2015

Development of an Automated Method for Identification of Wet and Dry Channel Segments Using LiDAR Data and Fuzzy Logic Cluster Analysis

Chris Rowney
University of Central Florida

DEVEOLOPMENT OF AN AUTOMATED METHOD FOR IDENTIFICATION OF WET AND DRY CHANNEL SEGMENTS USING LIDAR DATA AND FUZZY LOGIC CLUSTER ANALYSIS

by

CHRISTOPHER M. ROWNEY
B.S. Florida State University, 2007
B.S. University of Central Florida, 2012

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Civil, Environmental, and Construction Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2015

Major Professor: Dingbao Wang

ABSTRACT

Research into the use of LiDAR data for purposes other than simple topographic elevation determination, such as urban land cover classification and the identification of forest biomass, has become prominent in recent years. In many cases, alternative analysis methodologies conducted using airborne LiDAR data are possible because the raw data collected during a survey can include information other than the classically used elevation and coordinate points, the X, Y, and Z of the plane. In particular, intensity return values for each point in a LiDAR grid have been found to provide a useful data set for wet and dry channel classification. LiDAR intensity return data are, in essence, a numeric representation of the characteristic light reflectivity of the object being scanned; the more reflective the object is, the higher the intensity return will be. Intensity data points are collected along the course of the channel network and within the perceived banks of the channel. Intensity data do not crisply reflect a perfectly wet or dry condition, but instead vary over a range such that each location can be viewed as partially wet and partially dry. It is advantageous to assess problems of this type using the methods of fuzzy logic. Specifically, the variance in LiDAR intensity return data is such that the use of fuzzy logic to identify intensity cluster centers, and thereby assign wet and dry condition identifiers based on fuzzy memberships, is a possibility. Membership within a fuzzy data set is characterized by a value representing the degree of membership. Typically, membership values range from 0 (representing non-membership) through 1 (representing full membership), with many observations found to be not at either extreme but instead at some intermediate value representing partial membership. The ultimate goal of this research was to design and develop an automated algorithm to identify wet and dry channel sections, given a previously identified

channel network based on topographic elevation, using a combination of intensity return values from LiDAR data and fuzzy logic clustering methods, and to implement that algorithm in such a way as to produce reliable multi-class channel segments in ArcGIS. To enable control of calculations, limiting parameters were defined, specifically including the maximum allowable bank slope, and a filtering percentage to more accurately accommodate the study area.

Alteration of the maximum allowable bank slope has been shown to affect the comparative quantity of high and low intensity centroids, but only in extreme bank slope conditions are the centroids changed enough to hamper results. However, interference from thick vegetation has been shown to lower intensity values in dry channel sections into the range of a wet channel. The addition of a filtering algorithm alleviates some of the interference, but not all. Overall results of the tool show an effective methodology where basic channel conditions are identified, but refinement of the tool could produce more accurate results.

I dedicate this work to my father, my inspiration to become an engineer.

ACKNOWLEDGEMENTS

I want to thank Doctor Dingbao Wang for all his help and encouragement and for becoming my advisor on such short notice. I have learned a lot from him and all my teachers at UCF.

I would also like to thank my advising committee; Doctor Stephen Medeiros and Doctor Kelly Kibler, for their time and knowledge.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Lidar Analysis

Light detection and ranging (LiDAR) is a method of remote sensing where laser light is used to gauge distance to an object. While the primary purpose of airborne LiDAR data is the generation of highly accurate elevation models (NOAA, 2012), research into the use of LiDAR data for the determination of alternate land characteristic identification models, such as urban land cover classification (Chen & Gao, 2014) and the estimation of forest biomass (Gleason & Jungho, 2012), has become prominent in a variety of scientific fields. In many cases, the alternative analysis methodologies conducted using airborne LiDAR data are possible because the raw data collected during a survey can include information supplemental to the classically used elevation and coordinate points, the X, Y, and Z of the plane. The inclusion of intensity return, the number of returns per area, the time of the returns and even the angle at which the scan was taken, can all help identify telling characteristics for a given survey area (NOAA, 2012). In particular, intensity values for each point have been found to be a useful data set for wet and dry channel identification (Kim, Wang, & Medeiros, 2015).

Intensity Analysis

LiDAR intensity return data are, in essence, a numeric representation of the characteristic light reflectivity of the object being scanned; the more reflective the object, the higher the intensity will be when it returns to the sensor. As with ordinary light, objects that are lighter in color are more reflective to LiDAR than those with darker colors and tend return higher intensity values. Accordingly, and cover such as grass and tree tops have a high intensity return, while

building rooftops and road ways tend to have lower intensity returns because of their typically darker coloring.

Reflectivity is not the only factor affecting intensity returns. In the case of treed areas or areas of thick vegetation, LiDAR laser signals which pass between the upper canopies can be partially obscured for their return, effectively lowering the intensity of what is detected at that point. Water bodies, on the other hand, tend to scatter the LiDAR laser beam when it contacts the surface (Brzank & Heipke, 2006). This leads to a low or null intensity return for water bodies. It is because of this low intensity return for water surfaces, such as streams and rivers, that a methodology for identifying wet or dry channels was introduced by Kim et al. (2015).

Kim et al. (2015) proposed a methodology for identifying wet and dry channels using LiDAR intensity information. Intensity data are collected from the LiDAR grid over the course of the channel network and within the perceived banks of the channel. The distribution of the intensity points collected is then analyzed. The intensity value, where the probability of a channel section being wet or dry is equal, represents the threshold where a channels state changes from being more likely to be wet to more likely to be dry, or vice versa. Two other points representing when an intensity return would be considered fully dry or fully wet are identified from the distributions.

If the implementation of the method by Kim et al. (2015), as described above, into a computer model is the ultimate goal, certain alterations must be made for it to be viable. The use of points known to be wet or dry, based on their location within the channel network or by visually identifying very low and very high intensity return points is difficult to implement within the logic structure of a computer, as the program would have to be taught what a

relatively high and relatively low set of intensity values would look like. This level of subjective reasoning tends to be beyond a computer's ability to understand. For this reason, the inclusion of fuzzy logic, as a means of identifying intensity return patterns and clustering, was deemed advantageous.

Fuzzy Logic

Originally introduced by Zadeh, in a paper titled "Fuzzy Sets," fuzzy logic has grown into a computational methodology which does not rely on crisp data sets in order to identify likely outcomes (Zadeh, 1965). In the paper, Zadeh states that, often times, real world problem sets do not have strict "precisely defined criteria for membership." In essence, Zadeh explains that membership within a fuzzy data set is characterized as a gradient, with 0 representing non-membership and 1 representing full membership. Decimals between 0 and 1 represent a gradual increase of membership with the group. An item which has a value of 0.8, for example, is more a member of the group than an item with a value of, say, 0.2. Furthermore, an item can be considered to be a member of two groups at one time. A classic example is a window; when not fully open or closed, it may be considered to be partially open and partially closed at one and the same moment. In contrast with classic statistical methods that consider a possibility that the window may be open or may be closed at a moment in time, fuzzy logic inherently positions the problem in a way that accounts for the simultaneous existence of both conditions to some degree at a moment in time.

This characteristic makes fuzzy logic a useful way to approach the current problem, because in many cases rivers do not instantly transfer from a fully wet state to a fully dry state. Further, where they have a wet section and a dry section, the transition between them may not be

delineated by some sharp boundary. It is often the case that somewhere along their length they transition from wet to dry over a distance, displaying intermittent wet or dry sections over that distance. A segment of the river in this transition area has some wetness and some dryness. Put another away, there is not always a sharp line below which the river is wet, and above which the river is dry. In some cases, for example when the stream emerges from a spring, such a sharp boundary may be found; however, in many channel networks the existence of such a convenient demarcation is unlikely.

This complicates the use of LiDAR data to determine which length of a stream is wet and which is dry. The return values, in keeping with the physical problem, do not crisply define a point at which this transition can be said to occur. Instead, values become fuzzy along some distance, making it difficult to define a point where the river can be said to be wet or dry, or even to be in some intermediate state. Since it is of interest in some situations to estimate wet and dry reaches, it would be useful to have a method that can be used to estimate these conditions using LiDAR data.

In the confines of the intensity return problem, fuzzy logic can be used to identify the ranges of intensities for which a dry or wet channel could be included. Because the intensity return values have no set identifying values for "wet" or "dry," but a range of data where lower intensity returns are likely to be wet and higher intensity returns are likely to be dry, the use of a binary system where inclusion into a category is either true or false is challenging. Furthermore, just as there is no defined set of values for "wet" or "dry" the variation between these two states has no clear demarcation. The use of fuzzy logic changes what would be a binary "true" or

"false" response to "wet" or "dry" into an identification of membership for a specific data point into either condition.

Goal

The ultimate goal of this research was to design and develop an automated methodology to identify wet and dry channel sections, given a known channel network from topographic elevation, using a combination of intensity return values from LiDAR data and fuzzy logic clustering methods and then to produce reliable multi-class channel shape files in ArcGIS using said method. The classes to be identified were based on fuzzy logic membership to each data set. The classes of the shape files identify a channel section as "wet," "mostly wet," "dry," and "mostly dry" where "wet" and "dry" will have full membership into their respective clusters, but "mostly wet" and "mostly dry" will only have partial membership into their respective clusters.

LITERATURE REVIEW

LiDAR data are commonly used to determine land surface elevation, but beyond this have been found useful in a multitude of identification and classification techniques to determine characteristics other than elevation. For instance, LiDAR technology has been useful in archaeological work to identify Roman water systems in Spain (Fernandez-Lozano, Gutierrez-Alonso, & Fernandez-Moran, 2015). Schumacher et al. has developed an integrated LiDAR and precipitation model for estimated tree canopy throughfall in Denmark (Schumacher & Christiansen, 2015). In addition, LiDAR data were used to identify three previously unknown postglacial faults in Northern Finnish Lapland (Sutinen, Hyvonen, Middleton, & Ruskeeniemi, 2014). Classification methods for urban and natural land use have also been studied extensively for a variety of applications. For instance, a method of filtering LiDAR intensity data to remove extraneous data points for land cover classification has been studied (Song, Han, Yu, & Kim, 2002).

As airborne LiDAR is an infrared laser attached to an aircraft (NOAA, 2012), the intensity data derived from it are prone to variations based on incidence angle, range to target and the light absorbency of the surface the beam hits (Wolf & Zissis, 1978). Because of this variance, methods for the correction of the recorded intensity have been studied. Calibration of LiDAR data using commercially available substances, such as gravel, is one method that been shown to alleviate some error within the data (Kassalainen, et al., 2009). Another method is radiometric correction and normalization of LiDAR intensity returns for improving land cover classification by making corrections based on surface slope thresholds and normalizing overlapping intensity data, which was shown to result in a 16.5 percent increase in accuracy.

(Yan & Shaker, 2014). Furthermore, Jutzi and Gross (2009) showed that normalization of range and incidence angle can improve accuracy when using intensity values for analysis. However, Yoon et al (2008) showed that range is a major factor in the variance of intensity for most surfaces, and that normalization of vegetation without accounting for range would cause an over correction. This information corroborates a study in 2002 showing that an increase in vegetation density decreases the accuracy of the identification of terrain types (Bowen & Waltermire, 2002).

The classification of urban objects, using a combination of various imaging methods and LiDAR data, has been studied extensively to determine its effectiveness as a means of data analysis. Awrangjeb et al. (2010) for instance, used a combination of color orthoimagery and LiDAR data to identify residential buildings. Alonzo et al. (2014) integrated LiDAR data with hyperspectral imaging to identify 29 different tree species within an urban setting. In fact, urban land cover classification accuracy increases when a combination of LiDAR structural and intensity surface models is used. In addition, a finer spatial resolution tends to produces a higher accuracy (Singh, Vogler, Shoemaker, & Meentemeyer, 2012). Digital elevation models (DEM) derived from LiDAR data were consistently more accurate when compared to their reference data when the LiDAR cell size was smaller. However, higher resolution DEM grids, when used with streamflow simulations, do not always increase accuracy of the models (Yang, et al., 2014). Land cover classification has been improved via integration of LiDAR Digital Surface Models with LiDAR intensity, where an accuracy of 90.7%, was attained for the classification of buildings, pavement, trees/shrubs and grass (Zhou, 2013). A high degree of accuracy (80 to 100 percent) was found when using airborne LiDAR data for urban land cover classification with no

other data sources (Chen & Gao, 2014). A case study in 2014 in Nanjing, China found that urban vegetation extracted using LiDAR and intensity data also had a high degree of accuracy, at 94.6% (Han, Zhao, Feng, & Chen, 2014). Structure from motion and LiDAR was combined to model urban flooding where the addition of structure from motion increase accuracy as it identifies hidden objects otherwise missed by LiDAR (Meesuk, Vojinovic, Mynett, & Abdullah, 2015). Indeed, a multitude of processes involving LiDAR data for classification and identification purposes have been proposed in recent years.

Natural feature classification using LiDAR data has also been an extensively researched topic. Antonarakis et al. (2008) used a combination of elevation and intensity LiDAR data to classify ground and forest types. Improvement of estimates of forest carbon stock in Kalimantan using LiDAR point clouds has proven effective (Kronseder, Ballhorn, Bohm, & Siegert, 2012). While it was only 75% accurate, LiDAR intensity data were used for mapping lichens in forest understories (Korpela, 2008). Multispectral LiDAR data has improved accuracy when used to identify rock types in geological outcrops (Hartzell, Glennie, Biber, & Khan, 2014). The monitoring of hydromorphology and human intervention in the River Carron and Forth estuary in Scotland, using a combination of hyperspectral imagery and LiDAR data, has been shown to be a useful tool (Gilvear, Tyler, & Davids, 2004). Identification of morphologically distinct channel bed segments, using LiDAR data has been used to differentiate total river morphology (Cavalli, Tarolli, Marchi, & Fontana, 2008) and identify fluvial terraces (Val, Iriarte, Arriolabengoa, & Aranburu, 2014). Roughness calculations for floodplains have been improved using LiDAR, as the data can be used to calculate spatially distributed roughness information as opposed to constant roughness for the floodplain (Abu-Aly, et al., 2014). Water and land

boundaries have also been distinguished using LiDAR intensity data, for various river sections in Austria (Hofle, Vetter, Pfeifer, Mandlburger, & Stotter, 2009). An automated method to determine riparian zone ecological health in Belgium and northern France was also found to be effective (Michez, et al., 2013).

It is clear that a number of classification methods, such as an expectation-maximization algorithm used to identify roads, grass, buildings and trees (Lodna, Fitzpatrick, & Helmbold, 2007) or land cover classification using Maximum likelihood Gaussian process for use with hyperspectral imaging (Jun & Ghosh, 2011) can be effective. However, it is also clear variant conditions within a LiDAR dataset, including intensity changes as a consequence of changes in the incidence angle and range to target, can complicate the problem and lead to conditions where transition between one state and another is poorly defined, and it is difficult to confidently conclude a point on the surface is in one state (i.e. type of land cover) or another. This can be addressed as a probability (i.e. the point is likely to have a certain cover) or by means of fuzzy logic (i.e. the point is to some degree has a certain type of cover, and to some degree does not).

As noted above, this reality suggests consideration of fuzzy logic as a means of describing and analyzing the surface. The introduction of fuzzy logic in 1965 by Lotfi Zadeh showed an alternate problem solving structure, which did not rely on crisp data sets to solve problems (Zadeh, 1965). Zadeh posited that real world data do not typically have rigidly defined criteria and, as such, the methods used with the data should not have rigidly defined outputs. After the initial introduction to fuzzy logic, Zadeh and others have expanded upon the idea, identifying fuzzy logic based algorithms for use in a variety of fields. Zadeh went on to introduce fuzzy algorithms as a methodology, in which algorithms that use crisp data sets could

be altered to include fuzzy data sets or fuzzy algorithms could be developed independently (Zadeh, 1968). He then went on to introduce the concept of similarity relations and fuzzy orderings, in which a similarity relation is identified as reflexive, symmetric and transitive while a fuzzy order is a transitive fuzzy relation (Zadeh, 1971). Type 2 fuzzy logic systems were developed where the rule set can also have uncertainty (Karnik & Mendel, 1998). Type 2 fuzzy logic was expanded upon to help with the identification of clustering and pattern recognition where higher degrees of uncertainty compared to type 1 fuzzy logic systems exist. However, Type 2 fuzzy logic systems were found to be computationally time consuming (Melin & Castillo, 2014). A fuzzy logic based soil classification system using a similarity representation system was shown to improve soil survey efficiency (Zhu, Hudson, Burt, Lubich, & Simonson, 2001). Along a similar vein, fuzzy logic was used for reconnaissance-scale mapping of acid sulfate soils on the Finnish coast, where it was found to be a useful addition to large scale preliminary surveys (Beucher, Frojdo, Osterholm, Martinkauppi, & Eden, 2014). A fuzzy logic approach was also implemented for habitat mapping, resulting in a robust methodology (Petrou, et al., 2014).

As the transition between variant urban or natural objects can often be one which is not crisp, classification methods are a prime candidate for a fuzzy logic approach. For instance, Shackelfor et al. (2003) used a combination of fuzzy logic and multispectral imagery for classification of urban and suburban areas. The use of fuzzy logic was found to increase classification accuracy, when compared to a more traditional maximum-likelihood method. In the case studied, the accuracy increase was found to be between 8 and 11 percent. Fusion of

hyperspectral imagery with LiDAR data, with the inclusion of a fuzzy logic classifier system, has also proven to be effective (Bigdeli, Samadzadegan, & Reinartz, 2015).

As with the variance within LiDAR intensity return values for urban and natural objects, rigidly defined wet and dry conditions within a channel network are not feasible. Brzank et al. (2006) implemented a fuzzy logic approach to the identification of coastal water or land points. Membership values are calculated using a combination of height, slope, intensity, missed points, segment length, and point density, where the thresholds are identified as strictly monotonic alteration to the basic function. As the monotonic functions identify fully wet or dry conditions, these thresholds are set to 0 and 1 respectively. Application of this method was then implemented in another study in which the Wadden Sea was a test case for the identification wet and dry transitional points within tidal channels and depressions (Brzank A. , Heipke, Goepfert, & Soergel, 2008). While other methods of identifying water-land transitional points have been used and found to be effective (Smeeckaert, Mallet, David, Chehata, & Ferraz, 2013), the application of fuzzy logic has proven to be a useful tool within the context of LiDAR point identification and lends itself to the flexibility needed for use with intensity data.

While a combination of fuzzy logic and LiDAR intensity returns has been used for the identification of the transition point between wet and dry conditions within a model area, these methods have been used when there is a high degree of likelihood that a transition point between wet and dry exists. In particular, these methods use membership criteria between both wet and dry conditions to identify a fuzzy boundary between the two states. Applying fuzzy logic clustering techniques to definition of wet or dry sections within a channel network where there is no guaranty of either condition being prevalent is lacking.

Bezdek et al. (1984) introduced a clustering algorithm called Fuzzy c-means clustering in which a clustering analysis is performed using relative membership to the centroid of a given cluster. By automating the calculation of the centroids of the clusters that identify wet or dry conditions within the channel network, using fuzzy c-means clustering, and staying within the channel bed boundaries to minimize intensity variance due to vegetation, then identifying the fuzzy membership that points along the channel network have to the cluster centroids, a relationship between what is considered a wet or dry condition and the points should be attainable and time efficient. As no single method has been found to be effective for identify channel bank slopes (Vianello, Cavalli, & Tarolli, 2009), the method will rely on a known maximum allowable bank slope.

METHODOLOGY

Initial Program Characteristics

The nature of computer programming is such that subjective elements are incredibly difficult to impossible to program as they require a flexibility of behavior that has yet to be achieved in mainstream computers. In the case of channel network analysis, visually identifying channel sections from which to draw conditional data to test is difficult. To this end, more objective methods were developed for initial intensity analysis test sites, bank slope determination, high and low intensity identification, and channel type partitioning. While these alterations differ from the original methodology proposed by Kim et al. (2015), the basic premise of wet or dry stream identification based on LiDAR data should still hold.

Initial designs required three user inputs, a digital elevation model (DEM) raster file, an intensity raster file and a shape file of the channel network. Subsequent calculations involving specific channel locations were conducted using the channel network map as a basis. Any other data required from the surrounding area of individual channel locations were acquired using an X and Y coordinate system and the cell size of the intensity raster file as the step size.

