AMERICAN MUSEUM ⁂ NATURAL HISTORY
CENTER FOR BIODIVERSITY AND CONSERVATION

3 October 2013

Training Guide to Classify Satellite Images using Segmentation and Random Forests

Introduction

This guide explains how to classify satellite imagery using image segments instead of individual pixels. This technique is often called segment-based or object-based classification. The results from a segment-based classification have less speckle (salt and pepper effect) then you would get using a pixel-based approach. Another advantage is that once segments are created classification processing is much faster than using a pixel-based approach since the number of image segments being classified is much less than the number of pixels in the image.

This guide is designed to use freely available open source software. We will use the Orfeo Toolbox (OTB) for segmentation, R for image processing and recommend using QGIS for digitizing training data as well as image and vector display. All of the examples assume you are using a recent Windows operating system but all of the software runs on Mac OS and Linux. I work primarily with Ubuntu and have tested all of these steps on Ubuntu 12.04. I have not tested this guide using Mac OS. If you prefer to use different software to create image segments and/or calculate feature data you should look at the appendix section 6.4 for important information.

This guide is split into three sections:

1. Image segmentation
2. Calculating segment features
3. Classifying image segments
4. Suggestions to improve results

There are appendices at the end with information about installing the software required to follow this guide as well as information about licensing and citing this document.

Before continuing you will need to install image segmentation, GIS software, and R. When installing R you will need to import the following packages:

raster, data.table, maptools, randomForest, rgdal, sp

1. Image segmentation using the Orfeo Toolbox (OTB)

Image segmentation involves making contiguous groups of similar pixels in an image with the intent of transforming a regularly spaced grid of pixels to meaningful objects from which a broad range of segment features or descriptors can be derived. Each segment (often called objects) is a contiguous group of pixels that a segmentation algorithm determined were similar within the constraints of the algorithm parameters. There are several different segmentation algorithms. For this guide we will be using the mean-shift algorithm. OTB offers other segmentation algorithms and it's a good idea to experiment with the alternatives. Basic information about image segmentation and different methods that are used to segment images can be found on Wikipedia (http://en.wikipedia.org/wiki/Image_segmentation).

Segmentation algorithms usually have several parameters that you can define. It's important to understand what the parameters do and more importantly take time to try different parameters to see how the results change. Since many of the segmentation algorithms are computationally intensive it's a good idea to use one or more relatively small image subsets that represent the features you are interested in classifying so you can determine which parameters will produce the most meaningful segments. This trial and error process can be time consuming, often requiring several runs, but it is time well spent since the output from segmenting the entire image will impact the quality of the final classification.

1.1. Segmentation using OTB

The OTB segmentation application provides two output options, vector and image (raster) files. With image output each segment has the same unique pixel value. In other words, each segment will be a contiguous group of pixels with the same pixel value and no two segments will have the same pixel value. The vector output is a series of polygons defining segment borders with each polygon assigned a unique identifier in the attribute table. If the segmentation parameters are the same for both raster and vector output the segments will have identical shapes (the vector polygons follow image pixel edges) but their segment IDs will be different. If you want to have raster and vector output with the same segment IDs then I suggest you output a vector file and convert that to a raster image using GIS software or “gdal_rasterize” (appendix C) which is quite fast.

One limitation when outputting raster files is that it can require a lot of memory since the entire input image must reside in memory and additional memory is required to store temporary images created during the processing.

If vector output is needed or if there is not enough memory to segment a large raster image the OTB segmentation application provides an option to create vector polygon output. One advantage of outputting a vector file is that it is possible to apply the segmentation to subsets (tiles) of the input image and then assemble the tiles into a single vector file at the end of the processing. By segmenting one tile of the image at a time it is possible to segment large images without running out of memory. When performing segmentation on a tile-by-tile basis there will be artifacts at the edge of the tiles since the

tile boundary and segment boundaries do not match. In other words the segments at the tile boundary use the edge of the tile as the segment boundary (Figure 1). Fortunately there is an option to stitch these edge segments together using “Stitch Polygons” in the OTB application. The polygon stitching process merges segments from adjacent tiles that appear to be similar enough to combine. This process doesn't remove all of the artifacts but it does a pretty good job (Figure 2).

