**Project WebHunter**

Final Report

Spring 2017

Christopher McFall

Delaram Nikooei

John Plaschke

William Dew

**Table of Contents**

Abstract…………………………………………………………………………………………....3

Company Background…………………………………………………………………………….4

Project Overview…………………………………………………………………………….........5

Objectives…………………………………………………………………………………………8

Objective 1

LineHunter…………....………………………………………………………..…...……..9

Objective 2

Fiji………………………………………………………………………………………..13

Objective 3

Review of Previous Work……………………………………………………………......14

Objective 4 and 5

Edge Detection Background………………………………....…………………………..14

WebHunter Report…………………………………………………………………………....….17

LineHunter Recommendations……………………………………………………………….….22

Droplet Detection Methods……………………………………………………………………....22

Template Matching……………………………………………………………………………....23

Template Matching Recommendations………………………………………………...…....…...26

Droplet Detection Using Circular Regression ……………………………… ………...…….….27

LineHunter and WebHunter Recommendations………………………………………………....32

Machine Learning…….……………………………………………………………………....….33

Possible Bioengineered Solutions for the Future…………………………..…………………….37

Conclusion………………………………………………………………………………..……...38

Appendix I…………………………………………………………………………………….....39

References……………………………………………………………………………………….40

**Abstract**

The Dugway Proving Ground (DPG) is the nation’s center for Chemical and Biological Defense testing, which provides testing for counter chemical, biological, radiological, and explosives (CBRE) hazards. The Aerosol Technology Branch (ATB) of DPG has the mission to test biological and chemical aerosol weapons. Spider webs have recently replaced Goldberg drums to study the aerosols. The spider webs allow the aerosols to be suspended which allows to study how environmental conditions affect the potential threat to populations, i.e. what conditions increase or decrease the threat. From this research, countermeasures and threat assessment models can be developed. For the past decade, a goal was established to increase the efficiency and decrease the cost of aerosol testing (United States Army Dugway Proving Grounds, n.d.). Currently, scanning electron microscopy (SEM) is used to analyze aerosol distribution which is efficient, but time consuming manual measurements of the micrographs have established the need to automate the expensive process. For over two years, students in BTMN 670 (Capstone in Biotechnology) from the University of Maryland University College (UMUC) have been working with ImageJ to automate the characterization of the 3D aerosol cloud. The members of this team have identified a novel approach to increase the accuracy of measurements and speed of processing the spider web micrographs. Previous work focused on using macros and the ImageJ software, even though the results were encouraging the accuracy was not very high.

This paper will describe the four different research areas that the Spring 2017 capstone group pursued to solve the objectives of the Dugway Proving Ground project.

1. A novel ImageJ plugin approach instead of using a macro based approach as other teams did. 100% accuracy for line detection was achieved but the droplet detection only averaged 75% accuracy.
2. A Template matching approach was investigated and 100% droplet detection was achieved. However, some manual cropping of the image was needed to remove false positives detected in the annotation section of the micrograph.
3. Machine learning techniques were researched to determine if they could be applied to achieve the objectives of the capstone project.
4. Bioengineered material as an alternative material to capture aerosol droplets was also investigated.

**Company Background**

The Dugway Proving Ground (DPG) is an army research and development; engineering command. DPG is the nation’s center for Chemical and Biological Defense testing. President Franklin D. Roosevelt in 1942 signed over 126,720 acres in Utah about two hours west of Salt Lake City to be used as a testing facility for testing counter chemical, biological, radiological, and explosives (CBRE) hazards. DPG was deactivated for a short period of time due to World War II and following the Korean War it became a permanent installation. DPG has top of the line laboratories and experts which conduct chemical and biological protection, detection, contamination and decontamination testing by developing outdoor field testing and laboratory and chamber testing using chemical and biological simulants and agents. They receive most of their funding through federal agencies (United States Army Dugway Proving Ground, n.d.).

The Life Sciences Division at DPG encompasses several key research groups, including the Aerosol Technology Branch (ATB), which devotes considerable attention and efforts towards studies involving aerosolized transmission of biological pathogens. ATB operates the largest biosafety level-3 test facility in the United States (Kirschner, 2017). One of the ATB’s projects is to measure the effects of environmental conditions on aerosolized test materials by suspending aerosols of interest on spider web suspended on spindles. ATB uses Tarantula webs, which are used to suspend aerosol particles. This method mimics the behavior of the particles suspended in air (United States Army Dugway Proving Ground, n.d.).

According to Dr. Brian Bennett of the ATB branch, scientists at DPG have limited understanding of how biological warfare agents behave once airborne and released in an attack. They would like to understand how a detector interacts with a biological agent when subjected to climate change (Soldiers.2017. Retrieved from: <http://soldiers.dodlive.mil/2015/04/spider-webs-on-front-lines-of-bio-warfare-detection/>).

**Project Overview**

Project WebHunter was conceived by Dugway Proving Ground’s Aerosol Technology Branch to evaluate and continue research associated with its current Spiderweb-based aerosol capture technologies (Nichols, 2017). Specifically, applications are conducted by creating an aerosol vector containing a given concentration of pathogen and then introducing it to spider web fibers in an attempt to generate a representational 3D cloud comprised of aerosol droplets suspended within the web fibers (Nichols, 2017). This material can then be fixed, embedded, sectioned, and subsequently analyzed using scanning electron microscopy (Nichols, 2017).

The Scanning Electron Microscope (SEM) has been widely used as a 2D imaging instrument to determine the surface characteristics of microscopic objects. To completely measure and determine the surface characteristics, we need to be able to see the 3D image, which is not possible with the SEM microscopes. The SEM micrographs are 2D images. 3D images allow us to see the anatomic shape of the aerosols, which in turn provides us with the quantitative measurements, and informative characterization of the aerosol of interest (Retrieved from: https://petebankhead.gitbooks.io/imagej-intro/content/chapters/microscope\_types/microscope\_types.html).

Obtaining 3D surface reconstruction from a set of 2D images requires the use of numerous computational techniques, such as computer vision, machine learning, multi view geometry, and strategies to overcome problems transitioning from 2D images to 3D models (Figure 1) (Tafti et al., 2015).

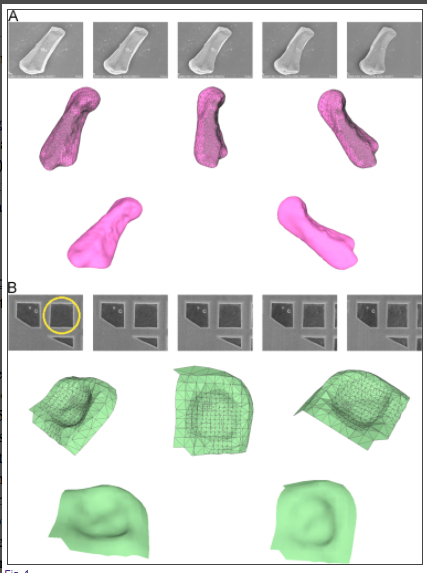


Figure 1. 2D and 3D images

Retrieved from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4685174/

The resulting SEM images are taken at differing magnifications and reflect different chemical and environmental treatment conditions that have been applied to the aerosolized spider webs during testing. Hence, the SEM images provided for Project WebHunter can be highly variable from one image to the next, making image analysis of droplet and web attributes technically challenging. The process of analyzing the resulting SEM micrographs has been accomplished using ImageJ, which is the software image analysis tool used by Project WebHunter (Nichols, 2017). ImageJ is best described as an open-source, Java-based image processing platform developed by the National Institutes of Health (NIH) (ImageJ Features, n.d.). Since ImageJ is Java-based, it can be readily run on Windows, Linux, and Macintosh operating systems (ImageJ Features, n.d.). ImageJ also benefits from its adaptability as well as having a community of open-source developers that exchange source code and methods used to develop plugins and macros (ImageJ Features, n.d.). Computer programming and methods development for an optimized imaging strategy is an ongoing pursuit for Project WebHunter.

