University of Nottingham
UK | CHINA | MALAYSIA

COMP 2032:

Introduction to Image Processing - Coursework

Date of Completion: 15/04/2022

Student ID	Name
20218675	Muhammad Hanz Affiq Bin Zuki

Table of Contents

1. Introduction

Image Processing has been a very multifaceted sector in computing. There are many functions to image processing. Some of these include being able to assist in research and capture data much more accurately as well as being able to create better images through the use of different filters.

The focus for this coursework will be to successfully automate a program to accurately segment an image. The images that we will be segmenting will be the leaves of the various types of plants. We will be utilizing the Matlab library of functions to perform this assignment. We will be using the Watershed segmentation algorithm to segment the images for this coursework.

Figure 1:Plant 001

Figure 2: Plant 002

Figure 3: Plant 003

These three images have been chosen to be segmented. Each image has different aspects to them. This means that we must create a program that will be able to compute the segmentation accurately based on their respective images. However, we must also make the program automated for ease of use for the user. This means we must figure out the best parameters for a general algorithm. This is because if the user decides to change the image in the future, they will still be able to utilize this algorithm for segmentation.

2. The Process

a. Sharpness Filter

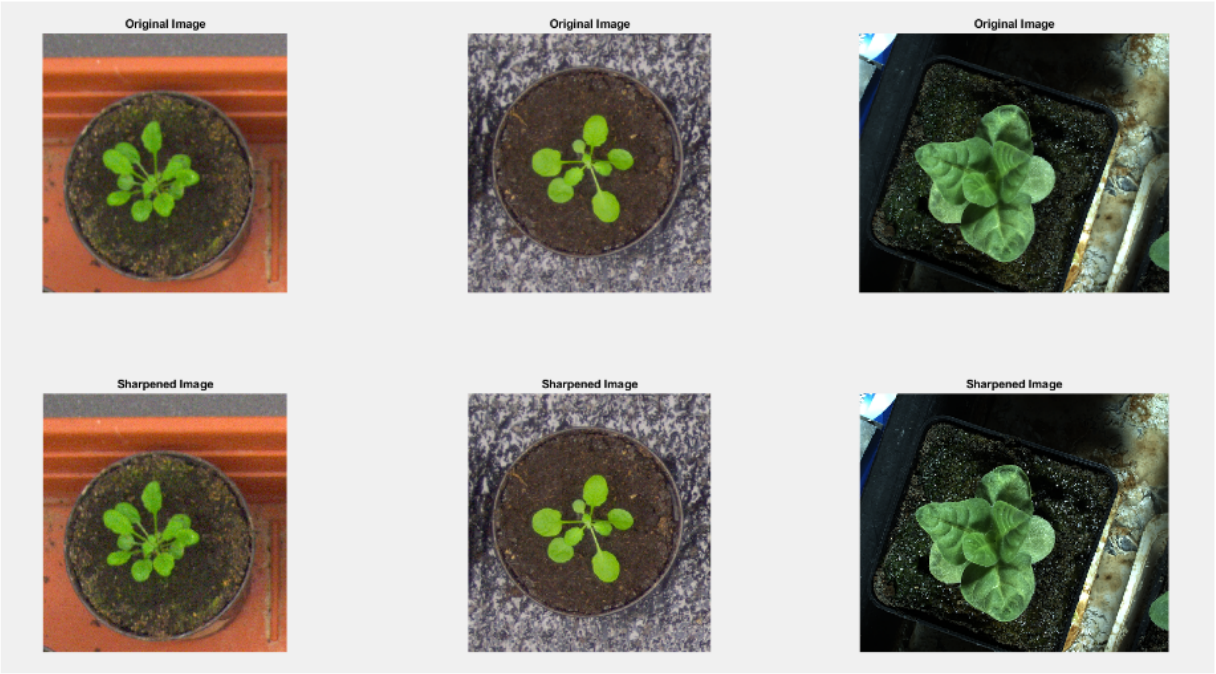
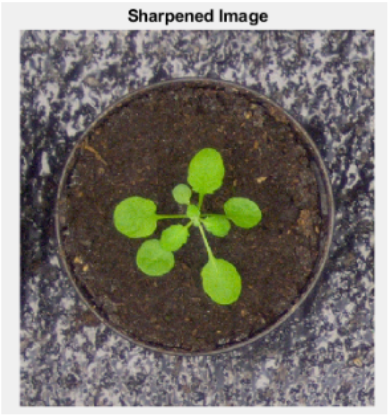
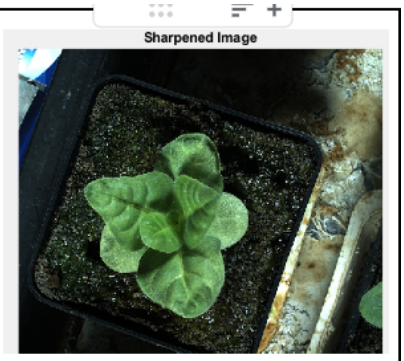


Figure a: The transformation of the Sharpened Image

To begin the pre-processing stage, we will start the transforming of the main images of the three plants and add a sharpening filter to the images.

This sharpening filter will sharpen the images RGB values to attain a more detailed and generally sharper image.

		
Figure a.1: Sharpened Image of Plant 001	*Figure a.2: Sharpened Image of Plant 002*	*Figure a.3: Sharpened Image of Plant 003*

This is done so that the image can be easily detected in the Matlab detection program for the leaves shape during the segmentation.

Due to this, the segmentation of the respective images are more detailed and accurate compared to the segmentation of the unsharpened images. This is because the unsharpened image may be under segmented due to the lack of detail from each of the leaves

I chose to use the values for the amount of 0.9 as it allows me to achieve a very balanced result for all of the leaves. If the amount of sharpening used is lower, then the segmentation of Figure a.1 will suffer. However, if the amount of sharpening used is higher, then the segmentation of Figure a.3 will be over segmented instead. I believe that the amount that I have chosen would result in a much more balanced and accurate segmentation process for all of the plants.

b. Greenness & Adjusting Filter

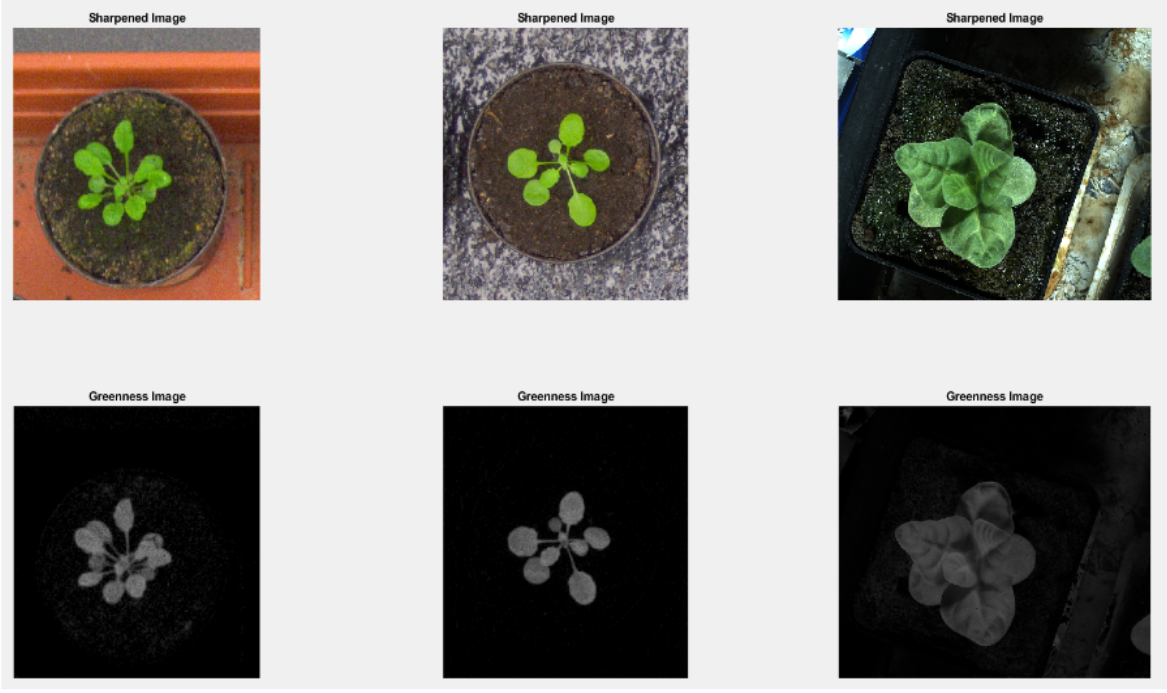


Figure b: The transformation of the Sharpened Image to the Greenness Image

The next step of pre-processing that we must implement in the images will be the Greenness and Adjustment Filter.

The Greenness filter acts as a method to only obtain the green color or G value of a specific image. It does this by separating the RGB values into their own respective channels. The greenness filter will then remove the red and blue channels from the image and will highlight the details of the greenness channel. The details will then be highlighted in the grayscale image of the greenness image.

Another function that has been added after the Greenness filter is the Adjust Filter. The imadjust or adjust filter is a filter that will adjust and saturate the highest and lowest value of pixels in the image. This in turn will allow the image's contrast to increase and then allow the finer details to be more visible.

