ChanGeom Manual

Prepared by G. Burch Fisher and Bodo Bookhagen
Last Updated 1/22/14

CONTACT INFO

G. Burch Fisher - burch@eri.ucsb.edu
Bodo Bookhagen - bodo@eri.ucsb.edu

INTRODUCTION

This manual has been created to help users with the application of the channel width algorithms and work flows developed by G. Burch Fisher and Bodo Bookhagen at UC Santa Barbara. These algorithms aim to accurately delineate single thread channel widths and planform geometry from digitized polygons derived from freely available satellite imagery (namely from Google Earth). The utility and specific applications of this technique are detailed in the following publications,

> **Fisher, G.B.**, B. Bookhagen, and C.B. Amos (2013), Channel planform geometry and slopes from freely available high-spatial resolution imagery and DEM fusion: Implications for channel width scalings, erosion proxies, and fluvial signatures in tectonically active landscapes,*Geomorphology*, 194, 46-56, doi:10.1016/j.geomorph.2013.04.011.
>
> **Fisher, G.B.**, C.B. Amos, B. Bookhagen, D.W. Burbank, and V. Godard (2012), Channel widths, landslides, faults, and beyond: The new world order of high-spatial resolution Google Earth imagery in the study of earth surface processes, in *Google Earth and Virtual Visualizations in Geoscience Education and Research*, edited by S.J. Whitmeyer, D.G. De Paor, J. Bailey, and T. Ornduff, *Geological Society of America Special Paper 492*, 1-22, doi:10.1130/2012.2492(01).

with updated algorithms and manuals available at the following web addresses for download:

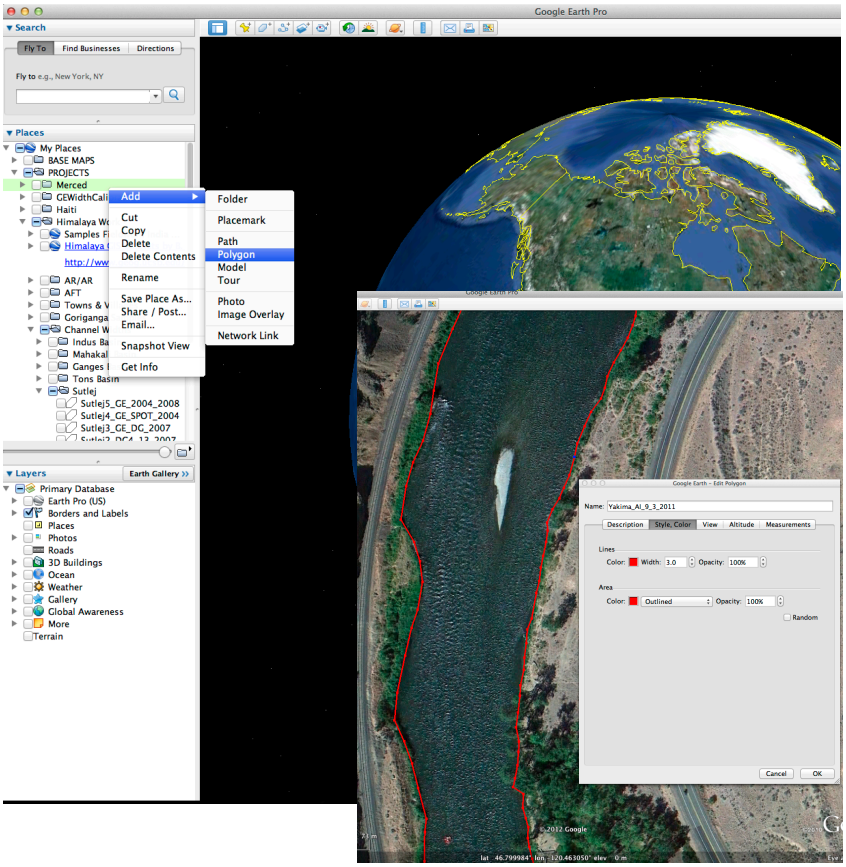
> http://people.eri.ucsb.edu/~burch/Burchfisher/DATA.html
> http://www.geog.ucsb.edu/~bodo/changeom

