THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

SCHOOL OF ENGINEERING

Micromechanical Modeling of Electrically Conductive Polymer Composites for Fuel Cell Bipolar Plates

MAX MYERS
SUMMER 2023

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Mechanical Engineering
with honors in Mechanical Engineering

Reviewed and approved* by the following:

Adam S. Hollinger
Associate Professor of Mechanical Engineering
Thesis Supervisor

Oladipo Onipede
Associate Professor of Mechanical Engineering
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

Hydrogen fuel cells use a chemical reaction to produce energy, yielding only water and heat as byproducts. One fuel cell yields a small amount of energy, but a stack of fuel cells can power cars, trains, or any multitude of other devices. Bipolar plates are the parts of the fuel cell that provide structure to the cell and stacks of fuel cells. They are commonly machined from steel or graphite. These materials are rigid to support stacks of cells yet also conductive to route energy for use in applications. Hydrogen fuel cells are underused in commercial settings in part due to the high cost of these materials and machining practices used to manufacture them. Reducing the cost of manufacturing would make hydrogen fuel cells more accessible. Previous research has shown that an injection-moldable blend of nylon and nickel-coated carbon fibers meets Department of Energy (DOE) standards for bipolar plate conductivity. However, the manufacturing process largely shapes the material properties of any part. Refining the injection molding process could increase the conductivity of future plates.

One factor that contributed to the conductivity of the plates is the average angle of the carbon fibers throughout the plate. Analyzing the relationship between different fiber angles, molding practices, and conductivity would allow for more conductive plates to be made. However, the average angle is very difficult to obtain and verify experimentally. Thousands of fibers through hundreds of images must be analyzed to determine the angle of the fibers, taking weeks of research time. This research develops an image processing program to reduce the analysis time to mere minutes with minimal penalty to accuracy. The program uses MATLAB Image Processing toolkit to distinguish fibers from the surrounding polymer and image artifacts to create a measure of average fiber angle accurate to previous research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thank you to my thesis supervisor, Dr. Hollinger. During our meetings and discussions, I always felt like an equal. I don't know if I would have continued to finish a thesis had you not been my advisor. I had become disinterested in research from past experiences, but our work together reminded me that failures are opportunities to learn and grow, not pointless setbacks. Thank you so much for working with me through each setback that we faced throughout the semester. I wish you and your family the best of health throughout the coming years.

Thank you to Dr. Beevers, who encouraged me throughout the process as well. Your feedback and general knowledge of software and programming techniques was very helpful throughout my time spent researching here.

Thank you to Dr. Brown, Mrs. Gummerson, and Dr. Carney. Your willingness to accommodate me and my schedule has been invaluable throughout my time at Behrend. Thank you for encouraging me to stick with the program and for accommodating my strange summer graduation into the legislature of the Schreyer Honors College. Your continual support of the Honors Program at Behrend is invaluable and undoubtedly formative to me during my time here.

Thank you to my senior design team, Deven Phillips, Morgan Tarbrake, and Zackary Vandervort. I had so much to get done this year, and I was inevitably going to fall behind and miss some deadlines at some point. Thank you for keeping me motivated, for being understanding of my situation, and for working with my schedule this year. I don't think I would have graduated if not for your help. Particularly large thanks to Zack who worked as a partner in past research projects.

Thank you to my parents and my siblings. Your support and your nagging have kept me going and working throughout the semester. I would not have made it so far in my academic career without your love and support.

Chapter 1 : BACKGROUND AND RESEARCH OBJECTIVES

Introduction to Fuel Cells and Bipolar Plates

Hydrogen fuel cells are an underused and underdeveloped source of potential energy. A hydrogen fuel cell makes use of the combustion reaction between hydrogen and oxygen to produce electricity [1]. This reaction produces heat and water as a result, as shown below.

$$2H_2 + O_2 \xrightarrow{yields} 2H_2O + \text{electricity} + \text{heat}$$

A fuel cell functions similarly to a battery. On one side, the anode, molecular hydrogen is introduced. This hydrogen contacts a catalyst — often made of platinum — and splits into two protons and two electrons [1]. The other side of the fuel cell is known as the cathode. The cathode breaks molecular oxygen into two oxygen ions. The protons, produced at the anode, diffuse through a polymer membrane to the cathode. The electrons are routed through a wire, and the electric potential can be used to power any matter of electrical device [1]. The electrons migrate to the cathode as the protons simultaneously move to the cathode as well. The electrons, protons, and oxygen ions combine to form water and release heat as a biproduct. Note that the protons diffuse across a selectively permeable membrane, so that oxygen cannot diffuse into the anode and disrupt the gradient [1].

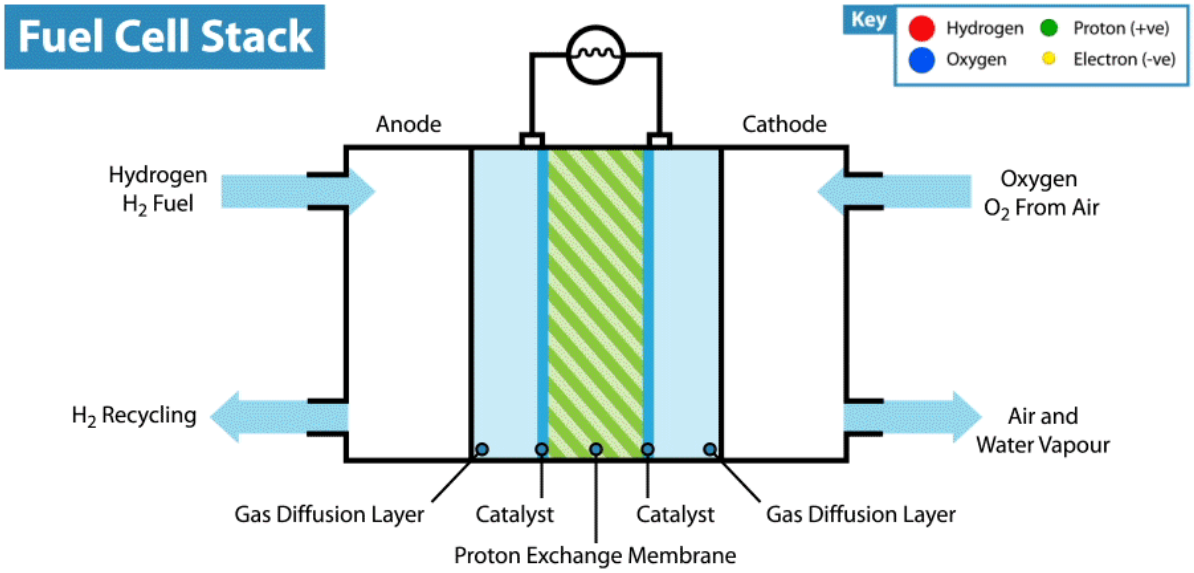


Figure 1: Hydrogen Fuel Cell Visualization [2]

Figure 1 visualizes the previously described process. The voltage produced by one fuel cell, while ultimately depending on the size of the cell, is often small, roughly 1 volt [3]. Similar to batteries, fuel cells can be combined in series to produce electric potentials necessary to power much larger devices. One the ends of each fuel cell are bipolar plates. These plates allow for fuel cells to be stacked in series or parallel. The plates must withstand the force of compression between cells and be tolerant to the biproduct heat from the combustion reaction. The plates must also be resistant to water created by the reaction, as well as low gas permeability to ensure diffusion of gases between fuel cells does not occur. As a result, bipolar plates are commonly machined from steel and graphite. Channels or grooves are machined into each plate to improve the diffusion of hydrogen and oxygen gases to their respective electrodes. Machining is laborious and significantly adds to the cost of an already expensive fuel cell.

The Need for Mass-Manufacturing

Reducing the cost of bipolar plates could allow for fuel cells to be more efficiently manufactured and more reliably used as an alternative energy source. The ability to mass-manufacture bipolar plates from cheaper material has been explored greatly in previous research. The existence of this work owes itself to previous studies exploring manufacturability, namely those carried out by the thesis advisor, Dr. Hollinger. Hollinger had developed injection-moldable bipolar plates that met a United States Department of Energy (DOE) conductivity goal of $100\ \frac{S}{cm^2}$ for bipolar plates in fuel cells [4, 5, 6]. These plates were fabricated from nylon, due to its relative ease of use with injection-molding techniques [7]. Throughout research, the use of various conductivity increasing additives was explored. Dog bone samples were produced with each new composition. The conductivity of samples at each composition was verified with a 4-point probe test, as described in Figure 2 below.

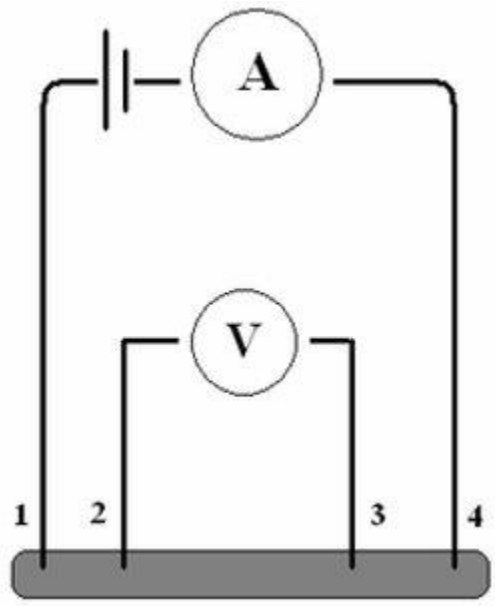


Figure 2: Diagram of 4-Point Probe Test [8]

A current source was connected to the ends of the dog bone sample, and a voltmeter was used to measure the potential from points 2 to 3. The recorded voltage and given current were used to determine the conductivity of the sample.

Verifying and Predicting Experimental Results

Experimentally identifying the conductivity was effective for the purposes of determining which additives should be used in the nylon to maximize conductivity. However, to properly verify found experimental results or predict the conductivity for a given composition, a mathematical description of the physical relationship would need to be used. Hollinger et al used the relationship shown below in Figure 3 to verify any results [9].

$$\sigma_{long} = \frac{4\phi_p d_c \ell \cos^2(\theta)}{\pi d^2 \rho_f X}$$

σ_{long} : conductivity of composite

ϕ_p : function of percolation threshold

d_c : contact diameter

ℓ : length of carbon fiber

θ : angle of fiber

d : diameter of carbon fiber

ρ_f : resistivity of carbon fiber

X : function of the number of contacts

Figure 3: Theoretical Formula for Longitudinal Conductivity

Most of the physical constants used to solve for the conductivity of the composite σ_{long} can be given by the supplier. Specifically, the resistivity ρ_f, contact diameter d_c, length ℓ, and diameter d can all be found from supplier data [10]. The function of percolation threshold ϕ_p, function of the number of contacts X, and the mean angle of the carbon fiber θ must be experimentally derived.

The mean fiber angle θ is particularly difficult to experimentally obtain and verify. To find the mean angle of each carbon fiber, a good first step is to neatly slice a dog bone sample in half and obtain a cross-sectional image of the composite. The carbon fibers have an average diameter on the scale of micrometers, so a scanning electron microscope (SEM) must be used to obtain detailed images of each cross section. One example of such an image is shown below in Figure 4:

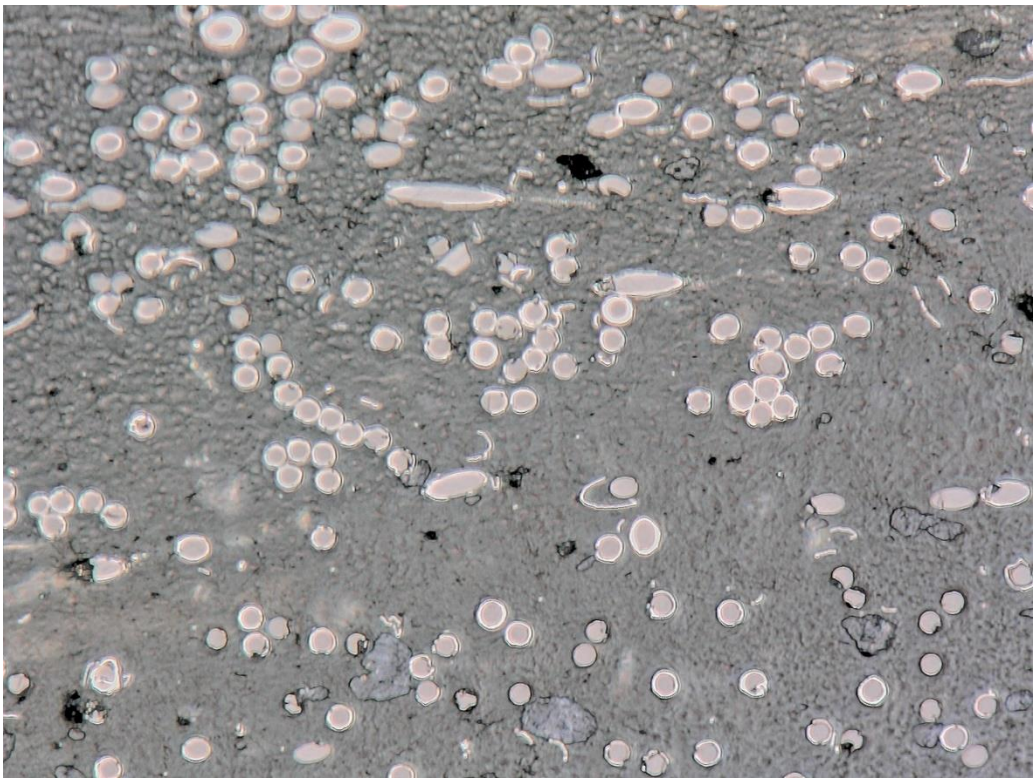


Figure 4: Example SEM Cross-Sectional Image

Each light gray oval in the image represents a carbon fiber. The darker gray area surrounding the ovals is nylon, and black areas indicate voids or air pockets. Metrics from each oval can be taken to find the orientation of the carbon fiber in the nylon. Circles and ellipses are conic sections. A perfectly circular fiber indicates that the fiber is normal to the cross-sectional area. An elliptical fiber indicates that the fiber is angled relative to the cross-section. Visualizing the circle and ellipse by their definitions as conic sections can be helpful and is shown in Figure 5.

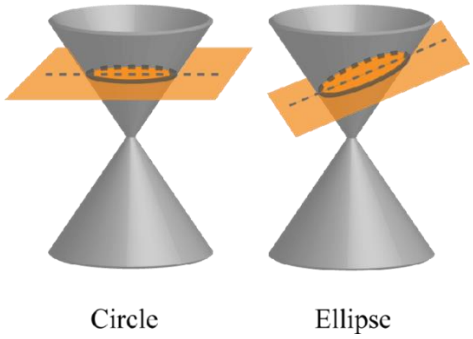


Figure 5: Conic Sections [11]

A circle is created by bisecting the cone with a reference plane that is parallel to the base of the cone. An ellipse is created by bisecting the cone with an angled reference plane. In the case of the SEM image shown above, the reference plane is always perfectly flat. In this case, the angle of the cone, or the angle of the fiber, is what changes. Whichever is chosen as a reference plane or as the cone does not matter; the angle will be the same. Hollinger et al chose the flat cross-section as the reference plane, and then used the formula shown below in Equation 1 to find the angle of the carbon fiber throughout the nylon.

Equation 1: Angle of Fiber Throughout Composite

$$\theta_f = \arccos\left(\frac{b}{a}\right)$$

In this formula, θ_f is the angle of the fiber with respect to the reference plane. b is the length of the minor axis, and a is the length of the major axis. This formula is a version of the plane angle formula taught in multivariable calculus courses at Penn State.

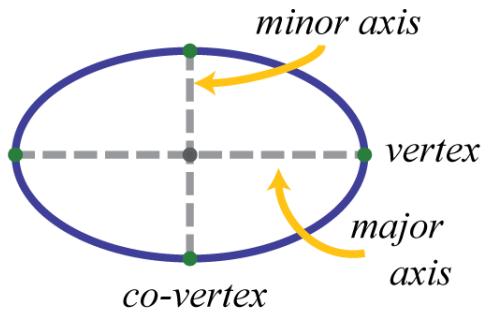


Figure 6: Major and Minor Axes of an Ellipse [12]

In previous work, Hollinger et al had captured hundreds of scanning electron microscopy (SEM) images of dog bone cross-sections. Approximately 500 images were captured from dog bone samples from three different versions of the nylon composite. The composites were distinguished by the weight percentage of nickel-coated carbon fibers that were mixed into the nylon. There were 20%, 30%, and 40% weight carbon fiber nylon samples. From each set of roughly 500 images, approximately 30 images that best reflected the distribution of fibers throughout the composite were chosen. Each of these 30 images were reviewed twice. First, each fiber on the image was reviewed for validity as a fiber. Some light grey spots in the image shown above may look like fibers at first, but under closer inspection, may just be image artifacts, fragments of nickel, or fibers that are too deformed to be properly measured. After identifying each fiber, the major and minor axes of each fiber were measured with MATLAB software. These values were then used to compute the angle of the fiber. After all the images were analyzed, the mean angle for each image and the population mean angle were calculated. Even by only reviewing

5% of the total data, thousands of fibers would need to be measured and reviewed. This process is time-consuming; 3 student researchers needed two 40-hour work weeks to complete the study.

In the past decade, image processing software tools have become far more robust. Image processing and identification are used to identify cancerous growths, prevent unsavory individuals from unlocking an iPhone, and analyze the fertility conditions of tracts of land gathered from satellite images. Many of these processes are entirely automated and taken for granted. This thesis explores the image processing tools available to mechanical engineers and lays the foundation for developing a fast, precise tool for automatically determining the population mean angle for all images within a weight percent data set.

Specifically, this thesis details the process of building an image processing program that was:

1. Capable of accurately analyzing the mean population angle of a data set.
    a. Instead of analyzing only thirty images, the program was to be able to analyze all 500 images and match the mean angle determined by the manual study.
2. Faster than the current manual method.
    a. Technically, anything that analyzed the images in less than two weeks would be an improvement. A goal of one hour of total processing time was set.

Additional goals were set, given that the primary goals were met.

1. A relationship between carbon fiber weight, mold shape, and injection-molding technique could be derived from the data and used to optimize conductivity and

cost efficiency. This was a very steep and flexible goal, given that the relationship between these variables could be several different theses on their own.

Chapter 2 :
EXPLORATION OF THE SOLUTION SPACE AND INTRODUCTION TO IMAGE PROCESSING WITH MATLAB

Choosing the Right Tool for the Job

There exists a plethora of software tools that can be employed for image processing. Among these, Python's OpenCV and machine learning libraries are standard [13]. Python is a robust language that is, compared to other languages, easy to learn and use. I had learned Python in high school and used it throughout my numerical analysis courses at Penn State. Despite my understanding of Python, I chose to disregard Python when searching for image processing tools. Beyond the goal of producing a successful program, I wanted whatever code I wrote to be understandable and easily modified by students who would continue my research. Python is not taught to mechanical engineers at Penn State Behrend, so using Python in my program would pose a significant challenge to future mechanical engineers. MATLAB, however, is taught to mechanical engineers at Behrend. Additionally, MATLAB has image processing and machine learning toolkits with similar capabilities to Python [14]. These toolkits have in-depth coursework and documentation for engineers learning how to use the tools.

A Brief Note on Image Processing

Image processing is a wide and experimental field. For any given image, there are numerous programming strategies that could be employed to achieve the same desired effect. Some strategies may be more efficient or accurate than others, but finding the optimal strategies or combination of strategies is very difficult. There are general rules of thumb that can be used to

determine when one strategy may work better than another, but the exact effectiveness of the method is difficult to quantify. Often, the best way to see if an image processing strategy is effective is to directly compare modified images to the original images. Throughout research, considerable time was spent comparing one image to the next to verify that use of each image processing strategy and to ensure that the program was correctly interpreting data. A summary of the available image processing techniques and those that were eventually selected for use in the program are outlined throughout.

MATLAB Image Processing Toolkit

Image processing techniques can be divided into two main groups: pre-processing and post-processing [14]. Pre-processing techniques are primarily used to improve the accuracy and efficiency of post-processing techniques. Pre-processing techniques remove image noise, enhance contrast, or otherwise improve the quality of the image [14]. Many post-processing techniques assume a certain level of quality that can sometimes only be achieved by first using pre-processing techniques. Without pre-processing techniques, post-processing techniques can fall victim to the "garbage in, garbage out" phenomenon often discussed in computer science. If the image data is initially flawed from a lack of or improper use of pre-processing techniques, the results can be totally erroneous. The efficacy of pre-processing techniques can vary from image to image; one set of criteria may work excellently for one image and poorly for another. With 500 images in a data set, some data is inevitably ignored to produce valid results.

Post-processing techniques are very diverse. Post-processing techniques are often used to segment images or break them down into like components that can be measured. Post-processing techniques segment primarily by color, texture, and shape. However, post-processing techniques

can also rotate images, apply machine learning models to image data, or analyze a variety of other image trends [14].

Throughout the research period, a wide variety of different pre-processing and post-processing techniques were explored. Some techniques were used only once before their shortcomings were made obvious. Other techniques were used intermittently throughout development. Several pre-processing and post-processing techniques were eventually used together at the end of development in April. Each of the various techniques used throughout the various iterations of the software are described in the following chapter with a full breakdown of the current image processing workflow.

Storing Image Data in MATLAB

MATLAB recognizes all variables as matrices [15]. For any image data that is loaded into MATLAB, a matrix is created where every pixel is represented by an entry in the matrix. For example, an HD image is a 720x1280 array of pixels and is stored as a 720 row by 1280 column matrix. This is only partially true, however, as one matrix is not always enough to store all the data in one image. Images can be categorized by the number of matrices needed to represent them and by the color space of the image. Table 1below provides a brief introduction to image types. Note that this table is not comprehensive.

Table 1: Description of Various Image Types [16]

Image Type	Description
Truecolor (RGB)	Image data is stored as an $m \times n \times 3$ matrix, where m is the number of rows, and n is the

	number of columns. 3 is the number of color channels in the image; there is one matrix for red data, one for green data, and one for blue data. Each entry in the matrix commonly scales between 0 and 255, where 0 is the absence of color and 255 is full intensity.
Multispectral and Hyperspectral	Image data is stored as an $m \times n \times c$ matrix, where m is the number of rows, n is the number of columns, and c is the number of color channels in the image. c is often greater than 3.
Grayscale (Intensity) Images	Image data is stored in an $m \times n$ matrix, where each entry is between 0 and 255. 0 represents black, while 255 represents white. All numbers in-between are shades of gray.
Binary Images	Image data is stored in an $m \times n$ matrix. Each entry is 0 or 1. 0 represents black, and 1 represents white. There are no shades of gray. White objects represent objects in the foreground, whereas black pixels make up the background. White foreground objects are known as “connected components.”

These image types will be referred to throughout the following chapters, particularly binary and grayscale images. The scale of each image is also assumed to operate on a uint8 basis, rather than a single, double, or uint16 basis.

Chapter 3 : IMAGE PROCESSING WORKFLOW

Program Workflow

The program, at any stage in development, can be divided into four distinct parts as described below:

1. Importing Data
    a. The program begins by creating a datastore object in MATLAB. MATLAB recognizes a user-specified folder that contains image data. MATLAB imports one image from this folder and initializes it into working memory.
2. Pre-Processing
    a. The program applies several different pre-processing techniques to the image. The pre-processing techniques eliminate excess noise and artifacts from the image. The pre-processing also converts the SEM image from a hyperspectral image to an RGB image, the RGB image to a grayscale image, and the grayscale image to a binary image.
3. Post-Processing
    a. The program applies several different post-processing techniques to the binary image. The program analyzes each grouping of white pixels as a connected component. Connected components are filtered by size, shape, and other metrics to ensure the program does not mistake any artifacts that surpassed pre-processing as fibers during post-processing. The remaining connected components are understood to be carbon fibers. The program

plots the binary image as a MATLAB figure, labeling each fiber with its major axis, minor axis, and angle. The total number of fibers is recorded, and the angle of each fiber is added to a running total.

4. Saving Data
    a. The MATLAB figure is saved as a .JPEG file. Saved JPEGS can be compared to the initial grayscale image to verify that the program is properly interpreting image data. The angle of each fiber is saved to a .csv file. The .csv file can be imported into tools like Microsoft Excel for continued analysis.

Chapter 4 : IMPORTANT TERMS, INITIAL CODE, AND IMAGE PROCESSING CHALLENGES

Developing Image Processing Code

Research began in the summer of 2022. Research started with an initial understanding of the fuel cells and prior research in the field of composite bipolar plates, as outlined in Chapter 1. A significant portion of the research period was spent becoming familiar with image processing and the various image processing capabilities of the MATLAB toolkit. By the end of the research period, one .PNG image could be interpreted by the program for one run of the program. The program could pre-process the PNG, post-process the PNG, display a MATLAB plot with various fiber metrics, and determine the mean angle of the fibers within the PNG. Unfortunately, the program could not yet save MATLAB plots or process more than one image. Despite these shortcomings, the first program was largely considered to be a success. Certain elements of the pre-processing and the post-processing from the first program continued to be effective in future iterations of the program.

```
s4 = imread("S4_9_02.png"); % import image
s4 = s4(:,:,1:3);          % extract only channels 1,2,3
S4gs = im2gray(s4);        % convert to grayscale image
s4adj = imadjust(s4gs);    % adjust for contrast
```

Figure 7: Code for Multispectral to Grayscale Adjusted Image

The first line of code in Figure 7 is responsible for transforming the image saved in the local directory into a matrix. The SEM images are hyperspectral images and have five channels. The second line of code removes channels four and five, keeping only channels 1, 2, and 3. The image is effectively converted from a hyperspectral image to an RGB image, as these channels represent color data for red, green, and blue. The third line of code converts the RGB image into

a grayscale image. The im2gray() function converts the RGB image to a grayscale image by forming a weighted sum of the red, green, and blue components [17]. The fourth line of code adjusts the grayscale image for contrast. This step was eliminated in future versions of the program, as the added contrast produced more issues during post-processing than resolved. Figure 8 shows the changes between certain lines of code. Although these changes are not visually apparent, changing the data type of each image is necessary for further image processing.

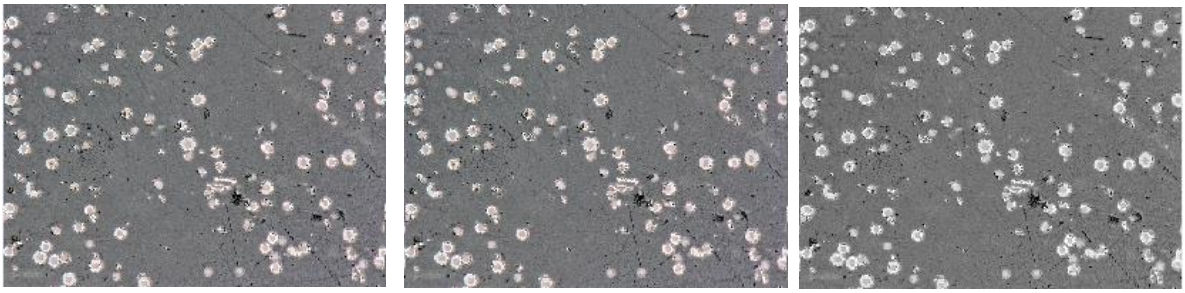


Figure 8: Hyperspectral Image, RGB Image, Grayscale Image (Left to Right)

```
s4smooth = wiener2(s4adj, 5); % remove noise
```

Figure 9: Code for Wiener Adaptive Noise Filtering

Continuing, the next line of code shown in Figure 9 employs an imaging filter known as a wiener filter over the image [18]. The wiener filter is an adaptive filter that estimates the local mean and variance around each pixel within a given neighborhood or surrounding pixels [18]. The wiener filter performs little smoothing where variance between pixel values is large and more smoothing when variance between pixel values is small [18]. This method increases contrast throughout an image depending on the surrounding pixels, meaning the method is effective at preserving edges between fibers and ensuring that separate fibers near one another are not counted as one large fiber. In this case, the filter is using a 5×5 square centered over a given pixel as the neighborhood for the pixel. Depending on the variance within the 5x5 square surrounding the

pixel, the filter will adjust the pixel to increase the overall variance and sharpen edges. Figure 10 shows the effects of Wiener filtering.

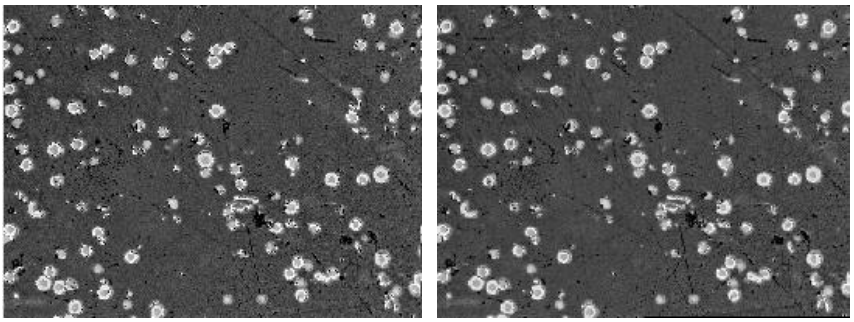


Figure 10: Contrast Adjusted Image, Wiener Filtered Image (Left to Right)

```
S4bw = imbinarize(S4gs);          % convert to binary image
```

Figure 11: Code for Creating a Binary Image

The next line of code shown in Figure 11 transforms the modified grayscale image into a binary image. imbinarize() uses Otsu's method to create a binary image [19]. To form a binary image, a threshold value between 0 and 255 must be selected so that every entry less than the selected value is overwritten to have a value of zero, and every entry greater than the selected value is overwritten to have a value of 1. Entries equal to 0 are black, and entries equal to 1 are white. Otsu's method tests every value between 0 and 255, calculating the total variance between pixels at each threshold [19]. The threshold that produces the maximum variance between pixels is chosen to form the binary image. Preserving variance equates to preserving detail as data is summarized into a binary image. An example of a binary image is shown below in Figure 12.