Overall, successful outcomes for this project will likely necessitate an image analysis workflow that is capable of generating quantitative data for both of the aerosolized droplets attached to the individual web fibers as well as measurements for the thickness and lengths of the web fibers by themselves. In retrospect, the limitations shown by previous groups’ attempts towards characterizing 3D clouds by attempting to alter existing ImageJ plugins has clearly demonstrated the need for an application-specific imaging plugin tool.

**Objectives**

The initial objectives were slightly modified to match what our team did, though they were kept inline with the overall requirements.

1. Familiarization with ImageJ & LineHunter

* Updated a 2016 macro to save results to a csv file

2. Familiarization with Fiji

* Our team determined that ImageJ was sufficient and only used ImageJ

3. Research the Work on WebHunter

* The previous group of students determined the direction to go with WebHunter
* Read the final report and the Final Powerpoint as a starting point to make WebHunter

4. Use ImageJ to Perform Fiber Analysis of Web

* Desired Function of WebHunter: collection of number of fibers and their respective thickness/diameter measurements
* Previous classes used the BoneJ suite for Thickness function. Our team decided to not use BoneJ as it is “overkill” and does not work with Java8.

5. Use ImageJ to Perform Droplet Analysis

* Desired Function of WebHunter:
  + Count Droplets
  + Measure Droplets
  + Indicate relationship of droplet to web fiber by location

Nichols, Kym (2017)(BTMN 670)

ImageJ was downloaded from the following URL:<http://rsbweb.nih.gov/ij/>.

**Objective 1 - LineHunter**

ImageJ is a general image processing tool and functions as a base development platform for the LineHutner program (ImageJ, n.d). Our project group downloaded and installed LineHunter according to the LineHunter Program User-Manual. Afterwards we began testing individual images in Semi-Auto mode as well as image sets in Automated Mode. Figure 2, details a comparison between an aerosolized web SEM when operated in Semi-Auto mode to the same analysis performed on one of the LineHunter sample images.

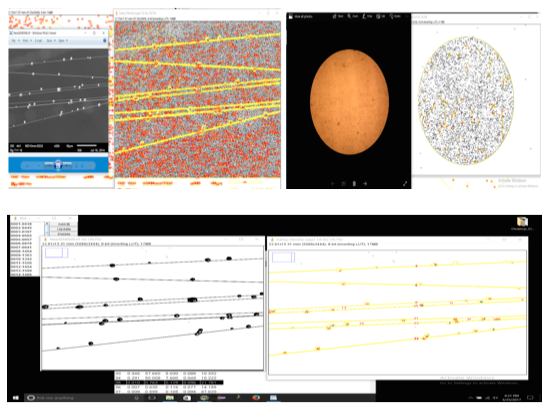
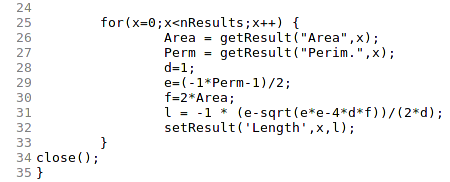


Figure 2. LineHunter Analysis of SEM micrographs vs LineHunter Sample Images

The second image from the top left of Figure 2 is a visual output showing the detection of 1914 ROIs in the unprocessed aerosolized web SEM micrograph (first image from top left) vs ROIs found in the sample image (first and second images top right). The bottom image in Figure 2 represents the aerosolized web SEM micrograph converted to binary format prior to Semi-Automated Operation Mode. 61 ROIs were detected for the formatted binary image, which indicated that LineHunter may not be the most suitable program for detecting aerosolized spider web fibers as shown in the SEM images used for Project WebHunter.

2016 Team WebHunter performed extensive testing of the LineHunter program in Automated mode (Belleman et al., 2016). The first area of investigation for this team involved fixing the line analysis output, as the program was detecting more lines for the images than was shown in the LineHunter manual (Belleman et al., 2016). The second problem that they encountered was a compiling effect for the Excel results for each image file (Belleman et al., 2016). Concerning the first problem, in their final report they demonstrate the corrected line analysis output as evidenced within the “Automated\_Linehunter\_Fall2016.ijm” run script (Belleman et al., 2016). For the second problem they provided a partial fix that compiles one .csv output file that is representative of all of the lines detected for a given series of images (Belleman et al., 2016). Specifically, in this output file measurements are given for Area, Perimeter, Major, Minor, and Angle (Belleman et al., 2016). 

Our group evaluated the Automated\_Linehunter\_Fall2016.ijm macro on several different sample image sets, which worked fine with the exception of not outputting the line lengths (Belleman et al., 2016). Instead, Section 4.2 of the LineHunter Manual accounts for this discrepancy with instructions for manually calculating the line lengths post-run within the Excel .csv that the LineHunter program compiles. Ideally, the approach in Section 4.2 should be eliminated altogether, as the LineHunter program can be adapted to process the complete data set by itself. Since the previous group was unable to implement the code needed to complete the data analysis for the line lengths, our group decided to attempt to rectify this problem and managed to do so successfully. With this in mind, we will be submitting a revised “Auto-Line-Hunter\_TEAM\_WEBHUNTER\_2017\_finis.ijm” file along with the remainder of our deliverables. This file should complete the LineHunter program, with results in the Imagej ‘Results’ window as well as within the outputted Excel .csv file. Both results sections now contain the line length measurements in addition to the line height measurements, so there is no longer any need for any further calculations to be performed in Excel. In the interest of providing clarity to this report, we have detailed the coding algorithm that was used to finalize the LineHunter program in Figure 3. The lower portion of Figure 3 demonstrates the corrected sample output with the line lengths included in the data outputted in the Excel .csv file. Furthermore, the code in Figure 3 was adapted/modified from Project WebHunter 2016 and incorporated into the modified LineHunter “Auto-Line-Hunter.ijm” as shown in Lines 56- 66 (Belleman et al., 2016). Based on our evaluation, the LineHunter program’s strength resides in it’s ability to operate in Automated mode and perform linear analysis for a series of images all at once and with a high degree of accuracy. Since our group was not working with ‘serial’ sectioned SEM images, or images that represented a specifically defined image series, it served as one of the driving factors to move forward with continued work on our own java-based plugin. This was further compounded by reviews of initial research and frustrating examinations of non-working macros provided by previous WebHunter teams. The key motivation behind our workings was to attempt to fully address the problems and objectives as they were given to us, which we felt could only be done via the use of Java-based plugins as opposed to building upon the work of previous project teams. This subject is covered in greater detail in “Objective 3 - Review of Previous Work.”

