

# Rosgen Stream Classification in GRASS GIS



Spring 2021

Prepared by Zach Hilgendorf (MSci)  
PhD Student, Geography  
School of Geographical Sciences and Urban Planning  
Arizona State University

and Joseph Holler (PhD), Middlebury College Department of Geography

## **Purpose**

The purpose of this primer is to establish the goals, methods, data, and software required to complete the module for reproducibility in the Rosgen Stream Classification method using geographic information software. A series of short lectures and handouts have been developed to provide ample background information and the skills to necessary to complete the assignment. Two scientific articles have also been provided as required reading.

## **Background Information**

### **Classification Methods:**

To set the stage, classification schemes, whether in geography, economics, biology, etc., are designed to try and “bestow order onto chaos.” If we can generalize a system, we can make certain assumptions about how a similar system may “behave” somewhere else. In geography, we adhere to Tobler’s First Law, which states that things closer to one another are more alike than things that are farther apart. However, we need to be able to compare differences and similarities between streams across the world. If we can classify a stream, based on a number of measured parameters, and we find similar streams, in similar systems, in different parts of the county or world, then we can start compare those streams. If we are trying to modify the stream, for example, we may need to have an idea of whether the modification we are making will work for their designed goal. We use classification methods, like the Rosgen Stream Classification system, to that end. One of the primary uses of the Rosgen system is to inform on the “type” of stream for planned restoration projects. Many local, state, and federal agencies employ this method and train their scientists to use it. As such, there is an important economic aspect at stake. Communities or other agencies have to allocate funds to pay for restoration efforts and depend on these classification schemes to accurately characterize what type of restoration will best fit their problem.

Two scientific articles have been provided as required reading for this module. The first, Rosgen (1994), is one of the original papers that defines and describes the steps necessary to complete the classification of a stream, based on a series of measured in situ parameters. This paper should serve as the baseline of knowledge for you to complete the module. The second, Kasprak et al. (2016), examines a number of different stream classification schema, including Rosgen (1994), to assess the parameters that each uses to classify a stream. Together, these articles offer a succinct view of how classification schemes in fluvial systems handle a wide variety of parameters, how the inclusion or exclusion of variables differs, and why it is useful to have ways in which to classify streams.

### **Physical Geography:**

Streams and rivers dissect the world around us, fill reservoirs, water crops, provide vital ecosystems for plants and animals, and have seen widespread degradation over the last 150 years. As we have mentioned, the Rosgen system has seen extensive use in

the field of stream restoration. The topic of fluvial geomorphology and hydrology is crucial to understanding the parameters we are collecting for the Rosgen system and what they mean. A series of five short (5-10 minute) lectures have been recorded and uploaded to YouTube to help teach you some of the essential concepts you will need to know to complete this module. Each of these videos takes a look at a single topic (channel form, sinuosity, longitudinal profiles, grains+transport, ratios) and breaks them down to an introductory level.

Channel Form: <https://youtu.be/OUT3emNhVwg>

Sinuosity: <https://youtu.be/rVs4MSw1e5s>

Longitudinal Profiles: <https://youtu.be/SK2BFg2PmlQ>

Grains and the Initiation of Motion: <https://youtu.be/61GNZwwu14U>

Ratios in Fluvial Geomorphology: <https://youtu.be/ex7E5GHPp6Q>

### **Software Skills:**

In an effort to make this module widely accessible, we have chosen to use the open source GIS platform GRASS as the preferred graphical interface. GRASS is an incredibly powerful and customizable GIS platform, with a devoted and active user base, widespread integration with other software platforms (e.g., QGIS, RStudio), and free and open source software and code. The handouts in this lab are designed to teach you how to install and set up GRASS, import and manipulate raster and vector datasets, and process data with base and imported packages. These handouts should be approached in the following order:

- 1) Starting In GRASS
- 2) Digitizing in GRASS
- 3) Transects and Profiles
- 4) Classifying A Stream

Later on, you will be asked to do some very simple coding in RStudio, which is a user-friendly “wrapper” for the R language, developed by the Comprehensive R Analysis Network (CRAN). This code has been heavily commented and set up to allow you to understand what the code is doing and why you are inputting the variables you are in the places you are putting them. To install RStudio, you first need to install R. We will be working out of an RNotebook, which allows us to run “chunks” of code at a time and split up the goals of each chunk in a sensible way. All download sites have been provided below.

GRASS GIS: <https://grass.osgeo.org/download/>

CRAN: <https://cran.r-project.org/>

RStudio: <https://rstudio.com/products/rstudio/download/#download>

# 1 Getting Started in GRASS GIS

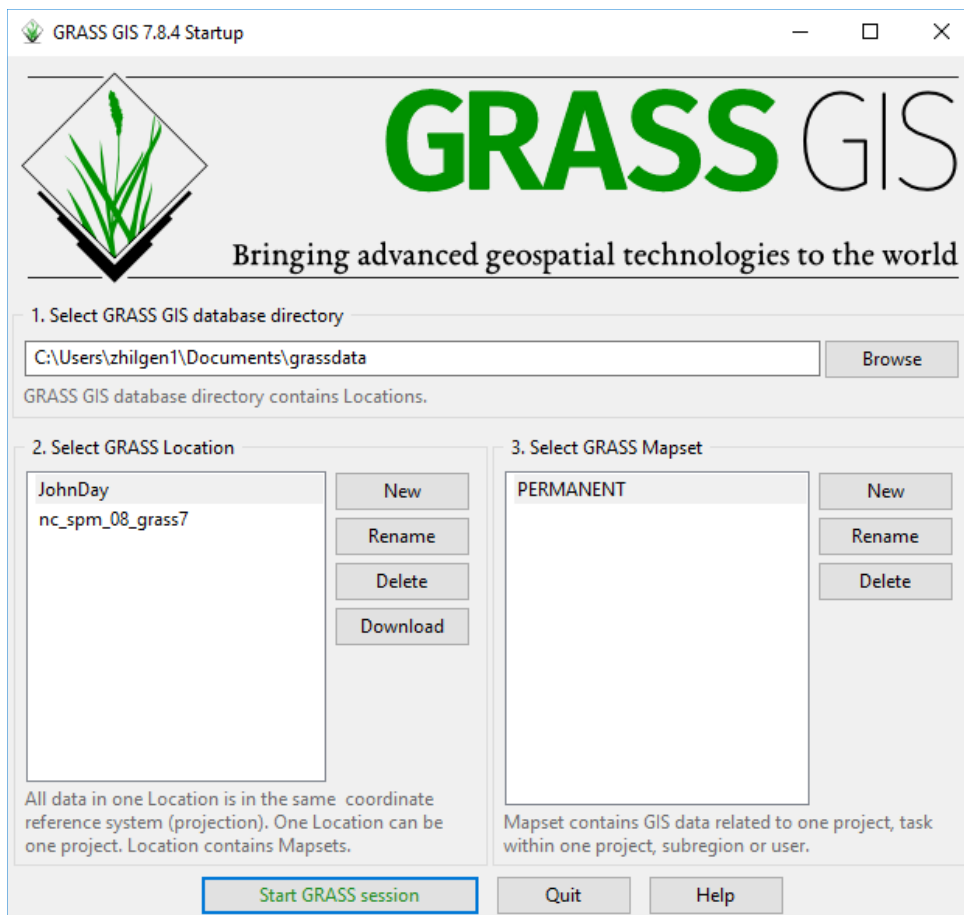
**Purpose:** The purpose of this handout is to teach you how to create a new project in the GIS software GRASS.

