Implementation of a hierarchical segmentation algorithm for radar and optical data using TerraLib

Douglas Messias Uba[1]
Luciano Vieira Dutra[1]
Marinalva Dias Soares[1]
Gilson Alexandre Ostwald Pedro da Costa[2]

[1] Instituto Nacional de Pesquisas Espaciais – INPE
Caixa Postal 515 – 12245-970 – São José dos Campos - SP, Brazil
{douglas, dutra, marinalva}@dpi.inpe.br

[2] Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Caixa Postal 38097 – 22453-900 – Rio de Janeiro - RJ, Brazil
gilson@ele.puc-rio.br

Abstract. This paper describes briefly the underlying algorithm and some of the key implementation aspects of a hierarchical segmentation program called MultiSeg. Built on a previous development called SegSAR, MultiSeg consists on a specialized segmentation technique for SAR and optical imagery. Initially, images are compressed at different rates creating an image pyramid, then a region growing procedure is used in combination with a split and merge technique at the different compression levels. In sequence, the program processes the image pyramid from the coarser to the finer compression levels, applying a border refinement heuristic each time it changes from one level to the next. MultiSeg was created in the C++ language, with the support from an open-source library (TerraLib). The devised software architecture permits easy extension of its capabilities. Additionally, preliminary tests have shown that MultiSeg is capable of processing large volumes of data efficiently. This paper also shows a few examples of segmentation results obtained with MultiSeg, which help to understand the influence of the program's parameters on the segmentation results. Systematic tests with MultiSeg are still under way, aiming at demining the limitations of the program in terms of the data volume, and proper parameter settings for different image types and target object classes. Preliminary results are, however, very promising.

Keywords: remote sensing, image processing, segmentation, radar

1. Introduction

The importance of Microwave Remote Sensing for applications on a wide variety of themes, such as forestry, agriculture and soils, snow and hydrology, oceanography, disaster management, among others is well known. Its main feature is the all-weather acquisition capability, which is especially important for applications in tropical environments.

The new generation of orbital Synthetic Aperture Radar (SAR) platforms is capable of delivering several types of products that can be used to observe and analyze a number of aspects of the land cover, which is often done in conjunction with information obtained by optical sensors, but in some cases radar data is the only choice available.

The main issue with radar data usage and processing is the multiplicative nature of the speckle noise, typical of this type of sensor, which has an impact on image statistics, and is normally very far from the optical, Gaussian hypothesis.

Object-based applications on SAR data usually rely on image segments whose extents are defined either from ancillary data (e.g., a co-registered multispectral image), or from segmentation of the SAR imagery direct, or after a speckle filtering step. Moreover, the majority of such applications rely on segmentation algorithms devised originally for optical data, and hence not properly considering the particular characteristics of radar imaging.

The MultiSeg segmentation algorithm was built on a previous development called SegSAR (Sousa Jr., 2007), a specialized hierarchical segmentation technique for SAR and optical imagery. In MultiSeg, images are compressed at different rates creating an image pyramid. A region growing procedure is used in conjunction with a split and merge technique

at the different compression levels. The algorithm processes the image pyramid from the coarser to the finer compression levels, and applies a border refinement heuristic each time it changes from one level to the next.

MultiSeg can be set to process either SAR or optical images. While processing SAR data, it employs statistical tests based on the Gama distribution, which is consistent with the statistical characteristics of radar data. MultiSeg can also be set to process optical imagery; in that case it relies on statistics that are consistent with the additive noise, Gaussian distribution assumption.

While SegSAR implementation was of prototypical nature, i.e. restricted to the processing of very small images, characterized by a poor computational performance, and built on the ENVI/IDL (Exelis, 2013) proprietary software environment, MultiSeg was implemented on an much more efficient programming framework, and is now available in the free, open-source InterIMAGE system (Costa et al., 2010).

This work presents the architecture and major characteristics of the implementation of the MultiSeg segmentation program, as well as examples of the segmentation of a few different SAR images.

2. MultiSeg Segmentation Process

MultiSeg is a specialized hierarchical segmentation technique. The algorithm integrates several segmentation strategies, which iteratively process data structured as an image pyramid.

