Remote Sensing Image Processing Functions in Lua Language

Article · December 2017

DOI: 10.6062/jcis.2017.08.03.0133

7 authors, including:

Rennan De Freitas Bezerra Marujo
National Institute for Space Research, Brazil
19 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Leila Maria Garcia Fonseca
National Institute for Space Research, Brazil
124 PUBLICATIONS **622** CITATIONS
SEE PROFILE

Thales Sehn Körting
National Institute for Space Research, Brazil
124 PUBLICATIONS **304** CITATIONS
SEE PROFILE

Hugo Bendini
National Institute for Space Research, Brazil
28 PUBLICATIONS **51** CITATIONS
SEE PROFILE

J. Comp. Int. Sci. (2017) 8(3):163-172
http://epacis.net/jcis/PDF_JCIS/JCIS-0133.pdf
jcis@epacis.net
©2017 PACIS (http://epacis.net)

Pan-american Associacion of Computational Interdisciplinary Sciences

Remote Sensing Image Processing Functions in Lua Language

Rennan de Freitas Bezerra Marujo[a1], Leila Maria Garcia Fonseca [a], Thales Sehn Körting[a], Hugo do Nascimento Bendini[a], Gilberto Ribeiro de Queiroz[a], Lubia Vinhas and Karine Reis Ferreira[a]
[a]National Institute for Space Research, São José dos Campos, SP, Brazil

Received on Nov 10, 2016 / accepted on June 08, 2017

Keywords: Applied Computing in Space and Environmental Sciences, Open Source, Remote Sensing, Lua.

Abstract

Geographic Information Systems users without appropriate programming skills perform repetitive tasks through Graphical User Interfaces when manipulating large datasets, such as remote sensing imagery. Scripting languages, such as Lua, make the process easier for those users, due to their higher abstraction and simplicity, than the more complex programming languages such as C or C++. Based on that, this paper describes a high-level, open source programming environment for remote sensing image processing. We present a Lua API (Application Programming Interface) that provides image processing functions of the geospatial library TerraLib. This API allows users to easily and efficiently create new algorithm prototype for remote sensing images using the high-level programming language Lua. To demonstrate this API, we show an application to fill gaps in Landsat-7 images using a multiscale segmentation approach.

1. Introduction

Geographic Information Systems (GIS) users perform various image processing techniques and many times have the need to prototype their own processing. Many professionals from this area are not familiar with programming languages, for example C++, which presents many implementation details, such as memory management, that makes its learning complex. Based on that, most of GIS and digital image processing software offer an environment where the users can create their own procedures using simpler languages. For instance, the commercial software ENvironment for Visualizing Images (ENVI) [1] incorporates a structured language called Interactive Data Language (IDL) [2] and the open source GIS SPRING provides map algebra through a spatial language called LEGAL [3].

Normally the languages used in image processing software are scripting languages. Those languages are considered to be high level because programmers can write simpler codes, since many details, as memory management, are hide and handled automatically. The scripting languages follow the idea that they are not intended for writing applications from scratch but rather from combining existing components [4]. Lua is an efficiently interpreted scripting language that has a procedural syntax, a dynamic typing system and can easily be embedded. It supports several programming language paradigms: imperative, object-oriented or functional [7].

A key feature in a script language design is its flexibility to integrate with system languages. Considering that many useful algorithms are written in C++ and that users need a high abstraction environment, more complex programming languages can be accessed by a more abstract language through a binding. In the

[1]E-mail Corresponding Author: rennan.marujo@inpe.br

programming context, a binding is a wrapper library that allows inter-language communication. That way, source code written in a language, such as C or C++, can implicitly be used by another language without users even knowing it [5].

In this context, TerraLib is an open source geospatial library, created in C++, that extends object-relational database management system (DBMS) to support spatial analysis, spatial data mining, spatiotemporal models and remote sensing image processing [8]. The TerraLib's raster processing module includes several image processing algorithms, including registration, mosaicking, fusion, filtering, contrasting, unmixing modeling, arithmetic operations, segmentation, feature extraction, classification among others. A binding of TerraLib in Lua allows users to use the TerraLib functionalities in Lua scripts.

This paper presents a Lua API for remote sensing image processing. The API provides the image processing functions of the TerraLib raster processing library through the high-level programming language Lua. It allows users to work in a high level of abstraction, without worrying about C++ implementation details. We demonstrate the potential of the proposed API in an application to filling gaps in Landsat-7/ETM+ images through a multiscale segmentation approach [9], showing the contribution to open source environments for remote sensing image processing.

1.1. Related Work

Language binding can be applied to a variety of languages and is useful to migrate codes from one to another [10, 11]. This procedure enables high performance code to be automatically wrapped in a desired higher level language, so that end users don't need to know implementation details of lower level components[12].

