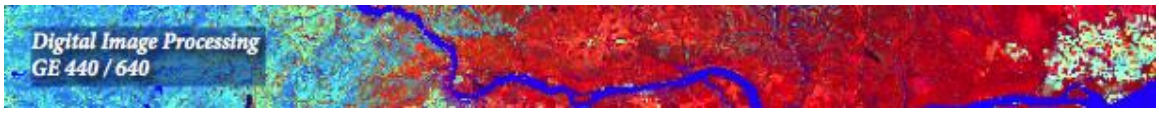


LAB: IMAGE SEGMENTATION

AUTHOR: Chris Holden (ceholden@bu.edu)

INTRODUCTION:

Image segmentation attempts to change the base unit of analysis in an image from a pixel to a collection of pixels representing continuous objects. Many applications, especially high spatial resolution remote sensing, are better performed on objects, segments, or polygons rather than pixels. When used for classification, segmentation can not only "clean-up" the final map by grouping pixels together, but can also provide additional contextual information about the size, elongation, "roundness" and "squareness", and variance of each object while dampening noisy high-frequency variation.

GOALS:

- Produce a segmentation image using software from Woodcock & Harward (1992)
- Combine the segmentation image with your classification to produce a segmented classification
- Gain insight into the segmentation parameters by comparing the output of various different parameter choices

TERMINOLOGY / NOTES:

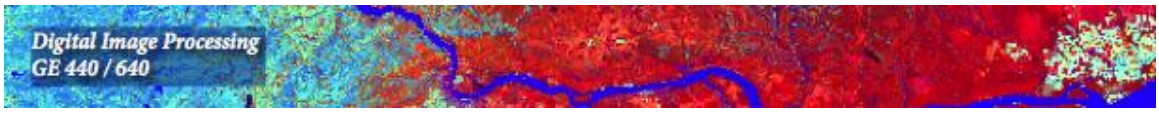
BIP	Image in Band Interleave by Pixel format
BSQ	Image in Band Sequential format
IPW	Image Processing Workbench library developed by James Frew in 1990

As a convention, I have formatted any command line input or output in bold font with one inch of indentation. You can distinguish command line input from output by noting that all input will being with ">", similar to the default C-shell on GEO.

PREPARATION:

In order to use the segmentation tools described below, we must first tell the computer where to look for the program files. While in a QSH session on the GEO/KATANA cluster, enter the following commands into your C-shell:

```
> setenv IPW /net/usr/local/ipw
> source $IPW/pub/cshrc
```



The first line creates an "environmental variable" named **IPW** that points to the installed location of the IPW library. The second line runs a file that adds the various segmentation programs to your PATH variable. On Unix/Linux systems, your PATH variable tells the computer where to look for programs you enter at the command line.

To permanently add the segmentation programs to your PATH, edit your ".cshrc" file (for C-shell, different for Bash and others) located inside your home directory. You may add these lines anywhere in the file, but it is a good idea to add them into their own section with comments describing what the commands do.

You may edit your "~/.cshrc" file from GEO using a terminal-based text editor such as "vim" or "emacs" or using a graphical editor such as "nedit". Alternatively, you can connect to GEO using a secure FTP connection and edit the file with a text editor installed on your computer. As an example, while in a QSH session, you could enter:

> nedit ~/.cshrc

The changes to your ".cshrc" file will be applied for all future logins. To apply the changes to your ".cshrc" file during your current session, you can source the file again:

> source ~/.cshrc

STEPS:

1. Locate your Landsat (or other) image on the GEO/KATANA cluster

Log into GEO, change directory to the location of your imagery and start a QSH interactive session. Please refer to the "Method" material provided in previous labs if you have any questions.

2. Choose a set of parameter inputs for the segmentation

The segmentation program has three main parameter types:

- Tolerance specifies the maximum Euclidian spectral distance for merging two regions. If two regions are above this threshold, they will not merge; below they may merge.
- Merge rate specifies the speed at which the algorithm increases the tolerance threshold. To ensure that regions are merged with their best match, the tolerance on the first pass begins close to zero and increases with every iteration until it reaches the maximum specified tolerance.
- Region size restrictions:
    - Nabsmin - Absolute minimum region size
        - Regions will not contain fewer than this number of pixels
    - Nnormin - Normal minimum region size

        - Algorithm will merge regions with fewer pixels than this number on the auxiliary pass, regardless of spectral distance
    - Nviable - Viable number of pixels for a region
        - Two regions are not allowed to merge if they contain this "viable" number of pixels, regardless of spectral proximity
    - Nmax - Maximum number of pixels for a region
        - Regions will not contain more than this number of pixels
    - Nabsmax - Maximum number of pixels for a region
        - Regions will not contain more than this number of pixels, even in the auxiliary passes

Don't worry about understanding the full meaning of the absolute region size restrictions. In practice, the minimum and absolute values are usually set to the same value.