## 1.1 Creating a GRASS Project

When you first open GRASS GIS, you will see a black command prompt screen and a GUI that lets you set up and create new **Locations**.

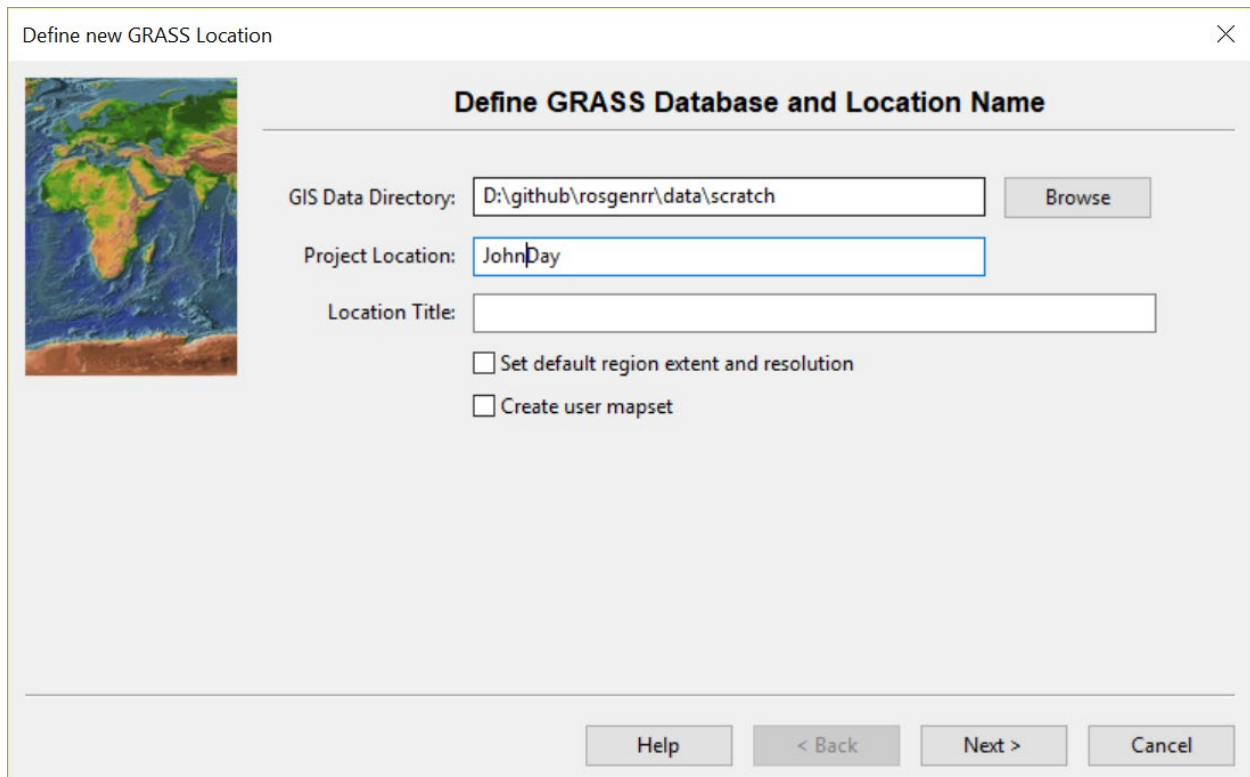
**GRASS NTK #1:** GRASS has its own topological data structure and database system for managing geographic data. Therefore, working in GRASS requires:

- 1) creating a **database directory** in which all GRASS data will be stored
- 2) creating a **Location** establishing the geographic framework for a project or set of projects, including the coordinate reference system to be used and default extents and spatial resolutions
- 3) creating a **Mapset** within a Location. We can use the default PERMANENT mapset, but you can organize different sets of geographic data layers for different projects within the same Location geographic frame by using mapsets.



**Figure 1.** The GRASS GIS startup window, where **directories**, **locations**, and **mapsets** can be created.

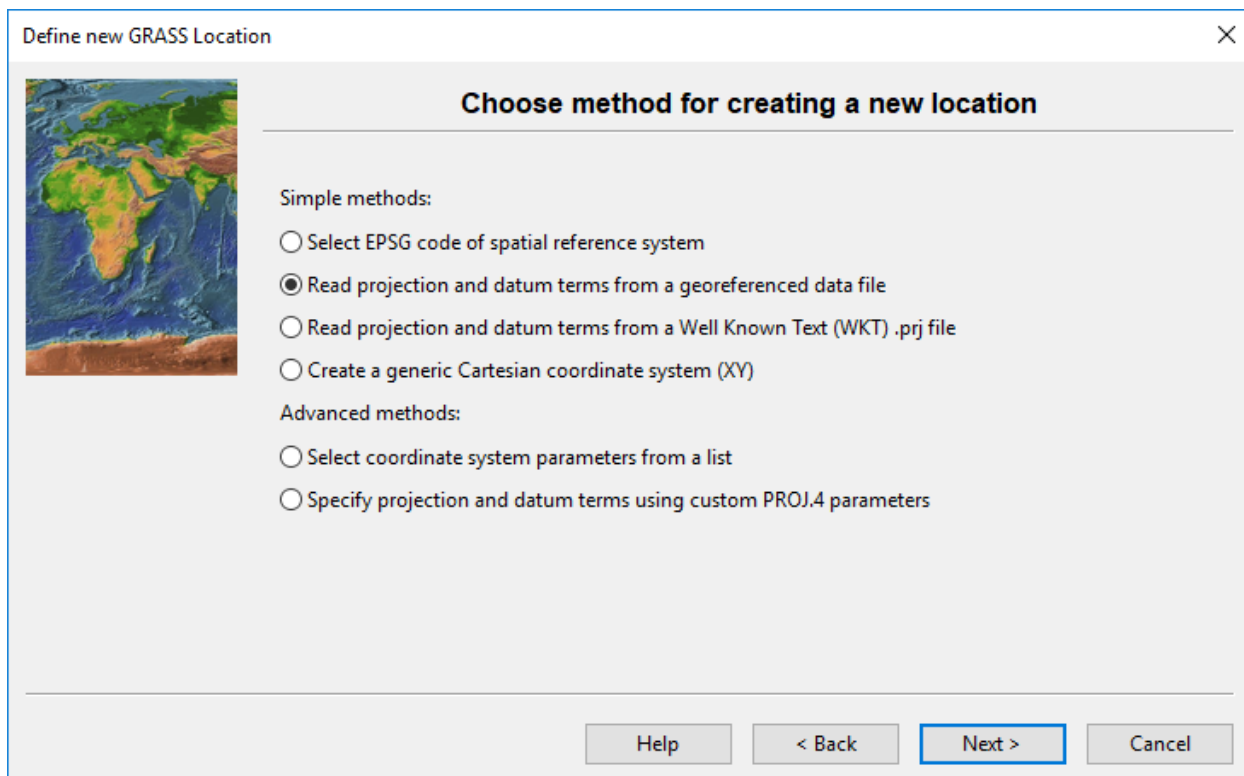
We will start by setting up a new **Location**. Click on New next to the “Select GRASS Location” box. When you do, a new window will open (see Figure 2). This allows you to create the default directory and name for the location. Set your default directory to the **scratch folder** in your **rosgenrr repository**, and give the project location a name. This could be the projection you are using (i.e. NAD 1983 (2011) UTM Zone 11N) or it could be more specific to the project. In this example, I called the location “**JohnDay**” in reference to the John Day River of Oregon, where our data is from.