The use of bindings provide users with a coding environment aimed at spatial domain experts rather than programming experts. It also increases flexibility since users can combine functionalities represented in a workflow to solve a specific problem. In the spatial software context, this strategy can be seen in several GIS applications and libraries.

Geospatial Data Abstraction Library (GDAL) and OGR are open source libraries used for image and vector geospatial data, providing read and write operations to a variety of formats through C++ libraries. These can be accessed by bindings in Python, Pearl, C# and Java [13].

The Geographic Resources Analysis Support System (GRASS) is an open source GIS with many 3D methods, that offers data management, image processing, spatial modeling, and visualization [14]. GRASS implementations can be accessed using bindings in Python [15] and R [16, 17]. QuantumGIS (QGIS) also provides Python bindings through GRASS GIS functionalities, and R scripts [18].

Since TerraLib offers a powerful set of image processing functions in C++ language [8], it is necessary to provide such functions in a high-level programming environment for users that do not have low-level programming skills.

2. Filling Landsat-7 SLC-off gaps

As an example of image processing application, Lua binding approach will be applied to process a Landsat image. Landsat program provides the most expansive range of orbital image series, providing observations since 1972 [19]. As the Earth is constantly changing, monitoring its surface is vital for the comprehension and characterization of changing processes. Landsat-7, one of the most used satellites, began to present hardware failures on the Enhanced Thematic Plus (ETM+) sensor Scan Line Corrector (SLC) in May 2003, resulting in image gaps of missing data. The images collected before the failure are called SLC-on images, and after the failure are referred as SLC-off. The gaps corresponds to a single pixel near the center of a path/row image and can reach 14 pixels width near the borders of the image [20], as illustrated on Figure 1.

Considering Tobler's first law that near objects are more related to than distant objects [21], interpolating Landsat-7 SLC-off missing data would fill the gaps with neighboring pixel values. This procedure is acceptable for small gaps, however its results may not be valid for larger gaps. A better attempt to fill those gaps consist in using approximate date images from an SLC-on image and use context information to fill the missing values [22]. That approach is based on three principles:

- adjacent pixels are more likely to be similar;

