Colorful Math: Developing Algorithmic Methodology to Visualize and Analyze the Dynamics of a Deciduous Tree

Shayne T. O'Brien
Department of Mathematics
SUNY Geneseo
Geneseo, NY 14454
sto2@geneseo.edu

Bo Song
Baruch Institute of Coastal Ecology and Forest Science
Clemson University
Georgetown, SC 29442
bosong@clemson.edu

Brian Williams
Baruch Institute of Coastal Ecology and Forest Science
Clemson University
Georgetown, SC 29442
bwilli8@clemson.edu

Shaowu Bao
Marine & Wetland Science
Coastal Carolina University
Conway, SC 29528
sbao@coastal.edu

Abstract The purpose of this study is to determine if it is possible to use LIDAR (light detection and ranging) point cloud data collected using a FARO Focus^{3D} laser scanner to: 1) generate a high quality 3D virtual representation of a scanned tree and reduce noise due to environmental variability; 2) develop an algorithm to differentiate between different sections of trees based on RGB values associated with these point clouds and calculate the respective volumes of these sections; and 3) project leaf color change dynamics over time. The visualization tools used are Matlab and R. Additional analysis was done with a high performance computing cluster (Clemson University's Palmetto cluster). The data is of an American Sycamore *Platanus occidentalis* over a period of eight weeks as the tree transitions from fall into winter. The conclusions of this study have implications in outlining innovative approaches to three-dimensional tree dynamic measurement using advanced technology and tools. These approaches can provide the baseline information of a forest structure for greater understanding of how a tree changes seasonally, as well as the establishment of methodology for making laser scanning software and Matlab's Image Processing Toolbox more flexible to use. The visualization aids in this study can inform and be used to estimate with greater precision the effects that disturbances, such as fires and hurricanes, have on forest structures.

Keywords LIDAR, Matlab, R, HPC, laser scanning, visualization, point cloud data, color quantization, forestry

I. Introduction

Although terrestrial laser scanning is rapidly becoming the standard in forestry research studies, there lacks literature that lays out methodology on how to appropriately handle, visualize, and quantify the point cloud data that laser scanners collect. As a result of this, the capabilities of the technological advances within laser scanning have not been fully capitalized on. What is more, current algorithms for non-destructive volumetric computation are based on gross estimation. While these results are very important in providing the baseline information of a forest structure, they often lack an impressionable degree of accuracy when compared to destructive volume calculation.

This research is funded by NSF ACI Grant No. 1359223.

In studying tree dynamics and the effects that disturbances such as fires and hurricanes can have on forest structures, it is necessary to quantitatively estimate volume. This is commonly done by using allometric equations. Although these are popular in the field of forestry, they heavily depend on knowing a tree's height and diameter. This constitutes a major limitation due to the intensive fieldwork required to both gather these measurements and harvest the vegetation needed to derive these equations. What is more, a single allometric equation cannot necessarily be applied to more than one component of a given tree species [1]. The authors of this paper believe laser scanning is the solution to these problems; improvements in scanning technologies are enabling researchers to estimate tree size and dynamics more accurately than ever before.

This paper investigates the feasibility of using a supercomputing cluster and point cloud data collected by a Faro Technologies (FARO) Focus^{3D} laser scanner to develop algorithmic methodology that accomplishes the following objectives: (1) denoise and 3D visualize the original data; (2) differentiate between the crown, branches and trunk of a deciduous tree and calculate the respective volumes of these components and; (3) project leaf color change dynamics over time. By analyzing the colors of a tree via collected light detection and ranging RGB data, the researchers have taken an innovative approach to differentiate tree components and estimate respective volume.

II. Background

Laser scanners work by stimulating the emission of photons from excited atoms and molecules. This generates an intense beam of monochromatic light that is characterized by a static phase difference between its wavelengths. The generated laser beam is deflected by a small mirror to a desired location with respect to the initial positioning of the scanner. The mirror rotates around an axis perpendicular to the plane that the scanner is placed on, causing it to be able to collect two-dimensional measurements. The third dimension, which is necessary to collect point cloud data, is included via the addition of a secondary rotational axis. The laser beam travels through this space medium and hits any surface within the detection range of its field of vision. The beam is absorbed, reflected, or transmitted depending on the reflectivity of the

surface. The return signal is then measured by the scanner receiver [2].