As it is difficult to automatically identify certain conditions inherent to a specific channel network, the addition of three extra parameters designed to specify the output data to the region in which the program was run First, the inclusion of a maximum bank slope parameter was identified as an effective user input, as it allows the user to distinguish specific identifying conditions within the region that could alter the outcome of the initial intensity analysis. As bank characteristics are dependent on the geomorphology of a region, especially in ephemeral streams, the addition of this user defined parameter would help improve the final high and low

intensity analysis by more effectively defining the channel bank within the region being studied. Additionally, the bank slope will be calculated using consecutive cells, based on the intensity map. This should help alleviate identifying minor variations in the channel bed as a bank.

Because the calculations to determine if a channel segment is wet or dry are based on the intensity return value of that segment, any object that is capable of absorbing the LiDAR light sufficient to lower the return value into the range of what is considered wet can cause channel sections, that would otherwise be identified as fully dry, to be seen as partially wet. For this reason, the addition of a filtering algorithm based on the dry percentage of a channel segment was deemed to be a necessary addition to the program. The user selects the percent threshold for which the channel segment will be considered to be all dry. If the section exceeds this user defined percentage, the entire channel section will be considered dry.

Secondary to the inclusion of the bank slope and filter percentage user inputs, an input allowing the user to change the percentage of each channel segment that is used when identifying wet or dry conditions. As each channel segment of the original channel network is of differing lengths, using small percentages while stepping through the line segments will give increased resolution when identifying wet and dry conditions. It should be noted that this assumption holds true until the minimum pixel resolution is reached, at which point there is no increase. However, as the percentage decreases the number of segments analyzed increases, which lead to increased program run times. For instance, if a channel network shape file originally has 1000 polyline segments and the code is set to step through each line segment at 1% intervals to identify wet or dry conditions, each polyline segment has 100 intervals to analyze. This results in a final tally of line segments in the output files of 100,000. While computers can run through

analysis quickly, 100,000 line segments would take approximately two and a half hours to complete on currently available consumer computer hardware. The addition of a user percentage adjustment would allow for less accurate, shorter analysis times.

Initial Intensity Analysis

As channel network shape files are typically made up of a large number of line segments, finding a test site for each line segment produces a relatively large population of intensity values. Sites were chosen at distances of 50% of the total length of the line segment. As data points are taken from each line segment, intensity returns for the entire channel network are more likely to be representative of conditions in the channel network, as a whole.

After the initial points within the stream network were identified, each point was analyzed using a cluster analysis technique called fuzzy C-means (Bezdek, Ehrlich, & Full, 1984). The use of a fuzzy logic method for identification of the centroids of the intensity clusters was decided upon because of the nature of the intensity values. As stated on the ArcGIS resources website by ESRI, "Intensity is relative, not quantifiable, therefore you cannot expect the same value off the same target from flight to flight or from elevation to elevation" (ESRI, 2014). In short, the expectation that the intensity values will be the same between maps does not hold, because the intensity values are only relative to themselves and no other dataset, even a dataset of the same area. The use of fuzzy logic allows for the approximate and relative nature of each dataset by identifying the relative membership of an intensity value to the cluster instead of a binary membership whereby the individual intensity values are either included in one group or another (Ross, 2010).

In the case of intensity classification, fuzzy C-means cluster analysis was used to identify a predefined number of intensity centroids within the dataset collected from the channel network and intensity raster files. As the intensity values required for analysis are the relative high intensity and relative low intensity within the channel network, the number of centroids to be identified is permanently set to two for the program. Unlike K-means cluster analysis, one of the more popular cluster analysis methods which identifies a data point as belonging entirely to the centroid it is closest to, C-means identifies the membership of a data point to the clusters that surround it (Ross, 2010).

The fuzziness (m) of the calculation will be set to 2, which will allow for a high degree of fuzziness between the datasets. As previously stated, the number of centroids (j) to be identified, in this instance, is 2. As a starting location is required to begin the calculations, the maximum and minimum intensity values found within the dataset are used as starting centroids. Because there is only one parameter to calculate, the intensity, the cluster analysis is simplified into two dimensional space. The distance (d_{ij}) between the calculated centroid (c_j) and the current intensity value (x_i) is a two dimensional line, and is the absolute difference between the two values (equation 1).

$$d_{ij} = \left|c_j - x_i\right| \quad (1)$$

The membership between any given intensity value and the maximum and minimum centroids is given as the membership of an intensity value to a specific centroid, divided by the sum of the individual memberships to the intensity value to each centroid (equation 2).

$$u_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m-1}}} \quad (2)$$

The each centroid is then calculated as the sum of the memberships, to the power of the fuzziness, of a point to that centroid, times the intensity value, divided by the sum of the memberships, to the power of the fuzziness (equation 3).

$$c_j = \frac{\sum_{i=1}^{n} u_{ij}^{m} x_i}{\sum_{i=1}^{n} u_{ij}^{m}} \quad (3)$$

These three equations form an iterative process that alters the centroids until ending conditions are met. When the difference between the new centroids and the previous set of centroids is minimal, 0.01 in this case, the maximum and minimum centroids are considered determined (Ross, 2010).

After the local maximum and minimum centroids are determined for a channel line section, the values are placed in an array. Upon completion of the calculations for the final channel line section, a second fuzzy C-mean cluster analysis is applied, using the collected local maximum and minimum centroids, to determine the ultimate maximum and minimum centroids, for the channel network.

Figures 1 through 2 are program flowcharts depicting the initial intensity analysis logic path. Figure 1 shows the details of the program section. More specifically, Figure 1 exhibits a visual representation of the initial point selection section of the program. The program first identifies the output file name and if it exists, ends the program. Next, the program finds the mid-point of each line section, finding the elevation and intensity values for each cell, as it moves out from the center. Finally, the code exits the loop after the final line segment has been tested and identifies the local maximum and minimum intensities. Figure 2 shows the specific steps of the fuzzy C-means cluster analysis, as it is used in the program. The program uses equations (1) and (2) in an iterative format, stepping through each intensity value in the array,

until each data point has been accounted for. Next, the program identifies the centroids and determines if the difference between the current centroid and the last centroid are less than or equal to 0.01, at which point the code exits out of the loop and continues on to the next section of code. The full code can be found in the appendix if a more in depth view of the program functions is desired.

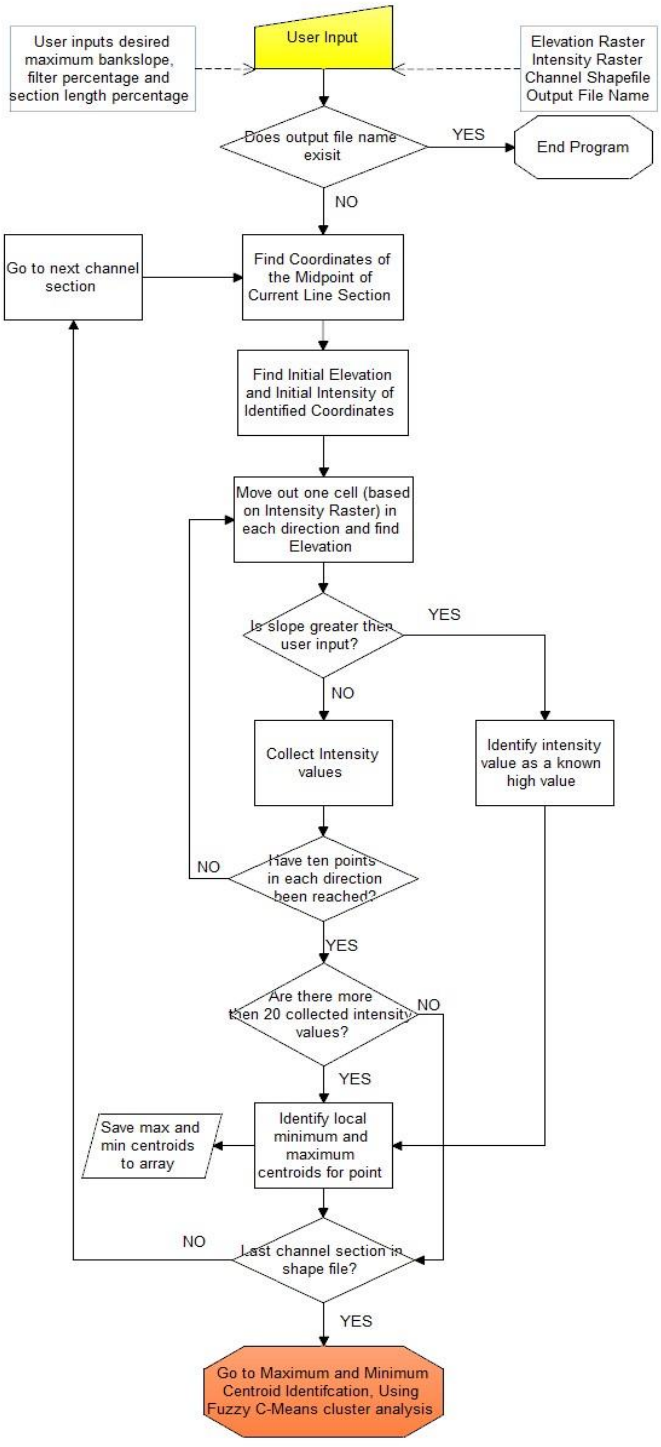

Figure 1: Program Flow Chart, Section 1

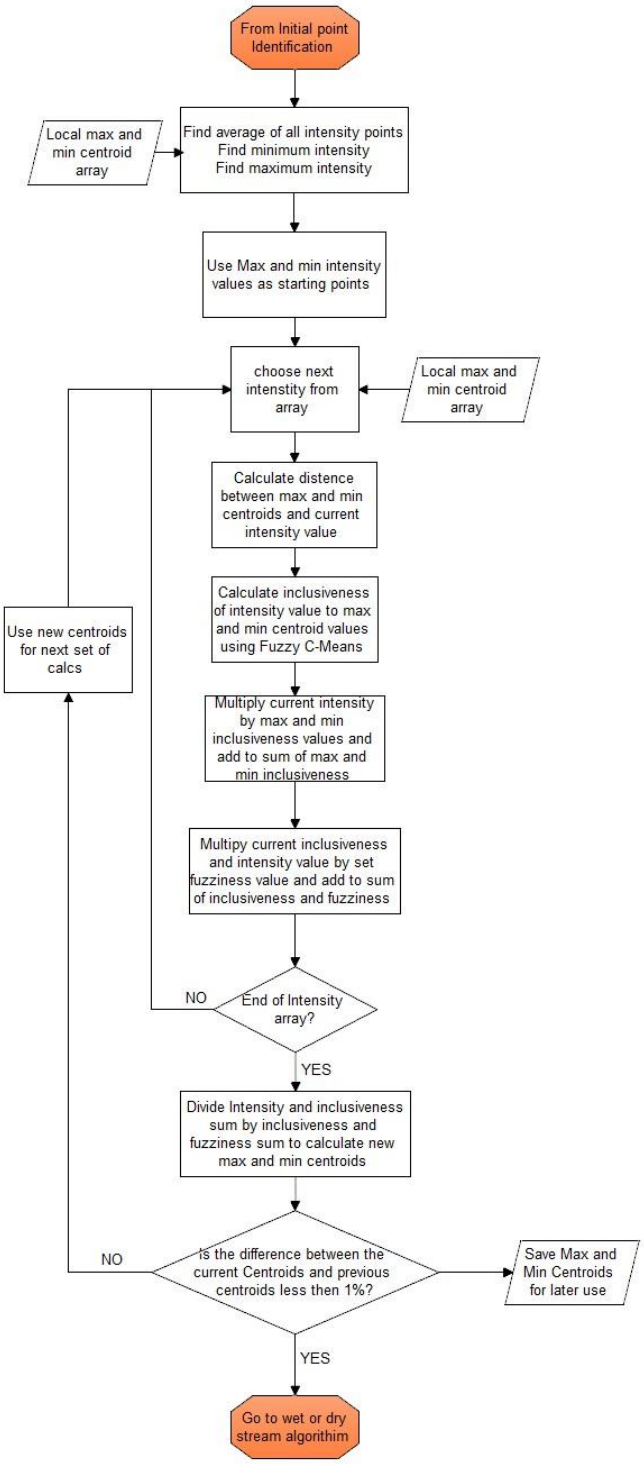

Figure 2: Program Flow Chart, Section 2

Channel Network Analysis

After the ultimate maximum and minimum centroids are determined for the channel network, the program returns to the beginning of the channel network to begin analyzing the likely condition of the channel. Each step through the line sections are classified as one of 4 categories: wet, dry, mostly wet or mostly dry. Each category is accounted for separately and creates individual shape files after the program finishes.

As seen in Figure 3, from the beginning of the channel network, the program begins by stepping through each line segment at a user specified increment, set as a percentage. For instance, if the user sets the step through percentage as 10 percent the line segment will be sectioned into 10 parts. Starting with the beginning of each line increment a node is tested to determine the intensity value at that point. The intensity value is then compared to the maximum and minimum intensity centroids. If the current intensity is larger than the maximum intensity centroid, the line increment will be identified as dry. Conversely, if the current intensity return is smaller than the minimum intensity centroid, then the line increment will be identified as wet. At this point, the same fuzzy logic algorithm for determination of membership into the maximum and minimum centroids is used for classification as either mostly wet or mostly dry. As the membership values returned from equation (1) are decimal values between 0 and 1, where a value has a higher inclusiveness to the centroid as the number reaches 1, and there are only two centroids to test against, the sum of the membership values for both the maximum and minimum centroids equal 1. If the membership equation, when applied to the minimum centroid, showed greater than or equal return value of 0.5, the section interval was identified as mostly wet

whereas. However, if the membership equation, when applied to the maximum centroid, showed a return value greater than 0.5, the section interval was identified as mostly dry.

The addition of a dry channel filtering algorithm was deemed to be beneficial, as it allows the user to select the level at which the program stratifies the dry channel sections. As trees and dense flora can absorb LiDAR, the returns can appear as low as wet or mostly wet, when being identified by the program. Because of this, the dry channel filtering algorithm was designed to help alleviate small errors, in forested channel sections, by simply identifying a channel section that is more than a predefined percentage dry or mostly dry as entirely dry. Unfortunately, dense groups of trees directly over the channel section can still be identified as wet, as the percentage of the channel section covered by trees will exceed the filtering amount, unless the user sets it very low.

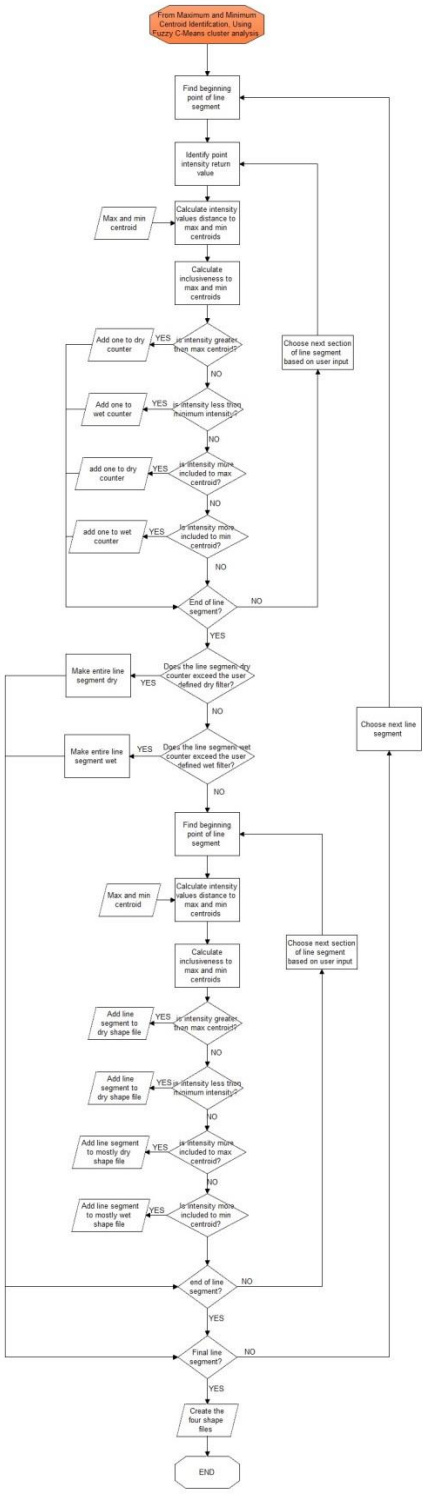

Figure 3: Program Flow Chart, Section 3

Sensitivity Analysis

Alongside analysis of the overall outputs of the tool to determine the effectiveness of the results, an analysis of the various alterable elements of the program was performed to ascertain the effectiveness and variance that each element contributes to the overall output. Analyses of four aspects of the tool were of primary concern:

1. alterations of the minimum allowable cluster analysis data set length,
2. local cluster analysis vs. global cluster analysis,
3. alterations of the user defined bank slope, and
4. river segment filtering.

First, given the variable nature of the available number of data points within the banks of a channel section, identifying a minimum acceptable number of data points, per channel section, becomes necessary. While there is no consensus on an effective minimum number of points for a cluster analysis, a small number of data points have the potential to alter the local cluster analysis and thus, skew the overall maximum and minimum intensity results. For each test location the minimum acceptable number of data points were tested at 5 point increments until the program failed to return results, i.e. no channel sections existed were the number of data points exceeded the minimum acceptable number of data points. Additionally, the slope was set at a constant five percent, for each run, to keep data acquisition consistent between runs.

Second, to help alleviate errors caused by potential outliers in the intensity data, two cluster analyses were performed. One analysis at a local channel section level and another analysis at a global channel network level. A sensitivity analysis was performed to identify if any errors were incurred by using a cluster analysis on the outputs of a previous cluster analysis.

The local cluster analysis was removed from the program and the same tests were performed as previously shown to identify the minimum allowable data set length. The results of both sets of tests were then compared to identify the difference between the two data sets.

Third, multiple channel networks were tested to determine what alterations of the bank slope would have on the outcome of the cluster analysis and the global maximum and minimum intensities obtained. Bank slopes of 0.5, 1, 3, 5, 10, 15 and 20 percent were run in the model. In addition to the four shape files output by the program, the raw intensity values used for the cluster analysis were captured and output. Histograms for each data set were plotted, as well as the cumulative percentage and percent distribution.

Finally, as previously discussed, Aerial LiDAR data are dependent on a clear return to be an effective measurement or identifying device. In the case of wet or dry channel identification, where the intensity of the return is the measurement medium, the accuracy of the process is specifically dependent on the intensity return from a dry or wet point originating from within a channel section. However, thick fauna coverage or inaccurate channel network points can lead to inaccuracies in the output files. For this reason, the inclusion of a filtering option within the program was implemented.

RESULTS

The developed methodology and program flowchart was realized using the Python programming language with the addition of the ESRI ArcGIS programming package, ArcPy. A variety of testing conditions were implemented to identify coding weaknesses. Comparisons of the individual components to hand calculated results, especially for the fuzzy logic elements, were conducted and found to be the same in each case.

Four test sites were analyzed: Clover, South Carolina; Basin 15, Stateline, Nevada; Basin 13, Oregon; Homewood, California. A total of 116 runs were performed for the previously mentioned sensitivity analyses. Because four shape files are produce for each run, this leads to a total number of 464 shape files. Additional testing criteria required additional data outputs, in the form of lists of intensity values which were used by each cluster analysis. Completion time for each run varied based on the size of the site being tested, if the filtering algorithm was in use, the set percentage length for analysis and maximum allowable bank slope. Completion times were as short as 1 hour to as long as 9 hours.

It was found that the nature of the data collected and the analysis performed using the aforementioned methodology complicates discussion of the results if developed separately from identification of the results themselves, and demands frequent cross-references and duplications if so separated. Because of this, the majority of the sensitivity analysis will be discussed in conjunction with identification of the sensitivity results.

