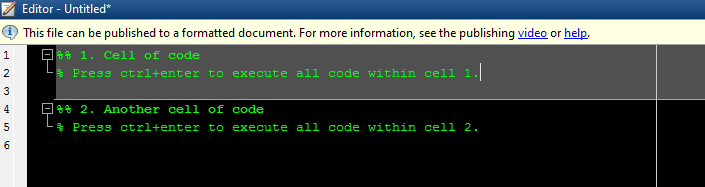
# RivMAP Demo Walkthrough

This demo walks through all the tools included in RivMAP to quantify planform changes. Descriptions of the tools are found in the accompanying paper (IN REVIEW). Channel masks are provided for a [reach of the Ucayali River](https://earthengine.google.com/timelapse/#v=-9.5,-74.13468,9.148,latLng&t=2.73) from 1985-2015. The results presented here are examples of applications of RivMAP and not intended to be comprehensive—many of the results computed in the demo are not plotted. RivMAP currently contains no functions for plotting, but the DEMO.m contains codes and hints for creating different types of maps and plots.

This document is meant to be referenced in tandem as you work through the RivMAP demo. The demo was constructed so even users with minimal Matlab experience should be able to execute it easily. You can begin by opening the DEMO.m file. There is no GUI; the demo is executed entirely from the Matlab Editor. The demo script is divided into “cells” that are delineated by two percent signs (“%%”). See the figure below (your color scheme may be different).



The entire code within a cell may be run by pressing “Ctrl+Enter” while the cursor is in the cell. Alternatively, you can right-click and select “Evaluate Current Selection.”

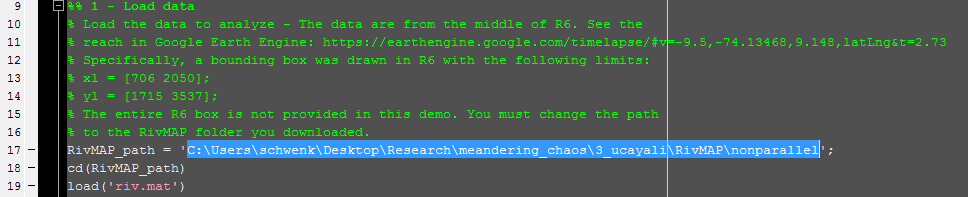
Each cell in the demo is numbered and corresponds to a section in this document. You do not need to reference this document as you work through the demo, as comments within each cell describe the analysis being performed. This document does contain additional notes and discussion not found in the demo script.

A note on parallelization: all parallel capabilities have been disabled in the released version of RivMAP. In many cases, parallelization adds runtime rather than reducing it, especially when runtimes are already quite small. However, you can easily parallelize a few functions by inserting *parfor* in place of *for* where desired. Matlab’s handling of parallel processing is quite memory inefficient; generally, you should parallelize loops at the lowest level possible to avoid large variables being passed around and copied. Three functions can be parallelized easily and effectively:  
 1) *width\_from\_banklines* - the only *for* loop  
 2) *banklines\_from\_mask* – the *for lr =…* loop  
 3) *spatial\_changes* - the *for ii=*… loop

Again, parallelization does not always improve runtimes (especially for the small sample dataset provided with this demo), so use at your own discretion.

# Load data

Important: This is the only portion of the code that you should need to modify. In line 17, you will need to set the path of the RivMAP folder you downloaded (highlighted below). Replace everything within the apostrophes with the path to the RivMAP folder.



The data are from the R6 bounding box defined in the accompanying paper (in review). You can see the time-lapse of this reach using [Google’s Earth Engine](https://earthengine.google.com/timelapse/#v=-9.5,-74.13468,9.148,latLng&t=2.73). The full R6 box is not provided; only the portion within a pre-defined bounding box is contained in the *riv* structure.

Data are stored in a Matlab structure variable called *riv*. Once loaded, you can explore the *riv* structure by opening it in the workspace. It currently contains two substructres: *riv.meta* and *riv.im*. In *riv.meta*, you will find metadata including year of imagery, exit sides of the image, and a nominal channel width. Normally georeferencing information is also stored in *riv.meta,* but it is not necessary for this demo. The *riv.im* substructure contains two channel masks: the hydraulically connected mask (*riv.im.hc*) and the single-thread mask (*riv.im.st*). You may notice that the *riv* structure contains 32 entries. Each entry corresponds to a year (1984-2015), and you can access a particular year by typing (e.g. for the single-thread mask in 1986) ‘riv(3).im.st.’ This type of data storing/referencing is used throughout the demo, and later on new substructures will be created to store results.

The variable *exit\_sides* contains two ‘NSEW’ characters that denote the sides of the image that the channel mask enters and leaves, respectively. I.e., the first character is the upstream exit side and the second is the downstream. This variable is important for many of the image processing techniques used by RivMAP and helps to arrange results in an upstream->downstream manner and differentiate between left and right banks.

You should see outputs of the exit sides, year, and nominal channel width for i=1 after running this cell.