III. Data

The data were collected of two sycamore trees and a cypress using a FARO Focus^{3D} laser scanner over a period of eight weeks as the trees transitioned from fall into winter and shed their leaves. Five scans were taken once a week every Monday for the duration of the data collection period. Scanning began at 10:00 A.M. and concluded by 12:00 P.M. on each day. The collected point cloud consists of high spatial frequency data and is in the formatting of an XYZ, RGB matrix. The focus of this study is the smaller of the two sycamores, chosen because its structure was only minimally obscured by neighboring trees as compared to the larger sycamore and the cypress.

IV. Materials and Methods

The steps below describe the algorithmic methodology developed by the researchers to generate high quality 3D virtual representations of scanned tree data, reduce noise due to environmental variability, differentiate between different components of trees based on RGB values and calculate the respective volumes of these components, and project leaf color change dynamics over time. These steps include: reformatting and importing the data, visualizing perceptual color maps, denoising, compensating for the skewing caused by the denoising process, generating 3D RGB histograms of each dataset's color distribution, allowing for point cloud compatibility within Matlab's Image Processing Toolbox, and color quantizing.

Data Collection: FARO

The laser scanner used in this study is the Focus^{3D} X 130, a mid-range scanner that has a range from 0.60 meters to 130 meters with a distance accuracy of up to ±2mm. The resolution of the scanner was set at ¼ for all scans. Five scans were taken from preset positions around the small sycamore for each data collection week.

Programming Software: Matlab, R, and Scene

Matlab was the primary programming language used to manipulate the data. In the context of this paper, manipulating the dataset involves denoising to remove perceptually skewing data that is resultant of variability in the environmental conditions that the scans were taken in, i.e., outdoors. Without denoising, visualizing the color distribution in an effort to analyze and project leaf color change dynamics over time would be impractical. Hence, a denoising algorithm was applied to the datasets to compensate for the effects that uncontrollable variables such as lighting, wind, and precipitation had on the quality of the collected data.

R programming language was the primary visualization tool for the study. R's interactive graphical system along with its extensibility, generality, demonstrated ability to generate high quality visualizations, and exceptional capacity for handling large point cloud data sets with relative ease made the programming language the ideal visualization tool for this study. The *plot3d* function, contained within the *rgl* package used for 3D visualization using OpenGL, outperformed several three-dimensional plotting software packages including: Matlab, Unity3D, Houdini, MeshLab, Spartan Student, and ParaView [3]. This outperformance was measured through cross comparisons between photographs of the trees and the visualizations generated by each software.

SCENE 3D is a software specifically designed for working with data collected by a FARO Focus^{3D} laser scanner. SCENE works by processing and managing scanned data efficiently using automatic object recognition, scan registration and positioning [2]. All preliminary editing of the trees out of the full point cloud, which included a large area of the surrounding environment, was completed using SCENE.

Step 1: Data Importation

The data that is collected using the FARO Focus^{3D} scanner is first imported into SCENE 3D so that the full point cloud of the trees and the environment around them may be edited. The editing process involves carefully cropping all of the point cloud data that is not of interest, i.e. anything that does not belong to the structure of the tree. Once this has been completed, the data must be exported from SCENE 3D into .xyz file format and then reformatted into .txt file format. The default delimiter in .txt files is a space character (" ").

After the format has been converted to .txt, the researchers imported the data into Matlab. This can be done using either the built in "Import Data" tool located within the "Home" tab of Matlab's GUI, or by making use of the *readtable* and *table2array* functions. Using these functions was found by the researchers to be more time efficient than the "Import Data" tool. Columns were assigned to independent variables and named based on correspondent week of data collection and whether data cells contained x, y, z, red, green, or blue values. Renaming the columns to independent variables was found by the researchers to reduce human error within this study caused by incorrect indexing of matrices.

Step 2: Color Maps

After assigning each of the columns to named variables, they will all be in the shape of a vector. This causes perceptual issues when attempting to import the data into the Image Processing Toolbox of Matlab for the denoising process. To address this, the researchers developed two different perceptual color map representations of the datasets. Both color maps are intended as visual aids in the denoising process. Each of the two maps makes use of a custom function called *SquareM* written by the researchers that reshapes the vector based on length into the smallest square matrix possible. If the length of the vector is not a square number, *SquareM* will elongate the vector using NaN (Not-a-Number) until it is a square number and can thus be reshaped into a square matrix. NaN is a numeric data type used for representing undefined or un-representable values.