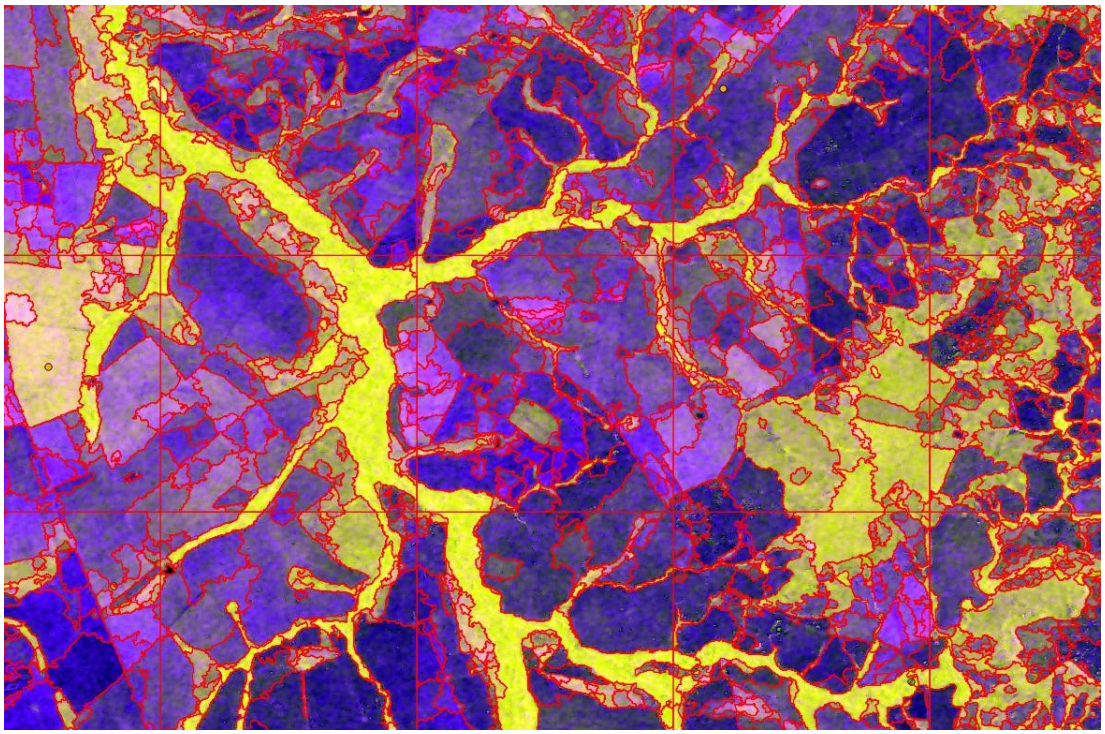

Figure 1: Vector segmentation result (red lines) using tiles

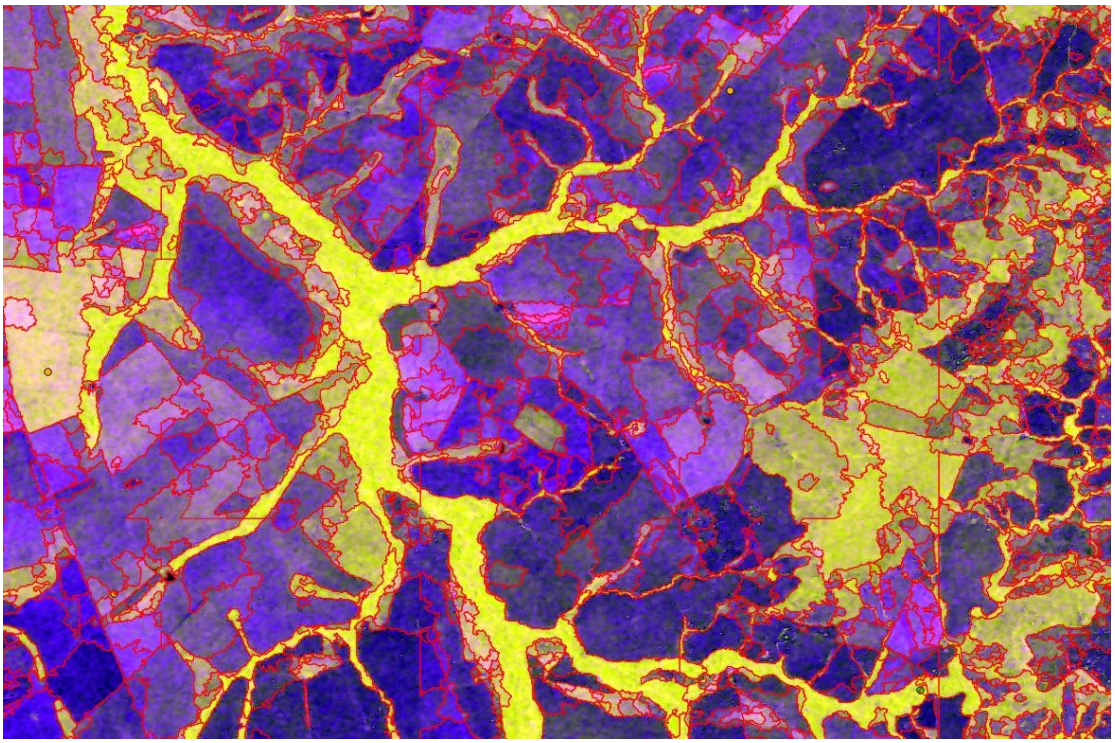

Figure 2: Vector segmentation (red lines) after polygon stitching

We OTB Segmentation application can be run using the Monteverdi application that effectively provides a graphical user interface for many functions in the OTB library. See Appendix 5.2 for information about installing Monteverdi. When you launch Montiverdi you will see the “OTB Application brower” in the lower-left corner of the Monteverdi window (Figure 3). To start the Segmentation application you need to Scroll down to the "Segmentation" group and double-click on the "Segmentation" program link. This will open the “Segmentation” window (Figure 4).

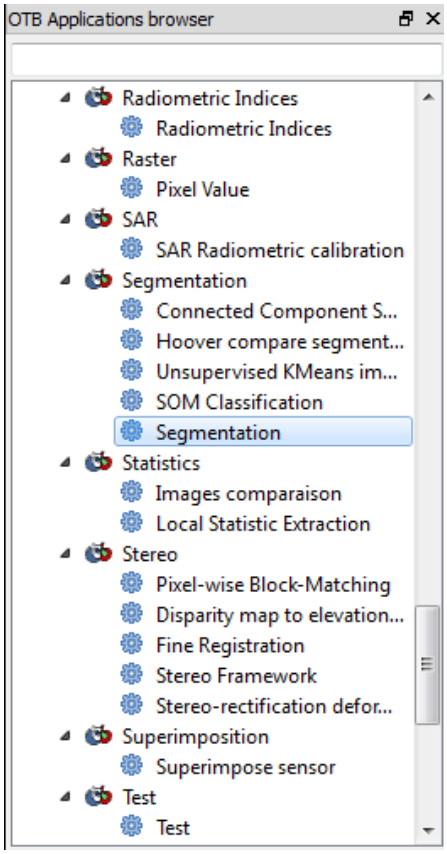

Figure 3: OTB Application Browser window in Monteverdi

1.1.1. Raster output

First we will illustrate how to create a raster image. You can experiment with different settings to better appreciate how each setting effects the output. This trial and error testing is best done on a small image (~1024 x1024 pixels) that is representative of the types of objects you want to classify. Larger images can also be used but they might take a while to create each output image. The parameters that you select can influence how fast segments are created. In general larger segments are processed faster than smaller segments. Here is an overview of the different settings along with some of my observations:

Input image: Enter the file name and location of the image that will be segmented.

