The Divide and Segment Method for Parallel Image Segmentation

Thales Sehn Körting, Emiliano Ferreira Castejon, and Leila Maria Garcia Fonseca

Brazil's National Institute for Space Research – INPE
Image Processing Division – DPI Av. dos Astronautas, 1758
São José dos Campos, Brazil
{thales,castejon,leila}@dpi.inpe.br

Abstract. Remote sensing images with large spatial dimensions are usual. Besides, they also include a diversity of spectral channels, increasing the volume of information. To obtain valuable information from remote sensing data, computers need higher amounts of memory and more efficient processing techniques. The first process in image analysis is segmentation, which identifies regions in images. Therefore, segmentation algorithms must deal with large amounts of data. Even with current computational power, certain image sizes may exceed the memory limits, which ask for different solutions. An alternative to overcome such limits is to employ the well-known divide and conquer strategy, by splitting the image into chunks, and segmenting each one individually. However, it arises the problem of merging neighboring chunks and keeping the homogeneity in such regions. In this work, we propose an alternative to divide the image into chunks by defining noncrisp borders between them. The noncrisp borders are computed based on Dijkstra algorithm, which is employed to find the shortest path between detected edges in the images. By applying our method, we avoid the postprocessing of neighboring regions, and therefore speed up the final segmentation.

1 Introduction

Remote sensing images are the only source capable of providing a continuous and consistent set of information about the Earth's land and oceans [8]. Combined with ecosystem models, remotely sensed data offers an unprecedented opportunity for predicting and understanding the behavior of the Earth's ecosystem [26]. Since the 1970s, the Landsat series of satellites have provided optical images of the land's surface of the Earth every 16 days at a resolution of 30 meters. The Landsat archive at the United States Geological Survey contains about 1 petabyte and is fully accessible worldwide [9]. From 2013 onwards, a new generation of optical remote sensing satellites from USA, China, Brazil, India and Europe will produce in one year as much data as 10 years of the Landsat-7 satellite.

However, our methods to analyze and understand massive datasets lag far behind our ability to produce and store this data [10,13,28]. And besides, it is

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02895-8_64

J. Blanc-Talon et al. (Eds.): ACIVS 2013, LNCS 8192, pp. 504–515, 2013.

still far from easy to search across large collections of satellite images for pictures containing a golf course, or a hurricane [14]. Therefore image analysis over large databases needs to be in the research agenda of the remote sensing community.

During the 1980s and 1990s, most remote sensing image analysis techniques were based on per-pixel statistical algorithms [7]. These techniques aimed at representing the knowledge about land cover patterns by a limited set of parameters, such as average and standard deviation values of groups of individual pixels. Recently, Object-Based Image Analysis (OBIA) has shown to be a good alternative to traditional per-pixel and region based approaches. Differently, OBIA approaches first identify regions in the image using segmentation, extract neighborhood, spectral and spatial descriptive features and afterwards combine regions and features for object classification. Although segmentation has a large tradition in image processing [16] and remote sensing [9], OBIA took a long time to reach mainstream users. This approach became popular when it combined image segmentation with good labeling methods that match the features to those of user-defined classes. However, remote sensing image analysis using OBIA can be lengthy and complex because of the difficulties related to image segmentation, the large number of features to be resolved [21] and the many different methods needed to model the semantic networks [17].

Segmentation of remote sensing images is a challenging field. Their results are expected to describe the regions found in images, allowing a deeper interpretation by experts or classification algorithms. The work of [16] defined segmentation as a way to separate the image into simple regions with homogeneous behavior. To partition automatically one image into regions, algorithms must consider the context, scale, neighborhood, meaning, and computational resources. In accord to [28], good quality results often come at the price of high computational cost. For example, the collection rate for IKONOS satellite is about 890 megapixels each minute [10]; for CBERS-2B is about 120 megapixels each minute. Considering current technology, even a tuned sequential segmentation algorithm is far slower than these rates.