A good set of parameters to start off using are the following:

Tolerance: 10
Merge: 0.1
Nabsmin: 11
Nnormin: 11 (11 Landsat pixels is about 1 hectare)
Nviable: 100
Nmax: 65535
Nabsmax: 65535

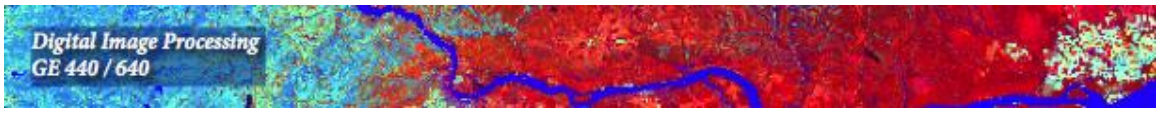
To get a better understanding of what each parameter does, please read the paper by Woodcock & Harward (1992) and read the descriptions located in the segmentation code. I have provided the paper as a scanned PDF (sorry, no digital copy exists) and the program comments as a text file accompanying this lab.

3. Run the segmentation for your chosen parameters

Once you've selected a set of parameters, you must store them in a text file for use in the segmentation program. In addition, each set of parameters must be specified in the style of command line options. Using the parameters shown above, this would mean:

-t 10 -m 0.1 -n 11,11,100,65535,65535

The "-t", "-m", and "-n" are ways of specifying command line options. When we write "-t", the program expects an integer value to follow and will use it for the tolerance threshold. The "-m" option, for merge coefficient, expects a real number between 0 and 1. The "-n" option requires a sequence of 5 integers for the region size restrictions.



Use a text editor to create a file named "parameter.txt" and copy the above example into the first line of the text file.

You may check if the text file was written correctly by printing its contents to the terminal using the "cat" program:

> cat param.txt
-t 10 -m .1 -n 11,11,100,65535,65535

Next, we will use a script named "batch_segment.sh" to perform the image segmentation. It should be in your PATH if you've correctly completed the steps in the preparations section.

To access the command line options for this script, and many other command line programs, you can type:

> batch_segment.sh -h
usage: /net/usr/local/ipw/contrib/bin/batch_segment.sh options

This script will read in text file of parameter values for
segment program and run segmentation for -i image for all options within
-p parameter file.

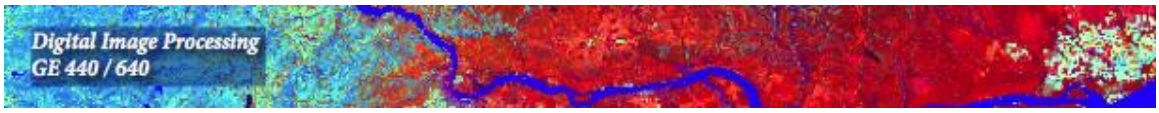
OPTIONS:
-i an ENVI format BSQ image
-p parameter file
-s produce shapefile of regions

The program needs us to specify two options to work correctly. The first, "-i", is the location of an ENVI formatted BSQ image. Find the location of your imagery and write down the location to use in this script. The second option, "-p", is the location of the "parameter.txt" file you just recently wrote. To run the segmentation, run the following command in a QSH session:

As an example for an image named "399_2009-08-12_qb_mss_orth", I entered:

> batch_segment.sh -i 399_2009-08-12_qb_mss_orth -p parameter.txt

/net/casrs1/volumes/cas/glcv/scratch/ceholden/399
Using existing IPW format of image name provided...
Tolerance: 10
Merge: .1
N: 11,11,100,65535,65535
m: 1
n: 11_11_100_65535_65535



t10-m1-n11_11_100_65535_65535
Input image has 4 bands, 7186 samples, and 7266 lines
The segmentation tolerance is 10.000000
The merge coefficient is 0.100000

After entering the "batch_segment.sh -i [YOUR_IMAGE] -p [PARAMETER_FILE]" command, you will see the script show the parameters you chose as well as some information about the image you've selected.