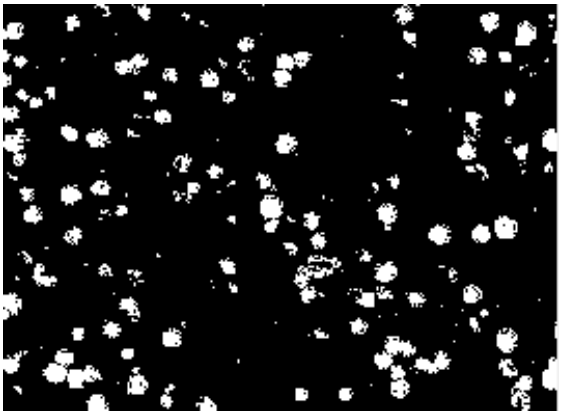


Figure 12: Binary Image

Forming binary images discards a significant portion of the image data. Although some detail is lost, considerable efficiency is gained. There are many image processing toolkit functions that require binary images to be effective. Further, computations are simpler, and less data is used overall when the image data can be summarized as Booleans instead of integers.

```
SE = strel("disk", 5);    % filter for opening image
s4open = imopen(s4bw, SE);  % open image, connect dark areas and remove small bright areas
s4close = imclose(s4open, SE);
```

Figure 13: Code for Opening, Closing Image

The final four lines shown in Figure 13 in the pre-processing segment modify the binary image. The first line creates a morphological structuring element. A morphological structuring element is another way of defining the neighborhood of pixels that a filter or morphological operation will consider when modifying a pixel in an image [20]. A disk-shaped structure is visualized below:

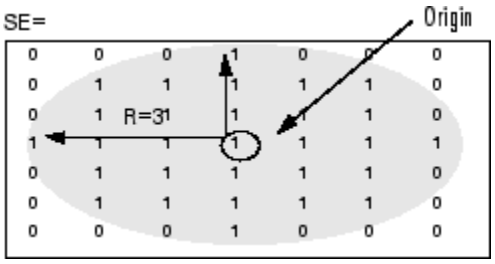


Figure 14: Example Disk Structuring Element for Morphological Operations [20]

The command imopen() opens the image. “Opening” an image refers to the process of rounding foreground objects (white pixels) by erosion and dilation [21]. Eroding an image shrinks white spaces, while dilating grows them [22, 23]. Both processes together have the effect of removing small image artifacts and smoothing edges of fibers so fibers that are very close to one another are analyzed as two separate fibers, not as one. The next line imclose() closes the image. Closing is another morphological operation and can be understood as the inverse process of opening; the image is dilated, then eroded [24]. Consequently, closing an image decreases the size of black spaces between fibers. Opening then immediately closing the image seems counterintuitive. Remember that each image filtering process can be understood as simplifying or removing information from an image. Even though opening and closing are inverse processes, enough data is removed each time that the opened image is different enough from the original binary image so that the closing the opened image produces a different image than before. Figure 15 helps to illustrate these differences. Neighborhood sizes and the order in which neighborhood sizes are opened or closed were explored throughout the other research periods.

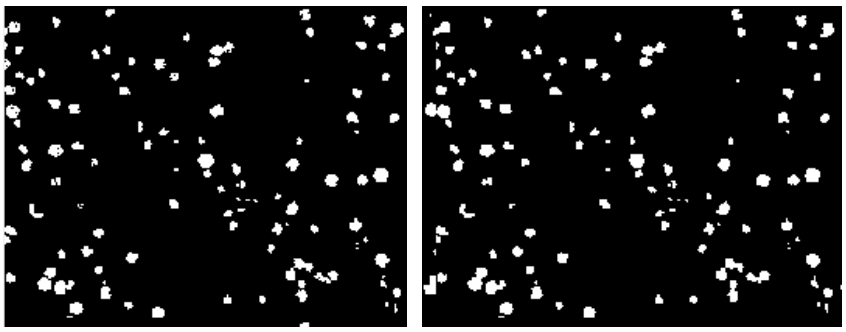


Figure 15: Opened Image, Closed Image (Left to Right)

```
S1bw = imfill(s4close, "holes");      % fill holes, may be necessary
```

Figure 16: Code for Filling Holes in Fibers

The next line of code shown in Figure 16 is designed to fill holes in the image. Holes are defined as clusters of background pixels, or black pixels, that do not connect to the edge of the

image [25]. Holes are clusters of black pixels that are truly surrounded by white pixels [25]. Filling these holes eliminates any holes inside fibers that were created as a result of image artifacts or other proceeding operations. Filling holes makes the binary image look neater and decreases the chance of errors in analysis throughout the process. Figure 17 helps illustrate the effects of filling holes.

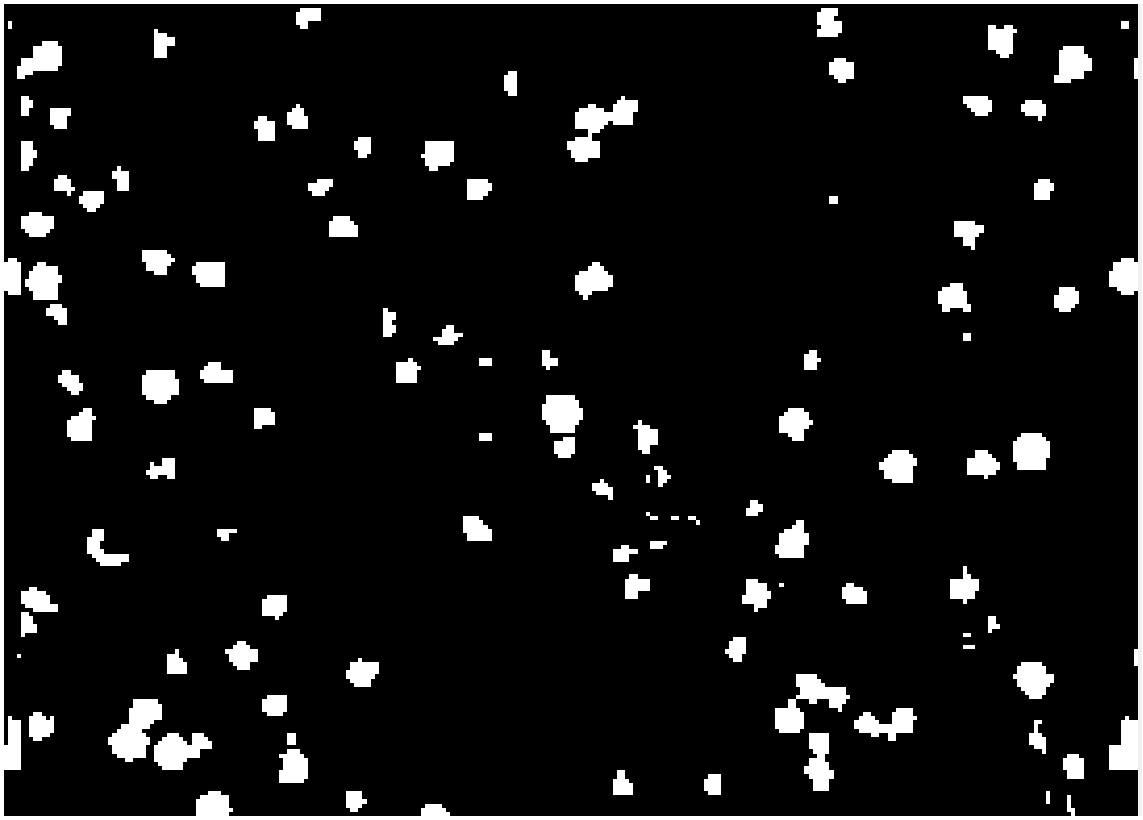


Figure 17: Binary Image with Filled Holes

```
ellipses = bwpropfilt(Slbw, "Area", [1000 10000]); % only show objects within this pixel area
```

Figure 18: Code for Removing Large and Small Artifacts

The line shown in Figure 18 marks the first line of post-processing. The command identifies foreground objects, or fibers, as connected components and eliminates components of a certain pixel area. Some images, like the one in Figure 19, had large artifacts, voids, streaky fibers, or clustered fibers that could not be distinguished from one another.

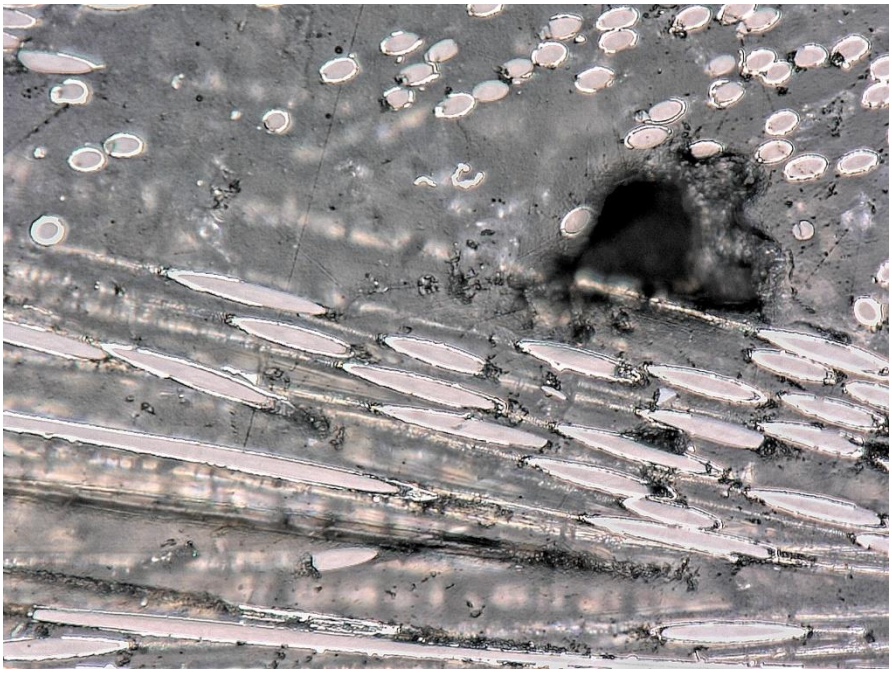


Figure 19: Long, Streaky Fibers Sometimes Cannot be Measured or are Outliers

The command bwpropfilt() allows for each connected component to be analyzed and filtered for several metrics [26]. The area constraint has a lower bound, meaning elements that are too small are also ignored by the filter [26]. The true impact and capabilities of bwpropfilt() are analyzed throughout development. Figure 20 shows an example of the remaining data that would be analyzed by bwpropfilt().

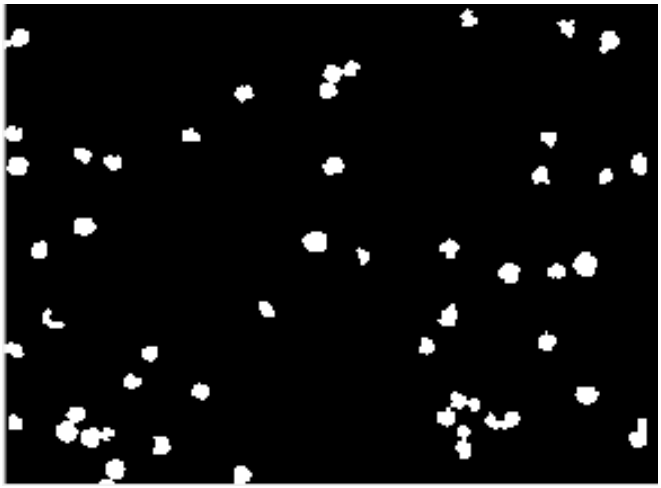


Figure 20: Final Binary Image to Be Analyzed in Post-Processing

All the remaining connected components can be summarized as a dictionary object in MATLAB.

```
props = regionprops(ellipses, 'Area', 'Centroid', 'MajorAxisLength', 'MinorAxisLength');
allAreas = [props.Area];
majorAxisLength = [props.MajorAxisLength];
minorAxisLength = [props.MinorAxisLength];
sum = 0;
```

Figure 21: Dictionary and Property Arrays

For each connected component, there are several properties that the regionprops() command can identify [27]. In Figure 21, the area, the x and y coordinates in pixels (where $x = 0$ at the bottom left corner and reaches its maximum value in the lower right corner, $y = 0$ at the top left corner and reaches maximum value in the bottom left corner), the major axis length, and minor axis length of each connected component are obtained and stored in the props dictionary [27]. These properties can be extracted into arrays as the following lines show. The final line creates sum which is later used to store the sum of all the fiber angles.

```
% display major / minor axis on each image
figure
imshow(ellipses);
axis('on', 'image');
impixelinfo;
hold on;
centroids = vertcat(props.Centroid);

for i = 1:length(props)
    x = props(i).Centroid(1);
    y = props(i).Centroid(2);
    a = props(i).MajorAxisLength;
    b = props(i).MinorAxisLength;
    plot(x, y, 'r+', 'LineWidth', 2, 'MarkerSize', 15);
    str = sprintf('        (%.1f, %.1f)', a, b);
    text(x, y, str, 'Color', 'r', 'FontSize', 6, 'FontWeight', 'bold');
end

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);

for i = 1:numel(majorAxisLength)
    angle = acosd(minorAxisLength(i) / majorAxisLength(i));
    sum = sum + angle;
end

numel(majorAxisLength)
meanAngle = sum / numel(majorAxisLength)
```

Figure 22: Plotting, Measuring Fiber Angle, Reporting Mean Fiber Angle

Figure 22 shows how a plot of the binary image is made. A "+" symbol is plotted at the centroid of the fiber. Next to the symbol are the major and minor axes of the fiber. After creating and saving the image, the angle of each fiber and the average fiber angle for the image is created. For this image, there were 44 identified fibers with an average angle of 37°. The full version of this code can be found in Appendix A: Code from Summer Development.

Key Image Processing Errors

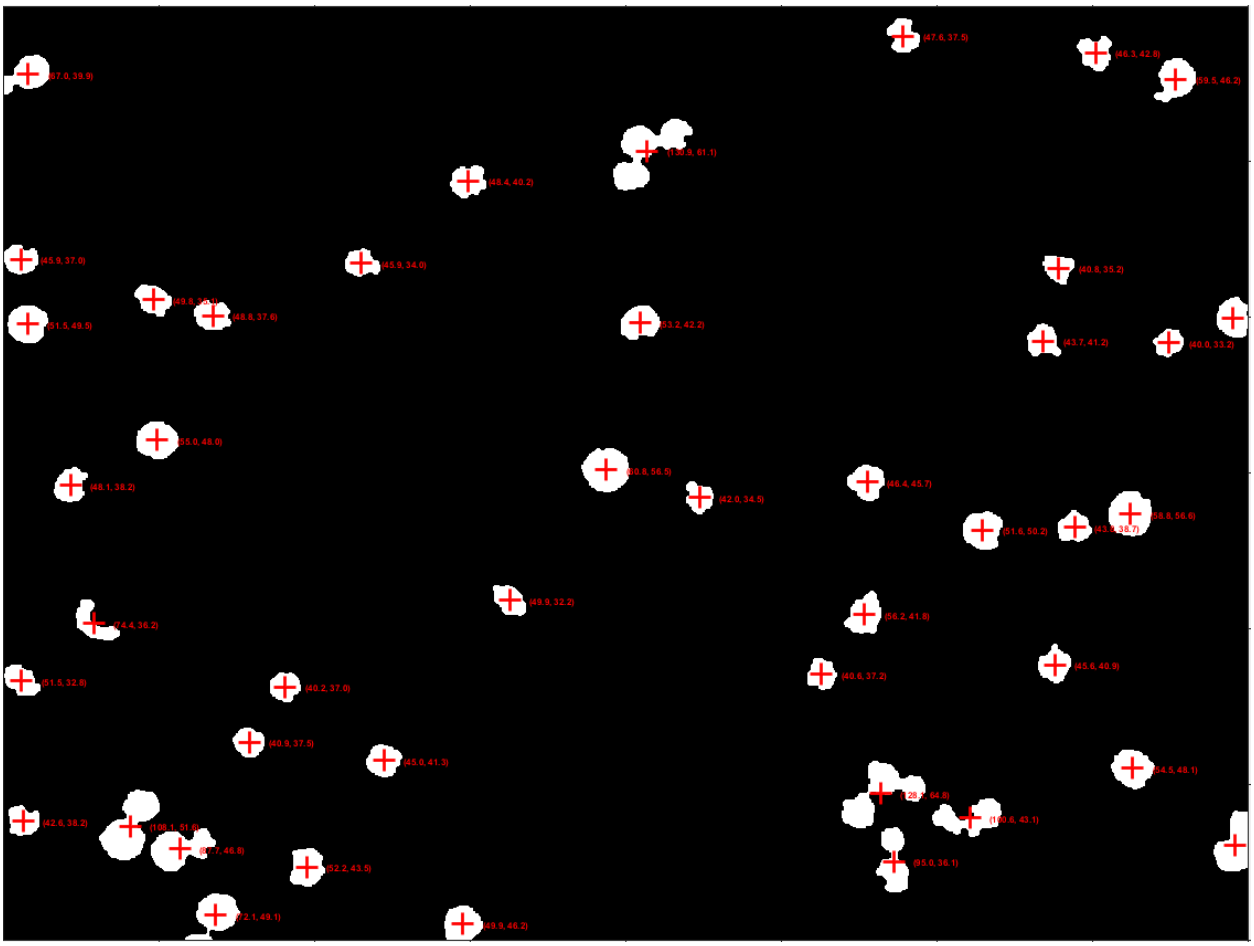


Figure 23: Final Image

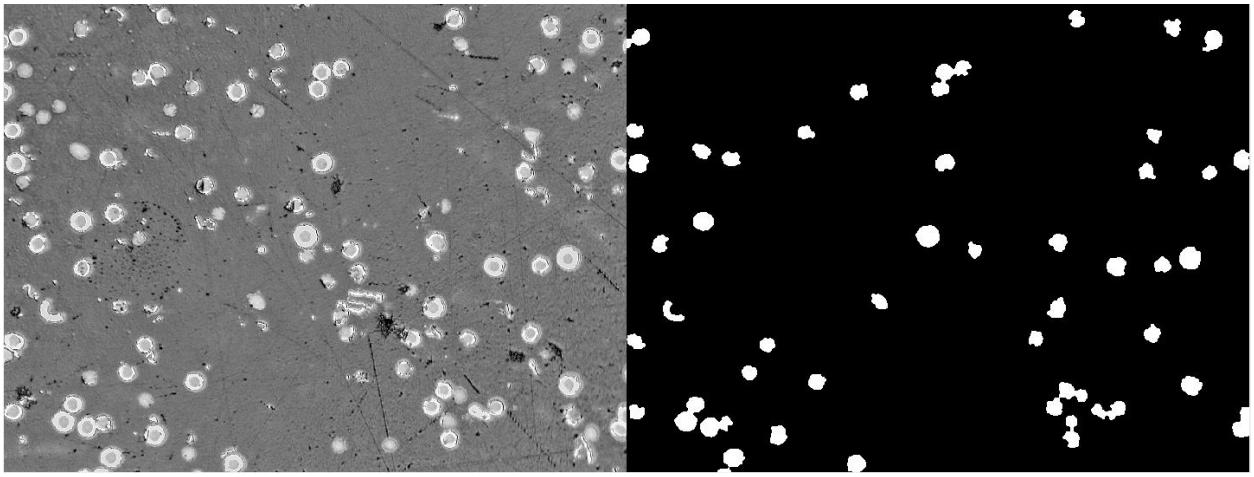


Figure 24: Grayscale Image and Final Binary Image

Figure 23 shows the final image with the centroid and axis data. Figure 24 compares the grayscale image to the final binary image. Both figures show important data that explains changes made to the program and further study described throughout. The binary image in Figure 24 generally shows that the program does an excellent job of removing artifacts, voids, and other errors in the image. Unfortunately, the program also seems to remove some slightly darker fibers as well. Both figures can be used to show how some fibers that are distinct yet very close to one another are evaluated as one large fiber. Note the locations of the centroids in Figure 23, namely in the lower right, lower left, and upper center parts of the image, and compare them to their corresponding clusters of fibers in the left hand side of Figure 24. When the fibers are separate, all of these fibers are relatively straight. Clumped together, these fibers are measured as very eccentric, or very slanted, ellipses, as in Figure 25. The program recognizing fewer ellipses at more dramatic angles generally skews results and portrays the average angle as much higher than what was originally discovered in the manual study.

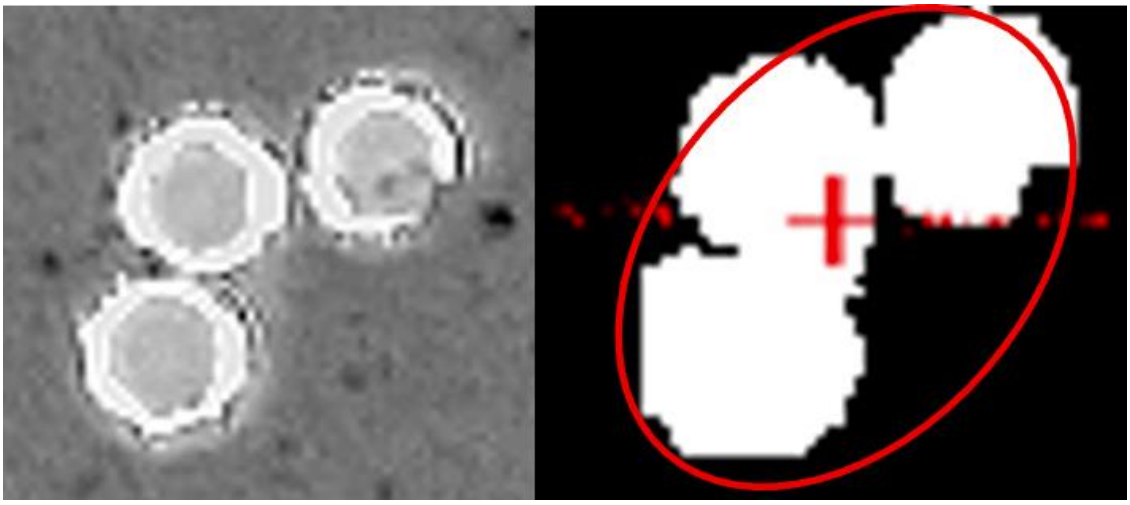


Figure 25: Several Straight, Clustered Fibers are Interpreted as One, Eccentric Fiber

The tendency for the program to cluster fibers also proved to be highly detrimental in the instance of clustered fibers, as shown in Figure 26.

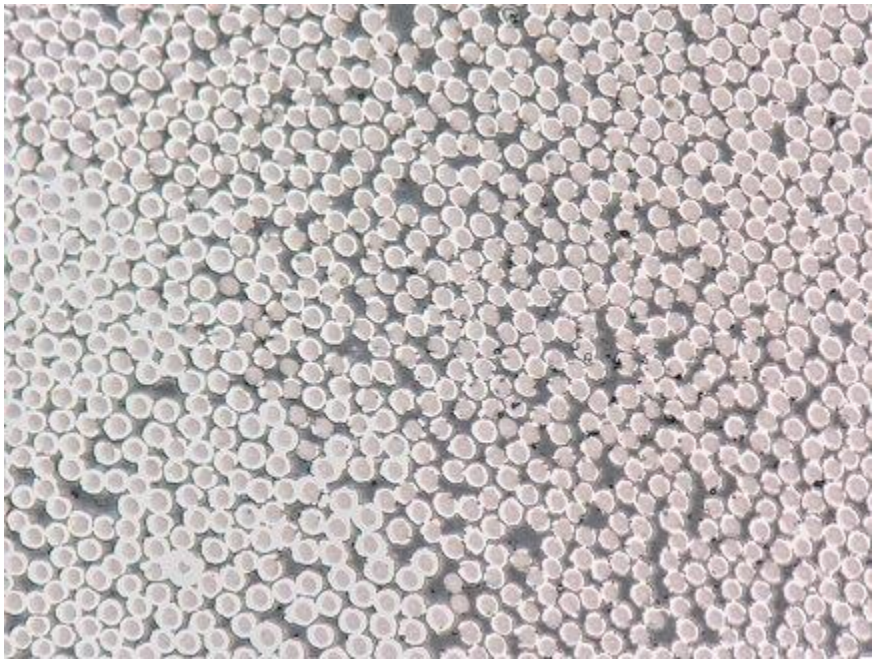


Figure 26: Clustered Fibers

If the same image processing method described throughout was applied to Figure 26, the resulting image would be recognized as one massive artifact or several massive artifacts and much of the data would be ignored by the software. An example of this is shown in Figure 27. From left to right and starting on the left again after the first three images, the image is transformed from a grayscale to binary image, opened, closed, filled in, and edited by bwpropfilt().

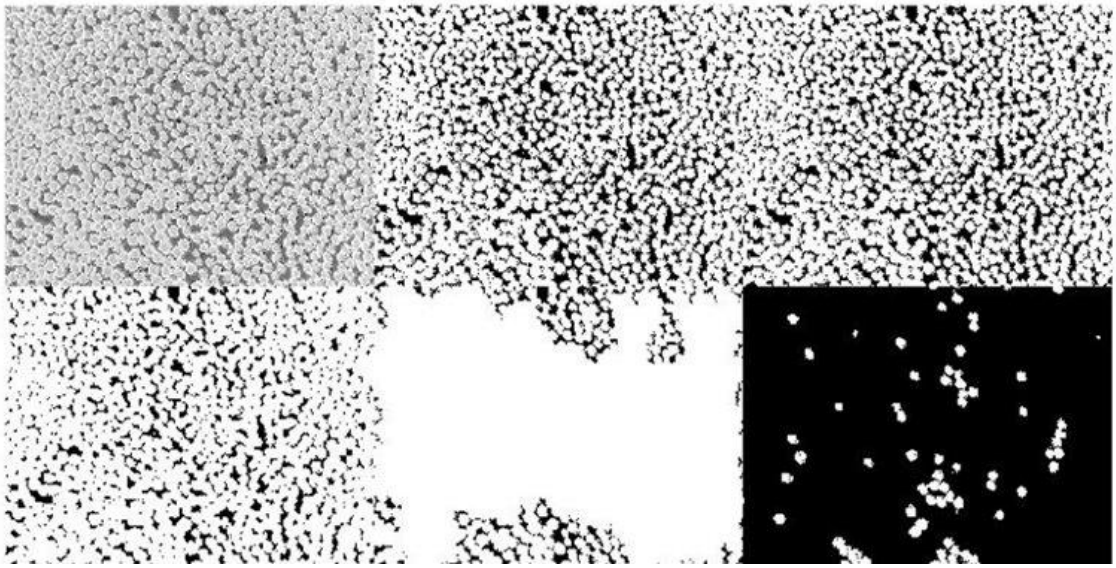


Figure 27: Processing Clustered Images

The sensitivity of the thresholding technique also leads to some error. This error can be observed in some of the clustered or overlapping fibers shown in Figure 24. Some artifacts are too big to be eliminated by opening or closing the image but are not so dark that they are merged into the foreground as part of the binary imaging. The threshold will not be perfect for each image, as some data is always lost in creating a binary image. However, the exact technique that would preserve the most data was unknown. A clearer illustration is shown in Figure 28.

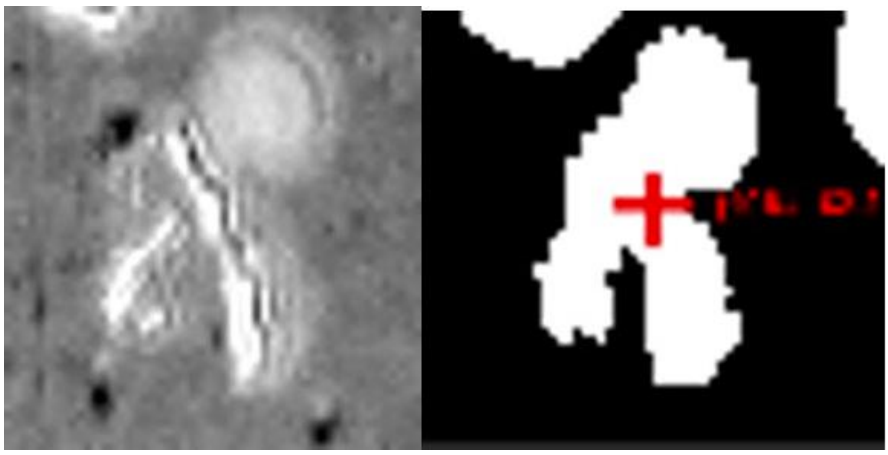


Figure 28: Thresholding Error

A critical part of developing the code was verifying results and adjusting the pre-processing commands to ensure that as much of the original data is preserved as possible. The size of structural elements was determined by repeatedly examining the effect different size elements had on image. The same applied for the type of commands used and the order commands were used in. The question of how much each command impacted the final image was asked multiple times throughout research. Discussion throughout the fall semester found that the easiest way to quantify the impact of a certain method was to test the code on the images that were analyzed in the statistical study and compare the number of fibers in the final image to the number of fibers counted in the original image. With the number of variables that have been identified throughout the discussion of the code thus far, running the code on each of the 3 sets of 30 images to try and understand the exact impact of each variable would be an endless chore. The comparison technique was only used in certain situations and is described further in work performed in the spring semester.

Chapter 5 : ANALYZING MULTIPLE IMAGES AND CALIBRATING CODE

Analyzing the Entire Image Population

The summer semester developed an understanding of the tools available for image processing and some of the capabilities of each tool. Work in the fall semester continued to explore MATLAB's image processing capabilities while also seeking a way to validate and improve the accuracy of the program. While there was an understanding of how each processing method would impact the data, the true size of the errors had not been quantified. The program had only analyzed images one at a time throughout the summer, so the true breadth of the data had not been explored. Small errors in the test images used in the summer could be large errors in others that had not been tested with the code. To eventually improve the program, the validity of the methods used over the summer needed to be tested on the entire data set. With an understanding of those errors, the summer methods could be refined over the spring into a working product.

Much of the code from the summer semester was reused throughout the fall. The same image processing techniques outlined above were studied for accuracy throughout the fall, so they could not be changed. Other minor modifications were made, however. The program was first modified to allow for batch processing of images, as shown in Figure 29 and Figure 30.

```
ds = imageDatastore("C:\Users\maxth\OneDrive\Desktop\20wt\20wtSpecial"); % create datastore object we will loop through
dataFilenames = ds.Files;
numImages = numel(dataFilenames)                % find number of images in folder, will use in for loop
```

Figure 29: Datastore Object for Multiple Images

In the first line of Figure 29, the datastore object is created. The datastore object is used for collections of data that are too large to fit within memory. The datastore object allows MATLAB to individually retrieve images from a hard drive or other like storage system available to the computer.

```
for i = 1:numImages

    [filepath, name, ext] = fileparts(dataFilenames(i));

    % start with reading an image from the datastore and do all general
    % preprocessing

    img = readimage(ds, i);
```

Figure 30: Reading Each Image with the Datastore

Images can be retrieved individually with the imread() function [28]. The size of the datastore is known, as shown in the third line of Figure 29. Therefore, a for loop like that shown in Figure 30 can be used to iterate through each image in each directory. After the img object is created, the same processing developed throughout the summer occurs. The mean fiber angle for each image is recorded, and the next image is processed. The process repeats itself until all images are processed. The mean of means is finally calculated and reported as the average fiber angle. This calculation uses the central limit theorem, assuming the fiber angles follow a normal distribution.