Remote sensing images often present large sizes. A typical Landsat scene contains at least 7800×7100 pixels, and 6 spectral channels with 30m resolution (bands 1 to 5, and 7), that results in more than 300 million individual pixels. The variety of spectral channels, that in one side adds rich information about the land targets, in the other side increases the volume of information. Even with current computational power, certain sizes may exceed the memory limits, claiming new solutions. Research in segmentation techniques for large images points out the division of the image into blocks of predefined sizes, hereby called *chunks*. These chunks are segmented independently, and a postprocessing step is needed to merge the segmentation results into a single one. The problem of this approach is to merge the neighboring chunks without prejudicing the homogeneity in bordering regions. Figure 1 shows one example of this problem. The image was divided into crisp chunks and afterwards each one was segmented independently. The postprocessing in this case would not merge the regions highlighted in red (note the object of the type *roof*), because they present different attributes, like

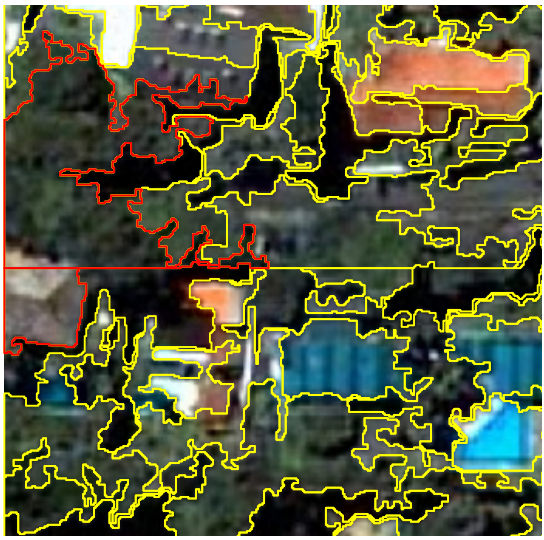

Fig. 1. Example of the traditional parallel segmentation. Regions highlighted in red will not be merged properly.

average pixel value or standard deviation of pixels. The piece of the roof in the top region was too small to create an individual region, and therefore was merged in the region containing trees. Due to this problem, the bottom region with the rest of the roof will not be merged, because the spectral difference between these two regions is too high.

Another problem of this approach is the time processing. For example, suppose one image with 5000 × 5000 pixels, divided in two chunks of 2500 lines × 5000 columns, and a segmentation which created regions with an average area of 100 × 100 pixels. The border between these chunks will have at least 50 regions for each chunk, and the regions from one chunk will touch at least 2 regions from the other chunk. In this case, the algorithm will have to perform 100 tests between these regions to check whether they must be merged or not. Such comparisons include data access to evaluate average pixel values, vector differences, check if candidate merged regions will comply the segmentation parameters and so on. We argue that this step does not produce good results, and improve significantly the processing time. Our proposal is to convert this postprocessing into a preprocessing stage, cutting out the need to perform exhaustive tests in resultant regions.

In this article we tackle the problem of creating chunks for parallel segmentation. Instead of creating crisp chunks using the block strategy (which creates crisp borders), we analyze the image contours in the chunks' neighborhood, and create adaptive chunks defined by these contours. We argue that by defining noncrisp borders between the chunks we avoid the postprocessing of neighboring regions, remove the segmentation errors in the borders, and speed up the final segmentation.

2 Related Work

Several image segmentation techniques arise from the well-known "region growing" strategy, relying on the similarity of near pixels. The method starts by defining candidate pixels (called seeds) in random positions of the image. Such seeds are compared to neighboring pixels, and according to their similarity, they are merged into homogeneous regions, therefore the pixels grow into larger regions. This process is repeated until all pixels are processed. [6] applied this approach in remote sensing images. Their method is based on the likeness between neighboring pixels and the smallest area allowed for a region. [2] is another example of region growing technique. In this approach, the algorithm minimizes the average heterogeneity of the regions. The heterogeneity balances the object's smoothness and compactness, resulting in more regular objects. It deals with the standard deviation of pixels for each band as well. We suggest the reading of [20] for a comparison of these two algorithms and alternatives for remote sensing segmentation.