# Plot channel masks

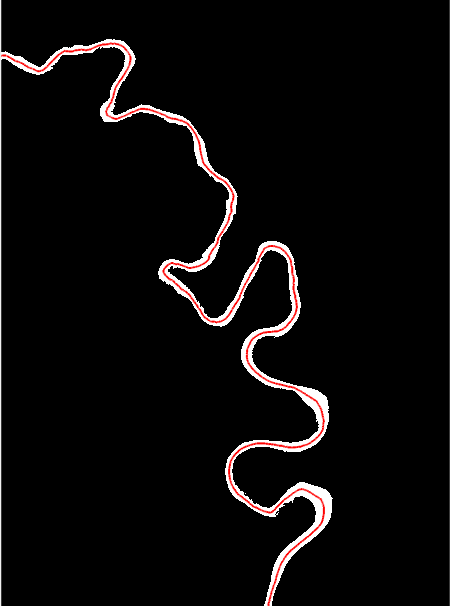
Let’s take a look at the channel masks evolving through time. The single-thread channel mask we will analyze is shown; you can also see the hydraulically-connected mask by replacing line 45 with ‘I = riv(i).im.hc’. A “video” should play when you run this cell.

# Analyze one year of planform characteristics

Before we analyze all the data, we will step through the processing of a single year to understand some of RivMAP’s functionality. We will look at 1984 (i=1). A variable called *plotornot* is set to 1 so results will be plotted. We also load the metadata necessary to analyze. There is no output when you run this cell.

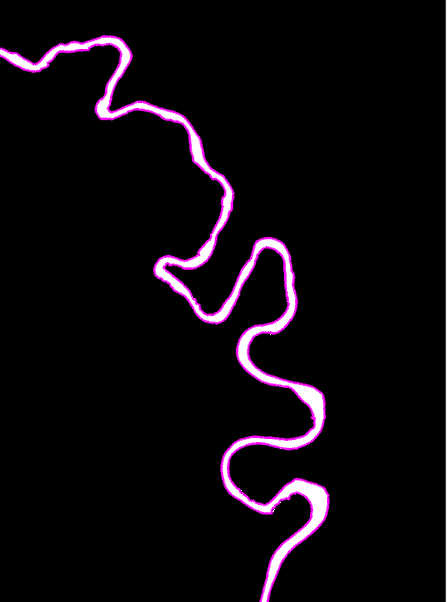
# 3a. Compute the centerline

The centerline is computed as described in the accompanying paper. We are analyzing the single-thread channel mask (*riv.im.Ist*). You should see the following output. The centerline is shown in red.



# 3b. Compute the banklines

You should see the following output after running this cell. Magenta lines are banklines, and the banks are returned as a two-element cell called *banks*. The first element contains the left bank; the second contains the right.



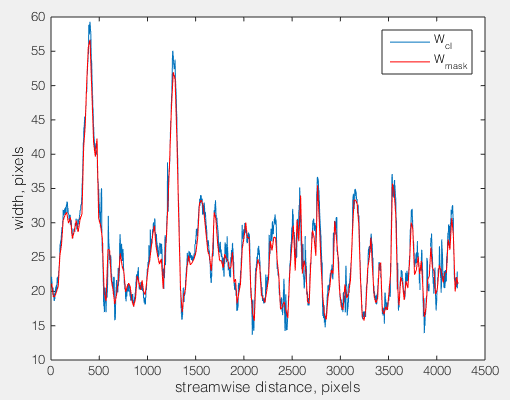
# 3c. Along-channel widths

Two methods for computing width are demonstrated. The first computes width at each pixel by intersecting a perpendicular vector with the banklines. This function, *width\_from\_banklines* is the most time-consuming of all RivMAP functions and can employ parallelization using Matlab’s *parfor* command.

The second method for quantifying along-channel width computes average width along the channel mask by creating polygon buffers at a given input interval. The channel mask area is found in each polygon buffer, and the width of the segment is found by dividing this area by the centerline length within the buffer.

To compare the methods, the along-stream distance (*S*) is computed along the centerline as a simple sum-of-squares between centerline nodes.

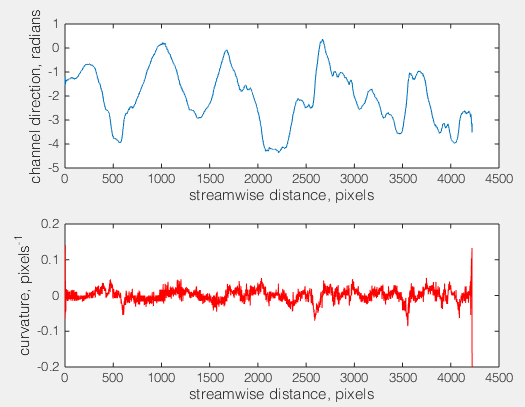
You should see the following graph comparing the two widths after running this cell.



# 3d. Angles and curvatures

Many of the RivMAP algorithms that require along-stream parameterization only function well with a smooth centerline. A Savitzkty-Golay filter is used to smooth the centerline coordinates, which are stored in the *cls* variable. Channel directions (angles) and curvatures are computed on the smooth centerline.

You should see the following output after running this cell.



You may notice the excess noise in the curvature signal in the lower panel. This is a common problem when computing second derivatives digitally. (Curvature is essentially a second difference of channel position.) This noise can be reduced by (1) further smoothing the centerline before computing curvature and/or (2) smoothing the curvature series itself. Note that in both these cases, you will alter the statistical properties of the curvature and care should be taken before interpreting results. Details of smoothing curvature are beyond the scope of this demo. You may also note that the ends of both signals are noisier than their middles; this is due to edge effects where the window size of the smoothing filter must shrink due to the signal’s boundaries.