DISCUSSION

Minimum Allowable Cluster Analysis Data Set Length

Tables 1 - 4 show the global maximum and minimum intensities for the four test cases. When graphed, the only discernable pattern is a sudden change in values at or around a minimum allowable point number of 25 to 30 (Figures 4 - 7). Tables 5 - 8 confirm that the standard deviation for the first 25 points is significantly lower in almost all cases, when compared to the standard deviation for the entire dataset. While it is clear, based on the figures, that the data are not normally distributed, the standard deviation can still be a useful parametric identifier representative of the central tendency of the data. However, a useful non-parametric indicator of central tendency can also be found via the absolute spread of the data, i.e. the difference between the maximum and minimum values. Both were therefore considered in this work. The reduction in the spread found for the first 25 points confirms the results found using standard deviation. The only standard deviation increase between the first 25 points and the full data set is the maximum intensity standard deviation for Homewood, where the difference is small enough to be overlooked.

Table 1: Clover Maximum and Minimum Intensities

Clover, South Carolina		
Minimum Number	Maximum Intensity	Minimum Intensity
85	N/A	N/A
80	52.743	33.833
75	48.62	25.692
70	48.113	23.403
65	46.962	21.71
60	46.22	21.79
55	45.066	20.007
50	44.85	19.754
45	45.627	22.318
40	45.484	22.112
35	45.443	22.299
30	45.751	22.814
25	45.698	22.763
20	46.531	24.224
15	47.062	25.074
10	47.301	25.877
5	46.186	25.499

Table 2: Basin 15, Maximum and Minimum Intensities

Basin 15, Stateline, NV		
Minimum Number	Maximum Intensity	Minimum Intensity
25	N/A	N/A
20	127.664	56.49
15	126.305	59.346
10	122.022	54.537
5	122.413	57.136

Table 3: Basin 13, Maximum and Minimum Intensities

Basin 13, Oregon		
Minimum Number	Maximum Intensity	Minimum Intensity
75	N/A	N/A
70	61.596	5.464
65	61.596	5.464
60	61.596	5.464
55	61.596	5.464
50	61.596	5.464
45	52.097	15.192
40	51.002	10.747
35	51.002	10.747
30	97.587	34.384
25	117.128	28.226
20	112.623	29.813
15	95.191	28.141
10	103.485	27.959
5	106.291	28.05

Table 4: Homewood, Maximum and Minimum Intensities

Homewood, California		
Minimum Number	Maximum Intensity	Minimum Intensity
45	N/A	N/A
40	122.837	103.786
35	122.837	103.786
30	122.837	103.786
25	114.575	56.22
20	112.929	60.52
15	121.042	45.125
10	121.621	45.18
5	122.314	58.215

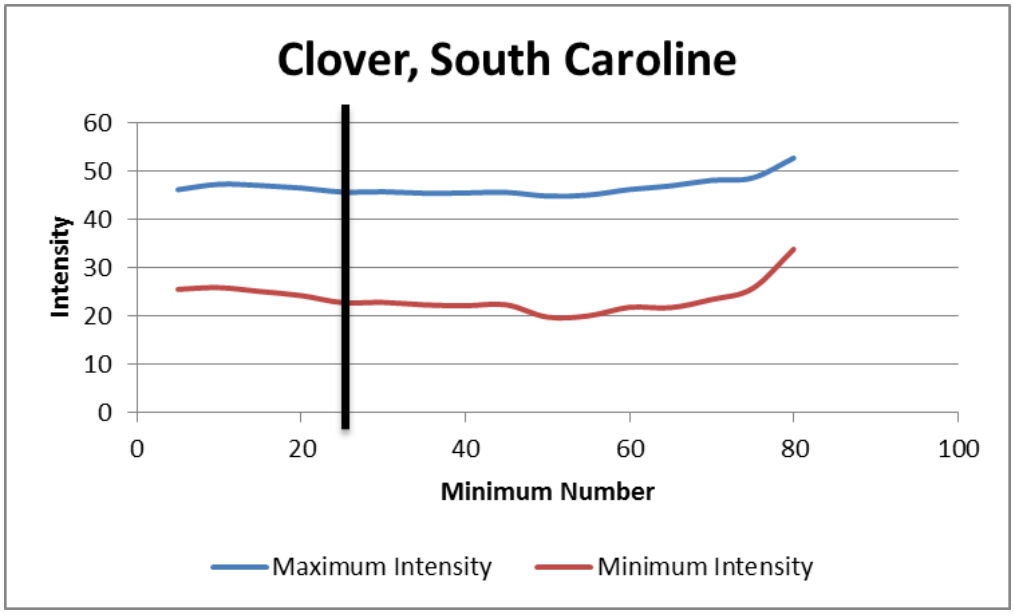

Figure 4: Clover, Intensity vs. Minimum Allowable Number of Points

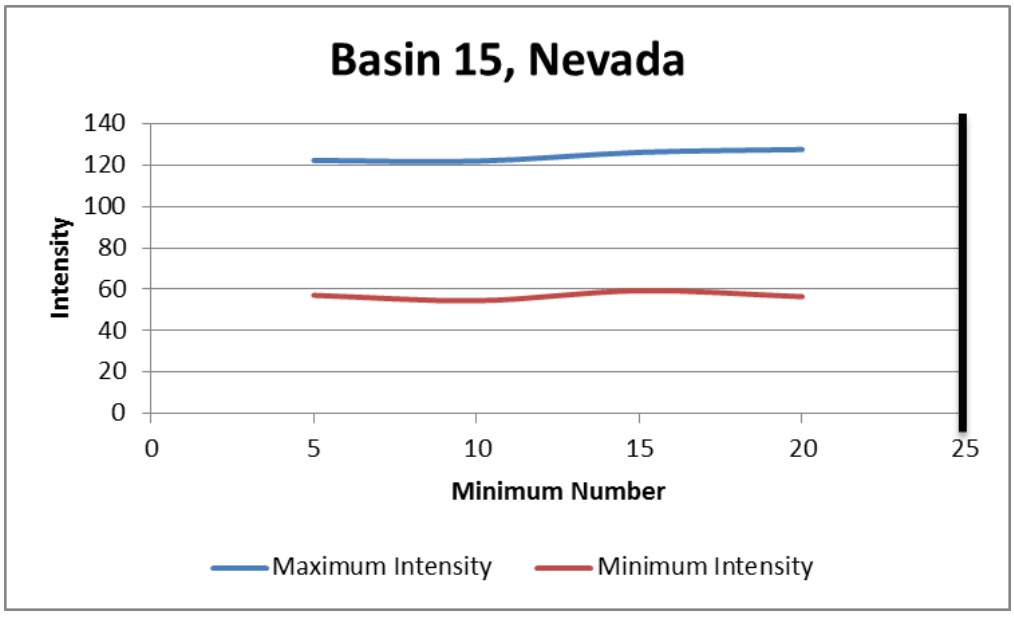

Figure 5: Basin 15, Intensity vs. Minimum Allowable Number of Points

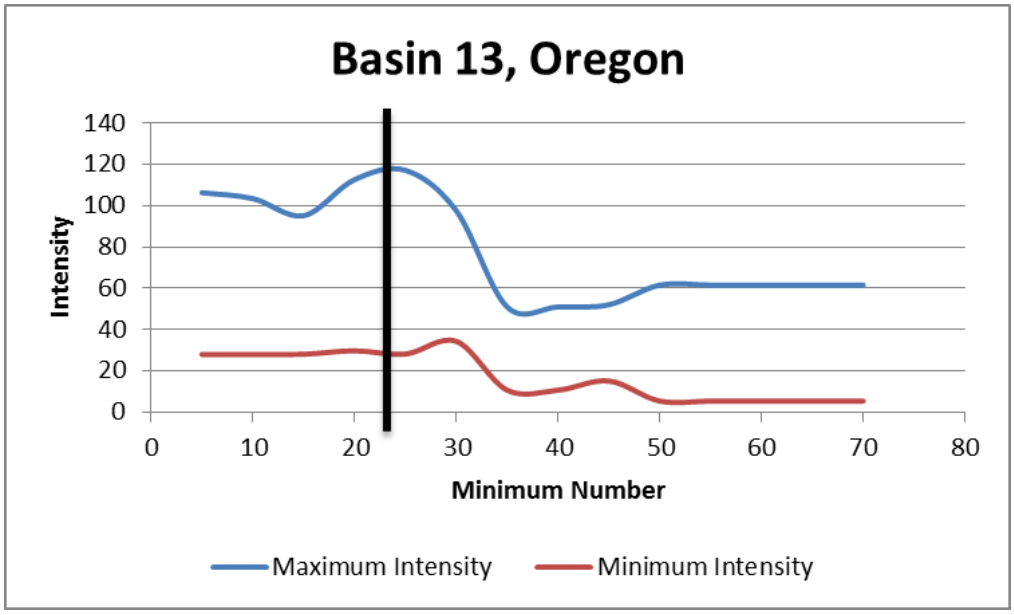

Figure 6: Basin 13, Intensity vs. Minimum Allowable Number of Points

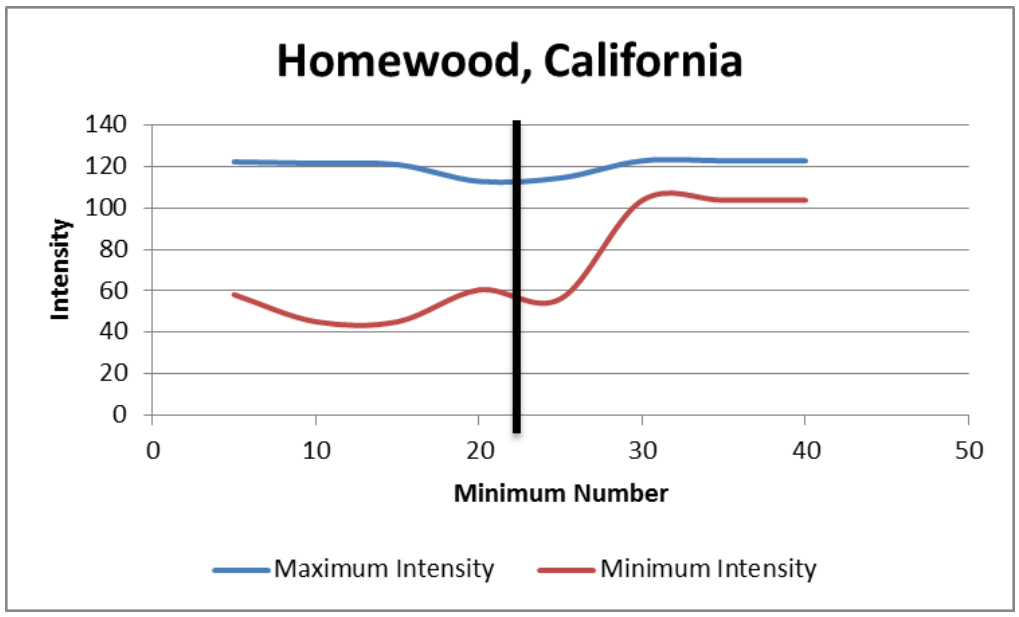

Figure 7: Homewood, Intensity vs. Minimum Allowable Number of Points

Table 5: Clover, Intensity Statistical Analysis

Clover, South Carolina		
Test	Maximum Intensity	Minimum Intensity
All Data Points		
Maximum Value	52.74	33.83
Minimum Value	44.85	19.75
Standard deviation	1.92	3.28
First 25 Data Points		
Maximum Value	47.30	25.88
Minimum Value	45.70	22.76
Standard deviation	0.67	1.35
Second 25 Data Points		
Maximum Value	46.22	22.32
Minimum Value	44.85	19.75
Standard deviation	0.48	1.18

Table 6: Basin 15, Intensity Statistical Analysis

Basin 15, Stateline, Nevada		
Test	Maximum Intensity	Minimum Intensity
All Data Points		
Maximum Value	127.66	59.35
Minimum Value	122.02	54.54
Standard deviation	2.81	1.98

Table 7: Basin 13, Intensity Statistical Analysis

Basin 13, Oregon		
Test	Maximum Intensity	Minimum Intensity
All Data Points		
Maximum Value	117.13	34.38
Minimum Value	51.00	5.46
Standard deviation	27.48	10.20
First 25 Data Points		
Maximum Value	117.13	34.38
Minimum Value	95.19	27.96
Standard deviation	8.48	2.52
Second 25 Data Points		
Maximum Value	61.60	15.19
Minimum Value	51.00	5.46
Standard deviation	17.92	4.05

Table 8: Homewood, Intensity Statistical Analysis

Homewood, California		
Test	Maximum Intensity	Minimum Intensity
All Data Points		
Maximum Value	122.84	103.79
Minimum Value	112.93	45.13
Standard deviation	4.01	26.84
First 25 Data Points		
Maximum Value	122.84	103.79
Minimum Value	112.93	45.13
Standard deviation	4.31	21.74

Based on this analysis, a minimum allowable number of points per river section was set to 10. The data show that a value between 5 and 25 will have a more consistent outcome, when compared to a larger minimum allowable number of points. In addition, Table 9 shows that the lower the minimum allowable number of points, the more overall data there is for the cluster

analysis. Because of this, a number on the lower end of the 5 to 25 point spectrum will generate more points for analysis while still retaining enough points per channel section for local cluster analysis. A minimum number of 10 points was chosen for a calculation limit.

Table 9: Minimum allowable number of points and the total points for analysis

Number of Data Points in Data Set: Minimum allowable points					
Location	Minimum	Number of Data Points	Location	Minimum	Number of Data Points
Basin 13	75	N/A	Clover	85	N/A
	70	70		80	1375
	65	70		75	2679
	60	70		70	3319
	55	70		65	3930
	50	70		60	4362
	45	115		55	4988
	40	159		50	5201
	35	159		45	5483
	30	290		40	5904
	25	534		35	5978
	20	954		30	6518
	15	3226		25	6736
	10	7830		20	7267
	5	20868		15	7924
Homewood	45	N/A		10	8805
	40	43		5	10077
	35	43	Basin 15	25	N/A
	30	43		20	43
	25	71		15	60
	20	160		10	131
	15	327		5	267
	10	694			
	5	2458			

Local Cluster Analysis vs Global Cluster Analysis

The difference as well as the percent error of the maximum and minimum intensities, with and without the local cluster analysis, was calculated (Tables 10 – 13). Results show that, except for rare cases, the difference between the two-cluster method and the one-cluster method is small. If the previously chosen minimum calculation limit of 10 points is taken, the difference is less than 4 in all study cases for the high intensity centroid and less than 5 for the low intensity centroid, in all but the Basin 15 test case.

The overall close nature of most of the differences between the two- and one-cluster analyses, even considering the relatively large difference found in the minimum intensity centroid of Basin 15, at the minimum calculation limit of 10, is such that it is not substantial enough to forgo using the two-cluster analysis method.

Table 10: Clover, Global Only Statistical Analysis

Clover, South Carolina						
Minimum Number	Maximum Intensity	Minimum Intensity	Max % Error	Min % Error	Max Difference	Min Difference
85	N/A	N/A	N/A	N/A	N/A	N/A
80	52.01	35.65	1.40	5.10	0.73	1.82
75	47.50	27.57	2.35	6.82	1.12	1.88
70	46.23	24.65	4.07	5.04	1.88	1.24
65	44.97	21.37	4.43	1.57	1.99	0.34
60	44.40	21.15	4.09	3.05	1.82	0.65
55	44.15	19.57	2.07	2.22	0.91	0.44
50	43.89	19.28	2.18	2.45	0.96	0.47
45	44.11	19.34	3.44	15.37	1.52	2.97
40	44.34	19.20	2.59	15.14	1.15	2.91
35	44.28	19.33	2.64	15.34	1.17	2.97
30	44.52	19.67	2.76	16.01	1.23	3.15
25	44.64	19.84	2.37	14.76	1.06	2.93
20	45.02	20.44	3.36	18.49	1.51	3.78
15	45.29	20.98	3.91	19.50	1.77	4.09
10	45.55	21.58	3.85	19.91	1.75	4.30
5	45.36	21.96	1.82	16.13	0.83	3.54

Table 11: Basin 15, Global Only Statistical Analysis

Basin 15, Stateline, NV						
Minimum Number	Maximum Intensity	Minimum Intensity	Max % Error	Min % Error	Max Difference	Min Difference
25	N/A	N/A	N/A	N/A	N/A	N/A
20	133.09	57.39	4.08	1.56	5.43	0.90
15	131.16	67.02	3.70	11.45	4.85	7.67
10	125.94	69.83	3.11	21.90	3.92	15.30
5	124.60	65.69	1.75	13.02	2.18	8.55

Table 12: Basin 13, Global Only Statistical Analysis

Basin 13, Oregon						
Minimum Number	Maximum Intensity	Minimum Intensity	Max % Error	Min % Error	Max Difference	Min Difference
75	N/A	N/A	N/A	N/A	N/A	N/A
70	61.60	5.46	0.00	0.00	0.00	0.00
65	61.60	5.46	0.00	0.00	0.00	0.00
60	61.60	5.46	0.00	0.00	0.00	0.00
55	61.60	5.46	0.00	0.00	0.00	0.00
50	61.60	5.46	0.00	0.00	0.00	0.00
45	43.00	9.01	21.15	68.69	9.10	6.19
40	42.25	7.00	20.71	53.48	8.75	3.75
35	42.25	7.00	20.71	53.48	8.75	3.75
30	103.33	22.71	5.55	51.40	5.74	11.67
25	109.81	21.87	6.66	29.07	7.32	6.36
20	113.86	24.85	1.08	20.00	1.24	4.97
15	97.87	24.52	2.74	14.78	2.68	3.62
10	105.20	25.56	1.63	9.40	1.71	2.40
5	107.29	26.50	0.93	5.85	1.00	1.55

Table 13: Homewood, Global Only Statistical Analysis

Homewood, California						
Minimum Number	Maximum Intensity	Minimum Intensity	Max % Error	Min % Error	Max Difference	Min Difference
45	N/A	N/A	N/A	N/A	N/A	N/A
40	122.84	103.79	0.00	0.00	0.00	0.00
35	122.84	103.79	0.00	0.00	0.00	0.00
30	122.84	103.79	0.00	0.00	0.00	0.00
25	116.70	57.03	1.82	1.41	2.12	0.81
20	112.19	59.18	0.66	2.27	0.74	1.34
15	122.89	46.73	1.50	3.44	1.85	1.61
10	122.09	43.94	0.39	2.82	0.47	1.24
5	122.40	55.97	0.07	4.02	0.09	2.25

Altering Bank Slope

The global high and low intensity values shown in Tables 14 - 17 identify how the alteration of the maximum allowable bank slope can affect the overall intensity values.

Table 14: Clover, Maximum and Minimum Intensity Based on Bank Slope

Clover, South Carolina		
Bank Slope (%)	Maximum Intensity	Minimum Intensity
20	48.63	33.07
15	49.79	33.26
10	72.43	34.75
5	46.53	24.22
3	45.07	20.77
1	43.10	15.60
0.5	41.32	15.30

Table 15: Basin 15, Maximum and Minimum Intensity Based on Bank Slope

Basin 15, Stateline, NV		
Bank Slope (%)	Maximum Intensity	Minimum Intensity
20	116.14	49.88
15	116.56	53.81
10	123.92	58.17
5	127.66	56.49
3	N/A	N/A
1	N/A	N/A
0.5	N/A	N/A

Table 16: Basin 13, Maximum and Minimum Intensity Based on Bank Slope

Basin 13, Oregon		
Bank Slope (%)	Maximum Intensity	Minimum Intensity
20	103.56	30.24
15	100.65	30.13
10	93.07	28.52
5	112.62	29.81
3	115.18	23.74
1	28.83	15.97
0.5	N/A	N/A

Table 17: Homewood, Maximum and Minimum Intensity Based on Bank Slope

Homewood, California		
Bank Slope (%)	Maximum Intensity	Minimum Intensity
20	123.64	53.94
15	124.61	55.16
10	122.68	57.33
5	112.93	60.52
3	52.16	34.68
1	47.46	31.94
0.5	N/A	N/A

An initial test of the program, using the raw intensity values, was a comparison length of each data set. Limiting the maximum allowable bank slope lowers the pool from which data points can be drawn, meaning that progressively decreasing the maximum allowable slope should, in turn, decrease the length of the corresponding data sets. Table 18 shows that, as expected, for the four basins tested there is a decrease in the number of points within the data sets, as related to a decrease in the maximum allowable bank slope.