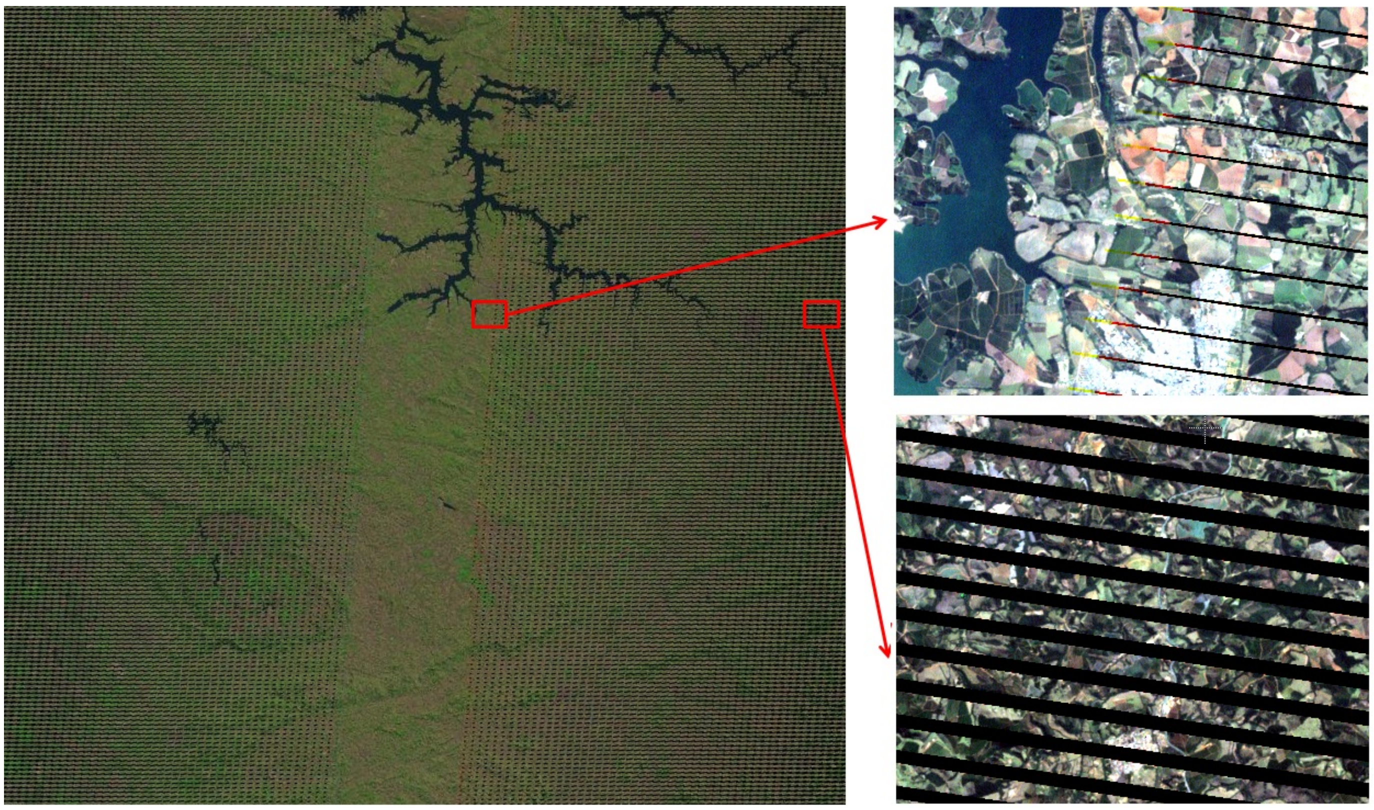

Figure 1: A Landsat-7/ETM+ SLC-off image illustrating that near the center of the image the gaps correspond to a single pixel in width, reaching 14 pixels in width near image borders.

- groups of the same landscape unit are likely to have similar spectral values;
- the vast majority of landscapes remain constant for periods.

Basically the Landsat-7 SLC-off missing data is filled using average values of nearby pixels, delimited by homogeneous regions captured by other image of an approximated date, like a Landsat-5/TM or Landsat-8/OLI image segmentation. As illustrated at Figure 2.

This approach was improved by using multiples segmentation levels, filling the gaps with the segment mean. In this approach, if a segment is fully contained in the missing data, a greater segment area level is used to fill the gap [9]. Algorithm 1 shows the approach using three consecutive segmentation levels, and a nearest neighbour approach in case the third level isn't big enough.