Please let us know if you are using these algorithms and feel free to enhance and augment them at will, as these are provided freely to the user community. **WE DO ASK THAT YOU PROPERLY CITE OUR WORK (ABOVE) IN ANY PUBLICATIONS OR PRESENTATIONS THAT ARISE FROM THE USE OF THESE ALGORITHMS IN YOUR RESEARCH.** If you do greatly enhance their function or find egregious errors please let us know and we will update the algorithm and make improved versions available. We acknowledge that there are more sophisticated ways to do things and you are welcome to make everything more efficient with respect to your individual goals and needs. Our goal was to make an efficient cookie-cutter work flow that is easy to understand for people with limited GIS and coding ability, as well as provide useful algorithms for more advanced programmers. If at anytime you need help with any aspect of the work flow or algorithms do not hesitate to contact us via our email addresses above.

WORK FLOW

STEP 1: DIGITIZING IN GOOGLE EARTH

Here we will describe the digitizing work flow related to using Google Earth as it contains one of the largest databases of imagery that can be freely interacted with and allows the export of spatial datasets. It is what we have used 99% of the time, however, we have also utilized Bing Maps within ArcGIS 10 as well as Landsat, ASTER, SPOT, and GeoEye satellite products (*Fisher et al., 2013; Fisher et al., 2012*). In these cases the digitizing will take place on whatever software platform is used to view and interact with the data (e.g. ArcGIS, GRASS, Quantum GIS, ENVI, ERDAS IMAGINE, Matlab, etc).



In Google Earth, begin by creating a new polygon by right clicking on the destination folder and selecting **Add > Polygon**. This will allow you to begin digitizing a polygon along the river channel margins of your study reach. It is essential for the algorithms that you use a polygon and not a separate path for each channel margin. Also avoid digitizing serious embayments and closed polygons along the shorelines as these will create unwanted spurs when running the Matlab algorithm. If you are doing multiple adjacent river reaches as separate polygons, make sure to provide a couple hundred meters of overlap as the Matlab routine will truncate and thin at the edges. Once the polygon is complete you should save it in Google Earth and then right click on it and go to **Save Place As** in the drop-down menu. Save it as a KML file. Note that all KML files exported from Google Earth are in Lat/Long WGS84 and will need to be converted into a projected coordinate system (e.g. UTM) for use with the Matlab algorithms.

POTENTIAL COMPLICATIONS/ CAUTIONS

The only complications in digitizing in Google Earth that we have run into is that you should always check to see which direction the active cursor is digitizing in relation to the last active point. Google Earth has a tendency to switch directions when you digitize

a polygon, save it, and then come back later to continue working on it again. Just make sure you always look at which point is active when you begin digitizing each session as this will save you some major headaches and time from not having to delete an hour of work because you neglected this!! Also note that the frequency and density of digitized points will depend on the channel characteristics and the goals of your project, but also will dictate the rate at which you can digitize a given channel distance. On average we have found that one can cover between 10-25 km of river length per hour in Google Earth for precise bedrock channels.

STEP 2: PREPARING THE DATA FOR MATLAB

Once you have exported your channel polygon as a KML file from Google Earth you need to (1) convert it to a shapefile, (2) change the projection to a projected coordinate system (i.e. UTM) from the unprojected, geographic coordinate system (WGS84), (3) rasterize the polygon, and (4) then export the data as a geotiff. We use ArcGIS 10 to do these steps, however, GRASS, QGIS, ENVI, and a number of other software packages have the same capability. Here we will detail the general steps taken and the importance of certain parameters.

(1) Converting KML to a Shapefile

This can be done in ArcGIS 10 by going to **Conversion Tools > From KML > KML To Layer.** This will convert your KML file into a layer file.