Initially the input image is resampled at different rates creating an image pyramid of various *compression levels*. Then a region growing procedure is performed at the highest compression level, i.e. the coarser level. In sequence, a split and merge-based technique is applied to the intermediate levels in turn, from the coarser to the finer level, i.e. the original image.

The region growing procedure initially considers each pixel as a segment. As segments are visited in a random order, the procedure evaluates if the current segment can be merged to its most similar neighbor, being the measure of similarity the absolute difference the segments' mean intensity pixel values. A segment is only merged to its *mutual best neighbor*, that is, the best neighbor of the current segment's best neighbor must be the current segment. There are two conditions for merging to occur: (i) if the similarity measure is below a user defined similarity threshold; and (ii) if the probability that means and variances of the candidates for merging are equal – above a user defined confidence level – according to a t-Student hypothesis test. The region growing procedure stops when no more merges are possible.

Processing starts at a new level with the refinement of the borders of each segment defined at the previous level, in order to adjust them to a higher spatial resolution. The refinement process follows the segment border visiting all intercepting pixels, and decides if each of such pixels should be kept in the segment or be assigned to a neighboring segment. The decision is based on a test that considers the distance from the current pixel value to the mean value of the neighboring segments, weighted by their variances, which in the case of SAR data takes into consideration the equivalent number of looks of the current compression level.

After the border refinement step, a homogeneity value for each segment is computed. The measure of homogeneity takes into consideration the coefficient of variation of segment's pixel values. A segment is considered heterogeneous if its homogeneity measure is below a *critical value* threshold, which is defined considering the different statistical properties of SAR and optical data. In short, homogeneity critical values for SAR data are defined based on

the assumption of a Gamma distribution of pixel values, and considering different numbers of equivalent looks, segment sizes and confidence values.

The pixels within segments that are considered heterogeneous are then subjected to the region growing procedure described above. After all heterogeneous segments are processed, the whole collection of segments in the compression level is also subjected to the region growing procedure. Again, region growing stops when no merges are possible. Processing resumes recursively at the finer compression levels until the original image level is reached. After that level is processed, all segments smaller than a user defined area threshold are merged to their most similar neighbor.

The algorithm described above for a single band image is generalized in MultiSeg for multiple bands (multispectral optical data or multiple polarized SAR data). MultiSeg's complete processing chain is depicted in Figure 1.

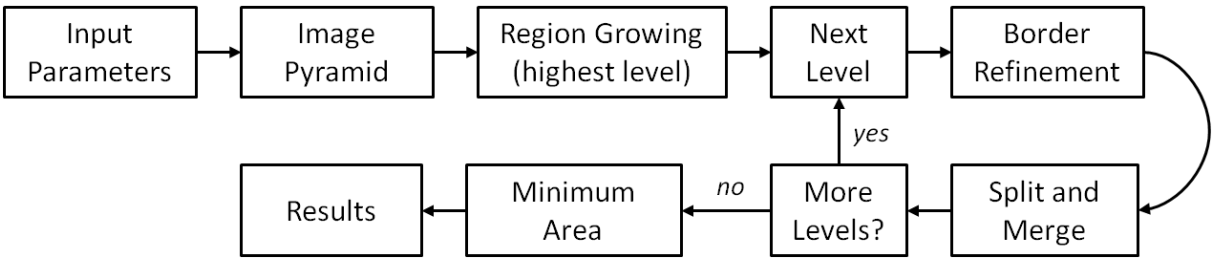
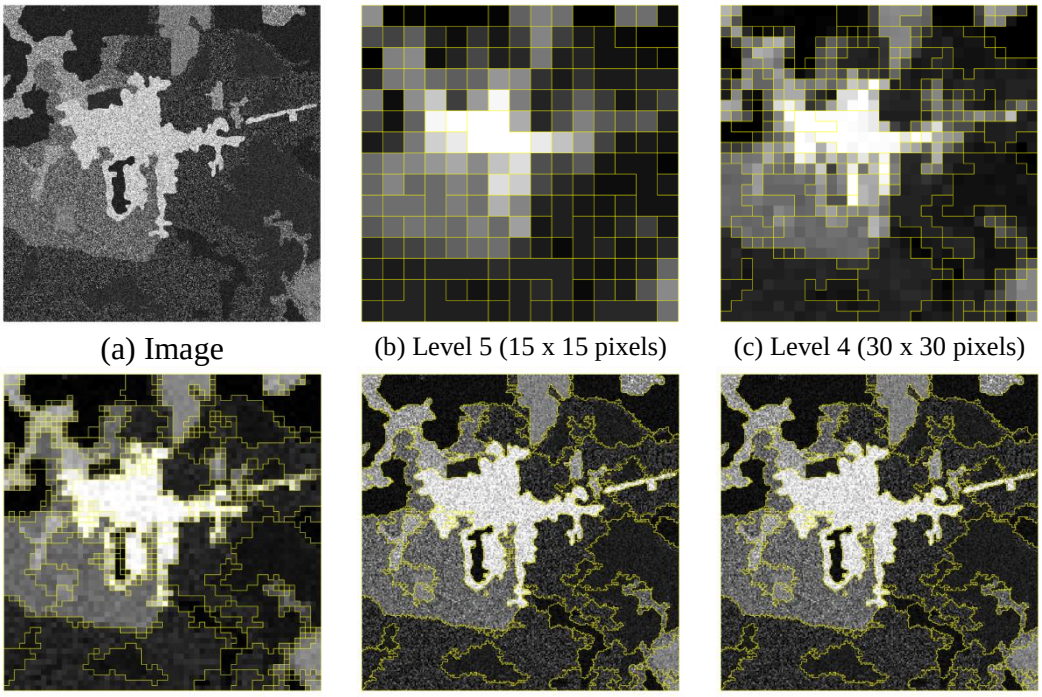