Table 18: Number of Points Per Bank Slope, Per Location

Number of Data Points in Data Set		
Location	Bank Slope (%)	Number of Data Points
Clover	20	31661
	15	22234
	10	14681
	5	7267
	3	5080
	1	1476
	0.5	451
Basin 13	20	17759
	15	10687
	10	4859
	5	954
	3	305
	1	20
	0.5	N/A
Basin 15	20	2534
	15	1479
	10	543
	5	43
	3	N/A
	1	N/A
	0.5	N/A
Homewood	20	4235
	15	2411
	10	984
	5	160
	3	22
	1	20
	0.5	N/A

This analysis also identified several cases where a maximum allowable bank slope would not return results. In the case of Basin 13 and Homewood, a bank slope of 0.5 percent did not include enough points per line segment to be used as a test point and would return a divide by

zero error in the next code section. Basin 15 would not return results for 0.5, 1 or 3 percent maximum allowable bank slope. As can be seen in Table 18, for each test case with a non-return, there are also bank slopes with minimal points in the data set. For instance, in addition to a bank slope of 0.5% returning null results, the Homewood test case also shows a low number of data points for 1 and 3 percent maximum allowable bank slopes. This leads to markedly skewed results for the maximum and minimum intensity values, indicated by Table 17, as a low number of channel sections could have been used for the calculations.

Bank slope testing also identified a limitation with the current design of the model. In a case where the channel slope exceeds the maximum allowable bank slope, the program does not function as intended. Basin 15 is located south east of Lake Tahoe where the channel network is on the side of a mountain. The average channel slope for the network is close to 4 percent. Because of this, when the program identifies the slope for comparison to the maximum allowable bank slope, it confuses the channel slope with the bank slope, as it does not currently identify the channel slope. This means that the program will not work for areas where the channel slope exceeds the maximum allowable bank slope, as the number of data points collected will be insufficient for later calculations.

Figure 8 shows the histogram distribution of intensity values for the Clover channel network. The increase of the maximum allowable bank slope has had the desired effect of increasing the number of points within the dataset. The dual mounding of the histogram indicates two clusters of data within the data set which correspond to low intensity returns and high intensity returns. Increases in maximum allowable bank slope result in a corresponding increase in the high intensity return cluster. This is as expected, as increasing the maximum

allowable bank slope beyond what the actual bank slope of the channel network is increases the number of "dry" or high intensity return values by allowing the program to identify points outside the true channel bank as points within the set boundaries of the program.

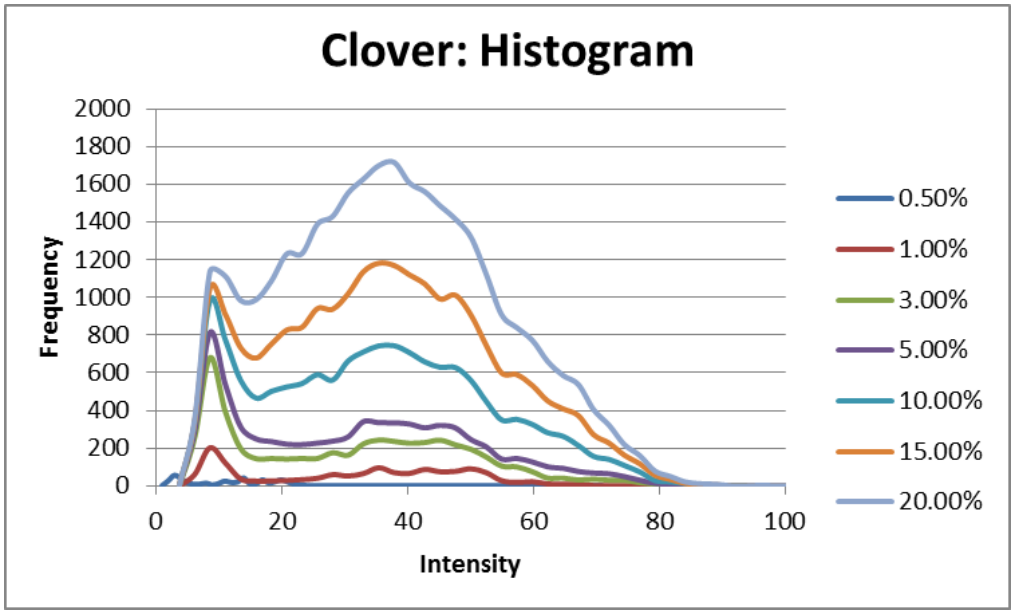

Figure 8: Clover, Histogram of Intensity Data vs. Frequency

Figures 9 - 11 show the histograms of the three remaining test sites. Clover and Homewood both show a double mounding that is expected. However, neither Basin 13 nor Basin 15 shows this trend. In the case of Basin 13, the majority of the intensity returns obtained are relatively low. Upon inspection of the intensity data in ArcGIS, Basin 13 showed a high amount of null filled data for the channel network. Figure 12 shows a section of Basin 13's null filled data. In essence, when the intensity return map was created, the areas with no or "null" data, were filled using a gradient from the last known point to next known point. In the case of Basin 13, the known points were shown to have relatively low intensities. Thus, the histogram

of the intensity return data set for every maximum allowable bank slow shows a large number of low intensity returns. Taking the low degree of accuracy of the intensity return data into account, the variance between the low and high global intensity values is relatively large, as can be seen in Tables 16. The use of fuzzy logic memberships for determination of the global intensity values allows for a degree of flexibility within the clustering of the data. As the clustering algorithm is set on defining two clusters regardless of point density, and as the low intensity cluster is well defined, the high intensity cluster must be identified using the remainder of the data set.

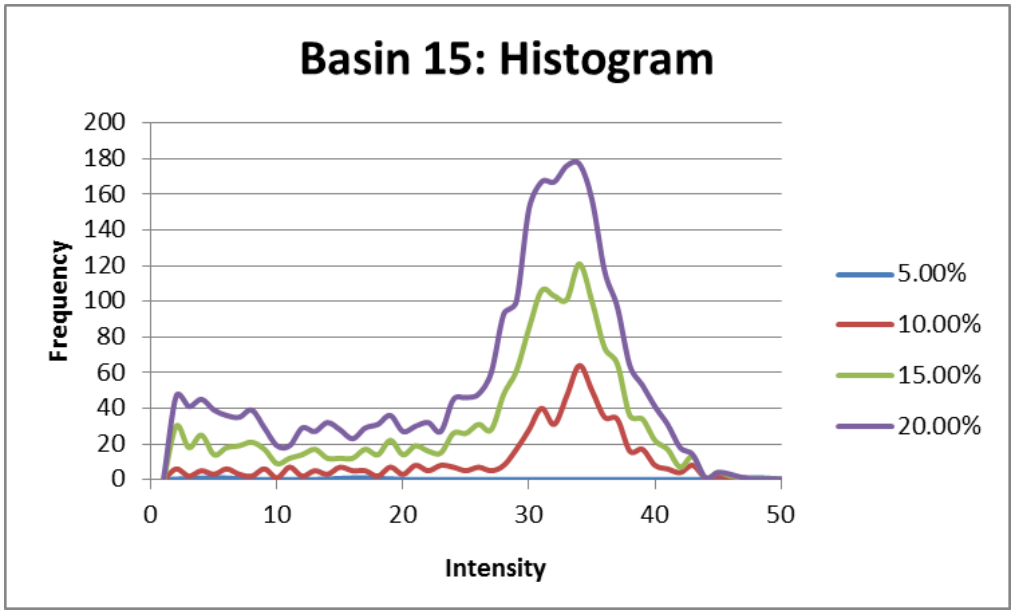

Figure 9: Basin 15, Histogram of Intensity Data vs. Frequency

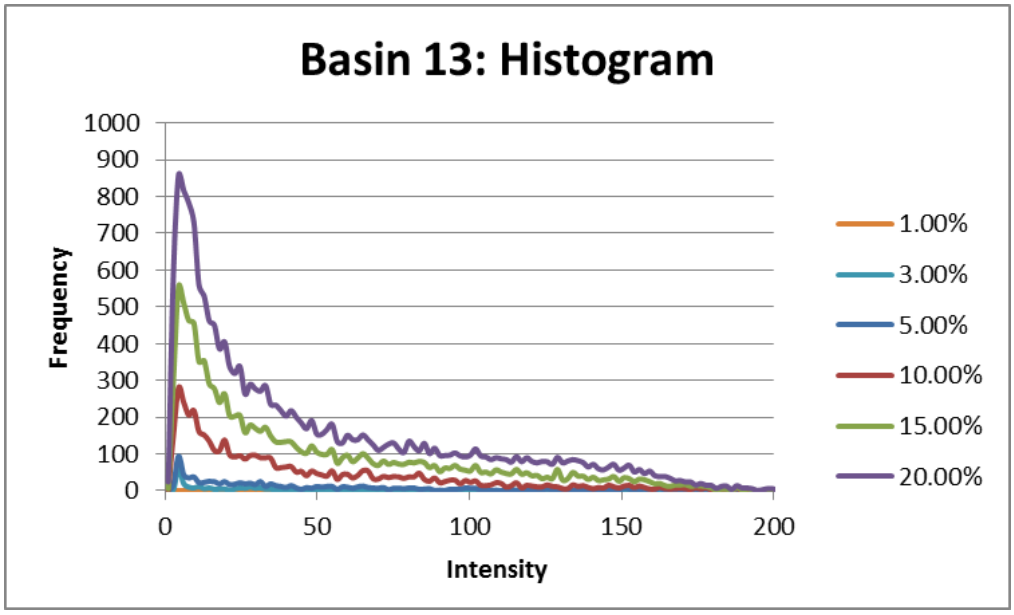

Figure 10: Basin 13, Histogram of Intensity Data vs. Frequency

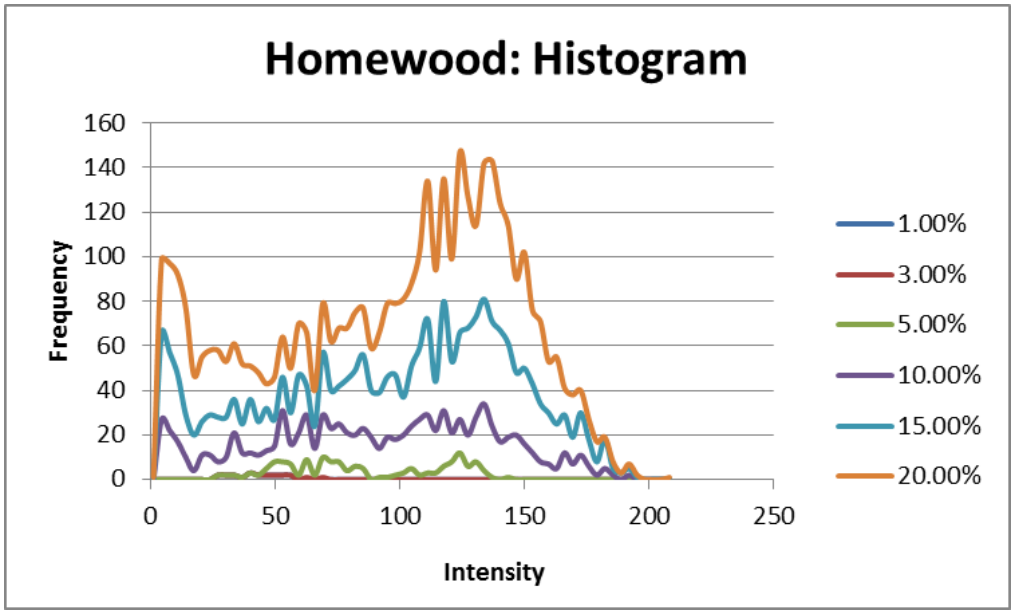

Figure 11: Homewood, Histogram of Intensity Data vs. Frequency

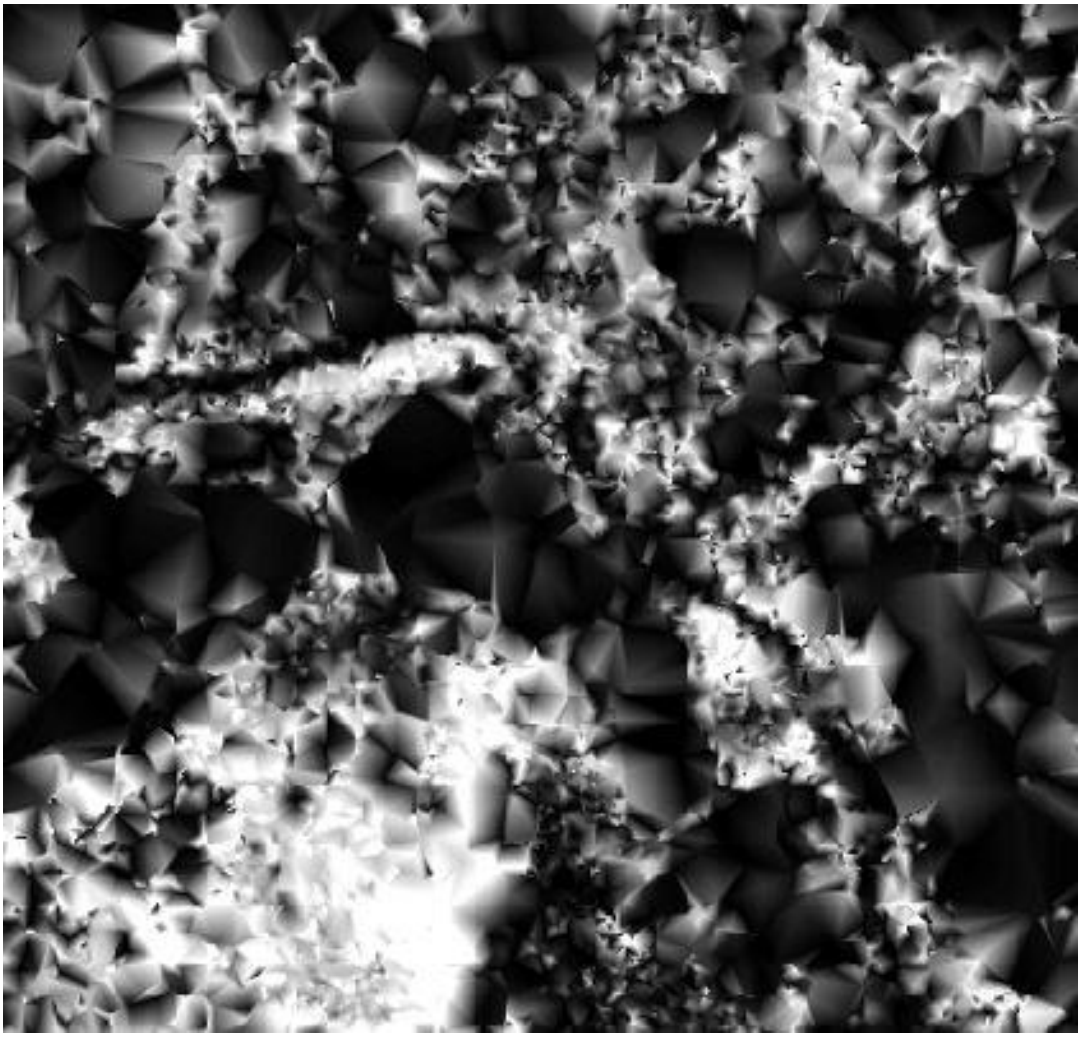

Figure 12: Null Field Fill Areas

While Basin 15 shows the same peaking trend as expected and seen in the Homewood and Clover test cases, truncation of the data because of the high channel slope becomes evident. When a comparison (Figure 13 - 15) of the peak sections of Clover and Homewood was made with the full histogram of Basin 15, the same pattern becomes evident. This shows that, while Basin 15's channel slope does not allow for a fully accurate representation of the wet or dry conditions within the channel network, presence of the expected trend in the higher bank slope

percentages allows for an approximate representation. In addition, given that cluster analysis methods focus on data cluster, and Basin 15 still has representative high and low clustering points, in much the same way as Homewood, calculation of high and low centroids is still a viable technique.

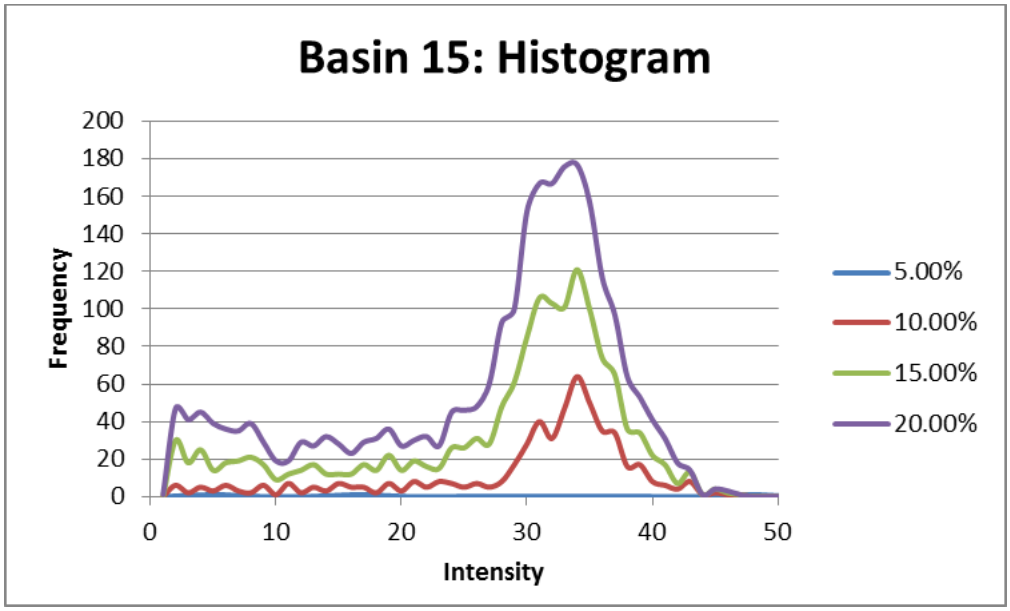

Figure 13: Basin 15, Histogram Peak Comparison

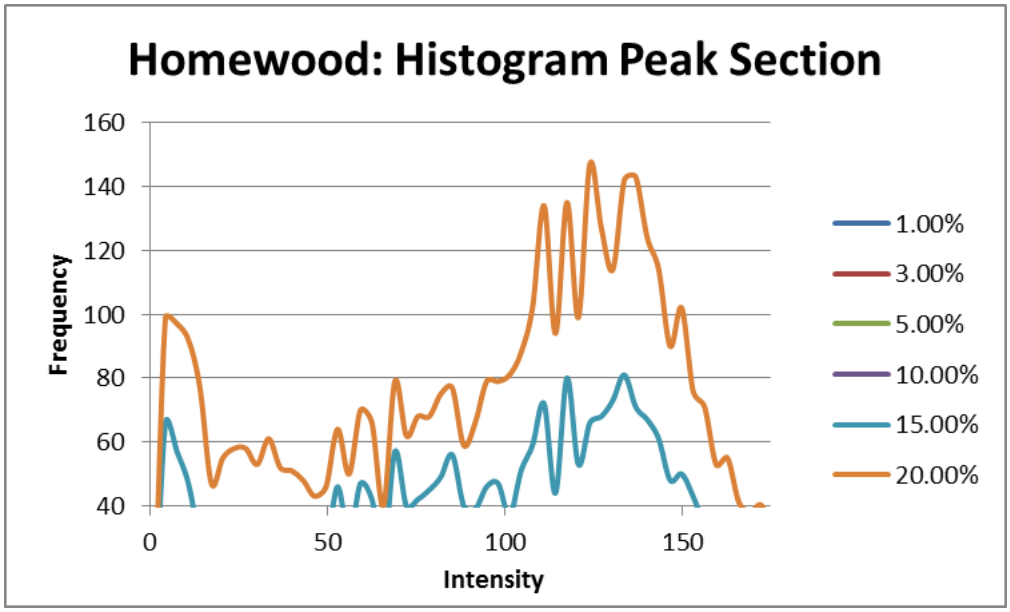

Figure 14: Homewood, Histogram Peak Comparison

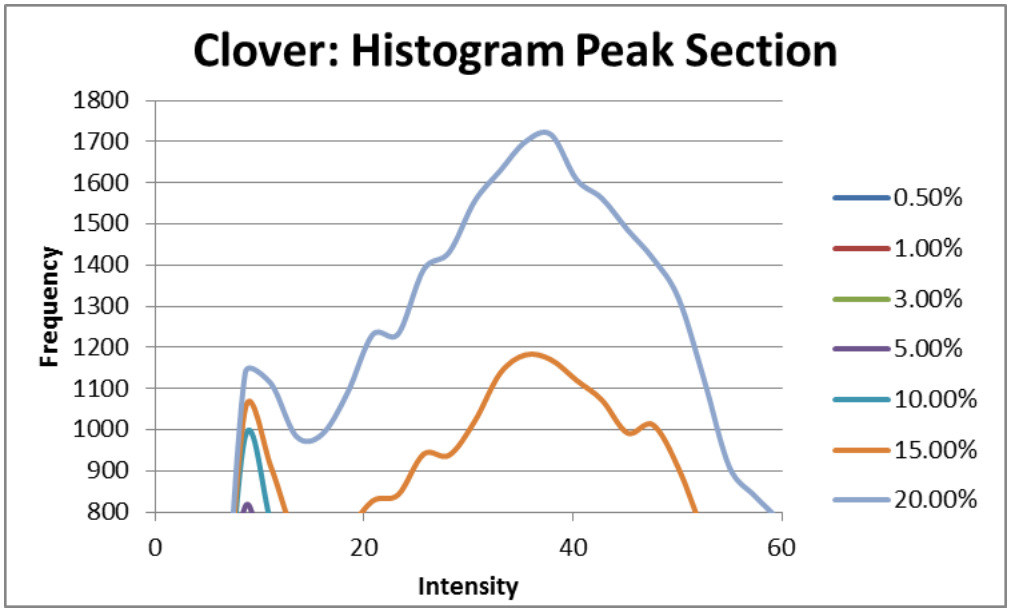

Figure 15: Clover, Histogram Peak Comparison

Analysis of the percent distributions of the test sites revealed another potential issue within the method that baseline histograms did not make readily apparent. The maximum and minimum intensity values calculated in Table 14, for the Clover test site at a 1% and 0.5% slope are similar to each other, yet when a comparison of their percent distributions is made; it becomes evident that the intensity values calculated for a bank slope of 0.5% are false returns (Figure 16). A percent distribution comparison shows that the intensity peaks for a bank slope of 1% are roughly in line with the maximum and minimum intensity values of 43.1 and 15.6 respectively, while the double peaks for a bank slope of 0.5% are both under an intensity value of 20 while clustering analysis identifies the peaks at 41.3 and 15.3. In the 0.5% bank slope case the tailing end of the distribution drew the maximum intensity clustering analysis higher. Figures 17 and 18 show that similar instances to the Clover test case can be seen in both Basin 15 and Homewood.

