INTERSEG: A DISTRIBUTED IMAGE SEGMENTATION TOOL

P. N. Happ [a], R. S. Ferreira [b], G. A. O. P. Costa [b], R. Q. Feitosa [a, c], C. Bentes [b], R. Farias [d], P. M. Achanccaray [a] *

[a] Pontifical Catholic University of Rio de Janeiro, Brazil – (patrick, raul, pmad9589)@ele.puc-rio.br
[b] IBM Research Brazil Lab in Rio de Janeiro, Brazil - rosife@br.ibm.com
[c] Rio de Janeiro State University, Brazil – gilson.costa@ime.uerj.br, cris@eng.uerj.br
[d] Federal University of Rio de Janeiro, Brazil – rfarias@cos.ufrj.br

KEY WORDS: Remote Sensing, Image Segmentation, Object-Based Image Analysis, Distributed Processing, Cloud Computing

ABSTRACT:

Motivated by the rapidly increase of remote sensing data over the last years, this paper presents a tool designed for distributed image segmentation. InterSeg is able to handle efficiently very large high-resolution images using scalable distributed segmentation methods over computer clusters. Through a graphic user interface, InterSeg hides the distributed processing complexity, allowing the user to easily perform distributed image segmentation on a cloud-computing environment. Experiments carried out with the tool attest its potential scalability and its capability to exploit distributed processing over many cluster nodes. Moreover, as an open-source solution, the software can be extended with different segmentation algorithms and methods.

1. INTRODUCTION

Modern advances in Earth Observation technologies are responsible for improvements in spatial, spectral and temporal resolution of remote sensing data. Such improvements, associated with the increasing number of aerial and orbital systems in activity, result in a huge growth in the volume of available remote sensing data. Furthermore, the cost of images provided by private companies has dropped considerably in recent years, and open-access datasets have been widespread, facilitating the access to this kind of information.

Timely analysis of such large datasets represents a great challenge for researchers in a number of application areas, such as agriculture, disaster response, urban planning, and mining operations, among others. These areas usually require automatic methods and tools to interpret remote sensing imagery, allowing the extraction of strategic information to be used by decision makers.

Currently, Object-Based Image Analysis (OBIA) is widely used in the environmental science community (Blaschke et al., 2014), and image segmentation is at the core of OBIA. The procedure is responsible for partitioning an input image in homogeneous pixel agglomerates that will be later classified.

Image segmentation is not a new subject and a wide variety of algorithms have been proposed in the past decades (Haralick and Shapiro, 1985; Pal and Pal, 1993; Dey et al., 2010). In addition, a number of automatic tools for image segmentation are now available. However, dealing efficiently with massive data volumes represents a major obstacle for the current segmentation solutions, which are generally not able to handle very large images.

Distributed computing is a common alternative for the processing of very large datasets, and in this context, cloud computing is a trend, as it can deliver a scalable infrastructure to support different processing needs (Fernández et al., 2014). Moreover, there are currently many cloud computing infrastructure providers that offer a pay-per-use model and deliver great computing power to users that do not need to be concerned with acquiring or maintaining complex hardware.

In this work, we use the distributed computing concept in order to provide a scalable and efficient solution to overcome current limitations for processing very large datasets. We introduce InterSeg, an open-source distributed processing tool designed to run image segmentation on physical or virtual computer clusters. InterSeg is an easy to use, scalable and efficient solution to handle very large remote sensing images.

The remainder of this paper is organized as follows. Some related work on image segmentation is presented in Section 2. An overview of InterSeg is described in Section 3. An example of usage of InterSeg is shown as an experiment in Section 4. Section 5 concludes this working and offers some directions for future work.

2. RELATED WORK

Image segmentation aims at identifying regions that represent objects of interest (Russ, 1998). This is an important step in image analysis, since it is responsible for delineating the regions that will be later classified by experts or by automatic classification procedures. Thus, the segmentation quality has a direct influence on the result of the whole image analysis process.

In the past few years, considerable efforts have been made to ease the computational burden of image segmentation of high-

* Corresponding author

resolution remote sensing imagery. These efforts usually rely on parallel processing solutions that explore symmetric multiprocessors (Moga et al., 1998), multicore processors (Happ et al., 2010) or even graphic processing units (Backer et al., 2013). In spite of achieving good accelerations, these algorithms still present problems to handle the growing scale of the remote sensing imagery.