Figure 1. The MultiSeg processing chain.

Figure 2 illustrates the segmentation process at different compression levels. A synthetic radar image with 480 x 480 pixels and equivalent number of looks equal to 8 was subjected to segmentation. The limits of the final segments obtained for each level are superimposed with the respective compressed image.



(a) Image (b) Level 5 (15 x 15 pixels) (c) Level 4 (30 x 30 pixels)

(d) Level 3 (60 x 60 pixels) (e) Level 2 (120 x 120 pixels) (f) Level 1 (240 x 240 pixels)

Figure 2. (a) Synthetic radar image. (b)-(f) Final segments computed for each level, superimposed with the respective compressed image.

3. Design and Implementation

SegSAR was originally coded in IDL – Interactive Data Language (Exelis, 2013), a proprietary, interpreted programming language that offers specific routines for the creation of image processing applications, including the manipulation of matrix structures and image visualization capabilities.

An important part of this work consisted in the translation of SegSAR's code into the C++ programing language, using object oriented programming concepts, and the tuning of particular sections of the code both in terms of computational efficiency and adherence to the algorithm's underlying concepts. Code design was oriented towards a modular and extendable architecture, which can easily accommodate new mechanisms in the segmentation process, or improvements of the current ones. Additionally, there was a special concern in improving computational efficiency in terms of processing time and handling large volumes of data. The next sections introduce MultiSeg's code design, beginning with a brief description of the support of TerraLib (Câmara et al., 2008) classes and methods in the code.

3.1. TerraLib Support

In MultiSeg's code, the support for processing geospatial data was provided by TerraLib, an open-source library of classes and methods written in C++. TerraLib was developed by the Image Processing Division (DPI) of the Brazilian National Space Research Institute (INPE). TerraLib's main purpose is to provide the basic infrastructure for cooperative development in the community of creators and designers of Geoinformatics systems and applications. The library provides a set of functionalities for the manipulation of raster and vector data, together with a variety of image processing algorithms.

The coding of MultiSeg was founded on the underlying concepts of the image processing module of the library. Such module defines a single interface for algorithms and a generic mechanism for the definition of parameters – `TePDIAlgorithm` and `TePDIParameters`, respectively.

TerraLib offers encoding and decoding mechanisms for different raster formats. Raster data support is given by three basic classes (Vinhas and Sousa, 2005):

- a generic class interface for accessing the element of raster structures, called `TeRaster`;
- a data structure to represent all the parameters that characterize a raster data set: `TePasterParams`;
- an abstract class for encoding and decoding raster formats and for the access of data storage devices: `TeDecoder`.