The segmentation program will not run through many iterations, or passes, performing the segmentation. An example output during the process is:

Pass 1 completed
Tolerance for pass was 0.316, (Tg = 10.000)
39525237 regions remain after this pass
The minimum nearest neighbor distance on this pass was 0.000
The largest region generated on this pass contained 4 pixels
Merges: attempted=39544615
nnbr_gone=2389496
wrong_partner=0
nnbr_d2_big=32436857
both_viable=0
npix_big=0
merging=4718262
Wrote log band for pass 1

Wait until the segmentation has completed before proceeding to the next step. In the future, you may wish to redirect this "standard out" output to a log file using the ">" operator.

The segmentation script will create a folder named according to the choice of parameters you've used. In the example above, the script created a folder named "t10-m1-n11_11_100_65535_65535". This folder location will be used to store the outputs of the segmentation program. For now, we will only use the "regionmap" image and its "regionmap.hdr" ENVI header file.

4. View the segmentation results in ENVI

When the segmentation has completed, you may wish to open the "region map" image in ENVI. The script you use for segmentation will output several files, including one called an ENVI formatted image named "regionmap".

Open ENVI and load this image. One complicating factor for viewing it is that, due to the very large number of segments in the image, it will be difficult to view more than a

small area at one time. When you first open the image, the scroll window will look very much like a gradient of back to white. ENVI displays your region map image this way because the top of the image contains the first segments with small corresponding region IDs (stretched to 0). Toward the bottom, the IDs are very large and thus are shown as white (stretched to 255).

In order to view the shape of the segments, you can re-stretch the image. I recommend using either the "Linear 2%" stretch in the Image or Zoom window. It is also helpful to link the region map image display to your original image.

5. Overlay your segmentation onto a classified image

Now that we have seen a "region map" image in ENVI, we can use a second program to apply the segmentation to a classified image. First, we must identify one more of the output files.

Look inside the folder generated by "batch_segment.sh". The folder will be named according to the parameters you used. In this example, the folder was named "t10-m1-n11_11_100_65535_65535". Inside, you will find five files not previously discussed:

> ls -l t10-m1-n11_11_100_65535_65535/
myseg.log.###
myseg.rmap.###
myseg.alog.###
myseg.armap.###
mysegment.log

The two files with "###" denotes that this file may have a number at the end of the filename. This number indicates the number of stages the segmentation performed during each stage.

The important files to remember are "myseg.armap.###" which contains the final stage (the auxiliary stage) region map image and the "mysegment.log" file which contains the log file for the segmentation output. In this step, we will use "myseg.armap", but it may be useful to refer to "mysegment.log".

In step 3, we created a segmentation map. In step 5, we will use another shell script named "apply_segment.sh" to apply the segmentation to a classified image. The usage for this script is as follows:

> apply_segment.sh -h
usage: apply_segment.sh options

This script will overlay segments onto a pixel based classified image to

produce an object based classification. The input pixel based classified image will be used to determine the output class of each segment as determined by a plurality rule.

OPTIONS:
-c input pixel based classified image in ENVI format
-s input segmentation in IPW format (*.armap.* or *.rmap.*)

This script will look at each segment in the region image and assign all pixels in the segment the land cover class with the highest frequency within the segment. So, the output of this script will be a "segmented" classified image that will have one class for each segment.

As an example, if we wrote:

> apply_segment.sh -s t10-m1-n11_11_100_65535_65535/myseg.armap.85 -c my_classified_image

the script would produce a new "segmented" classified image that will be saved in the same directory as your classification image. The filename will be a combination of the name of your original classification and the segmentation region map.

You may open this in ENVI. It will inherit any of the class labels and colors your original "pixel-by-pixel" classification used.

6. Repeat steps 2-5 varying the parameters to understand their effects

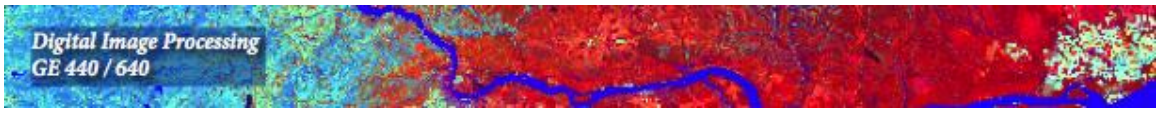
Now that you have produced one segmentation image and segmented classification image, you may wish to vary some of the segmentation parameters to produce a better map and further your understanding of each parameter's function.