Regarding distributed solutions, Tesfamariam (2011) proposes a different approach to edge detection models on a local cluster. Surve and Paddune (2014) and Jin et al. (2014) also present distributed versions based on clustering techniques being the latter focused on cloud computing. Pixel-based analysis techniques are easier to adapt to distributed processing environments, since they represent almost independent tasks at the pixel level. Region growing segmentation, however, has to deal with strong dependencies among adjacent segments, and requires a more complex distribution strategy. In general, the algorithms have to split the image into tiles and process them independently. The key problem is how to handle segments on the borders of the tiles, as there is a high level of dependency among neighboring segments.

InterSeg is based on a generic distributed segmentation strategy that can generate (image tile bordering) segments without artifacts. This strategy is based on image tile division and on a hierarchical stitching of the segments on tile borders (Happ et al., 2015). Additional segmentation algorithms can also be included in the tool as long as they comply with some rules.

3. INTERSEG OVERVIEW

InterSeg is an open-source image segmentation tool, which provides distributed processing on computer clusters or on cloud computing infrastructures. It can also be considered as an independent module of InterCloud, a distributed image interpretation platform for handling large remote sensing datasets (Ferreira et al., 2014).

3.1 InterSeg Architecture

Similar to InterCloud, the architecture of InterSeg consists of three layers containing a different representation of the segmentation tool in different levels of abstraction (Figure 1). The first layer works as a user interface. It focuses on the end-user and is related to definition of the segmentation inputs and configurations. For this reason, it is known as the project definition layer.

The second layer is associated to the algorithms and methods developed for the tool. It is called segmentation layer and it is organized on a high-level structure in order to hide the distributed programming complexity. A conventional programmer can interact with this layer in order to include new segmentation algorithms and other desired functions. The connection between project and segmentation layers is defined by a translation of the first into processing and data flow instructions present in the latter.

The third layer is the distribution layer. Since it is related to the distributed processing of the image segmentation, only programmers with distributed programming skills are able to interact with this layer. The connection between the segmentation and the distribution layers is defined again by a translation. The high-level instructions are compiled into distributed processing code.

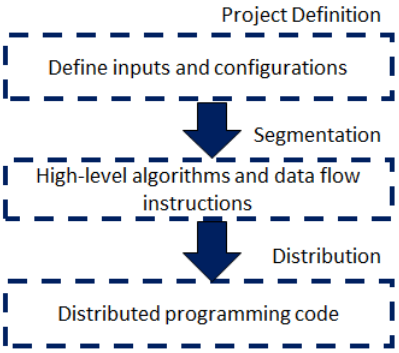

Figure 1. InterSeg's architecture.

There are many alternatives to implement the architecture of InterSeg. In the following, we will present our current implementation.

3.2 InterSeg Implementation

The distributed implementation is based on MapReduce (Dean and Ghemawa, 2008); a programming model widely used to process massive data volumes in the cloud (Assunção et al., 2014). More specifically, we use Apache Hadoop, which is MapReduce's most used implementation and offers an open-source, highly scalable and reliable framework for storing and processing large datasets.

In order to manage MapReduce, we use the Pig framework (Gates, 2011). Pig is composed by a high-level language for expressing data-flows (Pig Latin), and by an engine that compiles it into MapReduce jobs. Pig Latin provides an easier way to work with the distributed MapReduce model. It also offers an extension capability, of external libraries and methods, through its User Defined Functions (UDFs).

The image segmentation algorithms are coded in Java and are structured as Pig UDFs in order to be invoked by Pig Latin scripts. Additional algorithms can be embedded following the same structure. A parser translates the project definitions into such scripts, and the definitions are set by the user through a graphical user interface (GUI).

Lastly, the cloud computing services are accessed through an application programming interface (API) provided by the cloud computing infrastructure providers.

3.3 InterSeg as a tool

Considering InterSeg as an end-user tool, the focus relies on how to operate it in order to perform a segmentation task. InterSeg and its code are available at http://www.lvc.ele.puc-rio.br/wp/?cat=41. It is important to note that the tool is still under development and this is a Beta release, therefore some bugs are expected.

InterSeg has four tabs to define the segmentation project and run it on the cloud. The first is the database description: the user must define a local path or a cloud link for the raster image (Figure 2). If it is possible, InterSeg will load the geographic coordinates and additional information. The user is able to modify them and to set the Coordinate Reference System and the desired tile size.