The `TeRaster` class provides representation and access support for raster data that can be stored in primary memory, in files of different formats, or even in data base systems. For the decoupling of this class from the different storage alternatives, access requisitions for the elements of the raster representation are treated by the class `TeDecoder`. The library has a set of concrete decoders for the most common formats (e.g., GeoTIFF, TIFF, JPEG and RAW). Recently a decoder based on the GDAL library (GDAL, 2014) was incorporated in TerraLib, increasing the number of supported formats.

3.2. MultiSeg Architecture

Figure 3 depicts the major components of MultiSeg's architecture. The `Region` entity represents a particular extent (segment) in the image subjected to segmentation, it contain information about its spatial location (bounding box) and statistical properties (e.g., mean, variance) which are used in the algorithm. Additionally, each region contains a list of its adjacent regions.

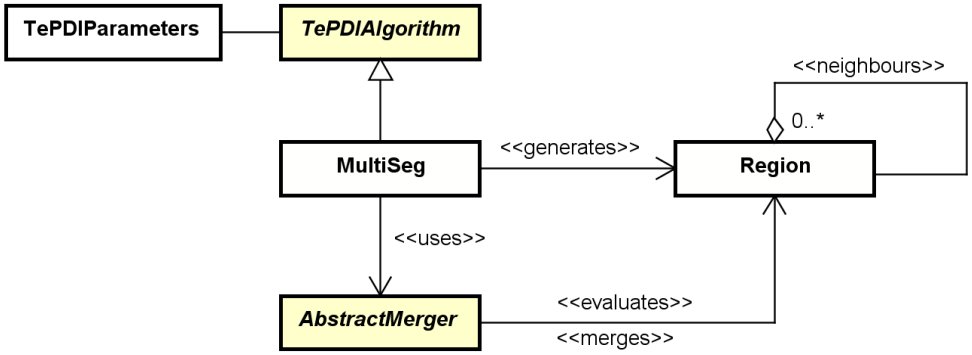
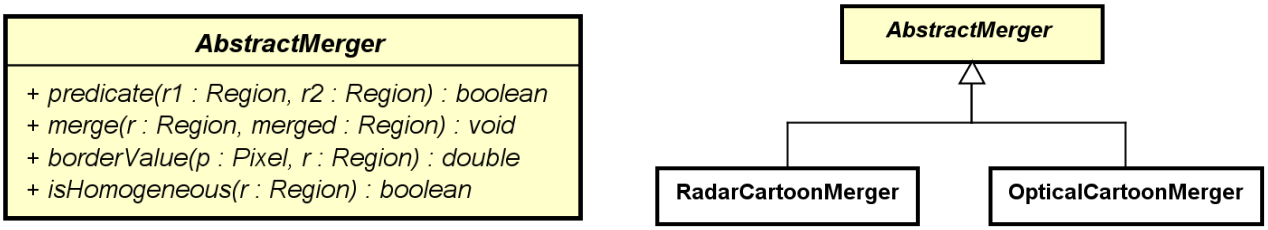


Figure 3. Major components of MultiSeg's architecture.

The `MultiSeg` class implements the different steps of the hierarchical segmentation procedure (Figure 1). It implements basically the generic steps of the underlying algorithm. For instance, the region growing procedure is independent of the image being segmented or of the statistical model defined to represent it – the strategy for selecting regions for growing and the similarity evaluation are generic steps. The statistic tests employed in the region growing process are carried out by an abstract entity called `AbstractMerger`. In this way the judgment that determines if two segments should be merged is decoupled from the MultiSeg class, creating thus an extensibility point.

The `AbstractMerger` interface defines four methods, being three of them associated to statistic tests (Figure 4.a):

1) a method for determining if two regions are similar:
   `boolean predicate(Region r1, Region r2);`
2) given a pixel and a region, a method that computes the computes the (distance) values to be considered in the border adjustment procedure:
   `double borderValue(Pixel p, Region r);`
3) a method for determining region homogeneity:
   `boolean isHomogeneous(Region r);`
4) finally, a method for updating statistical properties and region extents in the merging procedure: `void merge(Region r, Region merged).`



(a) (b)

Figure 4. `AbstractMerger`: (a) methods associated to the statistical tests; and (b) concrete implementation for optical and radar data.

