1. Preprocessing Techniques

1.1. Image 1 (StackNinja1.bmp)

Firstly, read the original image of the tip of plant root 1

Original Image

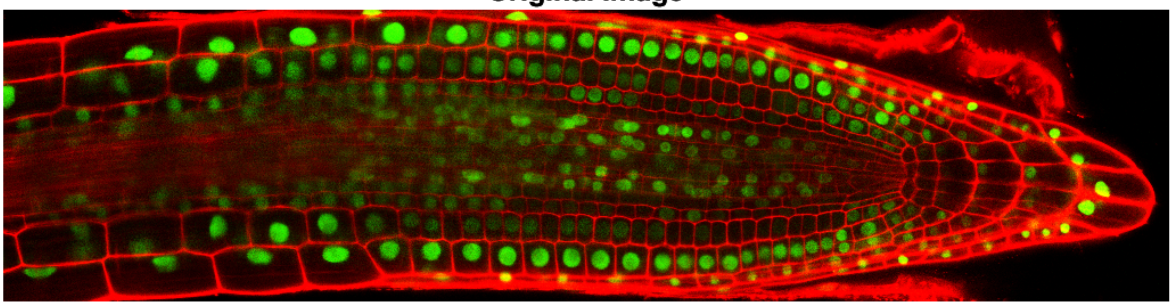


Figure 1.1.1

The first pre-processing technique for image 1 is to brighten up the whole image by using *imlocalbrigthen* with the default value 1. *Imlocalbrigthen* brightens the low-light areas of the original image from *Figure 1.1.1.*

Brigthen Image

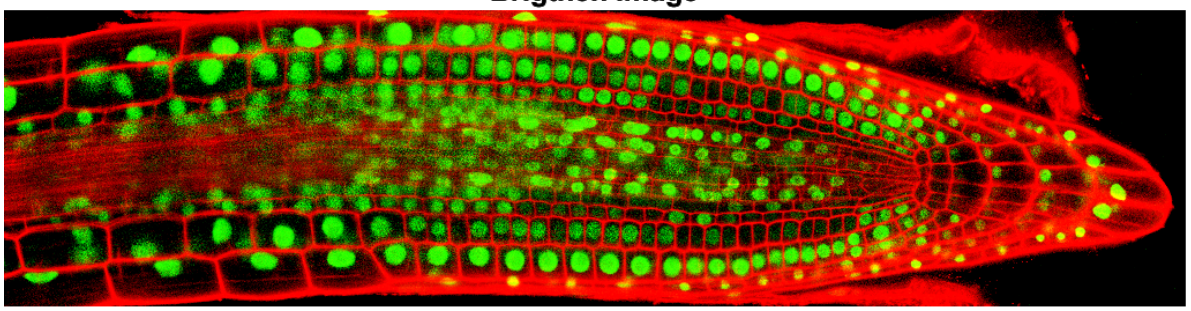


Figure 1.1.2

Next, choosing *HSV* as our colour space to extract the green channel of the image. The reason why using HSV is to extract the hue channel from the RGB image *Figure 1.1.2* by converting it into HSV color space, *Figure 1.1.3*. After extracting the hue channel, a binary mask is created to identify the green pixels in the image. The green pixels are those that fall within the range of *0.17 to 0.5* in the hue channel as shown in *Figure 1.1.4.*