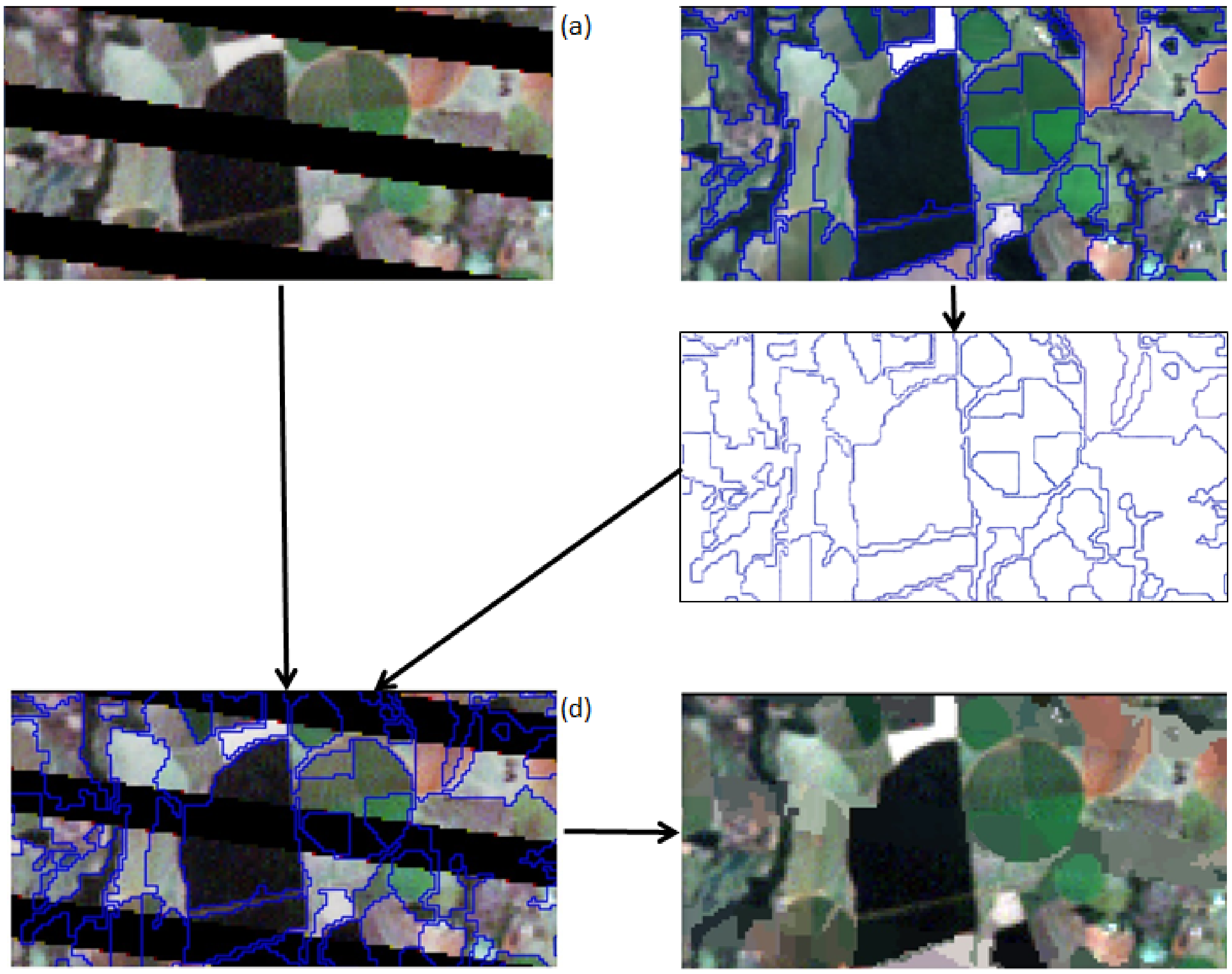

Figure 2: Maxwell (2004) segmentation approach to fill a Landsat-7 SLC-off image (a) by using an image from another source, such as Landsat-5 or Landsat-8 acquired within a close date range (b) and segmenting it (c). These segments are than applied in the Landsat-7 image (d) and the gaps are filled using the segments mean value (e).

Algorithm 1: MULTISCALE GAP FILLING METHOD PSEUDO-CODE [9]

```
   Input: Landsat7Image, SegmentationMask1, SegmentationMask2, SegmentationMask3
   Output: Filled Landsat-7 image
 1 foreach segment [i] of SegmentationMask1 do
 2     if there is a null value within segment [i] then
 3         if SegmentationMask1[i] mean $\neq$ 0 then
 4             use SegmentationMask1[i] mean to fill Landsat7Image null values
 5         else
 6             if imgSegMaskLvl2 segment mean $\neq$ 0 then
 7                 use SegmentationMask2[i] mean to fill Landsat7Image null values
 8             else
 9                 if SegmentationMask3[i] mean $\neq$ 0 then
10                     use SegmentationMask3[i] mean to fill Landsat7Image null
11                 else
12                     fill Landsat7Image null values with nearest neighbor
```

3. Methodology

In order to provide final users with a Lua environment, a Lua binding was constructed using the Simplified Wrapper and Interface Generator (SWIG) [23]. SWIG is an open source compiler interface which connects programs written in C and C++ with other languages, allowing native functions to be called by other programming languages without modifying it into low-level. Using SWIG, a connection was created from C code to Lua, by firstly defining a special function called *wrapper*. This wrapper interconnected the two languages, converting function arguments from Lua to C and returning usable code to the scripting language. Since the only data structure mechanism in Lua is table [7], conversion functions were implemented in SWIG's interface to convert structures, as arrays, in Lua tables. Functions which returned pointer references were treated by returning only the pointed value. In more details, when a C++ method would return a pointer, its content was stored in a temporary variable at the SWIG interface, than this variable was returned to Lua. Nested classes were treated using SWIG's interface to calls directly the desirable function. End users does not need to re-implement this binding, having the liberty to freely use the Lua syntax.

To adapt the Landsat-7 SLC-off filling gap methodology for recent dates, a Landsat-8/OLI image was used instead of Landsat-5/TM. A cloud-free subset, granted by Cfmask algorithm [24, 25] was made totalizing an area of 600x500 pixels in a Landsat-7/ETM+ SLC-off and a Landsat-8/OLI images (WRS 2 Worldwide Reference System 2, Path/Row 220/75). Landsat-7/ETM+ and Landsat-8/OLI images were acquired on 22 July and 30 July, respectively. Both images have 30 meters of spatial resolution and processed to Level 1 Terrain Corrected (L1T) and corrected for atmospheric conditions by the USGS EROS Science Processing Architecture (ESPA) [26].

As illustrated in Figure 3, the processes used to fill the gaps in Landsat-7 SLC-off implemented using TerraLib's Lua binding was composed by the Normalized Difference Vegetation Index (NDVI) [27] of two images, the Landsat-7/ETM+ SLC-off and the reference image from Landsat-8/OLI. After that, the Landsat-8/OLI NDVI image was segmented using multiresolution segmentation [28]. Finally, the Landsat-7/ETM+ SLC-off gaps were filled using the region information obtained by the segmentation of the Landsat-8/OLI NDVI [9].