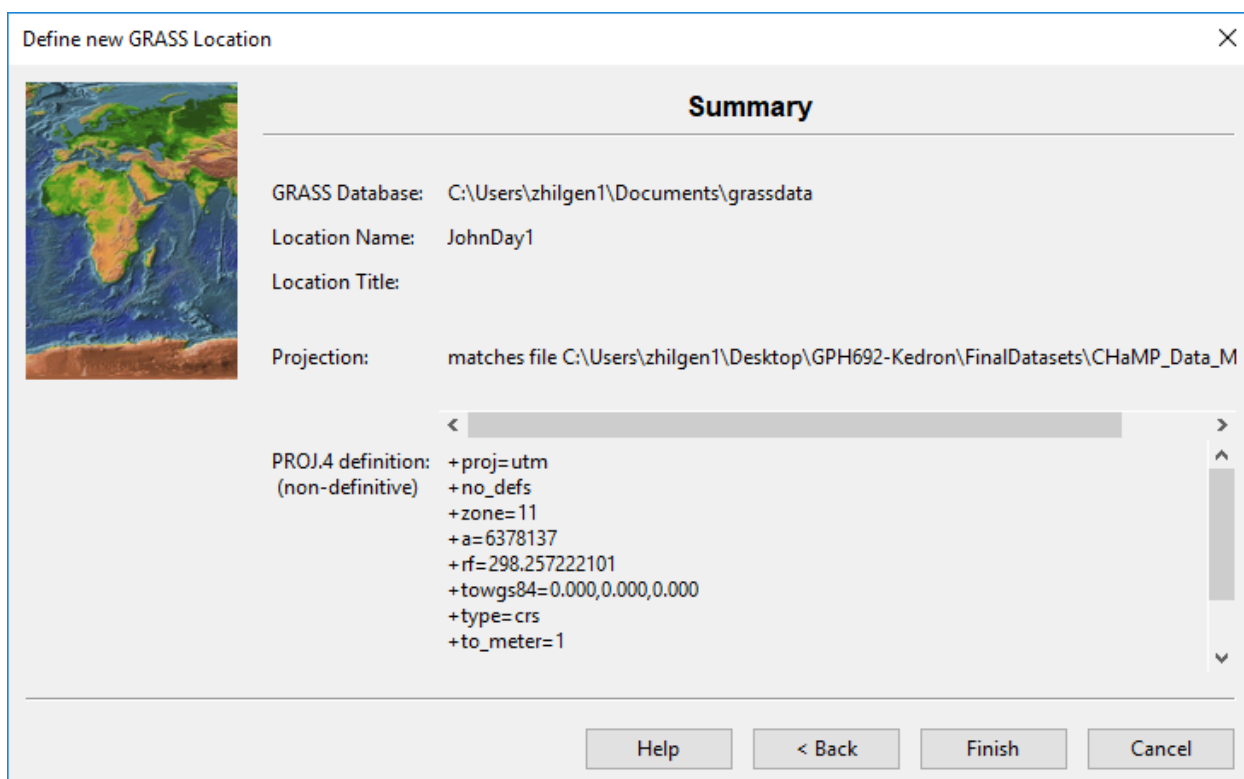
**Figure 2.** Window for setting up a new Location in GRASS.

Next, you will need to determine which **method** you want to use for **creating the new location**. GRASS has a lot of great ways to do this, from reading the European Petroleum Survey Group (EPSG) code, to reading the data directly from a file. We will employ the latter option for our project! Select the “Read projection and datum from a georeferenced data file” bubble and hit Next (Figure 3). In the next window, browse until you find the dataset you want to import the projection from. We will choose the CHaMP\_Data\_MFJD.prj file, as it is the projection for the CHaMP dataset you will be working with. Click Next after you have brought that file into the input box and you will be brought to a window that shows you the projection data that is being pulled from the file (Figure 4). Ensure that it is what you know the dataset to be. In this case, we are working in NAD 1983 UTM Zone 11 North. You will see that, in Figure 4, that “proj = utm” and “zone = 11,” confirming that we imported the projection from the file.





**Figure 3.** The window to choose how a location will acquire its spatial reference.



**Figure 4.** The summary window that shows the projection that is being pulled from the chosen file.

Once you have confirmed that the projection is correct, select “Finish” and the location will generate. You will get a popup window that asks if you want to import the dataset you used as a reference into the Mapset. Either choice is fine, but this is a nice way to get a jump on importing your data into GRASS!

For this project, we can use the default PERMANENT mapset. Select “**Start GRASS Session**” to launch the program!

**GRASS NTK #2:** GRASS uses the many windows approach.

- 1) The **Layer Manager** is the workhorse:
  - a) **Layers** tab is the equivalent to the Layers panel in QGIS, controlling display and order of layers in the map display
  - b) **Console** tab is the equivalent to the Log & History in QGIS, and also contains a command prompt where you type and run commands one algorithm at a time.
  - c) **Modules** tab is the equivalent of the Processing Toolbox: search and run algorithms interactively from here.
    - i) When you launch algorithms this way, a window opens and stays open even after you run the algorithm, making it easy to re-run the algorithm with the same settings and cluttering your desktop with excessive windows.
  - d) **Data** tab is the equivalent of browser, showing all of the spatial data in your current GRASS location.
- 2) The **Map Display** window is where you view and interactively select, query, or digitize spatial data.