Hue channel

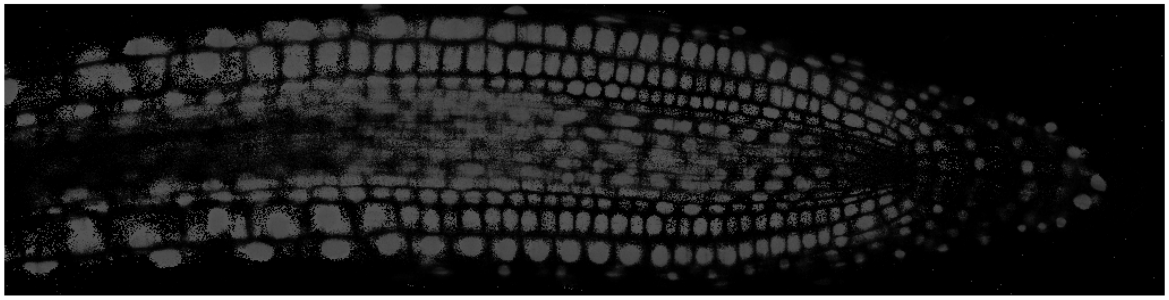


Figure 1.1.3

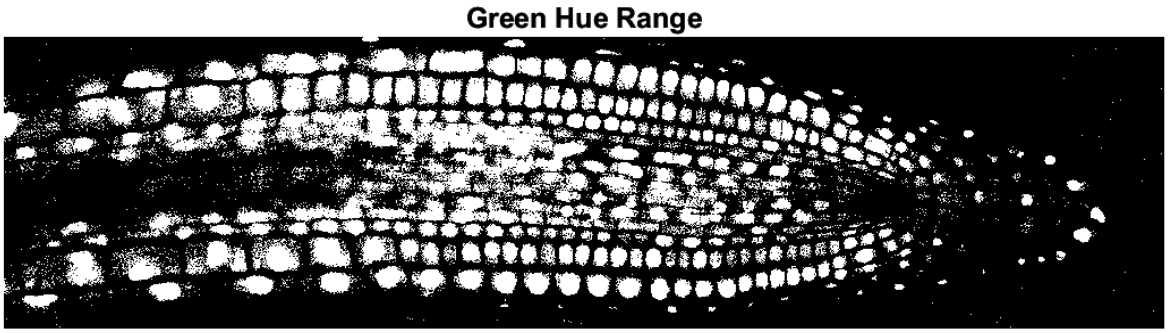


Figure 1.1.4

To apply this binary mask to the input image, the *green range* mask was repeated three times to match the dimensions of the HSV image using the *repmat()* function and stored in a new binary mask. After that, contrast stretching is applied to the hue, saturation, and value channels of the HSV image by multiplying the values of pixels that are not in the green range with a factor of *'1.5'.* Finally, the new binary mask will be applied to the brightened image in *Figure 1.1.2.* using element-wise multiplication with the *bsfun()* function, and processing a masked image in *RGB* form. The masked image shows only the green areas of the original image in grayscale, while the non-green areas are set to black as shown in *Figure 1.1.5.* The reason why the green area is used and found is that the tip of the plant root that we are interested to segment out is green color. The values of the pixels of the cell nuclei in the green channel would be higher compared to other regions and this would make it easier to binarize the image later.

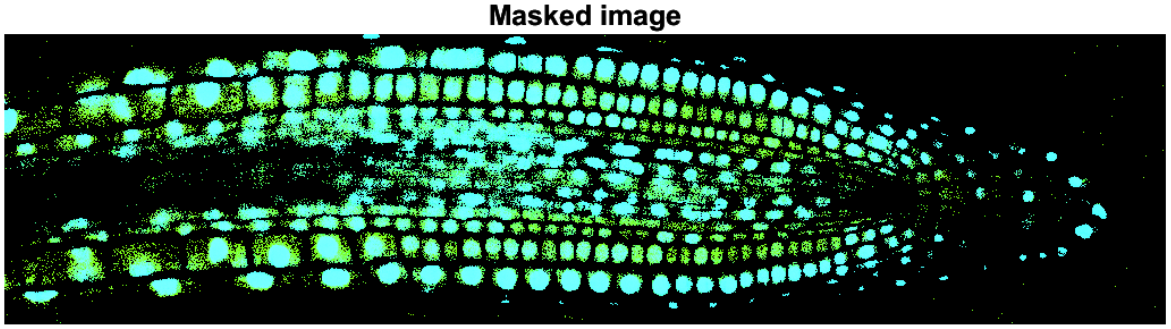


Figure 1.1.5

Furthermore, grayscaling the masked image as shown in *Figure 1.1.6* using rgb2gray and applying Median Filtering, *medfilt2* to reduce the "salt and pepper" noise shown in *Figure 1.1.7* with [3,3].

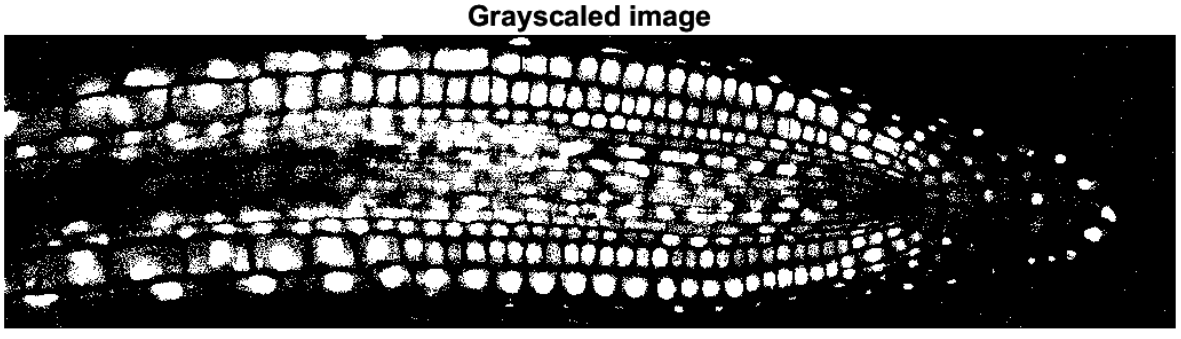


Figure 1.1.6

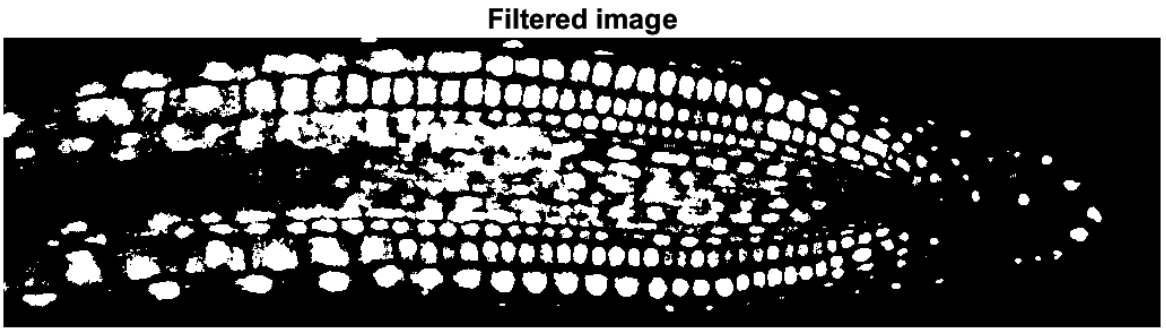


Figure 1.1.7

Applying adaptive thresholding to the filtered grayscale image to segment the image into binary regions with *adapthresh* function as shown in *Figure 1.1.8. Figure 1.1.9* is a binary image obtained by using the *imbinarize* function to binarize the grayscale and filtered image from *Figure 1.1.7* using the adaptive threshold from *Figure 1.1.8.*