(2) Changing the Projection

Once you have your KML converted into a layer file you can easily export a shapefile from the layer file as well as reproject it into a projected coordinate system (e.g. UTM or an equal area projection), which is a necessity for the Matlab algorithms. We leave it up to the user to figure out the best way to do this, as there are many ways to perform this step in ArcGIS 10.

(3) Rasterizing the Shapefile

Once you have the reprojected polygon shapefile with the proper coordinate system you then need to rasterize the vector shapefile. This creates a matrix that can be easily loaded and worked with in Matlab. You will need to know what raster resolution you want to use and this will dictate the eventual width measurement spacing and error derived from the Matlab algorithms. As always, it is a trade-off between computational efficiency, accuracy, and the nature of your channel system. The rule of thumb we use is that at a minimum you should use a resolution of 1/3 the narrowest width in your channel polygon. So if the narrowest part of your polygon is 9 meters, you should use at least a raster resolution of 3 meters in order to accurately quantify the width throughout. Using a raster resolution of 1 meter is even better but will lead to longer computational times, require more memory, and generate larger file sizes. The processing efficiency will further depend on the computational hardware you have available. Nevertheless, this step is where the resolution will be set and will require some thought and possibly some trial and error.

To rasterize a shapefile in ArcGIS 10 go to **Conversion Tools > To Raster > Polygon to Raster**. It is imperative that you set the polygon area values to 1 because the Matlab scripts use binary filtering algorithms. You may need to add a column to the shapefile attribute table beforehand if there is no column with a value of 1 that you can use. This will then provide you a raster with values equal to 1, which can then be exported.

(4) Exporting a Geotiff for Matlab

You can now export out a geotiff file for use in Matlab by simply right clicking on the file in ArcGIS and selecting **Data > Export Data**. Another key is to make sure the background values are equal to 0. This is crucial. ***THE BACKGROUND VALUES MUST EQUAL 0*** and the ***CHANNEL POLYGON VALUES MUST EQUAL 1*** in the exported geotiff. Once the geotiff has been exported you are ready for Matlab to be opened.

POTENTIAL COMPLICATIONS/ CAUTIONS

This section is pretty straight forward just remember that the raster resolution matters and that the resulting geotiff must be 1's (for the channel area) and 0's (for background values) or it will not work in Matlab. Also it is always beneficial to get rid of any extraneous data in the geotiff, meaning you do not want to have a matrix that is 20000 x 20000 if the channel fits into an area that is 5000 x 5000 as it will bog down your computer and make really large geotiff files for no reason.

STEP 3: PROCESSING THE DATA IN MATLAB

Now that you have a raster geotiff exported you can begin to run the algorithms in Matlab. Before you begin make sure you have the "ChanGeom" folder downloaded on your computer in a directory that is recognized in your Matlab path settings. If it is not recognized simply go to **File > Set Path** in Matlab and set the "ChanGeom" folder to be recognized in your path files. This will make it so that whenever you run the functions they are immediately recognized without having to be in that specific directory. In the "ChanGeom" folder you should see the following 6 m-files:

changeom.m
chanextract.m
endpoints.m
endpoints_fcn.m
riv_msk_fil.m
riv_msk.m

The main two functions are changeom.m and chanextract.m, with the remaining m-files called in the chanextract.m function. We discuss the two main functions below.

Changeom.m

This is actually not a function but just an M-file to batch process multiple channel files at once and to quality control/troubleshoot the datasets and outputs. This M-file is well

commented and should serve as place to call and manipulate the input/output of the chanextract.m function.