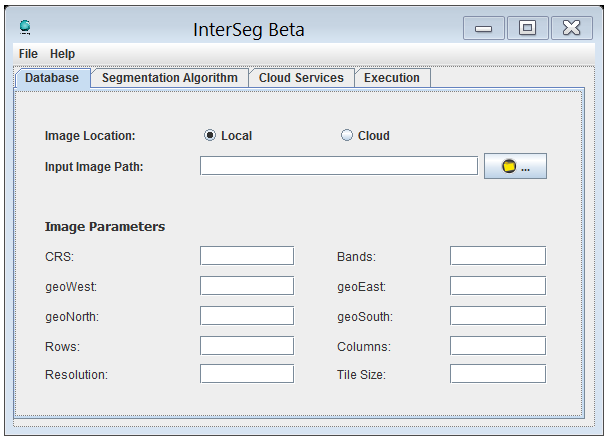

Figure 2. Defining the database

The second tab is related to the segmentation algorithm (Figure 3). The user should select one among the available implementations. At the moment, there are distributed versions of a chessboard, a multiresolution region growing and a thresholding segmentation. The segmentation parameters are automatically loaded after an algorithm is chosen and must be set by the user.

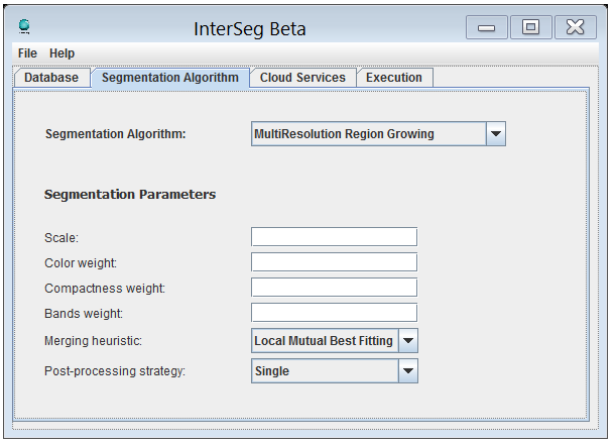

Figure 3. Selecting the segmentation algorithm

The third tab is based on the cloud computing settings (Figure 4). First, the user should choose one from the available cloud infrastructure providers. Then, according to the user selection, the required parameters are automatically loaded to be filled by the user.

The last tab concerns the segmentation execution. The user can select the type of the output such a JSON file or a shapefile. It is also possible to choose the output path and to select a location to download the result. After the whole configuration is done, the *start segmentation* button initiates the execution.

At the beginning of the process, image tile division is carried out. If the image is in a local path, each generated tile and its respective auxiliary files are uploaded to the defined file storage repository. The Pig script containing the distributed segmentation data-flow is also created and placed in a similar location. Then, the virtual cluster is instantiated at the desired cloud computing service. Next, the distributed segmentation starts and, when it is completed; the outcome is downloaded in the proper format to the desired local path.

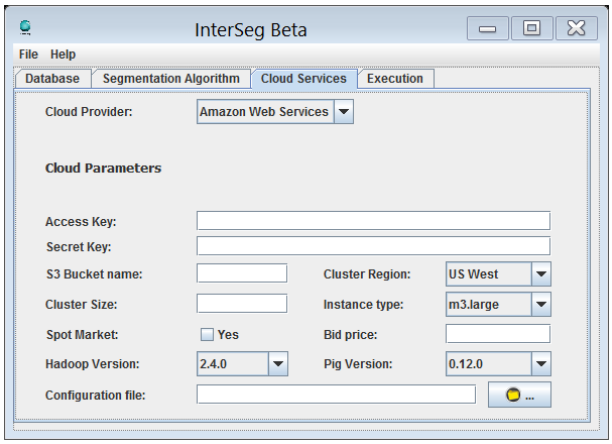

Figure 4. Configuring the cloud computing services

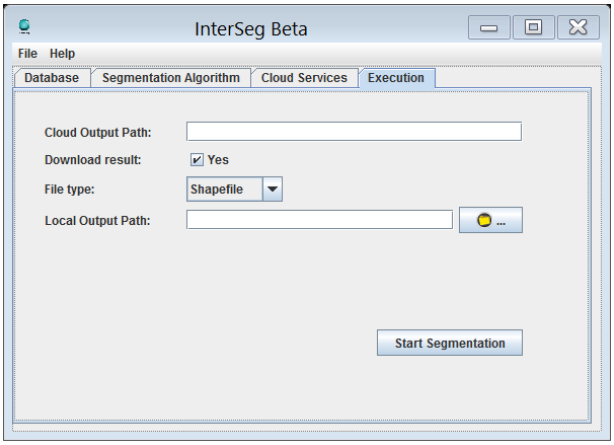

Figure 5. Running the segmentation

4. EXPERIMENTS AND RESULTS

To validate InterSeg, we performed an experiment using the tool to segment a Quickbird image on the cloud. The image is a 4000×4000 pixels subset, called hereafter 4K image (Figure 6), covering some districts of São Paulo city, Brazil. It has four spectral bands (blue, green, red and infrared) and a spatial resolution of 0.61 meters.