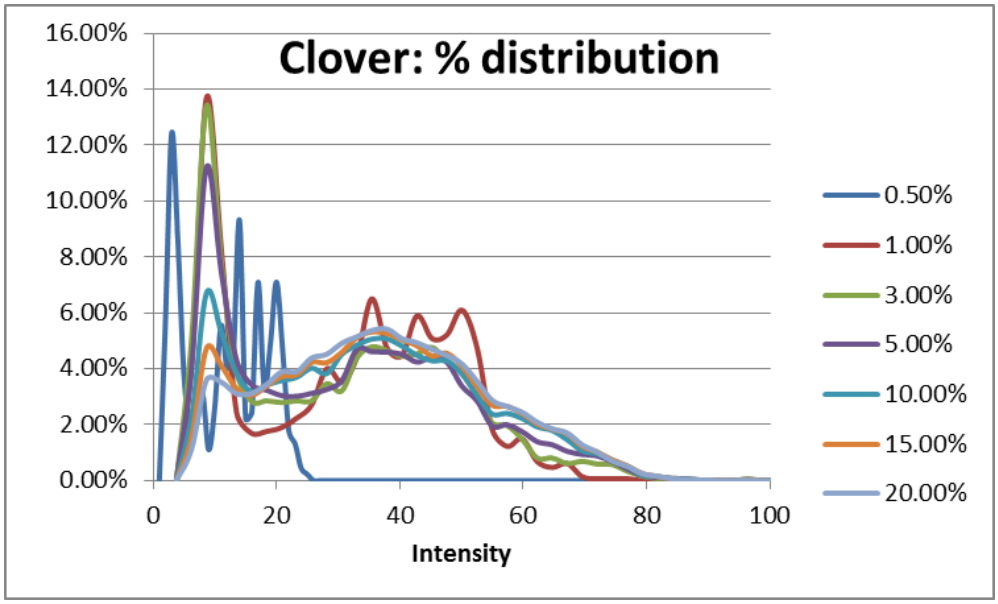

Figure 16: Clover, Percent Distribution of Intensities

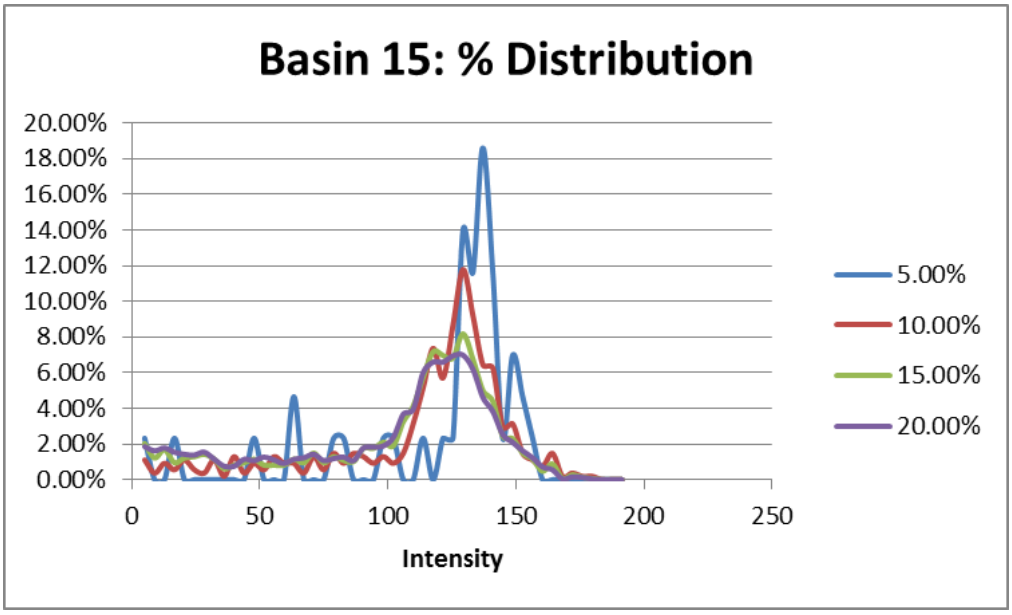

Figure 17: Basin 15, Percent Distribution of Intensities

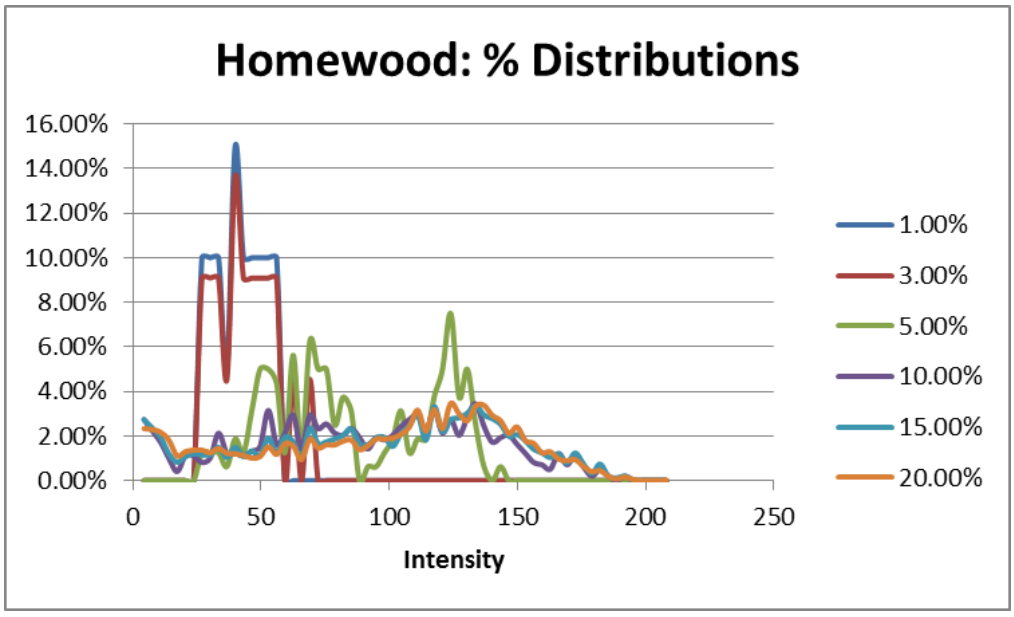

Figure 18: Homewood, Percent Distribution of Intensities

Further analysis using the cumulative percentage shows that, when graphed together, bank slopes within an acceptable range all have similar patterns (Figures 19 - 22). In the case of Clover, Basin 13 and Homewood, outliers from the group can be easily distinguishable from the whole, as they do not have similar slopes to the others.

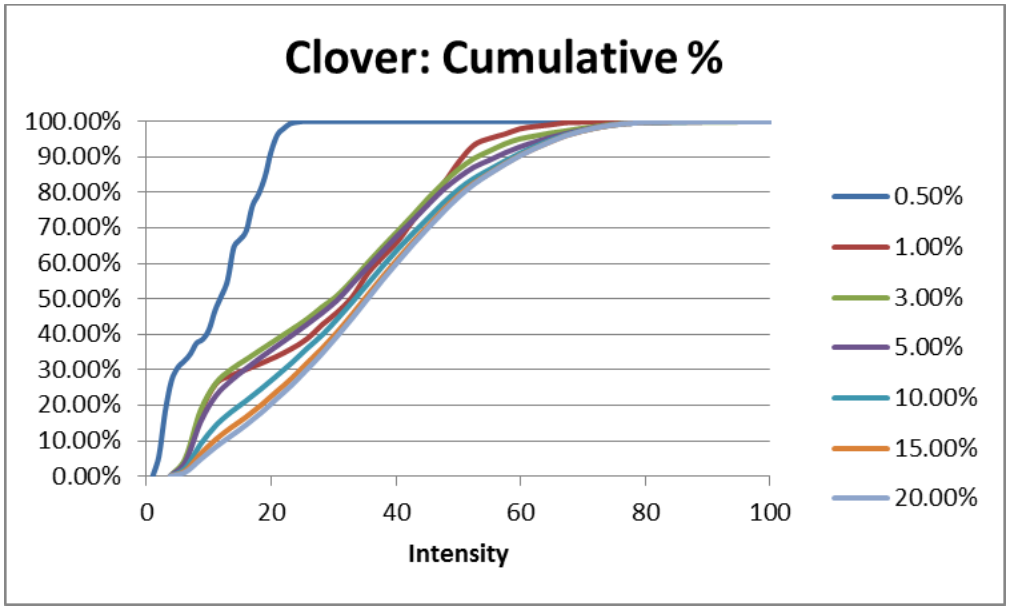

Figure 19: Clover, Cumulative Percentage of Intensities

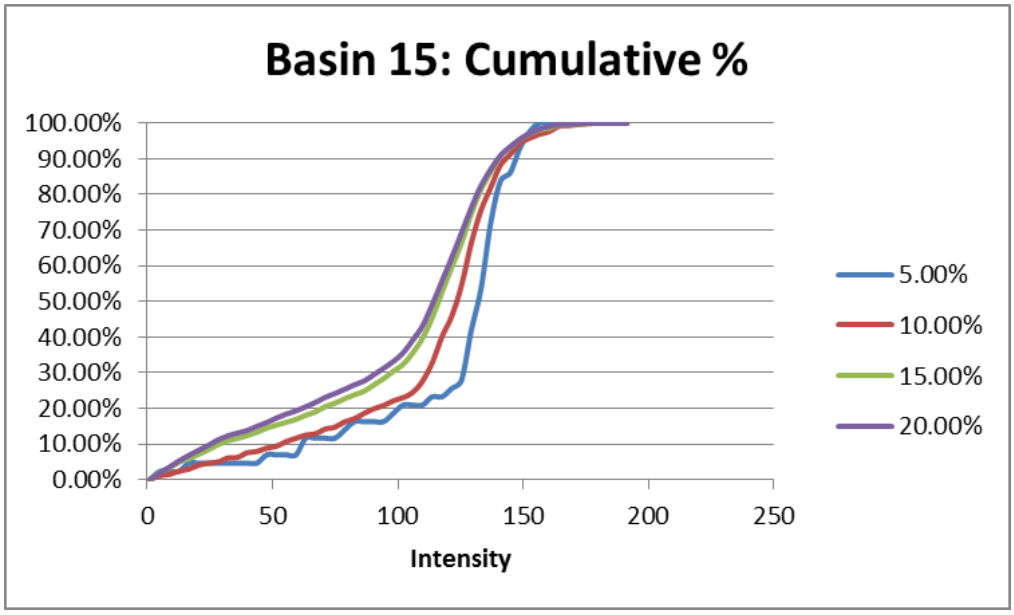

Figure 20: Basin 15, Cumulative Percentage of Intensities

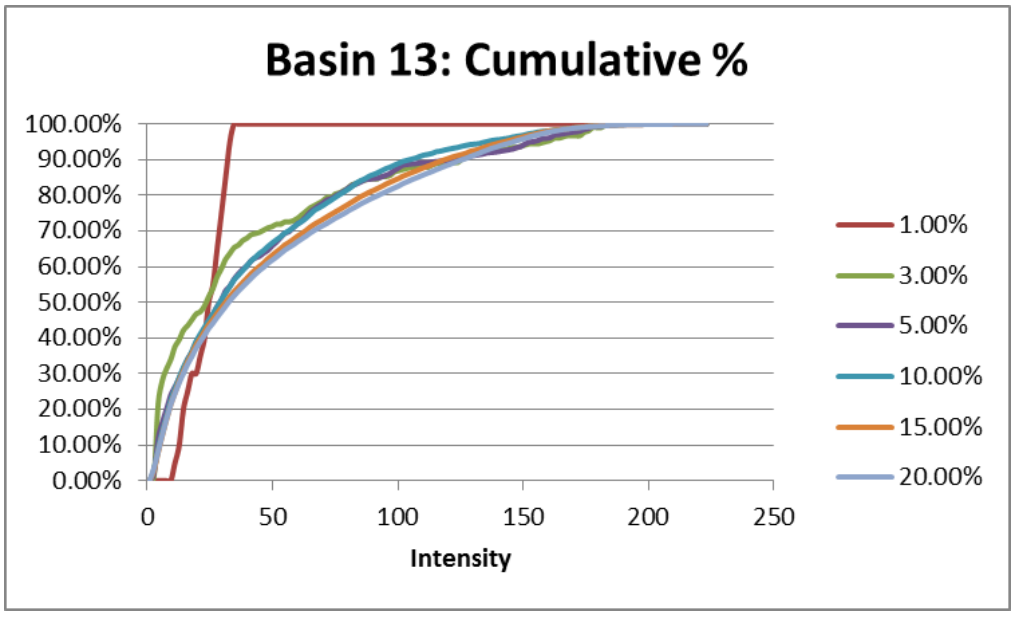

Figure 21: Basin 13, Cumulative Percentage of Intensities

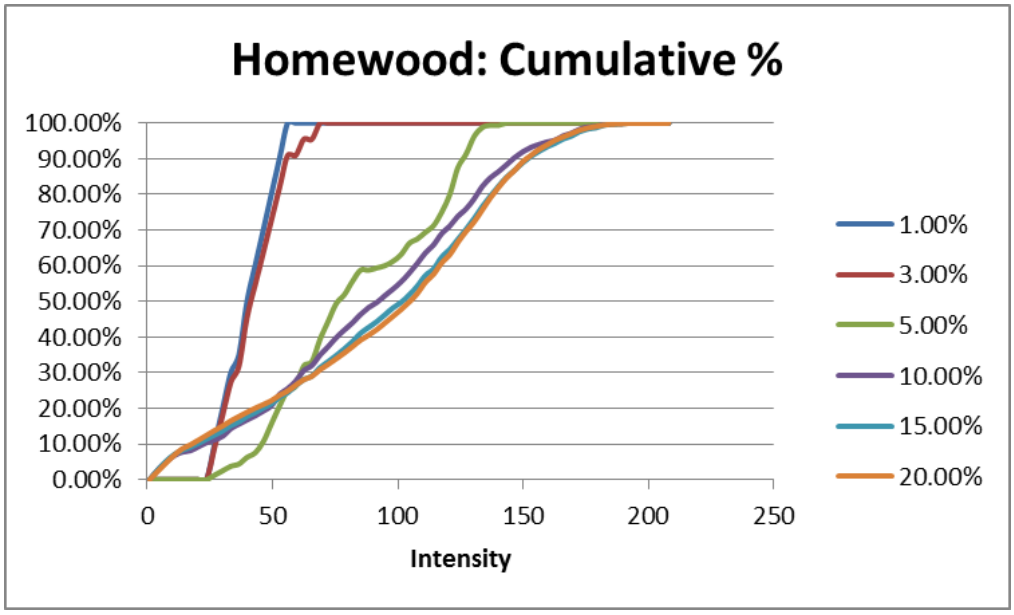

Figure 22: Homewood, Cumulative Percentage of Intensities

River Segment Filtering

The function of the user defined filtering percentage is to attempt to smooth out the aforementioned inconsistencies in the output shape files. For instance, Figure 23 is an incorrectly identified channel segment, where the channel line can clearly be seen to be traveling down a wet channel section. In this case, the small blue line segments represent wet, the green represent mostly wet, the yellow is mostly dry and the red is dry. The testing point was found to be within the bounds of a highly reflective surface, raising the intensity return to within a dry or mostly dry range. The use of the filtering function removed the erroneous section and produced a line segment considered to be entirely wet (Figure 24). It should be noted that, while this function will eliminate incorrect line segments, it will also remove correctly identified segments, if the percentage is set high enough to be considered incorrect.