A Brief Discussion of Efficiency

Reading all the images through a datastore object could slow the program. Repeatedly interfacing with a hard drive or remote drive is almost always slower than retrieving information from RAM. Pre-loading all images into working memory would allow for much faster processing.

Table 2: Memory Requirements for Photos

Weight Percent	Number of Images	Memory Requirement (GB)
20	536	2.93
30	480	2.63

40	533	2.92

Table 2 shows the memory requirements for loading any given collection of images into MATLAB. 2.93 gigabytes is a considerable amount of data, however loading all images into RAM is a possible on the given hardware. Unfortunately, attempts at increasing the speed of the program by loading all the images into RAM were ultimately fruitless. The readall() command uses a datastore to create a celled array in working memory that contains all the images as RGB images; these images are read with the imread() function [29]. The readimage() command requires a datastore and an integer value as arguments to determine which image from which folder should be read; each image is individually retrieved from the hard drive or remote data source [30]. The readall() approach should be faster. However, the readimage() process was ultimately found to be the quicker of the two approaches. When used to analyze the 40% image set, readimage() approach completed in 11 minutes, and the readall() approach completed in 12.5 minutes. Both approaches produced the same images and average fiber angle. Perhaps a larger difference would be found between the two approaches if the data were stored on a remote drive or if a larger quantity of data was read. Unfortunately, code efficiency was a much smaller concern than validating and refining results. These methods were the only ones examined throughout the semester for decreasing the load time of any images. Examining this process as part of future research may prove to be very useful, especially if more data is used in the future.

Storing and Tracking Program Data

The name of the image, the number of fibers in each image, and the average fiber angle were written to a .csv file during runtime. Saving the data this way made comparing program results to manual study results simple. The 20 wt% NiCF data from the manual study can be visualized as shown in Figure 31 and Figure 32.

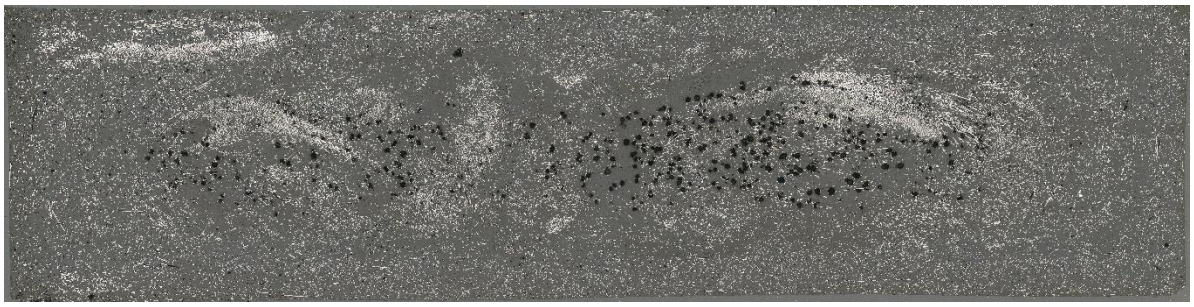


Figure 31: Cross-Section of Dog Bone Sample

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66
88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109
175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160	159	158	157	156	155	154
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197
263	262	261	260	259	258	257	256	255	254	253	252	251	250	249	248	247	246	245	244	243	242
264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285
351	350	349	348	347	346	345	344	343	342	341	340	339	338	337	336	335	334	333	332	331	330
352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373
439	438	437	436	435	434	433	432	431	430	429	428	427	426	425	424	423	422	421	420	419	418
440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461
527	526	525	524	523	522	521	520	519	518	517	516	515	514	513	512	511	510	509	508	507	506
528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549
615	614	613	612	611	610	609	608	607	606	605	604	603	602	601	600	599	598	597	596	595	594
616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637

Figure 32: Manual Study Data Selection

The cross section of the dog bone sample was divided into over 600 different SEM images. The images are numbered from left to right, until the end of the row where the proceeding row is numbered right to left. The yellow squares in Figure 32 represent the images that were selected for analysis in the manual study. The scattering of images across the cross-section shown in Figure 32

was deemed to be the best selection to represent the data by Hollinger et al. The data collected by Hollinger et al was illustrated in several ways, one of which was a fiber angle heat map.

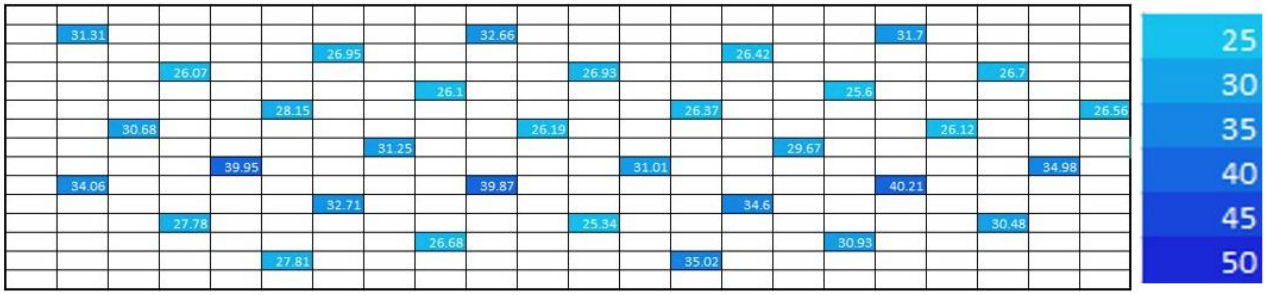


Figure 33: 20 Weight % NiCF, Average Fiber Angle, Manual Study

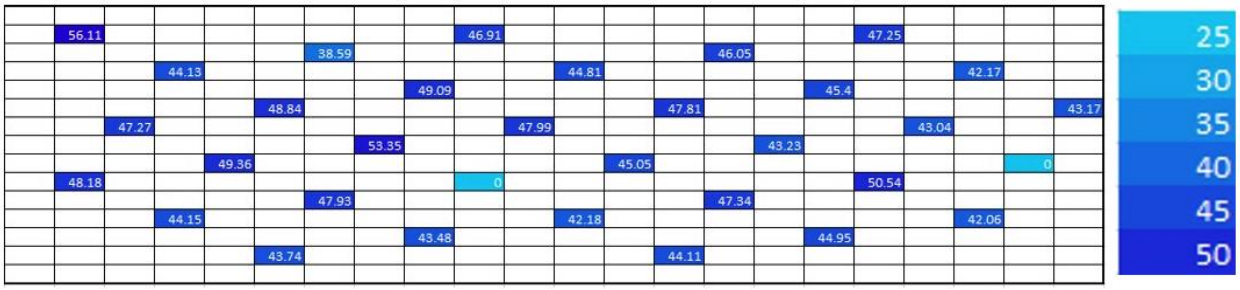


Figure 34: 20 Weight % NiCF, Average Fiber Angle, Program Results

Figure 33 is a fiber angle heat map of the 20 wt% NiCF samples as determined by the manual study performed by Hollinger et al. Figure 34 is a recreation of the fiber angle heat map where the mean fiber angle shown in each cell was determined by the image processing program developed over the summer. Both Figure 33 and Figure 34 share the same scale. Generally, the main sources of error (thresholding, overlapping fibers, tightly clustered images) caused the mean angle to appear inflated. Note the two light blue cells in Figure 34. At first glance, the light blue seems indicative of accuracy. However, the mean angle of both cells is zero. The mean angle is zero because both images were so tightly clustered that the program detected zero fibers and reported a mean angle of zero to indicate the error. Heat maps showing the absolute difference and

percent difference between the manual study results and the program results were made to illustrate the gravity of this error and provide further insight.

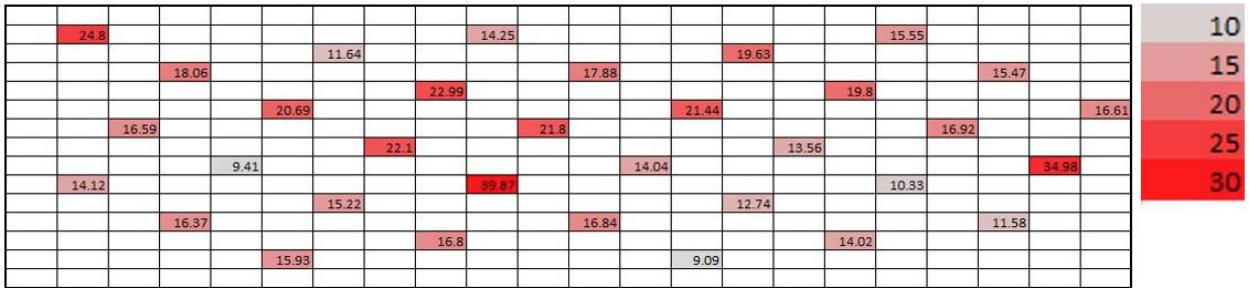


Figure 35: 20 Weight %, Absolute Difference Between Manual Study and Program Results

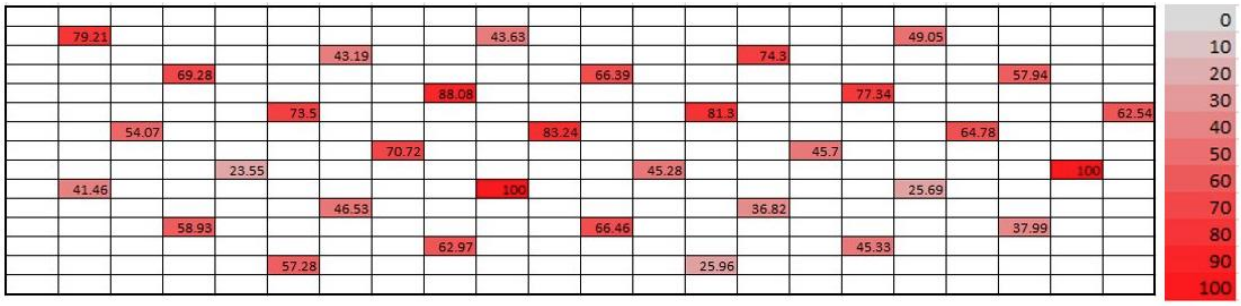


Figure 36: 20 Weight %, Percent Difference Between Manual Study and Program Results

Figure 35 and Figure 36 corroborate an insight promoted by Figure 33 and Figure 34. The current level of error is unacceptable, and results are inaccurate. The percent errors shown in Figure 36 are rather significant. However, such extremes in data are not necessarily an indication of poor image processing. The data was likely skewed by significant outliers, in the form of overlapping fibers or tightly clustered images where many straight fibers are ignored. Simply ignoring overlapping fibers or other outliers that are inevitably produced by applying a general image processing workflow to a broad set of data could significantly reduce the error. Fine-tuning the thresholding process to create a finer binary image or modifying the process to tackle clustered images would be more difficult processes that eventually spanned the entire spring and following summer development periods.

The heat map and error heat map were also constructed for the 30 and 40 wt% NiCF dog bone samples. The insights from these graphs vary little from the insights generated by the above figures. These graphs are included in Appendix B: Additional Heat Maps.

Hollinger et al. also produced histograms of the data. Each histogram is comprised of several thousand data points of fiber angles which were measured manually. Figure 37 is a histogram that shows the frequency of each fiber angle found throughout the sample images used in the manual study of the 20 wt% NiCF cross section by Hollinger et al. Figure 38 shows the program results for the frequency of the mean angles for each image within the population of 20 wt% images. Note that the summer development period found the mean angle for each photo, not across all photos. The differences between Figure 37 and Figure 38 mean that few valid conclusions can be drawn from comparing the two plots. These figures are included mostly for the purpose of showing the development of ideas throughout the project. The main benefit of the histogram created by Hollinger et al. was that the plot showed the spread of the population. A mean summarizes the entire spread of data with one number. Showing the spread of means ultimately shows a reduced population spread, which is counterintuitive to creating a histogram. Ultimately, the two graphs show two different data spreads. The trends in Figure 37 and Figure 38 are followed closely in the 30 and 40 wt% samples as well.

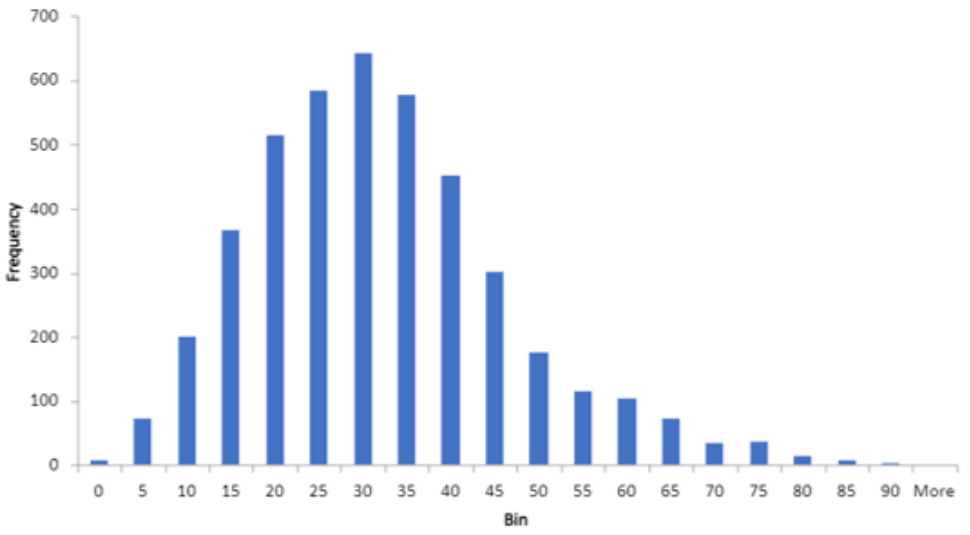


Figure 37: Angle Distribution from Sample Images of 20 wt% NiCF Cross Section

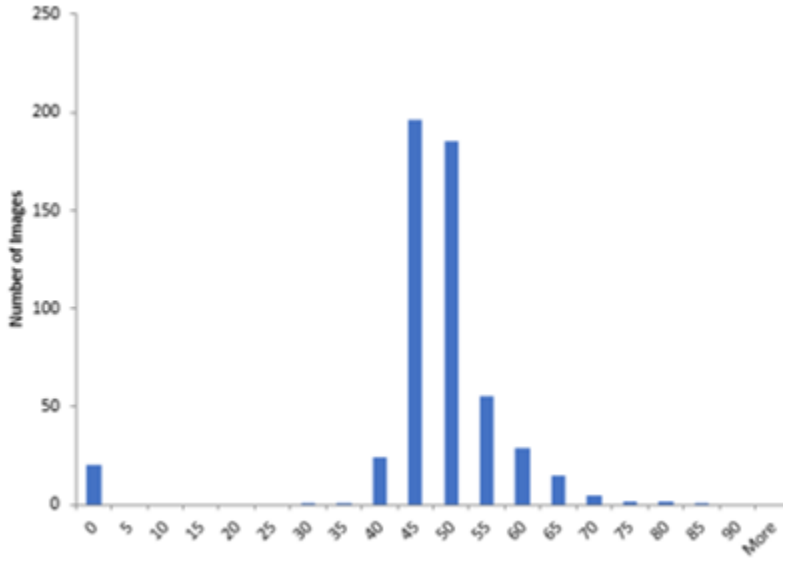


Figure 38: Mean Photo Angle Distribution from Population of Images of 20 wt% NiCF Cross Section

The central limit theorem states that the means of sufficiently large random samples (with replacement) will be roughly normally distributed regardless of trends in the original population [31]. The sample size, the number of images used to approximate the overall angle, is above 30 images in the 20 and 40 wt% NiCF groups; the sample size is large enough for the central limit theorem to safely apply to those groups, but the same cannot be said for the 30wt% group [32].

According to Hollinger, the images were also selected in a sufficiently random method. The distribution in Figure 38 is roughly normal, but notice the shift in the angle of the fibers. The histogram seems to corroborate the overestimation trend shown in the heat maps. However, the cause of the shift is still unclear. Further, the method that should be applied to revert or stop the shift is also unclear. The program could be consistently merging fibers together to create many fibers at an angle of 40 or 50 degrees, or the program could be merging fewer fibers together but ultimately creating higher-angle fibers. More generally, the shift could be caused by a misinterpretation of data or by a few impressive outliers (or perhaps both). Using the mean of the mean image angles was easier to program at the time, but a mean of means is arguably a poor representation of the mean angle due to the conditions of the central limit theorem and the comparisons that would be drawn between the program and the previous study. Given the capabilities of modern computing and the population size, computing the population mean angle is fully possible and was naturally the next step in development.

Chapter 6 : CORRECTING THE MEAN WITH DATA OMISSION TECHNIQUES

Continuation of Comparing Program Data to Manual Study Data

The program used throughout the summer and fall was modified to track the angle of the fibers across the entire population, not to track the mean angle across each image. Listed below are histograms comparing the program analysis of the entire population to the manual analysis of the sample population. The comparison of the samples and the population is valid. The study is statistically valid as previously discussed, and it is the only data that exists for comparison. However, analyzing the entire population with the program, instead of just analyzing the same images used in the study, seems tedious. The entire population was analyzed with the hopes of obtaining as much information as possible surrounding the shortcomings of the program. With a larger amount of data, trends are easier to identify. With 500 images, faults in image analysis would be more obvious faster.

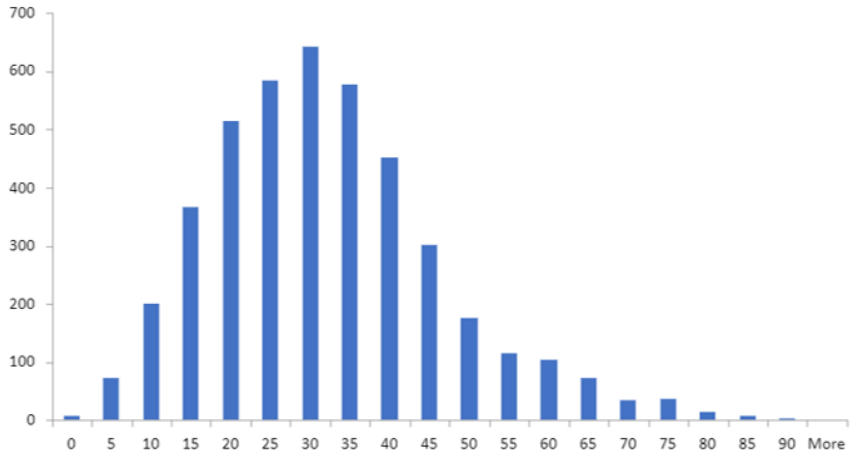


Figure 39: Angle Distribution from Sample Images of 20 wt% NiCF Cross Section

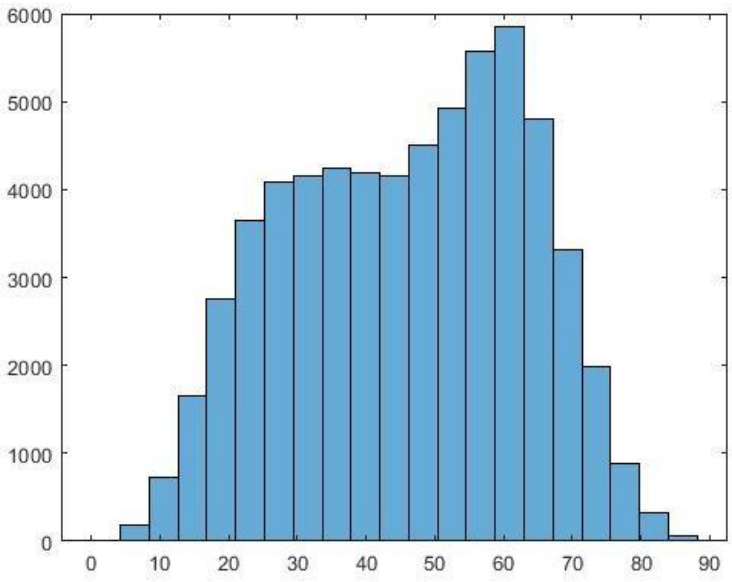


Figure 40: Population Angle Distribution from 20 wt% NiCF Cross Section

Figure 39 is identical to Figure 37 but is included above for convenience. Comparing the two figures, Figure 40 has considerably more fibers. Further, the histogram is saddle-shaped, with peaks near 30 degrees and 60 degrees. There are considerably more fibers between 30 and 50 degrees than in Figure 39. Further, the number of fibers above 50 degrees is an obvious deviation from the data showed in Figure 39. The program is not generating a few outliers, rather it overestimates the angle of fibers.

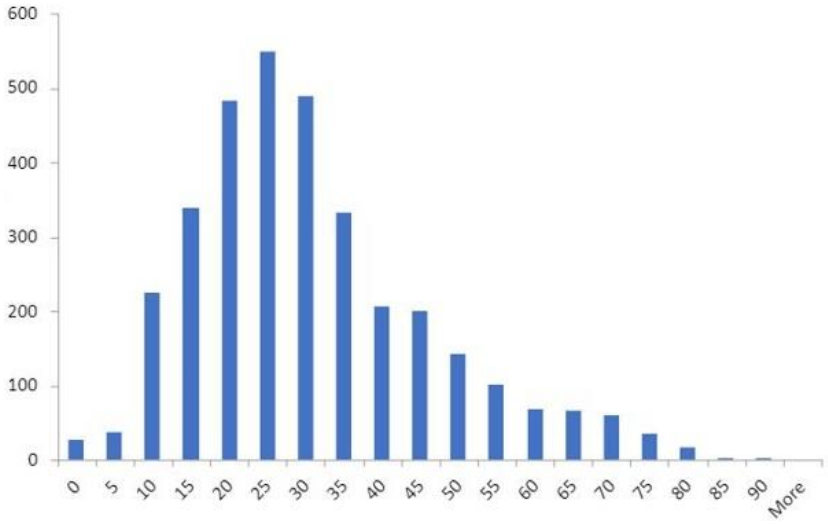


Figure 41: Angle Distribution from Sample Images of 30 wt% NiCF Cross Section

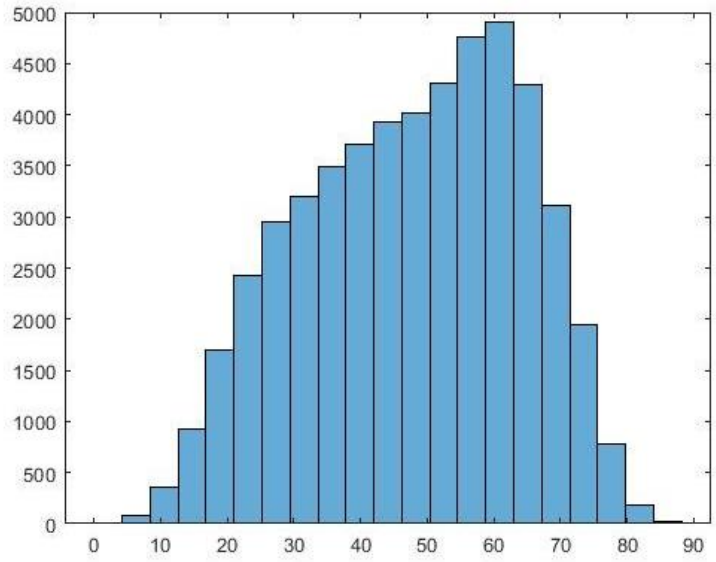


Figure 42: Population Angle Distribution from 30 wt% NiCF Cross Section

Figure 41 and Figure 42 are very similar to Figure 39 and Figure 40, the only difference is that Figure 41 and Figure 42 represent the data for the 30 wt% cross section. The overestimation is much more apparent in Figure 42 than in Figure 40. There is only one peak at 60 degrees, and generally there are more eccentric fibers.

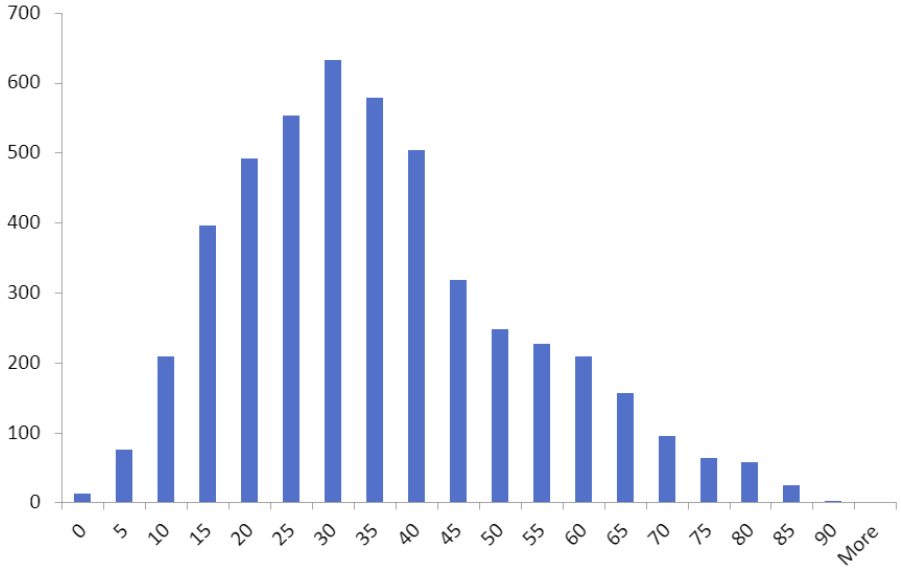


Figure 43: Angle Distribution from Sample Images of 40 wt% NiCF Cross Section

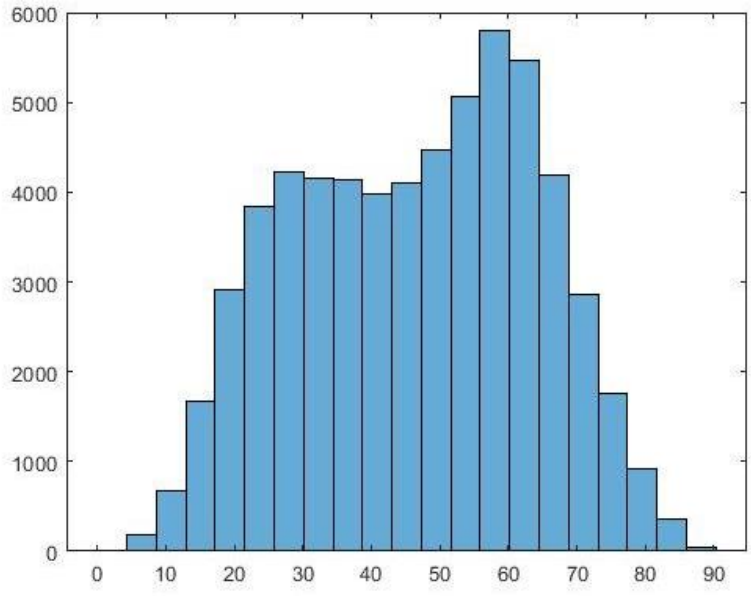


Figure 44: Population Angle Distribution from 40 wt% NiCF Cross Section

Figure 43 and Figure 44 show the results from the study and the program, respectively. Figure 44 is like Figure 40 with a saddle shape and peaks near 30 and 60 degrees. After studying the graphs and comparing the images from each data set to the respective program-generated images, the same errors with thresholding and clustered fibers seemed to be the cause of the inflated mean angle. Discussion found that there were two main ways to combat overestimation. Data points that

are multiple clustered fibers or artifacts that persist through image processing could be ignored in the final calculation. Alternatively, the image processing techniques could be refined to prevent errant data from entering the image.

Both techniques have different strengths and weaknesses. Ignoring data is a simple process, as outlined by the bwpropfilt() method shown in Figure 18. The command bwpropfilt() can filter objects by major axis length, minor axis length, area, area circumscribed by a convex polygon, percentage of area within the convex polygon filled by foreground components, perimeter, orientation, and many other properties [26]. Further, the command creates another binary image after removing connected components that do not match the criteria [26]. The regionprops() method shown in Figure 21 can be used to identify the same properties that bwpropfilt() uses to filter connected components [26, 27]. The two methods can be used in conjunction with one another to quickly identify the consequences of any data elimination. A downside to eliminating errant data is that some pertinent data is removed as well. Another downside is that the amount of pertinent data that is removed is difficult to gauge. In the manual study by Hollinger et al., the number of fibers measured in each sample image was recorded. However, the recorded fibers were not marked on the image. Designating which objects the program should recognize as fibers can be difficult. Further, knowing how much data is being ignored is also difficult for any images outside the sample images used in the initial study. The program can be compared to the sample image data, or it can be compared to the previous program iteration, so long as the previous iteration is understood to be an accurate or desirable inclusion of data.

Enhancing the image processing to create a binary image that accurately reflects the original image is another method that can combat overestimation. A more accurate binary image is almost always a benefit. With a more accurate image, less pertinent data is ignored. The new

binary image will never be perfect; some data will always be lost in making a binary image. But, the loss in data allows for use of more robust and faster image processing tools. Regardless, refined image processing techniques allow for more accurate binary images, which in turn indicate which properties should be used in bwpropfilt() to allow for precise removal of errant data. The downside to refining the image processing workflow is that the process is very difficult. There are many different image processing tools in the MATLAB toolkit. The best way to truly know if a method creates a more accurate binary image is to save and view the image with each change in methodology. This process is, naturally, very tedious. Further, certain methods may work excellently for some images but poorly for others. Additionally, the program cannot recognize errors in data as simply as the human eye can. The desire to omit or include a feature needs to be translated into a specific command or series of commands. Ultimately, there are infinitely many ways to refine the binary image, but only a handful will be effective.

Omitting Data by Major Axis Length

Omitting data is much easier than refining the image processing technique. Achieving proper results by omission of errant data alone would be ideal, considering data omission is already necessary to remove errant data that is inevitably created or recognized as part of binary imaging. To start, connected components were eliminated by the size of the major axis. The reasoning was that large connected components made of several different individual fibers would be eliminated by such restrictions.