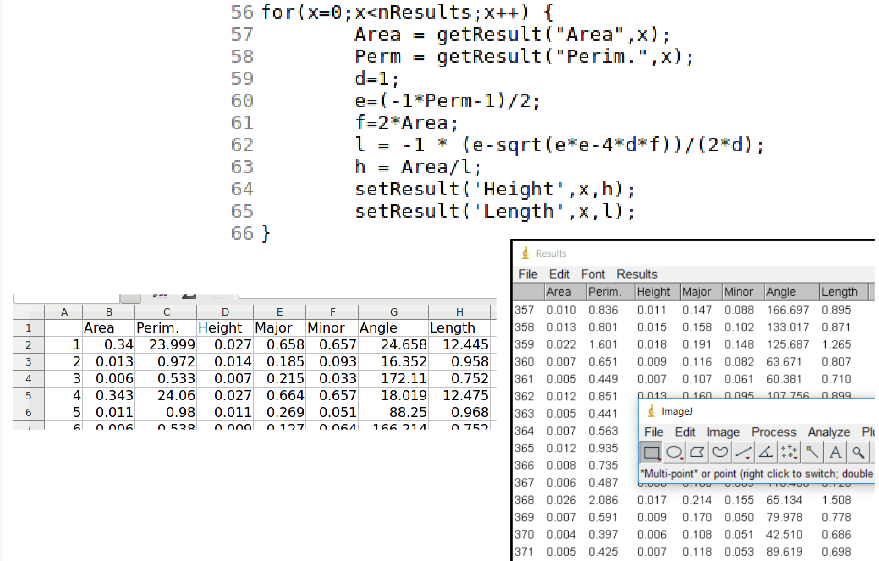


Figure 3. LineHunter data output from Automated mode using Auto-Line-Hunter\_TEAM\_WEBHUNTER\_2017\_finis.ijm

**Objective 2 - Fiji**

Fiji is just ImageJ with additional packages that concentrates on life sciences. Fiji has a 3D viewer which helps to acquire information through light microscopy resulting in image processing algorithms and it also adds power and flexibility. Fiji is very easy to use and install but we decided that Fiji was not needed to be used in our project. It has been very difficult to comprehend what the previous group has done. They have used macros, which do not fit the problem. A lot of macro parameter need to be set which can become too complex. Often one group of settings for macro parameters may not work for another set of images therefore we have taken a new approach in solving the problem and to improve the program.

Our team determined that Fiji was not needed and that ImageJ was sufficient to detect line and circle features in the micrograph and determine their spatial relationships

**Objective 3 - Review of Previous Work**

Objective 3 was to study what previous classes did and then extend and improve their macros. As a team, we decided to implement a completely different methods than previous classes because of inherent problems with macros. Previous teams used macros in an attempt to detect spindles, droplets and their spatial relationship. Given that macros do not allow access to coordinates and they are more “blackbox” than “whitebox”, it is difficult to meet the objective of providing spatial relationships. Our team developed a novel approach using edge detection and linear regression. Additionally, we used a priori knowledge of typical micrographs, e.g. the typical droplet and spindle sizes in micron units. Previous classes output size information in pixels. Because our team parsed the scale information we are able to provide spatial information in micron units. These unique approaches are described in detail in the following sections of this paper.

**Objective 4 and 5 - Edge Detection Background**

Edges are a change in local intensity of an image that is highly significant or dramatic. Edges usually happen between two different regions of an image. These intensity changes are caused by object boundary, surface boundary, shadows, and reflections. When an edge detection algorithm is applied to an image a drawing is produced of the scene where comers, lines and curves are determined (Trucco).

When looking for an edge detector algorithm there are certain steps that need to be applied. You need to smooth the image and suppress as much noise as possible, without destroying the true edges. Then you need to enhance the quality of the edges with a filter. Works like sharpening the image. The third step is to determine which edges are true edges and which are just noise and can be discarded. Thresholding provides the criterion used for detection. Localization is the last step. We determine the exact location of an edge sub-pixel resolution might be required for some of the applications. Sub-pixel is estimating the location of an edge by spacing between pixels. You also thin the edges and then link the edges (Trucco).

What makes a good edge detector algorithm? An edge detector algorithm minimizes the probability of false positives like spurious edges caused by noise. It also need to minimize false negatives so that a real edge is not missed. The edges that are detected must be close to the true edges. The last thing that a good edge detector algorithm has is a single response constraint. This allows only one point for each true edge point and minimizes the number of local maxima around true edges (Trucco).

The most common edge detector algorithm is the Canny edge detector. It was developed by John F. Canny in 1986. It uses the Gaussian filter to filter out the noise then finds the intensity gradient of the image. Non-maximum suppression is applied to remove pixels that are not part of the edge. The last step separates everything into two thresholds lower and upper. If a pixel gradient is higher that the upper threshold, the pixel is accepted as an edge. If the pixel gradient is below the lower threshold then the edge is rejected. If the pixel gradient is between the two thresholds then it will be accepted if it is connected to a pixel that is above the upper threshold (Trucco).

Our algorithm to detect edges was simple. First a thresholding of the whole image was done. Next transitions from background to edge pixel value was used to detect the edge. The unique aspect of our approach is to cluster these points and then linear regression to determine the linear equations which allows simple equations to be used to calculate spatial characteristics of the spindles.

The line function was the first to be designed and coded. The first attempt used a delta in pixel values (essentially gradient) to detect the edges of the lines, i.e. the difference between the upper pixel and lower pixel is an edge if the change (delta) is greater than a threshold. The basic algorithm for line detection is shown in pseudo code below:

For x = 0 to the width of the micrograph

For y = 0 to the height (excluding annotation) of the micrograph.

State Machine

State = Find top edge:

1) Find top edge which was determined by the change in pixel

values of current y and (y-4). If top edge found look for bottom

edge.

State = find bottom edge

2) Find bottom edge. If bottom edge found then calculate thickness.

State = calculate thickness

3) If thickness with in line thickness limits add point to Lines data structure.

Once points are collected perform linear regression to find lines.

Improvements to the code presented at the midterm status have been completed. Our team successfully captured all lines accurately for two micrographs (SEM506 and newSEM506 and newSEM501 after bugs were fixed in the previous version of the LineHunter code. Two conditions must be met for the algorithm to work well. Firstly, A startingX must be chosen such that all lines intersect the vertical line at the startingX value. Secondly, after the initial point is added adding new points to the correct line is essential. An improvement to the previous algorithm as made to calculate the initial slope from the first two points. This allows the subsequent points to be added based on the determining the closest line to a new point, i.e. it is essentially clustering the points into the most likely group of lines.

Linear equations for all the spindles are calculated and from these equations various spatial statistics are calculated. The following is a report that is created by the code automatically in HTML format. Please refer to the reports directory in our deliverables to see the reports within a browser.