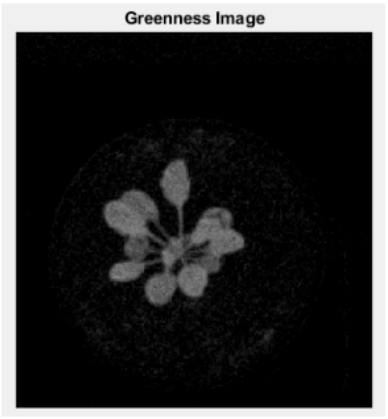
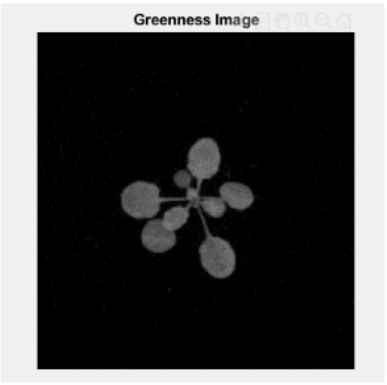
	

Figure b.1: Greenness Image of Plant 001	***Figure b.2: Greenness Image of Plant 002***	***Figure b.3: Greenness Image of Plant 003***

This filter is added because the program we are currently creating is focused on the detection of the leaves. Thus, the greenness of the leaves is a critical aspect of identifying the shape of the leaves in the given images.

The benefits of using the greenness filter and adjusting filter is that it allows us to easily detect the leaves shape based on the color. However, the greenness detection based on Figure b.1 and Figure b.3 is not seamless due to the shadows of the respective images. Due to this incomplete detection, the segmentation will not be perfectly isolated.

c. Median Filter

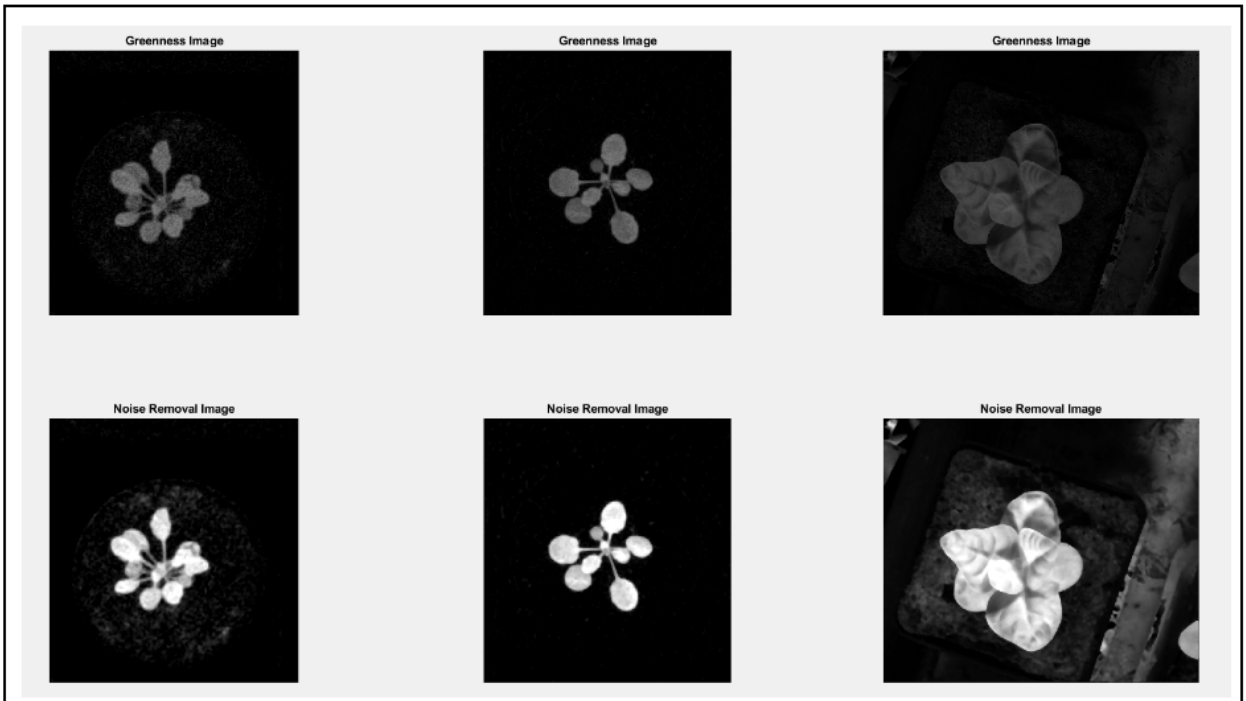


Figure c : The filtering of noise from the Greenness Image to the Noise Removed Image

The next step of preprocessing will be with the Noise Removal with the Median Filter. Initially, the use of functions such as imdilate, imerode, bwareaopen and many others were implemented to achieve this function so that the noise in this image is able to be removed. However, through various different experiments, we were able to discover the function of Median filter or medfilt was a more suitable function for this goal. The Median filter is a filter that will remove noise pixels on the image.

We are using this filter to remove any unnecessary noise from the Greenness image. This is because the unwanted noise pixels may accidentally be highlighted and magnified by the upcoming effect and filters. This filter will thus be able to remove any unwanted artifacts to leave a somewhat clean and focused image.

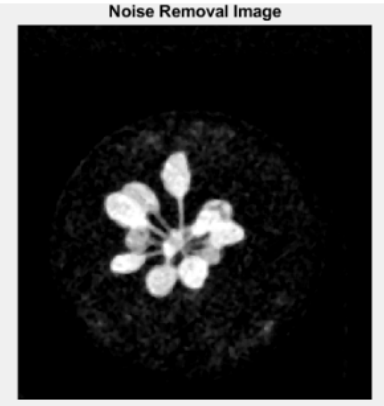
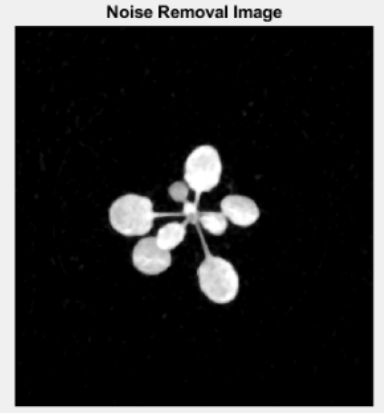
	

Figure c.1: Noise Removal Image of Plant 001	*Figure c.2: Noise Removal Image of Plant 002*	*Figure c.3: Noise Removal Image of Plant 003*

One of the negatives of using this method is that it does not completely remove the noise from the surrounding of the image. If we look at the transformation of image in Figure c.1, there are still minor noise pixels and unwanted artifacts around the image.

d. Binarization Filter

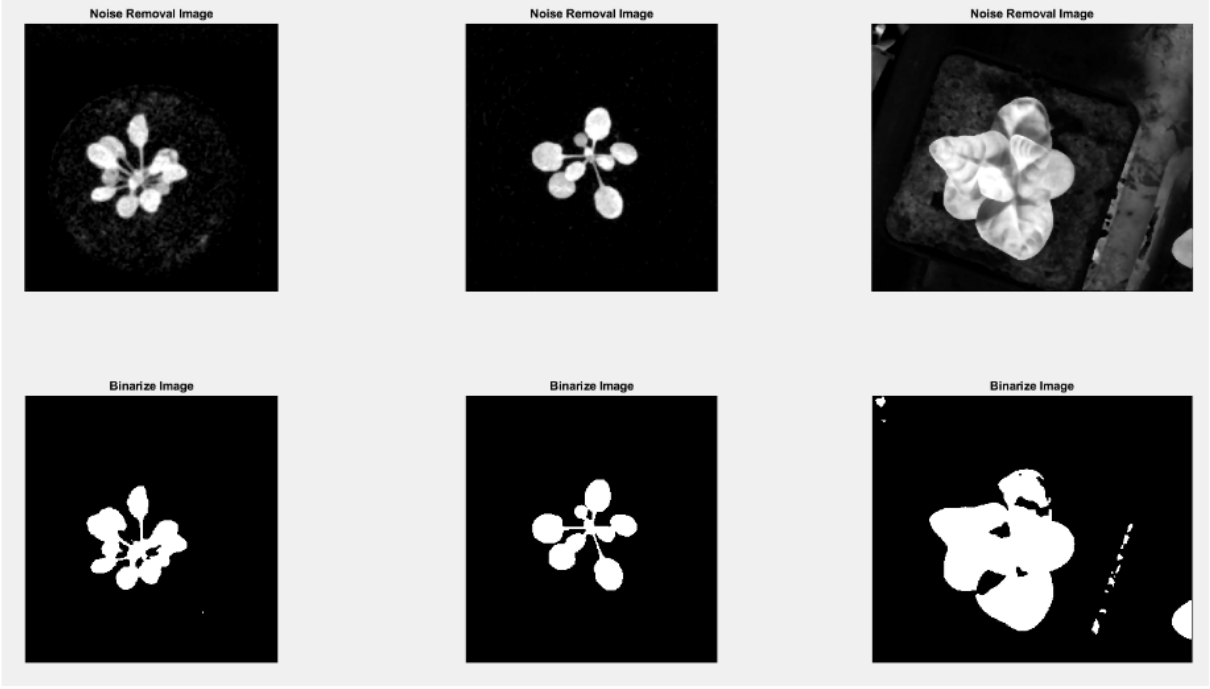


Figure d: The Binary transformation of the Noise Removal Image to Binarize Image

This step of preprocessing is known as imbinarize. Binarization is basically the process of dividing data into 2 groups and assigning one of two values to all members of the same group.