Segmentation algorithm: There are different segmentation algorithms that can be selected using the drop-down menu. I've had good luck using "Mean-Shift" and that is what will be used in this guide although it's good to experiment with other algorithms. Mean-shift is a segmentation algorithm that uses both feature space and the spatial domain to create segments.

Spatial radius: This determines how many pixels across the image the algorithm will search to define a contiguous segment. Increasing this value will tend to increase the size of the segments.

Range radius: This determines the range of units in feature space that will be searched to define a segment. If the image layers (bands) are all spectral data, such as a satellite image, then the feature units are pixel DNs. Increasing this value will allow more pixel value variation within a single segment so segments will be less homogeneous.

Mode convergence threshold: This defines a threshold that determines when the algorithm will stop iterating to define a segment. Increasing this tends to increase heterogeneity within segments by merging segments that would have been created using a lower convergence threshold into larger segments. The minimum threshold value is 0.1.

Maximum number of iterations: This determines the maximum number of iterations that will take place if the convergence threshold is not reached. Reducing this number has a similar effect as increasing the convergence threshold. Increasing the number of iterations over 50 seems to have little if any effect on the output.

Minimum region size: This determines the minimum number of pixels allowed in an individual segment. Increasing this number will merge smaller, sometimes very different, segments into neighboring segments. The minimum region size can be thought of as a minimum mapping unit.

Processing mode: Select "Standard segmentation with labeled raster output" from the drop-down menu. This is the selection to output a raster image. The vector option is described below.

Output labeled image: Enter the name of the output image in the text box. Make sure you include the file extension to specify the output image format. For example using ".tif"

will create a GeoTiff image. Any GDAL compatible output image format can be specified by using the appropriate filename extension (for more information go to: http://www.gdal.org/formats_list.html). Also, click on the button to the right of the text box and select “uint 16” or “uint 32” from the drop-down menu. This specifies the data type of the output image. I usually select 32-bit unsigned integers to insure there is a sufficient data range (segment IDs from 0 to 4,294,967,295) for the output segment numbers. In many case unit 16 (segment IDs from 0 to 65,535) will work fine and the output images will take up less space.

When all of the parameters have been entered click on the “Execute” button to start processing. There is a progress bar in the lower-left corner of the application window. When processing in complete the text “Ready to run” will be displayed at the bottom of the window.

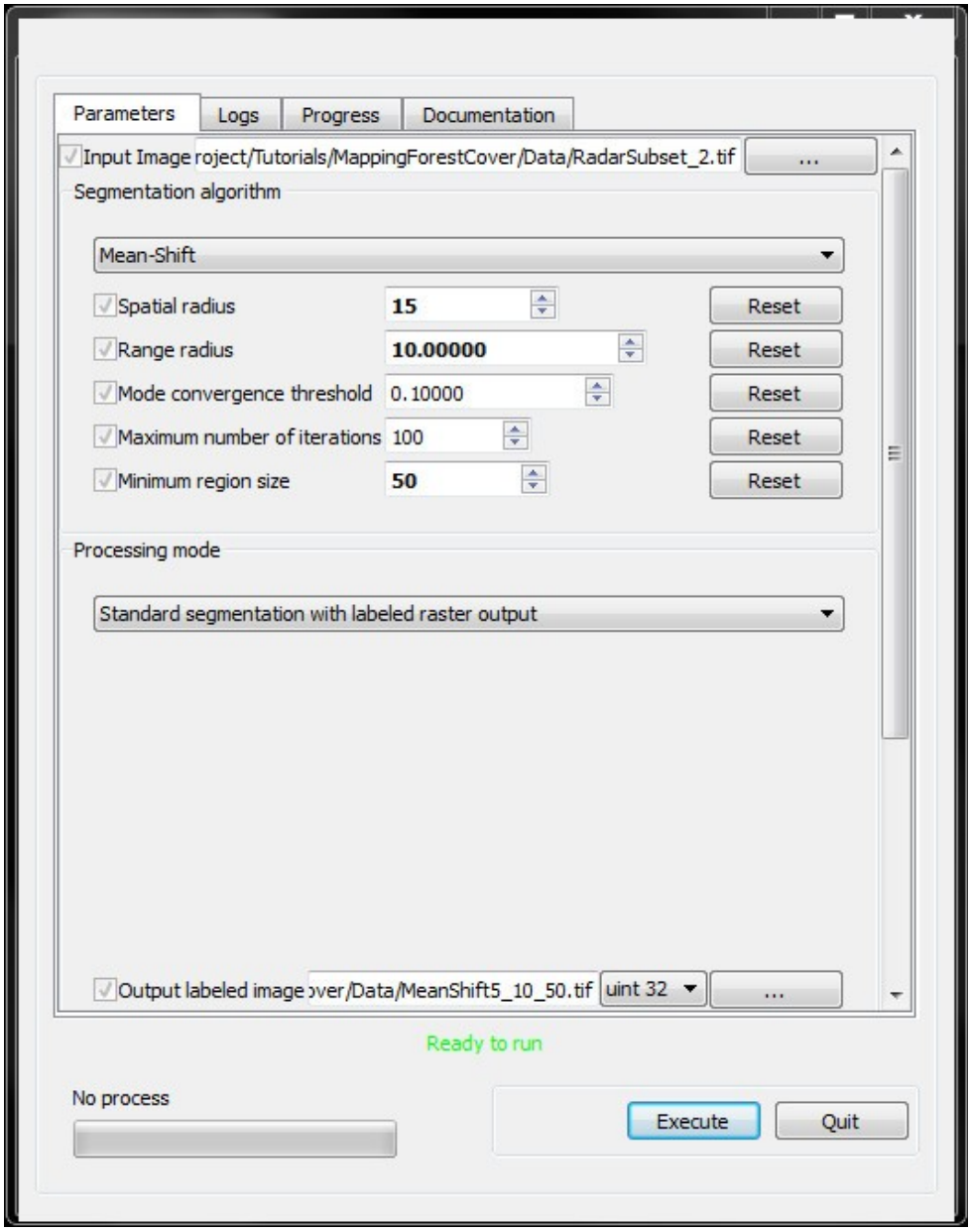

Figure 4: Input parameters for OTB segmentation with raster output

1.1.2. Vector output

To create vector output you can use the same parameters that work for raster output although you need to select "Tile-based large-scale segmentation with vector output" from the "Processing mode" drop-down menu. That will display a series of additional parameters (Figure 5). Here is an overview of the different settings along with some observations:

Output vector file: Enter the name of the output vector file in the text box. Make sure you include the file extension (e.g., ".shp") to specify the format of the output vector file. Any of the output formats supported by OGR (see this link for more information: http://www.gdal.org/ogr/ogr_formats.html) can be used.

Writing mode for the output vector file: This specifies the behavior of how the vector file will be created if a file with that name exists. The exact behavior depends on the output file format.

Mask Image: It is possible to use a mask image to define the pixels of the image being segmented that should be used. Any pixels in a mask image with positive integer values will be segmented. A mask image is helpful if you want to segment an image using multiple segmentation settings.

8-neighbor connectivity: If this is selected 8-neighborhood connectivity would be used when searching for similar pixels to include in a segment. The default is to use 4-neighborhood connectivity.

Stitch polygons: If this option is checked and tiling is used to create segments then segments in adjacent tiles will be connected if they are similar. This greatly reduces the artifacts that are a result from the tiling approach (see figures 1 and 2).

Minimum object size: This allows you to specify a minimum polygon size. This is not typically necessary if you define a *minimum region size*.

Simplify polygons: Simplifying polygons will output less jagged polygons but for our processing we want the polygons to match pixel boundaries so this checkbox should not be checked if you want to classify the segmented image.

Layer name: The meaning of "layer name" depends on the vector file format being created. The default "layer" is usually fine. If you want to change the layer name enter the name of the layer.

Geometry index field name: This is the segment ID attribute name for polygon attribute table. If you want to change this name enter the name of the attribute you want to use. The default "DN" is fine for our use.

Tiles size: This defines the size, in pixels, of the image subsets that will be used to break up the image into tiles that are small enough to be handled with your computer's memory.

If the entire input image can fit in memory then it is best to make the tile size the size of the largest image dimensions so tiling is not used. For example, if your image is 1500 lines x 2000 samples and you have enough memory to process the entire image without tiling then enter 2000 for this parameter. That will create an output vector file with no tiling and therefore no tile boundaries. If this is left blank a "optimal" tile size is selected according to available RAM.

Starting geometry index: This is the starting value of the polygon attribute named in the "geometry index field name" described above. The default value "1" is fine.

OGR options for layer creation: This allows you to enter a list of layer creation options in the form KEY=VALUE that will be passed directly to OGR without any validity checking. Options will depend on the output vector file format, and can be found in OGR documentation. This is for advanced users who are familiar with the options defined in the OGR file format specifications: http://www.gdal.org/ogr/ogr_formats.html.

When all of the parameters have been entered click on the "Execute" button to start processing. There is a progress bar in the lower-left corner of the application window. When processing in complete the text "Ready to run" will be displayed at the bottom of the window.