# 

# **WebHunter Report**

## **Parameters**

|  |  |
| --- | --- |
| **Parameter Name** | **Value** |
| **threshold pixel value** | **135** |
| **startingX** | **130** |
| **lineSep** | **5** |
| **xInc** | **10** |
| **max droplet diameter** | **2** |
| **max droplet diameter** | **4** |
| **max spindle thickness** | **0.8** |

## 

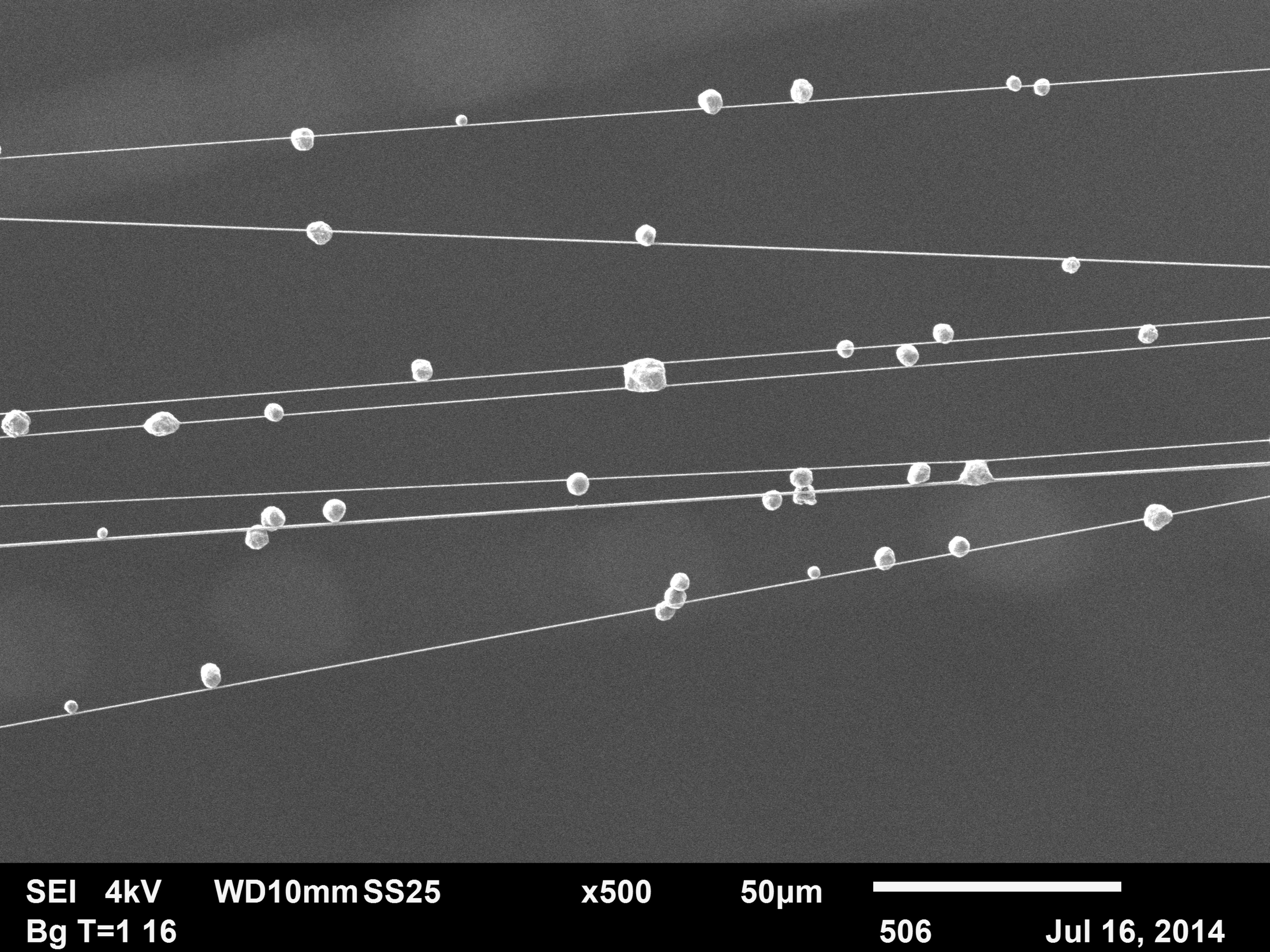
## **Original Image**

Micrograph width: 256.256µm

Micrograph height: 192.192µm

Pixels per micron: 0.050

Micrograph area: 49250.452µm²

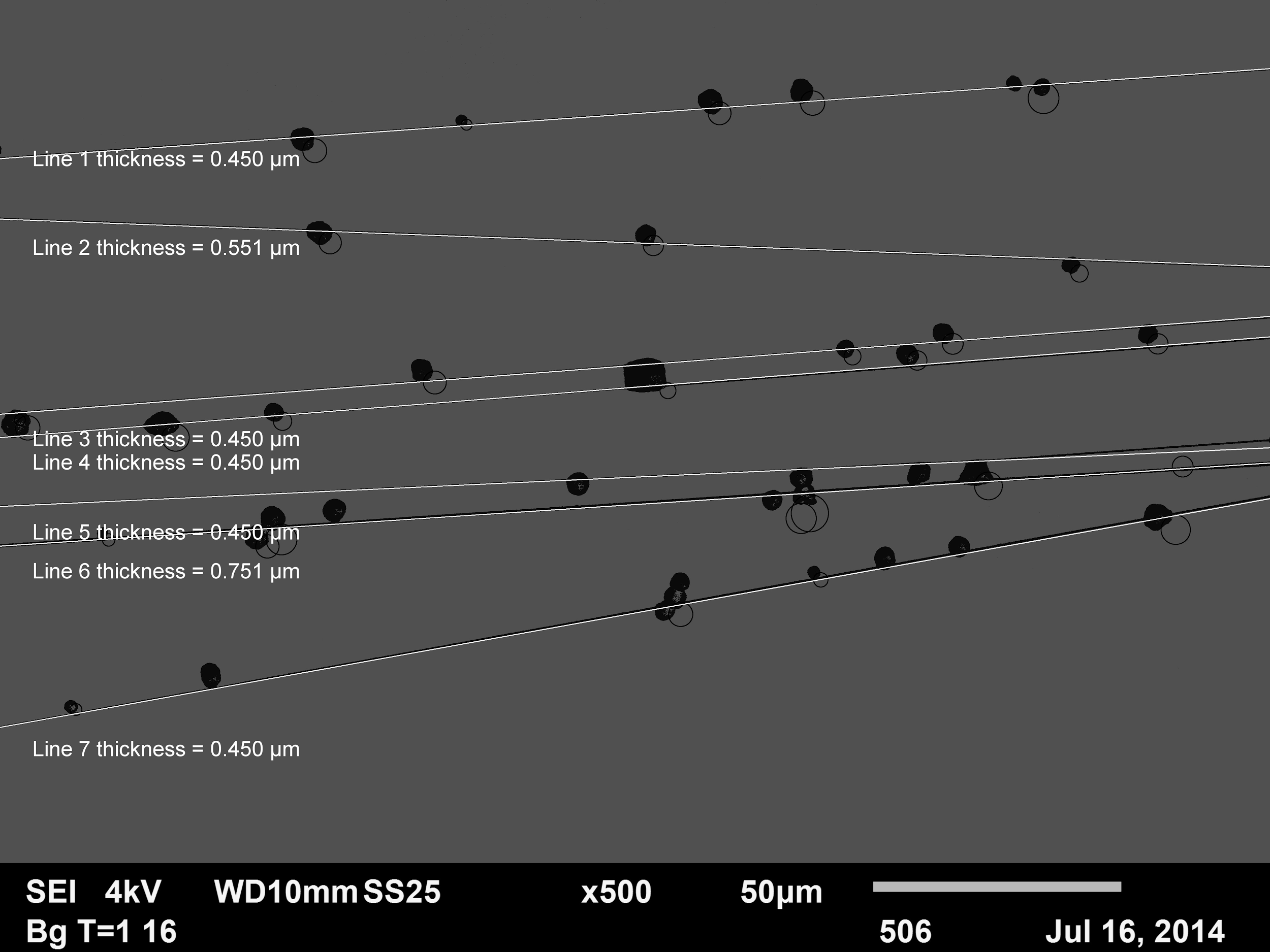


## 

## 

## 

## **Line Image**



## **Line Statistics**

Number of lines = 7

### **Equations of lines**

1. y = 0.071x +-638.642 thickness = 0.450 µm
2. y = -0.037x +-883.036 thickness = 0.551 µm
3. y = 0.077x +-1669.708 thickness = 0.450 µm
4. y = 0.079x +-1764.444 thickness = 0.450 µm
5. y = 0.046x +-2042.165 thickness = 0.450 µm
6. y = 0.065x +-2201.914 thickness = 0.751 µm
7. y = 0.180x +-2933.143 thickness = 0.450 µm

### **Thickness statistics**

|  |  |
| --- | --- |
| Minimum | 0.450 µm |
| Maximum | 0.751 µm |
| Mean | 0.508 µm |
| Standard Deviation | 0.114 µm |

### **Spindle Area Information**

|  |  |
| --- | --- |
| **Line Number** | **Spindle Area** |
| **Line 0** | **115.720 µm²** |
| **Line 1** | **141.180 µm²** |
| **Line 2** | **115.772 µm²** |
| **Line 3** | **115.793 µm²** |
| **Line 4** | **115.554 µm²** |
| **Line 5** | **192.796 µm²** |
| **Line 6** | **117.291 µm²** |
| **Total** | **914.107 µm²** |
| **Percent Coverage** | **1.856%** |

### 

### 

### 

### 

### **Spindle Separation Information**

|  |  |  |
| --- | --- | --- |
| **Distance between** | **Max distance** | **Min distance** |
| **1 to 2** | **798.677** | **244.394** |
| **2 to 3** | **244.394** | **786.672** |
| **3 to 4** | **786.672** | **200.824** |
| **4 to 5** | **200.824** | **94.737** |
| **5 to 6** | **94.737** | **83.256** |
| **6 to 7** | **83.256** | **446.958** |

## 

**LineHunter Recommendations**

The algorithm works very well for simple micrographs but has difficulty with more complex micrographs. Detection of spindles without the startingX and spindle thickness parameters would make the program more user friendly, Detection of spindles that intersect and/or aggregates would make the algorithm applicable to more micrographs. Future research could concentrate on determining if this algorithm is sufficient or if other approaches should be used. Template matching for lines and machine learning may help.

Future improvement could add the detection and removal of influential data points (see <https://onlinecourses.science.psu.edu/stat501/node/340>). Because false positive detection of edge points can significantly skew the calculated linear equation it is important to remove all outlines. The outliers should have been clustered with another line or they are just falsely detected edges because inherent variation in the micrograph.

Another approach would be to standardize the input micrographs and make them more been uniform. Dugway Proving Ground could require all micrographs have relatively parallel spindles and spaced more than a typical “large” droplet. The algorithm works with close to 100% accuracy for simple micrographs so this approach may be more economical than spending man hours on an what could be an intractable problem.

**Droplet Detection Methods**

Two different methods were used to detect droplets. One used the equations of the lines found to detect the edges of the droplets which is a continuation the path we took for spindle detection. Then circular regression was used to calculate the droplet center points and radius.