# 3e. Reach-averaged widths

We can compute the reach average width two ways. First, we can divide the channel mask area by the centerline length (*Wra*). Second, we can average the individual width values at each pixel along the centerline (*Wavg*).

You should see that values for both of these (*Wra*=25.1 pixels and *Wavg*=25.8 pixels) agree fairly well when you run this cell.

# Process all years

Now that we have demonstrated the planform characteristics computed by RivMAP, we will process all 32 years of data. Plotting is turned off. A new substructure is created, called *riv.vec*, that stores all our vector-based data (centerline coordinates, widths, angles, curvatures). Smooth versions of the centerline and banklines are also stored, and we add a new image to the *riv.im* substructure called *Icl*, which is a binary representation of the centerline.

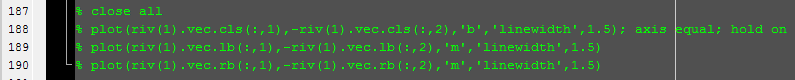
As noted above, this section may take awhile to compute due to the *width\_from\_banklines* routine. With parallelization enabled in *width\_from\_banklines* using four cores, processing all 32 years takes about 2-3 minutes. Without it, this section may take 10-15 minutes to run. The entire section may itself be parallelized by processing each year independently, but you would need to make some changes. The *riv* variable is not classifiable by Matlab as-is, and hence a wrapper function would need to be written. Even for the full study reach, processing times were short enough that I never wrote this wrapper.

As this section runs, it will update you with its progress. When it is finished, it will save the *riv* structure with the added results. If you want to pause the demo here and return later, you do not need to re-process the *riv* structure as it has been saved to your disk. Simply load it in and continue.

# Plot all years

In order to ensure RivMAP correctly computed centerlines and banklines, we plot them on the channel mask for each year. When you run this cell, you should see a “video” of results.

Note that if you would like to plot banklines and/or centerlines without plotting on top of an image, you need to multiply the y-coordinates by -1. This is because Matlab sets the origin for images at the top-left of the image, while for ordinary plotting the origin is at the bottom-left. You can uncomment and run the following lines to test for yourself.



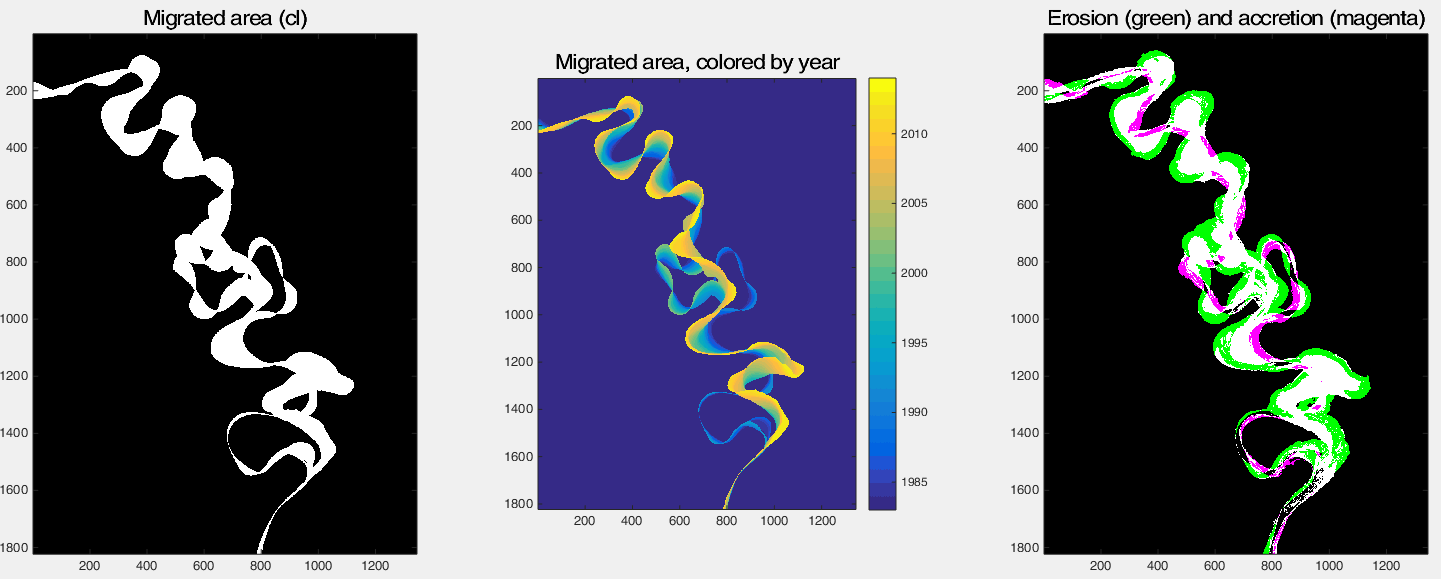
# Compute migrations

In this section, we compute centerline-migrated areas, cutoffs, erosional areas and depositional areas. A new substructure called *riv.mig* is appended to the *riv* structure to store these results. Within this new substructure are two sub-substructres called *riv.mig.cl* which stores centerline-based migration areas and *riv.mig.mask* which stores mask-differenced migration areas (erosion and accretion). Quantifying change requires two realizations, so we only loop from i=1 through 31 (total number of years minus one).