**GRASS NTK #3:** A **workspace** in GRASS is the equivalent of a Project in QGIS.

- 1) Save your configuration of a map layout and layers with **File** → **Workspace** → **Save**
- 2) Load your configuration of a map layout and layers with **File** → **Workspace** → **Load**

**GRASS NTK #4:** GRASS is *extremely particular* in managing its own spatial data structures. Therefore, you need to process all of your spatial data with an import procedure. Sometimes, we will also need to micromanage vector data creation steps, like creating attribute tables with unique IDs.

## 1.2 Addons for GRASS

We need three Addon modules (algorithms) for this analysis in GRASS. To do this, we'll first need to go to the **Settings** → **Addons extensions** → **Install extensions from addons [g.extension]** in the Layer Manager window. That will open a menu in which you can install and manage extensions within GRASS. Enter “**v.centerline**” in the search bar and double-click on it to install the extension.

Here install three Addons in total:

- 1) v.centerline
- 2) v.transects
- 3) r.clip

You can find the extension in the **Modules** tab in the Layer Manager window, under the Addons (expand the plus sign) (Figure 5).

You may need to close and restart GRASS in order to see the Addon in your own local GRASS modules.

### 1.3 Loading and Visualizing Data

Most of the analysis can be accomplished by running tools from the **Modules** tab of the **Layer Manager** window. The outline below mentions all settings that need to be changed. If a setting is not mentioned, you may use the default setting. The outline is arranged as follows:

- 1) Module name
  - a) Tab of module
    - i) Setting
  - b) Tab of module
    - i) Setting
  - c) The Manual tab is for “help manual”, not do-it-yourself by hand manual. Use it for clarification and assistance!



**GRASS NTK #5:** Several aspects of the graphic user interface may not update without manually using the refresh button, including the data list, layers list, and graphic model.

The following procedure will import the data, select a site for stream classification, and prepare raster imagery to guide your digitization efforts.

- |  |  |
|--|--|
| Import the digital elevation model (dem)   | 1) r.import (note: this process takes some time to run) <ol style="list-style-type: none"><li>a) Source settings<ol style="list-style-type: none"><li>i) File: JohnDayWSHed.tif (in data/raw/private folder)</li></ol></li></ol>   |
| Import the CHaMP survey points   | 2) v.in.ogr (note: double click this layer in the Data tab after it loads to add it to the map) <ol style="list-style-type: none"><li>a) Source Input<ol style="list-style-type: none"><li>i) CHaMP_Data_MFJD.shp (in data/raw/public folder)</li></ol></li></ol>  |
| Create a buffer around survey point of interest. Rosgen (1994) suggests analyzing a distance of 20 channel widths (10 upstream, 10 downstream), which we apply in the Distance settings. | 3) v.buffer <ol style="list-style-type: none"><li>a) Required<ol style="list-style-type: none"><li>i) Name of input vector map: CHaMP_Data_MFJD</li><li>ii) Name of output vector map: reachBuffer</li></ol></li><li>b) Selection<ol style="list-style-type: none"><li>i) Layer number: 1</li><li>ii) WHERE conditions of SQL statement without 'where' keyword:<br/>loc_id = <i>your location of interest id here</i></li></ol></li><li>c) Distance<ol style="list-style-type: none"><li>i) Name of column to use for buffer distances: BfWdth_Avg</li><li>ii) Scaling factor for attribute column values: 10</li></ol></li></ol> |



- Create an analysis region around the buffer.
- 4) g.region (Set region)
- a) Existing
    - i) Set region to match vector map: reachBuffer
  - b) Bounds
    - i) Number of cells to add to each side of the current region extent: 500
- Clip the elevation data to your analysis region
- 5) r.clip
- a) Required
    - i) Name of input raster map: JohnDayWSshed
    - ii) Name for output raster map: DEM
- Recolor the elevation data
- 6) r.colors
- a) Map
    - i) Name of raster map: DEM
  - b) Define
    - i) Name of color table: elevation
- Calculate hillshading
- 7) r.relief
- a) Required
    - i) Name of input raster: DEM
    - ii) Name for output shaded relief map: relief
- Apply hillshading to the elevation data
- 8) r.shade
- a) Required
    - i) Name of shaded relief or aspect raster map: relief
    - ii) Name of raster to drape over relief raster map: DEM
    - iii) Name of shaded raster map: demShade
- Calculate slope
- 9) r.slope.aspect
- a) Required
    - i) Name of input elevation raster map: dem
  - b) Outputs
    - i) Name for output slope raster map: slope
- Apply styling to the buffer
- 10) d.vect (open by opening the reachBuffer layer properties)
- a) Required
    - i) Name of vector map: reachBuffer
  - b) Colors
    - i) Feature Color: black
    - ii) Area fill color: transparent
  - c) Lines
    - i) Line width: 2

Create three blank vector map layers for digitizing banks  
*Note:* I have not sorted out a way to do this in a model.

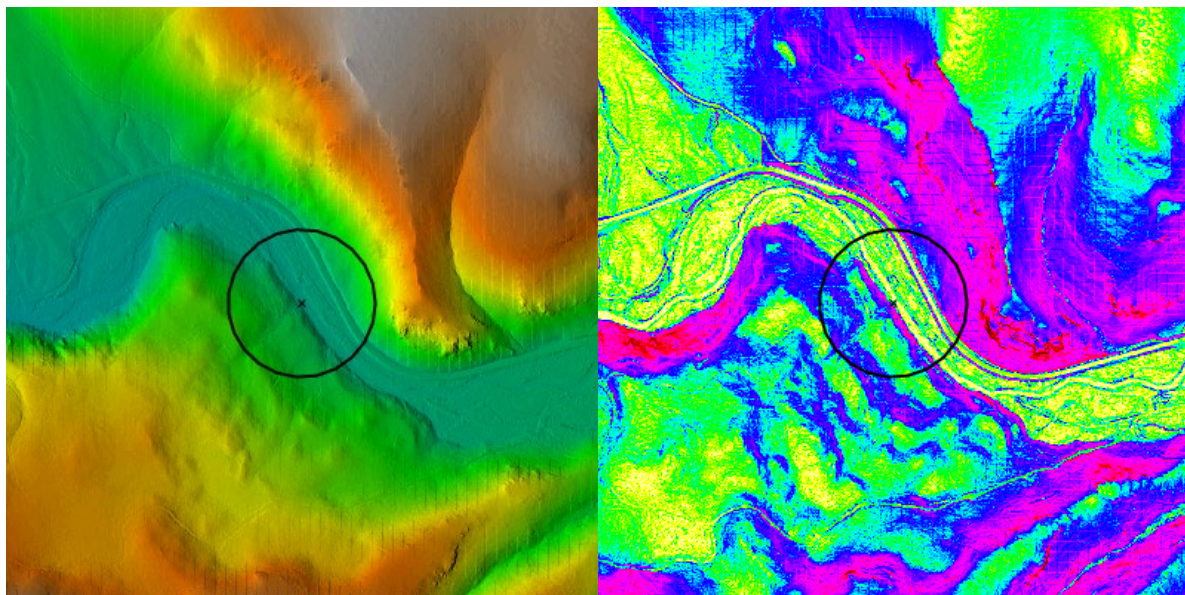
- 11) create new vector map  
a) name: banks1
- 12) create new vector map  
a) name: banks2
- 13) create new vector map  
a) name: banks3

Create three blank vector map layers for digitizing valley edges

- 14) create new vector map  
a) name: valley1
- 15) create new vector map  
a) name: valley2
- 16) create new vector map  
a) name: valley3

#### 1.4 Visualizations

At this point, you should be able to visualize the Buffer, CHaMPS site, and shaded elevation and shaded slope for a selected CHaMPS site (Figure 5).