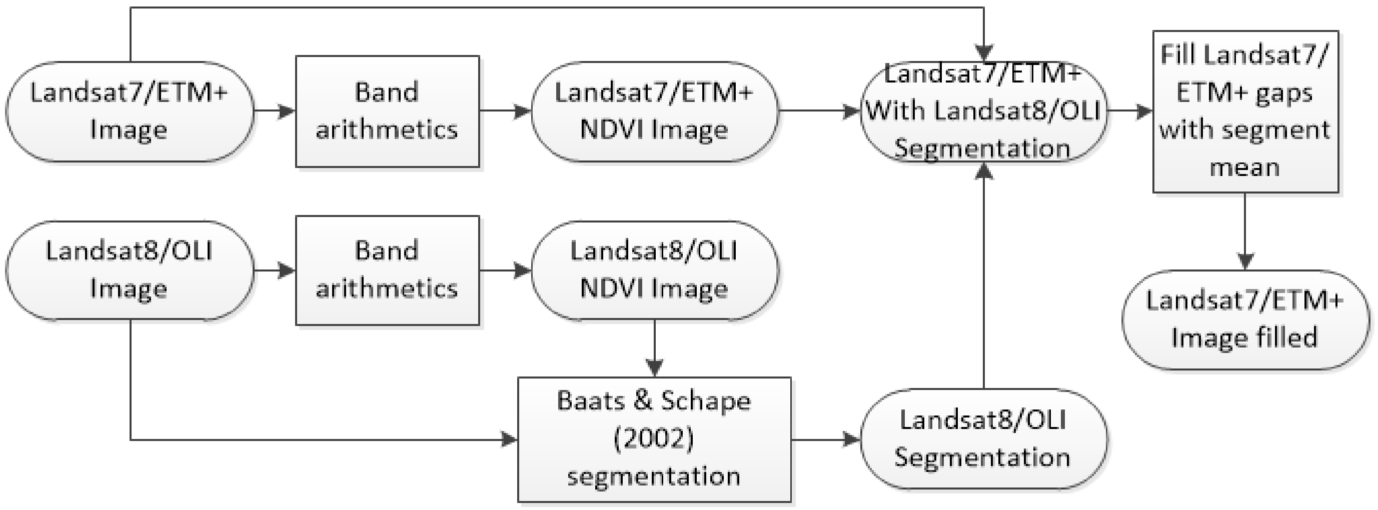
Landsat7/ETM+ Image
Band arithmetics
Landsat7/ETM+ NDVI Image
Landsat7/ETM+ With Landsat8/OLI Segmentation
Fill Landsat7/ ETM+ gaps with segment mean
Landsat8/OLI Image
Band arithmetics
Landsat8/OLI NDVI Image
Landsat7/ETM+ Image filled
Baats & Schape (2002) segmentation
Landsat8/OLI Segmentation

Figure 3: Processes diagram to fill Landsat7/ETM+ gaps using Landsat-8/OLI images.

4. Results

After the interface implementations, users could access the TerraLib's raster module libraries through Lua language. Figure 4 exemplifies a code that shows how to perform band arithmetics in order to calculate the NDVI. In line 1 we define an *ArithmeticOperation* object; line 2 the vegetation index formula is given as a string, in which we create the representation for the specific bands needed to compute the NDVI (*imgRed* and *imgNir*). Given a set of raster data, represented by R in the string, the reference *R1:0* will represent the first band of *imgNir*, while *R0:0* will point to the first band of *imgRed*. Line 3 sets the directory output where the resulting image will be saved. Figure 5 exemplifies another code with a different process, in which multiresolution segmentation [28] algorithm is performed. In line 1, a class is instantiated and in line 2 its parameters configurations are set.

```
local aop = te.rp.ArithmeticOperations()
aop:initialize("(R1:0-R0:0)/(R1:0+R0:0)", imgRed, imgNir)
aop:execute( outputDirectory, "GDAL")
```

Figure 4: Using the Terralib's Lua binding to process NDVI through raster processing method Arithmetic-Operations.

```
local seg = te.rp.Segmenter()
seg:baatzSeg(rasterImage, similarityThreshold, colorWeight,
↪  compactnessWeight, outputDirectory)
```

Figure 5: Using the Terralib's implementation of multiresolution segmentation [28] through Lua binding to segment an image given a set of input parameters, such as algorithm specifi parameters (similarityThreshold, colorWeight, compactnessWeight) and I/O parameters (rasterImage, outputDirectory).

In order to fill the Landsat-7 SLC-off gaps, a script to process the images was created. The code in Figure 6 uses TerraLib's functions to calculate the NDVI and segment it. In line 1 TerraLib's libraries are loaded; in lines 3 and 4 two Landsat-8/OLI images are loaded; both images are used to calculate the NDVI in lines 6 to 8; in line 9 the result image is opened and in lines 11 to 14 it is segmented with multiresolution segmentation algorithm [28]. For the segmentation, the following parameters were adopted: 0.3 for similarity threshold, 0.9 for color weight, 0.5 for compactness weight, while for each segmentation level the similarity threshold were defined into 0.15, 0.2 and 0.3. These parameters were defined empirically, since the human eye is a strong and experienced evaluator of segmentation techniques [29].

```
local seg = te.rp.Segmenter()
seg:baatzSeg(rasterImage, similarityThreshold, colorWeight,
↪  compactnessWeight, outputDirectory)
```

Figure 6: Full script to open an image, calculate NDVI and segment it using multiresolution segmentation [28] through TerraLib's Lua binding.