Chanextract.m

This is the main algorithm responsible for calculating a centerline, width values, and cumulative distance upstream at each centerline point. The algorithm uses image processing filters in the *Image Processing Toolbox* as well as functions from the *Mapping Toolbox,* so make sure that you have these toolboxes available to you before running. The code is commented fairly well if you would like to know more specifics, but essentially it uses a binary thinning routine to delineate a centerline from the original rasterized channel polygon and then uses a quasi-euclidean distance to measure from the centerline to the margins of the channel at each centerline point. It then assigns a half-width value to each pixel in the centerline, which is simply multiplied by two to get the full width. The code is simple to run and can be run by typing the following into the command window in Matlab:

chanextract(inputtif,cellsize,exporttif,NoData,start_pt)

where the input variables are as follows,

inputtif = The binary rasterized channel polygon geotiff prepared previously. Must have background values of 0 and channel area equal to 1 for the algorithm to work. All projection information will be maintained.

cellsize = The raster cell size AKA the raster resolution (usually in meters)

exporttif = The name of the exported geotiff containing centerline and channel width data. This file will have width values associated with each centerline point and the same projection information as the inputtif.

NoData = The value of no data (NaN) values that you want when exporting the tif files at the end. NOTE: We suggest to use -9999 because this is the no data value in an ArcGIS environment, however, this will vary across different software environments. We don't recommend using values of 0 or greater as this may cause actual values to be lost later on during post-processing (e.g. if you use 0 the first point will be lost when you SetNull in ArcGIS).

start_pt = The stream segment beginning where you want the algorithm to begin calculating distance from (i.e. 0 meters) for each pixel along the computed centerline. It is either the southern tail (i.e. the start point is more southern than the end point) (start_pt = 0) or the northern tail (i.e. the start point is more northern than the end point) (start_pt = 1) of the dataset

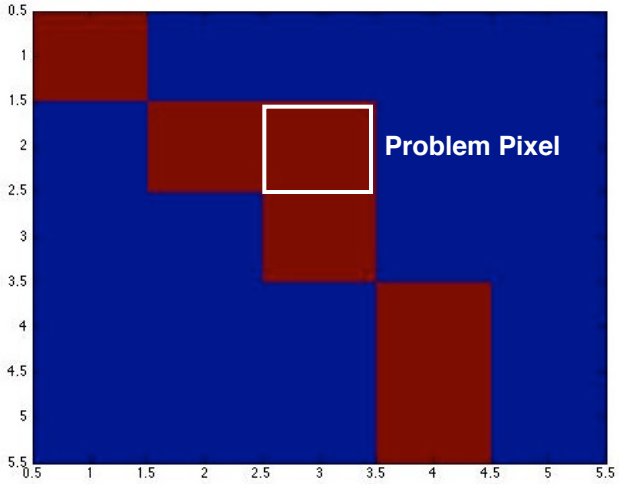
*(This was previously a separate function called centerlinedist.m)

EXAMPLE: chanextract('coloradoriver.tif',3,'coloradoriver_rwidth1.tif',-9999,0)

In this example the function would run, produce an image of the channel width output to view and check, save the channel width and centerline cumulative distance ***exporttif*** files ('coloradoriver_rwidth1.tif' and 'coloradoriver_rwidth1_cent_cumdist.tif'), and also save a .MAT file ('coloradoriver_rwidth1.mat') containing all the variables from the function. The geotiff file can then be easily brought back into your GIS environment.

POTENTIAL COMPLICATIONS/ CAUTIONS

The only problems that we have noticed using the chanextract.m algorithm is when the ***inputtif*** *file is not binary, in which case you will receive a cryptic Matlab error message. Always make sure the changeom.m file is in your Matlab path and that the* ***inputtif*** *file is a binary file of 1's (for the channel area) and 0's (for background values).The other problem that we have found is that when there are places where three pixels are adjacent instead of two (which can occur based on the thinning algorithm in changeom.m) the algorithm will stop there. This is rare but does happen from time to time. The easiest step is to look at the centerline_cumdist variable in the MAT file and it is generally quite obvious and will look something like the figure shown to the right. In changeom.m there is a little script where you can simply set the extra pixel to no data in the centerline variable and then rerun the second part of the chanextract.m algorithm manually (look for capitalized comment and triple lines) and you will be all set. We also recommend setting the same pixel to 0 for the channel width variable (rwidth1) and then re-exporting (not re-running!). NOTE if you rerun all of chanextract.m again it will just remake the same mistake that you just corrected. (NOTE this error is has largely been fixed in v0.3)*