We will use this filter to binarize the adjusted image of the Greenness image. This will allow us to acquire a black and white image of the said images. This filter will thus make the green values of the leaves of the plant images highlighted white while the background black.

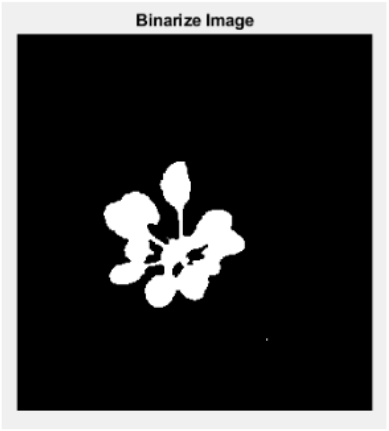
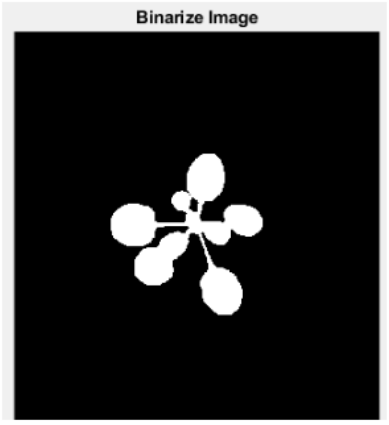
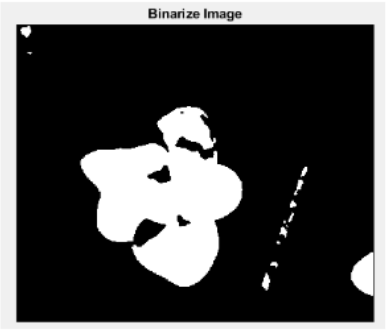
		

Figure d.1: Binarize Image of Plant 001	*Figure d.2: Binarize Image of Plant 002*	*Figure d.3: Binarize Image of Plant 003*

The benefit of using this filter is that it will allow the program to detect the leaves shape even better due to the lack of color variance. However, the downside is that minor noise pixels might become more visible due to the black background.

e. BW Area Filter

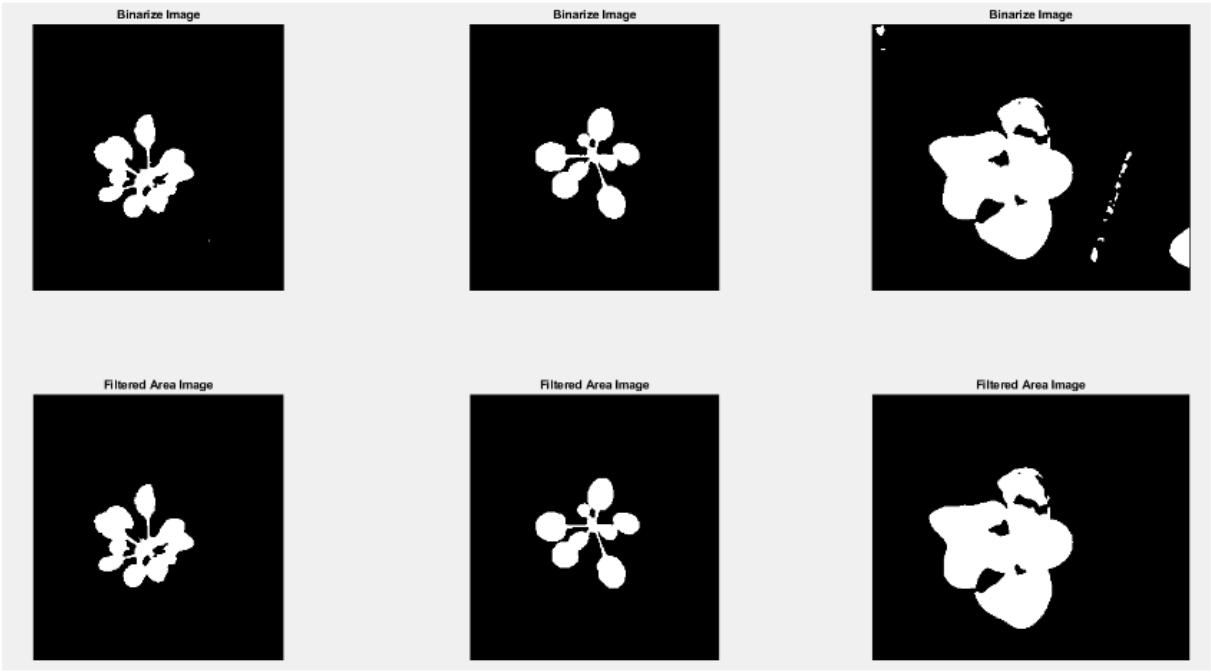


Figure e: The Filtered surroundings of the Binarize Image to the Filtered Area Image

This preprocessing stage is known as bwareafilt. bwareafilt is a function that is utilized to acquire all the connected components from the binarized image which will include only the connected components and will thus recreate a new binary image that is free of any noise or artifacts. ("Extract objects from binary image by size", n.d.)

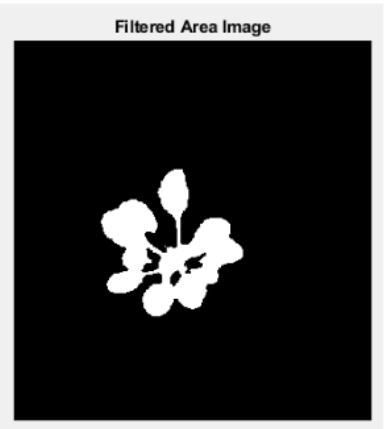
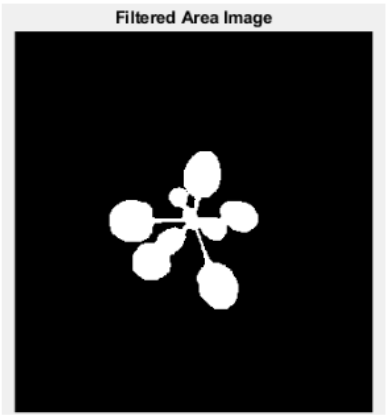
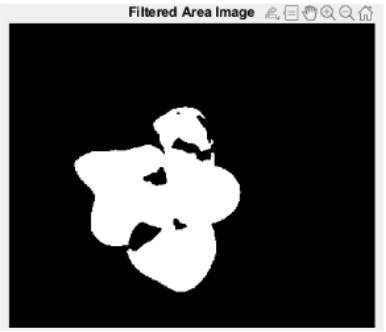
		

Figure e.1: Filtered Area Image of Plant 001	*Figure e.2: Filtered Area Image of Plant 002*	*Figure e.3: Filtered Area Image of Plant 003*

This is beneficial because it allows us to acquire a more clean and filtered image. This image can then be detected by the program much more accurately.

f. BW Distance Filter

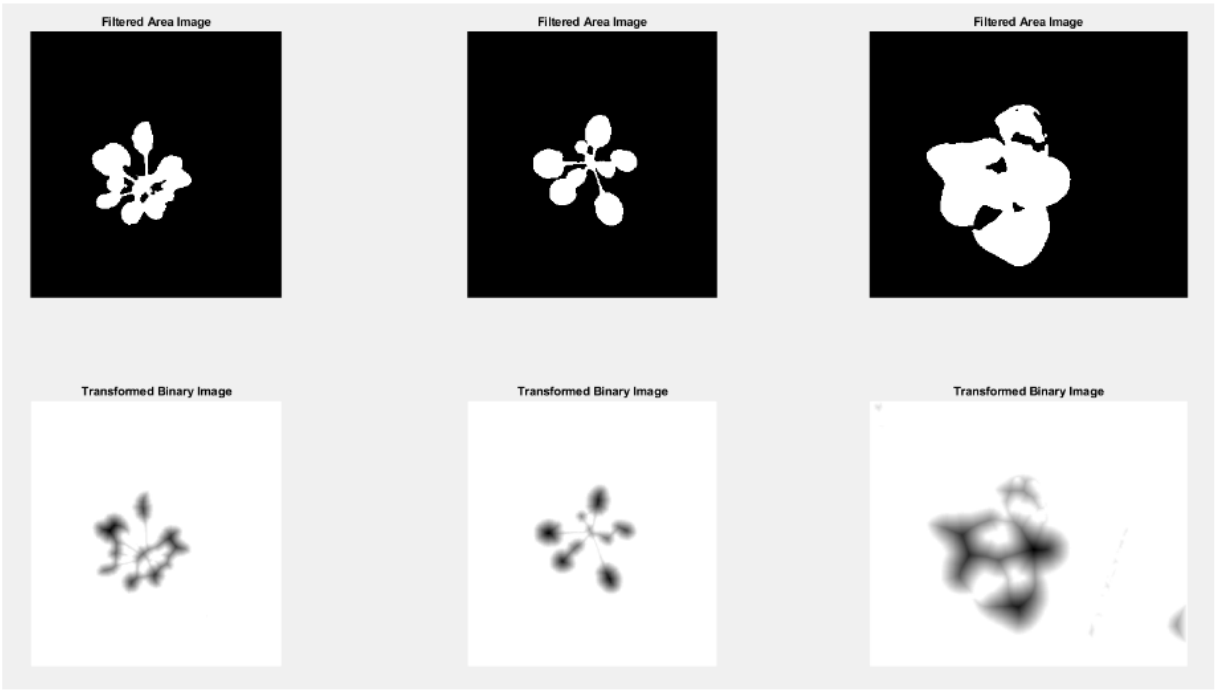


Figure f: The transformed Binary Image of the Filtered Area Image

This is the first stage of the segmentation phase known as bwdist. BW Distance Filter is a function that will allow us to calculate the length of the line segment between two separate points. To do this, the function will invert the binary image. This will cause the internal regions of the binary image to turn into grayscale and the outer region to white. This is so that it is able to determine the distance between the pixel and the closest non zero pixel ("Distance transform of binary image", n.d.).