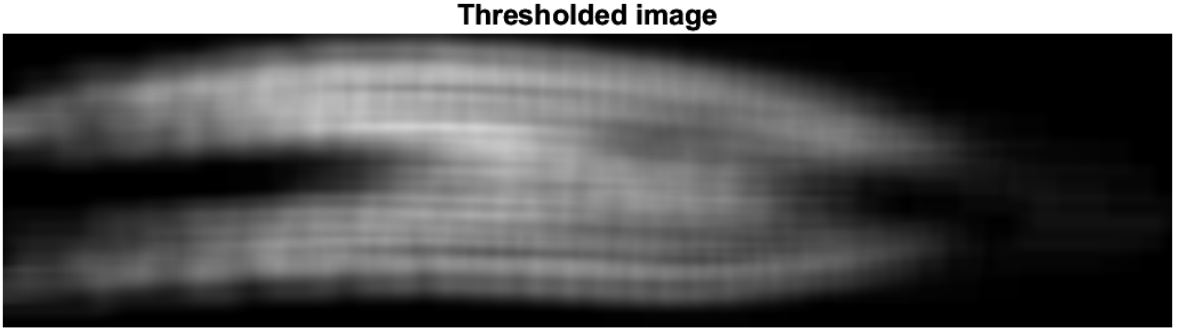


Figure 1.1.8

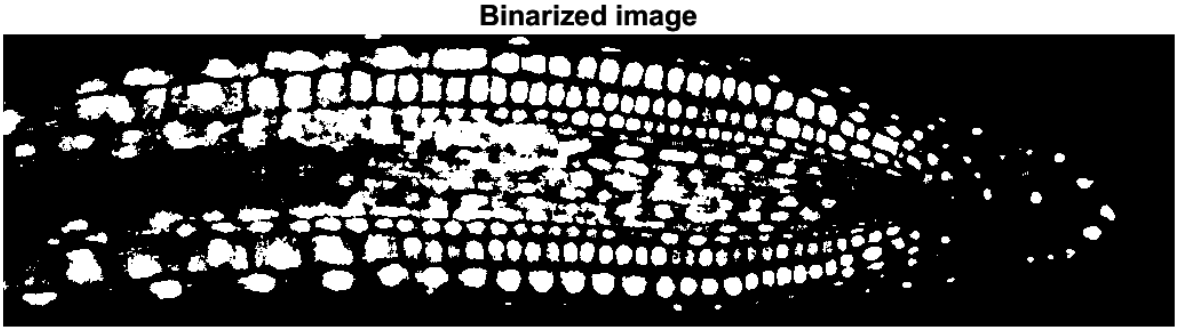


Figure 1.1.9

For Segmentation, I have decided to go with Watershed Segmentation. In *Figure 1.1.9,* there is a lot of extracted touching nuclei with one another. So, this algorithm is a useful algorithm for segmentation when extracting touching or overlapping objects in the image when doing preprocessing, such as the image in *Figure 1.1.10*. (Bandara, n.d.)

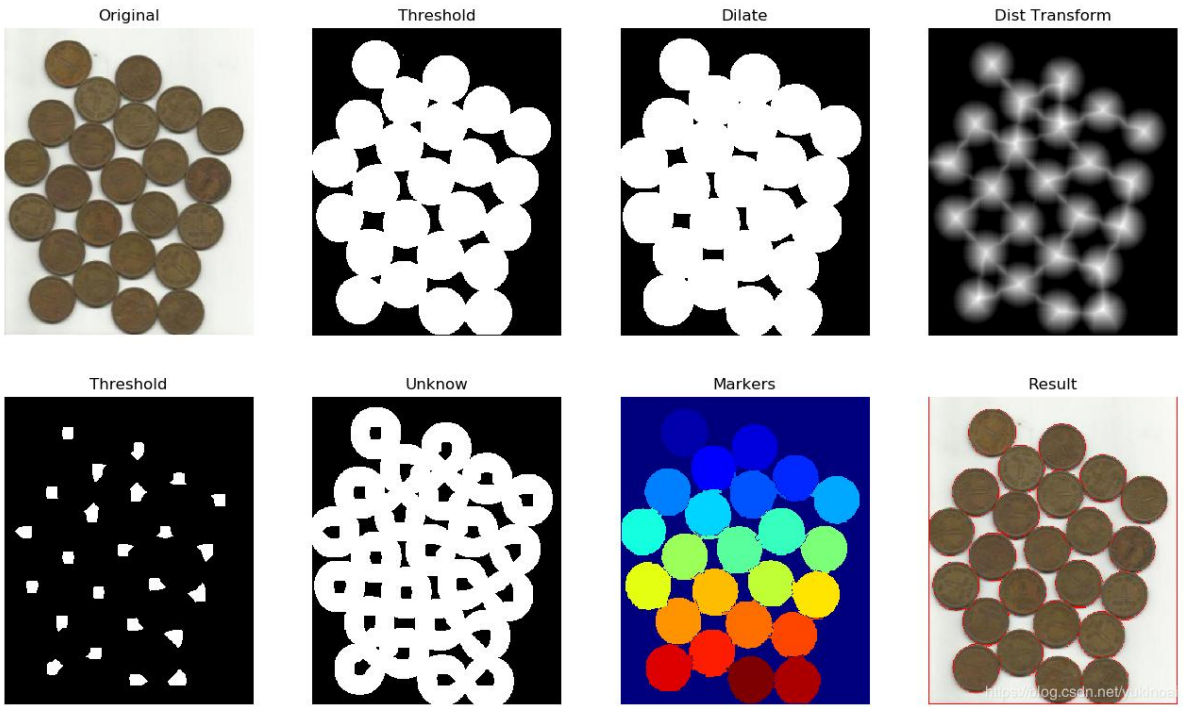


Figure 1.1.10

The steps to perform Watershed Segmentation are as follows:

According to Mathworks about Marker-Controlled Watershed Segmentation, firstly, calculate the distance transform of the complement of the binary image containing the outline of the cell nuclei by using the *bwdist* function. The distance transform of an image calculates the distance of each pixel in image 1 to the nearest non-zero pixel in the binary image. So, negating the distance transform to convert it into a gradient image, where the boundaries between objects have high-intensity values.