Table 3: Mean Angle Across Carbon Fiber Dog Bones

	40 wt %	30 wt %	20 wt %
Manually Estimated Average Angle (deg)	32.9	31.8	30.2
Program Estimated Average Angle (deg)	47.4300	46.9675	45.3030

Table 3 compares the mean angle from each sample according to the study by Hollinger et al. compared to the mean angle determined from the newest rendition of the program from the fall semester. With each modification made to the major axis criteria in the program, several histograms were generated and Table 3 was revised.

At first, connected components with major axis lengths of over 250 pixels were removed from the data. For context, each pixel is about 0.125 microns. 250 pixels was chosen as a starting point after examining images and noting the lengths of each fiber as indicated by the regionprops() method. Ideally, 250 pixels would remove large outliers while still retaining as much pertinent data as possible. Additionally, fibers with a major axis below a lower bound of 30 pixels were also eliminated with the intention of eliminating artifacts or objects that were too small to be considered fibers.

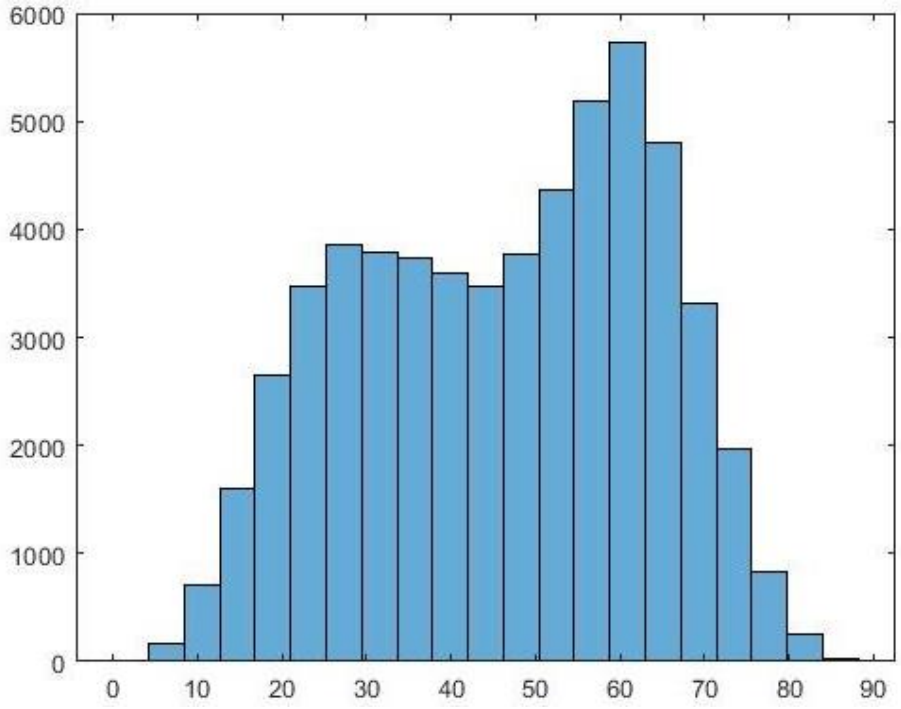


Figure 45: 20 wt% Distribution, 250 Pixel Max and 30 Pixel Min Suppression

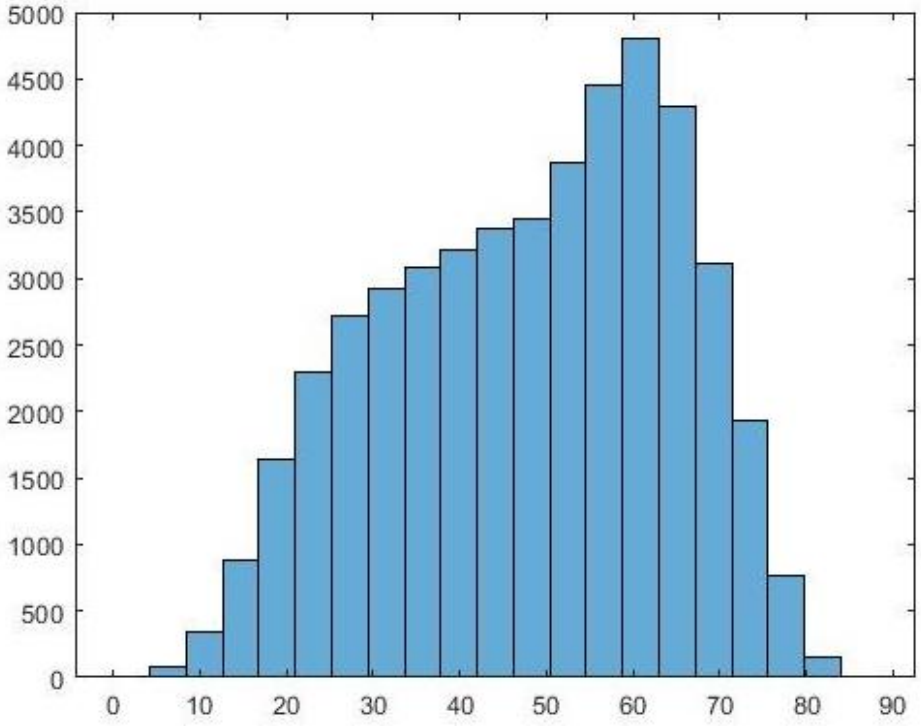


Figure 46: 30 wt% Distribution, 250 Pixel Max and 30 Pixel Min Suppression

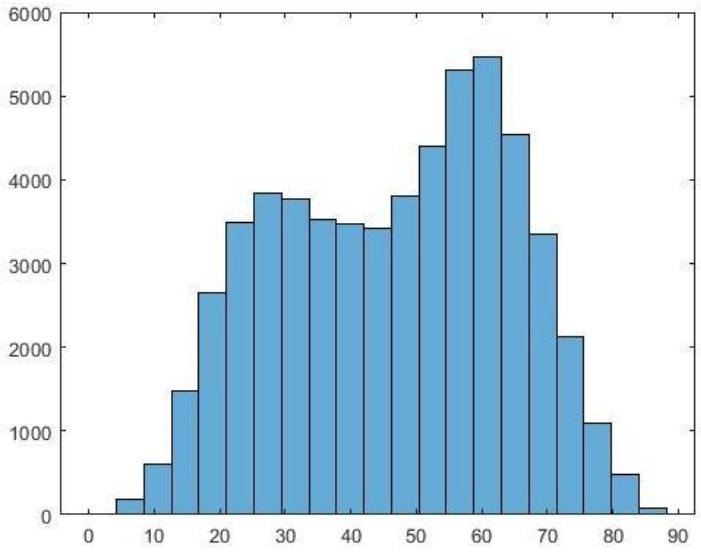


Figure 47: 40 wt% Distribution, 250 Pixel Max and 30 Pixel Min Suppression

Table 4: Mean Angle, 250 Pixel Max and 30 Pixel Min Suppression

	40 wt %	30 wt %	20 wt %
Manually Estimated Average Angle (deg)	32.9	31.8	30.2
Program Estimated Average Angle (deg)	47.8634	47.2984	45.5750

Figure 45, Figure 46, and Figure 47 all show minor improvements to the general shape of the histogram. Generally, the peaks are slightly more accentuated. There are slightly fewer fibers within the 40, 50, or 60-degree buckets. However, the means shown in Table 4 have slightly increased, perhaps from the elimination of many more small, circular artifacts outweighing the elimination of a few large elliptical fibers. Between all the newly saved binary images and the binary images formed before major and minor axis suppression was introduced, there are few notable differences.

Moving forward, a 150-pixel maximum for the major axis length was introduced. 250 pixels in length was too conservative. 150 was chosen after consulting the original binary images. Images like Figure 23 were generated with the major axis length listed on each fiber. Multiple

fibers across multiple images were reviewed to obtain this number. 150 pixels deletion, across 10 images, preserved roughly 75% of the components identified by the program. The 30-pixel minimum still applies as well.

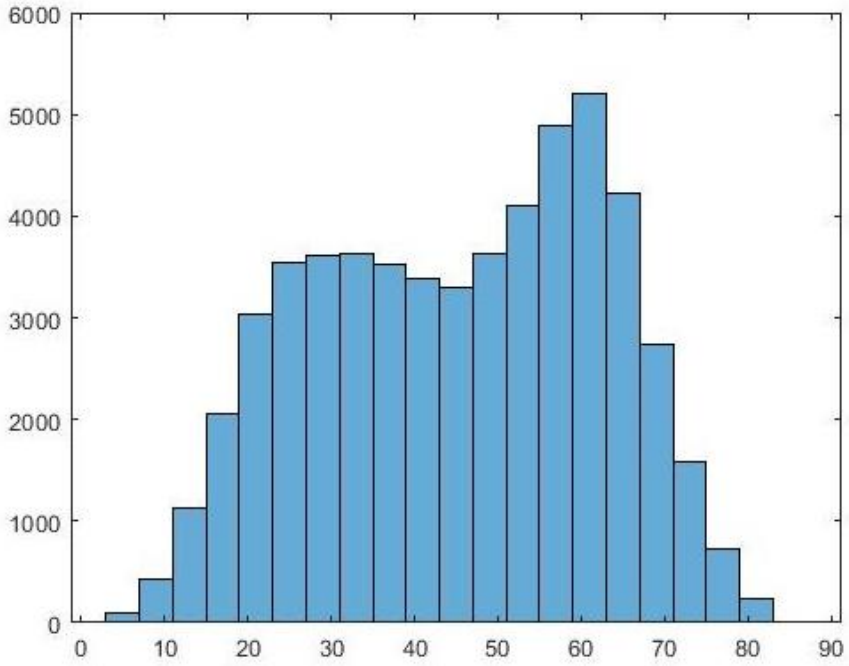


Figure 48: 20 wt% Distribution, 150 Pixel Max and 30 Pixel Min Suppression

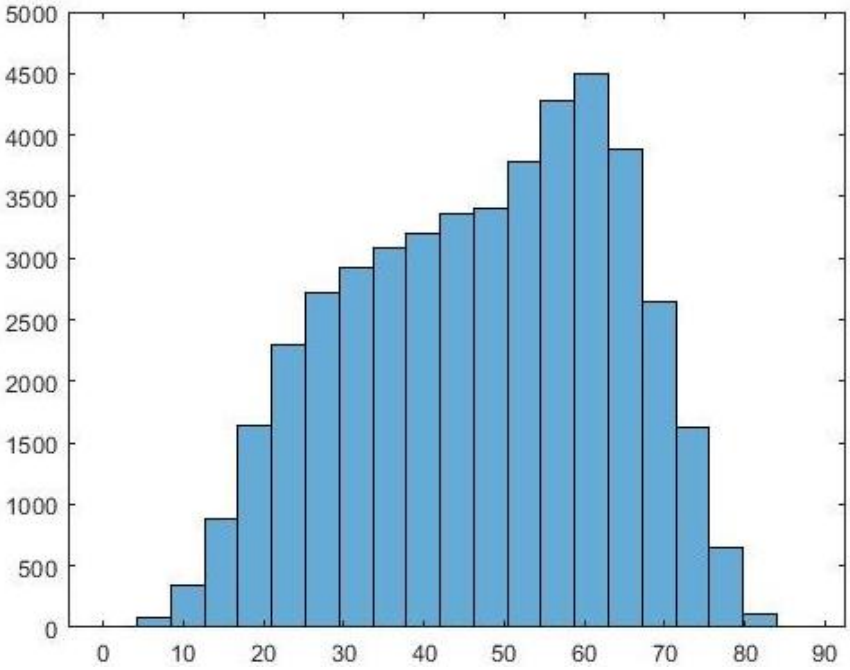


Figure 49: 30 wt% Distribution, 150 Pixel Max and 30 Pixel Min Suppression

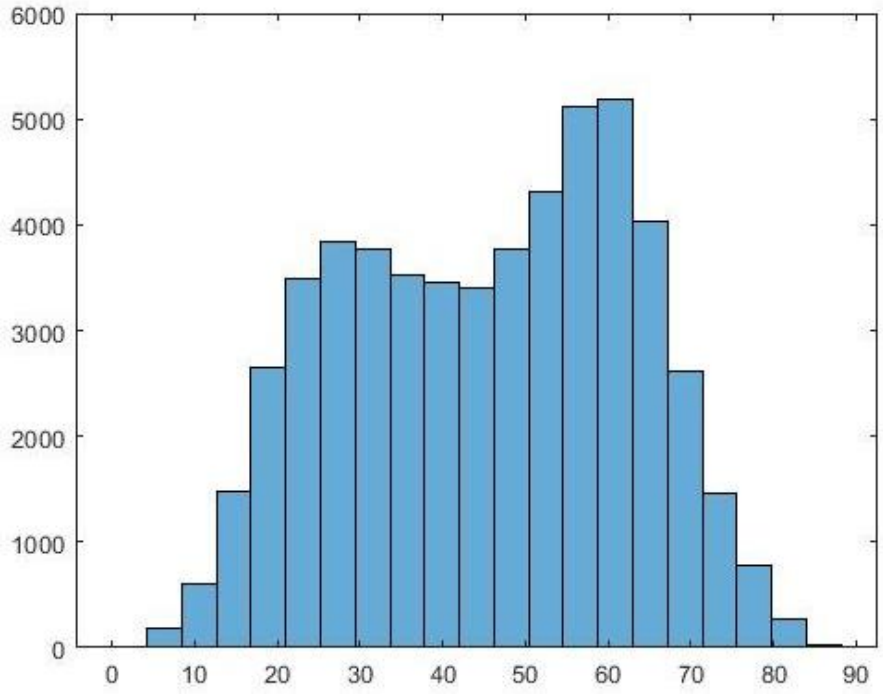


Figure 50: 40 wt% Distribution, 150 Pixel Max and 30 Pixel Min Suppression

Table 5: Mean Angle, 150 Pixel Suppression

	40 wt %	30 wt %	20 wt %
Manually Estimated Average Angle (deg)	32.9	31.8	30.2
Program Estimated Average Angle (deg)	46.7980	46.3750	44.8534

The plots shown in Figure 50, Figure 49, and Figure 48 still maintain much of the original shape seen in the previous histograms. The peaks are slightly more accentuated, however, comparison between the new and old binary images still shows little difference. The averages in Table 5 are slightly less than those shown in Table 4. Continuing the trend of suppressing smaller and smaller connected components will eliminate some pertinent data. Suppressing more data may provide additional insight about the peaks of the histogram. If the peak at 60 degrees is not eliminated by suppressing more data, a change in strategy is required.

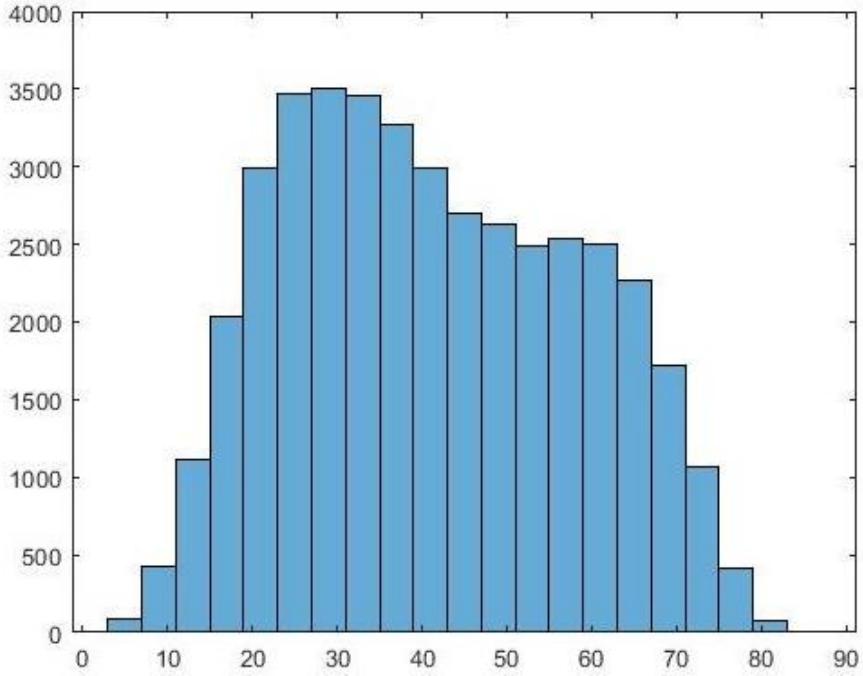


Figure 51: 20 wt% Distribution, 75 Pixel Max and 30 Pixel Min Suppression

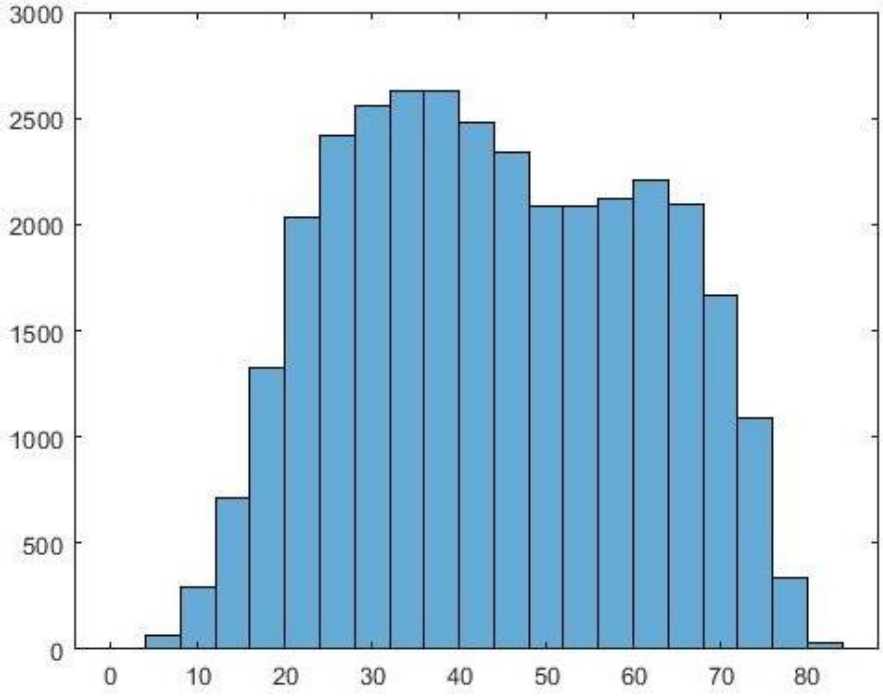


Figure 52: 30 wt% Distribution, 75 Pixel Max and 30 Pixel Min Suppression

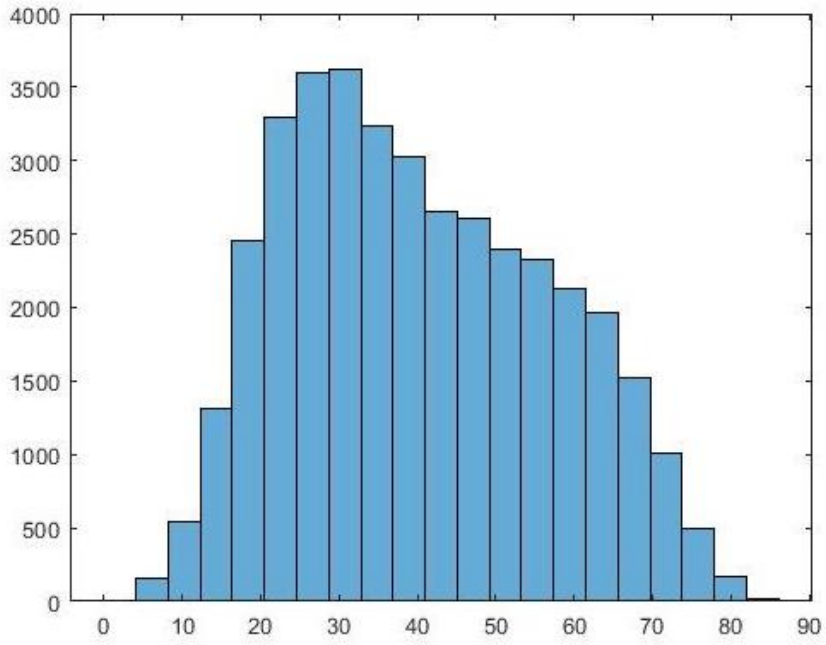


Figure 53: 40 wt% Distribution, 75 Pixel Max and 30 Pixel Min Suppression

Table 6: Mean Angle, 75 Pixel Suppression

	40 wt %	30 wt %	20 wt %
Manually Estimated Average Angle (deg)	32.9	31.8	30.2
Program Estimated Average Angle (deg)	42.3627	43.5870	40.9464

Generally, the shape of Figure 51, Figure 52, and Figure 53 are promising. While there are still many more fibers within the 40, 50, and 60-degree buckets than measured by the manual study, the peak at 60 degrees has diminished significantly. The shape of each figure is closer to that of the distributions in the manual study. The remaining difference between the data filtered for major axis length and the sample data from the study by Hollinger et al. has multiple implications. The data outside the sample used in the original study could be much different than the rest of the image data. This is unlikely, however. While the actual population mean will differ slightly from the means in the study, the differences in Table 6 are not realistic. Some difference between the means can be explained this way, but a quick glance between the binary images

filtered for major axis length and the original microscope images show that the binary imaging still has inaccuracies that can only be remedied by better data omission or binary imaging processes. Examining the binary images, many merged fibers and errant data were eliminated, but a significant amount of pertinent data were removed as well. There were also many errant data created because of poor thresholding that persisted through the major axis filtering. Filtering by major axis length alone is not enough to excise all the errant data.

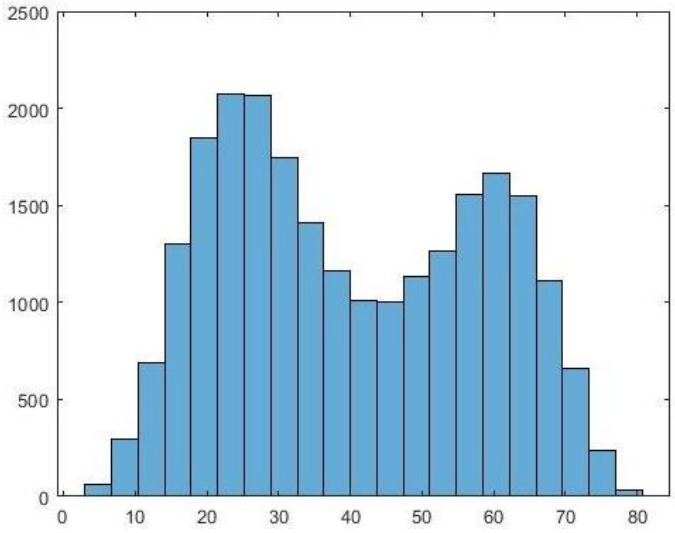


Figure 54: 20 wt%, 50 Pixel Max and 30 Pixel Min Suppression

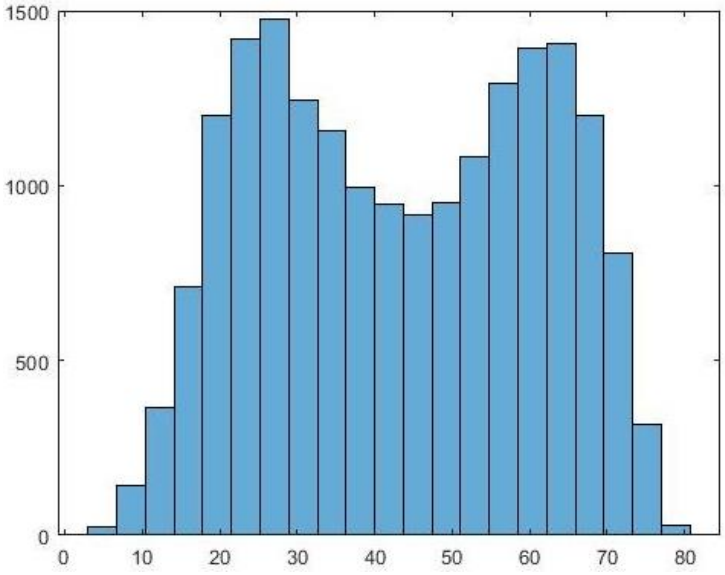


Figure 55: 30 wt%, 50 Pixel Max and 30 Pixel Min Suppression

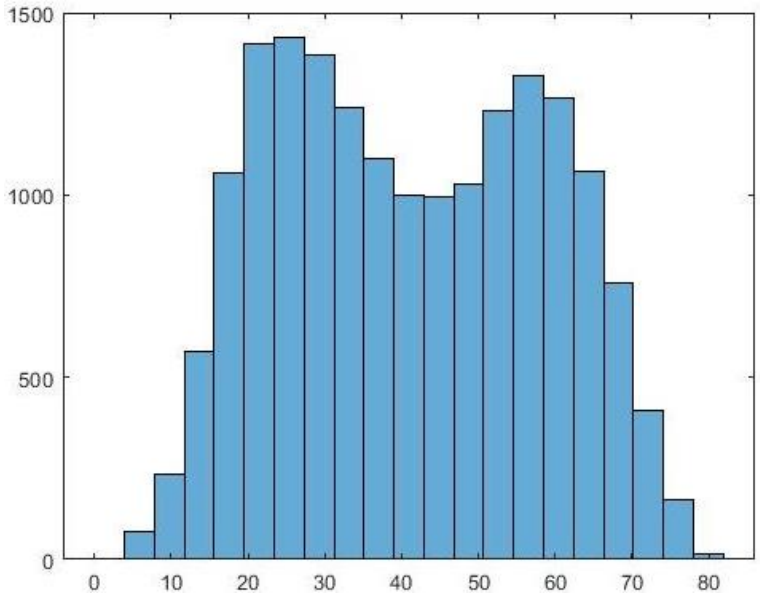


Figure 56: 40 wt%, 50 Pixel Max and 30 Pixel Min Suppression

Table 7: Mean Angle, 50 Pixel Max and 30 Pixel Min Suppression

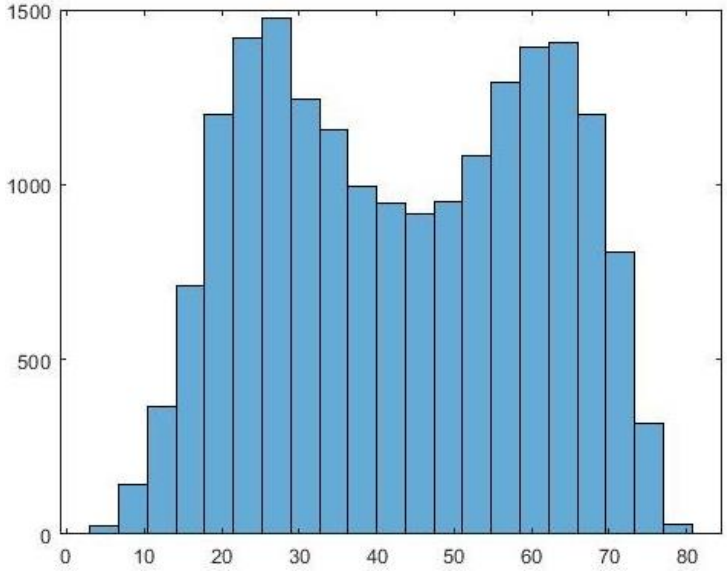
	40 wt %	30 wt %	20 wt %
Manually Estimated Average Angle (deg)	32.9	31.8	30.2
Program Estimated Average Angle (deg)	43.2492	41.6208	39.3759
Percent Difference	10.05	30.88	30.38

The 50-pixel major axis suppression produces some interesting results. Although the mean values shown in Table 7 are much smaller than previous data, the shape of Figure 54, Figure 55, and Figure 56 resemble the same saddle shape shown in earlier plots. Although even fewer fibers are found within the 40- and 50-pixel buckets, the peak at 60° has returned. These figures corroborate the idea that there are more exact methods of data omission than suppressing components based on major axis length. Additionally, note the scale on each of the figures shown above. The decrease in scale with each more restrictive pixel suppression shows that limiting data to objects equal to or below 75 pixels in length eliminates a considerable amount of the data. When

filtering by major axis length in the future, the exact pixel length will need to be studied further to ensure as much errant data is removed, and as much pertinent data is persevered as possible.

Results from this study showed that data omission would likely not be as simple as previously thought. Size is not the only valid criteria useful for filtering data. Further, there are multiple other criteria that could be used in conjunction with one another to produce optimal results. Omitting data, in this way, is very similar to refining the binary imaging technique. Additionally, this study on major axis length showed that simply omitting data would not be enough to achieve accurate results. The image pre-processing techniques would need to be revisited to ensure as much pertinent data was included as possible. Exploring the different data omission techniques was the next step in research during the spring semester.

Figure 54: 20 wt%, 50 Pixel Max and 30 Pixel Min Suppression



Chapter 7 : MORE DATA OMISSION TECHNIQUES

Omitting Data by Minor Axis Length

Another data omission technique that proved useful was filtering connected components by minor axis length. Filtering by minor axis length works similarly to filtering by major axis length. Filtering by major axis length can eliminate clustered fibers read as one large fiber or for large artifacts that exceed a certain length. However, the limits or the exact pixel quantity that should be used in the bwpropfilt() command are hard to identify. Previous discussion showed that limits on the major axis removed pertinent data as well as errant data. The major axis criteria, by itself, fails to distinguish between clustered fibers or elliptical fibers with a large major axis. The goal of the program is to obtain the actual population mean angle and to include as much data as possible. A new method of measurement needed to be used to retain pertinent data. The minor axis, unlike the major axis, does not change with respect to the angle of the fiber.