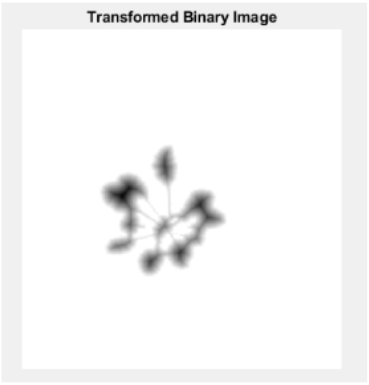
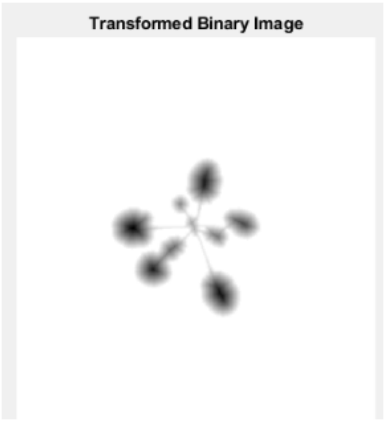
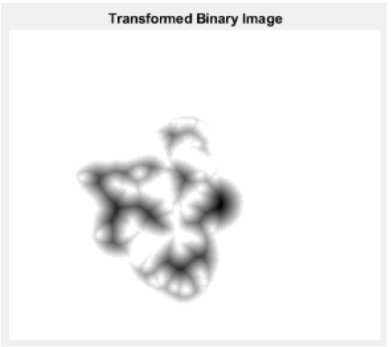
		

Figure f.1: Transformed Binary Image of Plant 001	***Figure f.2: Transformed Binary Image of Plant 002***	***Figure f.3: Transformed Binary Image of Plant 003***

This function is beneficial for the Watershed segmentation algorithm as it allows the program to be able to determine the images “watershed lines”. This is crucial as it is required for the segmentation to properly function.

g. Watershed Segmentation Algorithm Phase 1

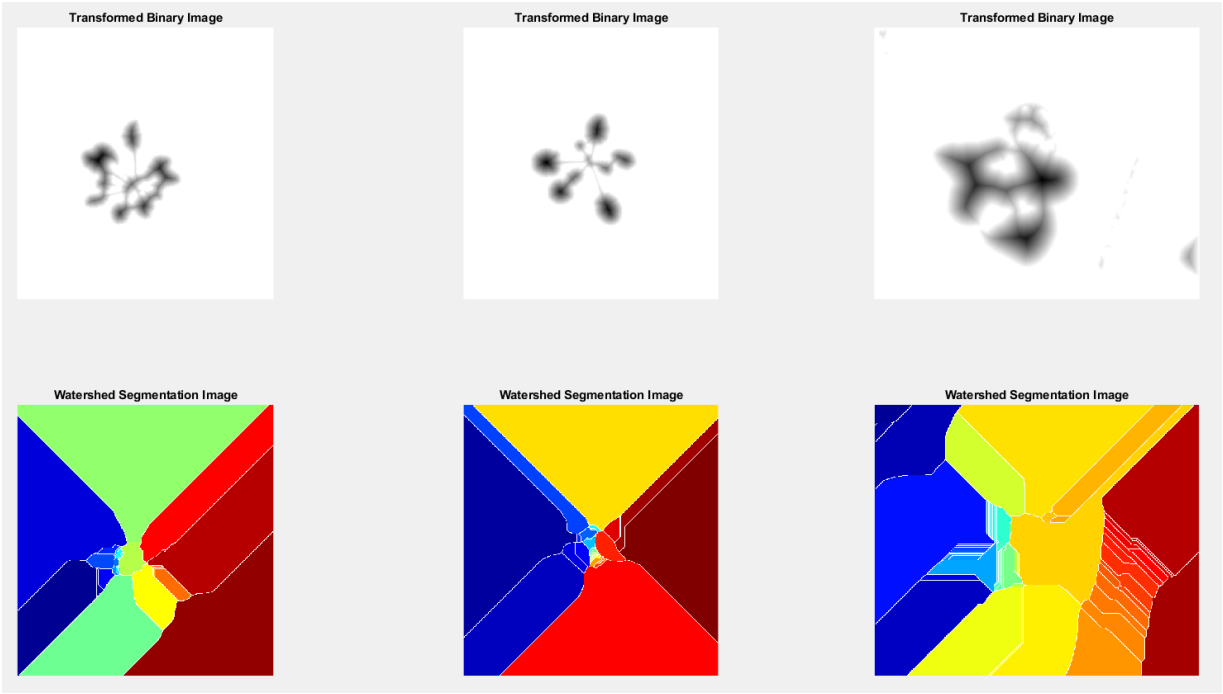


Figure g: The first segmentation of the transformed Binary Image by Watershed segmentation algorithm.

This process will implement the Watershed segmentation algorithm onto the binary image. The concept of Watershed segmentation algorithm can be described as two spatial coordinates and one intensity that will then look for and use the "watershed lines" to begin its segmentation (Seal, Das & Sen, 2015).

The segmentation will look for the pixels that represent a higher plane also known as brighter pixels as well lower plane pixels or otherwise known as darker pixels. The algorithm will then segment the image based on those lines.

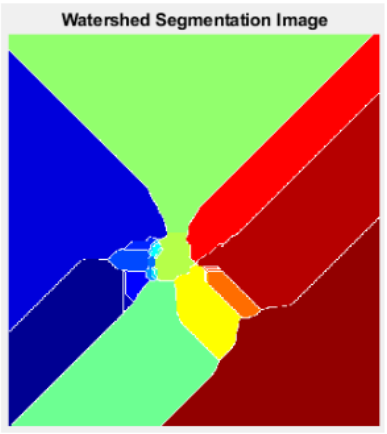
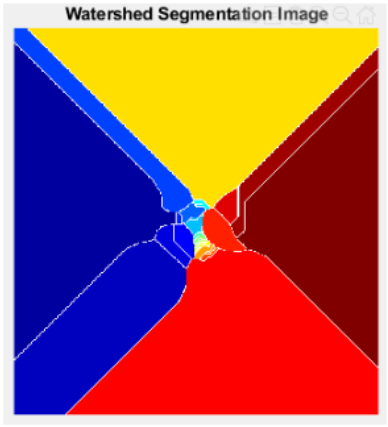
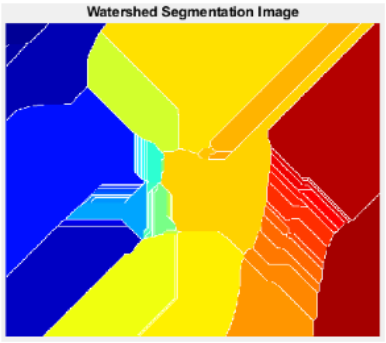
		

Figure g.1: First Watershed Segmentation of Plant 001	*Figure g.2: First Watershed Segmentation Image of Plant 002*	*Figure g.3: First Watershed Segmentation of Plant 003*

I have also added the label2rgb function. This function will allow us to easily identify the segmentation lines that have been created by the Watershed segmentation algorithm as shown in Figure g.1,g.2 and g.3.

h. First Reassigned Variable

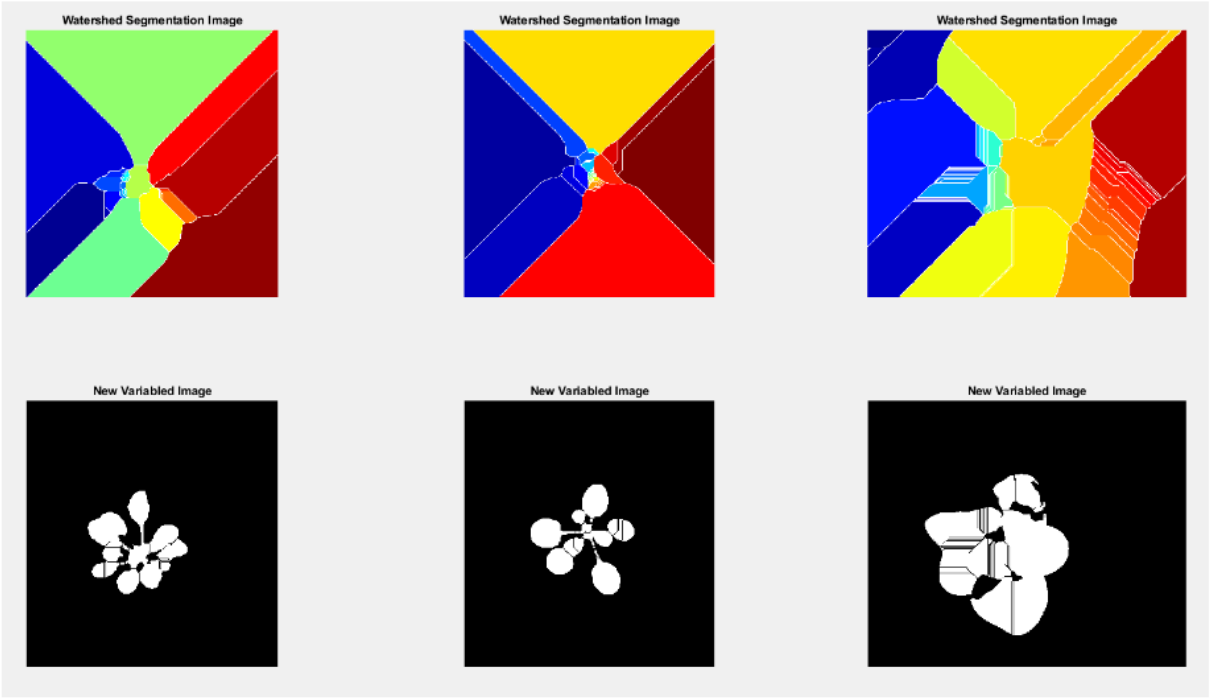


Figure h: The newly assigned variable of the First Watershed Segmentation image.