In the current version of MultiSeg, two concrete classes were created: `RadarCartoonMerger` and `OpticalCartoonMerger` (Figure 4.b). These classes implement the statistical tests for the cartoon model for radar and optical images, respectively.

It is important to note the possibility of seamlessly incorporating new rules and tests for the segmentation process, without the need for modifications of the structural, generic, classes that compose the architecture of MultiSeg. Therefore, in order to extend the code in that sense, it suffices to create new classes that implement the `AbstractMerger` interface.

4. Segmentation Examples

In this section we will show a few examples of segmentation results obtained with MultiSeg over SAR data, and the effect of varying the user defined parameters, which are described below. The user should refer to the description of the segmentation procedure described in Section 2.

- *N*: number of compression levels.
- *simil*: similarity threshold (dB) for decision on region growing/merging.
- *conf*: confidence level (%) for decision on region growing/merging and homogeneity hypotheses tests.
- *nel*: number of equivalent looks of original image.
- *mas*: minimum area size (pixels) of final regions.

Although MultiSeg is able to process multiband images, in order to make it easier for the reader to analyze the segmentation results, the examples presented below show the results of the segmentation of a single image band (polarization).

4.1. Segmentation of a RADARSAT image

Figure 5 shows a subset of a RADARSAT-2 image, VH polarization, with 907 x 680 pixels, acquired in 09/19/2009, covering a rural region of Tapajós in Pará, Brazil.

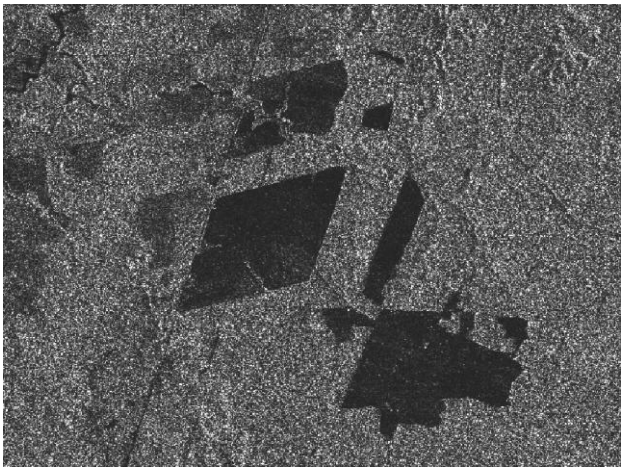
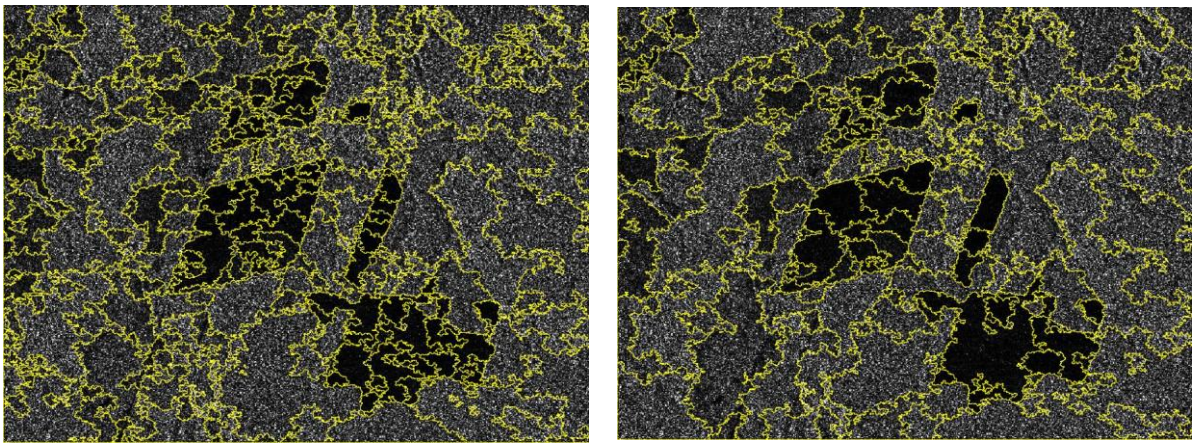


Figure 5. Original RADARSAT image (VH polarization).