The other method uses template matching since the images are all very similar and this technique will make it very easy to find lines and droplets. We already know that the lines are a certain thickness within a range and this knowledge has simplified the image processing. Temple matching creates a model of an image of interest and then it searches over the whole image to find the image of interest that matches the template. The algorithm is that it calculates for each pixel in the image of interest, the normalized cross correlation coefficient between the template and the underlying pixels in the image of interest.

**Template Matching**

Template matching is a technique that identifies the parts of an image that match a template that was predefined. Template matching algorithms allows us to find occurrences of a template even if there is a local brightness or different orientation. Template matching is very flexible and straightforward to use. Computational power limits temple matching when trying to identify big and complex templates. Template needs to address the following needs: provide a reference image and an image to be compared to so we can identify all the locations on the compared image that match the reference image (Perveen, Kumar & Bilaspur, 2013).

Naive template matching is a rather straightforward approach. The template image is compared to every possible location of the reference image. At each location some numeric measures of similarity between the template and the reference are computed. Then the positions that measure the best similarities are identified (Perveen, Kumar & Bilaspur, 2013).

Image correlation or cross correlation is the method for finding image correlation. It is a simple sum of pairwise multiplications of the corresponding pixel values in the image. Cross correlation is not very robust. It is biased by changes in global brightness. The brightness of an image can greatly increase the cross correlation with another image even if the second image is not at all similar. A better way is to do the Normalized cross-correlation. Normalized cross correlation global brightness changes between the two images and has no effect on the result. It also has the advantage that the correlation value is scaled to [-1,1]. Once all the correlation is found a predefined threshold will determine if the correlation are a match (Perveen, Kumar & Bilaspur, 2013).

Image pyramid is used to scale down images from the original image. This is to enhance the efficacy of the correlation based template detection. The template image is still discernible after significant scaling down of the template image. This make the correlation process faster (Perveen, Kumar & Bilaspur, 2013).

Grayscale based matching is an advanced technique that is used for searching for a template in any rotated position. This is achieved by computing a set of image pyramids templates. This set includes a template position for each template orientation. Also each level of the search only verifies pairs that scored well on the previous level or seemed to match the template in the image of lower resolution (Perveen, Kumar & Bilaspur, 2013).

Edge based matching enhances grayscale based matching by having the template being shaped mainly by its edges. Instead of matching the whole template the edges are extracted and this avoids unnecessary computations (Perveen, Kumar & Bilaspur, 2013).

William found an ImageJ plugin that would template match a circle and find the circle on the spider web image. The Template Matching plugin by Walter O’Dell uses normalized cross correlation coefficient (O’Dell, 2005). Source code is available and we have included it in the deliverables which could be used for future classes. The functions of this plugin are:

- Create Disk Template button: An example image is given of a circular template. This is good for binary images and works well with the circular nature of anthrax molecules.

- Inverty template button: Inverts the image so the template has a darker background and a bright center.