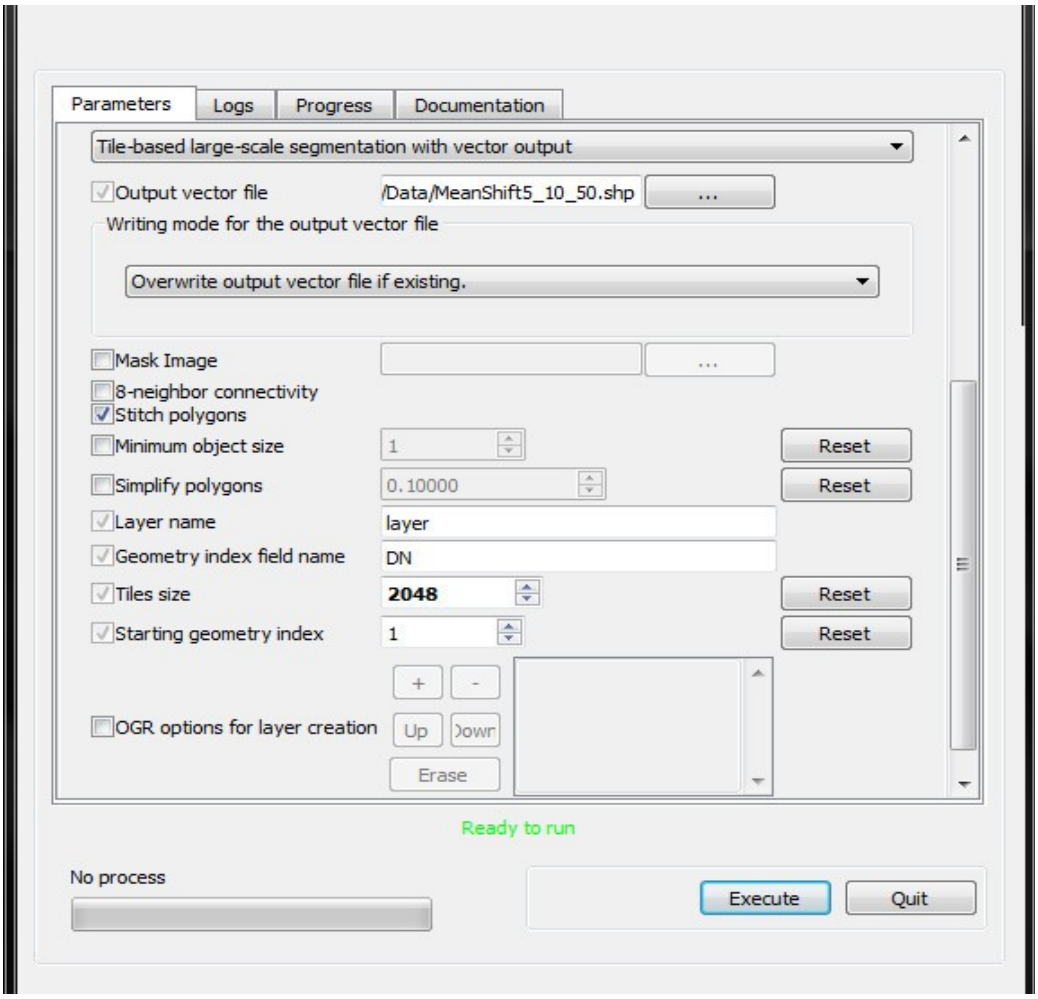

Figure 5: Input parameters that are unique for OTB segmentation with vector output

After the segmentation is finished you can close the segmentation window then close the OSGeo4W terminal window by entering “exit” on the command line and hitting return.

1.2. Displaying the segments

After segmenting an image it is a good idea to evaluate the results by overlaying the results on the image that was segments. It is easier to display vector rather than raster segments. The problem with raster segments is that it is difficult to see individual segments since their IDs range from 0 to several thousand and segments with ID values that are not very different will appear to be one big segment. If you overlay the vector segments on the image that was segmented you will need to remove the polygon fill value or make the polygon fill transparent so you only see the polygon outlines on the image.

2. Calculating segment features

Segment features are the predictor variables that will be used to create the random forest model in the next section. These features can be statistical variables such as per-band means of all pixels in a segment or shape characteristics such as area or length to width ratio. Features can also contain information about how they relate to neighboring segments. The current version of the feature calculation script calculates the following statistical features.

- Mean for each image band
- Standard deviation for each image band
- Coefficient of variation for each image band
- Mean, standard deviation and coefficient of variation for each band after removing 40% of the pixels by clipping 20% of the pixels and 20% of the pixels with the highest DNs to reduce effects from pixel outliers included in the segment
- Median for each band

Additional shape features such as area, perimeter, and shape index have been tested but a not in the scripts at this time. Neighborhood metrics are also being investigated.

The script provides options for creating features from either raster or vector segments. Using raster segments is much faster (~1000 times faster) but if the segment image is too large to fit in in your computer's memory you might need to calculate segment features using a vector segment file. Using vector segments the image is processed by extracting the pixels one segment at a time then calculating features as each segment is extracted. This is a very slow process and should only be used if processing the raster image is not possible. The results (output CSV file) will be identical regardless if vector or raster segment files are used.

The output from the R script will be a comma separated value (CSV) file with one row for each segment and columns for segment number and the features that are calculated. The CSV file will be used in the next step to create the classified map.

2.1. Editing the script

An R script has been created that calculates the above-mentioned features. Before the script can be run it is necessary to edit a few variables so it works with your data. You can edit the script in R or using any text editor. To edit the script in R you first need to start R by either clicking on the “R” icon which is probably on your desktop or by using “All Programs => R => R 2.15.2” (or whatever version you are using). When R open you will see the R Console window open inside the main R application. To open the script use File => Open script and open “CreateSegmentFeatures.R” on your computer. The script will open in its own script editing window.

It is important to note that R is case sensitive. In other words the letter “t” is different from “T” so be careful when you edit the script.

Near the top of the script, where it is written “SET VARIABLES HERE”, there are a number of attributes that can be changed to customize it for your computer. Below is a list of the variables with an explanation. Only change the text after the “<-” symbol. If you prefer, you can replace the “<-” with “=”. R interprets both of those the same.

satImage <- 'C:/Forest_cover_exercise/Data/RadarSubset_2.tif'

Name and path for the segment image. If using a vector file for segments use two single or double quotes with no space between them to specify no raster image is being used.

segVector <- ""

Name and path for the segment polygon vector file. If using a raster image for segments use two single or double quotes with no space between them to specify no vector data are being used.

idAttributeLabel <- "DN"

Label for the attribute field in the Shapefile that has segment IDs. This is ignored if segment data comes from an image.

segImage <- 'C:/Forest_cover_exercise/MeanShift15_10_50.tif'

Name and path for the segment image .

outFeatures <- 'C:/Forest_cover_exercise/Data/segmentFeatures.csv'

Name and path of the output CSV file

ndSat <- 0

No-data value for the input satellite image.

ndSeg <- 1

No-data value for the input segment image.

Note: When specifying the directory path in R for a Windows computer you may use a double-backslash (“\\”) or a single forward slash “/”). For example the directory path: C:\Data could be typed C:\\Data or C:/Data. Make sure the directory path and file name is exactly as it appears when you list the directory contents. In Linux systems only a single forward slash (“/”) can be used for directory paths. Also, note that in R file names should not start with a number.

Other parts of the script can also be modified but you will need to have a good understanding of how the different commands work. The script file must be saved as an ASCII text file.

2.2. Running the feature calculation script

Now it's time to run the script and create the CSV file. If R is running and the script is open then click on the script editing window to make it active and then use “Edit => Run all” to run the script. As the script runs you will be able to monitor the progress in the R

Console window. The red lines are the text from the script and the blue lines are the output messages from R.

If the script is running properly you will see the lines of code displayed in the R Console window. When the script is complete the time it took to create the CSV file is printed in the terminal window. Open the CSV file using Excel or any text reader such as Notepad. The first line should have the heading labels and the remaining lines should have the values for each segment.