This stage of the process is simply to show the Watershed segmentation in a binary image and reassign the values to a new variable.

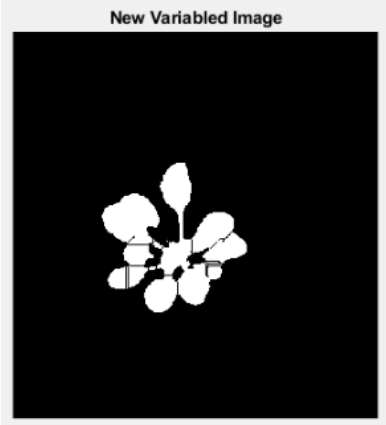
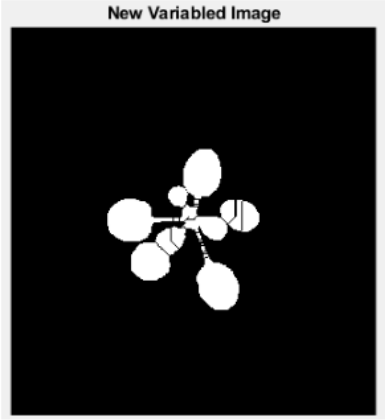
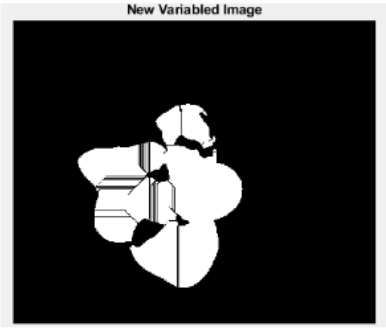
		

Figure h.1: First Watershed Segmentation Image of new assigned variable of Plant 001	*Figure h.2: First Watershed Segmentation Image of new assigned variable of Plant 002*	*Figure h.3: First Watershed Segmentation Image of new assigned variable of Plant 003*

i. Extended Minima Transformation Filter

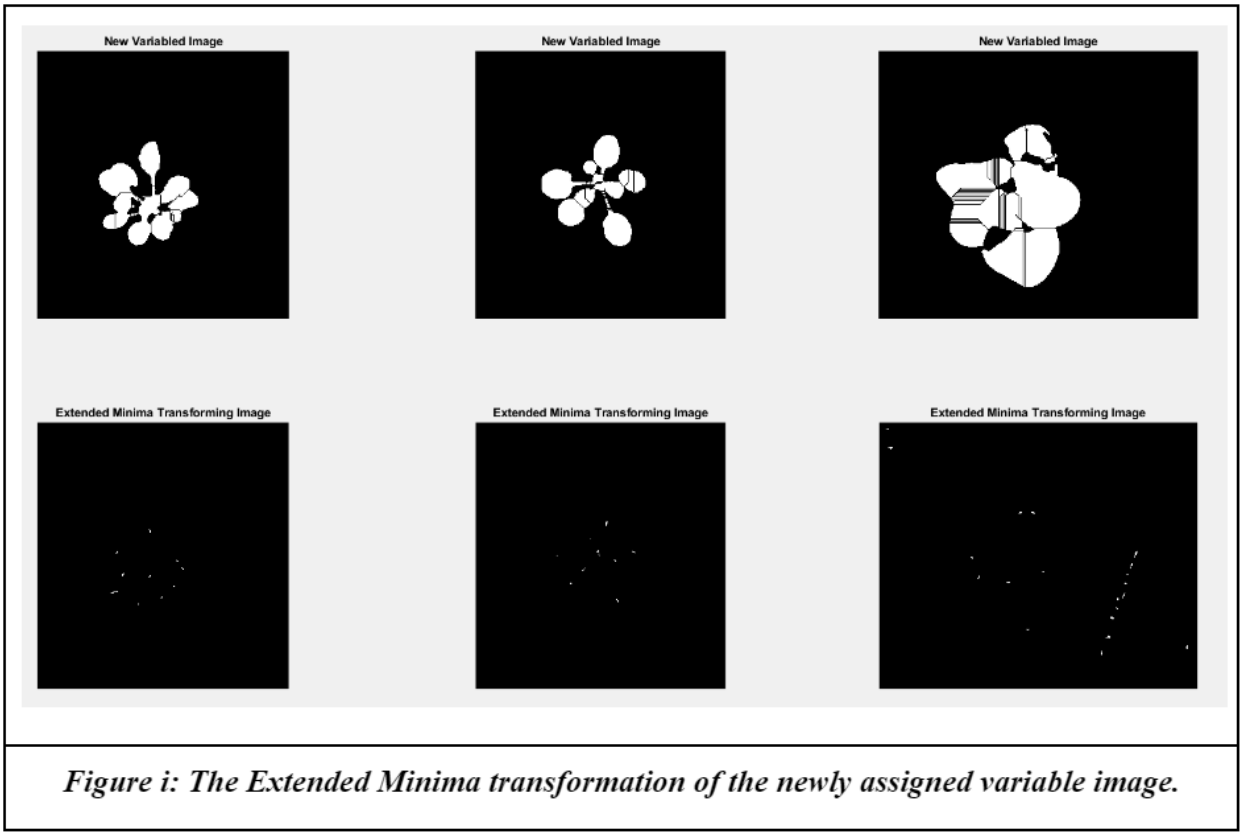


Figure i: The Extended Minima transformation of the newly assigned variable image.

This phase of the segmentation process is known as imextendedmin. The imextended function also known as the Extended Minima transform filter will compute region minima based on the H-minima transformation ("Extended-minima transform", n.d.).

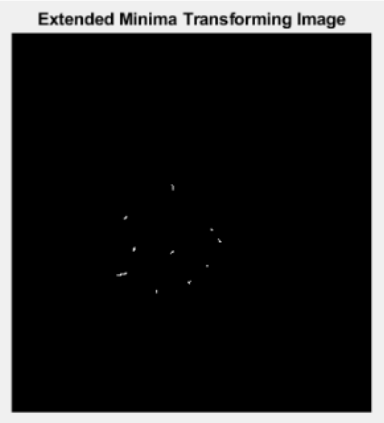
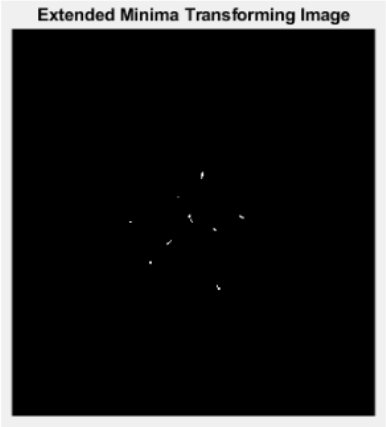
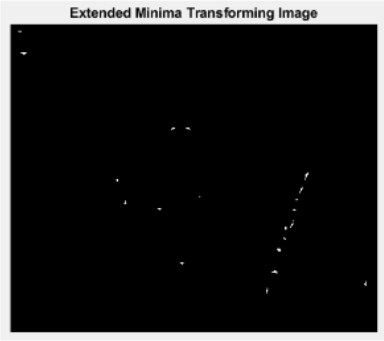
		

Figure i.1: The Extended Minima Transformed Image of Plant 001	***Figure i.2: The Extended Minima Transformed Image of Plant 002***	***Figure i.3: The Extended Minima Transformed Image of Plant 003***

The effect of the minima can be quite affected by the H-value. If the value of the H was higher, the output of the images would become more conjoined. However, if the value of H was lower, the output of the images would become more separated.