The image was replicated in order to assess the performance for large images. We generated images of 8000×8000 pixels (8K) 16000×16000 pixels (16K) and 32000×32000 pixels (32K). The tile size used was 1024x1024 pixels.

The selected image segmentation algorithm was the distributed version of the multiresolution region growing algorithm proposed in (Baatz and Shäpe, 2001). The parameters were kept the same for every run: *scale = 40; color weight = 0.84; compactness = 0.8; bands weights = 1, 1, 1, 1; merging heuristic = local mutual best fitting; post-processing strategy = single post-processing.*

We selected the Amazon Web Services (AWS) as the cloud-computing provider. AWS provides the Simple Storage Service (S3) as a file repository and the Amazon Elastic MapReduce (EMR) to build a Hadoop cluster using Amazon Elastic Compute Cloud (EC2) instances. We varied the cluster sizes (2, 4, 8, 16 and 32 nodes) in order to show the potential scalability of the tool. For every run, instances of type *m3.xlarge* were used under the *spot market*. These are machines containing Intel

Xeon E-5-2670 v2 processors operating at 2.5GHz with a 64-bit architecture, 15 GB of RAM and 2 disks with 40GB using SSD technology. The Hadoop version was set to 2.4.0 and the Pig version was 0.12.0.

Figure 6. QuickBird image subset called 4K.

All the images were successfully segmented on the cloud-computing environment. Figure 7 presents the execution times for the 4K, 8K, 16K and 32K images over 2, 4, 8, 16 and 32 cluster nodes. It is possible to observe that the execution time decreases, as more nodes are included. This reduction is more significant for larger images, since it can benefit more from the higher number of processing elements available.

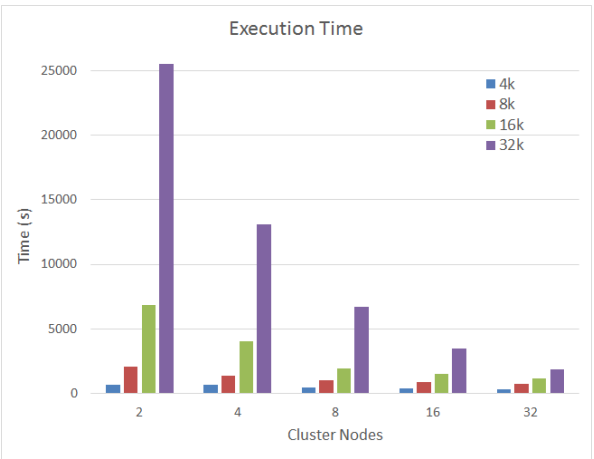

Figure 7. Execution time for every segmentation run

Figure 8 shows the same results, but exposed as speedups relative to the execution over 2 cluster nodes. The plot demonstrates that the speedup tends to increase with the image size. For the largest image, we obtained speedups higher than 13 when using 32 nodes. Small images are not able to exploit the additional processing power, and achieve smaller speedups.

It is worth mentioning that different amounts of cluster nodes produce exactly the same segmentation outcome. Figure 9 shows part of the outcome for 4k image. The result shows that overall segmentation presents good results with few artifacts. The artifacts are still present due to the *simple* post-processing strategy used as described in (Happ et al., 2015) Different strategies can produce results without artifacts.

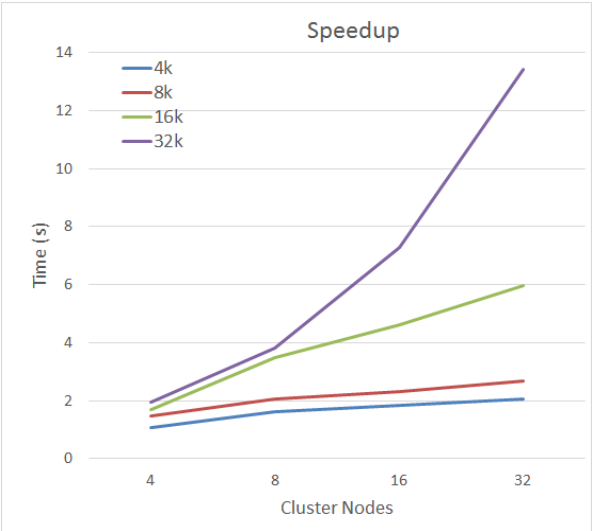

Figure 8. Speedups related to the 2 cluster nodes execution.