Next, calculate the watershed transform. The Matlab, *watershed* function is applied to the negated distance transform. This algorithm treats the intensity values of the gradient image as elevations and fills up basins starting from regional minima until the boundaries of the basins meet.

The *watershed* function assigns a label to each region and uses the *label2rgb* function to visualize the result, where each region is assigned a different color.

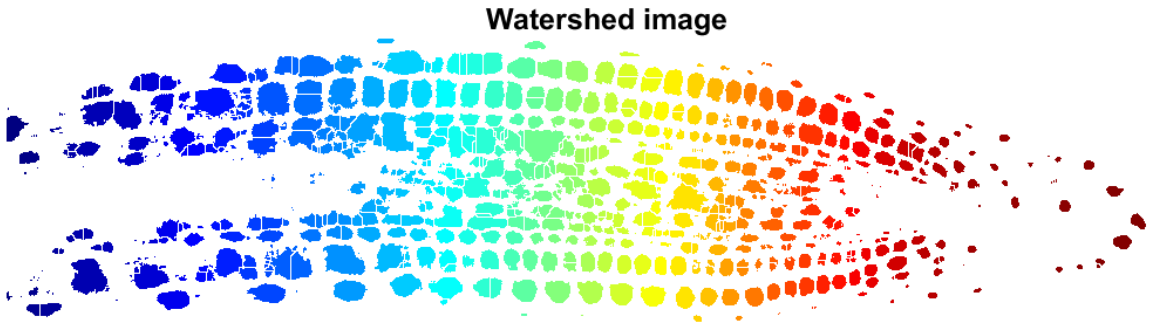


Figure 1.1.11

Since the resulting image has a black background and colored regions, the code converts the image to grayscale using the *rgb2gray* function. Creating a binary mask by thresholding the grayscale image with a threshold value of 250. This threshold value separates the background of the image from the regions. Finally, sets the pixel values of the background to white (255) and the pixel values of the regions to black (0) using the binary mask to get *Figure 1.1.12.*

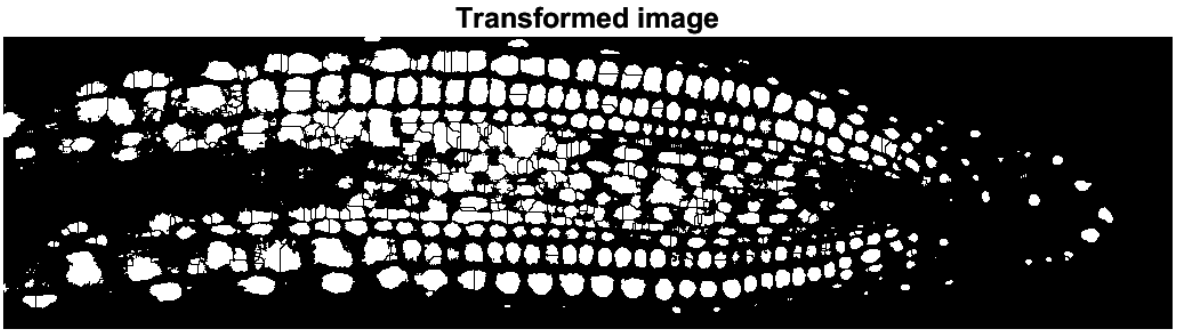


Figure 1.1.12

I use the Opening method for the morphological operation. The morphological opening is a morphological operation that involves erosion followed by the dilation of an image. It is used to remove small objects or details in a binary image.

The *strel* function is used to create a structuring element with a disk shape and a radius of 1 pixel. The *imopen* function is then used to perform the opening operation on the binarized image with the smaller noise removed and shrinking the remaining cell nuclei as shown in *Figure 1.1.13.*

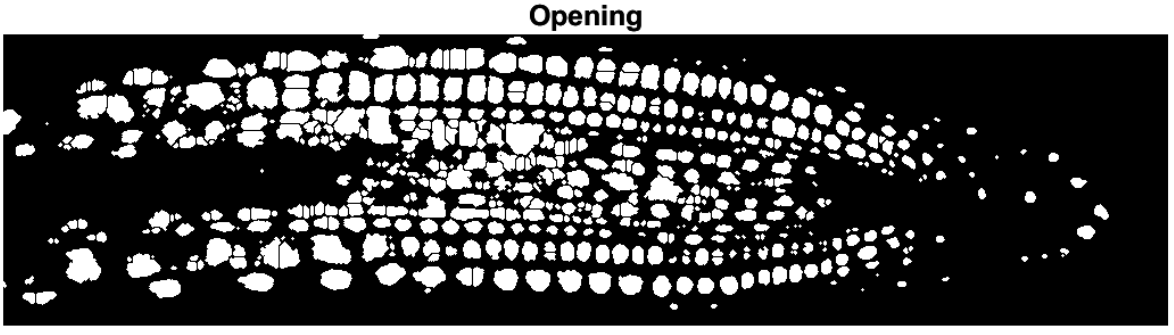


Figure 1.1.13

Lastly, I use *connected components labeling* methods to perform region-of-interest image processing to randomly color regions that correspond to nuclei *bwconncomp* function is used to label connected components in the binary image. Next, calculate the number of connected components. A color map will be created using the *rand* function with the size of the number of connected components and 3 for RGB values. Then, color each connected component in the image based on the corresponding label matrix with the number of the component and the color maps created earlier. Finally, each connected component colored in a random color

will represent each of the cell nuclei that were detected. *Figure 1.1.14* will be the end result of the preprocessing

