Change Detection Workflow:

Using Python and Arcpy Module

By

John Dees

Professor Miller

GISC 3200K

University of North Georgia

30 July 2014

Background

Detecting change over time is a common task for remote sensing professionals. Applications may include measurement of urban growth, detection of wetland change, and measurement of foliage and tree cover, among many others.[[1]](#footnote-1) Data sources for such projects are freely available in the public domain, such as NASA’s LANDSAT program and the Department of Agriculture’s NAIP program, which freely provide high quality satellite and areal imagery, respectively, over a broad timeframe. An analyst can easily access imagery of a location at multiple timeframes, and then compare either the raw or classified imagery via map algebra algorithms in a multitude of software packages. This project seeks to create an automated workflow for performing such comparisons, starting with only raw imagery. The final product of the script development will be a tool, which can be used and shared inside the ArcMap environment.

Requirements

ESRI’s Spatial Analyst extension for ArcMap offers several tools to help accomplish these types of classifications and change detection, both within ArcMap environment and externally in the PYTHON scripting environment by means of ESRI’s arcpy and arcpy.sa modules. Arcpy and arcpy.sa, together, offer access to all of ArcMap’s geoprocessing tools and Spatial Analyst’s more specialized toolbox. For classification purposes, Spatial Analyst offers the ISO Unsupervised Classification tool, which “performs unsupervised classification on a series of input raster bands using the Iso Cluster and Maximum Likelihood Classification tools.”[[2]](#footnote-2) For change detection purposes, it is possible to use Spatial Analyst’s Map Algebra capabilities to compare the individual pixels of two spatially consistent rasters using Boolean logic, and output a new raster that indicates locations where pixel classification has changed.[[3]](#footnote-3)

The data preparation and comparison process requires multiple steps, which if performed manually and repetitively by an analyst can be labor intensive. The images to be compared must be clipped to an area of interest (AOI) so that they are spatially consistent. The images must be classified independently. Then, the classified images must be compared using Map Algebra to output a change raster. The lengthy and iterative nature of this workflow offers a prime opportunity for automation with PYTHON scripting.

Methods

The automation of the change detection script is comprised of several distinct pieces: (1) Modules, (2) Environment, (3) Inputs, (4) Clipping, (5) Classification, (6) Map Algebra, and (7) Conversion to Tool. Each of these components were initially chained together in a script, external to the ArcMap environment, but employing the use of arcpy and the arcpy.sa Spatial Analyst toolset. The clipping and classification components make use of arcpy and arcpy.sa geoprocessing tools. The Map Algebra methods are not geoprocessing tools, but rather methods available with the arcpy.sa suite. The script logic will be explained in the remainder of this section, as well as the conversion of the script into an ArcMap tool. Issues and challenges which led to the development of the final workflow will be discussed in the “Results” section.

*Modules*

Several modules were imported for the creation of this tool:

* Arcpy – for the use of geoprocessing tools and environment settings
* Arcpy.sa – the Spatial Analyst extension for the ISO Unsupervised Classification and Map Algebra
* OS – for any file handling procedures (this import turned out to be unnecessary

.

*Environment*

Environment settings were of the utmost importance, for reasons that will be discussed later. Ultimately, it proved necessary to design the tool to work within the file geodatabase structure provided by Arcmap. Hence, arcpy’s env.workspace settings were set to a static geodatabase for testing purposes. Further development of the script will employ the designation of unique file geodatabases within the script. Also, since the script is designed to be used as a repetitive workflow, it was important to allow overwriting of existing data. For this reason, the env.overwriteOutput setting was set to “True.”

*Inputs*

The initial inputs for the script were static to make for easier testing. The inputs include an AOI file, for clipping the rasters to be compared to the same extent. This is a polygon file provided by the user. The most recent raw image to be compared is saved into a variable called “new\_image.” The older of the two images is saved into a variable called “old\_image.” There are also string inputs for the date of the new and old images (“new\_date” and “old\_date,” respectively). The date inputs are used for file naming conventions in the geoprocessing phase. Finally, the user must input the parameters that will be used in the unsupervised classification. These parameters include: the number of classes (num\_class), the minimum number of pixels per class (min\_class\_size), and the sample interval to be used in the classification (sample\_int).

*Clipping*

The clipping process is handled by arcpy’s data management Clip tool. Since more than one image is to be clipped, this portion of the script is written as a function to be called multiple times. The AOI, raster to be clipped, and image date are passed to the function (“clip\_it”) as arguments. The AOI file defines the clip extent, while the image date is passed to be used as part of the output filename. This function includes error handlers in the form of the “try” and “except” statements to help prevent problems with input. This function is called twice—once for each image.

*Classification*

The ISO Cluster Unsupervised Classification tool from the arcpy.sa extension is used to classify the clipped images. Like the clipping procedure, this tool must be called twice, and so it is written as a function (classify\_it). The raster to be classified, the date of the image, and the tool parameters (number of classes, minimum class size, and sample interval) are passed as arguments to the function. Again, the image date is used as part of the filename. This function includes error handlers in the form of the “try” and “except” statements to help prevent problems with input.