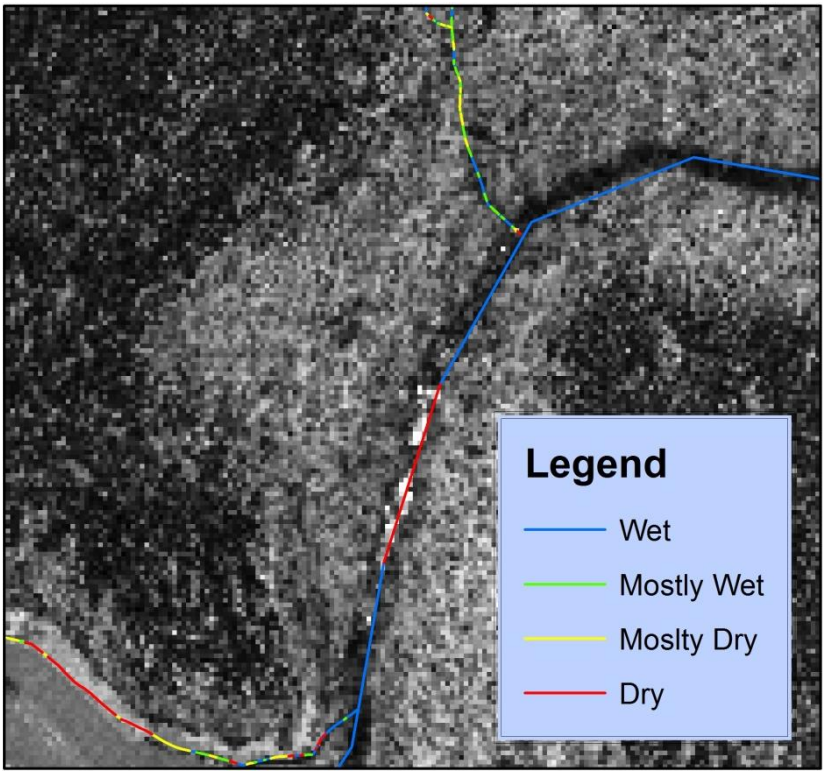

Figure 23: Incorrectly Identified Channel Section

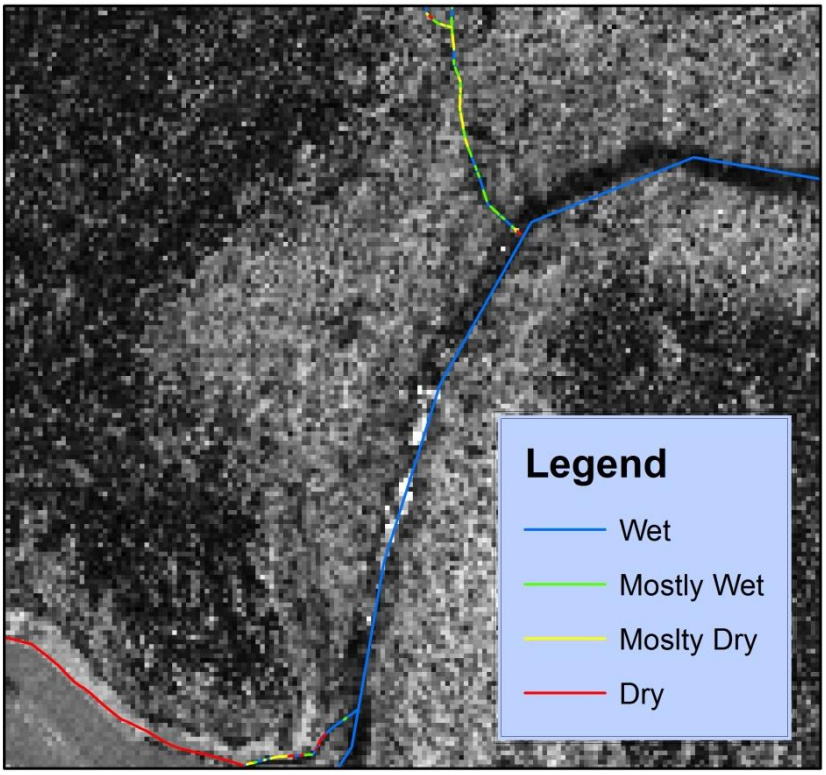

Figure 24: Filtered Channel Section

Overall Results

Using the Clover test case as an example, an average bank slope for the channel network was found to be close to seven percent. In addition, a wet filtering of 10% and a dry filter of 20% were implemented to smooth out inconsistences. Figures 25 - 28 show the individual shape files, using the digital elevation model as a back drop. The wet channel sections are more concentrated along the main trunk of the channel network, while the dry channel sections are more concentrated in the peripheral channels. This is in keeping with what is typical of channel networks, in geomorphological terms. Increases in stream order as a channel network moves toward the central channel, then out of the watershed, corresponds to increases in flow conditions (Strahler, 1969). When the Clover test case is examined, it can be seen that the transition points between each subsequently wetter condition roughly corresponds with increases in effective drainage area for each channel section. In other words, the channel sections become wetter as they come closer to the main trunk of the channel network.

A combination of all four maps produces the overall moisture conditions for Clover (Figure 29) Upon closer inspection, as seen in a close-up of a river segment with the LiDAR intensity map as a back drop (Figure 30), the darker shading, indicating wet channel sections, can be seen to match with the line representing wet channel sections.

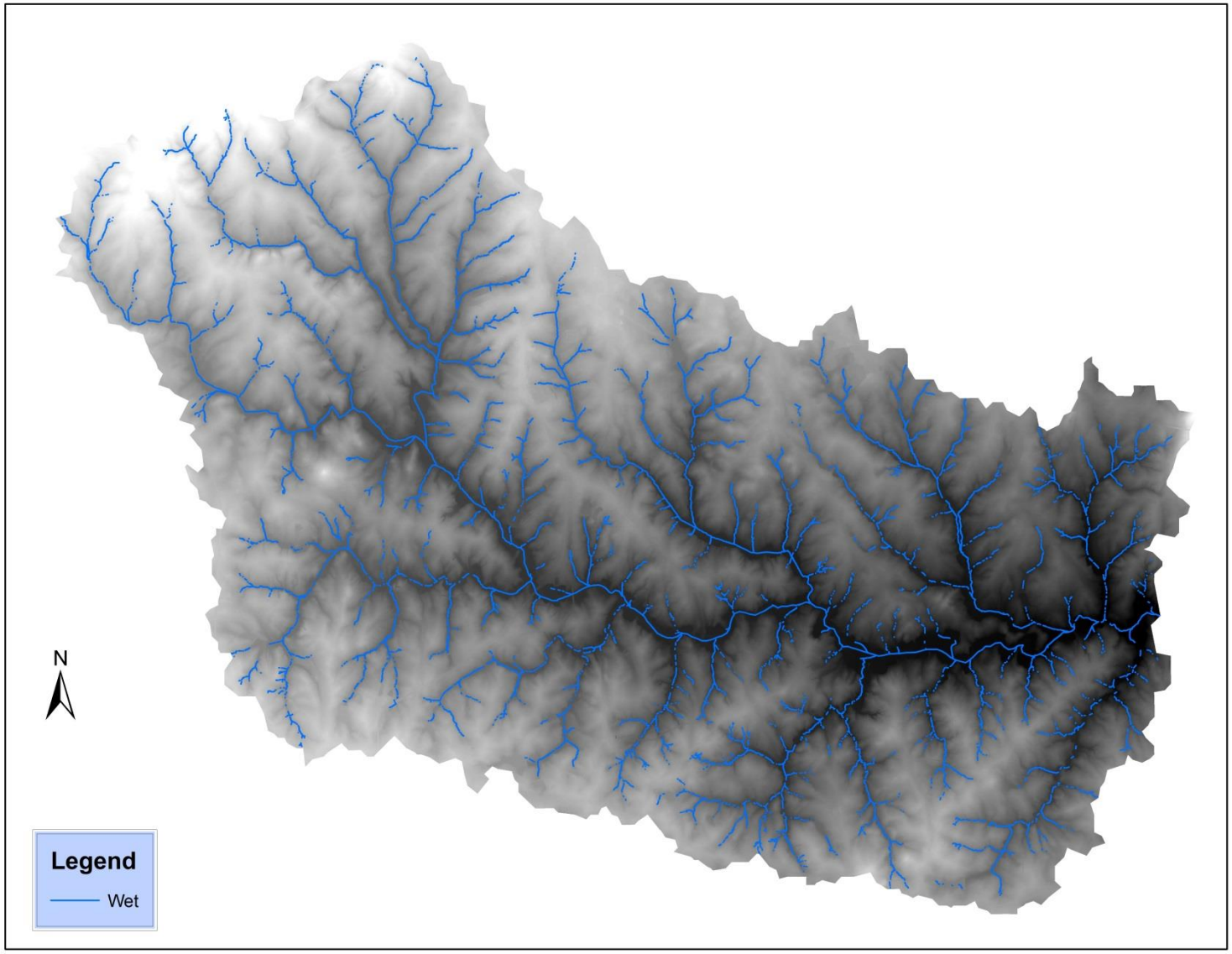

Figure 25: Wet Channel Section

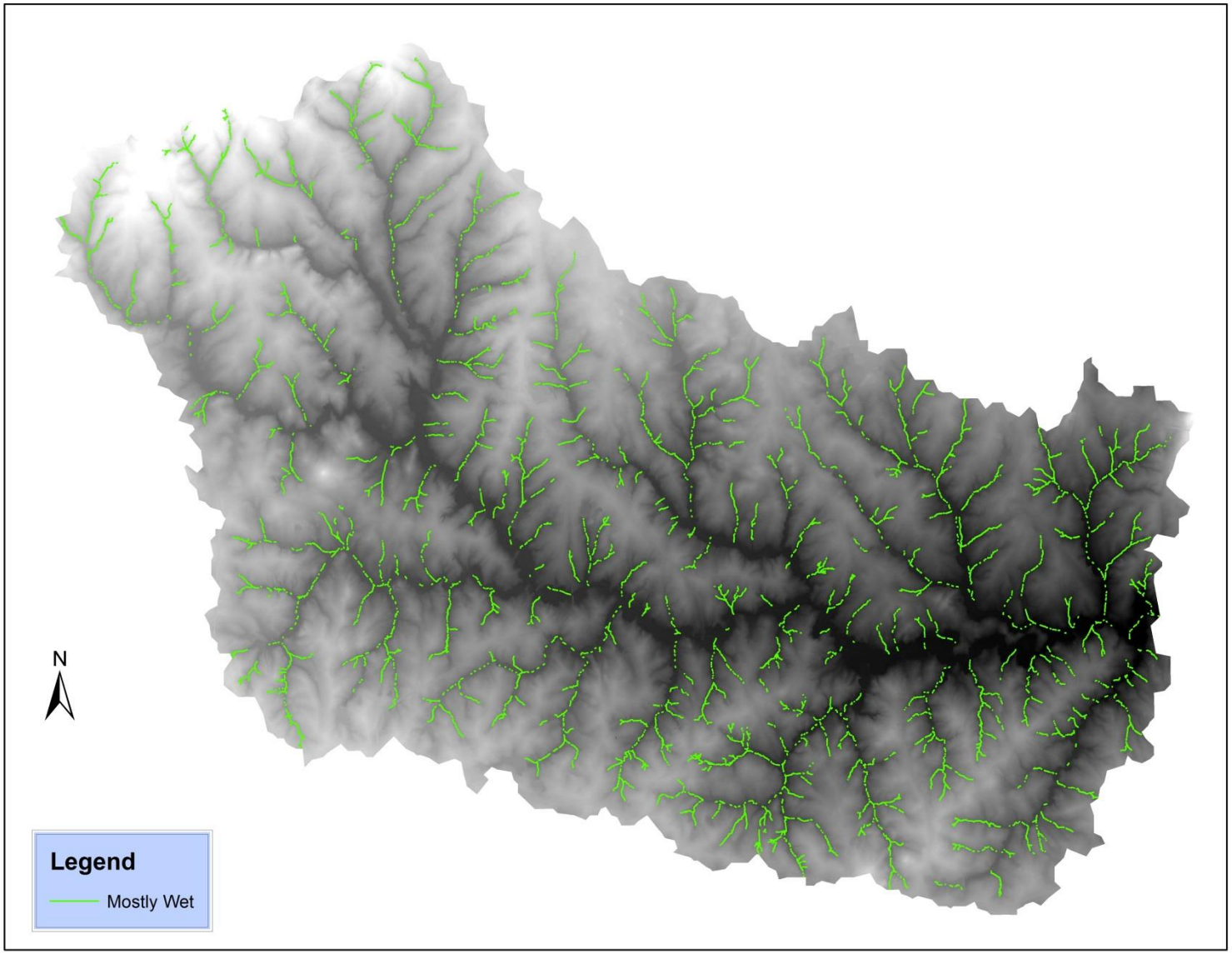

Figure 26: Mostly Wet Channel Section

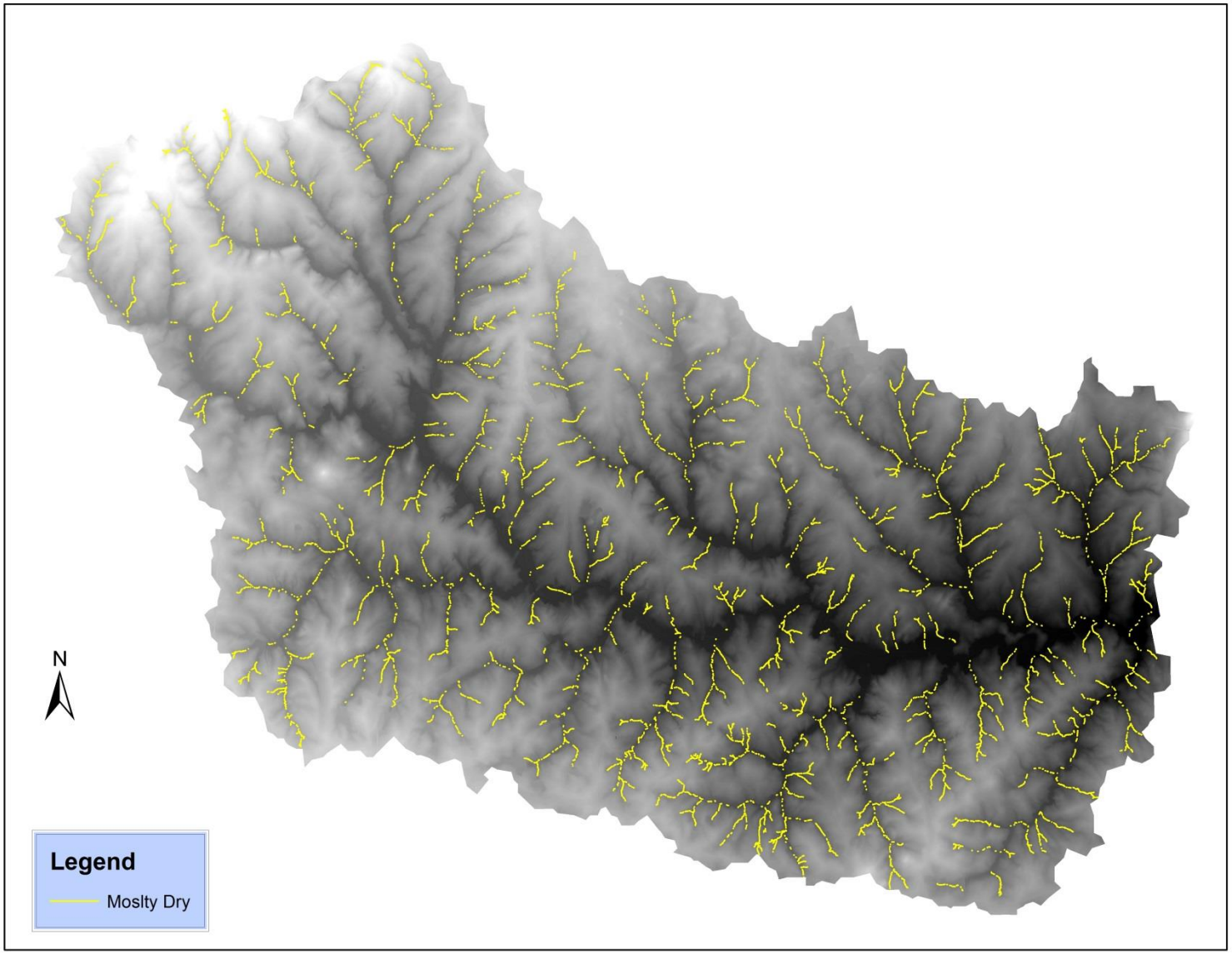

Figure 27: Mostly Dry Channel Section

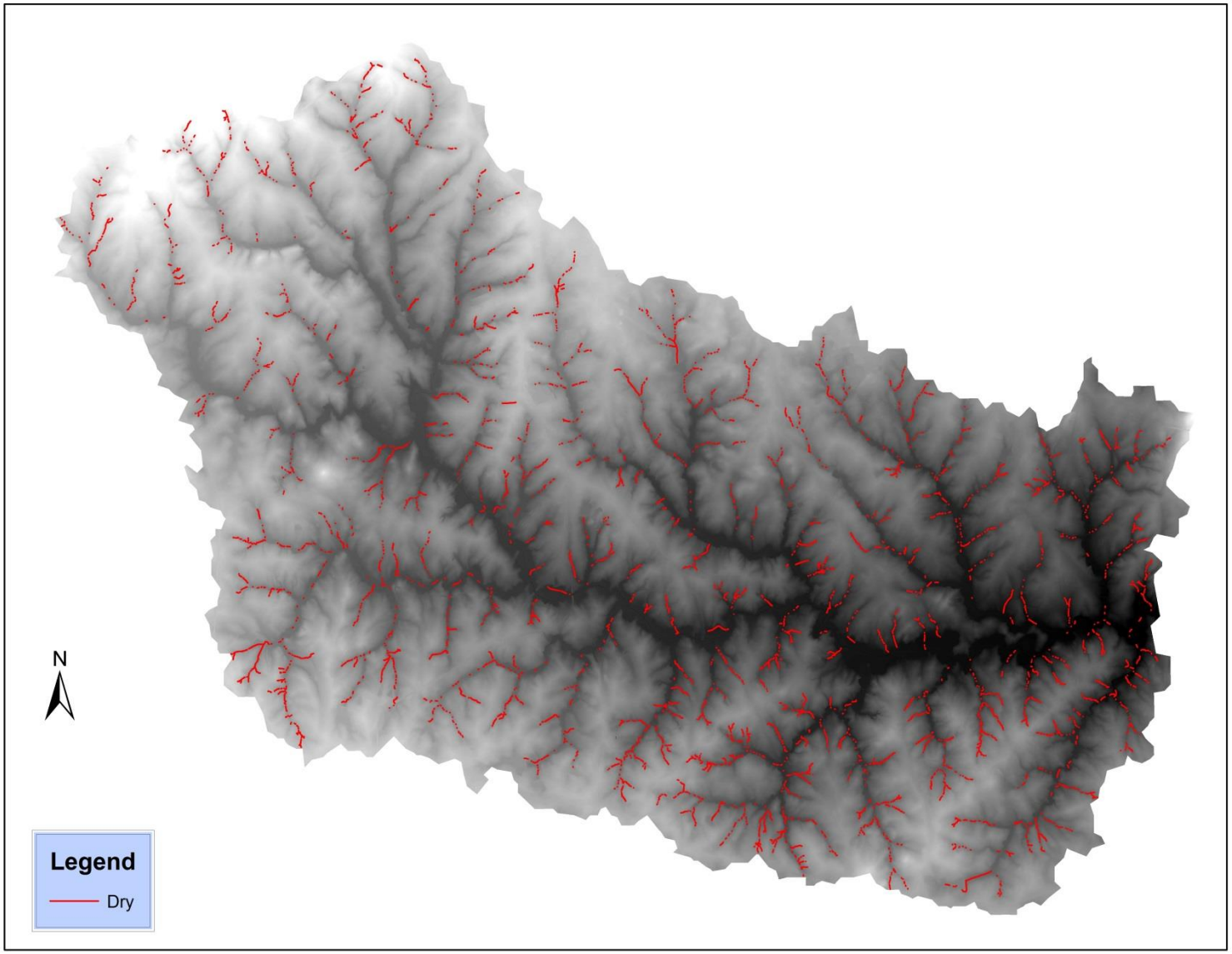

Figure 28: Dry Channel Section

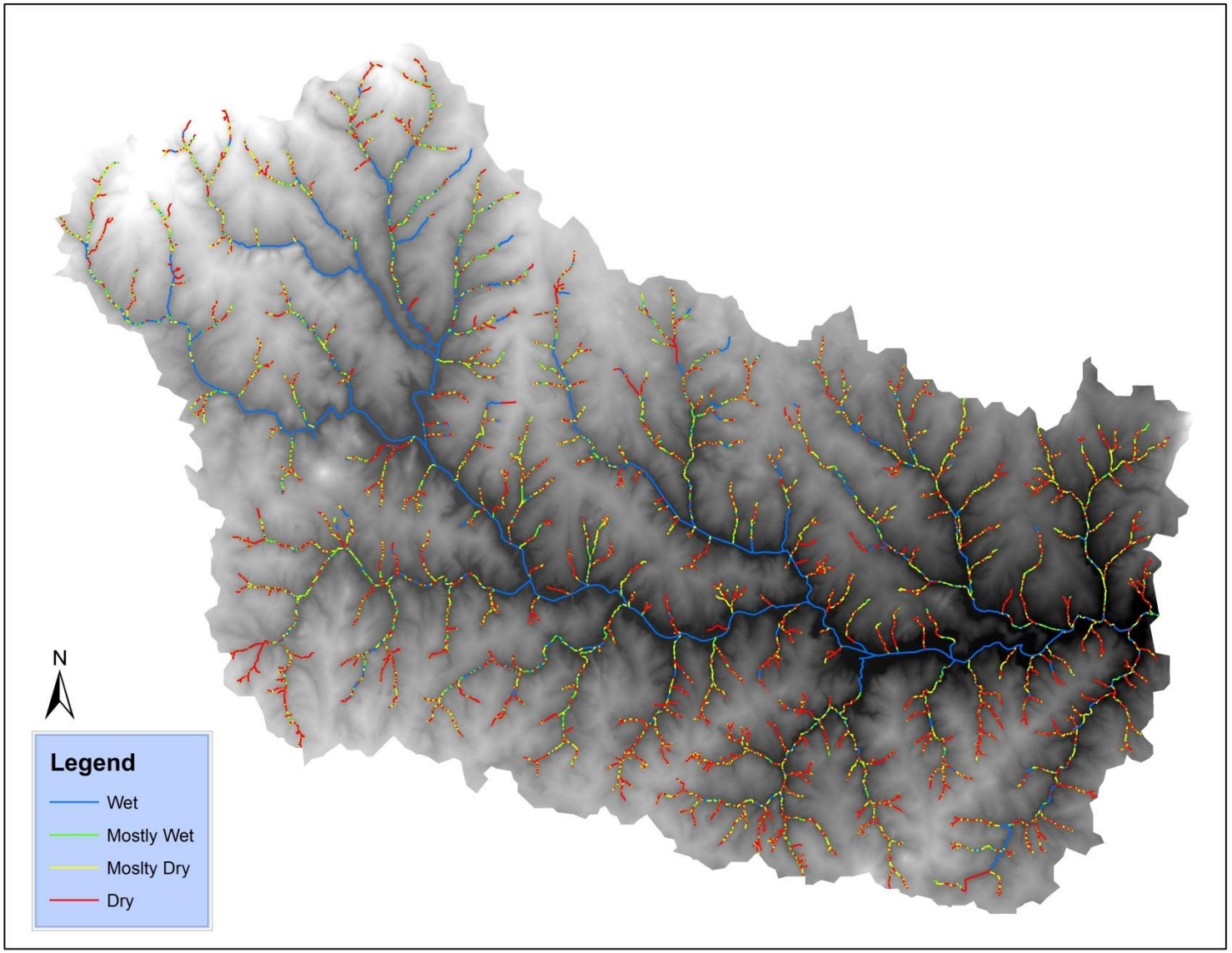

Figure 29: All Channel Section

Figure 30: Close up of Channel Section

FUTURE RESEARCH

The method produced from this research has been found to be effective in identifying wet and dry channel segments. It is suggested that future research consider improvements in several areas which might enable more accurate identification of wet and dry channels.