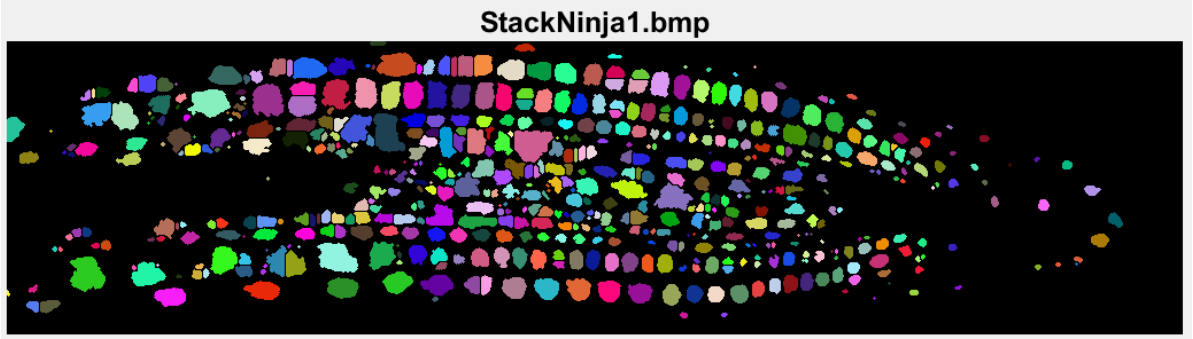


Figure 1.1.14

The same preprocessing method will be implemented for image 2 (StackNinja2.bmp) and image 3 (StackNinja3.bmp). *Figure 1.1.15* is the result after preprocessing techniques go through image 2 (StackNinja2.bmp) and *Figure 1.1.16* will be the result for image 3 (StackNinja3.bmp)