**Figure 5.** Shaded elevation map (left) and a Slope map in the vicinity of CHaMPS site CBW05583-275954. You should be able to create these two visualizations to guide your digitization.

#### 1.5 Remember how to save and reload the map

- 1) To save your map visualization work, go to **File** → **Workspace** → **Save**
- 2) When you reopen GRASS, you can similarly load previous workspaces at **File** → **Workspace** → **Open**

## 1.6 Hints for visualizing data in GRASS

- 1) Add data to the map: double-click the dataset in the **Data** tab
- 2) Reorder layers: click and drag layers on the layers tab
- 3) Change raster colors: right click the layer and **set color table interactively** (the example below uses the **elevation** color table)
- 4) Change layer transparency: right click the layer and **change opacity level**
- 5) Open a vector layer attribute table: right click the layer and **show attribute table**
- 6) Highlight a selected feature: double-click the feature's row in the attribute table

## 1.7 Summary

In this section, we launched the GRASS GIS program, selected a point of interest, and created a shaded elevation map, a slope map, and three blank vector maps for digitizing banks and valley edges. In the next section, we will get to work digitizing those banks and valleys.

## 2 Digitizing in GRASS

**Purpose:** One of the many great things we can use GRASS GIS for is to digitize, collapse, and extract longitudinal profiles from a stream. We can use these same tools to create a valley centerline or extract valley cross sectional profiles. This handout will walk through the tools and methods to extract longitudinal profiles, calculate stream length and valley centerline length, and create cross-sectional profiles of the stream valley. These outputs will be used, directly, as inputs into the Rosgen Classification table for slope and sinuosity. To learn more about these concepts, please refer to the recorded YouTube videos

- 1) Sinuosity: <https://youtu.be/rVs4MSw1e5s>
- 2) Slope/Longitudinal Profile: <https://youtu.be/SK2BFq2PmlQ>

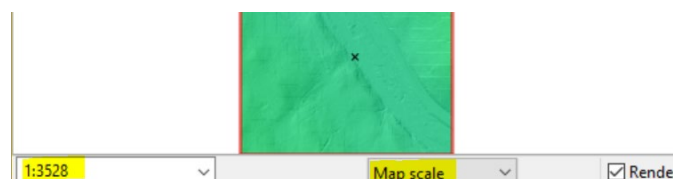
### 2.1 Digitize Stream Banks and Valley Sides

Our first goal will be to digitize the **river banks** and the **edges of the valley**.

Let's navigate to the a specific CHaMP location in order to digitize channel morphology. We will be using a site on the Middle Fork of the John Day River (Site: CBW05583-275954) for our example, but you should use your own site of interest.

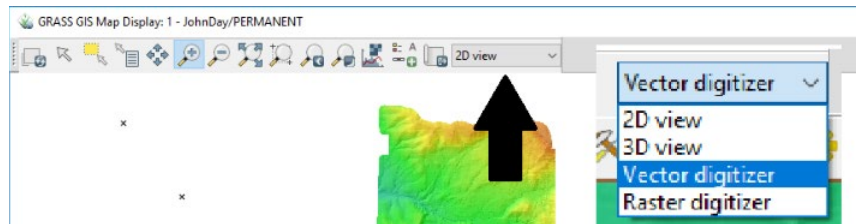
We'll use the following workflow for digitizing both the stream banks and the valley outline. Digitizing practices typically suggest digitizing the same features multiple times, to ensure consistency (Micheli et al., 2004). Sometimes, when constrained by resolution or with features that are tough to define, this practice can help delineate a range of possible extents and distinguish areas with greater uncertainty. We will observe the following standards:

- 1) All digitized features will be digitized three times.
- 2) All features will be digitized at a scale of 1:1500. (set this with the status bar at the bottom of the map: change Coordinates to Map Scale and type the scale in at left)
- 3) Name layers clearly and iteratively, for clarity (banks1/2/3, valley1/2/3).
  - a) Each layer will have two line features: the two sides of the river bank or the two sides of the valley.
- 4) Alternate between creating features (banks1, valley1, banks2, valley2, banks3, valley3) to reduce likelihood that "muscle memory" will skew outputs.



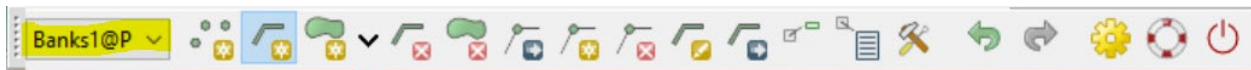
**Figure 1:** Use the status bar to change the map scale.

Start your digitizing work by opening the **Vector digitizer** map view by switching from 2D view to Vector Digitizer (Figure 2)



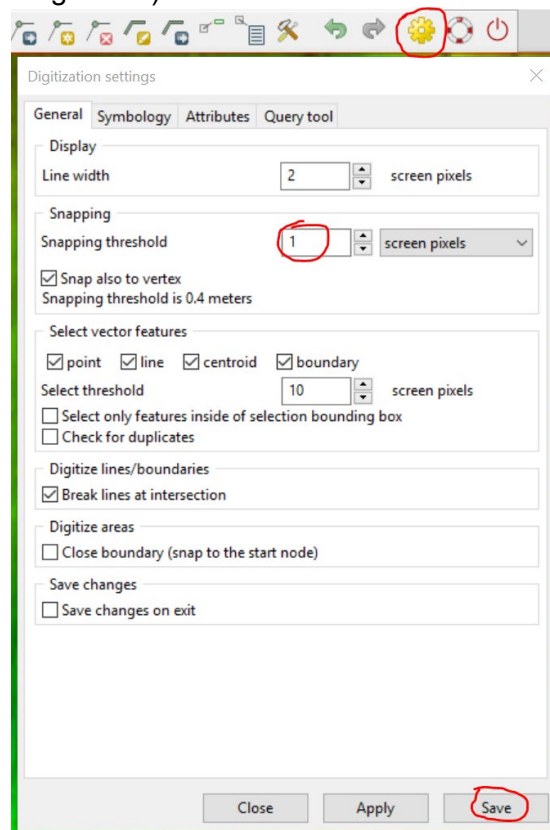
**Figure 2.** The black arrow is indicating the location of the digitizer option (shown in the callout box).