3. Classifying image segments

To classify the image segments you will need to select training data to inform the random forests model how each segment should be classified. Training data must be in a vector file and can be created using a GIS. Once training data are digitized you can edit and run the R script to create a classified image.

3.1. Selecting training data

To classify the image segments you will need to select training data to inform the random forests model how each segment should be classified. Training data must be in a vector file and can be created using information collected in the field or using a GIS. Once training data are digitized you can edit and run the R script to create a classified image.

3.2. Selecting training data

Training data can be saved as either vector points, lines or polygons. Picking points is probably easiest and they are the fastest to process in the script. A common way to select training data using a GIS is to display an image that can be visually interpreted in the background with a vector layer of the segment boundaries overlaid so it is easy to determine the dominant land cover type for each segment that is selected for training. If you decide to use points to save the training data you simply add a point in each segment polygon that you want to include in the training data set. This is usually done by clicking in the polygon then typing in the attribute information. When you are done selecting training data make sure you save the file. The script can read most vector file format although the most common is the ESRI Shapefile format.

When creating a vector file you must define an attribute of data type integer to store a class number for each class that will be included in the output classified map. Additional attributes will be ignored but it is helpful to also record a text attribute for each class as a way to verify that each training object (point, line or polygon) is assigned to the right class. Figure 6 is an example of an attribute table. Note that the “cover” attribute in the example in Figure 6 is optional and will be ignored by the script.

id	type_id	cover
0	3	Water
1	3	Water

2	4	Other
3	4	Other
4	4	Other
5	1	Tall shrub
6	1	Tall shrub
7	2	Short shrub

Figure 6: Example attribute table for training data

To select a segment using points you need to digitize one point in each segment that you want to use for training data. If you use vector lines any segment that is touched by a line will be selected. Polygons will select all segments that have at least one pixel center included in the polygon.

In general, the more training data you collect the better your results will be. When selecting training data it is important that you select segments that represent all of the variation in the image. For example if you are interested in classifying a forest class you need to make sure you select segments that cover all of the variation that can occur within the forest class (shadowed, bright, sparse, dense...). It's fine if there is a lot of variation in your training data for a single class. For example, you could have a cloud and shadow class that included training segments of cloud and shadow for that single class. That is one of the advantages of using non-parametric classifiers like random forests.

Toward the end of the script an assessment of the quality of the training data is conducted. The metric used for this assessment is “margin”. The margin of a training segment is the proportion of votes for the correct class minus maximum proportion of votes for the other classes for that segment. Positive margin values represent correct classification, and vice versa. The margin data are written to a point ESRI Shapefile so they can be overlaid on the image and segment polygons to assess which segments need to be removed and relabeled in the training data and it can help determine which classes needs additional training segments.

Up to three image layers can be created in the output image. The first choice is to create a classified image. This is the typical output of a classification algorithm. The classified image will have pixels labeled by the class number used in the training data.

There is also an option to create a class probability image which gives the probability that the classification label is correct for each pixel. A low probability means there was significant confusion with other classes and a high probability indicated low confusion with other classes.

The third type of output is a classified image that has all pixels with a class probability below a user-defined threshold set to zero. In other words, this output will look similar to the first option (classified image) except pixels that have a low probability value will be classified as zero (“0”). This can be helpful in determining which areas can use better quality training data.

After the classification is run you will likely want to edit the training vector file to improve the results.

3.3. Editing and running the classification script

The R script "SegmentRF_Classification.R" will be used to classify the image using training data created in step 3.1 and the segment features file created in section 2. The R script runs the random forests model to calculate relationships between the features calculated from the segments in the previous step (predictor variables) and written to a CSV file and the vector training data (response data). After the random forests model is calculated it is applied to the image segments to predict which class each segment belongs to.

Follow the same procedure outlined in section 2.1 to edit the variables in the script. Here is a list of the variables that should be edited:

SegCsv <- 'C:/Forest_cover_exercise/Data/segmentFeatures.csv'

Name and location of segment feature data CSV file output from section 2

segImage <- 'C:/Forest_cover_exercise/MeanShift15_10_50.tif'

Name and location of the segment raster image

nd <- 1

Segment raster nodata value.

outImage <- 'C:/Forest_cover_exercise/Forest_NonforestMapOriginal2.tif'

Name and location of the classified image. The output classified image format is GeoTiff.

outMarginFile <- 'C:/Forest_cover_exercise/marginData.shp'

Name and location of the output point Shapefile point file that will be created with the margin data. Note that if this file exists the write will fail with the message "Creation of output file failed".

trainingDsn <- 'C:/Forest_cover_exercise/trainingPoints'

Data set name for the vector file containing training data. This is often a file name or directory. This and "trainingLayer" are defined by the ORG drivers. Look at http://www.gdal.org/ogr/ogr_formats.html for more info about dataset name and layers for the format of the vector file you are using.

trainingLayer <- 'clippedTrainingData'

Layer name for the vector file. This is often the file name without an extension but it depends on the vector file format of the training data.

classNum <- "class_int"

Enter either the name (case sensitive and in quotes) or the column number of the field containing class (forest or non-forest) number

outClassMapCSV <- 'classMapping.csv'

Output CSV file with class mapping information. If this output is not needed you can enter two double or single-quotes (“” or '').

classImage <- TRUE

Option to output classification without applying threshold (enter TRUE or FALSE). This is the typical classification output.

probImage <- TRUE

Option to output probability image (enter TRUE or FALSE) with the probability of correct classification output for each pixel.

threshImage <- TRUE

Option to output classification and set pixels with a probability of correct classification under a user defined threshold (“probThreshold”) to 0 (enter TRUE or FALSE).

probThreshold <- 75

Probability of correct classification threshold in percent (values must be between 0 and 100) to be applied if “threshImage” is set to “TRUE”. This is only used if threshImage=TRUE.