To start, I would suggest varying the region minimum size restrictions as these are the most influential in creating different segmentations.

Does the segmentation improve your classification? Are there types of land cover that benefit from the segmentation more than others? What is the relationship between the magnitude of the minimum region size and the amount of "salt and pepper" high frequency variation remaining in your classification after segmentation?

REFERENCES:

Woodcock, C and VJ Harward. 1992. Nested-hierarchical scene models and image segmentation. International Journal of Remote Sensing 13: 3167-3187.



QUESTIONS:

Please refer all questions to Chris Holden (ceholden@bu.edu).

EXTRA:

- To make life easy for me and anyone else who uses this segmentation program, I have written two shell scripts that act as a giant user-friendly (I hope) wrapper around the original IPW programs. The script takes care of things like file format conversions, creating IPW headers, changing data types, and running through the many steps involved in creating a segmentation using the original algorithm. It may be worth your time to browse the source code for "batch_segment.sh" and "apply_segment.sh" to understand the steps taken by each script.

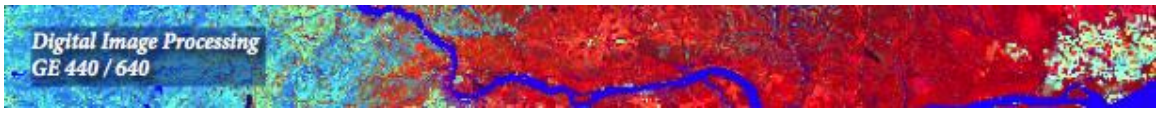
- If you analyze the segmentation log file or the console output during segmentation, you will notice that on each of the two passes (the normal pass and the auxiliary pass), the algorithm increases the tolerance threshold each iteration. Additionally, the algorithm will allow for a region to be merged once during an iteration. The purpose of this is to help ensure that each region is paired up with regions closest to it first. For example, if a region could merge with two of its neighbors, we would want to merge the region with its closest neighbor first. After this first merge, we will recalculate the distance (Euclidian in this case) between the newly merged region and others. This might result in the newly merged region not being capable of merging with neighbors it could before.

- The implementation of the algorithm you are using only looks at "4-way" neighbors instead of a full "8-way" neighborhood. What does this difference signify? Can you describe how the segmentation might be different using "8-way" neighborhood approach rather than "4-way"?

- The IPW library uses a very simple method of storing images. The "IPW format" is simply a binary image that is preceded by a simple header describing the number of lines, samples, and bands. To save space, IPW images are saved only using the maximum number of bytes required to store the image. For region maps used in segmentation, this maximum number is the number of regions in the image. How many bytes are used to store the following?:

  65530 regions requires _____ bits or _____ bytes
  88536 regions requires _____ bits or _____ bytes
  8378609 regions requires _____ bits or _____ bytes

  Some of these answers are common data types for modern software programs like ENVI. For example, an image storing 8 bits, or one byte, of data (2^8 possible values



0-255) is a Byte image while an image using 16 bits could be a signed or unsigned integer. To be able to read IPW images with non-standard data types (i.e. a 23 bit image), we must tack on additional unnecessary bits of information to meet the next standard data type. 13 bit images go to 16 bit signed or unsigned integers, 23 bit images go to a 32 bit signed or unsigned long integer.

Most of this information is useful for general knowledge, but is specifically applicable to this segmentation lab because the segmentation program only is capable of processing 8 bit imagery! Most high spatial resolution imagery (Quickbird, IKONOS, WorldView-2, GeoEye, etc.) use sensors that have a radiometric resolution of 11 bits per pixel. Up until the launch of Landsat 8, scheduled for February 11th, 2013, Landsat has used 8 bits per pixel while Landsat 8 will use 12 bits per pixel! By using more bits per pixel, a detector is capable of digitizing more precisely the analog radiation signal. By "quantizing" the signal using more bits, there is finer radiometric resolution and more capability of distinguishing between similar signals. So, unless this segmentation program is updated from its 1990 version, we will not be able to take advantage of advances in radiometric resolution.