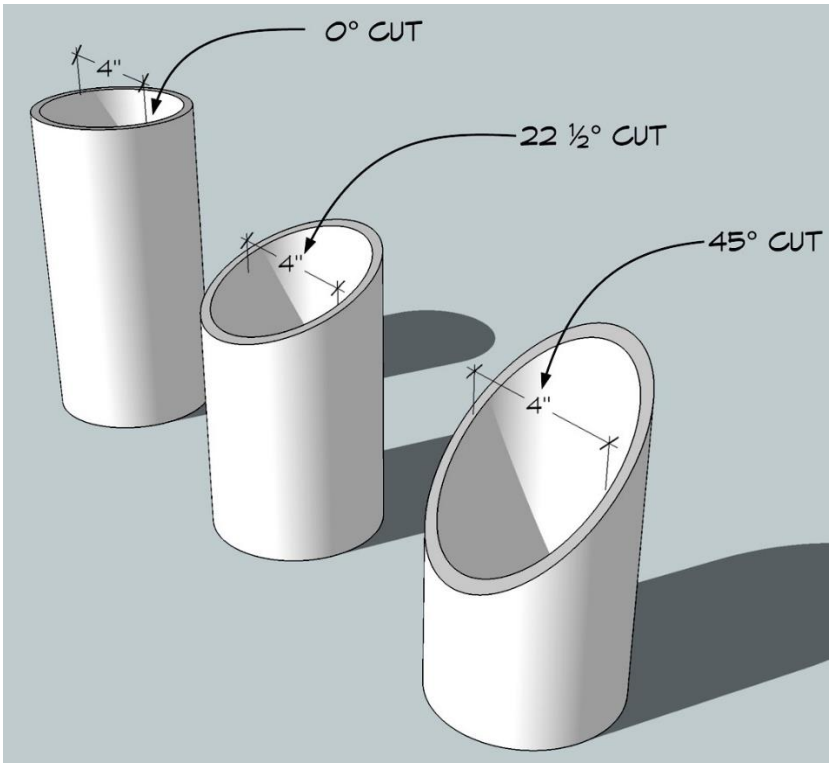


Figure 57: Minor Axis Independence [33]

Figure 57 shows how the shape of the cross section of a fiber changes as its angle within the surrounding nylon changes. The ellipse becomes more eccentric, but only the major axis changes. The minor axis never deviates from the diameter of the fiber. If the average diameter of a fiber is known, bwpropfilt() can be used to filter connected components based upon a tolerance about the average diameter of the fiber. If the minor axis far exceeds acceptable bounds, the connected component may be multiple clustered fibers or small artifacts formed as a result of poor binary imaging. Minor axis filtering will also remove any errant data initially missed by using bwpropfilt() to filter components based upon area. Components that exceeded the minimum area, but still were thresholding errors, could be eliminated by failing to meet the minor axis requirement. To start using minor axis filtering, the average diameter of a fiber needed to be understood in units of pixels.

The manufacturer data sheet indicated that the average fiber was 6.75 microns in diameter [10]. To find this length in pixels, an image and fiber were arbitrarily chosen to represent the average pixel length of the fibers. Measuring multiple different fibers to find an average length would have proven too time consuming for a method of data omission that, like major axis filtering, only partially solved the problem or yielded no real results. This fiber was chosen and then measured with ImageJ software; no scale or conversion was included in the images used by Hollinger et al. This process is visualized in Figure 58.

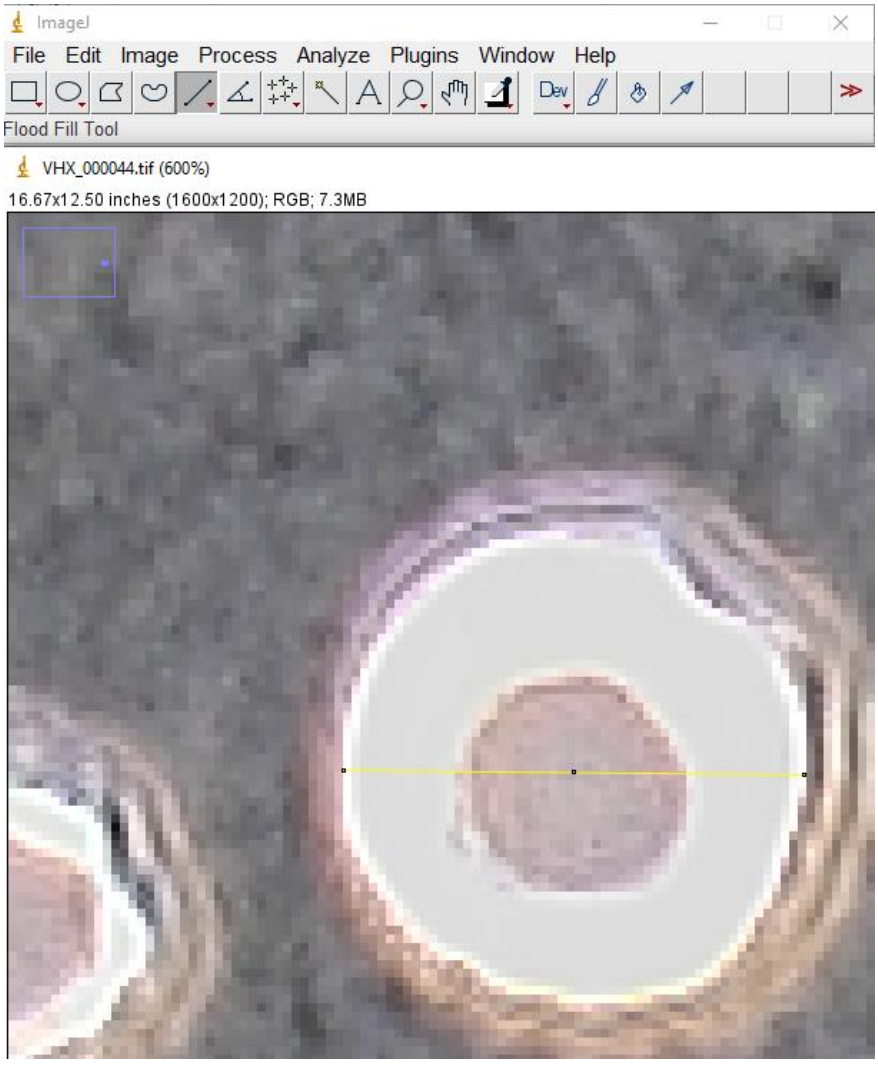


Figure 58: Measuring the Diameter in ImageJ

The line tool was used to draw a line across the diameter of the fiber. Holding the “SHIFT” key allows for the user to draw a straight line. With the “Set Scale” function beneath the “Analyze” tab, 6.75 microns was used as the scale and the known length of the fiber diameter. Figure 59 visualizes this process.

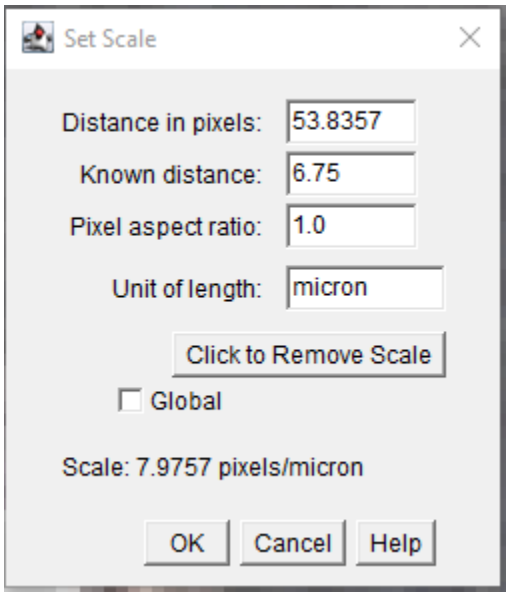


Figure 59: Scaling in ImageJ

ImageJ found the scale to be approximately 8 pixels to one micron. The “distance in pixels” entry in Figure 59 shows that the average fiber diameter is approximately 54 microns. Intervals surrounding 54 pixels should prove useful for finding the criteria for bwpropfilt().

One shortcoming from previous work in data omission with major axis length was that the exact quantity of omitted data was unknown. Omitting some pertinent data is unavoidable, but the goal of data omission is to eliminate as much errant data as possible while preserving as much relevant data. In addition to determining the average fiber diameter in terms of pixels, modifications were also made to the program so that one method of data omission could be compared to another or to a control image. Implementing this change in practice is relatively simple. After the image is processed and fiber angles are initially measured, the same image can

be saved to another variable as part of bwpropfilt(). The connected components in the new image can be analyzed in the same way as before. The filename of each image, the number of fibers in each image, and the total number of fibers for both sets of images can be recorded to a .csv file for further analysis in Microsoft Excel. Shown below is a continued analysis of major axis length filtering.

Table 8: Summary of Results from Study by Hollinger et al

NiCF Weight Percent	Number of Images	Mean Angle
20	536	29.53
30	480	28.30
40	533	32.89

Table 8 summarizes the results of the statistical study by Hollinger et al. Table 8 is included for convenience when reading future tables.

Table 9: 100 Pixel Length Maximum

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	48868	21.19	45.96	43.45	5.61	38.15
30	51080	39569	22.54	47.70	45.69	4.30	47.00
40	60816	46067	24.25	46.43	42.80	8.14	26.19

Table 9 shows the amount of data omitted or suppressed for each weight percent and the impact of the suppression. Table 9 compares the number of components found via the 100-pixel maximum filtering (for the minor axis) to the number of components found via a modified version of the image processing program developed over the summer. Unlike the summer program, the modified version does not open and close the image and uses bwareaopen(). The bwareaopen()

command does not open the image in the traditional sense with erosion and dilation, rather the method simply eliminates connected components under a certain pixel area.

```
img = readimage(ds, i);     % read in image
img = img(:,:,1:3);         % extract only channels 1,2,3 or R, G, B
imgray = im2gray(img);      % convert to grayscale image
imgbw = imbinarize(imgray); % convert to binary image (only black or white)

% edit binary image (remove artifacts, simplify data)(

imgbw = bwareaopen(imgbw, 200); % open the binary image (connect dark spaces, eliminate small white blobs)
imgbw = wiener2(imgbw, 5);      % apply a wiener filter to the image (sharpen edges, eliminate connections)
imgbw = imfill(imgbw, 'holes'); % fills any tiny black blobs within fibers, removes tiny white spaces

% connect objects, eliminate tiny marks, etc.
ellipses = bwpropfilt(imgbw, "Area", [250 10000]); % only show objects within this pixel area. remove more tiny/huge white spaces
```

Figure 60: Modified Summer Code for Comparison

The code in Figure 60 was modified from the original code after some experimentation. Ideally, the binary images produced by the code in Figure 60 capture as much data from the original image as possible. After much experimentation and reviewing images, opening and closing the image with imopen() and imclose() was found to have no real effect; these operations were therefore removed. Other changes in order of image processing operations or arguments in the program were found after much experimentation and review of the images.

The practice of comparison seen in Table 9 was used throughout development. Methods of data omission and different image processing techniques were compared to one another and to the modified summer code in Figure 60. The goal of comparing practices was to identify which eliminated the least amount of overall data while reducing the mean to be as close as possible to the means found by Hollinger et al. For comparing two methods of data omission or image processing on the same image, eliminating less data while lowering the mean angle is desirable. The second image ideally has more fibers and a lower mean angle than the first, meaning less data had to be removed and a lower mean angle was found: only errant data was removed, and pertinent data that would have been analyzed by Hollinger et al. or other researchers remains. Throughout development, tables like Table 9 are used to illustrate the effectiveness of any given method.

For a 100-pixel maximum for the minor axis, the means shown in Table 9 still differ from one another by over 25% for all weight percents. Additionally, the mean angle of the omitted data set only differs from the original data set by an 8% maximum. These differences are unimpressive and are also indicative that the data omission process should be further refined. A quick comparison between the sixth column of Table 9 to the final column of Table 8 further shows how little the omission shifted the data.

Note that this comparison made to illustrate data omission between two binary images is not perfect. The number of fibers within the initial binary image is not perfectly representative of all the data in the original image. Some excess fibers are included from thresholding errors, and other fibers are clustered together or ignored by other inevitable imperfections in the process. No process captures all the data short of manually measuring the data as Hollinger et al. did as part of a very time-consuming procedure. This process could not hope to be repeated for every image, as a glance at the middle column of Table 8 will show. The next best mode of comparison for images not previously analyzed by Hollinger et al. is a binary image produced by the program that includes as much relevant data as possible. The program cannot easily decide which data is pertinent and errant, so including all errant and pertinent data and trying to have as little data cut as possible while decreasing the distance between the program mean and the study mean as much as possible is the closest measure of making sure only errant data is removed. Minimizing the percent ignored data from one image to the next and the percent difference from the program to the study is an effective way of making sure changes are feasible and incremental in nature while also regarding the impact of the omission on the entire data set. Further, comparing the results of the program only to the sample images used by Hollinger et al. is not necessarily the best course of action, considering the underdeveloped state of the program. The goal is to see how each method affects

the entire spread of the data, not just one random sample. More images provide a broader understanding of the underlying issues with the program and the data.

Table 10: 75 Pixel Length Maximum

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	41774	32.63	45.96	41.29	10.70	33.21
30	51080	33212	34.98	47.70	44.09	7.87	43.62
40	60816	38543	36.62	46.43	40.24	14.28	20.10

Table 10 shows definite reductions in the mean angle at the cost of roughly 10% more data omission. Without comparing the binary images, the true impact of ignoring this quantity data is difficult to contextualize. Generally, ignoring 30% or more data from the original image showed noticeable but still acceptable levels of pertinent data omission in addition to errant data omission.

Table 11: Minor Axis Between 30 and 70 Pixels

Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	40040	35.42	45.96	40.95	11.53	32.41
30	51080	30700	39.90	47.70	42.54	11.44	40.20
40	60816	41422	31.89	46.43	42.28	9.36	24.98

Table 11 shows the results for minor axis filtering where minor axes less than 30 pixels and greater than 70 pixels are removed. 30 to 70 pixels was selected as an interval because the range encompasses the estimated 54-pixel average diameter and provides some room for deviation. The deviation or tolerance in diameter was not given by the manufacturer, so an interval of 30 to 70 pixels was created after viewing binary images. Compared to the major axis, more data is removed while the percent difference between the program and study is generally unchanged.

Table 12: Minor Axis Between 50 and 100 Pixels

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	12002	80.64	45.96	47.78	3.88	47.21
30	51080	12567	75.40	47.70	47.73	0.06	51.11
40	60816	17576	71.10	46.43	44.95	3.24	30.99

Table 12 shows the results for minor axis filtering for minor axes between 50 and 100 pixels in length. Much more data is removed, yet the average angle is relatively unchanged. These results do not necessarily imply that minor axis filtering is worse or has no place in the program. The results in Table 11 are comparable, if slightly worse, than results from Table 10. The interval used to form Table 12 is simply omitting the wrong spectrum of data. Too many smaller, more circular fibers are eliminated with the 50- and 100-pixel constraints. Calibration of minor axis filtering can possibly lead to better results. In fact, it is possible that the data removed by the minor axis filtering and that the data removed by the major axis filtering are two different sects of errant data. Major axis filtering is better for removing large artifacts or clustered fibers, but minor axis filtering is better for removing misshapen objects that are the result of thresholding errors. Combining both could improve the results considerably. For the following tables, let A represent the major axis and B represent the minor axis.

Omitting Data by Major and Minor Axis Length

Table 13: 30 < A < 100, 30 < B < 70

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study

20	62004	34116	44.98	45.96	36.74	22.30	21.76
30	51080	25606	49.87	47.70	38.27	21.94	29.95
40	60816	33552	44.83	46.43	36.84	19.57	11.33

Table 13 shows results for data omission when fibers are first rated against the major axis criteria and then the minor axis criteria; any remaining components satisfy both requirements. When filtering out connected components as in Table 13, about half of the data present in the original binary image is removed. Studying each binary image, many of the removed fibers are errant, but there is a significant amount of relevant data that is removed as well. Notice, however, that the mean angle is much closer to the means found in the study. Although more data is ignored than is desired, also notice that the amount of ignored data has increased marginally from the other methods despite the use of two data omission methods. Major axis filtering removes clustered fibers and long streaky fibers or artifacts, but it does not remove mishappen artifacts. Further, the average angle does not change much with major axis filtering alone. Minor axis filtering removes small mishappen artifacts but can also remove normal fibers. As a result, as many low angled fibers are removed as artifacts are, leading to a small reduction in mean angle. Both methods combined remove more errant data, but also more pertinent data, resulting in a smaller average angle.

Both methods together also have a negative geometric implication. By limiting the major and minor axis length, the maximum fiber angle is also limited. The inverse cosine function returns a value of 90 degrees for an input of 0, or a line where the minor axis is zero. For an input of 1 where the major and minor axis are equal, the return value is zero. In Table 13, the smallest minor axis is $b = 30$ pixels, and the largest major axis is $a = 100$ pixels. The largest possible angle for any fiber is 72.5 degrees. Any other fiber will be ignored. Although both methods do eliminate errant data, any pertinent data above 72.5 degrees is certainly eliminated.

$$\theta_f = \arccos\left(\frac{30}{100}\right) = 72.5°$$

Reviewing the histograms in Figure 40, Figure 42, and Figure 44, eliminating the entire population above 72.5 degrees seems reckless. Further, claiming that only errant data was removed and that the resulting images and statistics are a fair assessment of the data would be false. Although many fibers above 72.5 degrees are the result of image processing errors, there is no way to verify how much errant data is removed or how much pertinent data is retained short of counting individual fibers. These techniques should not be used together, or at least used sparingly together, to avoid this issue.

To better illustrate the effects of using both techniques together, the procedure was repeated with harsher constraints on the lengths of the major and minor axes.

Table 14: 30 < A < 75, 30 < B < 70

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	27892	55.02	45.96	32.48	34.37	9.51
30	51080	20146	60.56	47.70	34.05	33.39	18.44
40	60816	27060	55.51	46.43	32.45	35.45	1.35

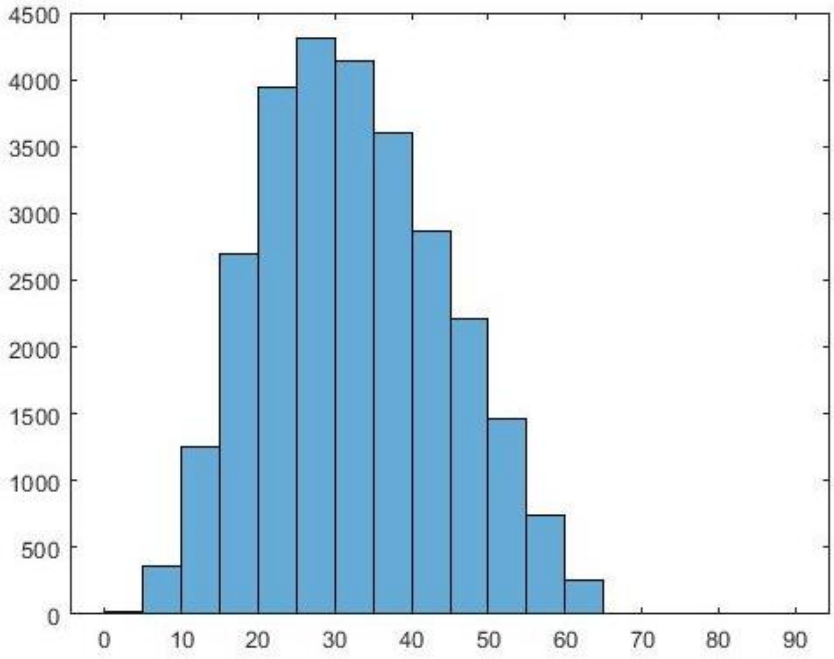


Figure 61: 20 Weight %, 30 < A < 75, 30 < B < 70

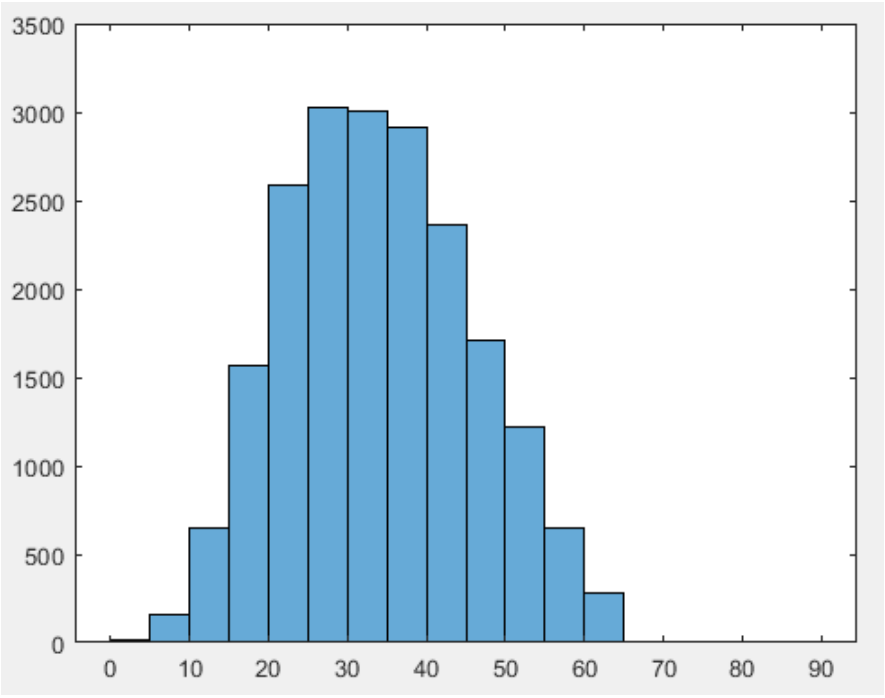


Figure 62: 30 Weight %, 30 < A < 75, 30 < B < 70

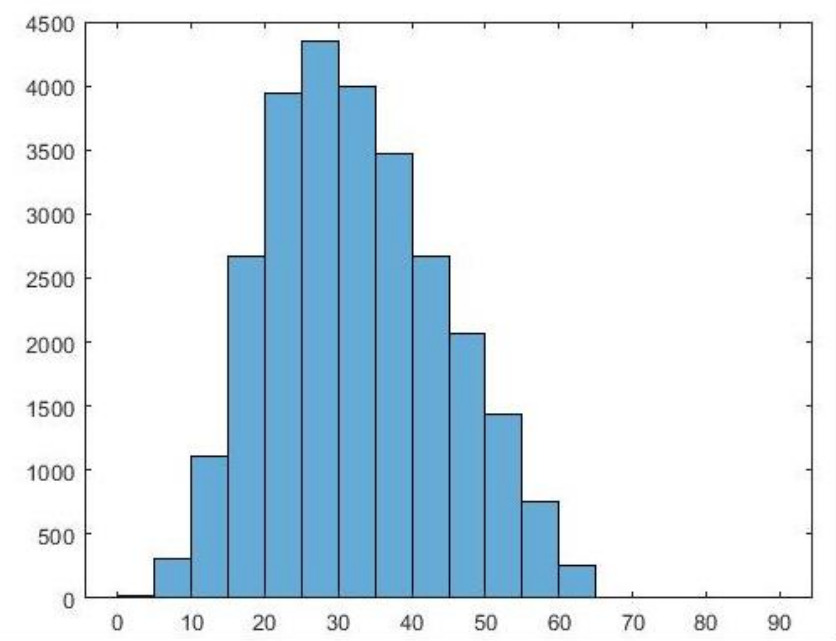


Figure 63: 40 Weight %, 30 < A < 75, 30 < B < 70

The data reduction can be seen more harshly in Table 14 and Figure 61, Figure 62, and Figure 63. Although the mean values are much closer to those shown in Table 8, the data reduction has increased to half of the data within the original binary image. Additionally, the histograms show no data in buckets for fibers angled above 65 degrees. These histograms also show less skewness than previous program generated graphs or the graphs from the original study. The distributions are arguably normal in the most recent plots. Viewing binary images now, the amount of pertinent data ignored is significant. Harsher constraints beyond those shown in Table 14 found the means delving below the values found in the original study.

Omitting Data by Solidity

One other data omission property that was analyzed was the solidity of a figure. Solidity was mentioned briefly as bwpropfilt() was mentioned at the start of discussion of data omission.

Solidity is the ratio of white pixel area to total area found within a convex polygon circumscribed to any connected component [34]. A simpler way of envisioning solidity is wrapping a rubber band around a connected component [34]. The ratio of the white area inside the rubber band to the total area inside the rubber band is solidity.

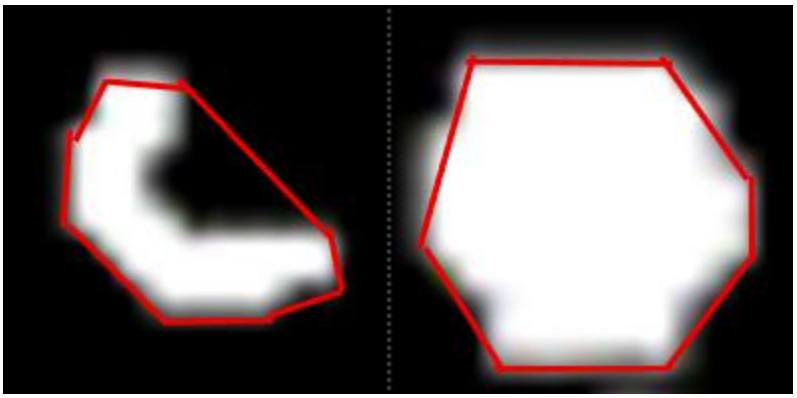


Figure 64: "Hollow" Component (Left) versus "Solid" Component (Right)

Figure 64 helps to clarify the concept of solidity. Solidity is an attractive property because of its ability to remove artifacts or thresholding errors. As previously mentioned, thresholding can create oddly shaped components that are recognized as fibers. Additionally, thresholding can also cause several very clustered fibers to be recognized as one large fiber. Clusters of fibers and artifacts often form irregular or "hollow" shapes, whereas fibers themselves are very regular "solid" shapes. Filtering by solidity could eliminate more errant data and preserve more relevant data than major and minor axis filtering, especially considering the geometry constraints imposed by major and minor axis filtering.

Table 15: Suppressing Components Less Than 0.85 Solidity

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	41123	33.68	45.96	39.78	14.42	29.58
30	51080	29503	42.24	47.70	40.70	15.84	35.94
40	60816	43601	28.31	46.43	41.24	11.84	22.53

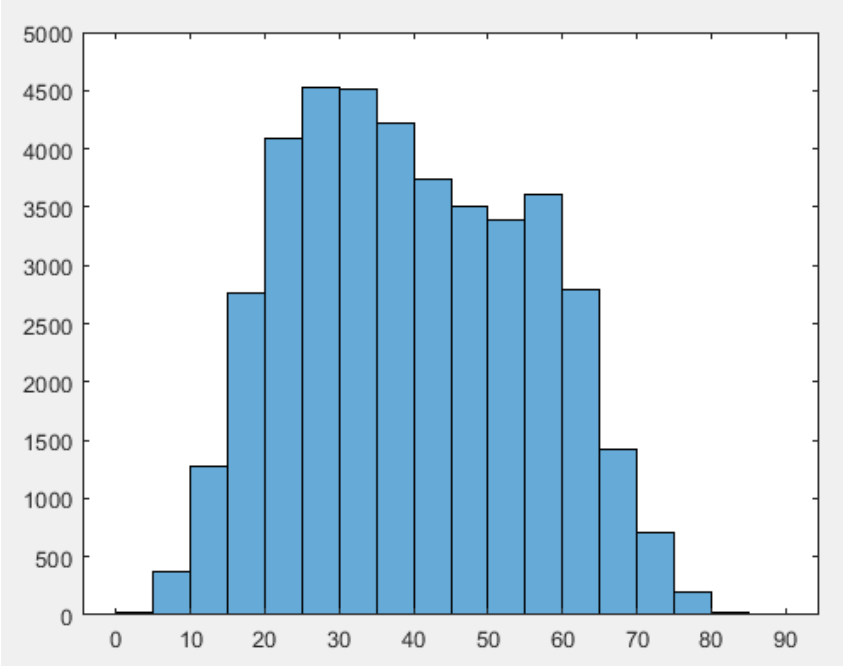


Figure 65: 20 wt%, 0.85 Solidity

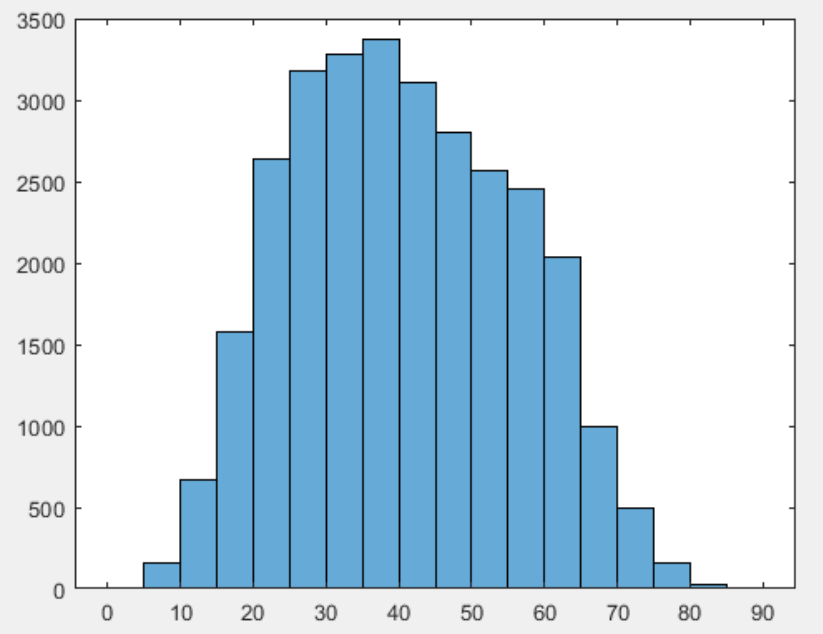


Figure 66: 30 wt%, 0.85 Solidity

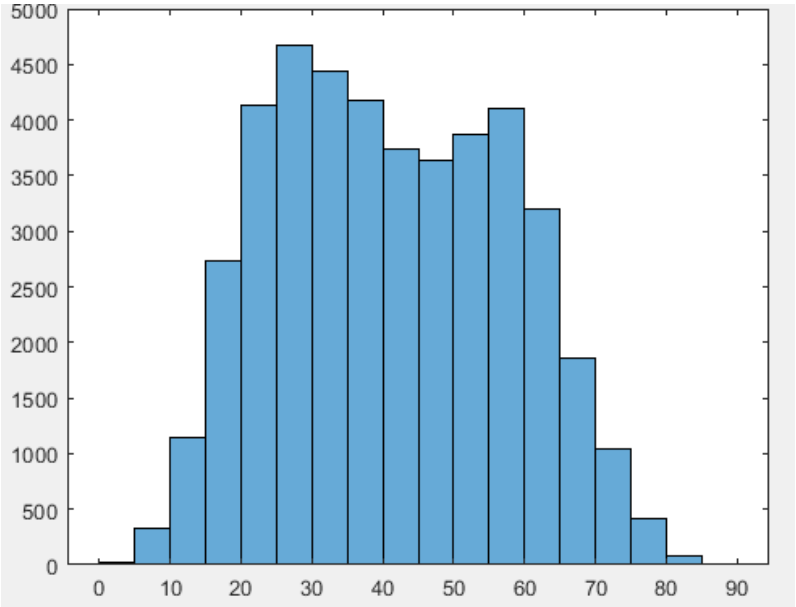


Figure 67: 40 wt%, 0.85 Solidity

Figure 68: Example Component Under 0.85 Solidity

Figure 68 shows an example of a figure with less than 0.85 solidity. This component has a solidity of 0.83. Note that this image is not a perfect match for any component under a solidity of 0.85, but it does create a satisfactory mental image. Although these figures do not perfectly match the histograms from the original study, note the smaller peak at 60° present in Figure 65, Figure

66, and Figure 67. Also, note the percent ignored data and mean angle values in Table 15 compared to Table 9, Table 11, or Table 13. Both sets of values are considerably lower with solidity than with other tests. Increasing the solidity requirement will likely omit more data and decrease the mean angle like the other properties did. However, solidity provides an increased level of confidence that the data points that persist after filtering are truly fibers.