The progress should be displayed when you run this cell. You will notice that it is substantially faster than the processing step. Parallelization of this section is possible but unnecessary due to the fast runtimes. After all years have been analyzed, the *riv* structure is again saved with the added results.

# Plot migration maps

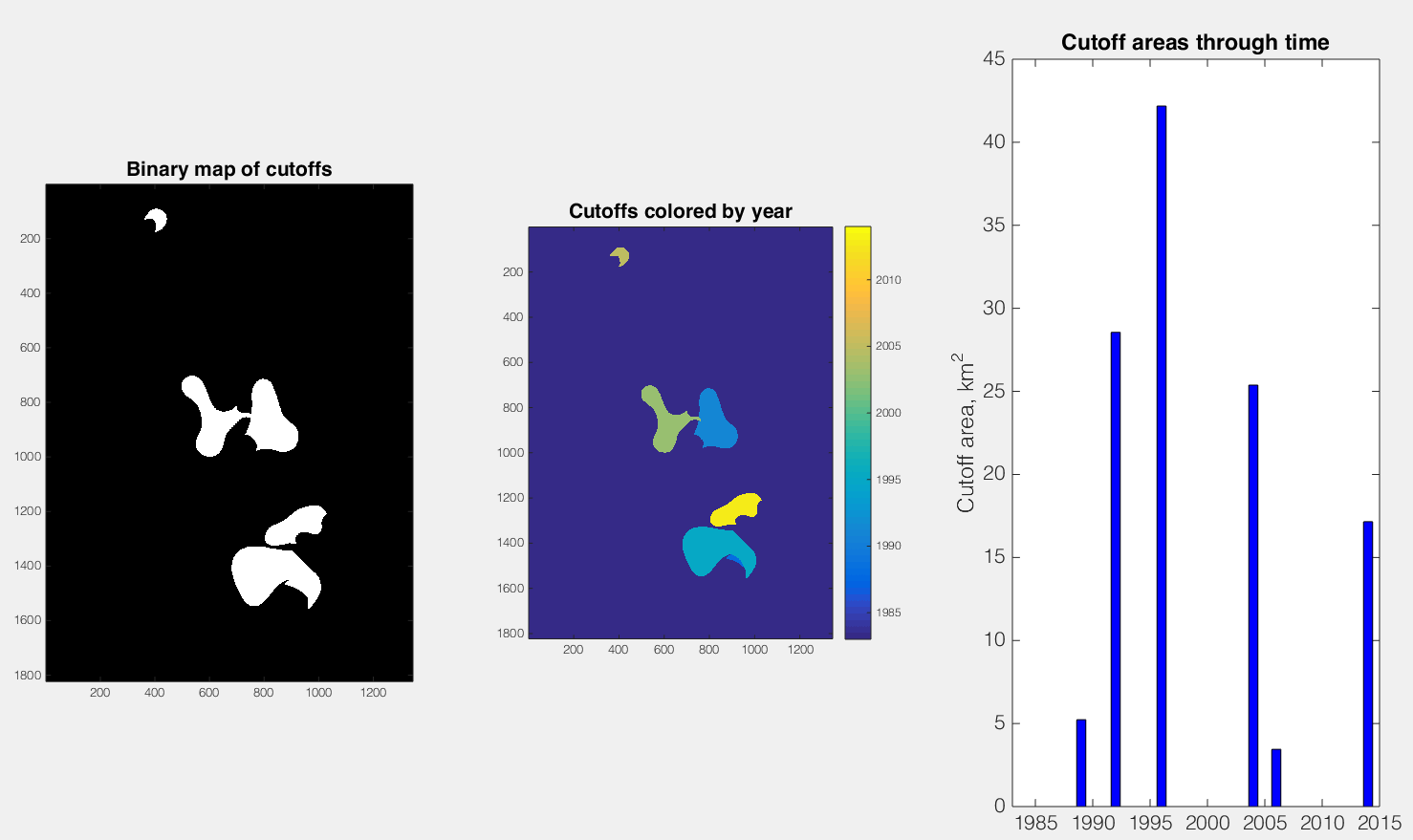
A benefit of using image processing is the rapid creation of migration maps. In this cell, migration maps for centerline migrated, eroded, and accreted areas are generated and displayed. Each map shows the total areas migrated over the 32 year time span. The images are created by initializing a blank binary image. We loop through each year and populate the empty image with the migrated pixels from each year. Running this cell should produce the figure below.



We can quickly see where reaches have migrated substantially or minimally. We also see that cutoffs have occurred, and we will plot those next. In the third panel, it appears as if more erosion (green) has occurred than accretion (magenta) during the period, and we will explore this later as well.

# Plot cutoffs

We can examine the cutoffs that occurred throughout the period, as well as plot the time series of cutoff areas. Running this cell should generate the figure below.