j. Morphological Reconstruction Filter

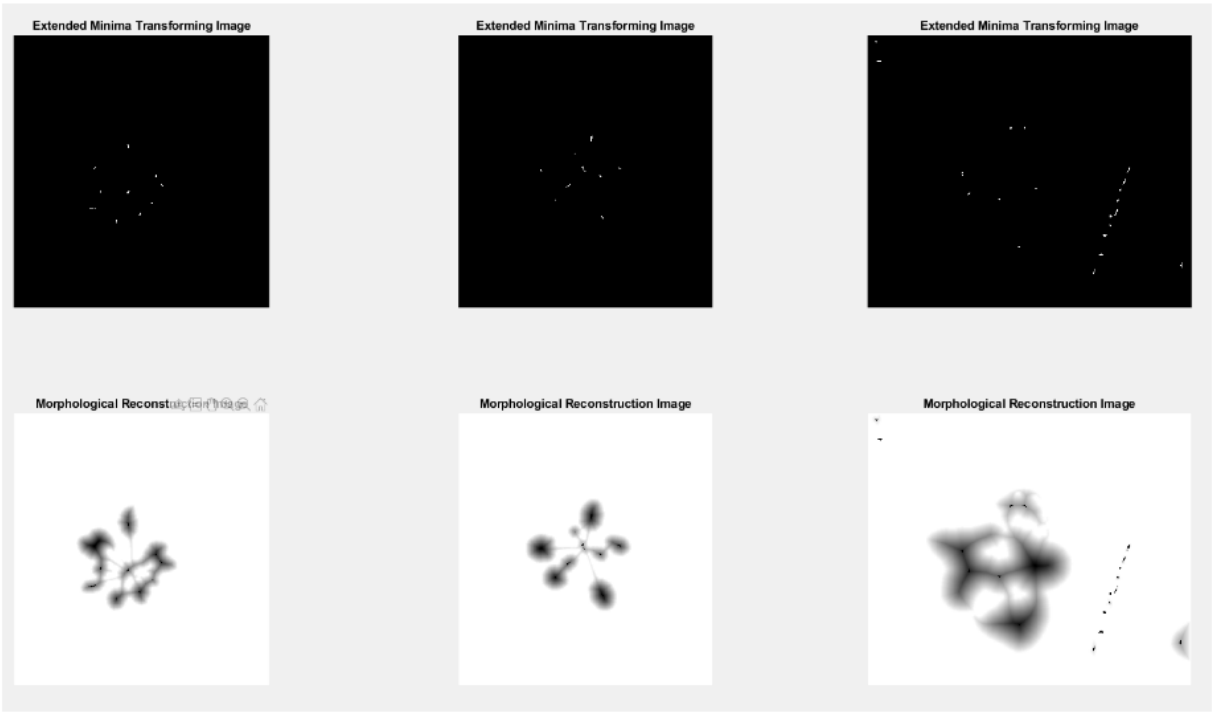


Figure j: The Morphological Reconstruction of Extended Minina Image

This phase is known as imimposemin. The imimposemin or Masked Minima function will be used to modify the extended minima transformed images. It does this by modifying the grayscale mask image through morphological reconstruction ("Impose minima", n.d.) This will then reconstruct the morphology of the image based on the extended minima.

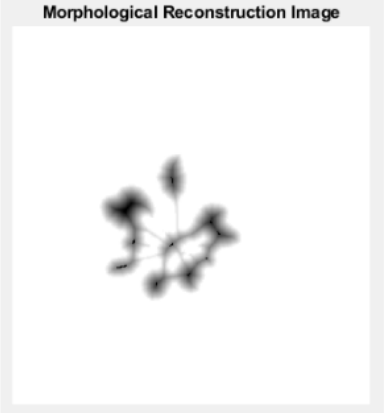
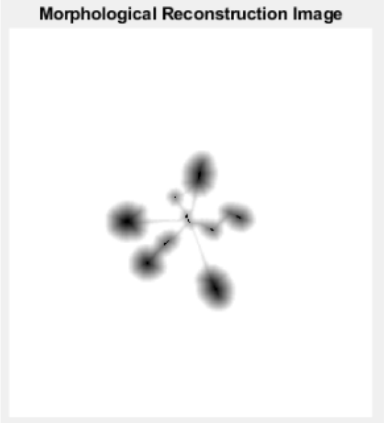
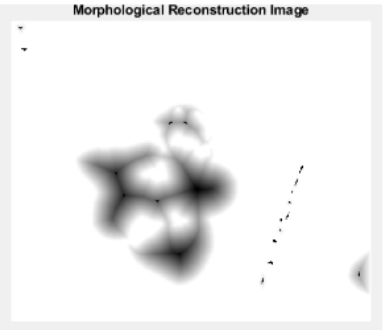
		

Figure j.1: The Morphological Reconstruction Image of Plant 001	*Figure j.2: The Morphological Reconstruction Image of Plant 002*	*Figure j.3: The Morphological Reconstruction Image of Plant 003*

k. Watershed Segmentation Algorithm Phase 2

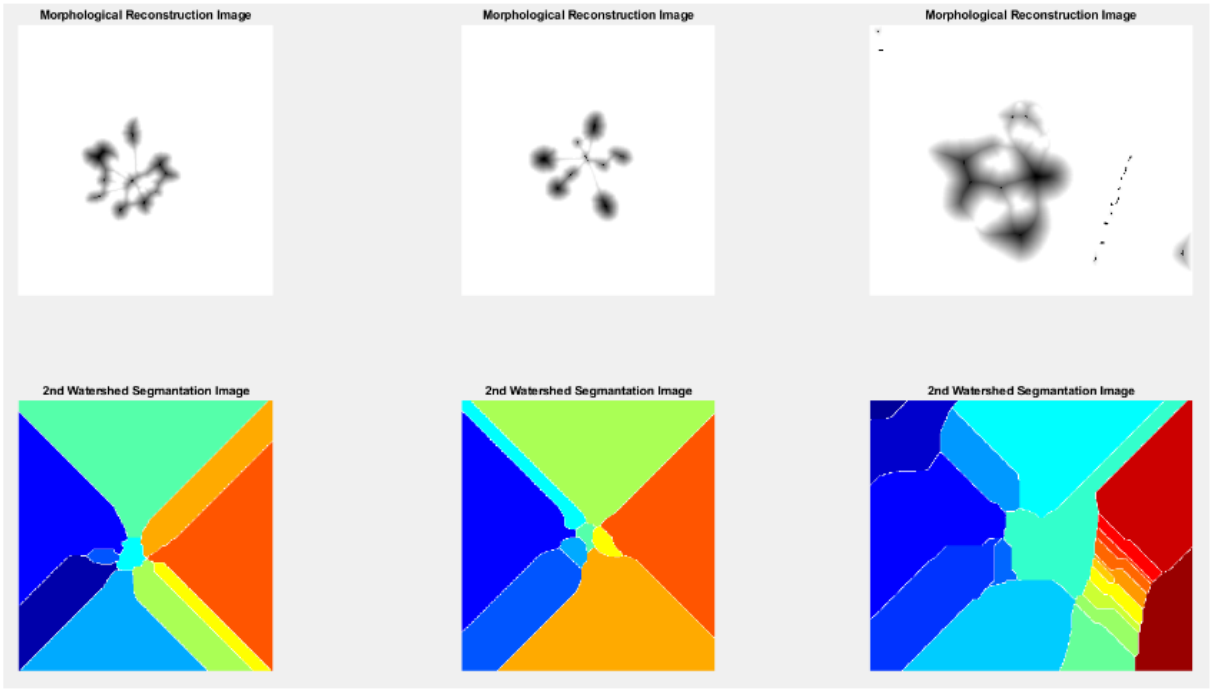


Figure k: The second segmentation of the Morphological Reconstruction Image by Watershed Segmentation Algorithm.

This will be the second implementation of the Watershed segmentation algorithm. This is to allow the segmentation algorithm to acquire more details pertaining to the image and create more accurate segmentation lines.

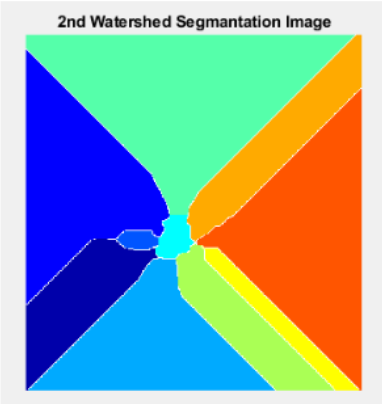
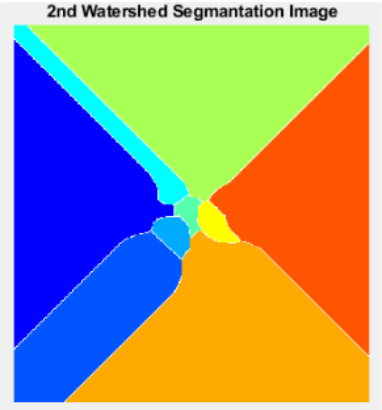
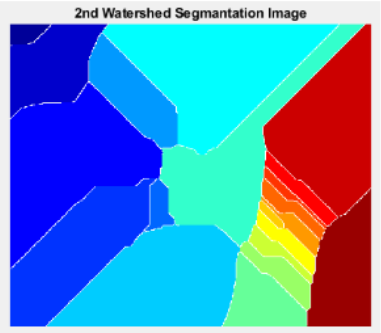
		