A new toolbar will appear in the Map Display window with options for vector creation and editing. On the left side of the toolbar, choose to start editing the **banks1** map layer you created.



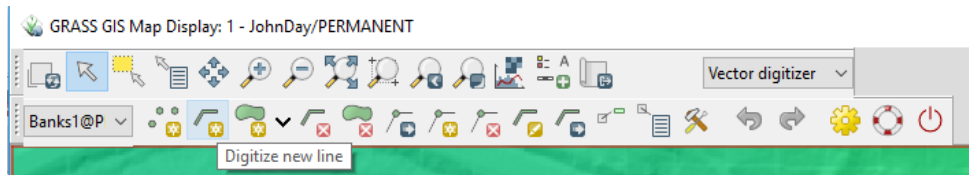
**Figure 3.** Vector digitizer toolbar and the dropdown menu for selecting which vector to edit, with option to create a new vector.

We need to be able to digitize very precise lines close together, so we need to reduce the snapping distance. Go to the Settings (yellow gear) and set the **snapping threshold** to **1 screen pixel** and **save** (see figure 3b).



**Figure 3b.** Vector digitizer toolbar and the dropdown menu for selecting which vector to edit, with option to create a new vector.

After choosing the map layer to edit, select the “Digitize new line” tool (Figure 4). There are many different methods for digitizing streambanks, depending on the available dataset. To maintain consistency, we will utilize the John Day elevation relief and slope to guide our digitization. The cells of this raster dataset have a spatial resolution of 1 meter. This is an elevation-based dataset, so we do not have to worry about vegetation obstructing our vision. However, resolution and color ramps can impact how we interpret the elevation models.

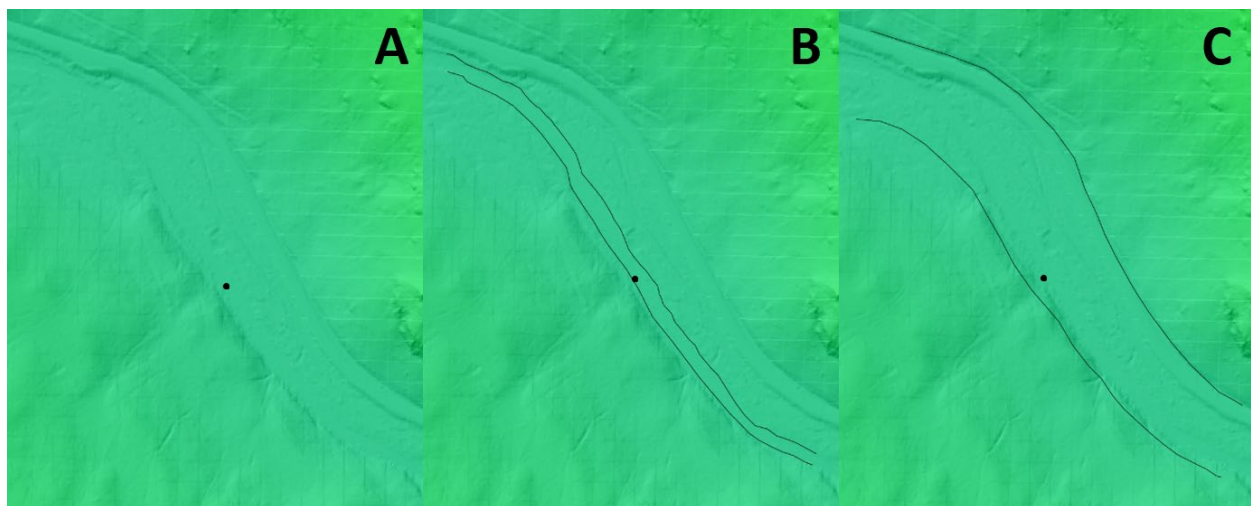


**Figure 4.** Location of the “Digitize new line” tool. Make sure that the layer you want to be adding a line to is selected in the left dropdown menu.

If we were digitizing a satellite image or an orthomosaic, we may be able to see the streambanks, but they could be covered by vegetation or flooded water. We will not have to deal with it in this dataset, but there are some examples of the different ways to digitize orthomosaic imagery in the Appendix.

For the channel banks, we will want look for distinct breaks in slope around the channel or obvious slopes in the hillshade. We’ll want to be careful! The road on the north side of the valley looks really similar to a river, but we can see that it is raised above the valley floor. It can be a bit tough to make out the edges of the river, but with careful digitizing at a consistent scale, we can get a good idea of the outline of the channel (Figure 5B).

Rosgen (1994) suggests analyzing a distance of 20 channel widths (10 upstream, 10 downstream), which we applied in the Distance settings when we created our buffer. Therefore, digitize banks and valley edges just beyond the extent of the buffer so that we can later clip the lines with the buffer.





**Figure 5.** A) DEM of the Middle Fork of the John Day River with the example datapoint in the center of the map. B) An example of the digitized streambank lines from our first digitized banks. C) An example of the digitized valley extent from our first dataset.

The methods for digitizing the valley will be similar. We want to capture the extent of the stream valley and cover the same distance as we did with the banklines (Figure 5C). We can see some obvious breaks in slope, evident by the underlying hillshade. Sometimes, it is a bit tougher to tell, but we mostly want to get the general shape of the valley floor so that our valley centerline is accurately represented. You may also use the slope map to identify the flat areas encompassing stream surfaces and floodplains.

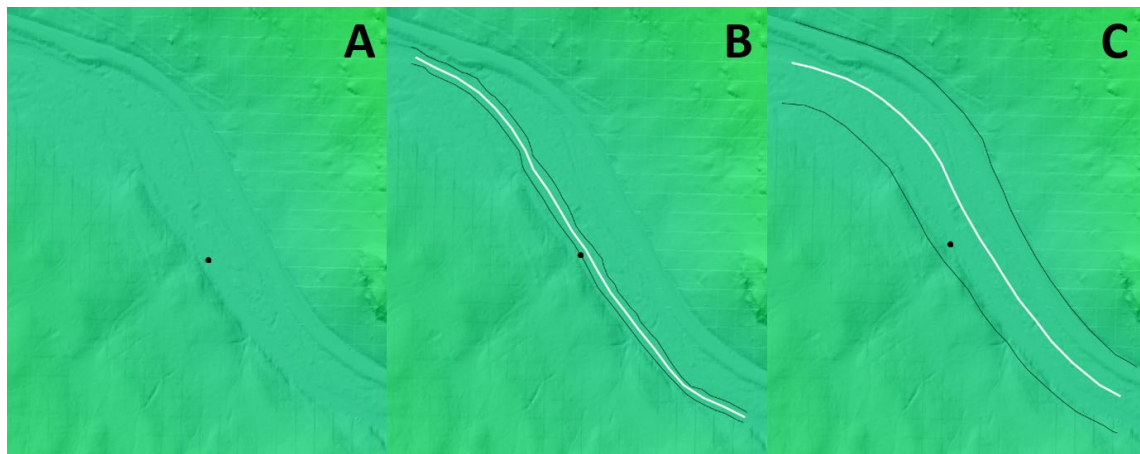
To digitize the line, **left-click** for each **vertex** and **right-click** to **complete** the feature. A window for entering attributes pops up, but we have no attributes so you can just **submit**.