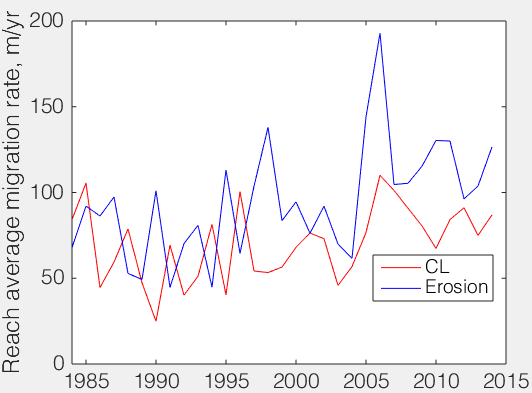


In the first panel, a binary map of cutoffs is shown. It appears that five cutoffs have occurred. Coloring the cutoffs by year (second panel) suggests that six cutoffs have occurred. Finally, the time series of cutoff areas (third panel) shows that in fact seven cutoffs have occurred—in the cutoff images, a small cutoff is being hidden by a larger one that occurred at a later time.

# Annual reachwide migration rates

Up to this point, we have looked at migrated areas for the reach. We may also be interested in migration rates, defined as the migrated area divided by the centerline length (divided by the time elapsed between images = 1 year). These rates for centerline migration, erosion, and accretion are computed in this cell, but there is no output.

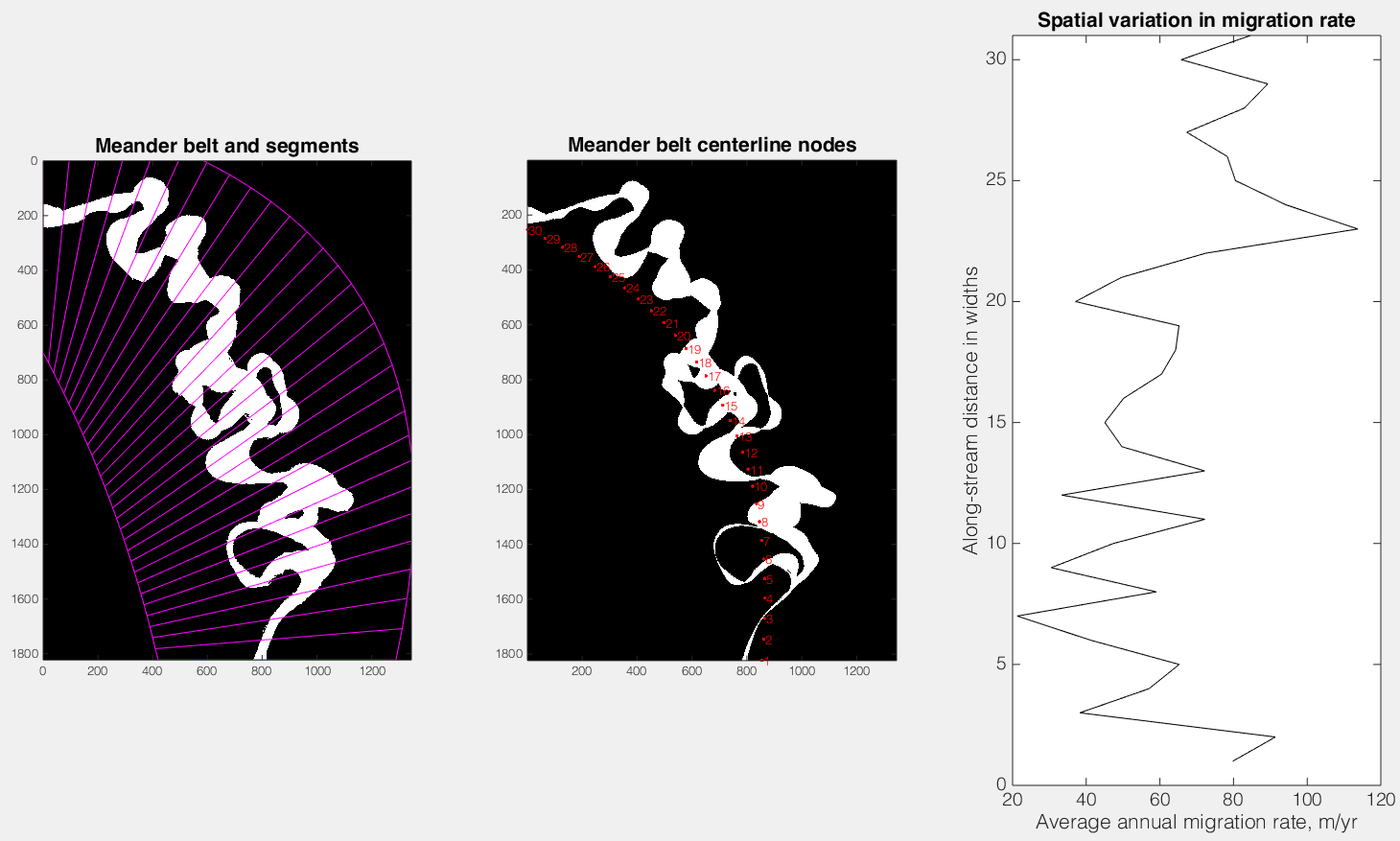
9a. Reachwide migration rates through time  
In this cell, we plot the migration rate for the entire reach for each year. Migration rate can be defined based on centerline migration or erosion; both are plotted. The output from running this cell is shown below.