The first color map is created by sorting each of the red, green, and blue dimensions of the data according to size using the *sort* function. The sort function reorganizes the vector according to increasing value. After this has been completed,

the red, green, and blue values are each assigned their own dimension in a new, multidimensional array variable within Matlab and then *SquareM* is applied to the new variable. The result of this can be seen in Fig. 1.

Fig. 1 Perceptual color map one of two.

This color map is useful for visualizing the proportion of the data has been reassigned to black during the denoising process. The procession of color appears uniform to the user and goes from black, through the shades of green, ultimately to white.

The second color map generation is similar to that of the first. The only difference between this and the first is that the red, green, and blue vectors are not sorted. This visualization is useful in removing colors based on perceived threshold and can be seen in Fig. 2.

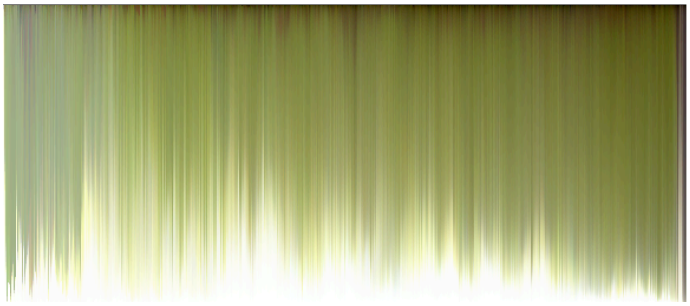

Fig. 2 Perceptual color maps two of two.

For both color maps, it is important to normalize the square multidimensional array by converting its type into unsigned integers of eight bits (uint8) using the built-in Matlab function *uint8*. This data type contains only integers in the range [0, 255] and is commonly used for colorized graphics.

Step 3: Color Thresholding

The Image Processing Toolbox is downloadable software within Matlab. It provides a set of algorithms, functions, and applications for processing, analyzing, and visualizing images and videos [4]. The reformatting of the data completed in the previous steps allows users to extend the capabilities of this software to manipulate multidimensional and point cloud data.

Once the color maps of the data have been generated, we can use the Color Thresholder application within the Image Processing Toolbox to automatically load them for denoising. The screen that appears asks the user to pick one of four color spaces: RGB, HSV, YcbCr, or L*a*b*. The research team recommends using either the HSV or L*a*b* color spaces as they were found to be the most effective in denoising. These two denoising interfaces are shown in Fig. 3.

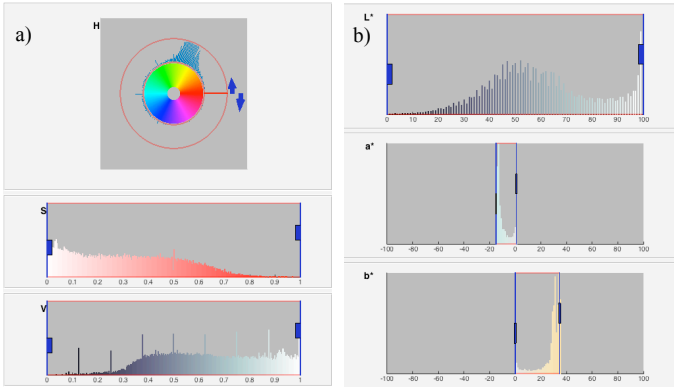

Fig. 3 Pictured is how the denoising graphical user interfaces appear to the user. a) HSV color space. b) L*a*b* color space.

After the data has been denoised in a way that is judged acceptable by the user, it is necessary to utilize the "Export Function" option, which is located under the "Export Images" tab. After the function has been exported, it is automatically loaded into the user's Matlab editor dock. The call for this function is named *createMask* by default and can be customized according to the user's preference. Customizations include renaming the function and more precisely editing the channel thresholds interval [0.000, 1.000] and renaming the function. After customizations have been completed, this function should be applied to the multidimensional array containing the red, green, and blue columns of the original point cloud data.