Figure 9. Part of segmentation outcome.

5. CONCLUSION

In this work, InterSeg was presented, which is a free tool designed to perform distributed image segmentation over computer clusters. The first Beta release of the software is available at http://www.lvc.ele.puc-rio.br/wp/?cat=41.

Through a graphical user interface, a user is able to choose a segmentation algorithm and process a very large image using a scalable cloud-computing infrastructure. The number of available algorithms, as well as the cloud infrastructure providers, are still limited; however, it is possible to extend the software including new alternatives.

The experiments conducted in this work demonstrated the potential scalability of the software since using an increasing amount of cluster nodes led to higher speedups when the images were large enough.

ACKNOWLEDGEMENTS

The authors acknowledge the funding provided by FAPERJ (Carlos Chagas Filho Foundation for Research Support in Rio de Janeiro), CNPq (National Council for Scientific and Technological Development) and CAPES (Coordination for the Improvement of Higher Education).

REFERENCES

Baatz, M., Schäpe, A., Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation, In: *XII Angewandte Geographische Informationsverarbeitung*, Wichmann-Verlag, Heidelberg, 2000.

Backer, M.; Tünnermann, J.; Mertsching, B. Parallel K-Means Image Segmentation Using Sort, Scan and Connected Components on a GPU. In: *Facing the Multicore-Challenge III*, 108–120. Berlin: Springer, 2013.

Blaschke, T., Hay, G. J., Kelly, M., Lang, S., Hofmann, P., Addink, E., Feitosa, R. Q., Der Meer, F., Der Werff, H., Coillie, F., Tiede, D., Geographic object-based image analysis – towards a new paradigm, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 180-191, 2014.

Dean, J., Ghemawa, S., MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, v. 51, n. 1, p. 107-113, 2008.

Dey, V., Zhang, V., Zhong, M., A review on image segmentation techniqueswith remote sensing perspective, In: *Proc. ISPRS*, pp. 5-7, 2010.

Fernández, A., del Río, S., López, V., Bawakid, A., del Jesus, M. J., Benítez, J. M., Herrera, F., Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 5, pp. 380-409, September/October 2014

Ferreira, R. S., Oliveira, D. A. B., Happ, P. N., Costa, G. A. O. P., Feitosa, R.Q., Bentes, C., InterIMAGE 2: The Architecture of an Open Source, High Performance Framework for Automatic, Knowledge-Based Image Interpretation, In: *International Geographic Object-Based Image Analysis Conference*, Thessaloniki, 2014.

Gates, A., *Programming Pig*, O'Reilly, 2011.

Happ, P. N., Ferreira, R. S., Bentes, C., Costa, G. A. O. P., Feitosa, R. Q., Multiresolution Segmentation: a Parallel Approach for High Resolution Image Segmentation in Multicore Architectures, In: *3rd International Conference on Geographic Object-Based Image Analysis*, 2010, Ghent, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Enshede: ITC, 2010. v.XXXVII

Happ, P. N., Ferreira, R. S., Costa, G. A. O. P., Feitosa, R. Q., Bentes, C., Gamba, P., Towards distributed region growing image segmentation based on MapReduce, In: *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Milan, 2015, pp. 4352-4355.

Haralick, R. M., Shapiro, L. G., Image segmentation techniques. *Computer Vision Graphics Image Process*, 29:100–132, 1985.

Jin, X.; Sieber, R.; Kalacska, M. The challenges of image segmentation in big remotely sensed imagery data. *Annals of GIS 20.4* (2014): 233-244, 2014.

M. D. Assunção, Calheiros, R. N., Bianchi, S., Netto, M. A. S., Buyya, R., Big Data computing and clouds: Trends and future directions, *Journal of Parallel and Distributed Computing*, Aug. 2014.

Moga, A.; Cramariuc, B.; Gabbouj, M., Parallel watershed transformation algorithms for image segmentation, *Parallel Computing*, 1998.

Pal, N. R., Pal, S. K., A review of image segmentation techniques, *Pattern Recognition*, 26(9):1277-94, 1993.

Russ, J. C. *The image processing handbook* - 3rd ed. Materials Science and Engineering Department North Carolina State University Raleigh - North Carolina, 1998.

Surve, A. R.; Paddube, N. S., A Survey on Hadoop Assisted K-Means Clustering of Hefty Volume Images. *International Journal on Computer Science and Engineering* 6.3 (2014): 113, 2014.

Tesfamariam, E. B. "Distributed processing of large remote sensing images using MapReduce - A case of Edge Detection," 2011.