OTHER VARIABLES OF INTEREST

centerline_numbers contains upward counting numbers for each pixel along the centerline (e.g. 0,1,2,3,4.....)

centerline_distance contains distances between each pixel using either 1 x cellsize or the sqrt(2) x cellsize.

centerline_cumdist integrates centerline_distance creating a cumulative distance from the start point along the centerline path (this is what we most commonly use)

STEP 4: POST-PROCESSING THE DATA IN ARCGIS

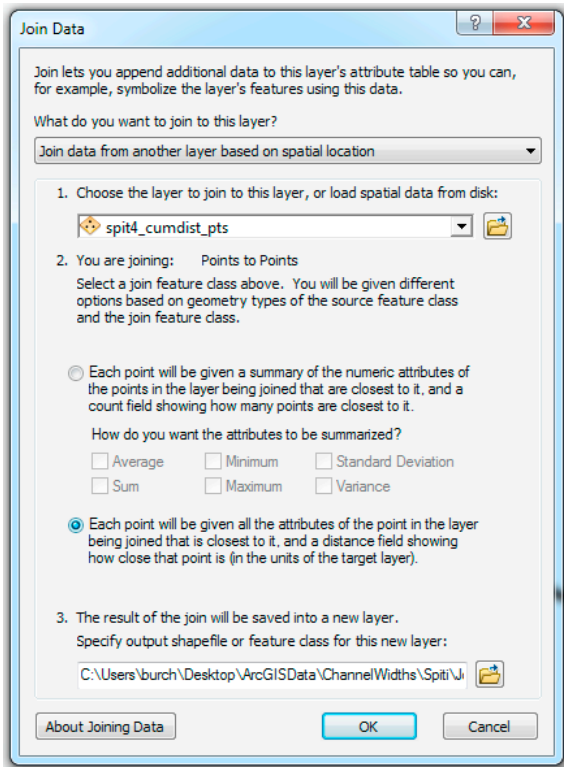
You now have rasters that can be easily loaded back into ArcGIS (geotiffs) delineating the channel centerline with width values and cumulative distance upstream for each centerline point. You can now begin working with the data in whichever way you see fit,

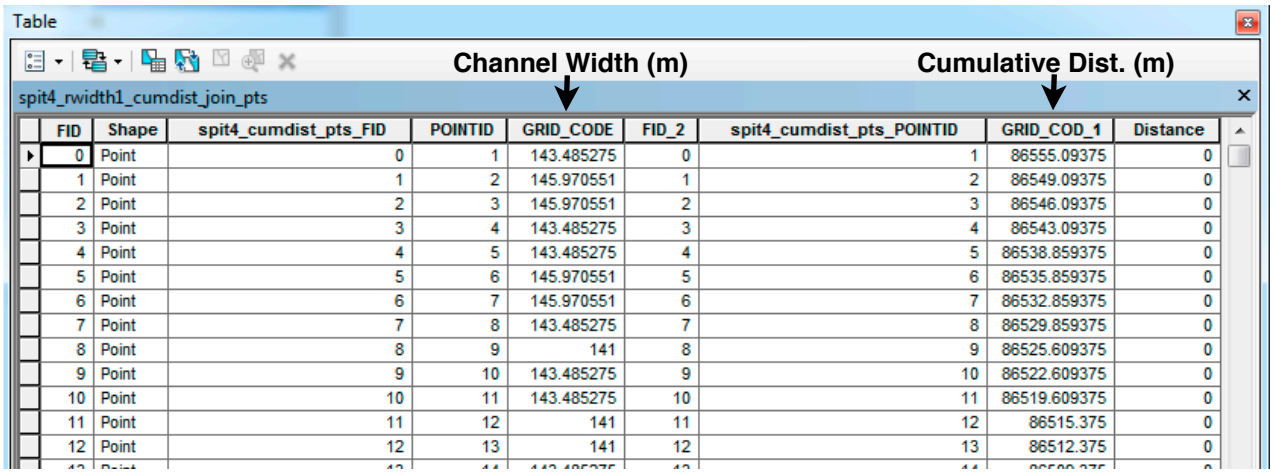
but we will briefly just mention some of the basic steps that we perform post-processing that makes working with the data easier.