Step 4: Blackout and Whiteout

Subsequently, the next step is to compensate for how the denoised data is handled because the Color Thresholder assigns all denoised data to black. For example, if the user specified to remove white, all element indices where the first, second and third dimensions have values 255 will be reassigned to zero. Since the objective of denoising the data is to remove artifacts and thereby more accurately reflect the true appearance of the sycamore, this needs to be addressed.

The blackout and whiteout scripts work by initially detecting whether or not the data has been altered during thresholding and correcting if it has been back to the integer interval [0, 255] for red, green, and blue cells. This is done by using Matlab's *max* function and multiplying the array by 255 if the max is less than or equal to one. Figure 4a shows how the color distribution may not be properly visualized if this check is not completed. Based on this check and depending on if the user is using the whiteout or the blackout script, all black or white points are reassigned to NaN.

Step 5: 3D RGB Histogram Generation

A script was developed to visualize the relative color distribution of images. It is available to any Matlab license owner via the MathWorks online community [5]. Modifications were made to this script in order to make it compatible with point cloud data. The script works by assigning each RGB cell to a histogram bin, computing the frequency of each of these cells, reshaping the three-dimensional array for the point cloud data, compensating for

the possibility of non-uniform length histogram cells which may be caused by how cell frequencies are calculated, emphasizing smaller frequency colors so they are visible on the plot, and then plotting the 3D histogram. See Fig. 4 for example visualizations.

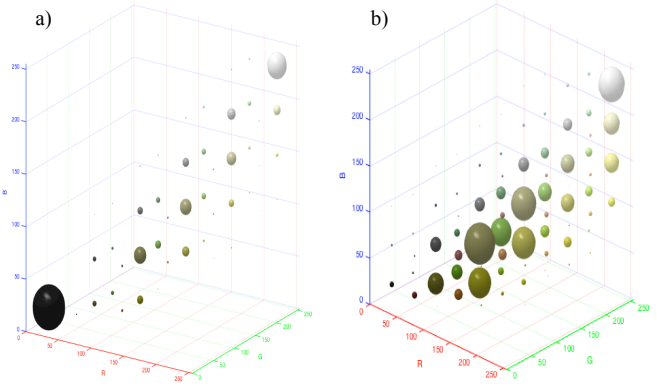

Fig. 4 3D RGB histograms. a) Skew caused by black after denoising. b) Color distribution of week 0 before denoising.

Each color is represented by a sized and colored sphere. The color of corresponds to the RGB values and the size to the relative frequency of the color being represented. In the modified script used by the researchers the gamma value, which is used in emphasizing smaller frequency colors, was hardcoded in at 0.50. The script compensates for NaN values.

Step 6: Point Cloud Compatibility

Once the data has been denoised and the color distribution has been visualized, the next step is to make the denoised point cloud data capable of being visualized. The researchers developed a script that asks for and uses inputs from the user to do this. This is possible by considering three different matrices: the original, the denoised, and a created zeros matrix that is the same size as the other two. Potential differences in size between matrices are accounted for.

If all three of these values are equal, the script will change the corresponding row of the zeros matrix to the XYZ and RGB row of the original matrix so that it will be plotted. If these three dimensions are not equal in the original matrix and the denoised matrix, the script assigns the corresponding rows in the zeros matrix to NaN for all columns. After the script has finished copying all appropriate rows of the original matrix to the zeros matrix, it next finds and removes all rows of the zeros matrix that contain a value of NaN. Finally, the script uses the *dlmwrite* function to transfer the data to a .txt file for visualization in R. The user specifies the name of this .txt file.

Visualization in R involves reading in the data of the new .txt file name, specifying the delimiter used by *dlmwrite* where the default is a comma (“,”), renaming the columns of the data appropriately, normalizing the RGB data to be contained in [0, 255], setting the size of each pixel to 0.001 with point type ‘p’, and finally removing undesired graphical labels and axes.

Step 7: Component Differentiation via *rgb2ind* Color Quantization/the L*a*b* Color Space

Tree component differentiation can be achieved in two different ways. The first method uses *rgb2ind*; a built-in Matlab function that reassigns the color frequencies to *n* specified bins based on minimum variance quantization of the RGB vectors. The number of bins can be verified by converting the data to grayscale and showing the histogram of this converted dataset using *rgb2gray* and *imshow* successively. After reducing the bins to a manageable number, repeat Step 3, 4, and 6.