(a) (b)

Figure 6. MultiSeg segmentation of the RADARSAT image: (a) *N*=6; *simil*=1.0; *conf*=0.95; *nel*=1.0; *mas*=20 and (b) *N*=6; *simil*=1.0; *conf*=0.995; *nel*=1.0; *mas*=20.

Figure 6.a shows the segments obtained with MultiSeg for the RADARSAT (VH polarization) image with the following parameter values: *N*=6; *simil*=1.0; *conf*=0.95; *nel*=1.0; *mas*=20. Figure 6.b shows the segments obtained with *N*=6; *simil*=1.0; *conf*=0.995; *nel*=1.0; *mas*=20.

From the comparison of the two segmentations one can notice the effect of altering the confidence threshold for decision on region growing/merging and homogeneity hypotheses tests (parameter *conf*). Basically, the higher the confidence threshold, the fewer are the segments generated by the algorithm, because the parameter influences how strict must be the similarity (hypothesis) between two region candidates (adjacent segments) for merging.

The segmentation shown in Figure 6.a was performed with MultiSeg in 3 seconds, while it took the old implementation of SegSAR 852 seconds (approximately 14 minutes) to perform the same task.

4.2. Segmentation of an ALOS/PALSAR image

Figure 7 shows a subset of an ALOS/PALSAR image, HH polarization, with 865 x 592 pixels, acquired in 06/21/2010, covering a rural region of Tapajós in Pará, Brazil.

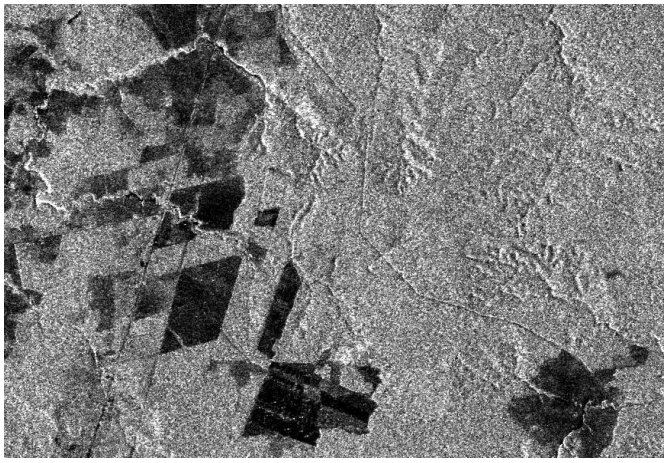
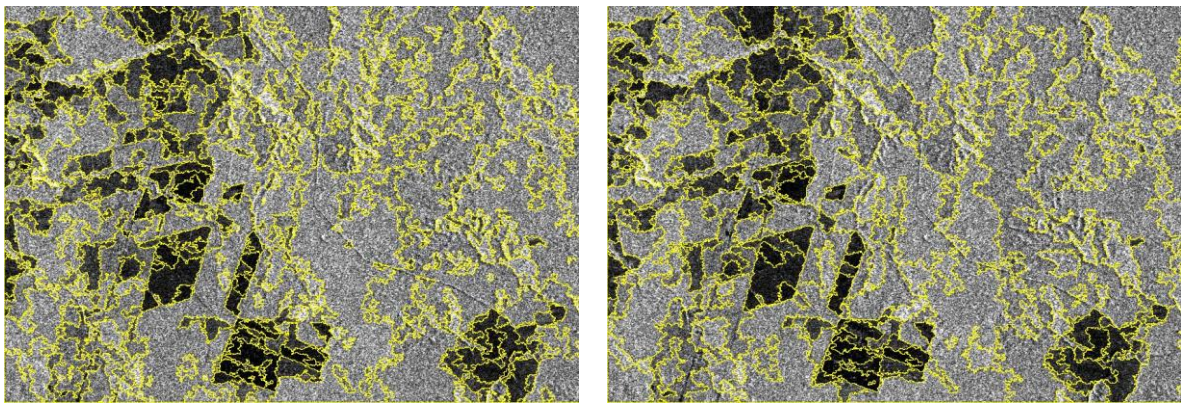


Figure 7. Original ALOS/PALSAR image (HH polarization).