Once you are done with digitizing one set of banks, find the layer you were editing in the **Layers** panel.

- Right-click the layer and **stop editing**.
- Uncheck the layer to remove its visibility from the map.
- Return to the vector digitizer and select a new set of valley edges or banks to digitize (see Figure 3) until you have completed three sets of banks and three sets of valley edges.
- Try digitizing some layers with the slope background and some with the shaded relief background.

## 2.2 Calculate Center Lines and Line Lengths

In this section, we will transform the stream and valley banks into **center** lines (illustrated in figure 6), **patch** the three versions of center lines into one, and then calculate their **lengths** so that we can find **sinuosity**.



**Figure 6.** A) DEM of the Middle Fork of the John Day River with the example datapoint in the center of the map. B) An example of the digitized streambank lines (in black) and the related centerline (in white). C) An example of the digitized valley extent (in black) and the related centerline (in white).

- Find the centerline of each set of banks
- 1) v.centerline
    - a) Required
      - i) Name of input vector map: banks1
      - ii) Name for output vector map: banks1c
    - b) Optional
      - i) Number of vertices for center line: 100
  - 2) REPEAT this step for banks2 and banks3 so that you have banks1c, banks2c, and banks3c
- Patch the three centerlines into one map layer
- 3) v.patch
    - a) Required
      - i) Name of input vector maps: banks1c,banks2c,banks3c
    - b) Name for output vector map:
      - i) banksPatched
- Find the centerline of the three centerlines
- 4) v.centerline
    - a) Required
      - i) Name of input vector map: banksPatched
      - ii) Name for output vector map: banksCenterLine
    - b) Optional
      - i) Number of vertices for center line: 100
- Clip the centerline to the buffer delineating your stream reach
- 5) v.clip
    - a) Required
      - i) Name of vector map to be clipped: banksCenterLine
      - ii) Name of clip vector map: reachBuffer
      - iii) Name for output vector map: banksLineClipped
- Add a unique ID (cat) to the center line
- 6) v.category
    - a) Required
      - i) Name of input vector map: banksLineClipped
      - ii) Action to be done: add
    - b) Optional
      - i) Name for output vector map: banksLineCat
      - ii) Category value: 1
- Add an attribute table to the center line with an attribute or length
- 7) v.db.addtable
    - a) Required
      - i) Name of vector map: banksLineCat
    - b) Definition
      - i) Layer number where to add new attribute table: 1
      - ii) Name of key column: cat
      - iii) Name and type of the new column(s):  
length double

- Calculate length      8) v.to.db
- a) Required
    - i) Name of vector map: banksLineCat
    - ii) Value to upload: length
    - iii) Name of attribute column(s) to populate: length
  - b) Optional
    - i) Allow output files to overwrite existing files
    - ii) Units: meters
- Visualize uncertainty      9) Save a map image of your three centerlines and the final line, to help visualize uncertainty.
- 10) Repeat the steps above for valley edges, concluding in a valleyLineCat layer with the length in meters.

### 2.3 Summary

You have finished with this section when you have calculated the stream length (estimated from the center of the banks) and the valley length (estimated from the center of the valley edges).

In this section, we talked about how to digitize lines and the methods for digitizing them in GRASS GIS. We also learned how to install add-on packages and use the *v.centerline* tool to collapse lines to a single average line. These skills will prove useful on the next handout, where we learn how to use those centerlines to extract and graph raster cells that intersect our lines, to calculate slope values to input into our classification scheme!

## 3 Extract River Profiles

**Purpose:** The purpose of this handout is to take the methods from the GRASS GIS digitization handout and employ them to extract data from raster cells and put them into a two-dimensional view to calculate slope or to better understand the shape of the valley. If you are in need of a refresher, refer back to the digitization workflow, as this picks up where that one left off. The concepts to consider in this lab are best portrayed by the following YouTube videos:

Sinuosity: <https://youtu.be/rVs4MSw1e5s>

Slope/Longitudinal Profile: <https://youtu.be/SK2BFq2PmlQ>

In the previous walkthroughs, we went over how to build locations and mapsets, and then how to digitize vector maps, add packages, and use new tools, such as the *v.centerline* tool, using GRASS GIS. The primary purpose of this walkthrough is to teach you how to extract raster cells, along a line, to plot, view, and analyze the data in your preferred plotting software (i.e. Microsoft Excel, Google Sheets, R, Matlab, etc.).

We will start off where the digitization handout concluded. Make sure you have the John Day DEM, Banks\_Center vectors, Valley\_Center vectors, and CHaMP data vector loaded into the Map Display.

### 3.1 Extract Longitudinal Profile

A longitudinal profile will allow us to visualize the slope of the river as it descends from its headwaters to its outlet.

Use the following procedure to extract the profile.

- |   |  |
|---|--|
| Transform the bank center line into points.                     | 1) v.to.points   |
|   | a) Required  |
|   | i) Name of input vector map: banksLineCat (layer output from step 2 above)                                   |
|   | ii) Name for output vector map: longitudinal_points  |
|   | b) Optional  |
|   | i) Use line nodes (start/end) or vertices only: vertex   |
| Export the longitudinal points to simple text format            | 2) v.out.ascii   |
|   | a) Required  |
|   | i) Name of input vector map: longitudinal_points   |
|   | ii) Output format: point   |
|   | b) Output  |
|   | i) Save as <b>longpoints.txt</b> in your <b>data\derived\public</b> folder.                                  |
|   | c) Points  |
|   | i) Field separator: comma  |
| Extract elevation data along lines defined by point coordinates | 3) r.profile   |
|   | a) Required  |
|   | i) Name of input raster map: JohnDayWSHed  |
|   | b) Optional  |
|   | i) Name of file containing coordinate pairs: <b>longpoints.txt</b> in your <b>data\derived\public</b> folder |
|   | ii) Name of file for output: <b>longprof.txt</b> in your <b>data\derived\public</b> folder                   |

The file you just created, **longprof.txt**, will be used in RStudio for graphing the longitudinal profile of this section of river and calculating its slope.

### 3.2 Extract Cross-Sectional Profile


Cross sectional transects are useful for examining the shape of a river valley as the river advances downstream. The cross-sectional shape of the valley informs us about the evolutionary history of the river, its potential for flooding, and even the potential threat of natural hazards, such as mass wasting events.