To take advantage of the spatial similarity between neighboring pixels, the graph-based techniques create region adjacency graphs considering pixels as edges, and the differences between neighbor pixels as the edges. The work [12] presented the image foresting transform (IFT), a generalization of Dijkstra's algorithm, which is a graph-based approach to the design of image processing operators based on connectivity. This method considers one image as a directed graph whose nodes are the image pixels and whose arcs are the neighboring pixel pairs. It was applied to find homogeneous regions in 3D images and to perform boundary tracking, whose goal is to estimate an optimal curve, constrained to a given sequence of landmarks on the object's boundary.

[24] proposed an approach that extracts the global impression of an image, by treating image segmentation as a graph partitioning problem. The authors proposed a novel global criterion, called the normalized cuts, that measures both the total dissimilarity between the different groups of pixels as well as the total similarity within the groups. The algorithm has been tested in static images as well as motion sequences. The main idea is to create an adjacency matrix connecting all pixels in the image, and after perform the bipartitioning of the graph through normalized cuts, where the partitions will define the homogeneous regions in the image.

In the area of merging for creating mosaics of remote sensing images, [4] presented a blending technique, based on multi-resolution decomposition. The authors defined a cut line, considering texture information from overlapping regions of mosaicking images. The method found automatically the transition zone size and the cutting line on satellite and aerial images based on the minimum path using Dijkstra's algorithm.

The work of [5] presented one algorithm based on the morphological image compositing technique and applied to automatically generate a European wide image mosaic based on over 800 Landsat ETM+ scenes. A quantitative measure was also developed to estimate the quality of automatically delineated borders.

According to [19], parallel architectures are becoming a standard for handling complex operations that need significant computational power. Large size images include medical data sets of magnetic resonance imaging (MRI) [23], or remote sensing hyperspectral and multitemporal images [27,22]. Therefore parallel algorithms arise as an alternative to overcome those limits. As previously stated, such methods usually split the image into chunks, and segment each one individually.

The chunks often present regular sizes to be assigned equally among the processors [3]. However, according to [28] the results of coupling chunks are not acceptable because border objects are not correctly handled. A common solution is to adopt overlapping chunks, which is also inadequate because there is no upper bound on the size of objects of interest (e.g. rivers or roads). [25] proposed a parallel method for the seeded region growing algorithm ([1]), based on spreading seeds in different threads, each one growing in parallel. The authors needed to deal with simultaneous access for the same pixels. To avoid this problem, images were divided in square windows, employing a postprocessing step to join regions.

The technique proposed by [15] employed the traditional parallel segmentation. Each chunk was processed by a different thread, through a sequential algorithm based on [2]. This method falls on the same problem of treating boundary segments. The number of boundary objects, which depends on the image, can be prohibitive in certain cases.

3 Method

This work extends our previous research in parallel segmentation [18]. In the present approach, the algorithm finds automatically optimal cutting lines to divide the image into noncrisp chunks. Traditional parallel schemes first divide the image into crisp chunks, and after treat bordering regions in a postprocessing step. Some methods create chunks with overlapping regions, but fall into the same problem of postprocessing. Figure 2 describes our approach, and as follows we describe each step in detail.

3.1 Create Image of Edges

The first step is to obtain an image of edges based on the input data. From the basics of image processing, the well-known method to estimate the magnitude of the edges is the gradient function. Considering that in one image every pixel can be described as a function $f(x, y)$, where x and y are the coordinates of the pixel, the gradient is computed as the two-dimensional column vector:

$$\nabla \mathbf{f} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix} \tag{1}$$

which is a vector that indicates, for each pixel, the intensities of the border in horizontal and vertical directions. The magnitude of this vector points out the border's strength, and is computed by the following equation:

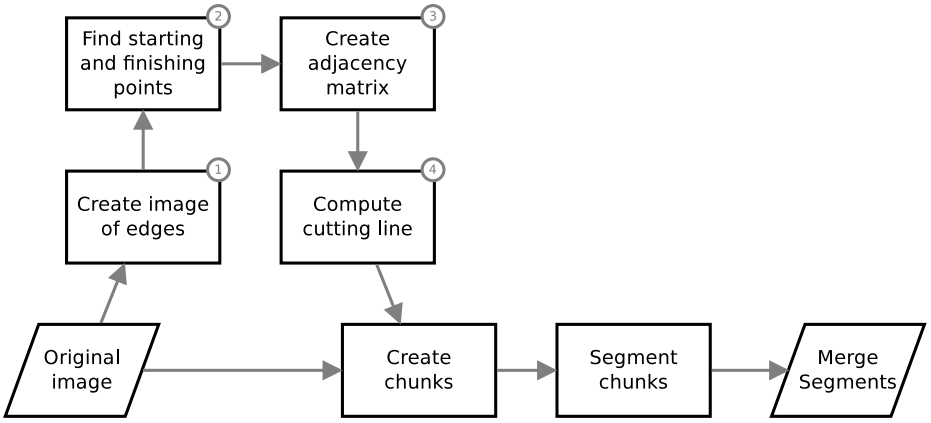

Fig. 2. Main diagram to find the optimal cutting line that divides the image into noncrisp chunks

$$mag(\nabla \mathbf{f}) = \left[\left(\frac{\delta f}{\delta x} \right)^2 + \left(\frac{\delta f}{\delta x} \right)^2 \right]^{\frac{1}{2}}. \tag{2}$$

By applying these equations to all pixels of the image we obtain an image of edges. This image will be processed to find the optimal cutting line. The chunks must be defined with an average size (width and height) and a cutting line will be adapted according to the image of edges. Figure 3 shows one example of input image and its image of edges. Suppose this example image will be divided into 2 chunks. Therefore, a middle line (shown in yellow in Figure 3) is the candidate line. This candidate line will adapt to the shortest path between left and right edges. We also define a *region of interest*, limited by a maximum displacement between this candidate line to avoid chunks with very different sizes (shown in red in the Figure).

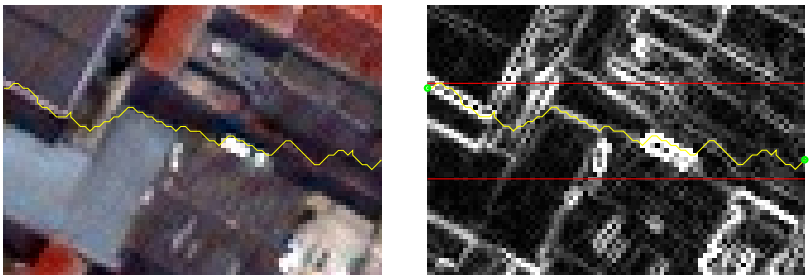

Fig. 3. The example input image (left) and its image of edges (right)

3.2 Find Starting and Finishing Points

To compute the cutting line, we must previously define starting and finishing points. We use the maximum value in the image of edges (higher magnitude in the gradient stands to the strongest border) at the beginning and end of the chunk. Considering the image of edges in Figure 3, they are represented by the higher values in the first and last columns, highlighted in green.

After, we create an adjacency graph (represented by a matrix) that connects all pixels in the candidate region. The Dijkstra's algorithm will be used to find the best path between starting and finishing points. This path will define the cutting line used to divide the image into noncrisp chunks.

3.3 Create Adjacency Matrix

The adjacency matrix is a graph, whose nodes are the image pixels and whose arcs are defined by an adjacency relation between pixels. The cost of a path in this graph, according to [12], is determined by an application-specific path-cost function, which usually depends on local image properties along the path, such as color, gradient, and pixel position. In our approach, the adjacency between the pixels is defined by 5 connections, including top, top-right, right, bottom-right and bottom pixels, as shown in Figure 4.

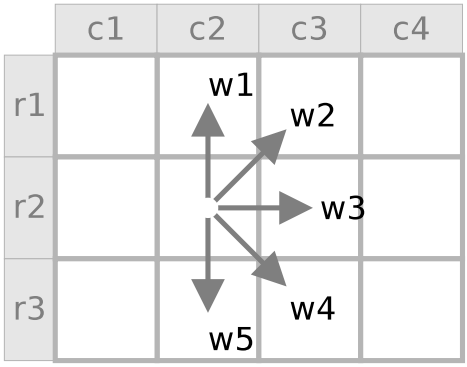

Fig. 4. The 5 weights associated to each pixel to build the adjacency matrix. The directions are top, top-right, right, bottom-right and bottom (w1 to w5).