An alternative to *rgb2ind* is to use the L*a*b* (Lab) color space option within the Color Thresholder. This color space offers a three-dimensional color space in which perceived color differences correspond to colorimetrically measured distances. L* represents the lightness whereas a* and b* represent opposite color dimensions; a* measures from green (-a) to red (+a) and b* measures from blue (-b) to yellow (+b) [6]. The Lab color space has the advantage over *rgb2ind* by maintaining the full color distribution, although it may be found to be less intuitive compared to using *rgb2ind*. To use the L*a*b* color space to differentiate, repeat Step 2, 3, 4, and 6. When repeating Step 3, focus on thresholding the L* and b* axes from the left in the interface and a* from the right.

Step 8: Volume Computation

The volume computation script relies heavily on calculating the absolute extrema of the spatial data and using these values as inputs for the *meshgrid* and *griddata* functions within Matlab. This creates a convex box around the point cloud dataset. A linear spacing of 0.01 was used for each of these functions, which correlates to the volume being calculated in cm^3. These two functions interpolate the scattered data and use the Delaunay triangulation algorithm to subdivide the point cloud. All XYZ points within the data cloud are placed on the circumcircle of the tetrahedron generated by the *griddata* function; no points lie within the tetrahedron. All space within each of the tetrahedron generated by the triangulation is calculated as partial volume. It is highly recommended by the researchers to use natural neighbor Delaunay triangulation for best results. Do not denoise datasets in any way before volume computation.

Step 9: Shell Script

A shell script has been created by the research team that runs all of the previous steps in an appropriate order. This was developed to improve the user friendliness of our algorithms. The shell script asks the user to specify the variable names of the data and then uses these to run the above processes with minimal user interaction. The primary step that requires user interaction is Step 3, during denoising. All visualizations appear after the color thresholding has been completed by the user.

V. Results

The researchers were able to successfully accomplish each of the outset objectives of this study: denoise and visualize the point cloud data, differentiate components of the tree and calculate respective volumes, and analyze and project leaf color change dynamics over time. Each of these was completed following the algorithmic methodology outlaid in

this research paper. Visualization played an important role in this study in finding and fixing errors.

By manipulating the hue, saturation, and value (luminance) thresholds within the HSV color space, the researchers were able to remove perceptually discernable noise caused by environmental variability. A before and after denoising comparison is shown in Fig. 5.

Fig. 5 Tree visualization before (left) and after (right) denoising process.

During component differentiation, utilizing *rgb2ind and* the HSV color space within the Color Thresholder proved moderately effective in removing the majority of the trunk and branches with *n*=65. A histogram visualization of this is pictured in Fig. 6. At this number of bins, some of the tree trunk's fringe was not removed during thresholding due to the nature of the color quantization function. While the L*a*b* yielded better results than *rgb2ind* methods, it involved much more skillful and precise thresholding which was difficult to consistently accomplish. Results for both methods included slight overestimation when compared to projected tree volume. This was expected by the researchers.

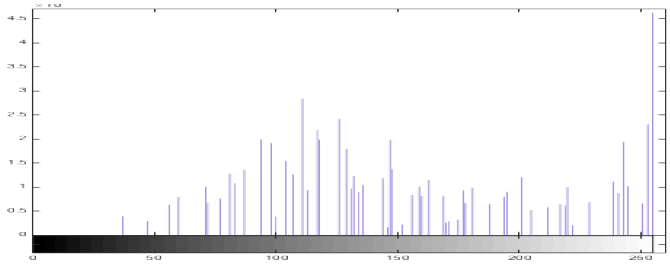

Fig. 6 Color quantization to reduce color distribution within data to 65 colors. Size of data is preserved.

The 3D RGB histogram representation of the dataset color distributions is the first step towards constructing a color projection model that could be incorporated into simulation studies of forests. Other visualizations of the color space planes helped further the researchers' understanding of leaf color change dynamics over time.