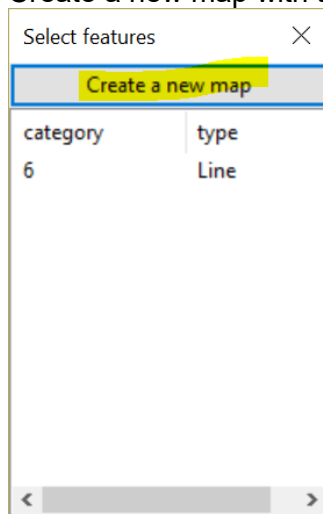
Use the following procedure to extract the profile.

The goal is to create a transect very close to your CHamPS point and spanning the valley. If necessary, adjust the transect spacing and distance transect extends parameters to achieve this goal.

- 1) v.transects
  - a) Required
    - i) Name of input vector map: valleyCenterCat
    - ii) Name for output vector map: transects
    - iii) Transect Spacing along input polyline: 50
  - b) Optional
    - i)  Allow output files to overwrite existing files
    - ii) Distance transect extends to the left of input line: 250
    - iii) Distance transect extends to the right of input line: 250
    - iv) Input feature type: line
    - v) Determines how transect spacing is measured: along
    - vi) Determines which line is the transect perpendicular to: trend

Select the closest transect to your point of interest and create a new map layer with it

- 2) Please use the following steps to extract one transect:
  - a) Highlight the transect layer in the **Layers** list.
  - b) Select the transect  closest to the CHaMPS data point you are working on.
  - c) Create a new map with the selected transect.



- d) You should have a new data layer with just one line, the transect nearest to your CHaMPS point.

Transform the transect into a series of points

- 3) v.to.points
  - a) Required
    - i) Name of input vector map: transects\_selection\_\_ (layer output from step 2 above)
    - ii) Name for output vector map: cross\_points
  - b) Optional
    - i)  Interpolate points between line vertices
    - ii) Maximum distance between points in map units: 1

Use the series of points to extract elevation data

- 4) v.out.ascii
  - a) Required
    - i) Name of input vector map: cross\_points
    - ii) Output format: point
  - b) Output
    - i) Save as **crosspoints.txt** in your **data\derived\public** folder.
  - c) Points
    - i) Field separator: comma

Extract elevation data along lines defined by point coordinates

- 5) r.profile
  - a) Required
    - i) Name of input raster map: JohnDayWSshed
  - b) Optional
    - i) Name of file containing coordinate pairs: **crosspoints.txt** in your **data\derived\public** folder
    - ii) Name of file for output: **crossprof.txt** in your **data\derived\public** folder

### 3.3 Summary

In this handout, we learned how to generate transects and convert lines into points, that we then used to extract values from a raster for further assessment. With this workflow, we are able to get the slope and sinuosity values needed for the Rosgen classification!

You should now move on to the RStudio notebook for viewing and analyzing the river transect profile, using the 3-Classification.pdf file as guidance.

## 4 References

Micheli, E. R., J. W. Kirchner, and E. W. Larsen. "Quantifying the effect of riparian forest versus agricultural vegetation on river meander migration rates, Central Sacramento River, California, USA." *River research and applications* 20, no. 5 (2004): 537-548.

Rosgen, D.L., 1994, A classification of natural rivers: CATENA, v. 22, p. 169–199, doi:10.1016/0341-8162(94)90001-9.



## 5 Appendix: Other Bank Digitization Approaches



This method of bankline digitization utilizes the crown (middle) of the trees that surround the river to identify a reasonable bankline. While it is possible that this method overestimates the width, it is likely that the surrounding trees are overhanging the river, so the crown is likely near the bankline.

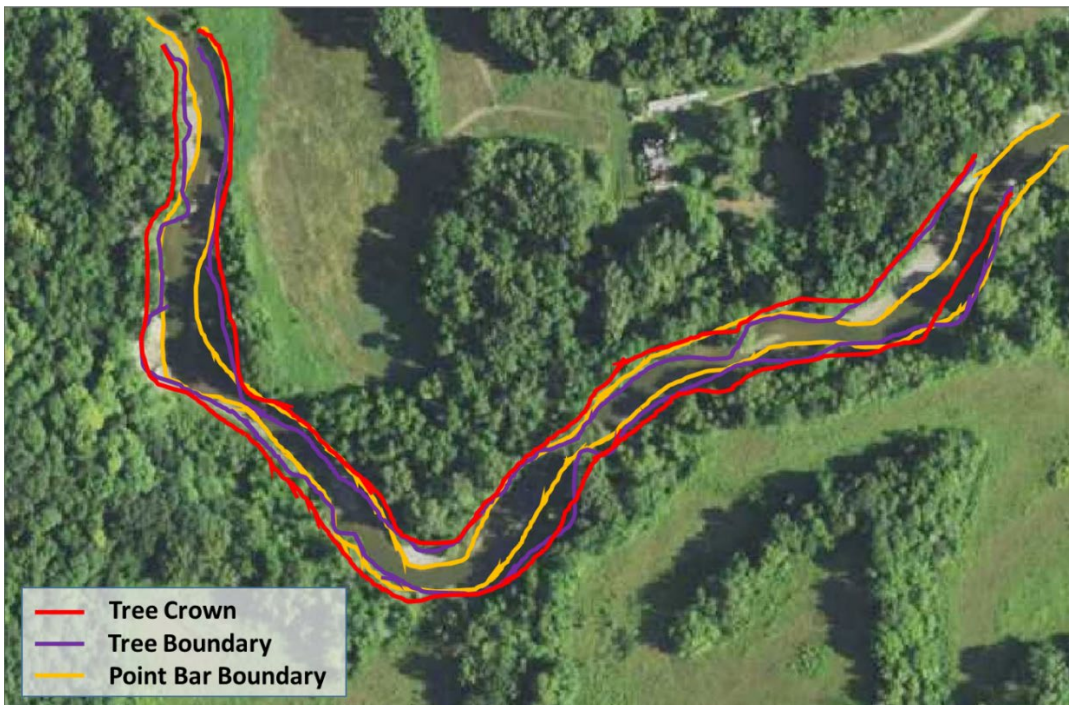


This method digitizes utilizes point bars, depositional features on the inside of a river meander bend, to approximate the banklines. In low flow situations, this is reasonable, but may increase sinuosity by adding curvature to the river.





This method digitizes the boundary of the treeline, as we do not technically know where the bank exists underneath the tree and this focuses more on the water we can see. Notice how “lumpy” the boundary lines look here. You can imagine this may add unnecessary distance to the end product.



Here all three methods are shown on top of one another. The preferred and suggested method is to use the crown of the tree when the bank is not obviously visible. This provides the most reasonable approximation of the bankline, when not visible.