*Map Algebra*

The final stage of the workflow makes use of the arcpy.sa Map Algebra operators. These operators allow for the comparison of spatially registered raster objects. In this case, the Boolean operator “==” is used to compare the two classified images. Using this operator, the Map Algebra tool output as raster of ones and zeros (True or False), to indicate whether a classified pixel has changed from its value in the old image to the new image. Unfortunately, this method only indicates change. It does not indicate what kind of change has taken place.

*Conversion to Tool*

The conversion of this script to an Arcmap tool is relatively straightforward. Each of the input parameters were changed from static values to user input using the arcpy.GetParameterAsText method. Each input parameter is numbered sequentially in the manner it will appear in the ArcMap tool dialogue box. A toolbox is then created in Arcmap; the script is indicated in the tool settings, and dialogue box inputs are set appropriately to receive strings for parameters or geodatabase feature classes and raster datasets, as appropriate.

Results

The final results of this workflow are promising, though the tool requires further development to be of any practical use. The required file outputs are created: two clipped images, two classified images, and a change raster. However, the most difficult challenge that presented itself was a conflict between the ISO Cluster Unsupervised Classification tools requirements and the workspace settings within the script. Initially, the script environment was not using a file geodatabase. The environment was set to a regular OS working folder. After the initial debugging phase of development, the script ran into a recurring problem whereby the script would continually fail during the classification procedure. The error was recorded by the exception handler as follows: “ERROR 999999: Error executing function. Failed to open raster dataset Failed to execute (MLClassify).” After conferring with more experienced professionals on GIS Stackexchange, I learned that an ESRI raster file can only have nine characters.[[4]](#footnote-4) However, the ISO Cluster tool creates multiple, temporary “isocluster” raster during the processing of the image. The “isocluster” filename exceeds the nine character limit set forth. Hence, the only legal way create the filenames is within a file geodatabase. The script was amended to create a file geodatabase as part of the process. However, this created additional problems with the workspace.

The new logic for this script was to access the raw imagery from the OS workspace. Create a geodatabase, and then, via a path definition, direct all subsequent outputs to the geodatabase, but this solution still created a problem for the ISO cluster tool. Since the workspace is still defined as the OS directory, the “isocluster” rasters were still created there, even though the final output would be directed to the newly created geodatabase. However, if the geodatabase was set as the work environment, it made accessing the files within the OS directory more difficult. With time being of the essence, the problem was overcome by manually creating a file geodatabase, and then importing the raw imagery before running the script. In this manner, the workspace could be set to the geodatabase and all further work could be processed there. In a subsequent revision, it seems that either the workspace could be changed after the geodatabase is created or the raw imagery could be copied to the geodatabase after its creation.

Conclusion

Ultimately, this project illustrates a promising “proof of principle” for this workflow. However, the logic needs work to become practical. Moreover, it would better if the tool could not only identify change but the type of change that has taken place. With some minor modifications, this project could significantly improve efficiency in change detection processes. The benefits are numerous. It could decrease labor costs. It could decrease the necessity to purchase more expensive software, outside of the ESRI suite. Additional research into the Spatial Analyst toolset could provide solutions to these shortcomings by use of reclassification methods and/or more rigorous Map Algebra operations.

References

Dees, John. "Iso Cluster tool in script failing to open clip raster." python. http://gis.stackexchange.com/questions/108832/iso-cluster-tool-in-script-failing-to-open-clip-raster (accessed July 29, 2014).

ESRI. "ArcGIS Desktop – An overview of the Map Algebra toolset." ArcGIS Desktop. http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//009z000000z6000000.htm (accessed July 31, 2014).

ESRI. "ArcGIS Desktop - Iso Cluster Unsupervised Classification (Spatial Analyst)." ArcGIS Desktop. http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//009z000000pn000000.htm (accessed July 31, 2014).

Ramachandra, T.V., and Uttam Kumar. "Geographic Resources Decision Support System for land use, land cover dynamics analysis." http://www.researchgate.net%2Fpublication%2F257084788\_GRDSS\_for\_Land\_use\_land\_cover\_analysis (accessed July 28, 2014).

1. See Ramachandra, T.V., and Uttam Kumar, "Geographic Resources Decision Support System for land use, land cover dynamics analysis," http://www.researchgate.net%2Fpublication%2F257084788\_GRDSS\_for\_Land\_use\_land\_cover\_analysis (accessed July 28, 2014). [↑](#footnote-ref-1)
2. ESRI, "ArcGIS Desktop - Iso Cluster Unsupervised Classification (Spatial Analyst)," ArcGIS Desktop, http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//009z000000pn000000.htm (accessed July 31, 2014). [↑](#footnote-ref-2)
3. ESRI. "ArcGIS Desktop – An overview of the Map Algebra toolset." ArcGIS Desktop. http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//009z000000z6000000.htm (accessed July 31, 2014). [↑](#footnote-ref-3)
4. John Dees. "Iso Cluster tool in script failing to open clip raster." python. http://gis.stackexchange.com/questions/108832/iso-cluster-tool-in-script-failing-to-open-clip-raster (accessed July 29, 2014). [↑](#footnote-ref-4)