Figure k.1: The 2nd Watershed Segmentation Image of Plant 001	*Figure k.2: The 2nd Watershed Segmentation ImageImage of Plant 002*	*Figure k.3: The 2nd Watershed Segmentation Image of Plant 003*

l. Second Reassigned Variable

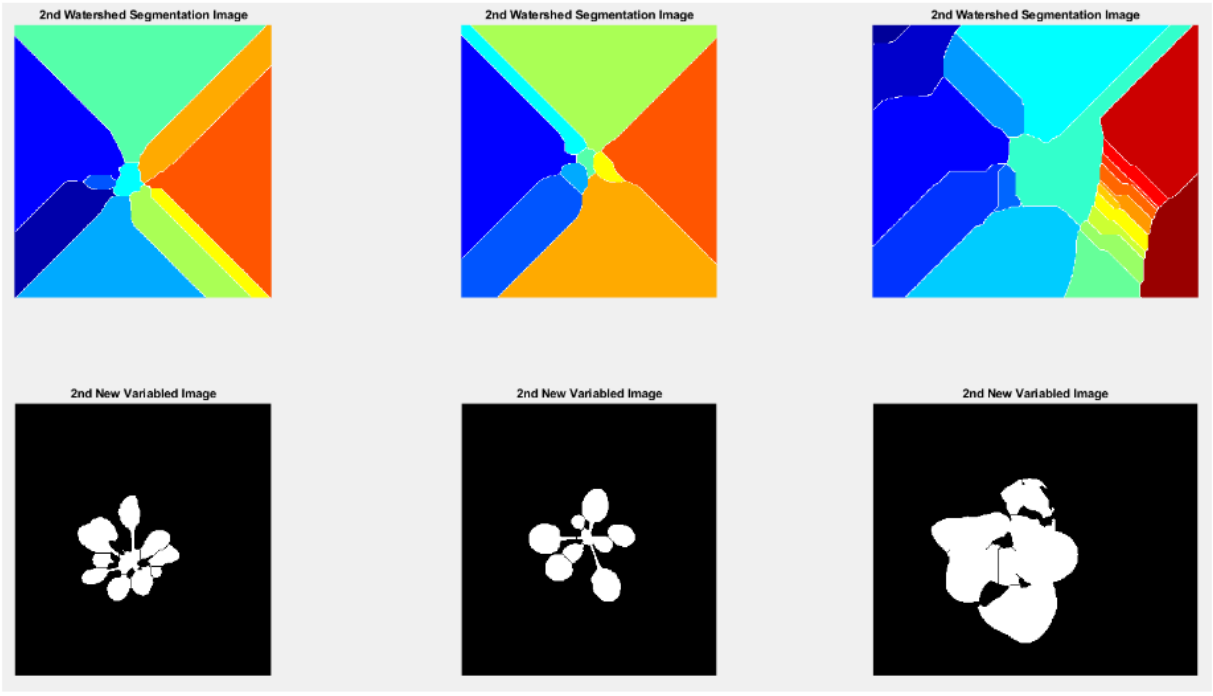


Figure l: The newly assigned variable of the Second Watershed Segmentation image

This process will output the binary image of the second Watershed segmentation as represented in a binary image.

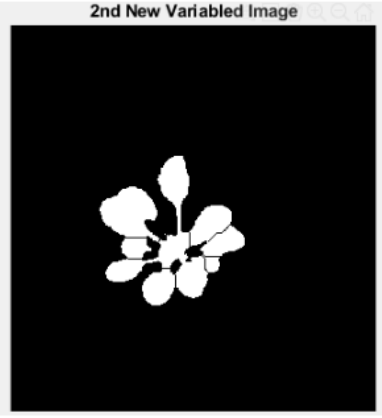
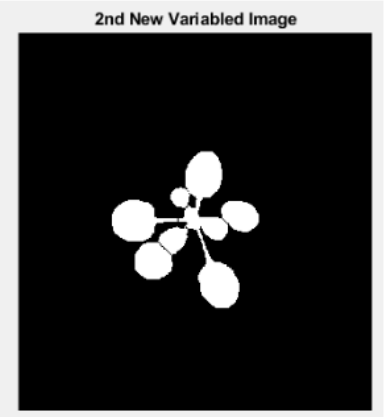
	

Figure l.1: Second Watershed Segmentation Image of new assigned variable of Plant 001	*Figure l.2: Second Watershed Segmentation Image of new assigned variable of Plant 002*	*Figure l.3: Second Watershed Segmentation Image of new assigned variable of Plant 003*

m. **Label Matrix (Bwconncomp)**

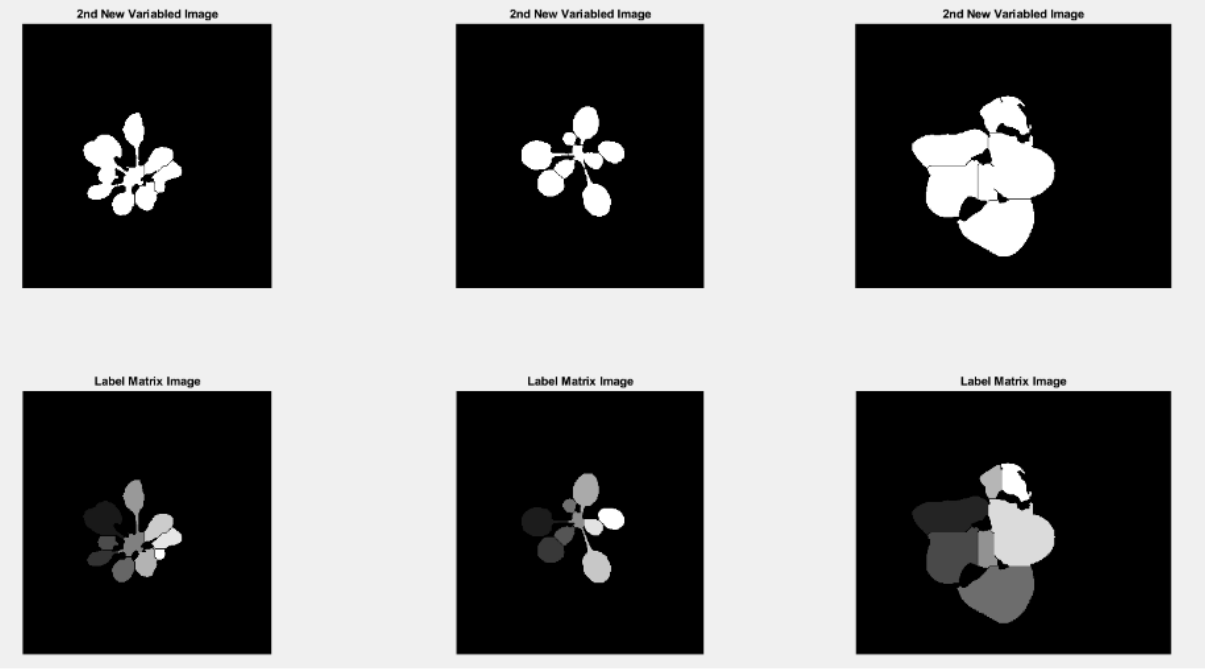


Figure m: The Label Matrix of the new variabled images

This is the second to last step of the Watershed segmentation algorithm which is the Label Matrix. The Label Matrix function will label the connected components that have different values with the bwconncomp function and allow us to identify each respective connected component in the binary image ("Create label matrix from bwconncomp structure", n.d.).

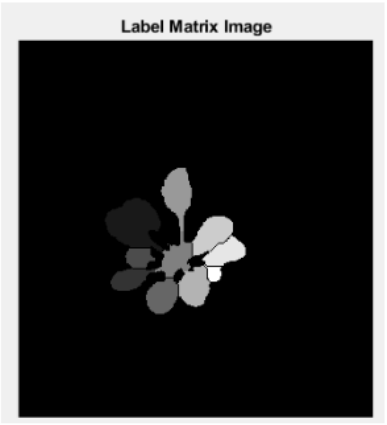
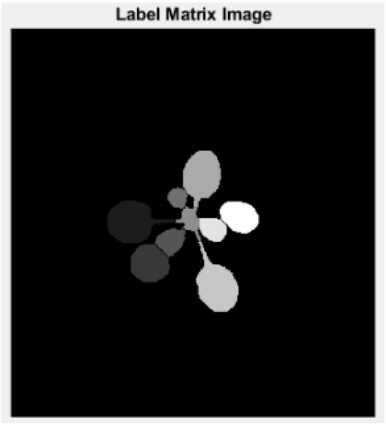
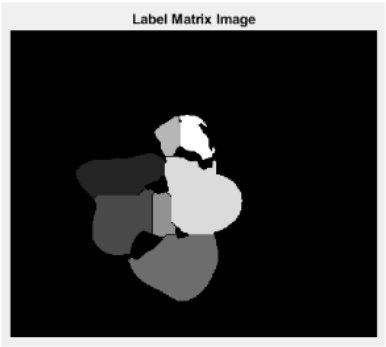
		