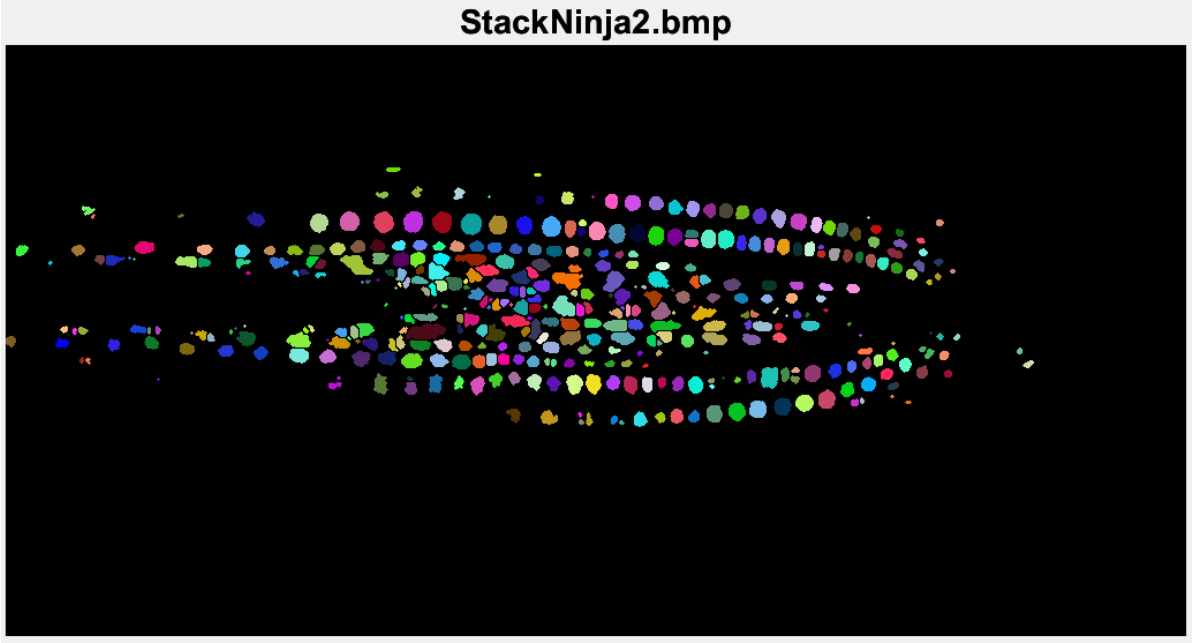


Figure 1.1.15

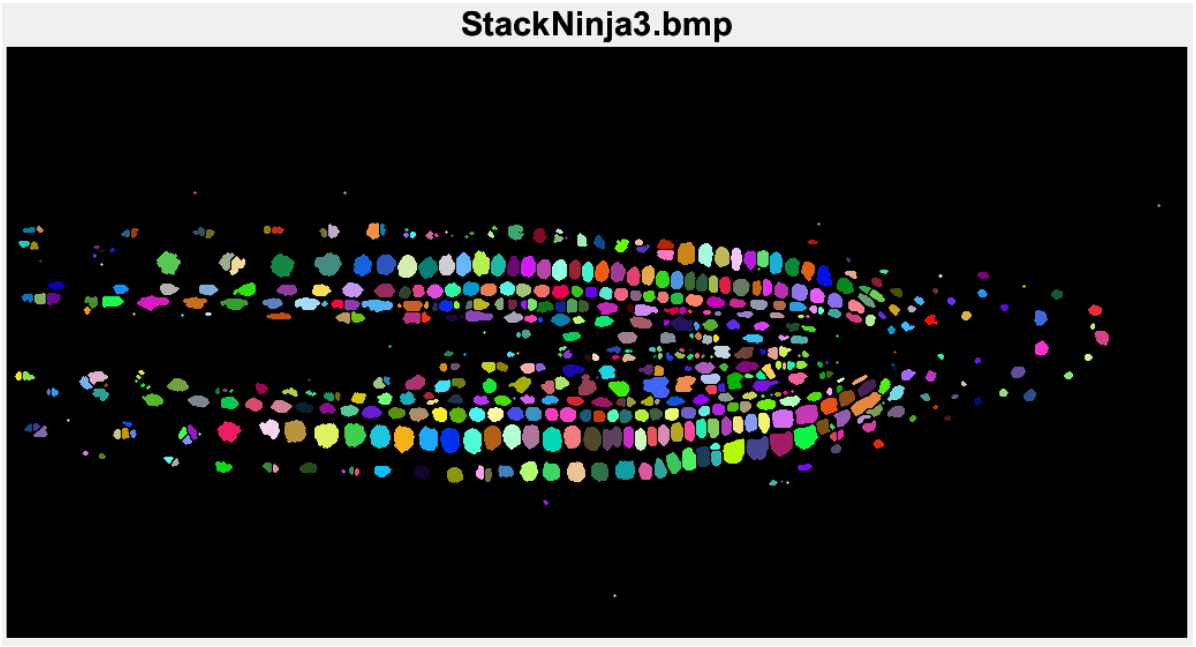


Figure 1.1.16

2. Critical Analysis

2.1. Color Space Conversion and Color-Based-Segmentation (Green Mask)

In color-based segmentation, *HSV* and *L*A*B** are two frequently chosen color spaces for color image segmentation research. Bora, Kumar Gupta, and Khan. (2008) stated that *HSV* color space performs better than *L*A*B** color space. Through their experimental results, *HSV* has a high PSNR (Peak Signal to Noise Ratio) value and a low MSE (Mean Squared Error) value. So, *HSV* color space will be more suitable for dealing with the segmentation of noisy color images and it is easier to implement. The reason why *HSV* is used is that the original image for the three tips of the cell plant only consists of 2 different regions of color, red and green (). With HSV, it can extract the green range from the hue channel which varies from 60 to 180 degrees out of 360' which is *0.17 to 0.5 (Yang et al., 2015),* on the nuclei of the tip of the 3 different plants as in *Figure 2.1.1, Figure 2.1.2* and *Figure 2.1.3*