The plot indicates that migration rates from centerline migration and erosion are similar until 1995, when the erosion migration rate becomes larger than the centerline migration rate. Both rates seem to increase beginning in 1995. We saw in section 8 that the largest cutoff (by area) occurred in 1995…

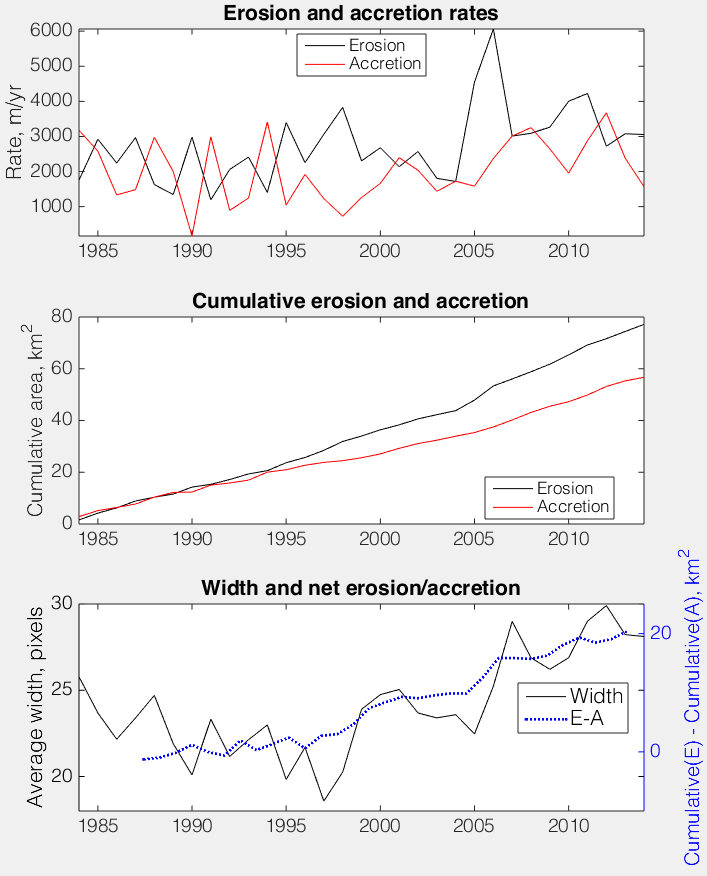
9b. Spatial variation of migration rates  
We can also look at how migration rates change spatially by defining a meander-belt centerline (similar to a valley-line) and averaging migrations through time within segments along this centerline. The RivMAP function *spatial\_migration* is called to compute migration rates within segments along the meander-belt centerline. We must input the image of all channel positions and the image migrated areas for all years that we computed in cell 7. The spacing that sets the length of each segment must also be defined. We will compute migration rates every 2W, although you can play with this spacing. This function may take a while to run, especially if not run in parallel, but remaining time estimates are provided.

The results from running this cell are shown below.



In the first panel, the image is of all channel positions through time (1984-2015) is shown with the segments overlain. The second panel shows all the migrated areas over the time period with the centerline segments numbered (each segment is 2W long with respect to the meander belt centerline). The numbers in the second panel correspond to those on the y-axis in the third panel, which shows the spatial change in average migration rates over 32 years. The reach underwent significant during the 32 years, except near its upstream end. You can try finer spacings to better-resolve the spatial signal of migration rates. Smaller spacings will result in more variable migration rates and longer runtimes.

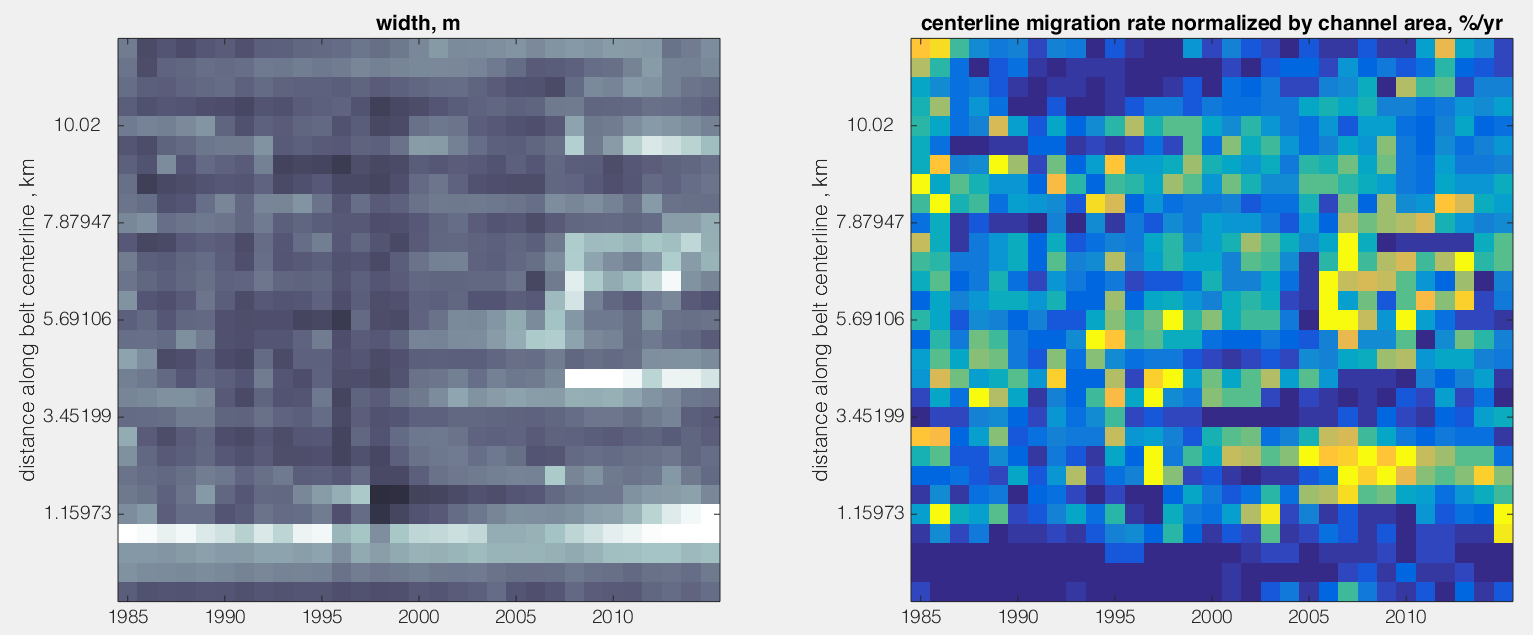
9c. Erosion and accretion rates  
Here we look at the annual balance between erosion and accretion. Using the maps we’ve already created, we can easily quantify how erosion and accretion change in time throughout the reach. Running this cell results in three plots shown below.