One area of improvement is in the way trees are accounted for. As noted in the main body of this work, high density trees have a tendency to obscure ground returns, thus lowering the intensity data for the LiDAR and producing low value locations that could be mistaken for wet or mostly wet conditions. This potential influence on results is complicated by the fact that trees themselves in some circumstances behave in ways that do not lead to crisply defined areas; clustering or a ragged demarcation may exist. The addition of an algorithm to identify tree clusters and adjust the intensity values accordingly would be beneficial to this method.

As previously identified, channel slopes that exceed the minimum allowable slope falsely identify the channel slope as the bank slope, for a channel section. The addition of a channel slope identifier would increase the effectiveness and usefulness of the method to include high slope channel conditions. Finally, because the program is dependent on an accurate channel network, as an input, errors in the provided data will induce errors in the end results. The inclusion of a channel location check to confirm that the channel network line is within the boundaries of the channel, based on the digital elevation model, would improve accuracy of the model, as a whole.

The actual definition of ‘wet’ and ‘dry’ is also an area of interest from a biological point of view. For some purposes (e.g. habitat suitability determinations), the degree of wetness that defines a point of demarcation may differ from a hydrologically oriented determination.

Consequently, it is of interest to consider how to incorporate criteria for defining ‘wet’ and ‘dry’ in differing ways to suit differing needs. Coupled with this is the need to develop primary data sets, based on ground proofing that establishes ‘wet’ and ‘dry’ delineations based on observed physical criteria, and that are therefore useful in ground truthing the defined demarcation point.

CONCLUSION

LiDAR intensity data have been used in a multitude of alternate analysis methods, such as urban land use identification, forest biomass quantification and coast line identification. The method proposed in this paper attempts to use a combination of LiDAR elevation and intensity data, coupled with fuzzy logic, to autonomously identify wet and dry channel segments, within a previously identified channel network. Fuzzy logic is used as the LiDAR intensity values have no pre-defined ranges in which wet or dry conditions are dominant. Thus, an independent analysis of the local high and low intensity values is necessary in order to identify the transition points between wet and dry conditions. Given the fuzzy logic framework of the method, four conditions are identified: Dry, Wet, Mostly Dry, Mostly Wet.

A minimum allowable number of points to perform a cluster analysis was set to 10, for local cluster analysis instances. While there is technically no minimum allowable number of points, when performing a cluster analysis, a sensitivity analysis shows that a minimum allowable number of points of 10 was within a range that produced the most consistent results. When the first 25 sets of data were compared to the entire set of data, it was found that the standard deviation for the first 25 sets of data was lower. Given this information, supplemented with data showing that the lower the minimal allowable number of points, the higher the total number of points used, led the aforementioned minimum number of 10 points per cluster analysis.

The method, currently, produces results that identify the four conditions with a reasonable degree of accuracy, based on remote evaluations of the target areas. When a comparison of the methods is made, the difference between a two-cluster method, where a

cluster analysis is performed at the local and global levels shows a small enough difference to the one-cluster method, where a cluster analysis is performed at only the global level, that the use of the two-cluster method is used. The local cluster analysis is meant to reduce the effect outlier intensity data has on the global outcome. Of course, a more robust methodology could be employed, at a later date, to eliminate outliers from the data entirely, but the current method is effective and will be used.

Alteration of bank slope has been shown to minimally affect the outcome of the maximum and minimum global intensity values, except in extreme slope ranges, where minimal data can be used for calculations. Given this information, staying within a reasonable range of actual average bank slope of the area being tested will give the most reasonable results. If the channel slope exceeds that of the bank slope, the method cannot distinguish between the slope of the channel and the bank slope. Because of this, even though the program can produce results if a minimum bank slope is set greater than the channel slope, caution must be used when employing this method in highly sloped locations, such as mountain sides.

Overall, the results produced, using fuzzy logic cluster analysis and LiDAR intensity data, have been satisfactory. Barring field tests to confirm results, the maps produced visually make sense, where it is typical for the main trunk of the channel network to be wet or mostly wet and the peripheral channels are dry or mostly dry. The final results show that this model shows potential for effective use in the identification of channel conditions in a variety of terrain and has the potential to produce good final products when implemented.

APPENDIX: PYTHON CODE

```
import arcpy
import os
from timeit import default_timer
import sys

start = default_timer()
# input file locations
elevationrasterLocation = "C:/GISData/test_deck/KIM/01_09_basin/04_DEM/3_3ft.tif"
intensityrasterLocation =
"C:/GISData/test_deck/KIM/01_09_basin/03_intensity/02_intensity_6ft.flt"
shapeLocation =
"C:/GISData/test_deck/KIM/01_09_basin/02_draw_wet_channel/stream_network.shp"
outpath="Clover_Filtering_dry20_wet10_Bankslope_7_Mincount_10.shp"

# User Defined Inputs
bankslopepercent= 7
SegpercentLength =1
percentoverride = 20
wetfilterpercent = 10
minintensitycount= 10

#conversion to decimal for future calculations
bankslope = float(bankslopepercent)/100
segdecimalLength = float(SegpercentLength)/100
decimaloveride = (100-float(percentoverride))/100
wetfilterdecimal = (100-float(wetfilterpercent))/100

#output location
drive, path = os.path.splitdrive(shapeLocation)
path, outputname = os.path.split(path)

#test for duplicate files
if arcpy.Exists(path + "/wet_"+outpath):
    sys.exit("Output file name already exists, please choose another name")
if arcpy.Exists(path + "/dry_"+outpath):
    sys.exit("Output file name already exists, please choose another name")
if arcpy.Exists(path + "/mostly_wet_"+outpath):
    sys.exit("Output file name already exists, please choose another name")
if arcpy.Exists(path + "/mostly_dry_"+outpath):
    sys.exit("Output file name already exists, please choose another name")

#location determination
linePosition = arcpy.SearchCursor(shapeLocation)

#check intensity resolution and cell size
cellsizeArcOutput =
arcpy.GetRasterProperties_management(intensityrasterLocation,"CELLSIZEX")
cellsize = cellsizeArcOutput.getOutput(0)
spatial_ref = arcpy.Describe(intensityrasterLocation).SpatialReference.linearUnitName
totalintensity = []
highIntensity = []
allintensity=[]
```

```
linecounter= 0

#cycle through shape file and use DEM to identify channel boundaries
#record intensity points that are within channel boundaries
for feature in linePosition:
    linecounter= linecounter+1
    #print linecounter
    Midpoint = feature.shape.positionAlongLine(0.50,True).firstPoint
    #set initial points
    initialX = Midpoint.X
    initialY = Midpoint.Y

    initialCoords =str(initialX) + " " + str(initialY)
    initialElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,initialCoords,"1")
    initialElevation = initialElevationArcOutput.getOutput(0)
    initialIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation,initialCoords, "1")
    initialIntensity=initialIntensityArcOutput.getOutput(0)
    if not initialIntensity == "NoData":
        firstElevation = initialElevation
        listofintensity = [float(initialIntensity)]

    # 1
        for x in range(1,11):
            newX = initialX + x*float(cellsize)
            newY = initialY
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
                #check the slope
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
                    else:
                        break
            else:
```

```
                break
            firstElevation = secondElevation

        # reset first elevation location
        firstElevation = initialElevation

    # 2
        for x in range(1,11):
            newX = initialX + x*float(cellsize)
            newY = initialY - x*float(cellsize)
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
                #check the slope
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
                    else:
                        break
            else:
                break
            firstElevation = secondElevation

        # reset first elevation location
        firstElevation = initialElevation

    # 3
        for x in range(1,11):
            newX = initialX + x*float(cellsize)
            newY = initialY + x*float(cellsize)
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
                #check the slope
```

```
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
                    else:
                        break
            else:
                break
            firstElevation = secondElevation

        # reset first elevation location
        firstElevation = initialElevation

    # 4
        for x in range(1,11):
            newX = initialX - x*float(cellsize)
            newY = initialY
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
                #check the slope
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
                    else:
```

```
                    break
            else:
                break
            firstElevation = secondElevation

        # reset first elevation location
        firstElevation = initialElevation

    # 5
        for x in range(1,11):
            newX = initialX - x*float(cellsize)
            newY = initialY - x*float(cellsize)
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
                #check the slope
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
                    else:
                        break
            else:
                break
            firstElevation = secondElevation

        # reset first elevation location
        firstElevation = initialElevation

    # 6
        for x in range(1,11):
            newX = initialX - x*float(cellsize)
            newY = initialY + x*float(cellsize)
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
```

```
                #check the slope
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
                    else:
                        break
            else:
                break
            firstElevation = secondElevation

        # reset first elevation location
        firstElevation = initialElevation

    # 7
        for x in range(1,11):
            newX = initialX
            newY = initialY - x*float(cellsize)
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
                #check the slope
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
```

```
                    else:
                        break
            else:
                break
            firstElevation = secondElevation

        # reset first elevation location
        firstElevation = initialElevation

    # 8
        for x in range(1,11):
            newX = initialX
            newY = initialY + x*float(cellsize)
            newCoords = str(newX) + " " + str(newY)
            secondElevationArcOutput =
arcpy.GetCellValue_management(elevationrasterLocation,newCoords,"1")
            secondElevation=secondElevationArcOutput.getOutput(0)
            if not secondElevation == "NoData":
                #check the slope
                if abs((float(secondElevation) -
float(firstElevation))/float(cellsize)) > bankslope:
                    highIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newHighIntensity = highIntensityArcOutput.getOutput(0)
                    if not newHighIntensity == "NoData":
                        highIntensity.append(float(newHighIntensity))
                    break
                else:
                    newIntensityArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation, str(newX) + " " +
str(newY),"1")
                    newIntensity=newIntensityArcOutput.getOutput(0)
                    if not newIntensity == "NoData":
                        listofintensity.append(float(newIntensity))
                    else:
                        break
            else:
                break
            firstElevation = secondElevation

        # Fuzzy c-means cluster analysis
        if len(listofintensity) >=minintensitycount:
            #add all intenstiy lists together
            allintensity += listofintensity

            centroidofIntensity = sum(listofintensity) / float(len(listofintensity))
            minofIntensity = min(listofintensity)
            maxofIntensity = max(listofintensity)
            fuzziness = 2

            #initialize max and min centroids and other lists and variables
```

```
            MaxCentroid1 = maxofIntensity*(2)
            MaxCentroid2 = maxofIntensity
            MinCentroid1 = minofIntensity*(2)
            MinCentroid2 = minofIntensity
            listofMaxIntensity = []
            listofMinIntensity = []

            sumuCMaxPM = 0
            sumuCMinPM = 0
            sumuCMaxPMX = 0
            sumuCMinPMX = 0

            #continue until difference is only 1%
            while abs((MaxCentroid1-MaxCentroid2)/MaxCentroid2) and
abs((MinCentroid1-MinCentroid2)/MinCentroid2) >= .01:
                MaxCentroid1 = MaxCentroid2
                MinCentroid1 = MinCentroid2
                for intensity in listofintensity:
                    #check which centroid intensity is closer to
                    distancetoMaxCentroid = abs(MaxCentroid1-intensity)
                    distancetoMinCentroid = abs(MinCentroid1-intensity)

                    if distancetoMaxCentroid <= 0 or distancetoMinCentroid <= 0:
                        if distancetoMaxCentroid <= 0:
                            uCMax=1
                            uCMin=0
                        if distancetoMinCentroid <= 0:
                            uCMax=0
                            uCMin=1
                    else:
                        uCMax = (1/distancetoMaxCentroid**(1/(fuzziness-
1))/(1/distancetoMaxCentroid**(1/(fuzziness-
1))+1/distancetoMinCentroid**(1/(fuzziness-1))))
                        uCMin = (1/distancetoMinCentroid**(1/(fuzziness-
1))/(1/distancetoMaxCentroid**(1/(fuzziness-
1))+1/distancetoMinCentroid**(1/(fuzziness-1))))

                    uCMaxPM = uCMax**fuzziness
                    uCMinPM = uCMin**fuzziness

                    uCMaxPMX = uCMaxPM*intensity
                    uCMinPMX = uCMinPM*intensity
                    sumuCMaxPM += uCMaxPM
                    sumuCMinPM += uCMinPM
                    sumuCMaxPMX += uCMaxPMX
                    sumuCMinPMX += uCMinPMX

                MaxCentroid2 = sumuCMaxPMX/sumuCMaxPM
                MinCentroid2 = sumuCMinPMX/sumuCMinPM

            #add values to intensity list if there is at least 30 values per list.
            totalintensity.append(MaxCentroid2)
```

```
            totalintensity.append(MinCentroid2)

    '''**********TEST*****************************
    totalintensitylength = len(totalintensity)
    if totalintensitylength >= 1000:
        break
    **********TEST*****************************'''

#determine transition points using fuzzy C
centroidofTotalIntensity = sum(allintensity) / float(len(allintensity))
minTotalofIntensity = min(allintensity)
maxTotalofIntensity = max(allintensity)
fuzziness = 2

#initialize max and min centroids and other lists and variables
MaxTotalCentroid1 = maxTotalofIntensity*(2)
MaxTotalCentroid2 = maxTotalofIntensity
MinTotalCentroid1 = minTotalofIntensity*(2)
MinTotalCentroid2 = minTotalofIntensity

sumuCMaxPM = 0
sumuCMinPM = 0
sumuCMaxPMX = 0
sumuCMinPMX = 0

    #continue until difference is only 1%
while abs((MaxTotalCentroid1-MaxTotalCentroid2)/MaxTotalCentroid2) and
abs((MinTotalCentroid1-MinTotalCentroid2)/MinTotalCentroid2) >= .01:
    MaxTotalCentroid1 = MaxTotalCentroid2
    MinTotalCentroid1 = MinTotalCentroid2
    for intensity in allintensity:
        #check which centroid intensity is closer to
        distancetoMaxCentroid = abs(MaxTotalCentroid1-intensity)
        distancetoMinCentroid = abs(MinTotalCentroid1-intensity)

        if distancetoMaxCentroid <= 0 or distancetoMinCentroid <= 0:
            if distancetoMaxCentroid <= 0:
                uCMax=1
                uCMin=0
            if distancetoMinCentroid <= 0:
                uCMax=0
                uCMin=1
        else:

            uCMax = (1/distancetoMaxCentroid**(1/(fuzziness-
1))/(1/distancetoMaxCentroid**(1/(fuzziness-
1))+1/distancetoMinCentroid**(1/(fuzziness-1))))
            uCMin = (1/distancetoMinCentroid**(1/(fuzziness-
1))/(1/distancetoMaxCentroid**(1/(fuzziness-
1))+1/distancetoMinCentroid**(1/(fuzziness-1))))

        uCMaxPM = uCMax**fuzziness
```

```
            uCMinPM = uCMin**fuzziness
            uCMaxPMX = uCMaxPM*intensity
            uCMinPMX = uCMinPM*intensity

            sumuCMaxPM += uCMaxPM
            sumuCMinPM += uCMinPM
            sumuCMaxPMX += uCMaxPMX
            sumuCMinPMX += uCMinPMX

        MaxTotalCentroid2 = sumuCMaxPMX/sumuCMaxPM
        MinTotalCentroid2 = sumuCMinPMX/sumuCMinPM

position = 0.0
averageHighIntensity = sum(highIntensity)/float(len(highIntensity))

'''****************************************TEST************************************
print MaxTotalCentroid2
print MinTotalCentroid2
print (default_timer() - start)/60
end = "end"
print end
with open("C:/Other Data/School Work/Thesis/Output_intensities/" + outpath
+"_allintensities.txt" , 'w') as f:
    for s in allintensity:
        f.write(str(s) + '\n')
    f.close()
sys.exit(0)
**************************************TEST**********************************'''

#create the shapefile
wetarray = arcpy.Array()
wetarray2 = arcpy.Array()
dryarray = arcpy.Array()
dryarray2 = arcpy.Array()
mostlywetarray = arcpy.Array()
mostlywetarray2 = arcpy.Array()
mostlydryarray = arcpy.Array()
mostlydryarray2 = arcpy.Array()
wetFeaturelist = []
dryFeaturelist = []
mostlywetFeaturelist = []
mostlydryFeaturelist = []
linePosition2 = arcpy.SearchCursor(shapeLocation)
altintensityCoords = arcpy.Point()
altendIntensityCoords = arcpy.Point()
wetexists = False
dryexists = False
mostlywetexists = False
mostlydryexists = False

linecounter=0
for feature1 in linePosition2:
```

```
    position = 0.0
    position2 = 0.0
    position3 = 0.0
    position4 = 0.0
    drycount = 0
    wetcount = 0
    if not percentoverride == 0:
        while position2 <= 1:
            wet = False
            dry = False
            mostlywet = False
            mostlydry = False

            intensityLocation2 =
feature1.shape.positionAlongLine(position2,True).firstPoint
            intensityEndLocation2 =
feature1.shape.positionAlongLine(position2+segdecimalLength,True).firstPoint
            pointX2 = intensityLocation2.X
            pointY2 = intensityLocation2.Y
            pointEndX2 = intensityEndLocation2.X
            pointEndY2 = intensityEndLocation2.Y
            intensityCoords2 =str(pointX2) + " " + str(pointY2)
            altintensityCoords2 = arcpy.Point(pointX2,pointY2)
            endIntensityCoords2 = str(pointEndX2) + " " + str(pointEndY2)
            altendIntensityCoords2 = arcpy.Point(pointEndX2,pointEndY2)
            intensitytoCheckArcOutput2 =
arcpy.GetCellValue_management(intensityrasterLocation,intensityCoords2, "1")

            if not intensitytoCheckArcOutput2.getOutput(0) == "NoData":

                intensitytoCheck2 = float(intensitytoCheckArcOutput2.getOutput(0))
                distancetoMaxCentroid2 = abs(MaxTotalCentroid2-intensitytoCheck2)
                distancetoMinCentroid2 = abs(MinTotalCentroid2-intensitytoCheck2)

                if distancetoMaxCentroid2 <= 0 or distancetoMinCentroid2 <= 0:
                    if distancetoMaxCentroid2 <= 0:
                        maxMembership2=1
                        minMembership2=0
                    if distancetoMinCentroid2 <= 0:
                        maxMembership2=0
                        minMembership2=1
                else:

                    maxMembership2 = (1/distancetoMaxCentroid2**(1/(fuzziness-
1))/(1/distancetoMaxCentroid2**(1/(fuzziness-
1))+1/distancetoMinCentroid2**(1/(fuzziness-1))))
                    minMembership2 = (1/distancetoMinCentroid2**(1/(fuzziness-
1))/(1/distancetoMaxCentroid2**(1/(fuzziness-
1))+1/distancetoMinCentroid2**(1/(fuzziness-1))))

                if MaxTotalCentroid2 <= intensitytoCheck2:
```

```
                    wet = False
                    dry = True
                    mostlywet = False
                    mostlydry = False
                if MinTotalCentroid2 >= intensitytoCheck2:
                    wet = True
                    dry = False
                    mostlywet = False
                    mostlydry = False

                if MinTotalCentroid2 < intensitytoCheck2 < MaxTotalCentroid2:
                    if maxMembership2 >= 0.5:
                        wet = False
                        dry = False
                        mostlydry = True
                        mostlywet = False
                    if minMembership2 > 0.5:
                        wet = False
                        dry = False
                        mostlydry = False
                        mostlywet = True

                position2 += segdecimalLength

                if dry == True:
                    drycount += 1

                if mostlydry == True:
                    drycount += 1

                if wet == True:
                    wetcount += 1

                if mostlywet == True:
                    wetcount += 1

            else:
                position2 += segdecimalLength

    # If the amount of dry sections exceeds the overide value, the entire section
will be considered dry
    if (drycount)/(1/segdecimalLength)>=decimaloveride:

        while position3 <= 1:

            intensityLocation3 =
feature1.shape.positionAlongLine(position3,True).firstPoint
            intensityEndLocation3 =
feature1.shape.positionAlongLine(position3+segdecimalLength,True).firstPoint
            pointX3 = intensityLocation3.X
            pointY3 = intensityLocation3.Y
            pointEndX3 = intensityEndLocation3.X
```

```
            pointEndY3 = intensityEndLocation3.Y
            altintensityCoords3 = arcpy.Point(pointX3,pointY3)
            altendIntensityCoords3 = arcpy.Point(pointEndX3,pointEndY3)

            dryarray2.add(altintensityCoords3)
            dryarray2.add(altendIntensityCoords3)
            drypolyline2 = arcpy.Polyline(dryarray2,shapeLocation)
            dryarray2.removeAll()
            dryFeaturelist.append(drypolyline2)

            position3 += segdecimalLength
        #skip position segment
        position=1+segdecimalLength

# If the amount of wet sections exceeds the overide value, the entire section will be
considered wet
    if (wetcount)/(1/segdecimalLength)>=wetfilterdecimal:

        while position4 <= 1:

            intensityLocation3 =
feature1.shape.positionAlongLine(position4,True).firstPoint
            intensityEndLocation3 =
feature1.shape.positionAlongLine(position4+segdecimalLength,True).firstPoint
            pointX3 = intensityLocation3.X
            pointY3 = intensityLocation3.Y
            pointEndX3 = intensityEndLocation3.X
            pointEndY3 = intensityEndLocation3.Y
            altintensityCoords3 = arcpy.Point(pointX3,pointY3)
            altendIntensityCoords3 = arcpy.Point(pointEndX3,pointEndY3)

            wetarray2.add(altintensityCoords3)
            wetarray2.add(altendIntensityCoords3)
            wetpolyline2 = arcpy.Polyline(wetarray2,shapeLocation)
            wetarray2.removeAll()
            wetFeaturelist.append(wetpolyline2)

            position4 += segdecimalLength
        #skip position segment
        position=1+segdecimalLength

    while position <= 1:
        wet = False
        dry = False
        mostlywet = False
        mostlydry = False
        intensityLocation =
feature1.shape.positionAlongLine(position,True).firstPoint
        intensityEndLocation =
feature1.shape.positionAlongLine(position+segdecimalLength,True).firstPoint
        pointX = intensityLocation.X
        pointY = intensityLocation.Y
```

```
        pointEndX = intensityEndLocation.X
        pointEndY = intensityEndLocation.Y
        intensityCoords =str(pointX) + " " + str(pointY)
        altintensityCoords = arcpy.Point(pointX,pointY)
        endIntensityCoords = str(pointEndX) + " " + str(pointEndY)
        altendIntensityCoords = arcpy.Point(pointEndX,pointEndY)
        intensitytoCheckArcOutput =
arcpy.GetCellValue_management(intensityrasterLocation,intensityCoords, "1")

        if not intensitytoCheckArcOutput.getOutput(0) == "NoData":

            intensitytoCheck = float(intensitytoCheckArcOutput.getOutput(0))
            distancetoMaxCentroid = abs(MaxTotalCentroid2-intensitytoCheck)
            distancetoMinCentroid = abs(MinTotalCentroid2-intensitytoCheck)

            if distancetoMaxCentroid <= 0 or distancetoMinCentroid <= 0:
                if distancetoMaxCentroid <= 0:
                    maxMembership=1
                    minMembership=0
                if distancetoMinCentroid <= 0:
                    maxMembership=0
                    minMembership=1
            else:

                maxMembership = (1/distancetoMaxCentroid**(1/(fuzziness-
1))/(1/distancetoMaxCentroid**(1/(fuzziness-
1))+1/distancetoMinCentroid**(1/(fuzziness-1))))
                minMembership = (1/distancetoMinCentroid**(1/(fuzziness-
1))/(1/distancetoMaxCentroid**(1/(fuzziness-
1))+1/distancetoMinCentroid**(1/(fuzziness-1))))

            if MaxTotalCentroid2 <= intensitytoCheck:
                wet = False
                dry = True
                mostlywet = False
                mostlydry = False
            if MinTotalCentroid2 >= intensitytoCheck:
                wet = True
                dry = False
                mostlywet = False
                mostlydry = False

            if MinTotalCentroid2 < intensitytoCheck < MaxTotalCentroid2:
                if maxMembership >= 0.5:
                    wet = False
                    dry = False
                    mostlydry = True
                    mostlywet = False
                if minMembership > 0.5:
                    wet = False
                    dry = False
                    mostlydry = False
```

```
                mostlywet = True

#this is an attempt to capture areas where there is no dry channels
        if MaxTotalCentroid2/MinTotalCentroid2 <= 1.2:
            wet = True
            dry = False
            mostlydry = False
            mostlywet = False

        position += segdecimalLength

        if wet == True:
            wetexists = True
            wetarray.add(altintensityCoords)
            wetarray.add(altendIntensityCoords)
            arraynameprint = "Wet "
            #print arraynameprint
            wetpolyline = arcpy.Polyline(wetarray,shapeLocation)
            wetarray.removeAll()
            wetFeaturelist.append(wetpolyline)

        if dry == True:
            dryexists = True
            dryarray.add(altintensityCoords)
            dryarray.add(altendIntensityCoords)
            arraynameprint = "dry "
            #print arraynameprint
            drypolyline = arcpy.Polyline(dryarray,shapeLocation)
            dryarray.removeAll()
            dryFeaturelist.append(drypolyline)

        if mostlywet == True:
            mostlywetexists = True
            mostlywetarray.add(altintensityCoords)
            mostlywetarray.add(altendIntensityCoords)
            arraynameprint = "mostly wet "
            #print arraynameprint
            mostlywetpolyline = arcpy.Polyline(mostlywetarray,shapeLocation)
            mostlywetarray.removeAll()
            mostlywetFeaturelist.append(mostlywetpolyline)

        if mostlydry == True:
            mostlydryexists = True
            mostlydryarray.add(altintensityCoords)
            mostlydryarray.add(altendIntensityCoords)
            arraynameprint = "mostly dry "
            #print arraynameprint
            mostlydrypolyline = arcpy.Polyline(mostlydryarray,shapeLocation)
            mostlydryarray.removeAll()
            mostlydryFeaturelist.append(mostlydrypolyline)

    else:
```

```
            position += segdecimalLength


#if the sections exist, create the shape files
if wetexists==True:
    arcpy.management.CopyFeatures(wetFeaturelist,path + "/wet_"+outpath)
if dryexists==True:
    arcpy.management.CopyFeatures(dryFeaturelist,path + "/dry_"+outpath)
if mostlywetexists==True:
    arcpy.management.CopyFeatures(mostlywetFeaturelist,path + "/mostly_wet_"+outpath)
if mostlydryexists==True:
    arcpy.management.CopyFeatures(mostlydryFeaturelist,path + "/mostly_dry_"+outpath)

#delete excess data
del wetFeaturelist
del dryFeaturelist
del mostlywetFeaturelist
del mostlydryFeaturelist

#print to screen the global max and min centroid results and time to completion
print MaxTotalCentroid2
print MinTotalCentroid2
print (default_timer() - start)/60
end = "end"
print end

'''****************************************TEST************************************
with open("C:/Other Data/School Work/Thesis/Output_intensities/" + outpath
+"_allintensities.txt" , 'w') as f:
    for s in allintensity:
        f.write(str(s) + '\n')
    f.close()
****************************************TEST************************************'''
```