VI. Discussion

In Step 3 of Materials and Methods, it is stated that the user should use either the L*a*b* or HSV color space to denoise the data. The decision is ultimately a matter of user preference. At the same time, color theory literature suggests that HSV is generally found to be the most comfortable and intuitive to use of the color spaces [6]. This being said, L*a*b* was determined by the researchers to be more effective in denoising than HSV despite unfamiliarity. Nevertheless, it is critical for the user to carefully analyze the 3D RGB histogram of the data before and after denoising to more fully see the effects of the denoising process. It may be necessary to apply Step 4 and 5 to the original data before completing Step 3. The shell script does this automatically.

Using the command *imhist(rgb2gray(colormap))* reduces the risk of subjectivity in the denoising process, which can result from removing too much of the data. The logic behind this is that the command visualizes the non-normality of the original RGB data and allows the user to threshold according to expected trends of non-noisy data (Fig. 7).

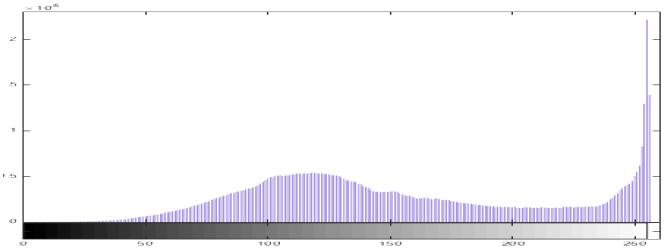

Fig. 7 Grayscale histogram displays non-normality of the week 0 data.

By the Central Limit Theorem, the non-normality of the data lends considerable support to the research hypothesis that the datasets require denoising to more accurately visualize of the true nature of the tree over time. Furthermore, the laser scanner takes high quality photographs of its field of vision at the start of each scan. In one of the photographs for the week 3 data, a crepuscular ray was registered. This caused a veil illumination effect in the right half of the photograph. Through analyzing the 3D RGB histogram of the photo's area of interest, it was discovered that the majority of this photo had the same color range as the areas of the RGB data that were suspected to be noise. This finding endows additional support to the aforementioned hypothesis.

The researchers found it advantageous to use the *rgb2ind* function to identify the regions that should be removed to denoise and then move into the L*a*b* color space to threshold more precisely. Support for removal of specific regions can be attained through comparative analysis of generated 3D RGB histograms. Doing this minimizes the chance for over-denoising of data.

By means of the *griddata* function, the researchers used natural neighbor Delaunay triangulation as the method for computing volume due to the data being sparse in some areas and dense in others. It is important to note that while this algorithm worked well for the first six weeks of the data collection period while the tree still has a full or partial crown, it inevitably overestimated the volume of trees with empty leaf crowns. This may be due to insufficient point density of the data. Results suggest that high quality, very dense data may be more suitable for volumetric estimations.

Visualization of the Delaunay triangulation using the *trisurf* function played an important role in the researchers' decision to use this algorithm. These visualizations even helped to detect outliers without the use of statistical methodology. This is shown in Fig. 8.

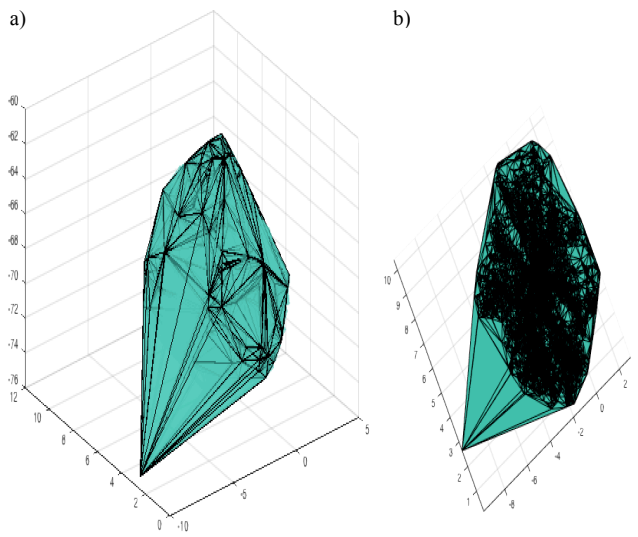

Fig. 8 Using Delaunay triangulation visuals to aid in outlier detection. a) 3D Delaunay triangulation outlier visual. b) Figure 8a converted to 2D.

