which is widely regarded as the benchmark for all unsupervised classification algorithms) and the S-PCT [1] (a spectral screening algorithm which utilizes the principal component transform to decorrelate the data prior to classification). As shown by Table 1, the best classification scores were obtained by the proposed method with a disk-shaped SE of 15-pixel radius, denoted by B_{15}^{disk}. Overall, results in Table 1 reveal that the proposed multichannel watershed algorithm could achieve superior classification results than those provided by other standard methods in a complex analysis scenario given by agricultural classes with very similar spectral features. For illustrative purposes, Table 1 also shows the single-processor execution times for the considered algorithms, measured in Thunderhead node. As noted, the times reported on the table are significantly high.

In order to investigate the efficiency of the three algorithms in Table 1, we implemented them in parallel using the C++ programming language with calls to message passing interface (MPI). Table 2 shows the measured execution times, speedup factors (with regards to the single-processor

Figure 3. Thunderhead Beowulf cluster.

Table 1. Classification accuracy (in percentage) and single-processor execution time (in seconds) obtained by the different parallel algorithms after processing the Salinas AVIRIS image on Thunderhead.

Method tested	Multichannel watershed with SE: B_3^{disk}	B_7^{disk}	B_{11}^{disk}	B_{15}^{disk}	S-PCT	D-ISODATA
Time	2549	6399	8209	13728	41239	49912
Acc.	77.48	84.25	89.65	93.84	82.25	69.84

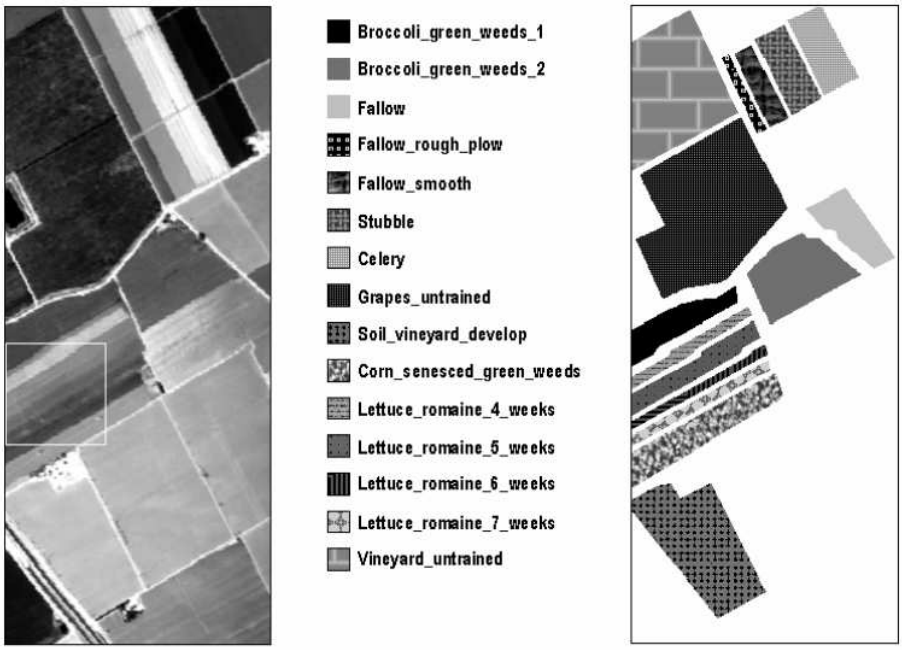

Figure 4. Spectral band at 488 nm of an AVIRIS scene collected over Salinas Valley, California (left) and land-cover ground-truth classes (right). This figure originally appeared in [12]

executions in Table 1) and imbalance scores for different multi-processor runs of the parallel algorithms on Thunderhead. Here, 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 $D = 1$, which means that the maxima and minima times across all processors are the same.

In Table 2, we report the imbalance considering all processors, D_{All}, and also considering all processors but the root, D_{Minus}. As shown by the table, the parallel watershed algorithm scaled reasonably well on Thunderhead, in particular, for very large SE sizes (results for two SE sizes, B_7^{disk} and B_{15}^{disk} are reported on the table). Regarding load balance, it is also clear from Table 2 that the multichannel watershed algorithm always showed similar values of D_{All} and D_{Minus}. Quite opposite, scores for D_{All} are always higher than those for D_{Minus} in the S-PCT and D-ISODATA algorithms (this means that the master node has higher computational load in those cases).

It is also clear from Table 2 that the watershed algorithm provided higher values of D_{All} and D_{Minus} (and hence the highest imbalance) when the size of the considered SE was B_7^{disk}. On the other hand, when the size of the SE was increased to B_{15}^{disk}, the same algorithm resulted in values of D_{All} and D_{Minus} which were very close to 1. This indicated that the proposed parallel method achieved better load balance as the problem size was increased. Finally, the execution times reported on Table 2 also reveal that only the parallel watershed algorithm was able to provide a classification response in near real-time (it should be noted that the classification accuracy of the parallel watershed algorithm was also higher than that found by the other methods, in particular, when large SEs were used). For instance, the watershed algorithm implemented with For instance, the watershed algorithm implemented with B_{15}^{disk} provided a highly accurate classification result in about five minutes using 64 processors, and it only required 106 seconds to complete the calculations when all available processors were used (S-PCT and D-ISODATA required twice and three times as much time, respectively).