REFERENCES

Abu-Aly, T., Pasternack, G., J.R., W., Barker, R., Massa, D., & Johnson, T. (2014). Effects of LiDAR-Derived, Spatially Distributed Vegetation Roughness on Two-Dimensional Hydraulics in Gravel-Cobble River at Flows of 0.2 to 20 Times Bankfull. *Geomorphology*, 468-482.

Alonzo, M., Bookhagen, B., & Roberts, D. A. (2014). Urban Tree Species Mapping Using Hyperspectral and LiDAR Data Fusion. *Remote Sensing of Environment*, 70-83.

Antonarakis, A., Richards, K., & Brasington, J. (2008). Object-Based Land over Classification Using Airborne LiDAR. *Remote Sensing of Environment*, 2988-2998.

Awrangjeb, M., Ravanbakhsh, M., & Fraser, C. S. (2010). Automatic Detection of Residental Buldings Using LiDAR Data and Multispectral Imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 457-467.

Beucher, A., Frojdo, S., Osterholm, P., Martinkauppi, A., & Eden, P. (2014). Fuzzy Logic for Acid Sulfate SOil Mapping; Application to the Southern Part of the Finnish Coastal Areas. *Geoderma*, 21-30.

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The Fuzzy c-Means Clustering Algorithm. *Computers & Geosciences*, 191-203.

Bigdeli, B., Samadzadegan, F., & Reinartz, P. (2015). Fusion of Hyperspectral and LiDAR data Using Decision Template-Based Fuzzy Multiple Classifier System. *Internation Journal of Applied Earth Observation and Geoinformation*, 309-320.

Bowen, Z. H., & Waltermire, R. G. (2002). Evaluation of Light Detection and Ranging (LiDAR) for Measuring River Corridor Topography. *Journal of the American Water Resources Association*, 33-41.

Brzank, A., & Heipke, C. (2006). Classification of LiDAR Data Into Water and Land Poitns in Coastal Areas. *Remote Sensing and Spatial Information Sciences*, 197-202.

Brzank, A., Heipke, C., Goepfert, J., & Soergel, U. (2008). Aspects of Generating Precise Digital Terrain Models in the Wadden Sea from LiDAR-Water Classification and Structure Line Extraction. *ISPRS Journal of Photogrammetry & Remote Sensing*, 510-528.

Cavalli, M., Tarolli, P., Marchi, L., & Fontana, G. D. (2008). The Effectiveness of Airborne LiDAR Data in the Recognition of Channel-Bed Morphology. *Catena*, 249-260.

Chen, Z., & Gao, B. (2014). An Object-Based Method for Urban Land Cover Classification Using Airborne Lidar Data. *IEEE Journal of Selected Topic in Applied Earth Observations and Remote Sensing*, 4243-4254.

ESRI. (2014, March 5). *What is LiDAR Intensity Data?* Retrieved from ArcGIS Resources: http://resources.arcgis.com/en/help/main/10.2/index.html#//015w00000051000000

Fernandez-Lozano, J., Gutierrez-Alonso, G., & Fernandez-Moran, M. A. (2015). Using arborne LiDAR sensing technology and aerial orthoimages to unravel roman water supply systems and gold works in NW Spain (Eria valley, Leon). *Journal of Archaeological Science*, 356-373.

Gilvear, D., Tyler, A., & Davids, C. (2004). Detection of Estuarine and Tidal River Hydromorphology Using Hyper-Spectral and LiDAR Data; Forth Estuary, Scotland. *Estuarine Coastal and Shelf Science*, 379-392.

Gleason, C. J., & Jungho, I. (2012). Forest Biomass Estimation from Airborne LiDAR Data Using Machine Learning Approaches. *Remote Sensing of Environment*, 80-91.

Han, W., Zhao, S., Feng, X., & Chen, L. (2014). Extraction of Multilayer Vegetation Coverage Using Airborne LiDAR Discrete Points with Intensity Information in Urban Areas: A Case Study in Nanjing City, China. *Internation Journal of Applied Earth Observation and Geoinformation*, 56-64.

Hartzell, P., Glennie, C., Biber, K., & Khan, S. (2014). Application of Multispectral LiDAR to Automated Virtual Outcrop Geology. *ISPRS Journal of Photogrammetry and Remote Sensing*, 147-155.

Hofle, B., Vetter, M., Pfeifer, N., Mandlburger, G., & Stotter, J. (2009). Water Surface Mapping from Airborne Laser Scanning Using Signal Intensity and Elevation Data. *Earth Surface Processes and Landforms*, 1635-1649.

Jun, G., & Ghosh, J. (2011). Spatially Adaptive Classification of Land Cover with Remote Sensing Data. *IEEE Transactions on Geoscience and Remote Sensing*, 2662-2673.

Jutzi, B., & Gross, H. (2009). Normalication of Lidar Intensity Data Based on Range and Surface Incedence Angle. *Proceedings of Laserscanning 2009* (pp. 213-218). Paris: IAPRS.

Karnik, N. N., & Mendel, J. M. (1998). Introduction to Type-2 Fuzzy Logic Systems. *IEEE World Congress on Computational Intelligence* (pp. 915-920). Anchorage: IEEE.

Kassalainen, S., Hyyppa, H., Kukko, A., Litky, P., Ahokas, E., Hyyppa, J., . . . Pyysalo, U. (2009). Radimetric Calibration of LIDAR Intensity With Commercially Available Reference Targets. *IEEE Transactions on Geoscience and Remote Sensing*, 588-598.

Kim, S., Wang, D., & Medeiros, S. C. (2015). Flowing Channel Network Identification Based on Intensity of LiDAR Returns. Orlando, Florida: Department of Civil, Environmental, and Construction Engineering, University of Central Florida.

Korpela, I. S. (2008). Mapping of Understory Lichens with Airborne Discrete-Return LiDAR Data. *Remote Sensing of Environment*, 3891-3897.

Kronseder, K., Ballhorn, U., Bohm, V., & Siegert, F. (2012). Above Ground Biomass Estimation Across Forest Types at Different Degradation Levels in Central Kalimantan Using LiDAR Data. *International Journal of Applied Earth Observation and Geoinformation*, 37-48.

Lodna, S. K., Fitzpatrick, D. M., & Helmbold, D. P. (2007). Aerial LiDAR Data Classification Using Expectation-Maximization. *SPIE 6499.* SPIE.

Meesuk, V., Vojinovic, Z., Mynett, A. E., & Abdullah, A. F. (2015). Urban Flood Modelling Combining Top-View LiDAR Data with Ground-View SfM Observations. *Advances in Water Resources*, 105-117.

Melin, P., & Castillo, O. (2014). A Review on Type-2 Fuzzy Logic Applications in Clustering, Classification and Pattern Recognition. *Applied Soft Computing*, 568-577.

Michez, A., Piegay, H., Toromanoff, F., Brogna, D., Bonnet, S., Lejeune, P., & Claessens, H. (2013). LiDAR Derived Ecological Integrity Indicators for Riparian Zones: Application to the Houille River in Southern Belgium/Northern France. *Ecological Indicators*, 627-640.

NOAA, N. O. (2012). Lidar 101: An Introduction to Lidar Technology, Data, and Applications. NOAA Coastal Services Center.

Petrou, Z. I., Kosmidou, V., Manakos, I., Stathaki, T., Adamo, M., Tarantino, C., . . . Petrou, M. (2014). A Rule-Based Classification Methodology to Handle Uncertainty in Habitat Mapping Employing Evidential Reasoning and Fuzzy Logic. *Pattern Recognition Letters*, 24-33.

Ross, T. J. (2010). *Fuzzy Logic with Engineering Applications, Thrid Edition.* John Wiley and Sons, Ltd.

Schackelford, A. K., & Davis, C. H. (2003). A Hierarchical Fuzzy Classification Approach for High-Resolution Multispectral Data Over Urban Areas. *IEEE Transactions on Geoscience and Remote Sensing*, 1920-1932.

Schumacher, J., & Christiansen, J. R. (2015). Forest canopy water fluxes can be estimated using canopy structure metrics derived from airborne light detection and ranging (LiDAR). *Agricultural and Forest Meteorology*, 131-141.

Singh, K. K., Vogler, J. B., Shoemaker, D. A., & Meentemeyer, R. K. (2012). LiDAR-Landsat Data Fusion for Large-area Assessment of Urban Land Cover: Balancing Spatial Resolution, Data Volume and Mapping Accuracy. *ISPRS Journal of Photogrammetry and Remote Sensing*, 110-121.

Smeeckaert, J., Mallet, C., David, N., Chehata, N., & Ferraz, A. (2013). Large-Scale Classification of Water Areas Using Airborne Topographic LiDAR Data. *Remote Sensing of Environment*, 134-148.

Song, J.-H., Han, S.-H., Yu, K., & Kim, Y.-I. (2002). Assessing the Possibility of Land-Cover Classification Using LiDAR Intesnity Data. *Remote Sensing and Spatial Information Sciences*, 259-262.

Strahler, A. N. (1969). *Physical Geography: Third Edition.* New York: John Wiley and Sons, Inc.

Sutinen, R., Hyvonen, E., Middleton, M., & Ruskeeniemi, T. (2014). Airborne LiDAR detection of postglacial faults and Pulju moraine in Palojarvi, Finnish Lapland. *Global and Planetary Change*, 24-32.

Val, M., Iriarte, E., Arriolabengoa, M., & Aranburu, A. (2014). An Automated Method to Extract Fluvial Terraces from LiDAR Based High Resolution Digital Elevation Models: The Oiartzun Valley, a Case Study in the Cantabrian Margin. *Quaternary International*, 1-9.

Vianello, A., Cavalli, M., & Tarolli, P. (2009). LiDAR-Derived Slopes for Headwater Channel Network Analysis. *Catena*, 97-106.

Wolf, W. L., & Zissis, G. J. (1978). *The Infrared Handbook.* Washington: Environmental Research Institue of Michigan.

Yan, W. Y., & Shaker, A. (2014). Radimetric Correction and Normalization of Airborne LiDAR Intensity Data for Improving Land-Cover Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 7658-7673.

Yang, P., Ames, D. P., Fonsec, A., Anderson, D., Shrestha, R., Glenn, N. F., & Cao, Y. (2014). Waht is the Effect of LiDAR-Derived DEM resolution on Large-Scale Watershed Model Results? *Environmental Modelling & Software*, 48-57.

Yoon, J.-S., Shin, J.-I., & Lee, K.-S. (2008). Land Cover Characterisitcs of Airborne LiDAR Intensity Data: A Case Study. *IEEE GeoScience and Remote Sensing Letters*, 801-805.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 338-353.

Zadeh, L. A. (1968). Fuzzy Algorithms. *Information and Control*, 94-102.

Zadeh, L. A. (1971). Similarity Relations and Fuzzy Orderings. *Information Sciences*, 177-200.

Zhou, W. (2013). An Object-Based Approach for Urban Land Cover Classification: Integrating LiDAR Height and Intensity Data. *IEEE Geoscience and Remote Sensing Letters*, 928-931.

Zhu, A. X., Hudson, B., Burt, J., Lubich, K., & Simonson, D. (2001). Soil Mapping Using GIS, Expert Knowledge, and Fuzzy Logic. *Soil Science Society of American Journal*, 1463-1472.