- Crop Template from Image button: If there is an object on the image that is like the template you can draw an ROI around it and use the crop template from image button to apply the ROI as the template. You can also save the image for use on other images that you import into ImageJ.

- Load Template from File button: If you have an ROI image you want to use outside the image already saved you can load the pre-existing template and use it as the current template.

- Perform Statistical Correlation: When this button is pressed normalized cross correlation computation is invoked. This operation is an O(n3) so large template image will cause the computation to take a few minutes. It is a good idea to shrink the base image by a factor of 2.

- Set threshold and get particle analyzer results: This button uses built in ImageJ analyzers. The threshold levels and particle sizes those of the build in particle analyzer and are applied to the correlation image. The program turns the corresponding pixels ‘red.’ The matching particles are then outlined with a green box on both the correlation and original image. The number of counts and the pixel locations are reported in the results text window (O’Dell, 2005).

When using this plugin you need to apply the image pyramid on the reference image. The spider web images are way too big for this plugin. If the normal sized reference image is used the computation takes forever and usually crashes. ImageJ has a process built in that will reshape the image. If you go to the Image tab then go to scale you can adjust the pixels this will reshape the image with either the bilinear or bicubic algorithms. By shaping the image by down by a scale of 10 it is a lot easier for the computation to find the templates by either using the create disk or crop template buttons. Best results are achieved when we scale the original image by 10. Then creating a disk that is about 5-10 pixels, inverting the template, and then running the computation. This only takes a few minutes and it finds every droplet.

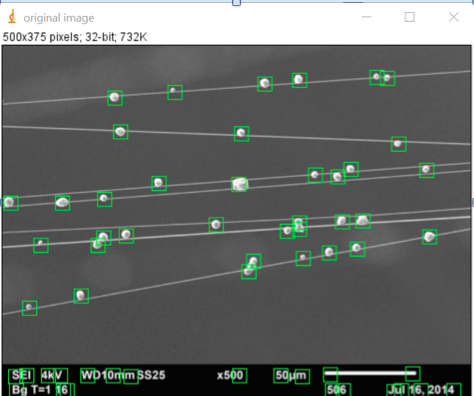
****

Figure 4. After running the ImageJ Template matching plugin all aerosol droplets were found

**Template Matching Recommendations**

The template matching plug by O’Dell has source code available. Future classes could combine the WebHunter spring 2017 team’s source code and the template matching code from O’Dell. Notice in the above micrograph, that droplets are detected in the annotation part of the micrograph. The read\_scale class from WebHunter 2017 could be used in the template matching code to avoid the false positives found by the O’Dell’s template matching program.

**Droplet Detection Using Circular Regression**

Similar to the linear regression technique to find spindles, a circular regression was used to detect droplets. The prior information from the linear regression and linear equations are used to optimize the speed of the algorithm. The linear equations are used to guide the detection for left and right edges of the circles. Parallel lines above and below the previous found spindles are traversed to detect edges of the circles. When a left and right edge is detected it is clustered into groups of close points and then a circular regression is done to find the center points and radii of the droplets. Information about the typical size of a droplet is used to avoid false positives and negatives. The algorithm is described in more detail below.

For each linear equation that was found

1. Traverse parallel lines above the equation to find left and right edges of droplets
2. Traverse parallel lines below the equation to find left and right edges of droplets

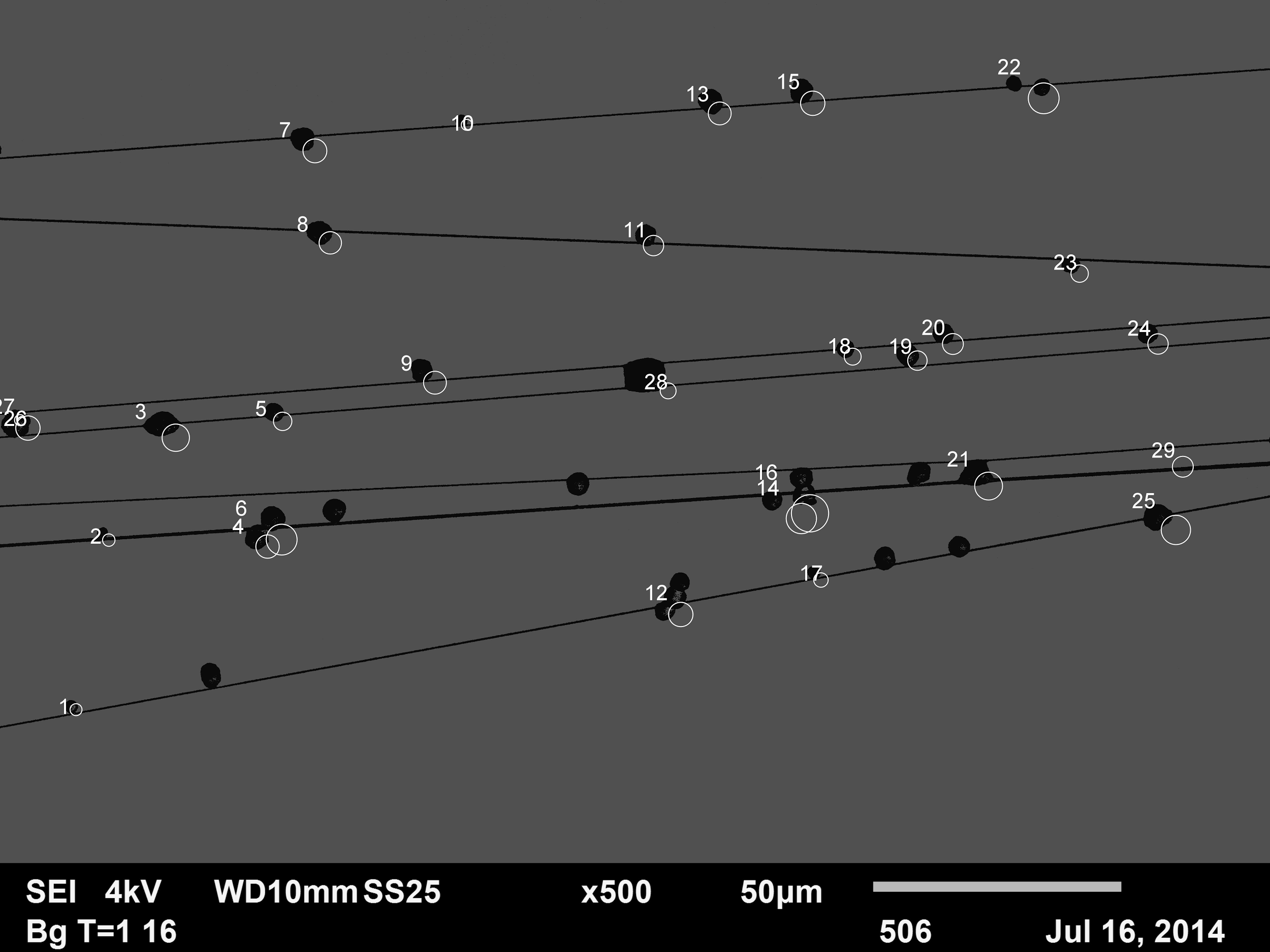
Once points are collected perform circular regression to find droplets.