In the top panel, the erosion and accretion rates are shown through time for the whole reach. They appear to begin diverging around 1995. It is easier to compare their balance by plotting their cumulative areas over time, which is shown in the middle panel. By the end of 2014, almost 100 km2 more erosion occurred compared to accretion, with erosion overtaking accretion beginning in 1995. Greater erosion than accretion indicates channel widening, so we should see the reach widen beginning around 1995. The bottom panel confirms that the reach widened substantially (roughly 250 meters!) from 1995-2015, in tandem with larger erosion than accretion areas.

# Spacetime maps

Planform changes for the entire reach may quickly be observed via a spacetime map, which is simply an image where the x-axis is time, and the y-axis is along-stream distance (space). We can plot the outputs from *spatial\_changes*. Running this cell should produce the following plot:

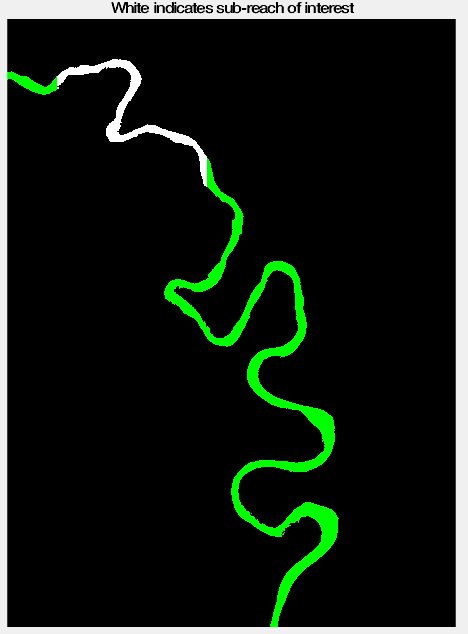


For a finer resolution plot, you can change the spacing of the polygon buffers and re-run *spatial\_changes*. Colorbars may also be added; you can click the pixels within the Matlab figure display to query the values.