The associated cost to each weight is defined by the magnitude of the gradient, therefore the highest magnitude has a lower cost. Since we defined 5 connections, we rank the cost with values in the interval [1, 5], where 1 is the highest magnitude and 5 is the lowest. The choice of using 5 connections is to remove the adjacency between a pixel and its left side, so the discovered path will always have the direction tending to the right[1].

[1] This method is easily extensible to vertical cutting lines. In this case, the adjacency graph will be created to maintain a direction tending to the bottom, and the starting and finishing points will be defined in the top and in the bottom of the chunk.

3.4 Compute Cutting Line

Since we defined the starting and finishing points, and the adjacency matrix, it is possible to compute the best path between these two points. For this, we employ the Dijkstra's algorithm [11] using the weights between the pixels to find the best cutting line. This line will define the noncrisp border between the chunks, and will allow the segmentation algorithm to run independently in both chunks. The resulting segmentation will be the simple merge of regions detected in all chunks.

4 Results and Discussion

To evaluate the method we performed 2 experiments using images with different spatial and spectral resolutions. The purpose of the following experiments is to show that the resultant cutting line divided the image into independent chunks, in terms of the resultant regions from segmentation. For the sake of comparison, we tested our previous approach [18] in the same images, using equivalent parameters.

In the first experiment we used an image crop of a World View 2 scene from São José dos Campos, Brazil, with 3200 × 2400 pixels. The pixels of this image have a spatial resolution of 0.5m. The image was obtained in 2012. We defined the region of interest with a size of 120 pixels. Figure 5 (top) shows the detected cutting line. After this step, we applied segmentation (based on [6]) in both chunks and obtained the regions depicted in Figure 6.

Comparing our results, the cutting line created by our previous approach is shown in Figure 5 (bottom). It is possible to see that both results followed the edges inside the region of interest. By applying the Dijkstra's algorithm the cutting line got a smoother transition between pixels, while the previous approach resulted in a line with several spikes. It is noticeable that the cutting line divided the road (in the left part of the image) in two pieces, which is a limit of our method. When the object does not follow the direction of the cutting line (in this example, horizontal direction), the segmentation will produce divided regions. The second mistake is in the grass field; however this mistake were caused because of the size of the region of interest.

The second experiment used an image crop of a Quickbird scene from São Paulo, Brazil, with 1000 × 1175 pixels. The pixels of this image have a spatial resolution of 0.6m. We defined the region of interest with a size of 150 pixels. Figure 7 (top) shows the resultant cutting line. After this step, we applied the segmentation in both chunks (also using the approach based on [6]) and obtained the regions depicted in Figure 8. The cutting line produced by our previous approach is shown in Figure 7 (bottom). It is possible to see that in this result, the cutting line is not smooth, and presents high and low peaks along the line.

Fig. 5. First experiment: the cutting line created by our approach (top), in comparison to our previous method (bottom).

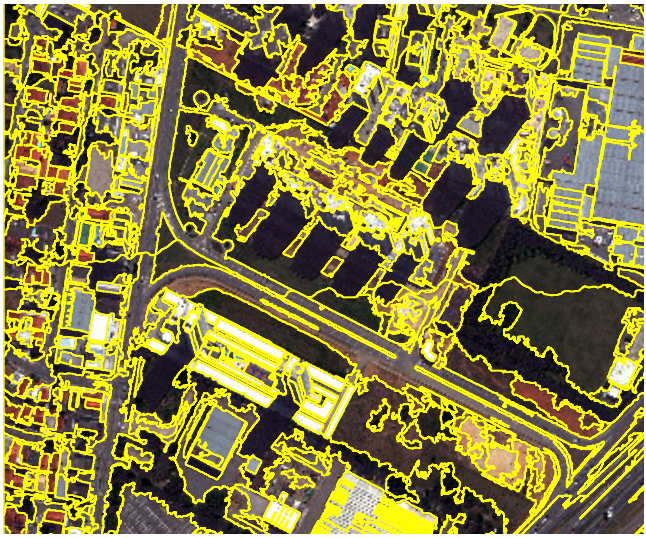

Fig. 6. Detail of the resulting segmentation, after merging segments from both chunks