The segmentation results of the Landsat-8/OLI image can be seen in Figure 7, where *a* is the original NDVI image, while *b*, *c* and *d* are the three different segmentation levels.

Code 6 provided all the requisites to perform the Landsat-7 SLC-off multiscale gap filling method. The original Landsat-7 SLC-off image was overlapped with the Landsat-8/OLI segmentation levels and the the multiscale gap filling method was applied, being a single band result shown in Figure 8. It can be verified that the gaps were successfully filled with homogeneous areas.

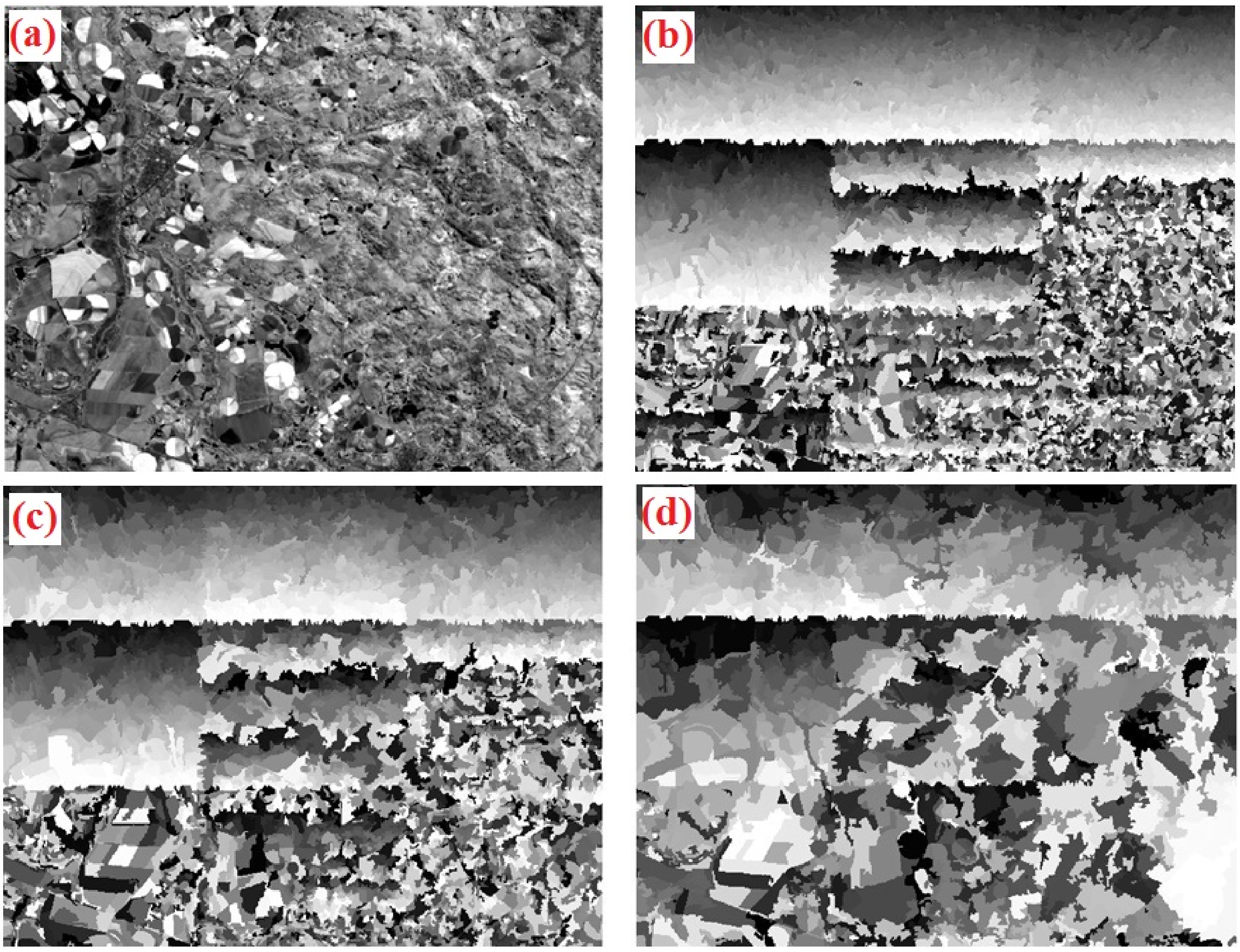

Figure 7: Landsat-8/OLI NDVI (a), multiresolution segmentation [28], with parameters 0.9 color, 0.5 compactness, 10 minimum segment size for level 1 (b), 2 (c) and 3 (d), with similarity coefficient of 0.15, 0.2 and 0.3, respectively.