# Planform changes for a portion of the reach

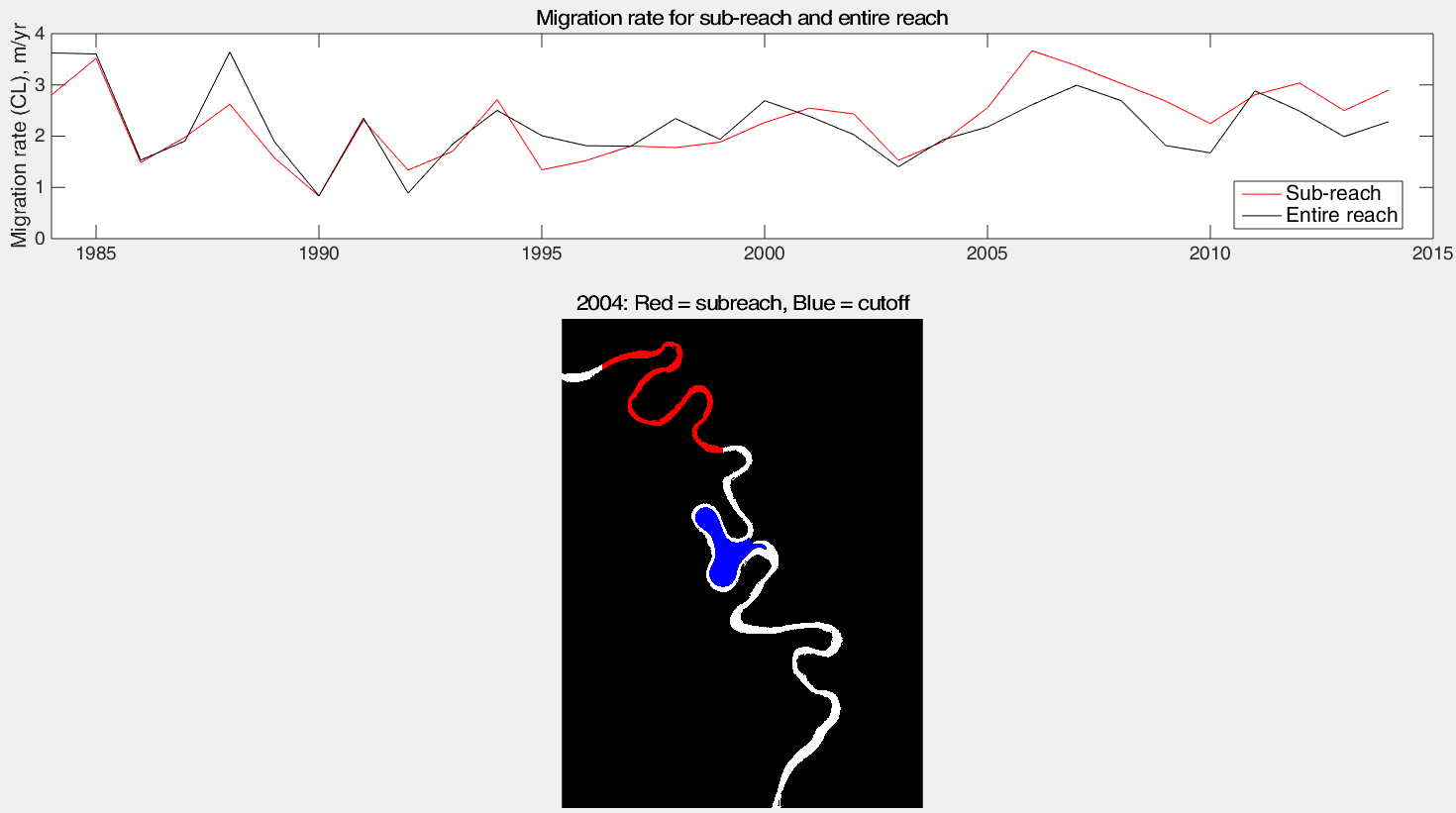
Perhaps we are interested in only a particular portion of our study reach rather than the entire domain. There are a few methods we could use. The simplest method, and the one we will use here, is to define a bounding box with which we will mask our domain, and quantify planform changes only within this box. However, a potentially more useful method is one that tracks individual meander bends (or any two points along a centerline) through time. RivMAP does not currently include functions for this type of tracking, but beta versions have been developed and may be released in RivMAP updates.

Here is a demonstration of computing annual centerline migration rates within a bounding box. The same techniques can be applied to erosion or accretion areas. A bounding box is defined and used to create a mask that is applied to each year’s image that we analyze. We will look at an upper portion of the reach, shown in white by the figure below.



Once a bounding box is defined, we can easily compute migrated areas for each year by masking each image by the box. However, to compute migration rate we also require the centerline length. To find the centerline length, we use the binary centerline image we created in the processing step, *riv.im.Icl*. Each centerline image is masked by the bounding box, and the two endpoints of the masked centerline are found. The centerline length within the bounding box is computed, and migration rates are then computed as before.

Executing this cell should result in the figure below.



In the top panel, the migration rate for the sub-reach through time is shown, along with the migration rate for the entire reach. From 2005-2010, the sub-reach experienced higher migration rates than the entire reach. Could this be due to an upstream cutoff? From our cutoff areas time series, we saw there was a large cutoff within the reach in 2004. The lower panel shows this cutoff (in red) was directly upstream of the sub-reach (in blue). It is possible that this cutoff perturbation induced the accelerated migration rates within the sub-reach from 2005-2010.

# Combining georeferenced images/results

If you are analyzing a large spatial domain, i.e. one that covers multiple Landsat scenes, or a small domain that crosses Landsat scenes, you may choose to divide the domain into smaller portions. For example, in the accompanying paper the Ucayali River was divided into four boxes, R6-R3. This also allows parallelization of the analysis. Landsat data may be downloaded as georeferenced .tiff files, and if you perform pre-processing (such as clipping the Landsat scene) in GIS, you can export the result as a geotiff.

The RivMAP function *combine\_georeffed\_images* allows you to stitch your results back together after analysis. It will stitch both images and vector-based results like centerline coordinates, and examples of both are shown below.

**Note**: You should be aware when exporting geotiffs from a GIS software that you may export the geotiff in the native geographic coordinate system (i.e. Landsat’s UTM/WGS 84 coordinate system) or a projected coordinate system (i.e that of your map). The RivMAP function to stitch imagery back together will only work for a projected coordinate system. Georeferenced images are loaded into Matlab via the *geotiffread* command; if your coordinate system is projected, the loaded georeferencing variable will be of type “MapCellsReference.” If you load an image with a geographic coordinate system, the variable type will be “MapPostingsReference.” Both variable types contain similar information, so you can modify *combine\_georeffed\_images* accordingly if you are working with a geographic coordinate system.

Running this cell will produce two figures; the first shows the channel mask images from each of the R boxes and their stitched image. The second figure shows centerlines in vector coordinates stitched together.