Vectorizing and Joining Datasets

First, we always verify that the dataset is comprehensive and matches the spatial area of the original channel shapefile. If it doesn't you know there was a mistake somewhere along the way. If it matches and you are happy the next step is to make sure all of the background values equal NoData so that when you vectorize the raster it does not make points for every background point. If you used a -9999 value for your NoData value in the export then you should be ok, though ArcGIS is known for its quirkiness! Otherwise, there are a number of ways to set the background values to NoData in ArcGIS (setnull, raster calculator, export, etc).

Next we suggest to convert the raster dataset into a point shapefile where each centerline pixel is turned into an individual point. This is achieved in the ArcGIS toolbox by going to **Conversion Tools > From Raster > Raster to Point**. We do this for both the channel width and centerline cumulative distance datasets so that we can then join them into one shapefile. This is done by right clicking on the channel width point file in the Table of Contents and selecting **Joins and Relates > Join**. Once the dialogue box is open select "*Join data from another layer based on spatial location*" then select the centerline cumulative distance point shapefile to join to that layer (#1). Next, select the bottom option in part #2, which joins the point





FID	Shape	spit4_cumdist_pts_FID	POINTID	GRID_CODE	FID_2	spit4_cumdist_pts_POINTID	GRID_COD_1	Distance
0	Point	0	1	143.485275	0	1	86555.09375	0
1	Point	1	2	145.970551	1	2	86549.09375	0
2	Point	2	3	145.970551	2	3	86546.09375	0
3	Point	3	4	143.485275	3	4	86543.09375	0
4	Point	4	5	143.485275	4	5	86538.859375	0
5	Point	5	6	145.970551	5	6	86535.859375	0
6	Point	6	7	145.970551	6	7	86532.859375	0
7	Point	7	8	143.485275	7	8	86529.859375	0
8	Point	8	9	141	8	9	86525.609375	0
9	Point	9	10	143.485275	9	10	86522.609375	0
10	Point	10	11	143.485275	10	11	86519.609375	0
11	Point	11	12	141	11	12	86515.375	0
12	Point	12	13	141	12	13	86512.375	0

closest to each point in the channel width layer. Because they are spatially the same dataset and each point overlaps, the join distance will be 0 and each channel width point will get its own distinctive cumulative distance value. Lastly, save the file (#3) and the resulting attribute table should look similar to that above.

The dataset can now be merged with other DEM derived characteristics, such as drainage area and channel slope or even fused with these datasets to improve erosional proxies (kS or specific stream power) and reach scale slope values from coarser DEMs, such as ASTER and SRTM datasets (*Fisher et al., 2013*).

POTENTIAL COMPLICATIONS/ CAUTIONS
The only problem that we have noticed during the post-processing of data is that sometimes ArcGIS does not acknowledge the background values of -9999 as NoData (likely related to some other characters being written into the file). If this is the case, it means you need to do the extra step of setting the background values to NoData using SetNull before you convert it to a point file. For alternative GIS software platforms you will need to figure out what the NoData value is and set it when you call the Matlab functions.

PLEASE DON'T HESITATE TO CONTACT US WITH ANY QUESTIONS OR COMMENTS PERTAINING TO ANY ASPECT OF THIS MANUAL, SOFTWARE, OR ANALYSES. WE ARE ALWAYS WILLING TO HELP, COLLABORATE, AND LEARN.

SINCERELY,
BURCH AND BODO