Important insights into the color distribution of the sycamore were achieved through advanced visualization techniques in R and Matlab. These insights were found through analysis of generated graphics. This is representative of the first step towards more realistic and informative forest study simulations. To the researchers' knowledge, current literature does not speak to these insights. This opens the door of opportunity for mathematical color change projections, and is discussed in detail in Section VIII of this paper.

VII. Limitations

The uncontrollable variables mentioned in Section IV: Programming Software proved to be especially problematic in the last two weeks of the data collection period; the sky was overcast at the scanning location on both of these collection days, thereby causing the background of the scans to be a gray shade. While this may not have been a problem in earlier weeks when the tree still had a full crown, the bark of an American Sycamore peels as it transitions from fall into winter, thereby revealing its characteristic gray-white cambium. This resulted in the laser scanner having difficulties in correctly recording the RGB distribution of the sycamores.

The structure of the sycamore was difficult to fully capture due to limitations in the research team's ability to scan from more than one altitudinal vantage point. The data becomes increasingly sparse as the height of the tree increases. This is a direct result of the scans being taken only from ground level and the high leaf density within the tree's crown. Additionally, the researchers did not use the maximum resolution setting offered by the laser scanned due to time constraints during data collection. Instead, the guidelines outlaid by the FARO training manual for scanning outdoors spaces were followed [6]. Increasing the resolution, number of scans, and number of vantage points would result in higher quality, denser point cloud data.

Due to their close proximity, extraneous branches proved difficult to remove during the editing process of the data in SCENE 3D. This could have led to higher volume estimations due to overlapping components from separate trees being included in the smaller sycamore's data files. The researchers found that measuring single trees within a crowded environment is difficult even for highly trained research associates. Moreover, environmental variability between each data collection week played an important role in the quality of the datasets.

This methodology was established over a period of six weeks while the research team's REU student had access to the data.

VIII. Implications and Future Work

This research study improves the flexibility of laser scanning technology through supplementary Matlab scripts and demonstrates that the default capabilities of the Image Processing Toolbox can be enhanced to handle three-dimensional data; currently, this application is only branded for use with video and images. The research team accomplished the objectives of this study – all of which were concerned with three-dimensional data – using the Image Processing Toolbox software in spite of its branding.

If given more time, the research team would further optimize the methodology established in this study. Additionally, the potential for applying statistics or machine learning for greater insight into the nuisances of the data appears to be very high. A regression analysis may prove to be highly effective in the projection of the leaf color change dynamics over time. The results from these future analyses could be incorporated in more precise and accurate forestry research simulations. These improvements may provide greater insight into how disturbances affect forest structures relative to season.

With the possible exception of volumetric computation due to high diversity of tree structure between species, the methodology in this study can be applied to any tree structure with respect to the outset objectives. Further testing is required to determine adaptations are needed for usage on multi-tree datasets.

Acknowledgements

The researchers would like to graciously thank Dr. Vetria Byrd, REU site coordinator and Principal Investigator, for serving as our REU student's visualization mentor over the course of this study. The team would also like to thank Dr. Lori Tanner for her input in the editing of this paper.

References

[1] Feliciano, E., Wdowinski, S., & Potts, M. (2014). Assessing Mangrove Above-Ground Biomass and Structure using Terrestrial Laser Scanning: A Case Study in the Everglades National Park. *Wetlands*, *34*(5), 955-968. doi:10.1007/s13157-014-0558-6

[2] Faro Technologies. (2014). *Focus3D X 130: Training manual*. Lake Mary, FL: Author.

[3] Daniel Adler, Duncan Murdoch and others (2014). rgl: 3D visualization device system (OpenGL). R package version 0.93.1098. http://CRAN.R-project.org/package=rgl

[4] Mathworks. (2015). *Image Processing Toolbox: User's Guide* (r2015a). Retrieved July 01, 2015 from http://www.mathworks.com/help/releases/R2015a/pdf_doc/images/images_tb.pdf

[5] Rajmic, P. (2013). *3D histogram of RGB image - Mathworks.com*. Retrieved 20 June 2015, from http://www.mathworks.com/matlabcentral/fileexchange/38685-3d-histogram-of-rgb-image

[6] Birren, F., & Cleland, T. (1969). *A grammar of color* (pp. 11-40). New York: Van Nostrand Reinhold Co.