Fig. 7. Second experiment: the cutting line created by our approach (top), in comparison to our previous method (bottom)

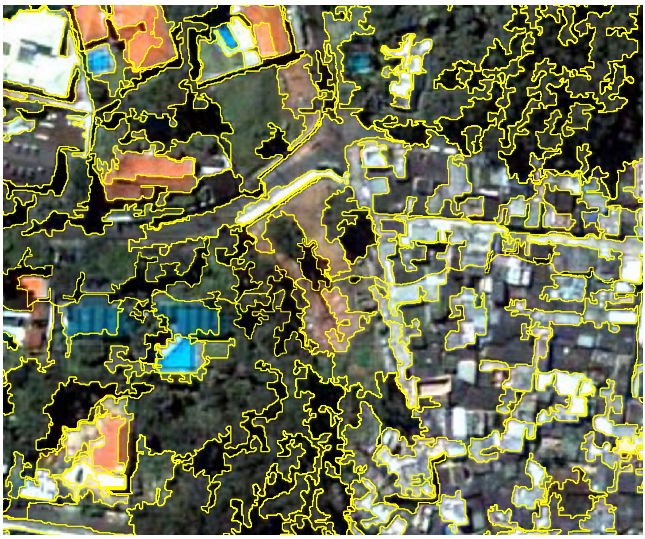

Fig. 8. The resulting segmentation, after merging segments from both chunks

5 Conclusion

This article tackled the specific problem of defining chunks for parallel segmentation. Current methods create crisp chunks, needing postprocessing steps to get final regions. In certain cases, such methods create inconsistent objects, demanding computational power to deal with bordering regions. Postprocessing detects the bordering regions, testing the best combination of regions to merge. This step aims to keep the consistence of the new regions to specific segmentation

parameters, as spectral homogeneity and size. Our method proposes to change this postprocessing step into a preprocessing stage, by creating adaptive cutting lines and dividing the images into chunks without crisp borders.

However, the shape of certain objects near the cutting line does not allow creating a proper cutting line, like homogeneous regions whose size extrapolates the region of interest defined by the region of interest. Therefore dealing with these regions still remains an open problem, currently unsolved by our approach. Albeit in the results section we showed only horizontal chunks, this method easily extendable to vertical cutting lines. Another point is that Dijkstra's method is a greedy algorithm, which depending on the size of the adjacency matrix, can become a bottleneck in this operation. Future works on this issue include subdividing the cutting line into smaller subsets, and applying our approach individually on each subset, then recomposing the final cutting line by merging all subsets.

Acknowledgments. The authors acknowledge Digital Globe for providing the WorldView 2 imagery used in this article.

References

1. Adams, R., Bischof, L.: Seeded region growing. Pattern Analysis and Machine 16 (1994)
2. Baatz, M., Schape, A., Schäpe, M.: Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation. In: Wichmann-Verlag (ed.) XII Angewandte Geographische Informationsverarbeitung, pp. 12–23. Herbert Wichmann Verlag, Heidelberg (2000)
3. Bader, D., Jaja, J., Harwood, D., Davis, L.: Parallel algorithms for image enhancement and segmentation by region growing with an experimental study. In: Proceedings of International Conference on Parallel Processing, pp. 414–423 (1996)
4. Bagli, V., Fonseca, L.: Seamless mosaicking via multiresolution analysis and cut line definition. In: Signal and Image Processing. ACTA Press (2006)
5. Bielski, C., Grazzini, J., Soille, P.: Automated morphological image composition for mosaicing large image data sets. In: IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2007, pp. 4068–4071 (2007)
6. Bins, L., Fonseca, L., Erthal, G., Ii, F.: Satellite imagery segmentation: a region growing approach. Simpósio Brasileiro de Sensoriamento Remoto 8, 677–680 (1996)
7. Blaschke, T.: Object based image analysis for remote sensing. ISPRS Journal of Photogrammetry and Remote Sensing 65(1), 2–16 (2010)
8. Bradley, B., Jacob, R., Hermance, J., Mustard, J.: A curve fitting procedure to derive inter-annual phenologies from time series of noisy satellite NDVI data. Remote Sensing of Environment 106(2), 137–145 (2007)
9. Câmara, G., Souza, R., Freitas, U., Garrido, J., Li, F.: Spring: Integrating remote sensing and gis by object-oriented data modelling. Computers and Graphics 20(3), 395–403 (1996)
10. Dial, G., Bowen, H., Gerlach, F., Grodecki, J., Oleszczuk, R.: IKONOS satellite, imagery, and products. Remote Sensing of Environment 88(1-2), 23–36 (2003)
11. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische mathematik, 269–271 (1959)