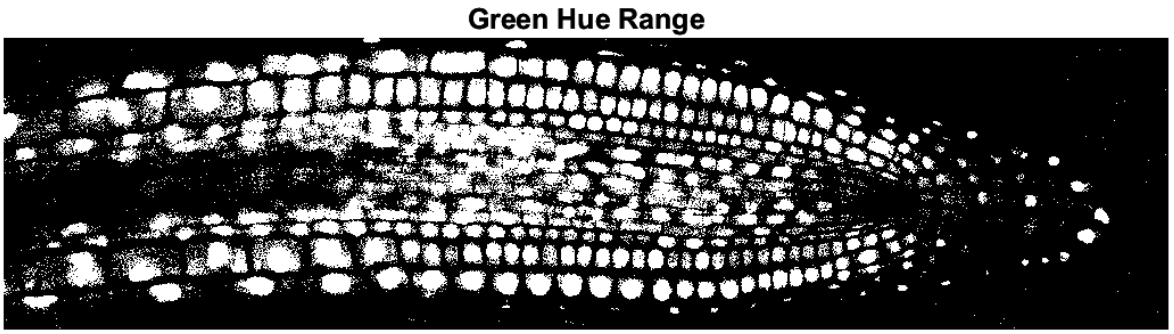


Figure 2.1.1

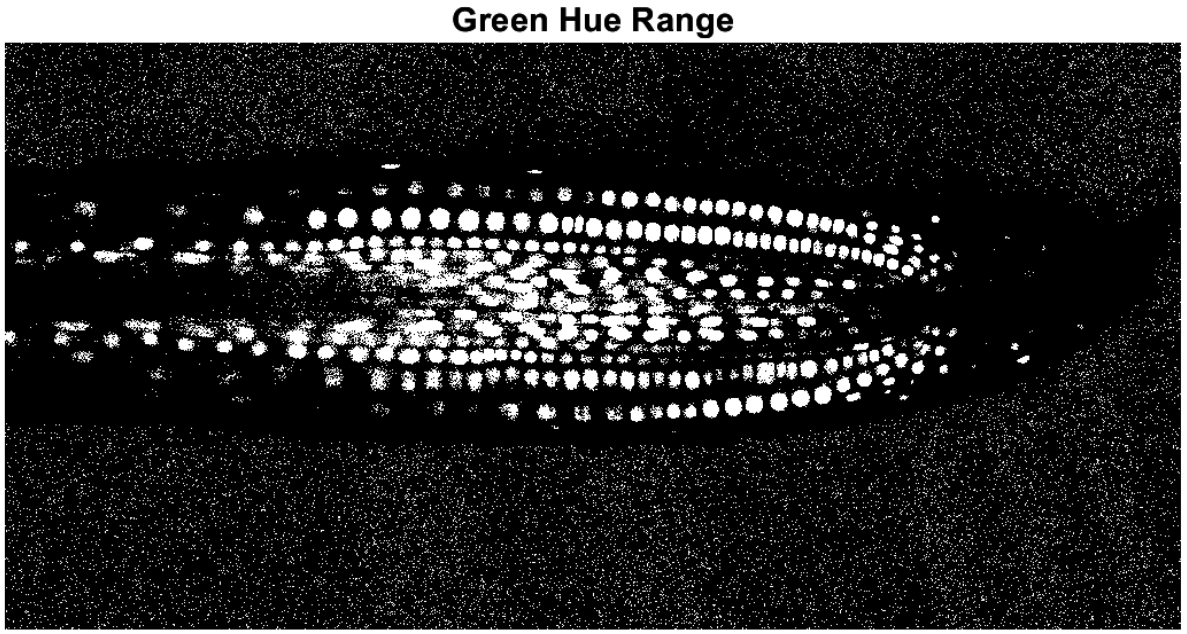


Figure 2.1.2

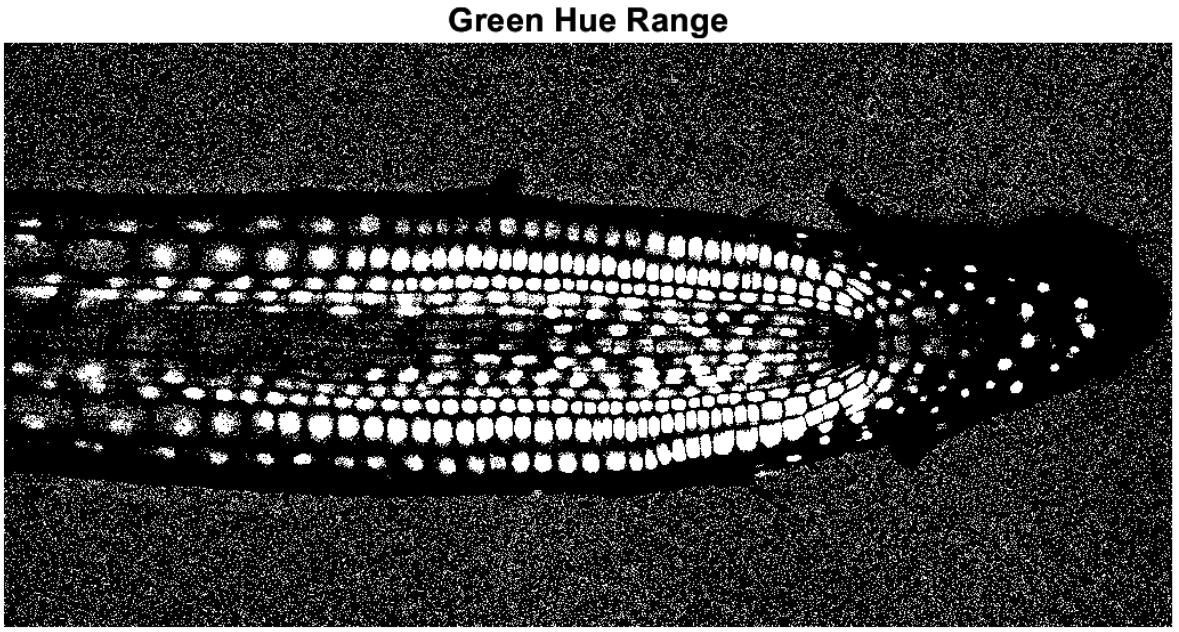


Figure 2.1.3

From the Mathworks website for Segment Image and Create Binary Mask, I have decided to create a binary mask to set the hue values outside of the green range to 0 and the hue values inside the green range to 1. This mask is then replicated across all color channels to match the size of the original image, creating a 3-channel mask. By using this method, I can selectively process only the regions of interest that are green in the image while leaving the rest of the image unaffected and not adding any extra noise as shown below in *Figure 2.1.4, Figure 2.1.5,* and *Figure 2.1.6*