Table 16: Suppressing Objects Less Than 0.9 Solidity

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	30968	50.05	45.96	35.89	24.61	19.44
30	51080	20734	59.41	47.70	36.85	25.67	26.25
40	60816	33444	45.01	46.43	37.72	20.70	13.68

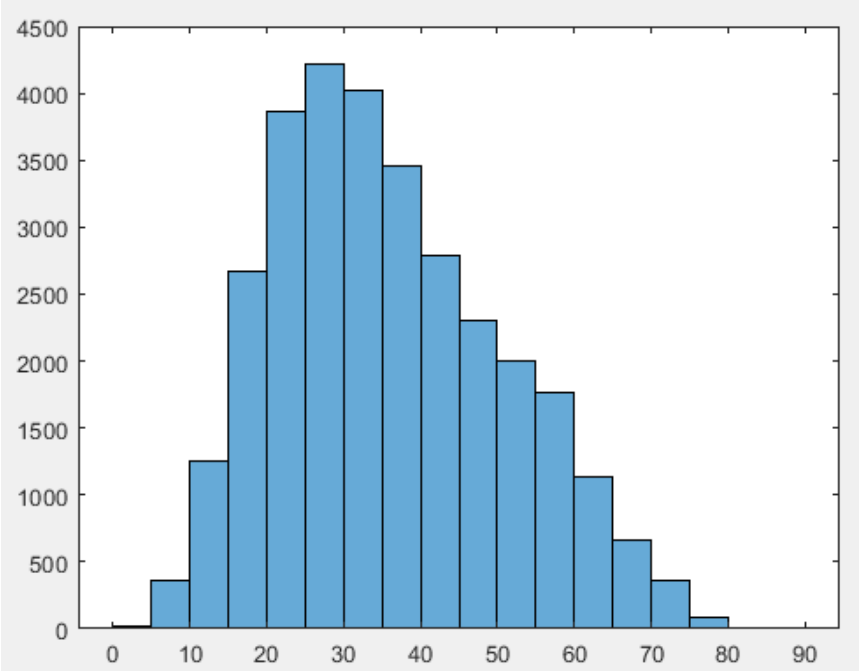


Figure 69: 20 wt%, 0.9 Solidity

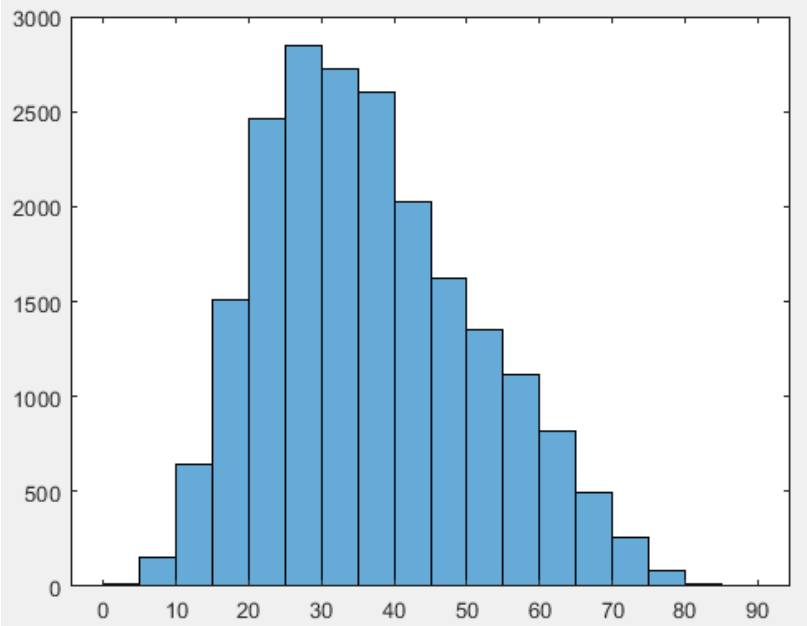


Figure 70: 30 wt%, 0.9 Solidity

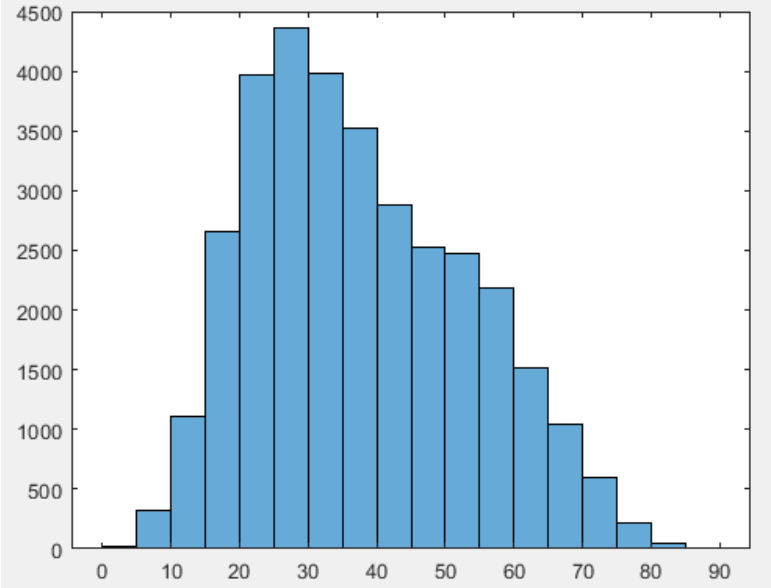


Figure 71: 40 wt%, 0.9 Solidity

The shape of Figure 69, Figure 70, and Figure 71 more closely resemble the skewed shape of the initial graphs (Figure 39, Figure 41, Figure 43). The second peak at 60° is largely diminished, the initial peak at 30° is much more defined, and there are generally less fibers above 60°. Additionally, the means also seem smaller for data lost compared to Table 14. Because of the shape

of the histograms and the additional level of confidence that comes with the solidity property, this increase in lost data can be treated less as an error and more as a statement concerning the binary imagine process. The thresholding process is not perfect, rather it is less accurate than previously thought. Comparing the solidity images to the original images, one can see that very solid fibers are ignored. Artifacts introduced by thresholding and pre-processing are “contaminating” otherwise normal fibers and preventing the program from recognizing them.

Table 17: Suppressing Objects Less Than 0.925 Solidity

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	24968	59.73	45.96	33.46	31.45	12.48
30	51080	15858	68.95	47.70	34.51	32.09	19.77
40	60816	26640	56.20	46.43	35.40	26.96	7.35

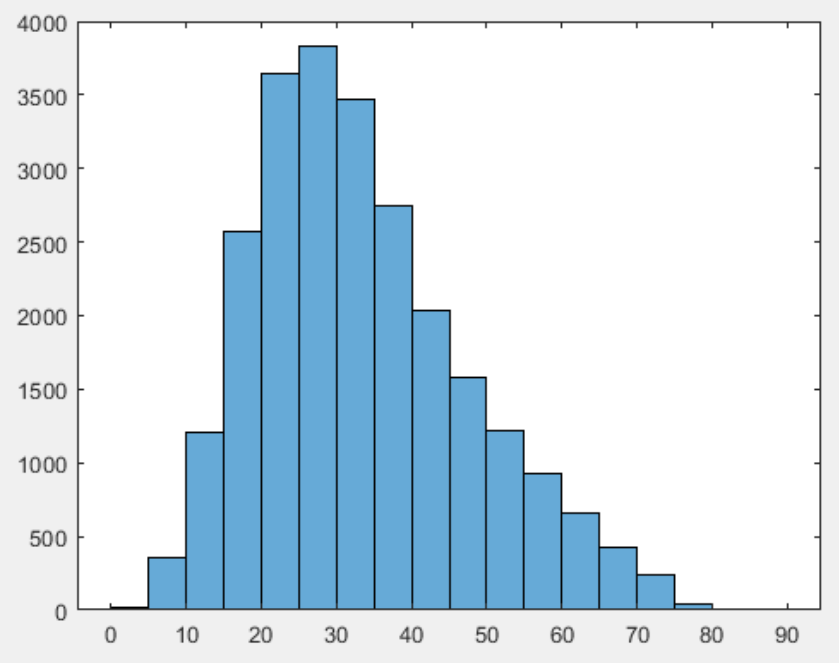


Figure 72: 20 wt%, 0.925 Solidity

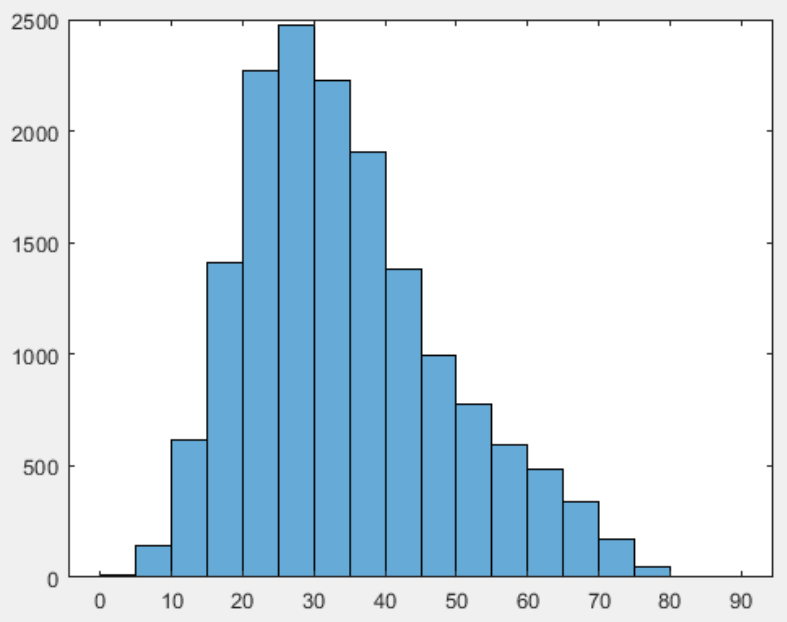


Figure 73: 30 wt%, 0.925 Solidity

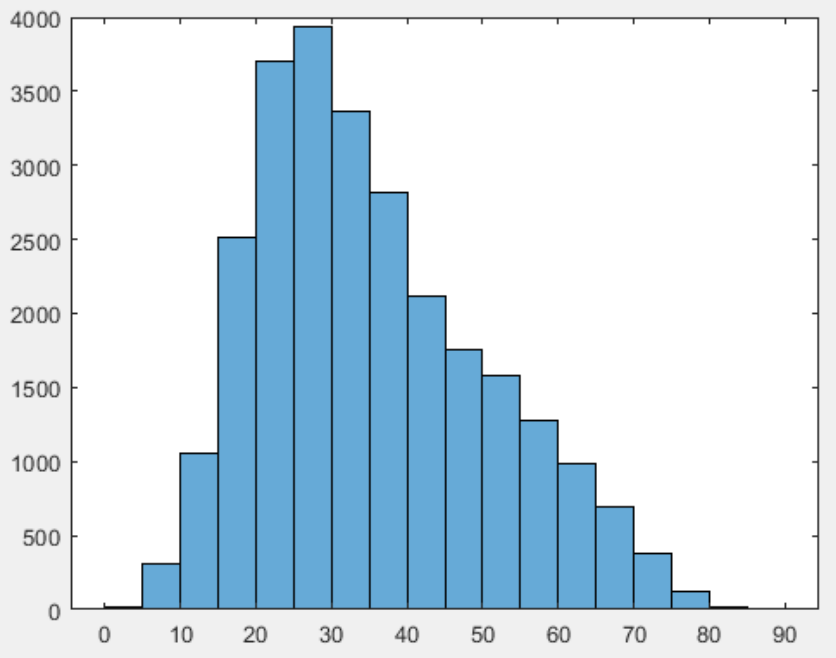


Figure 74: 40 wt%, 0.925 Solidity

Further solidity filtering enhances the shape of the plots, as seen in Figure 72, Figure 73, and Figure 74. Beyond the positive changes to the shape of the graph, Table 17 and the new images show that the increased solidity requirements are too restrictive and are eliminating too much

pertinent data: almost no fibers were visible as solidity was increased above 0.95. Despite the shape of the graph, there are still many small artifacts that bypass the area filtration requirement yet are too solid to be omitted by the solidity requirement. In the future, it is likely that a combination of the minor axis and solidity properties will be used to properly omit data. Continuing forward, the image pre-processing must be revisited. Trying to solve the problem solely with data omission has shown multiple times that the thresholding process and other image processing techniques need refined to preserve as much of the original data as possible.

Comparing to the Original Study

When changing the image processing techniques used to create binary images, smaller sets of data were focused. Specifically, the sample sets used by Hollinger et al. were analyzed. Recorded for these sample sets are the number of fibers found in each image by Hollinger et al. As mentioned previously, finding the correct processing methods, input requirements and rules for these methods, and the order in which to apply each method is challenging. With a large set of data, one can easily make hundreds of changes and feel as though none of them modified the results. The best way to refine pre-processing techniques is to change only one variable at a time and carefully observe the impact of the change on a small set of data. After understanding how the change affects the data on a small scope, only then can the change be applied to a large set of data and modified further.

These changes required many, many iterations. Please understand that not every step could be recorded or described in-depth, or the thesis would never be published. Chapters 8 and 9 serve as a summary of the many different techniques used and eventually which techniques produced

fair results. The process and results could only be verified on the 20 wt% samples. For completeness, the results for the same process applied to the 30 and 40 wt% samples are included as well.

Chapter 8 : REVISIONS TO IMAGE PROCESSIN CODE AND FINAL RESULTS

Binary Imaging Technique: Constant Threshold, Adaptive Thresholding

Earlier, Figure 11 and the surrounding text described how imbinarize() used Otsu's method to maximize variance at the edges of objects to minimize the loss of data when creating a binary image. Otsu's method is robust and can be applied to most images with success, however, it is not necessarily the fastest or most accurate solution for every image. Sometimes extra details and artifacts are included when the binary image is created, which necessitates data omission and inevitably causes errant data to skew results. When faced with the task of improving the thresholding process, two methods seemed applicable.

First, a constant threshold could be used for each set of data. Unlike Otsu's method where each image has a unique threshold value computed at runtime, the user would instead decide on one threshold value that would apply to all images in one data set. An advantage to choosing a constant threshold to apply to each image is that it is considerably faster than finding the optimal threshold for each image. Additionally, a value can be picked to the convenience of the user. If the user does not desire complex data omission techniques to be used, a more conservative threshold can be chosen to eliminate artifacts. The downside to choosing a more conservative threshold that works for each image is, of course, the data that is inevitably lost with a more conservative threshold. Clustered images often suffer from the error shown in Figure 27; all the fibers are considered one large fiber and ignored.

Second, thresholds can be generated adaptively for each section of the image. When an adaptive threshold is generated, a threshold is generated based on the local mean intensity in the neighborhood of each pixel [35]. Such a technique uses much more computing power than simply

choosing a threshold for each image or applying Otsu's method. However, adaptive thresholds give much greater control over the quality of the binary image.

Both methods require the user to have a certain understanding of the code and willingness to modify any other preexisting image processing or data omission techniques to remove data. Personally, I found choosing an adaptive threshold was easier. With a constant threshold, I struggled to determine an exact threshold. Much of image processing is based off heuristics and user satisfaction, but I could not tell the difference between thresholds or determine an optimal threshold value. I found development to be much easier when the threshold had an empirical source that could be explained and verified as more than a matter of personal preference. However, each set of images used by Hollinger et al. were tested. Intensity values above 180 were found to produce reliable results for all three data sets.

Eroding Images

Figure 13 and surrounding text discussed opening and closing an image. Opening an image was composed of an erosion and a dilation. Connected components were made smaller and then made larger; the consequence of these transformations was that tiny, dark spaces were widened and that the edges of any connected components appeared smoother and more solid (in the understanding of the bwpropfilt() command for filtering by solidity). Opening the image alone is rarely enough to separate clustered fibers. For clustered fibers and images full of clustered fibers, an erosion and only an erosion would need to be used to separate fibers before other operations could be used.

Eroding an image is like any other morphological operation in MATLAB. A structuring element must be designated [36]. A disk element was used to maintain as much of the original shape of the fibers as possible, as practice found that small shapes were slightly distorted after erosions. After certain morphological operations including erosion, wiener filtering was used to smooth any jagged edges that may occur as a result.

Shelling Images

For clustered fibers, one other technique was employed along with erosion to ensure separation. Eroding the image multiple times or with larger structural elements produced deformed fibers or fibers that were small enough to be mistaken for erroneous elements. The bwperim() function was used to find foreground pixels on the perimeter of any connected components [37]. Because binary images are either black or white, the pixels on the perimeter can be easily set to black.

```
for q = 1:3
    shell = bwperim(border); % get outside perimeter of fibers
    border(shell) = false; % set those pixels to black
end
```

Figure 75: Setting Perimeter Pixels to Black [38]

Figure 75 visualizes the explanation above. The perimeter of the image in question, border, is saved in shell. The second line in the for-loop changes the found perimeter into black pixels.

Excluding Elements on the Edge of the Image

Finally, one last major change was made to the pre-processing code. In all the previously mentioned images, there are fibers that lie on the edge of the field of view of the camera. These fibers appear as extreme ellipses to the program.

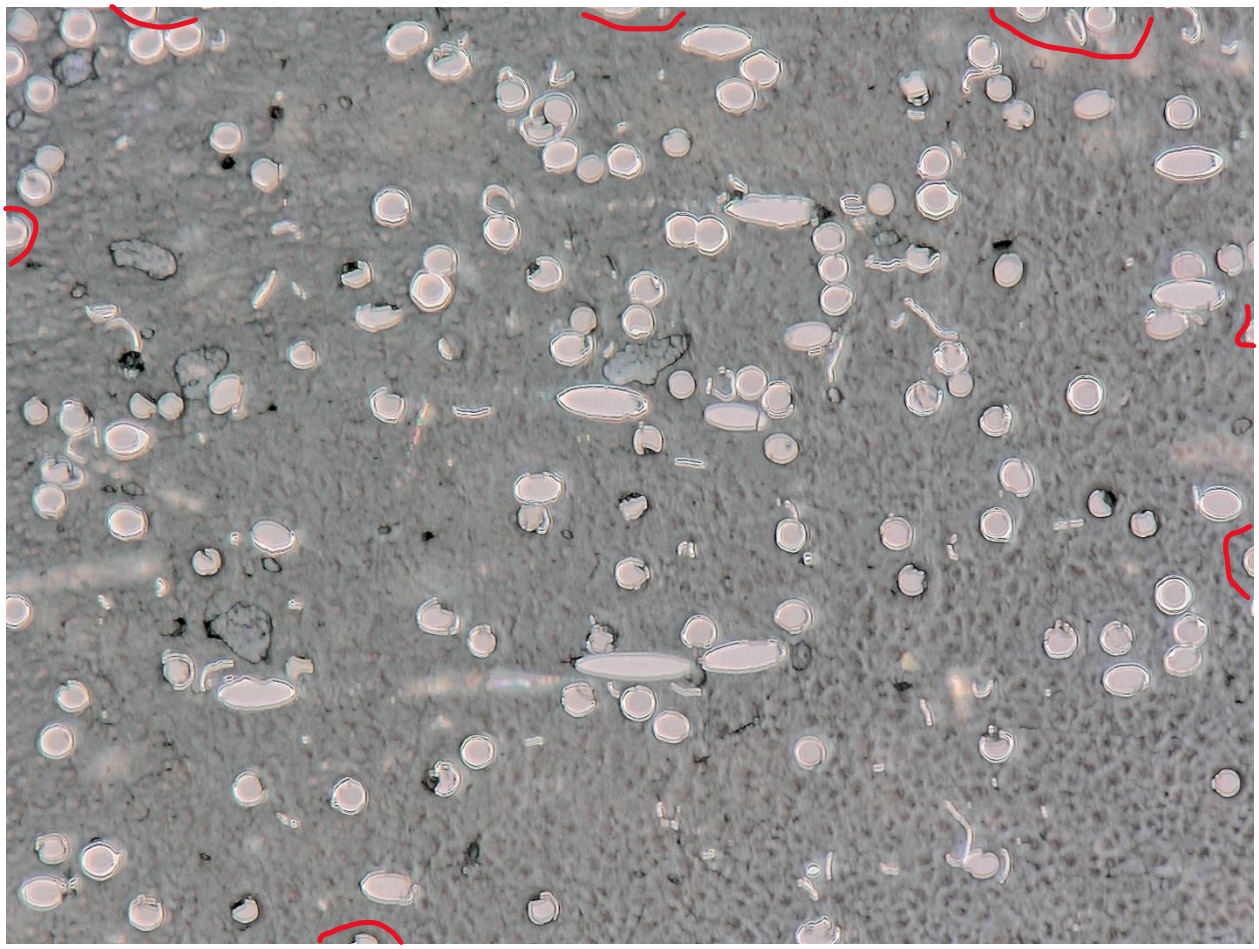


Figure 76: Fibers on the Edge of the Image

Figure 76 highlights some of these fibers with red circles. Fortunately, MATLAB has a very convenient function for identifying and eliminating objects on the edge of images. The command imclearborder() was used to eliminate fibers on the edge of the image [39].

Revised Code

```
img2 = imread("VHX_000293.tif");
img2 = img2(:,:,1:3);          % extract only channels 1,2,3 or R, G, B
img2gray = im2gray(img2);       % convert to grayscale image
img2gray = imsharpen(img2gray);
im2gbw = imbinarize(img2gray, "adaptive", "Sensitivity", 0.75); % convert to binary image (only black or white)
SE = strel("disk", 4);    % filter for opening image

% edit binary image (remove artifacts, simplify data)(
img2open = bwareaopen(im2gbw, 200); % open the binary image (connect dark spaces, eliminate small white blobs)
erode = imerode(img2open, SE);
before_open = im2gbw;
perim = erode;
border = imclearborder(perim);

% eliminate outer shell
for q = 1:3
    shell = bwperim(border); % get outside perimeter of fibers
    border(shell) = false; % set those pixels to black
end

filled = imfill(border, 'holes'); % fill any holes

% connect objects, eliminate tiny marks, etc.
ellipses_area = bwpropfilt(filled, "Area", [200 10000]); % only show objects within this pixel area. remove more tiny/huge white sp
ellipses_wiener = wiener2(ellipses_area, [2 2]);

ellipses_open = imopen(ellipses_wiener, SE);
ellipses2 = bwpropfilt(ellipses_open, "Solidity", [0.9, 1]);

montage({img2gray, before_open, img2open, erode, perim, border, filled, ellipses_area, ellipses_wiener, ellipses_open, ellipses2})
```

Figure 77: Revised Image Processing

After reading the image, transforming the image into an RGB image, and then transforming the image into a grayscale image, the image was sharpened. The command imsharpen() provides extra contrast about the edges of objects within the image. The command imsharpen() uses a technique known as unsharp masking, where a blurry version of the same image is subtracted from the original image, producing a clearer image [40].

A binary image is formed from the sharpened image in the following line. A sensitivity of 0.75 was found to work especially well for clustered images over many different tests. The image is then filtered for objects larger than 200 pixels in area (objects smaller than 200 pixels are removed). The command bwareaopen() is deceptively named; the command does not truly erode and dilate connected components, rather it acts in the same way as bwpropfilt() when filtering objects by area [41]. The image is then eroded, shrinking the fibers. The image is eroded before

the borders of the image are cleared because before erosion many elements touch elements that also touch the edge of the image and clearing the border would clear many relevant fibers. After the border is cleared, the outer shell of each component is removed three times. Three was found to be the number of iterations that provided the most clarity while also retaining the original shape of the object. After removing the perimeter of each component, the holes within the image were filled. Filling holes is important because fibers with holes in the center could fail solidity criteria used later in the program. The binary image is then filtered by area and smoothed with a wiener filter. The image is opened afterwards, also to smooth the fibers. Finally, the remaining fibers are filtered by solidity. Although not pictured in Figure 77, the final image was filtered by minor axis length to remove any artifacts or seriously deformed fibers.

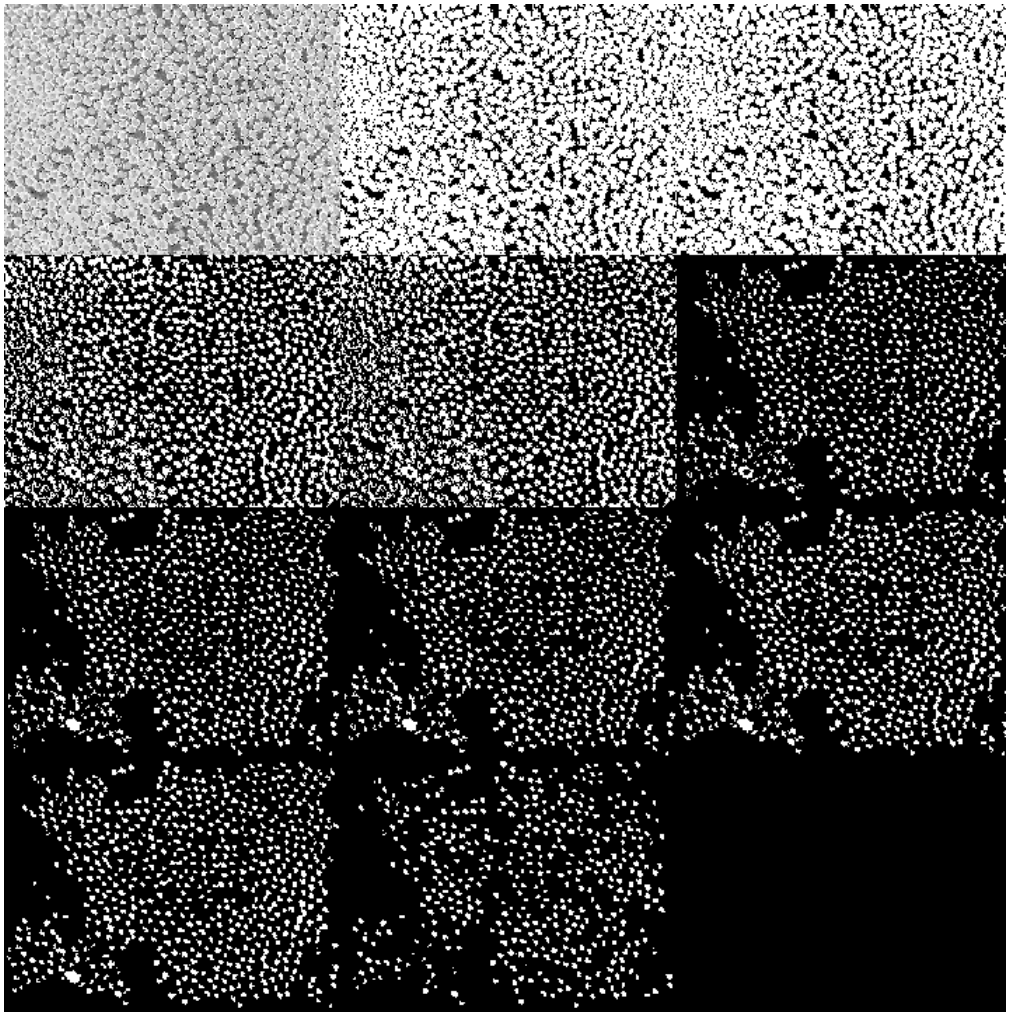


Figure 78: Images Corresponding to "Montage" Command

Ignore the final blank square in Figure 78; montage() shows all the images in a row. Because there is an odd number of images, there is one cell left blank. Each image can be compared to the last to show which fibers were removed by the relevant command. This technique, when compared to previous techniques, leads to some interesting conclusions.

Chapter 9 : FINAL RESULTS

Table 18: Statistics for New Image Processing Technique

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	62004	42152	32.02	45.96	36.33	23.41	18.43
30	51080	42723	16.36	47.70	35.09	30.46	9.84
40	60816	58402	3.97	46.43	42.07	9.85	24.46

The code used to obtain the results in Table 18 is included in Appendix C: Final Code, Comparison of Initial Methods to Final Methods. The new image processing technique eliminates considerably less data while decreasing the mean angle significantly. To even get below 36°, data omission alone was often over 50% of the data, as shown in Table 17. The new technique produces impressive results for the 30-weight percent specimen, reducing a mere 16% of the data and differing from the study results by only 10%. The 40-weight percent samples, however, behave very strangely. Only 4% of the data was ignored, yet the percent difference between the program mean and the manual study mean is the largest of Table 18 at 24%. The reduced number of ignored data shows that the new technique preserves many more fibers than past techniques using the summer code and data omission could. To gain a better understanding of the new techniques, the new binary images must be viewed.

Previously, new image processing techniques had been compared to old image processing techniques. The percent difference in fiber count between the two techniques is rather large, which indicated that a technique omitted too much data. Example images like those shown above suggest the new methodology is more accurate than the original, even though the number of fibers is lower. Comparing these two techniques, especially over the scope of the entire population, does not allow

for direct conclusions. For example, comparing two data omission methods makes sense; whichever omits less data and has means closer to the study means is a better technique. However, comparing two entirely different techniques only by the number of fibers collected and the mean angle fails to account for several confounding variables. One image processing technique could allow for more artifacts to be counted as fibers in the new binary image. The artifacts could be small and circular, which would ultimately decrease the mean. One technique could produce an image that is very different than the original, but the data could imply that the image is more accurate to the original study. A similar scenario could be causing the 40 wt% specimen to have confusing results. Instead, the two techniques will be compared over the sample data set used by Hollinger et al. This way, the exact number of fibers found by both image processing techniques can be compared to the true number of fibers within an image. The binary images produced by both techniques can also be compared visually to determine which methodology is more accurate and further any errors in the new methodology.