12. Falcão, A., Stolfi, J., Lotufo, R.: The Image Foresting Transform: Theory, Algorithms, and Applications. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(1), 19–29 (2004)
13. Fayyad, U., Shapiro, G., Smyth, P.: The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM 39(11), 27–34 (1996)
14. Goodchild, M.F.: Geographic information systems and science: today and tomorrow. Annals of GIS 15(1), 3–9 (2009)
15. Happ, P., Ferreira, R., Bentes, C., Costa, G., Feitosa, R.: Multiresolution segmentation: a parallel approach for high resolution image segmentation in multicore architectures. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2010)
16. Haralick, R., Shapiro, L.: Image segmentation techniques. Applications of Artificial Intelligence II 548, 2–9 (1985)
17. Hay, G., Castilla, G.: Geographic Object-Based Image Analysis (GEOBIA): A new name for a new discipline. In: Blaschke, T., Lang, S., Hay, G. (eds.) Object-Based Image Analysis: Spatial Concepts for Knowledge-Driven Remote Sensing Applications, chap. 1.4, pp. 75–89. Springer, Heidelberg (2008)
18. Körting, T., Castejon, E., Fonseca, L.: Divide and Segment - An alternative for parallel segmentation. In: Proceedings of XII GeoINFO, pp. 97–104. INPE, Campos do Jordão (2011)
19. Lenkiewicz, P., Pereira, M., Freire, M., Fernandes, J.: A new 3D image segmentation method for parallel architectures. In: 2009 IEEE International Conference on Multimedia and Expo, pp. 1813–1816 (June 2009)
20. Meinel, G., Neubert, M.: A comparison of segmentation programs for high resolution remote sensing data. International Archives of Photogrammetry and Remote Sensing 35(Part B), 1097–1105 (2004)
21. Pinho, C., Silva, F., Fonseca, L., Monteiro, A.: Intra-urban land cover classification from high-resolution images using the C4.5 algorithm. ISPRS Congress Beijing 7 (2008)
22. Plaza, A., Plaza, J., Paz, A., Sanchez, S.: Parallel Hyperspectral Image and Signal Processing. IEEE Signal Processing Magazine 28, 119–126 (2011)
23. Prassni, J., Ropinski, T., Hinrichs, K.: Uncertainty-aware guided volume segmentation. IEEE Transactions on Visualization and Computer Graphics 16(6), 1358–1365 (2010)
24. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
25. Singh, D., Heras, D., Rivera, F.: Parallel Seeded Region Growing Algorithm. In: VIII Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes, Bilbao, Spain (1999)
26. Tan, P., Steinbach, M., Kumar, V., Potter, C., Klooster, S., Torregrosa, A.: Finding Spatio-Temporal Patterns in Earth Science Data. Earth Science, 1–12 (2001)
27. Valencia, D., Lastovetsky, A., O'Flynn, M., Plaza, A., Plaza, J.: Parallel Processing of Remotely Sensed Hyperspectral Images on Heterogeneous Networks of Workstations Using HeteroMPI. IJHPCA, 386–407 (November 2008)
28. Wassenberg, J., Middelmann, W., Sanders, P.: An efficient parallel algorithm for graph-based image segmentation. In: Jiang, X., Petkov, N. (eds.) CAIP 2009. LNCS, vol. 5702, pp. 1003–1010. Springer, Heidelberg (2009)