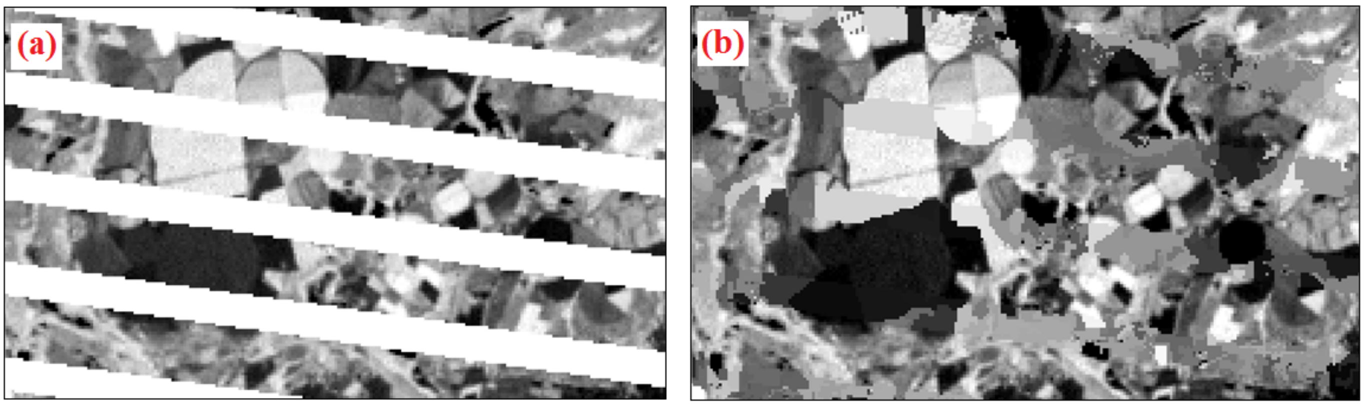

Figure 8: Landsat-8/OLI NDVI (a), multiresolution segmentation [28], with parameters 0.9 color, 0.5 compactness, 10 minimum segment size for level 1 (b), 2 (c) and 3 (d), with similarity coefficient of 0.15, 0.2 and 0.3, respectively.

5. Conclusions and Future Work

In this paper an adaptation of a method to fill Landsat-7 SLC-off gaps through multiscale segmentation approach was performed to fill recent Landsat-7 SLC-off images using segmentation of Landsat-8/OLI data through a language binding that uses Lua language to access TerraLib C++ methods. Briefly it can be concluded that:

- The TerraLib's Lua binding offered users with an open source scripting environment, initially for image processing. It allows accessing C/C++ implementations through Lua language without the need to deal with low level programming languages;
- TerraLib's implementation of multiresolution segmentation [28] was efficient to delimit NDVI images calculated from Landsat-8/OLI;
- The multiscale segmentation approach to fill Landsat-7 SLC-off gaps [9], was successfully implemented using open source methods and allowed recent Landsat-7 SLC-off images gaps to be filled using Landsat-8/OLI contours.

Future work will be focused on simulating more realistically gap filling techniques, since the method used fills the gaps with homogeneous areas; and reducing cloud quantity on orbital images using the multiscale segmentation approach.

Acknowledgments. The authors would like to thank the advices of Rodrigo Avancini regarding SWIG and the Brazilian National Institute for Space Research (INPE) and FAPESP (Project # 2014/08398-6 and Process # 2016/08719-2), São Paulo Research Foundation for funding this research.

References

[1] Exelis Visual Information Solutions ENVI Users Guide. Technical Report September, Boulder, Colorado. 2004.

[2] BOWMAN, K. P. An Introduction to Programming with IDL: Interactive Data Language. Academic Press. 304p. 2005.

[3] CÂMARA, G., SOUZA, R. C. M., FREITAS, U. M.; GARRIDO, J. Spring: Integrating remote sensing and gis by object-oriented data modelling. Computers and Graphics, v.20, n.3, p.395-403. 1996. doi:10.1016/0097-8493(96)00008-8

[4] OUSTERHOUT, J. Scripting: higher level programming for the 21st Century. Computer, v.31, n.3, p.23-30. 1998. doi:10.1109/2.660187

[5] REDDY, M. API Design for C++. Morgan Kaufmann. 472p. 2011.

[6] IERUSALIMSCHY, R., FIGUEIREDO, L. H.; CELES, W. Passing a Language through the Eye of a Needle. Queue, v.9, n.5, p.38-43. 2011. doi:10.1145/1978862.1983083

[7] IERUSALIMSCHY, R. Programming in Lua. Lua Org, 3rd edition. 347p. 2013.

[8] CÂMARA, G., VINHAS, L., FERREIRA, K. R., QUEIROZ, G. R. D., SOUZA, R. C. M. D., MONTEIRO, A. M. V., CARVALHO, M. T. D., CASANOVA, M. A., AND FREITAS, U. M. D. TerraLib: An Open Source GIS Library for Large-Scale Environmental and Socio-Economic Applications. In: HALL G.B., LEAHY M.G. (eds) Open Source Approaches in Spatial Data Handling. Advances in Geographic Information Science, vol 2. Springer, Berlin, Heidelberg, p.247-270. 2008.