Overview

Table 19: Statistics for Sample Image Sets

NiCF Weight Percent	Original number of fibers	Suppressed Number of fibers	Percent Ignored Data	Mean Angle, original	Mean Angle, suppressed	Percent Difference, original to suppressed	Percent Difference, program to study
20	3822	2680	29.88	46.23	36.77	22.80	19.62
30	1683	1766	-4.93	46.81	35.87	26.46	12.03
40	3545	2987	15.74	46.45	42.63	8.57	25.76

Overall, the results from the sample tend to accurately reflect the results shown in Table 18, which gives further credibility to the sampling used by Hollinger et al. to represent the

respective populations. The results from the samples match the results of the population, yet the results from both do not match the results from the study shown in Table 8. Future versions of the program may be able to use a small sample of the total images to achieve accurate results much faster. Regardless, the sampling of images for the 30-weight percent, although having fewer images than the others, is accurate to the results for the entire 30-weight percent population. The 30-weight percent population has a negative percent ignored data; the value is listed as negative to indicate that the new image processing techniques outlined in Figure 77 found more data than the modified summer code did.

20 Weight Percent

Table 20: Elements and Mean Angle Found for 20 wt% Sample Images

Photo	# elements (Summer)	# Elements (New)	# elements (Study)	Mean Angle (Old)	Mean Angle (New)	Mean Angle (Study)
70	158	81	139	47.2521	34.2427	31.7
78	100	161	234	46.9125	37.2747	32.66
86	20	9	26	56.1095	44.542	31.31
94	100	158	279	38.5861	36.0593	26.95
102	182	64	45	46.0517	33.8573	26.42
156	81	61	64	42.1698	35.9982	26.7
164	132	69	90	44.8109	37.0089	26.93
172	132	40	144	44.1256	34.3855	26.07
184	205	50	91	49.0886	36.1745	26.1
192	100	48	67	45.3968	34.1714	25.6
242	106	75	96	43.1667	33.0503	26.56
250	160	145	229	47.811	38.2453	26.37
258	173	43	42	48.8433	40.8371	28.15
266	115	58	82	47.2682	38.1134	30.68
274	104	290	513	47.9851	32.5518	26.19
282	122	195	284	43.0411	37.5918	26.12
336	1	104	140	55.7002	36.5595	29.67
344	1	48	103	77.2056	36.6349	31.25
356	198	31	83	55.1417	38.0126	39.95
364	122	78	137	45.0455	38.2389	31.01
372	0	97	152	0	37.868	34.98
422	166	72	116	50.5443	43.6986	40.21
430	0	43	75	0	39.6694	39.87
438	124	23	76	48.1792	41.2674	34.06
446	113	98	134	47.9264	39.9466	32.71
454	152	75	118	47.341	40.4717	34.6
508	113	69	85	42.0552	37.7279	30.84
516	161	106	149	42.1748	33.6257	25.34
524	135	47	92	44.1541	36.6978	27.78
536	129	52	94	43.4788	35.8394	26.68
544	139	65	95	44.9449	33.9733	30.93
602	142	78	101	44.1121	41.5363	35.02
610	136	47	104	43.7432	38.6884	27.81

Table 20 displays the number of elements found by the modified summer code (intended to include as much data as possible), the number of elements found by the new image processing code developed in the spring semester, the number of elements found by Hollinger et al. in the manual study, and the mean angles for each of the three aforementioned entities. Generally, the new image processing code finds fewer elements than the summer code and by the study. Despite this, the mean angle for the new code is lower than that found by the summer code but is still higher than the mean angle found by Hollinger et al. in most cases. Inspecting the binary images formed by the new code, the program seems to continue to overestimate the angle of fibers. Fibers

are slightly distended, perhaps by some of the morphological operations. Although clumping of fibers is much less of an issue with the images formed by the new code, some fibers still suffer from clumping or thresholding errors. These errors can be seen especially in clustered images. Although many more fibers are found by the new code than the previous code, which often omitted all the fibers in a clumped image as errant data, at best only half the fibers in clustered images are found. Ignoring so many low-angled data points can skew the data higher. Additionally, many fibers are omitted by the solidity and minor axis criterion because of these errors. The lack of extra data points also contributes to the gap between the results of the new code and the manual study.

30 Weight Percent

Table 21: Elements and Mean Angle Found for 30 Weight % Sample Images

Photo	# Elements (Summer)	# Elements (New)	# Elements (Study)	Mean Angle (old)	Mean Angle (New)	Mean Angle (study)
51	167	108	251	46.9469	35.1271	26.24
59	149	73	112	44.8331	33.6794	26.26
105	137	99	171	43.0178	29.9741	19.3
113	169	83	145	47.5317	35.0901	32
121	145	77	191	43.4848	32.0518	24.46
133	123	90	128	44.7432	35.278	30.3
141	117	101	172	44.5781	33.7255	28.28
191	12	94	158	58.6047	34.7167	31.64
199	153	92	148	44.503	35.415	21.42
207	131	48	101	50.6728	38.6944	33.91
215	134	70	110	54.8063	64.038	66.23
223	0	43	79	0	45.5768	40.88
277	108	196	289	46.843	32.6826	25.85
285	6	51	88	56.1465	34.9123	27.26
297	112	121	248	47.7205	42.1478	41.66
305	6	74	121	55.4453	34.1468	29.36
313	14	346	868	52.7962	33.3263	18.89

Table 21 shows similar results to Table 20. The program finds fewer fibers than what a human could find in a manual study in clustered images. Something to note about both data sets is that the number of fibers found by the new code never exceeds the number of elements found in the manual study. Inspection of the images supports the idea that the new code does not create

significant quantities of errant data from thresholding errors, as can be seen in images and in data from the summer code. The new code also finds more data points overall than the summer code and produces a lower mean angle. This observation speaks partially to the success of the new code, but also to the shortcomings of the old code. Even by reviewing the images, it is unclear whether the old model is a poor fit for the data or if the new model is an excellent fit for the data. The new code appears to find fewer of the available points (according to the study) in the 30 weight percent images than with the 20 weight percent images, so the data likely does not favor the new code as much as it rejects the old code. Regardless, note that the model has varying levels of effectiveness between populations. Eventually, the goal for the program is to accurately read any data regardless of differences in the quality of photos or in the method in which images are captured. Despite this, certain variations in image quality between populations may be unavoidable. Future code should be written to incorporate these differences. Machine learning or perhaps more advanced image processing techniques may be used to make sure all images are of the same quality when analyzed by the program.

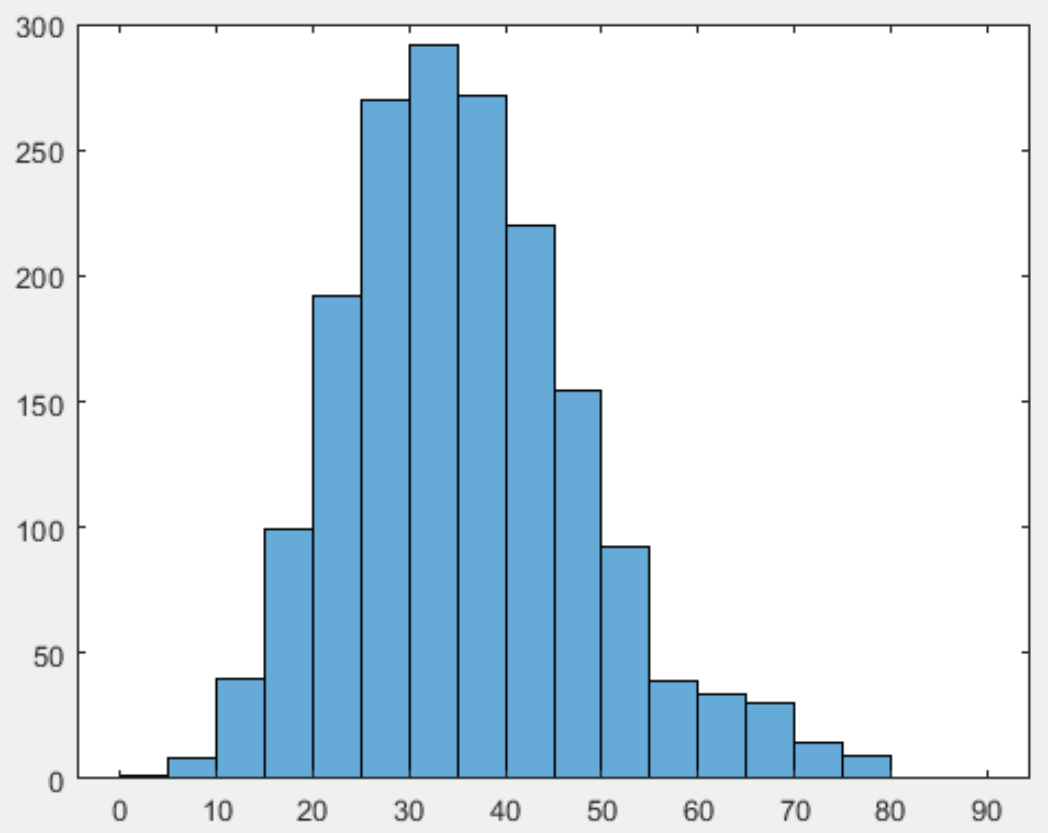


Figure 79: New Code Angle Distribution from Sample Images of 30 wt% Cross Section

Figure 79 shows the spread of the data. Compared to Figure 41, these figures are very similar. Figure 79 is shifted about 5 degrees to the right, which is expected given that the mean of the program-assessed data is about 5 degrees higher than the mean from the manual study. The bins do not align exactly between the two graphs, especially with the bins housing fibers angled between 40 and 50 degrees. Such a shift is likely a continuing issue with the program, given that the sample mean aligns so closely with the population mean. A histogram of the entire population could provide some insights, however.

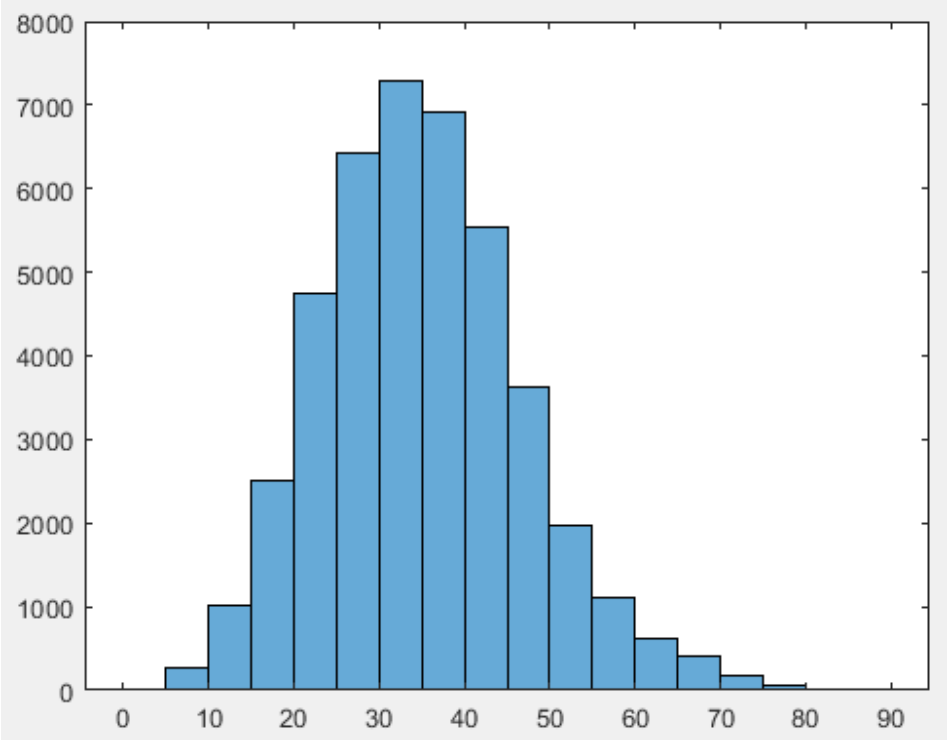


Figure 80: New Code Angle Distribution from Population Images of 30 wt% Cross Section

Figure 80 is unsurprisingly very similar to Figure 79. The trend exists throughout the population, not only in the sample image.

40 Weight Percent

Table 22: Elements and Mean Angle Found for 40 wt% Sample Images

Photo	# Elements (Summer)	# Elements (New)	# Elements (Study)	Mean Angle (Summer)	Mean angle (New)	Mean Angle (Study)
48	136	84	115	41.0541	37.3265	23.54
56	154	106	118	39.7999	37.5081	23.13
112	129	156	290	45.5291	38.4804	23.37
120	146	110	161	44.7121	41.2781	22.87
132	155	100	107	37.7594	34.7226	25.62
140	119	109	208	46.0757	41.126	34.11
148	131	210	419	44.8115	35.1807	25.5
200	110	67	145	57.8683	60.8283	63.9
208	136	153	222	43.9636	39.1617	31.64
216	152	65	100	39.2464	33.6733	21.34
224	49	221	620	48.3162	39.6825	33.44
232	50	247	429	54.3299	51.0116	47.65
288	127	25	36	54.257	53.518	58.74
296	135	29	108	48.565	46.4128	31.83
308	12	170	564	61.7561	39.079	23.23
316	131	32	35	60.8279	51.2143	61.7
368	65	109	57	60.2239	66.4283	64.98
376	153	86	70	45.7249	40.2196	33.24
384	123	70	63	41.777	36.9928	28.97
392	141	67	60	49.298	40.7125	32.79
400	125	56	49	56.4454	57.4244	64.68
456	106	39	51	57.6391	54.5321	58.4
464	16	60	48	54.8596	42.267	38.36
476	139	54	67	38.6671	35.418	25.44
484	118	144	117	53.0571	57.045	56
492	163	146	119	45.9523	38.3296	27.02
544	8	18	39	54.8291	50.685	47.51
552	121	54	87	42.2876	36.0327	31.1
560	133	44	91	37.2017	31.2153	23.07
568	132	74	117	43.8072	39.0239	22.66
576	130	82	138	41.1284	36.8558	22.56

Table 22 shows the results of the final sample image set. Generally, the results follow those seen in the previous tables; the new image processing code predicts lower angles while finding slightly less fibers. In most clustered or streaky images, image 232 or 544 for example, the program struggles to find more fibers. Both images and their respective binary images can be seen in Figure 81, Figure 82, and Figure 83 and Figure 84, Figure 85, and Figure 86, respectively.

Figure 81: Image 232, 40 wt%

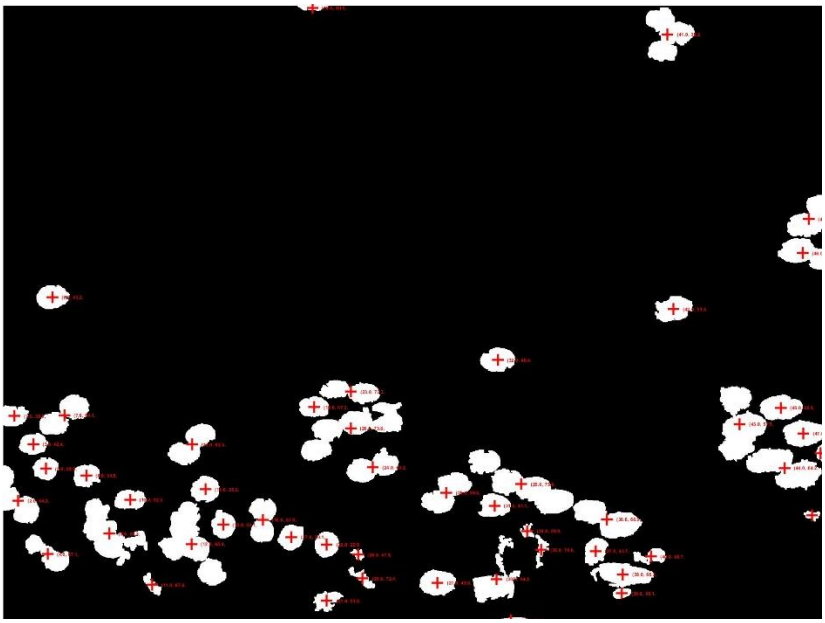


Figure 82: Image 232, 40 wt%, Summer Code

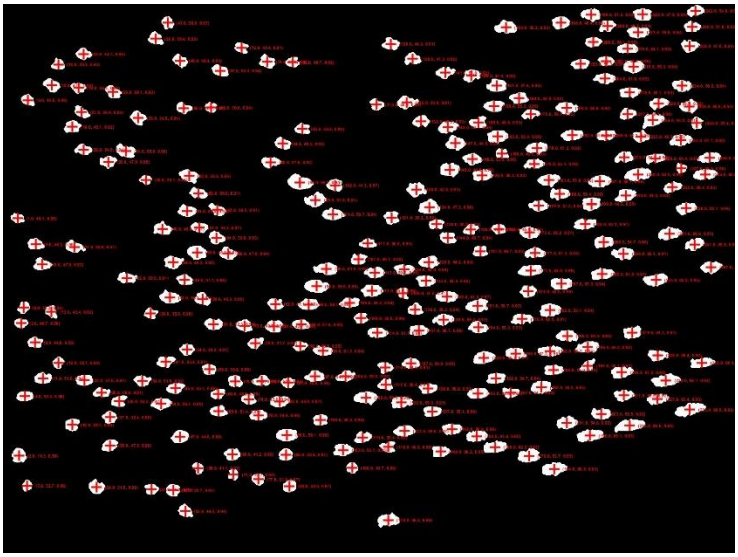


Figure 83: Image 232, 40 wt%, New Code

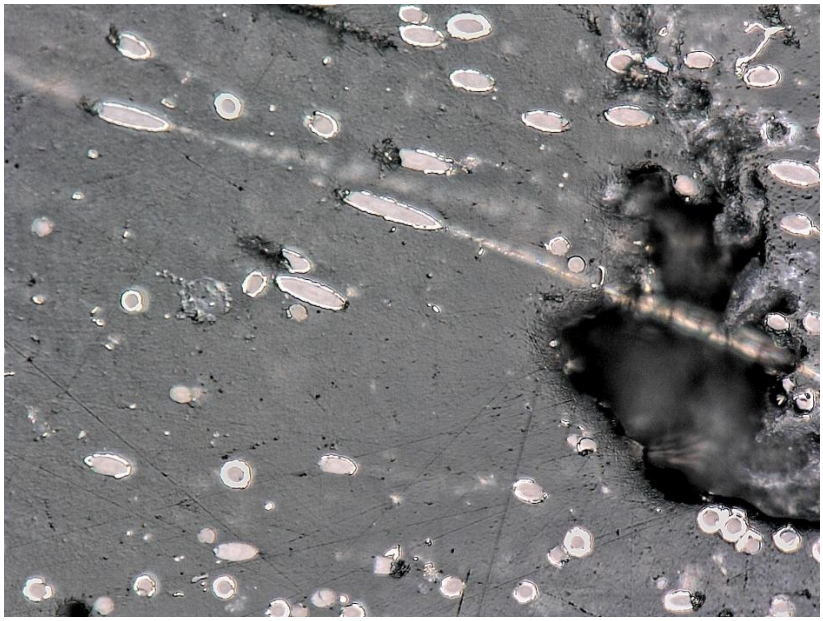


Figure 84: Image 544, 40 wt%

Figure 85: Image 544, 40 wt%, Summer Code

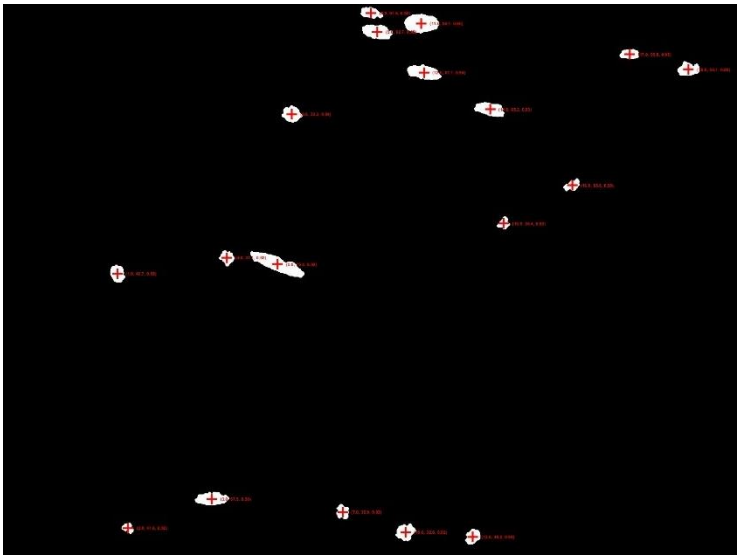


Figure 86: Image 544, 40 wt%, New Code

Table 22 also shows that the new code finds more fibers in images 484 and 492 than were found in the original study.

Figure 87: Image 484, 40 wt %

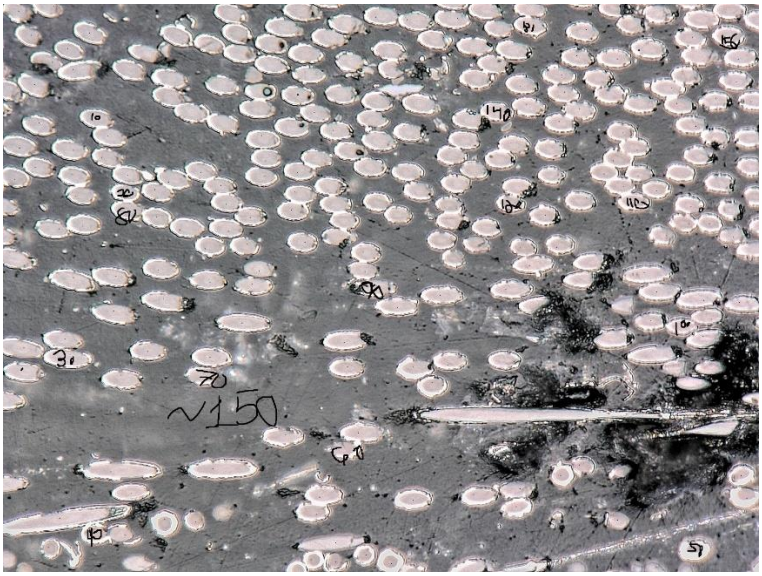


Figure 88: Image 484, 40 wt %, Counted Fibers

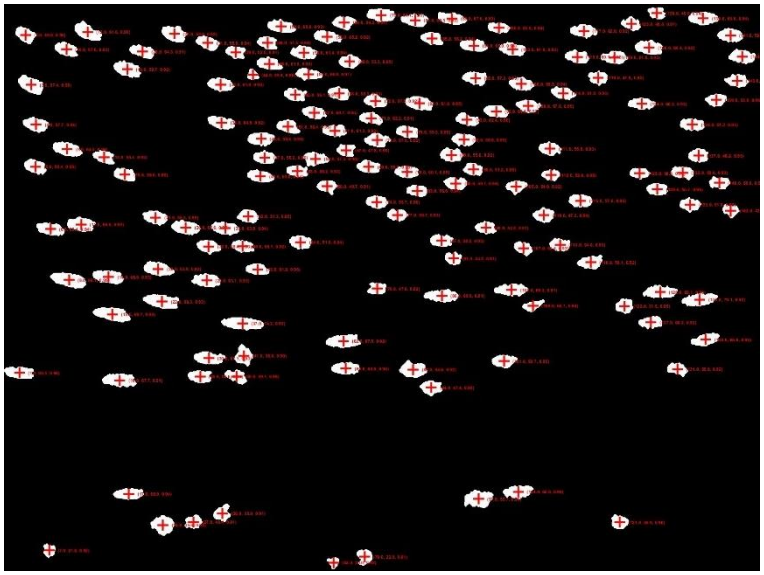


Figure 89: Image 484, 40 wt %, New Code

Figure 87 shows many fibers, but many of them appear to have broken or eroded edges. Figure 88 depicts the same image but with fibers that I would have liked the program to consider. I have almost as many fibers as the program, but still more than Hollinger et al. Hollinger et al. likely saw many of these fibers as incomplete and broken and did not measure them as a result. Many of these fibers that closely overlap with each other, when separated, have pieces missing from the overlap. However, these fibers still passed the solidity filtering implemented in the code. For the 40-weight percent population, where very little data is eliminated, the solidity requirement could possibly be made more stringent. Whatever modification is eventually made, the final binary images must be consulted to make sure that the program is making a fair interpretation of the data; the means could match the study at the cost of eliminating significant quantities of pertinent data.

Chapter 10 : FINAL CONCLUSIONS AND EXTENSIONS OF THE PROJECT

Although the program is currently impressive, it still cannot properly estimate the true means of each population. Future developers should consider implementing machine learning into the program to optimize the use of several different image processing techniques at the same time to properly identify fibers.

Future developers should also consider the speed and efficiency of the program. Much of the past year and a half of development was dedicated to improving the accuracy of the program. There is little discussion dedicated to the speed of the program. For reference, analyzing one population of over 500 images requires approximately 20 minutes. If using the program that compares one set of binary images to another, the program requires upwards of 40 minutes to obtain results (the program creates two sets of binary images, hence doubling the required time). The program requires roughly 5 minutes to analyze a sample set of 30 images, up to 10 minutes if the code for comparing images is run. Another version of the code that does not save images or write to a .csv file was developed; this program analyzed and compared two image processing techniques over an entire population set of more than 500 images in less than 10 minutes. While some of these results are promising, the runtime will only increase as more features are added to the code. Almost no effort was made to optimize the code, so there are many ways to improve. Future projects could also implement multithreading or multiprocessing elements to decrease program runtime for large sets of images.

The image processing code can and should be modified to analyze factors other than fiber angle. Figure 3 shows many variables that could be verified by the program in addition to finding fiber angle. The program could also identify voids of plastic in the dog bone samples as well. Darker or lighter plastic surrounding the fibers may indicate a certain density of plastic. Knowing

factors about not only the fibers but the nylon surrounding them could be used to develop an understanding of how the composite fills the mold. Further, the data could be used in CFD applications like ANSYS Polyflow to predict the average fiber angle and the conductivity of the specimen before the sample is even manufactured. The applications of image processing in quality engineering are truly limitless.