Figure 8.a shows the segments obtained with MultiSeg for the ALOS/PALSAR (HH polarization) image with the following parameter values: *N*=3; *simil*=1.0; *conf*=0.95; *nel*=3.0; *mas*=20. Figure 8.b shows the segments obtained with the following parameter values: *N*=4; *simil*=1.0; *conf*=0.95; *nel*=3.0; *mas*=20.



(a) (b)

Figure 8. MultiSeg segmentation of the ALOS/PALSAR image: (a) *N*=3; *simil*=1.0; *conf*=0.95; *nel*=3.0; *mas*=20 and (b) *N*=4; *simil*=1.0; *conf*=0.95; *nel*=3.0; *mas*=20.

From the comparison of the segmentations depicted on Figures 8 one can notice the effect of altering the number of compression levels (parameter *N*). Basically, the more compression levels, the fewer are the segments generated. This is because the segmentation procedure starts on a more compressed (generalized) level, therefore fewer segments will be created in the beginning of the segmentation process, and thus propagated onward to the finer compression levels.

The segmentation shown in Figure 8.b was performed with MultiSeg in 2 seconds, while it took the old implementation of SegSAR 128 seconds (approximately 2 minutes) to perform the same task.

5. Conclusions and Future Work

This paper described briefly the underlying algorithm and the most important implementation aspects of the hierarchical segmentation program MultiSeg, and showed a few examples of the segmentation of SAR images with the program.

MultiSeg can be regarded as an improvement of a previous development called SegSAR. While SegSAR was implemented in a proprietary development environment and requires the user to acquire a particular software package, MultiSeg was created in a generic programming language (C++) with the support from an open-source library (TerraLib). MultiSeg's architecture was devised to allow easy extension of its capabilities, by adding new statistical tests and region growing/splitting rules. Additionally, MultiSeg is capable of processing large volumes of data efficiently, as preliminary tests have shown.

A number of systematic tests with MultiSeg are under way. There is still the need to determine the limitations of the program in terms of the data volume it can bare efficiently, and to investigate the proper parameter setting ranges for different image types and target object classes. The preliminary results are, however very promising.

The authors also plan to adapt the program to be able to jointly process co-registered optical and radar image data, thus fusing both types of information within the same segmentation process. The objective of this further research will be to investigate the usefulness of combining/fusioning optical and SAR data at the segmentation stage.

Acknowledgments

The authors would like to acknowledge the support from FAPERJ, CNPq (through the RHAE program and process 307666/2011-5) and the TOLOMEO Project on the development of this research.

References

Câmara, G.; Vinhas, L.; Ferreira, K.R.; Queiroz, G.R.; Souza, R.C.M.; Monteiro, A.M.V.; Carvalho, M.T.; Casanova, M.A.; Freitas, U.M. Terralib: An open source GIS library for large-scale environmental and socio-economic applications. In: HALL, G. B. (Ed.). Open Source Approaches for Spatial Data Handling. Berlin: Springer, 2008. p. 24.

Costa, G.A.O.P.; Feitosa, R.Q.; Fonseca, L.M.G.; Oliveira, D.A.B.; Ferreira, R. S.; Castejon, E. F. Knowledge-Based Interpretation of Remote Sensing Data with the InterIMAGE System: Major Characteristics and Recent Developments. In: 3rd International Conference on Geographic Object-Based Image Analysis (GEOBIA 2010), Gehnt. Enshede: ITC, 2010. v.XXXVII.

Exelis. Visual Information Solutions. 2013. Available at: <http://www.exelisvis.com/>.

GDAL. Geospatial Data Abstraction Library. 2014. Available at: <http://www.gdal.org/>.

Sousa Jr., M. A. Segmentação multi-níveis e multi-modelos para imagens de radar e ópticas. Thesis (Doctorate in Applied Computing), National Institute for Space Research, São José dos Campos, Brazil, 2007.

Vinhas, L.; Souza, R. C. M. d. Tratamento de dados matriciais na TerraLib. In: Casanova, M. A. et al. (Ed.). Bancos de dados geográficos. São José dos Campos: Mundogeo, 2005. p.426-462.