[9] MAXWELL, S. K., SCHMIDT, G. L., STOREY, J. C., MAXWELL, S., K., SCHMIDT, G., L., STOREY; J., C. A multi-scale segmentation approach to filling gaps in Landsat ETM+ SLC-off images. International Journal of Remote Sensing, v.28, n.23, p.5339-5356. 2007.
doi:10.1080/01431160601034902

[10] GONZALEZ-AGULLA, E., OTERO-MURAS, E., GARCIA-MATEO, C.; ALBA-CASTRO, J. L. A multiplatform Java wrapper for the BioAPI framework. Computer Standards and Interfaces, v.31, n.1, p.186-191. 2009.
doi:10.1016/j.csi.2007.11.004

[11] ERICSON, K.; PALLICKARA, S. Adaptive heterogeneous language support within a cloud runtime. Future Generation Computer Systems, v.28, n.1, p.128-135. 2012.
doi:10.1007/s11265-014-0916-x

[12] LI, M., WALKER, D. W., RANA, O. F.; HUANG, Y. Migrating legacy codes to distributed computing environments: A CORBA approach. Information and Software Technology, v.46, n.7, p.457-464. 2004.
doi:10.1002/spe.614

[13] GDAL. GDAL - geospatial data abstraction library. 2010. URL: http://www.gdal.org. Accessed: November 21, 2016.

[14] GRASS Development Team. Geographic Resources Analysis Support System (GRASS GIS) Software, Version 7.0. Open Source Geospatial Foundation. 2016.

[15] NETELER, M., BOWMAN, M., LANDA, M.; METZ, M. GRASS GIS: a multi-purpose Open Source GIS. Environmental Modelling & Software, v.31, p.124-130. 2012.

[16] BIVAND, R. Using the RGRASS interface: Current status. OSGeo Journal, v.1, p. 36–38, May, 2007.

[17] BIVAND, R. spgrass6: Interface between GRASS 6 and R. R package version 0.8-9. 2016. Available: http://CRAN.R-project.org/package=spgrass6. Accessed: November 21, 2016.

[18] QGIS Development Team. QGIS Geographic Information System. Open Source Geospatial Foundation. 2009.

[19] COHEN, W. B.; GOWARD, S. N. Landsats Role in Ecological Applications of Remote Sensing. BioScience, v.54, n.6, p.535-546. 2004.
doi:10.1641/0006-3568(2004)054[0535:LRIEAO]2.0.CO;2

[20] STOREY, J., SCARAMUZZA, P.; SCHMIDT, G. Landsat 7 Scan Line Corrector-Off Gap-Filled Product Gap-Filled Product Development. In: Pecora. Proceedings. v.16, p.23-27. 2005.

[21] TOBLER, W. R. A computer movie simulating urban growth in the Detroit region. Economic Geography, v.46, p.234-40. 1970.
doi:10.2307/143141

[22] MAXWELL, S. Filling Landsat ETM+ SLC-off Gaps Using a Segmentation Model Approach. Photogrammetric Engineering & Remote Sensing, v.70, n.10, p.1109-1111. 2004.

[23] BEAZLEY, D. M. SWIG: an easy to use tool for integrating scripting languages with C and C++. In: Fourth USENIX Tcl/Tk Workshop Proceedings. p. 129-139, 1996.

[24] ZHU, Z.; WOODCOCK, C. E. Object-based cloud and cloud shadow detection in landsat imagery. Remote Sensing of Environment, v.118, p.83-94. 2012.
doi:10.1016/j.rse.2011.10.028

[25] ZHU, Z., WANG, S.; WOODCOCK, C. E. Improvement and expansion of the fmask algorithm: cloud, cloud shadow, and snow detection for landsats 47, 8, and sentinel 2 images. Remote Sensing of Environment, v.159, p.269-277. 2015.
doi:10.1016/j.rse.2014.12.014

[26] JENKERSON, C. User guide: Earth resources observation and science (EROS) center science processing architecture (ESPA) on demand interface. 2013. URL: http://pubs.er.usgs.gov/publication/70057873. Accessed: November 21, 2016.

[27] ROUSE, J. W., HAAS, R. H., SCHELL, J. A.; DEERING, D. W. Monitoring vegetation systems in the Great Plains with ERTS. Proceedings, 3rd Earth Resource Technology Satellite (ERTS) Symposium, v. 1, p. 48–62. 1974.

[28] BAATZ, M.; SCHAPE, A. Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation. Angewandte Geographische Informations verarbeitung XII. p.12–23. 2000.

[29] GAMANYA, R., DE MAEYER, P.; DAPPER, M. An automated satellite image classification design using object-oriented segmentation algorithms: A move towards standardization. Expert Systems with Applications, v.32, n.2, p.616-624. 2007.
doi:10.1016/j.eswa.2006.01.055