Figure m.1: The Label Matrix Image of Plant 001	*Figure m.2: The Label Matrix Image of Plant 002*	*Figure m.3: The Label Matrix Image of Plant 003*

n. **Color Output**

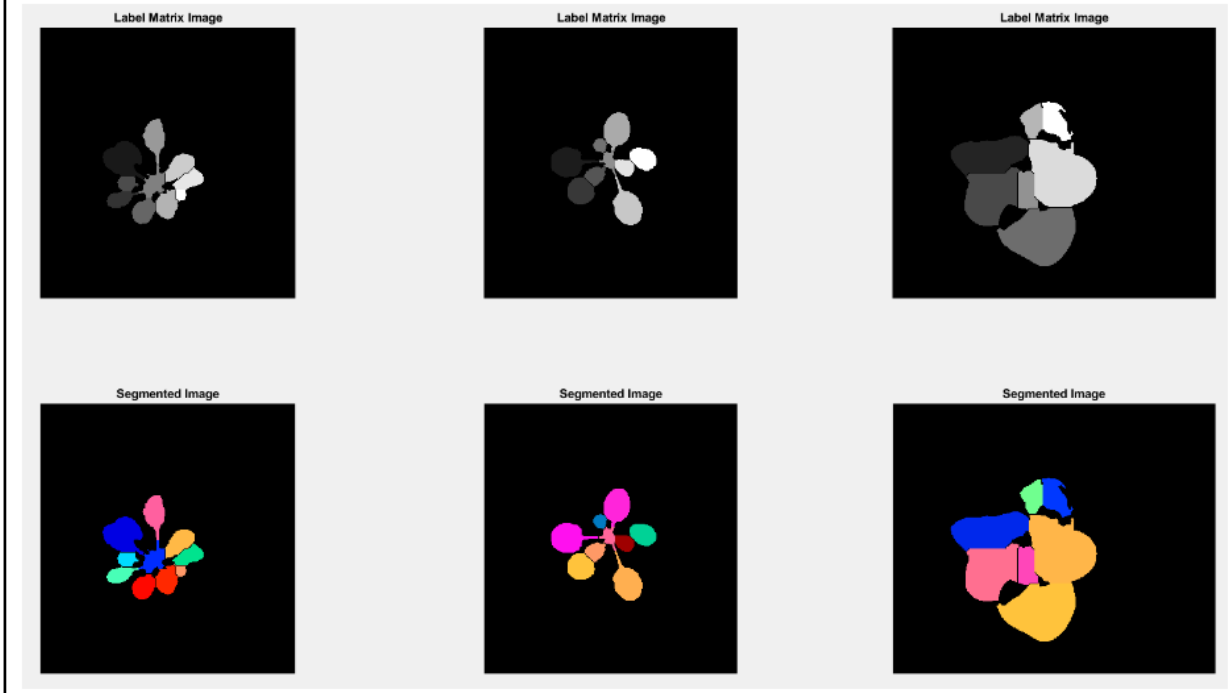


Figure n: The input of the RGB values onto the Label Matrix image.

This will be the final phase of the Watershed segmentation algorithm which will be the Color function. The Color function will allow us to input colors into our Watershed segmented binary image.

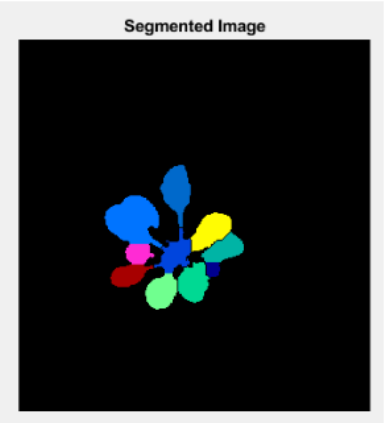
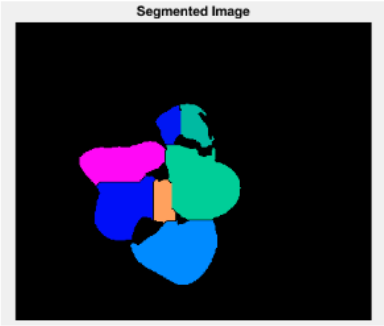
		

Figure n.1: The RGB Label Matrix Image of Plant 001	*Figure n.2: The RGB Label Matrix Image of Plant 002*	*Figure n.3: The RGB Label Matrix Image of Plant 003*

The output of color of the Label Matrix has been made with the colormap function. This function allows us to set the colormap to the figure to one of the predefined colormaps ("View and set current colormap", n.d.). This acts as a method to randomize during each run as it will allow us to properly identify each leaf and determine the accuracy of the Watershed segmentation algorithm as seen in the Figure n1, n.2 and n.3.

4. Final Output

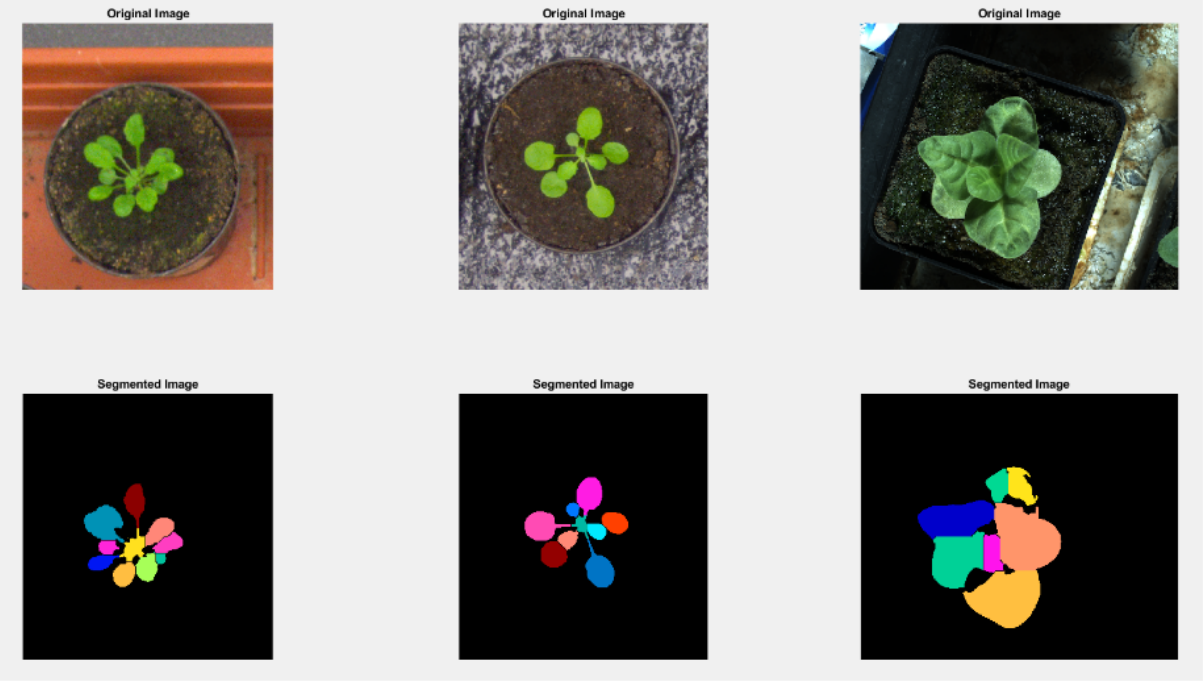


Figure 4: This will be the output of the segmentation program utilizing Watershed Segmentation Algorithm.

This is the final output of the program after utilizing Watershed segmentation algorithm for the image. Each respective image has been successfully segmented and randomly filled with various colors from different colormaps to show the details of the segmentation.

5. Conclusion

In conclusion, there are many aspects that will affect the success rate of the segmentation of the respective images.

For example, the quality of image may be able to result in a completely different segmentation of the image. If the image has a higher resolution of pixels, the segmentation might not be accurate due to the detail that is present compared to the same image with a lower resolution. For example, based on the sharpening of the images, if it is higher , then the image in Figure 3 will suffer heavily and oversegment. Comparatively, if the sharpness is too low, the image in Figure 1 will begin under segment causing the separate leaves to begin conjoining and becoming one whole segment.

On the other hand, the shadows of the image can also affect the segmentation of the image. This is because an image with shadows may prove much more difficult to segment due to the lack of pixels in specific areas of the image. This lack of pixels will lead to a gap in the connected components that will thus cause certain areas to be segmented incorrectly.

The color of the image can also affect the compatibility of segmentation drastically. If the image contains very few colors, it may allow for the segmentation to have a higher success rate compared to having too many different colors.

Most important of all, the content of the image is one of if not the biggest aspect of determining whether or not the segmentation will be successful. This can be explained by using this coursework as well. For example, the segmentation for the image in Figure 2 is more accurate compared to the images in Figure 1 and 3 respectively. This is because the image in Figure 2 is much more simpler and less detailed compared to the images in Figure 1 and 3. This made it much more difficult to create a universal algorithm.

To conclude, for the automation of a segmentation algorithm for a general range of images, there will definitely be imperfections made to the segmentation. For example, if the sharpen filter utilizes a high value, then images in Figure 1 and 2 will provide a good segmentation but Figure 3 will suffer considerably. To avoid this unbalanced segmentation, the program has been finely tuned so that I will get the most balanced segmentation output of all of the images that are given.

6. References

1. Seal, A., Das, A., & Sen, P. (2015). Watershed: an image segmentation approach. *International Journal of Computer Science and Information Technologies*, *6*(3), 2295-2297.
2. View and set the current colormap. Retrieved 15 April 2022, from https://www.mathworks.com/help/matlab/ref/colormap.html
3. Extended-minima transform. Retrieved 15 April 2022, from https://www.mathworks.com/help/images/ref/imextendedmin.html#:~:text=%2CH%2Cconn)-,Description,h%20is%20a%20nonnegative%20scalar.
4. Create label matrix from bwconncomp structure. Retrieved 15 April 2022, from https://www.mathworks.com/help/images/ref/labelmatrix.html
5. Impose minima. Retrieved 15 April 2022, from https://www.mathworks.com/help/images/ref/imimposemin.html
6. Distance transform of binary image. Retrieved 15 April 2022, from https://www.mathworks.com/help/images/ref/bwdist.html
7. Extract objects from binary image by size. Retrieved 15 April 2022, from https://www.mathworks.com/help/images/ref/bwareafilt.html