HTML reports have been created similar to the spindle reports. Statistics on spatial distribution and droplet size are included in the report. A sample report is shown below.

## 

## 

## **Droplet Image**



**Droplet Statistics**

Number of droplets = 29

### **Droplet Area Information**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Droplet Number** | **x** | **y** | **Area (µm²)** | **Diameter (µ)** |
| Droplet 1 | 282 | 2838 | 3.772 | 2.402 |
| Droplet 2 | 413 | 2153 | 3.929 | 2.503 |
| Droplet 3 | 653 | 1710 | 8.644 | 5.506 |
| Droplet 4 | 1031 | 2157 | 7.386 | 4.705 |
| Droplet 5 | 1102 | 1662 | 5.815 | 3.704 |
| Droplet 6 | 1073 | 2114 | 9.744 | 6.206 |
| Droplet 7 | 1221 | 560 | 7.544 | 4.805 |
| Droplet 8 | 1286 | 934 | 7.072 | 4.505 |
| Droplet 9 | 1707 | 1497 | 7.229 | 4.605 |
| Droplet 10 | 1859 | 481 | 3.457 | 2.202 |
| Droplet 11 | 2593 | 949 | 6.443 | 4.104 |
| Droplet 12 | 2695 | 2429 | 7.701 | 4.905 |
| Droplet 13 | 2855 | 411 | 7.229 | 4.605 |
| Droplet 14 | 3169 | 2030 | 9.587 | 6.106 |
| Droplet 15 | 3227 | 367 | 7.701 | 4.905 |
| Droplet 16 | 3190 | 1995 | 11.787 | 7.508 |
| Droplet 17 | 3280 | 2310 | 4.558 | 2.903 |
| Droplet 18 | 3403 | 1404 | 5.343 | 3.403 |
| Droplet 19 | 3659 | 1415 | 6.129 | 3.904 |
| Droplet 20 | 3799 | 1345 | 6.601 | 4.204 |
| Droplet 21 | 3929 | 1904 | 8.801 | 5.606 |
| Droplet 22 | 4145 | 334 | 9.744 | 6.206 |
| Droplet 23 | 4317 | 1068 | 5.501 | 3.504 |
| Droplet 24 | 4627 | 1346 | 6.443 | 4.104 |
| Droplet 25 | 4681 | 2078 | 9.272 | 5.906 |
| Droplet 26 | 58 | 1672 | 3.457 | 2.202 |
| Droplet 27 | 63 | 1677 | 7.701 | 4.905 |
| Droplet 28 | 2661 | 1544 | 5.029 | 3.203 |
| Droplet 29 | 4726 | 1840 | 6.601 | 4.204 |
| Total |  |  | 200.218 µm² |  |
| Percent Coverage |  |  | 0.407% |  |

### **Area statistics**

|  |  |
| --- | --- |
| Minimum | 3.457 µm |
| Maximum | 11.787 µm |
| Mean | 6.904 µm |
| Standard Deviation | 0.104 µm |

### 

### 

### 

### 

### **Diameter statistics**

|  |  |
| --- | --- |
| Minimum | 3.457 µm |
| Maximum | 11.787 µm |
| Mean | 6.904 µm |
| Standard Deviation | 0.104 µm |

### 

### 

### **Distance from middle statistics**

|  |  |
| --- | --- |
| Minimum | 19.486 µm |
| Maximum | 125.839 µm |
| Mean | 77.512 µm |
| Standard Deviation | 1.561 µm |

**LineHunter and WebHunter Recommendations**

Regression techniques can be very sensitive to the effects of outliers and just one data point can cause large errors in the analysis (Rousseeuw, P. J., & Leroy, A. M., 2005). A seminal paper on analysis of leverage points in linear regression developed a simple metric is now called Cook’s distance (Cook, 1977). Cook’s distance can be used to identify outliers and has been used to develop robust regression algorithms that can remove outlier errors. Due to the limited development time for the BMTN 670 capstone project it was not possible to add robust regression to the plugin code. Implementing robust regression may improve the algorithm’s accuracy.

Both the linear and circle algorithms used for the WebHunter 2017 use a starting x,y coordinate and then essentially cluster the new edge points as they are detected. A critical aspect is that errors in clustering will results in large errors in the regression analysis. Moreover, if lines intersect or are parallel and close then errors will likely show in the results.

Based on the results of the two different approaches, template matching appears to be the most accurate. The main disadvantage of the template matching algorithm is that the complexity is O(n³). The disadvantage of the regression approach is that accuracy is not as good but because the algorithm essentially samples the image; however, it is very fast.

A possible approach for future classes is to combine the two algorithms. The WebHunter 2017 code that was developed could use different functions of the O’Dell’s template matching code to find spindles and circles. Because it is known that droplets will only be found near spindles the linear equations can guide the detection of the template matcher. This modification should change the Big-O from cubic to linear O(n).

**Machine Learning**

Machine Learning is a type of artificial intelligence. It provides a way for computers to learn without being explicitly programmed. When new data is exposed to the computer program the program can change according to the data. The program is looking for patterns in the data but instead of extracting the data for human comprehension the program will find patterns and adjust the program’s actions accordingly. There are two types of machine learning algorithms: supervised and unsupervised. Supervised algorithms apply what has been learned from past data to the new data. Unsupervised algorithms gains inferences from the datasets that it learns from (Sommer).

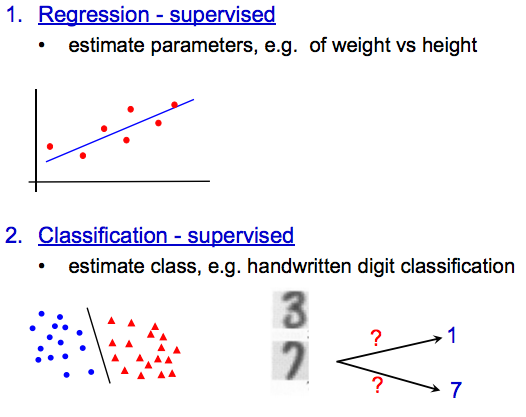


Figure 5. Supervised Machine Learning

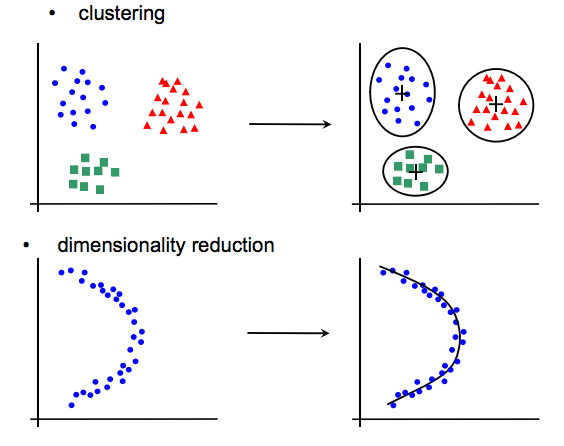


Figure 6. Unsupervised Machine Learning

Zisserman (2015). Retrieved from: http://www.robots.ox.ac.uk/~az/lectures/ml/lect1.pdf

When working on a problem that will require machine learning there is a six steps that will help to figure out the problem. They are as follows:

1. Problem Definition: we need to understand clearly and be able to describe the problem that needs to be solved.

2. Analyze Data: The data that will be processed needs to be understood.

3. Prepare Data: The dataset needs to be discovered, cleaned and structured.

4. Evaluate Algorithms: Develop a robust test harness and an accuracy test to which your algorithm will be checked.

5. Improve Results: After testing leverage results to develop better models.

6. Present Results: Be able to describe the problem and solution clearly to someone outside the project (4-Steps).



Figure 7. Shows the Machine Learning process tree

Weka Machine Learning Workbench is an open source platform for beginners and pros alike to learn how to build machine learning algorithms. It has some great features to help. It provides a simple graphical interface for the user. It lets the user explore algorithms and datasets. It also contains state of the art algorithms. It also has a lot of Decision trees, rule based algorithms and ensemble methods (Weka).