Use the same steps described in section 2.2 to run the script. If R is running and the script is open then click on the script editing window to make it active and then use “Edit => Run all” to run the script. As the script runs you will be able to monitor progress in the R Console window. The red lines are the text from the script and the blue lines are the output messages from R.

If the script is running properly you will see the lines of code displayed in the R Console window. After a few seconds you should see a graph like the one in Figure 7. This graph shows the importance of each of the predictor variables. Circles further to the right indicate that the variable has a higher influence (it's more important) on the forest / non-forest prediction. In the R Console you will see “OOB estimate of error rate:” with the error in percent followed by the confusion matrix calculated using the out-of-bag (randomly selected samples used for testing the model and not used for training the model) samples.

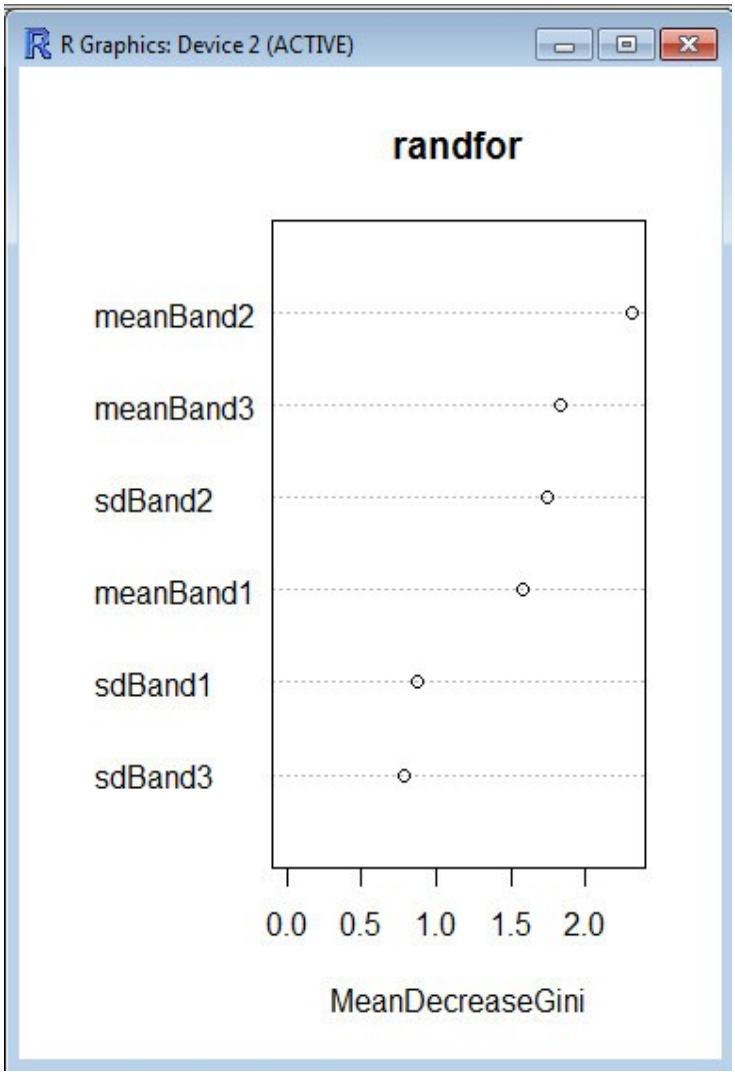

Figure 7: Variable importance plot

After the classification is finished you should display the results in a GIS to see how it looks and if necessary edit the training data and try again.

4. Edit classification results

After the classified map is output you will likely see some errors. One way to address errors is to improve your training data or obtain additional layers that can be added to the image stack being classified. Another option is to manually change (correct) the classification. The script "Edit_Segment_Classification.R" was designed to facilitate manually editing an output segment map.

The “Edit_Segment_Classification.R” script is used to edit the classified image output from the script “SegmentRF_Classification.R”. When running the classification script you need to make sure you set the variable "outClassMapCSV" to a file. This CSV file is used to keep track of the class assignment (label) for each segment.

To edit the class labels you need to create a point vector file in a GIS to define the new class label for specific segments. The point file will need at least one attribute of type integer to enter the new class labels (Figure 8). For each segment to you want to edit place a point and enter and new class label that should be assigned to that segment. The new class labels are used by the script to create a new classified image with the new class labels.

	newClass
0	2
1	2
2	2
3	2
4	1

Figure 8: Sample of the attribute table for a point vector file holding new class labels for a classified segment image.

4.1. Editing and running the segment editor script

The R script “Edit_Segment_Classification.R” will be used to edit the classified image. Follow the same procedure outlined in section 2.1 to edit the variables in the script. Here is a list of the variables that should be edited:

segImage <- 'C:/Forest_cover_exercise/MeanShift15_10_50.tif'

Name and location of the segment raster image

nd <- 1

Segment raster nodata value.

outImage <- "C:/Forest_cover_exercise/Forest_NonforestMapEdited.tif”'

Name and location of the edited classified map

inClassMapCSV <- 'C:/Forest_cover_exercise/classMapping.csv'

Input class mapping CSV file created form the SegmentRF_Classification.R script

editPointsDsn <- 'C:/Forest_cover_exercise /TestData/editSegs.shp'

Data set name for the vector point file containing locations and class assignments for segments to be modified. This is often a file name or directory. This and "layer" are defined by the ORG drivers.

editPointsLayer <- 'editSegs'

Layer name for the vector file. This is often the file name without an extension

newClassNum <- "newClass"

Enter EITHER the name (case sensitive and in quotes) or the column number of the field containing class (forest or non-forest) number

5. Suggestions to improve results

The results from the classification can often be improved by adding additional layers to an image, such as a multi-band satellite image, that is being classified. Some example additional layers are NDVI, principle component bands, and texture bands derived from NDVI or individual image bands. You can also add other layers such as slope, aspect, elevation, distance to roads, soil maps, and just about any other layer that would help separate the class you want to map. When using these other layers is will likely be best if you create the segmented image using only spectral layers (e.g., the original satellite image) and then add the additional layers to the original image before running the script that calculates segment features. By reviewing the variable importance plot (Figure 7) you can get an indication which features are most important (have the greatest impact) for your classification task.