Appendix A: Code from Summer Development

```
%{
Written by Max Myers under Dr. Hollinger
6/20/2022 - Image processing for determining orientation of NiCF fibers
Made for processing the small .png files, used to build .tif analysis
%}
% start with general pre-processing
s4 = imread("S4_9_02.png"); % import image
s4 = s4(:,:,1:3);        % extract only channels 1,2,3
S4gs = im2gray(s4);      % convert to grayscale image
s4adj = imadjust(s4gs);  % adjust for contrast
s4smooth = wiener2(s4adj, 5); % remove noise
S4bw = imbinarize(S4gs);          % convert to binary image
SE = strel("disk", 5);   % filter for opening image
s4open = imopen(s4bw, SE);  % open image, connect dark areas and remove small
bright areas
s4close = imclose(s4open, SE);
S1bw = imfill(s4close, "holes");     % fill holes, may be necessary
% connect objects, eliminate tiny marks, etc.
ellipses = bwpropfilt(S1bw, "Area", [1000 10000]); % only show objects within
this pixel area
% separate overlapping objects
% find properties
props = regionprops(ellipses, 'Area', 'Centroi', 'MajorAxisLength',
'MinorAxisLength');
allAreas = [props.Area];
majorAxisLength = [props.MajorAxisLength];
minorAxisLength = [props.MinorAxisLength];
sum = 0;
% display major / minor axis on each image
figure
imshow(ellipses);
axis('on', 'image');
impixelinfo;
hold on;
centroids = vertcat(props.Centroid);

for i = 1:length(props)
    x = props(i).Centroid(1);
    y = props(i).Centroid(2);
    a = props(i).MajorAxisLength;
    b = props(i).MinorAxisLength;
    plot(x, y, 'r+', 'LineWidth', 2, 'MarkerSize', 15);
    str = sprintf('        (%.1f, %.1f)', a, b);
    text(x, y, str, 'Color', 'r', 'FontSize', 6, 'FontWeight', 'bold');
end

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);

for i = 1:numel(majorAxisLength)
    angle = acosd(minorAxisLength(i) / majorAxisLength(i));
    sum = sum + angle;
end

numel(majorAxisLength)
meanAngle = sum / numel(majorAxisLength)
```

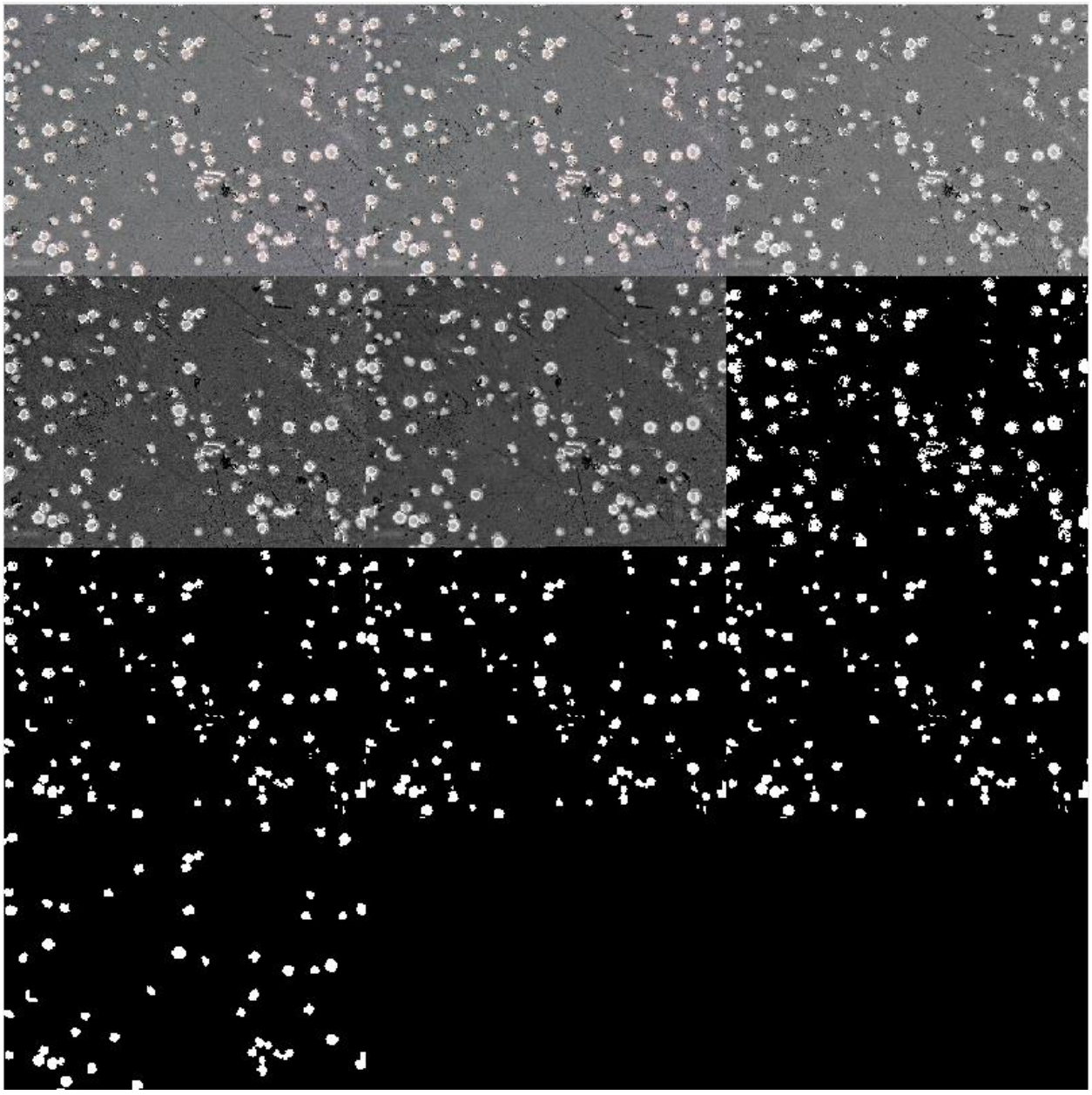


Figure 90: Changes to Image From Each Line of Pre-Processing

Appendix B: Additional Heat Maps

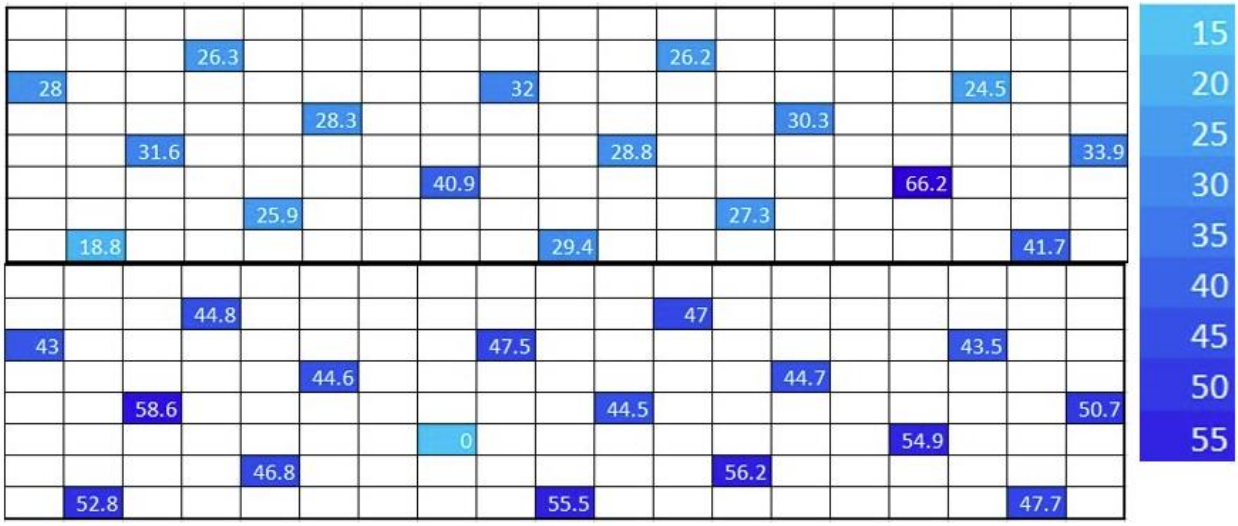


Figure 91: 30 wt% Angle Heat Map

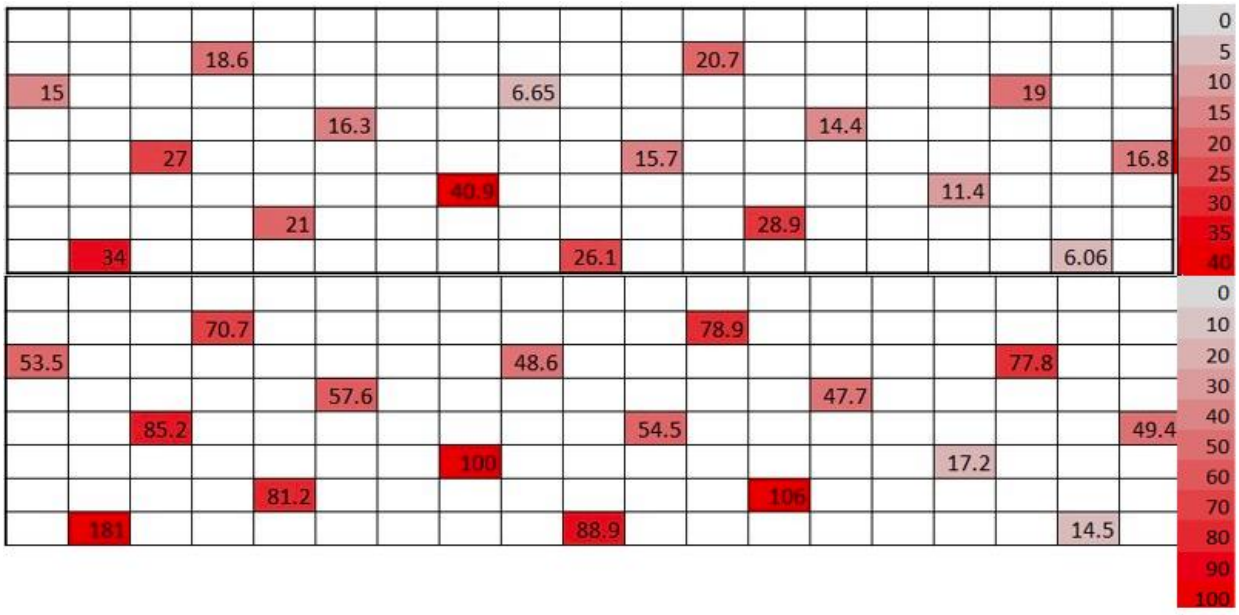


Figure 92: 30 wt% Error Heat Map

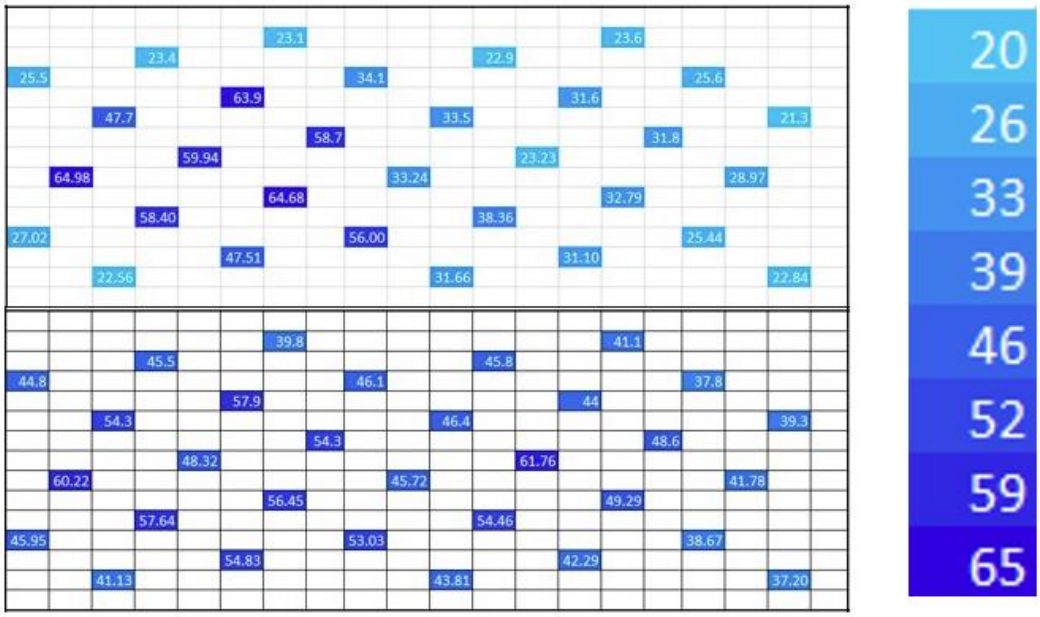


Figure 93: 40 wt% Angle Heat Map

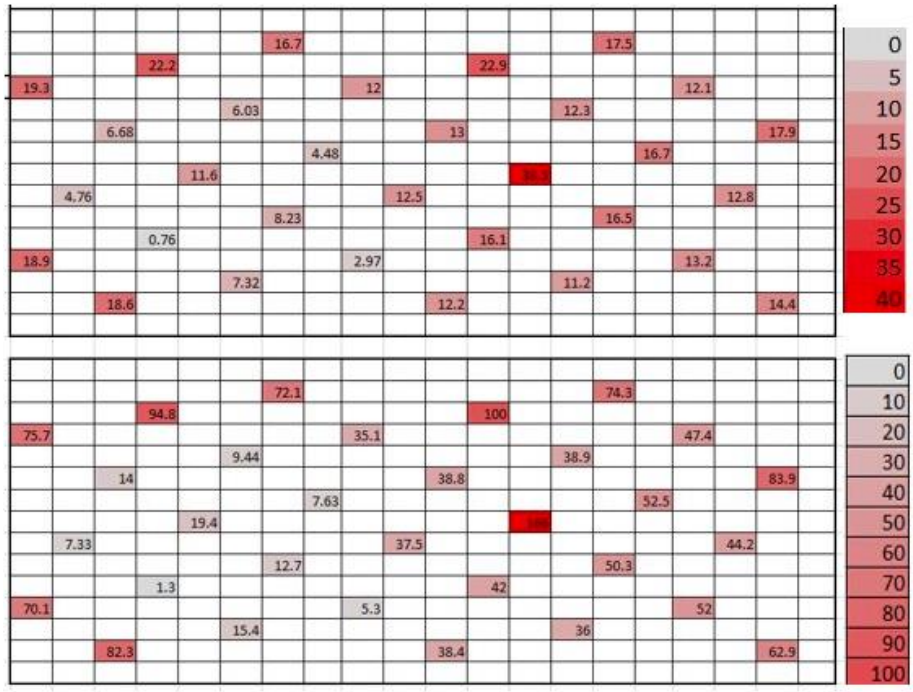


Figure 94: 40 wt% Error Heat Map

Appendix C: Final Code, Comparison of Initial Methods to Final Methods

```
tic

ds = imageDatastore("C:\Users\maxth\OneDrive\Desktop\30wt\20210115_142814 (1)\Images"); % create datastore object we will loop through
dataFilenames = ds.Files;
numImages = numel(dataFilenames)                % find number of images in folder, will use in for loop
sumOG = 0;                                        % declare an empty variable to hold the mean angle of each image
sumNEW = 0;
numFibersOG = 0;                                  % count number of fibers to find
numFibersNEW = 0;

popAngles = []; % an empty array here that all the angles will be appended to, to make a histogram

% open a csv file that we can write data to
%file = 'results_hist_30.csv';
%line = ['Photo', 'element', 'Angle'];
%writematrix(line, file)

% use a for loop to edit every image in the folder, save the edited images
% into another folder, and paste results into a text file

for i = 1:numImages

    [filepath, name, ext] = fileparts(dataFilenames(i));

    % start with reading an image from the datastore and do all general
    % preprocessing

    img = readimage(ds, i);     % read in image
    img = img(:,:,1:3);         % extract only channels 1,2,3 or R, G, B
    imgray = im2gray(img);      % convert to grayscale image
    imgbw = imbinarize(imgray); % convert to binary image (only black or white)

    % edit binary image (remove artifacts, simplify data)(

    imgbw = bwareaopen(imgbw, 200); % open the binary image (connect dark spaces, eliminate small white blobs)
    imgbw = wiener2(imgbw, 5);      % apply a wiener filter to the image (sharpen edges, eliminate connections)
    imgbw = imfill(imgbw, 'holes'); % fills any tiny black blobs within fibers, removes tiny white spaces
% connect objects, eliminate tiny marks, etc.
ellipses = bwpropfilt(imgbw, "Area", [250 10000]); % only show objects within this pixel area. remove more tiny/huge white spaces

% find objects and measure
props = regionprops(ellipses, 'Area', 'Centroid', 'MajorAxisLength', 'MinorAxisLength', 'Solidity'); % find said props of objects in image
%allAreas = [props.Area];                                                    % get areas as array
majorAxisLength = [props.MajorAxisLength];                                  % get major axis lengths as array
minorAxisLength = [props.MinorAxisLength];                                  % get minor axis lengths as array
%Solidity = [props.Solidity];

% loop through each computer identified object in image, measure major
% and minor axis, find angle, save data to edited image and text file

numFibersOG = numFibersOG + length(props);                    % add number of fibers to find population mean later

 % display major / minor axis on each image
%figure;
%imshow(ellipses);
%axis('off', 'image');
%impixelinfo;
%hold on;
%centroids = vertcat(props.Centroid);

for k = 1:length(props)                                     % for loop to calculate angle and to write new images
    %x = props(k).Centroid(1);                               % x centroid on the given ellipse
    %y = props(k).Centroid(2);                               % y centroid on the given ellipse
    a = props(k).MajorAxisLength;                           % major axis length for the given ellipse
    b = props(k).MinorAxisLength;                           % minor axis length for the given ellipse
    angle = acosd(minorAxisLength(k) / majorAxisLength(k)); % calculates the angle of the filament
    %solid = Solidity(k);

    check = isnan(angle); % make sure that no wonky values somehow make it through, this will show up on Excel file when searching for bad readings
    if check == 1
        angle = -1;
    end

    %line = [i, k, angle];
    %writematrix(line, file, 'WriteMode', 'append');

    %popAngles(end+1) = angle;                   % add all angles for the image into the big array for all images
    sumOG = sumOG + angle;                      % adds angle to the cumulative sum to make the average

    %plot(x, y, 'r+', 'LineWidth', 2, 'MarkerSize', 12);                % plots a marker (+) on the pair of centroids
```

```
    %str = sprintf('       (%.1f, %.1f, %.2f)', k, angle, solid);           % makes a string of the element, angle, how circular
    %text(x, y, str, 'Color', 'r', 'FontSize', 5, 'FontWeight', 'bold'); % plots string next to marker

end

% run it back again with the new filtering requirement, take total
img2 = readimage(ds, i);
img2 = img2(:,:,1:3);         % extract only channels 1,2,3 or R, G, B
img2gray = im2gray(img2);      % convert to grayscale image
img2gray = imadjust(img2gray);
im2gbw = imbinarize(img2gray, "adaptive", "Sensitivity", 0.7); % convert to binary image (only black or white)

SE = strel("disk", 4);   % filter for opening image

% edit binary image (remove artifacts, simplify data)(
im2gbw = bwareaopen(im2gbw, 200); % open the binary image (connect dark spaces, eliminate small white blobs)
im2gbw = imerode(im2gbw, SE);
im2gbw = imclearborder(im2gbw);

% eliminate outer shell
for q = 1:3
    shell = bwperim(im2gbw); % get outside perimeter of fibers
    im2gbw(shell) = false; % set those pixels to black
end

im2gbw = imfill(im2gbw, 'holes'); % fill any holes

% connect objects, eliminate tiny marks, etc.
ellipses2 = bwpropfilt(im2gbw, "Area", [100 10000]); % only show objects within this pixel area. remove more tiny/huge white spaces
ellipses2 = wiener2(ellipses2, [2 2]);
ellipses2 = imopen(ellipses2, SE);
ellipses2 = bwpropfilt(ellipses2, "Solidity", [0.9, 1]);
ellipses2 = bwpropfilt(ellipses2, "Area", [200 10000]);
ellipses2 = bwpropfilt(ellipses2, "MinorAxisLength", [20 80]);
props2 = regionprops(ellipses2, 'Area', 'Centroid', 'MajorAxisLength', 'MinorAxisLength', 'Solidity'); % find said props of objects in image
majorAxisLength = [props2.MajorAxisLength];                                                            % get major axis lengths as array
minorAxisLength = [props2.MinorAxisLength];                                                            % get minor axis lengths as array
solidity = [props2.Solidity];

numFibersNEW = numFibersNEW + length(props2);                    % add number of fibers to find population mean later
    for k = 1:length(props2)                                    % for loop to calculate angle and to write new images
        %x = props(k).Centroid(1);                              % x centroid on the given ellipse
        %y = props(k).Centroid(2);                              % y centroid on the given ellipse
        a = props2(k).MajorAxisLength;                          % major axis length for the given ellipse
        b = props2(k).MinorAxisLength;                          % minor axis length for the given ellipse
        angle = acosd(minorAxisLength(k) / majorAxisLength(k)); % calculates the angle of the filament
        %solid = Solidity(k);

        check = isnan(angle); % make sure that no wonky values somehow make it through, this will show up on Excel file when searching for bad readings
        if check == 1
            angle = -1;
        end

        %line = [i, k, angle];
        %writematrix(line, file, 'WriteMode', 'append');

        popAngles(end+1) = angle;                    % add all angles for the image into the big array for all images
        sumNEW = sumNEW + angle;                       % adds angle to the cumulative sum to make the average

        %plot(x, y, 'r+', 'LineWidth', 2, 'MarkerSize', 12);                % plots a marker (+) on the pair of centroids
        %str = sprintf('       (%.1f, %.1f, %.2f)', k, angle, solid);           % makes a string of the element, angle, how circular
        %text(x, y, str, 'Color', 'r', 'FontSize', 5, 'FontWeight', 'bold'); % plots string next to marker

    end


    %set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);   % sets properties on the image
    name = name + "MOD"
    %FILENAME = string(strcat("C:\Users\maxth\OneDrive\Desktop\Schreyer Research\over 50\end", name, '.jpeg'));
    %saveas(gcf, FILENAME);    % write edited image to Folder
    %close gcf % close out of image so 533 tabs don't open immediately and crash PC

end

%edges = [0 5: 5 10: 10 15: 15 20: 20 25: 25 30: 30 35: 35 40: 40 45: 45 50: 50 55: 55 60: 60 65: 65 70: 70 75: 75 80: 80 85: 85 90]
%h = histogram(popAngles, edges)

populationOG = numFibersOG
populationNEW = numFibersNEW
populationMeanOG = sumOG/numFibersOG % find mean of all images, display to terminal
populationMeanNEW = sumNEW/numFibersNEW
used = numFibersNEW/numFibersOG;
ignored = (1 - used)*100
```

```
        angle = acosd(minorAxisLength(k) / majorAxisLength(k)); % calculates the angle of the filament
        %solid = Solidity(k);

        check = isnan(angle); % make sure that no wonky values somehow make it through, this will show up on Excel file when searching for bad readings
        if check == 1
            angle = -1;
        end

        %line = [i, k, angle];
        %writematrix(line, file, 'WriteMode', 'append');

        popAngles(end+1) = angle;                   % add all angles for the image into the big array for all images
        sumNEW = sumNEW + angle;                      % adds angle to the cumulative sum to make the average

        %plot(x, y, 'r+', 'LineWidth', 2, 'MarkerSize', 12);                % plots a marker (+) on the pair of centroids
        %str = sprintf('       (%.1f, %.1f, %.2f)', k, angle, solid);            % makes a string of the element, angle, how circular
        %text(x, y, str, 'Color', 'r', 'FontSize', 5, 'FontWeight', 'bold'); % plots string next to marker

    end


    %set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]);   % sets properties on the image
    name = name + "MOD"
    %FILENAME = string(strcat("C:\Users\maxth\OneDrive\Desktop\Schreyer Research\over 50\end", name, '.jpeg'));
    %saveas(gcf, FILENAME);    % write edited image to Folder
    %close gcf % close out of image so 533 tabs don't open immediately and crash PC

end

%edges = [0 5: 5 10: 10 15: 15 20: 20 25: 25 30: 30 35: 35 40: 40 45: 45 50: 50 55: 55 60: 60 65: 65 70: 70 75: 75 80: 80 85: 85 90]
%h = histogram(popAngles, edges)

populationOG = numFibersOG
populationNEW = numFibersNEW
populationMeanOG = sumOG/numFibersOG % find mean of all images, display to terminal
populationMeanNEW = sumNEW/numFibersNEW
used = numFibersNEW/numFibersOG;
ignored = (1 - used)*100
edges = [0 5: 5 10: 10 15: 15 20: 20 25: 25 30: 30 35: 35 40: 40 45: 45 50: 50 55: 55 60: 60 65: 65 70: 70 75: 75 80: 80 85: 85 90];
h = histogram(popAngles, edges)

toc
```

References

[1] F. C. &. H. E. Association, "Fuel Cell Basics," Fuel Cell & Hydrogen Energy Association, 2022. [Online]. Available: https://www.fchea.org/fuelcells. [Accessed 29 May 2023].

[2] New Age Metals Inc, "PGM-Based Fuel Cells: Applications for Industry," New Age Metals Inc, 2021. [Online]. Available: https://newagemetals.com/pgm-based-fuel-cells-applications-for-industry/. [Accessed 29 May 2023].

[3] DOE Staff, "Hydrogen Fuel Cells," November 2008. [Online]. Available: https://www.nrc.gov/docs/ML1002/ML100280723.pdf. [Accessed 29 May 2023].

[4] DOE Staff, "Hydrogen and Fuel Cell Technologies Office," Department of Energy, 2022. [Online]. Available: https://www.energy.gov/eere/fuelcells/hydrogen-and-fuel-cell-technologies-office. [Accessed 29 May 2023].

[5] R. Zameroski, C. Krypta, B. Young, S. Sanei and A. Hollinger, "Mechanical and electrical properties of injection-molded mwcnt-reinforced polyamide 66 hybrid composites," *Journal of Composites Science*, vol. 177, p. 4, 2020.

[6] C. Krypta, B. Young, A. Santamaria and A. Hollinger, "Multiwalled Carbon Nanotube-Filled Polymer Composites for Direct Injection Molding of Bipolar Plates," *ECS Transactions*, pp. 109,199, 2022.

[7] Texas Injection Molding Staff, "Plastic Injection Molding With Nylon (Polyamide)," Texas Injection Molding, 2023. [Online]. Available: https://texasinjectionmolding.com/nylon-pa-injection-molding/#:~:text=Nylon%20is%20easy%20to%20process,that%20demonstrate%20many%20attractive%20properties.. [Accessed 29 May 2023].

[8] "Four-terminal sensing," Wikipedia, 13 December 2019. [Online]. Available: https://en.wikipedia.org/wiki/Four-terminal_sensing. [Accessed 29 May 2023].

[9] M. Weber and M. Kamal, "Estimation of the volume resistivity of electrically conductive composites," *Polymer composites*, pp. 711-725, 1997.

[10] Teijin Carbon Europe GmbH, "Teijin Carbon Fiber Business," [Online]. Available: https://www.teijincarbon.com/products/short-fibers/chopped-fibers. [Accessed 5 July 2023].

[11] carattinim, "Classifying Conic Sections Notes," Geogebra, 23 May 2017. [Online]. Available: https://www.geogebra.org/m/tavYVNth. [Accessed 29 May 2023].

[12] Saylor Academy, "Ellipses," Saylor Academey, 2012. [Online]. Available: https://saylordotorg.github.io/text_intermediate-algebra/s11-03-ellipses.html. [Accessed 29 May 2023].

[13] S. Patel, "Best Image Processing Tools used in Machine Learning," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/general/212054. [Accessed 29 May 2023].

[14] The Mathworks, Inc, "Image Processing Toolbox," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/index.html?s_tid=hc_product_card. [Accessed 29 May 2023].

[15] The Mathworks, Inc., "Image Processing Onramp," Mathworks, 2023. [Online]. Available: https://matlabacademy.mathworks.com/details/image-processing-onramp/imageprocessing. [Accessed 29 May 2023].

[16] The Mathworks, Inc., "Image Filtering and Enhancement," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/image-types-in-the-toolbox.html. [Accessed 29 May 2023].

[17] The Mathworks, Inc., "im2gray," Mathwokrs, 2023. [Online]. Available: https://www.mathworks.com/help/matlab/ref/im2gray.html. [Accessed 29 May 2023].

[18] The Mathworks, Inc., "wiener2," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/wiener2.html#d124e325808. [Accessed 29 May 2023].

[19] The Mathworks, Inc., "imbinarize," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-method. [Accessed 29 May 2023].

[20] The Mathworks, Inc., "strel," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/strel.html#d124e304921. [Accessed 29 May 2023].

[21] The Mathworks, Inc., "imclose," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/imclose.html. [Accessed 29 May 2023].

[22] R. Fisher, S. Perkins, A. Walker and E. Wolfart, "Erosion," W3C Markup Validation Service, 2003. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm. [Accessed 29 May 2023].

[23] R. Fisher, S. Perkins, A. Walker and E. Wolfart, "Dilation," W3C Markup Validation Service, 2003. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm. [Accessed 29 May 2023].

[24] The Mathworks, Inc., "imclose," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/imclose.html. [Accessed 29 May 2023].

[25] The Mathworks, Inc., "imfill," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/imfill.html. [Accessed 29 May 2023].

[26] The Mathworks, Inc., "bwpropfilt," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/bwpropfilt.html. [Accessed 29 May 2023].

[27] The Mathworks, Inc., "regionprops," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/regionprops.html. [Accessed 29 May 2023].

[28] The Mathworks, Inc., "imread," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/matlab/ref/imread.html. [Accessed 29 May 2023].

[29] The Mathworks, Inc., "readall," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/matlab/ref/matlab.io.datastore.readall.html. [Accessed 29 May 2023].

[30] The Mathworks, Inc., "readimage," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/matlab/ref/matlab.io.datastore.imagedatastore.readimage.html. [Accessed 29 May 2023].

[31] W. W. LaMorte, "Central Limit Theorem," Boston University School of Public Health, 24 July 2017. [Online]. Available: https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html#:~:text=The%20central%20limit%20theorem%20states,will%20be%20approximately%20normally%20distributed.. [Accessed 29 May 2023].

[32] A. Ganti, "Central Limit Theorem (CLT): Definition and Key Characteristics," Investopedia, 10 March 2023. [Online]. Available:

https://www.investopedia.com/terms/c/central_limit_theorem.asp#:~:text=Key%20Takeaways,for%20the%20CLT%20to%20hold.. [Accessed 29 May 2023].

[33] M. Sloggatt, "The Elegant Ellipse," Kaz Roadshow, 30 March 2012. [Online]. Available: https://www.thisiscarpentry.com/2012/03/30/the-elegant-ellipse/. [Accessed 29 May 2023].

[34] The Mathworks, Inc., "solidity concept in labeled components," Mathorks, 14 August 2012. [Online]. Available: https://www.mathworks.com/matlabcentral/answers/45999-solidity-concept-in-labeled-components. [Accessed 29 May 2023].

[35] The Mathworks, Inc., "adaptthresh," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/adaptthresh.html#namevaluepairarguments. [Accessed 29 May 2023].

[36] The Mathworks, Inc., "imerode," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/imerode.html. [Accessed 29 May 2023].

[37] The Mathworks, Inc., "bwperim," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/bwperim.html. [Accessed 29 May 2023].

[38] The Mathworks, Inc., "Is it possible to erode only the outermost layer of a circle?," Mathworks, 31 May 2022. [Online]. Available: https://www.mathworks.com/matlabcentral/answers/1730775-is-it-possible-to-erode-only-the-outermost-layer-of-a-circle. [Accessed 29 May 2023].

[39] The Mathworks, Inc., "imclearborder," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/imclearborder.html. [Accessed 29 May 2023].

[40] The Mathworks, Inc., "imsharpen," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/imsharpen.html. [Accessed 29 May 2023].

[41] The Mathworks, Inc., "bwareaopen," Mathworks, 2023. [Online]. Available: https://www.mathworks.com/help/images/ref/bwareaopen.html. [Accessed 29 May 2023].

ACADEMIC VITA

Summary:

Mechanical engineering Schreyer Honors Scholar applying for full-time position while seeking opportunities for learning and continued growth.

Education and Academic Honors:

Mechanical Engineering Bachelor of Science, French and Math minors - Graduating 8/2023
Pennsylvania State University: The Behrend College

- Schreyer Honors College, Fall of 2020
- Tau Beta Pi, Fall of 2022

Engineering Experience:

Senior Design Project: Tennis Prosthesis (September 2022 - April 2023)

- Designing and manufacturing a body-powered prosthesis to help serve in tennis matches

Schreyer Honors College Undergraduate Thesis (May 2022 - August 2023)

- Image processing in MATLAB to improve manufacturing of hydrogen fuel cells

Wabtec Corporation Co-op (August - November 2022, January - April 2023)

- Working in safety and compliance for marine diesel engines
- Internally audited documents required for type approvals

Wabtec Corporation Internship (May - July 2022)

- Created an Excel VBA tool to predict the cost of a project during the tender phase
- Communicated with managers of product lines to revise work breakdown structures

Behrend Undergraduate Computer Engineering Research (May - August 2021)

- $4500 to independently develop an IoT system that quickly locates faults in power grids

Lemelson-MIT InvenTeam Grant Recipient (2018)

- Received $10,000 to design a bluetooth stethoscope and self-cleaning carrying case for clinical settings

Work Experience:

Tutor at Behrend Learning Resource Center (2020-present)

- Tutor math, physics, computer science, engineering, and French

Engineering Dynamics Teaching Assistant (2021)

- Fairly award partial credit, help students understand errors, analyze student performance

Lifeguard (2015-2022, seasonal)

- Red Cross Certified Lifeguard, ensured safety of patrons and taught swim lessons

Skills and Languages:

CAD Modeling/Simulation

- FDM/SLA 3D printing, Autodesk Inventor, ANSYS Mechanical and Fluent, GRANTA, ImageJ, SolidWorks

Programming Languages/Computer Skills

- MATLAB, C++, Python, VBA, Microsoft Office, Google Suite,Ubuntu/Debian Linux

Languages

- English (native), French (conversant)