When applying machine learning to WebHunter there a couple characteristics that we could look at. We could train the machine to predict spindles distance, size, and web configuration. We have two variables we are trying to learn about the aerosol droplets and the spider webs. Using Weka, a training image is needed such that spindles and droplets are highlighted and then a classifier is created. The classifier is then used on a test dataset to detect the features. Probabilities are given and a image is created that identifies features from the classifier. The above steps are done iteratively until a good classifier with required sensitivity and selectivity is found.

Notably many of the machine learning techniques described were actually used in our code. Regression was used extensively and also template matching could be viewed as a simplified version of machine learning.

**Possible Bioengineered Solutions for the Future**

Team WebHunter 2017 approached the task of modifying the WebHunter project in a manner that, ultimately, lead us to approach the project in a different way than previous teams. It is not surprising that our associated research efforts brought us down a similar path, as we began to examine the potential for re-designing certain aspects of the testing process, most notably the possibility of using bioengineered spider webs as a replacement for current testing methods that use native spider webs. A bioengineered spider web might benefit the project in a variety of ways and it serves as a feasible substitute. Markus J. Buehler’s research group at M.I.T. have devoted significant efforts towards understanding the intricacies of spider web architectures (Qin et al., 2015). Furthermore, Qin (2015) and associated researchers maintain that “finding suitable natural samples for experimental testing of entire webs in a repeatable way is effectively impossible” (Qin et al., 2015). With this in mind, this group is currently using microscale 3D printing to engineer and test synthetic web scaffolds mimicking orb webs constructed out of polydimethylsiloxane (PDMS) (Qin et al., 2015). Importantly, the web construction methods can be adapted to other combinations and architectures as well (Qin et al., 2015). After the web is printed, aqueous glue is applied to the spiral threads resulting in a sticky surface coating for trapping prey (Qin et al., 2015). Effectively, this research group is able to alter web geometry through changing the numbers of spiral threads which can result in a material distribution within the web that is optimized and exhibits improved trapping strength (Qin et al., 2015). These synthetic webs have low density and high strength which makes them ideal for a host of engineering applications (Qin et al., 2015). Our team feels that since these webs have highly engineered geometries along with preconfigured web distances and thicknesses that they become more uniform and possibly well-optimized for quantitation (Qin et al., 2015). These webs possess information that could be retained prior to and after testing, which may be useful when paired with the WebHunter Project. Some of the problems that we encountered when designing the WebHunter program involved the clumping together web fibers and general randomness of this substrate material of web fibers. A bioengineered web material might be engineered with polymers that solve for this problem. With this in mind, it is our hope that bioengineered possibilities be considered for testing purposes in the future.

**Conclusion**

The team of project WebHunter took a novel approach to overcome all the objectives. 100% accuracy for line detection was achieved using plugin approach instead of macros. 100% droplet detection was achieved using Template matching. Extensive research was done about Machine learning techniques to determine if they could be applied to achieve the objectives of the capstone project.

## **Appendix I**

### **Deliverables**

We have created a number of deliverables that will facilitate knowledge transfer to the next Capstone class.

1. WebHunter 2017 User Guide
2. Installation folder
3. WebHunter 2017 code is a zipped file. Code can also be found on github at <https://github.com/jplaschke/webhunter>
4. Updated 2016 macro to automatically write .csv file
5. Quick Start Simple Development Guide
6. Quick Start Advanced Development Guide
7. Test dataset
8. Results file
9. O’Dell’s Template matching source code
10. A revised LineHunter .ijm file that corrects the data output for line lengths

**References**

Belleman, M., Depew, J., Jenkins, S., Mitthivong, O., Okolo, C. & Salzano, K. (2016).

*Team WebHunter Fall 2016.* Unpublished manuscript, Department of Biotechnology, University of Maryland University College, Baltimore, United States.

Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*,

*19*(1), 15-18.

ImageJ Features*.* (n.d.). Retrieved from http://imagej.net/Fiji

Kirschner, S.G. (2017). Dugway Proving Ground Mission & Capabilities Overview. Retrieved

from<http://www.dugway.army.mil/documents/DPG%20Command%20Brief.pdf>

Nichols, K.L. (2017). Project WebHunter [Project Selection]. Retrieved from

<https://learn.umuc.edu/d2l/le/content/202223/viewContent/8950617/View>

O'Dell, W. (2005). Template Matching. Retrieved April 09, 2017, from

https://imagej.nih.gov/ij/plugins/template-matching.html

Perveen, N., Kumar, D., & Bilaspur, I. (2013). An Overview on Template Matching

Methodologies and its Applications. *International Journal of Research in Computer and*

*Communication Technology,* *2*(10), 988-995.

Qin, Z., Compton, B. G., Lewis, J. A., & Buehler, M. J. (2015). Structural optimization of

3D-printed synthetic spider webs for high strength. *Nature Communications*, *6*, 7038. <http://doi.org/10.1038/ncomms8038>

Rousseeuw, P. J., & Leroy, A. M. (2005). Robust regression and outlier detection (Vol. 589).

John wiley & sons.

Soldiers.2017. Retrieved from:

<http://soldiers.dodlive.mil/2015/04/spider-webs-on-front-lines-of-bio-warfare-detection/>)

Sommer, C., & Gerlich, D. W. (n.d.). Machine learning in cell biology - teaching computers to

recognize phenotypes. *Journal of Cell Science*. doi:10.1242/jcs.123604

Tafti, P.A., Kirkpatrick, A.B., Holz, J.D., Owen, H.A, Yu, Z., (2015). 3DSEM: A 3D

microscopy dataset. *Data Brief*. Mar 6:112-116 doi: 10.1016/j.dib.2015.11.018

Trucco, Chapter 4 AND Jain et al., Chapter 5

United States Army Dugway Proving Grounds (2017). Retrieved from:

<http://www.dugway.army.mil/>

Weka 3: Data Mining Software in Java. (n.d.). Retrieved March 26, 2017, from

http://www.cs.waikato.ac.nz/ml/weka/

Zisserman, A. (2015). Machine Learning. Retrieved from:

<http://www.robots.ox.ac.uk/~az/lectures/ml/lect1.pdf>