Use the same steps described in section 2.2 to run the script. If R is running and the script is open then click on the script editing window to make it active and then use "Edit => Run all" to run the script. As the script runs you will be able to monitor progress in the R Console window. The red lines are the text from the script and the blue lines are the output messages from R.

6. Appendices:

6.1. Download and install R

R can be found on the R Project web site: http://www.r-project.org/. There is also a guide available on the CBC website with detailed instructions on downloading and installing R on Windows: http://biodiversityinformatics.amnh.org/index.php?section=R_Scripts.

You can follow the guide using the default tools that come packaged with R but I recommend you download and learn to use an Integrated Development Environment (IDE) like Rstudio. RStudio is an open source IDE for R that provides a simple but powerful interface for Windows, Mac and Linux computers. You can download RStudio from the website: http://www.rstudio.com/ide. There are several helpful guides to learning to use RStudio on their website: http://www.rstudio.com/ide/docs/.

6.2. Download and install the Monteverdi software

The open source Monteverdi software can be downloaded from their Sourceforge web site: http://sourceforge.net/projects/orfeo-toolbox/files/. The current stable version for Windows as of September 2013 is "Monteverdi2-0.4.0-win32.exe". You can install this like you would install any Windows software.

6.3. Convert vectors to an image using gdal_rasterize

"gdal_rasterize" is a utility provided by the GDAL library which was downloaded along with OTB. The GDAL library is another open source software package that provided high performance image processing capabilities. Additional information about GDAL and instructions to run it on other operating systems can be found at: http://www.gdal.org/. If you want to convert a vector file to a raster Geotiff file you can use a command like this:
gdal_rasterize -ts 2048 2048 -a DN MeanShift15_10_50.shp MeanShift15_10_50Rasterize.tif

Where:
-ts 2048 2048: is the width and height (in pixels) of the output image
-a DN: is the attribute in the vector file that will be used to specify the output pixel values
MeanShift15_10_50.shp: is the input vector file
MeanShift15_10_50Rasterize.tif: is the output image file

More information about gdal_rasterize parameters can be found at the gdal_rasterize home page: http://www.gdal.org/gdal_rasterize.html.

6.4. Using other software to derive segments and features

A number of people use other software to delineate segments and calculate feature information to be used to classify an image. If you want to use these R scripts to calculate feature data from your segments produced using other software you should be able to follow the instructions in section 2 of this guide without any problem. I strongly suggest that you use an image for the segment information (variable segImage) instead of a vector. The segment image should be a single band image with pixel numbers representing segment IDs. Converting from a vector layer to an image (raster) layer is described in appendix 6.3.

If you use other software to create the segment features you should be able to follow the instructions in section 3 although you will need to be sure that the label for the column containing segment IDs in the CSV file holding the segment feature data (variable segCsv) is "segnumber". This is the column (field) that holds the segment id for each segment (Figure 9). The segment numbers in that column should correspond to segments

in the segment image. Formats other than a CSV file can be used to enter information about segment features but you will need to make changes in the script so that a data frame is created and the label for the field holding segment IDs is is "segnumber".

	A	B	C	D	E	F	G
1	segnumber	meanBand1	meanBand2	meanBand3	sdBand1	sdBand2	sdBand3
2	1	124.6981132075	80.5471698113	44.1509433962	4.3393765312	6.8263235903	7.2573470105
3	2	112.9248120301	62.2330827068	50.6917293233	4.7444130782	7.5707643604	8.6244722729
4	3	103.6315789474	45.701754386	57.9298245614	14.904016212	21.8950132224	15.5401089915
5	4	94.9113924051	48.1392405063	46.9367088608	12.4140014762	15.5586023507	12.5497387573
6	5	105.2476190476	67.7047619048	37.7523809524	11.2067810451	10.2450023553	12.4805856559
7	6	122.641025641	90.4145299145	32.2863247863	8.145102047	7.1605711638	8.5069565674
8	7	103.3197674419	45.7441860465	57.5755813953	10.4135013551	11.3963018999	12.3676661633
9	8	106.4611111111	46.9166666667	59.5444444444	6.7529127908	12.2542319677	14.0463407916
10	9	134.9803921569	79.8235294118	55.1568627451	10.9132766777	10.5236987459	14.0987553337
11	10	131.6111111111	85.5555555556	46.0555555556	8.0667420341	13.3694479454	9.9819018619
12	11	121.3780487805	74	47.3780487805	5.970380065	5.5355194254	7.9762614528
13	12	82.2	32.6272727273	49.7272727273	12.2944009036	12.0723044104	15.3813356997
14	13	116.7777777778	76.6349206349	40.1428571429	10.9108475831	8.5010465908	10.5904051871
15	14	97.893081761	46.9245283019	50.9685534591	12.7252321831	14.1767982896	13.6114347794
16	15	111.0327868852	66.7704918033	44.262295082	8.8543157332	6.5711324306	7.8928272065

Figure 9: Example CSV file containing feature information

6.5. Citations and license information

If you cite this document we ask that you include the following information:
Horning, N. 2013. Training Guide to Classify Satellite Images using Segmentation and Random Forests v0. American Museum of Natural History, Center for Biodiversity and Conservation. Available from http://biodiversityinformatics.amnh.org/. (accessed on *the date*).

This document is licensed under a Creative Commons Attribution-Share Alike 3.0 License. You are free to alter the work, copy, distribute, and transmit the document under the following conditions:

- You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

Any questions or comment related to this document should be sent to Ned Horning – horning@amnh.org.

6.6. Acknowledgements

I would like to thank Google.org for supporting this effort.