Overall, results in Table 2 reveal that the proposed parallel watershed algorithm offers significant improvements over the single-processor implementation, especially in light of the very high execution times reported on Table 1. provided a highly accurate classification result in about five minutes using 64 processors, and it only required 106 seconds to complete the calculations when all available processors were used (S-PCT and D-ISODATA required twice and three times as much time, respectively).

Results in Table 2 also reveal that the proposed parallel watershed algorithm offers significant improvements over the single-processor implementation, especially in light of the very high execution times reported on Table 1.

Table 2. Execution time (T), speedup (S), and imbalance (D) scores for the parallel algorithms using different numbers of processors on Thunderhead.

Number of processors:		4	16	36	64	100	144	196	256
Multichannel watershed with SE: B_3^{disk}	T	967	268	179	145	107	71	56	39
	S	2.63	9.56	14.34	17.43	23.45	35.10	45.67	62.45
	D_{All}	1.16	1.12	1.09	1.10	1.08	1.12	1.07	1.05
	D_{Minus}	1.04	1.03	1.05	1.03	1.02	1.02	1.03	1.01
Multichannel watershed with SE: B_{15}^{disk}	T	4250	1222	465	296	215	158	134	106
	S	3.23	11.23	29.45	46.34	63.45	86.28	102.34	129.23
	D_{All}	1.12	1.10	1.07	1.09	1.07	1.11	1.06	1.05
	D_{Minus}	1.02	1.01	1.03	1.02	1.01	1.01	1.02	1.01
S-PCT	T	13521	4314	1759	884	572	392	314	265
	S	3.05	9.56	23.45	46.67	72.12	105.23	131.23	155.45
	D_{All}	1.47	1.35	1.32	1.30	1.23	1.24	1.21	1.19
	D_{Minus}	1.08	1.05	1.05	1.04	1.03	1.04	1.04	1.03
D-ISODATA	T	21330	5907	2428	1299	865	630	444	386
	S	2.34	8.45	20.56	38.43	57.67	79.23	112.34	129.21
	D_{All}	1.74	1.62	1.57	1.42	1.39	1.28	1.31	1.27
	D_{Minus}	1.13	1.08	1.10	1.04	1.07	1.05	1.07	1.06

6. Conclusions and Future Lines

We have developed an innovative parallel technique for unsupervised classification of multichannel image data sets. The proposed MPI-based parallel implementation enhances load balance, and can be ported to any type of distributed memory system. Experimental results reveal that the parallel algorithm also achieved good results in terms of classification accuracy and scalability in the context of a remotely sensed hyperspectral image analysis application. As future work, we plan on implementing the parallel algorithm using other high performance parallel computing architectures, such as heterogeneous networks of workstations [9]. We are currently working towards the integration of the algorithm onto a near real-time forest fire tracking system in conjunction with Junta de Extremadura (local government).

7. Acknowledgement

This research was supported by the Marie Curie Research Training Network project entitled 'Hyperspectral Imaging Network' (HYPER-I-NET), contract number MRTN-CT-2006-035927. The author would like to thank Drs. John E. Dorband, James C. Tilton and J. Anthony Gualtieri for many helpful discussions.

References

[1] T. Achalakul and S. Taylor. A distributed spectral-screening pct algorithm. *Journal of Parallel and Distributed Computing*, 63:373–384, 2003.

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

[3] M. K. Dhodhi, J. A. Saghri, I. Ahmad, and R. Ul-Mustafa. D-isodata: A distributed algorithm for unsupervised classification of remotely sensed data on network of workstations. *Journal of Parallel and Distributed Computing*, 59:280–301, 1999.

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

[5] N. Malpica, J. Ortuno, and J. Santos. A multichannel watershed-based algorithm for supervised texture segmentation. *Pattern Recognition Letters*, 24:1545–1554, 2003.

[6] A. Moga and M. Gabbouj. Parallel marker-based image segmentation with watershed transformation. *Journal of Parallel and Distributed Computing*, 51:27–45, 1998.

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

[8] A. Plaza, P. Martinez, J. Plaza, and R. Perez. Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):466–479, 2005.

[9] A. Plaza, J. Plaza, and D. Valencia. Impact of platform heterogeneity on the design of parallel algorithms for morphological processing of hyperspectral image data. *Journal of Supercomputing*, 66(3):81–107, 2007.

[10] A. Plaza, J. Plaza, D. Valencia, and P. Martinez. Efficient information extraction from hyperspectral imagery using heterogeneous networks of workstations. *IEEE Intl. Geoscience and Remote Sensing Symp.*, 8:5598–5601, 2006.

[11] 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.

[12] 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. *IEEE Intl. Conf. on Cluster Computing*, 8:150–158, 2006.

[13] P. Soille. *Morphological image analysis: Principles and applications, Second Edition*. Springer-Verlag: Berlin, 2003.