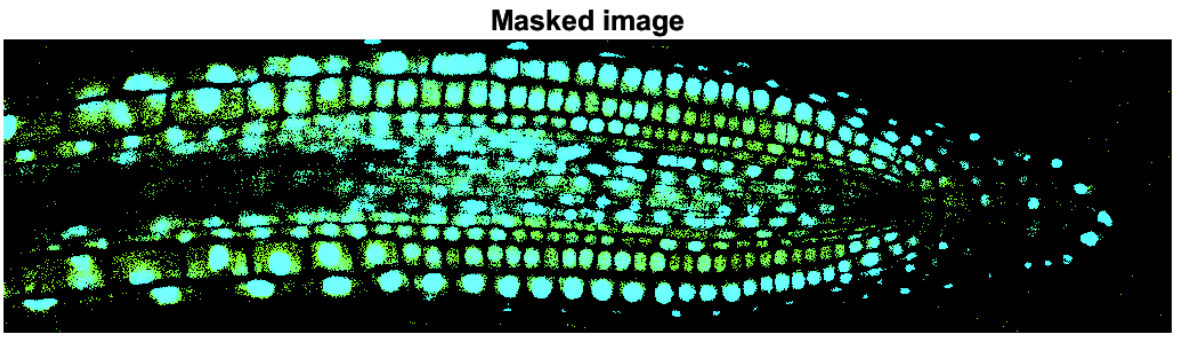


Figure 2.1.4

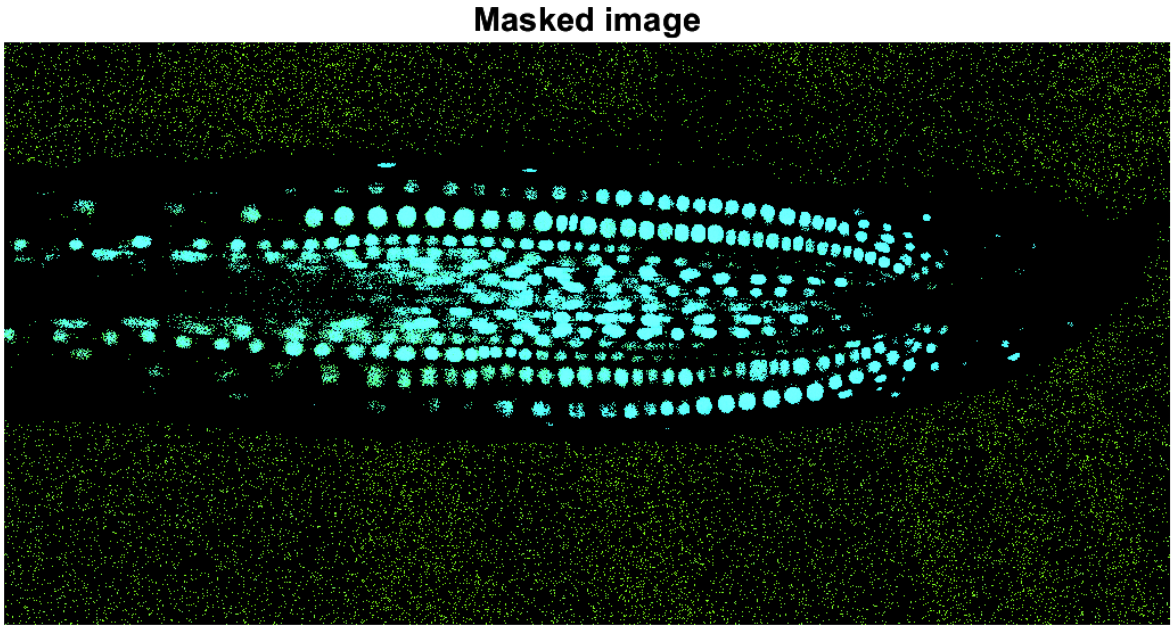


Figure 2.1.5

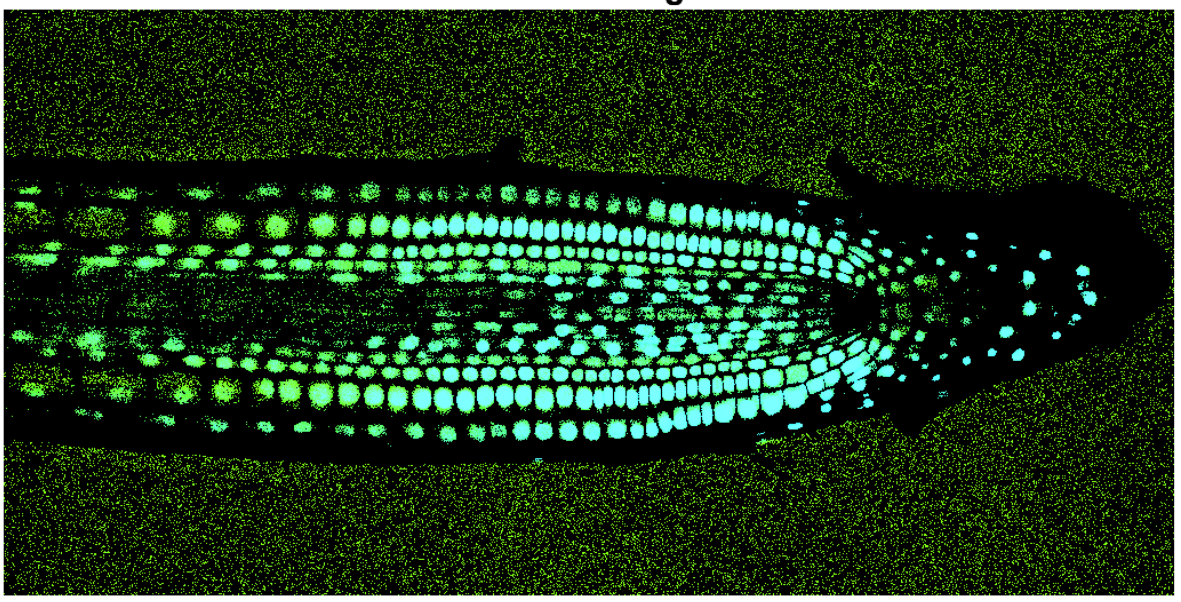


Figure 2.1.6

However, the limitation is if the green object is under a different lighting condition, the hue channel may not accurately capture the green color range. Additionally, this method assumes that the green object of interest falls within the hue range of 0.17 to 0.5, which may not always be the case. Therefore, it is important to evaluate the performance of this method on different green objects and under different lighting conditions.

2.2. Contrast Stretching

The strength of contrast stretching is it improves the overall contrast and brightness of an image, making it easier to distinguish different objects or features within the image. This is especially helpful for the nuclei green that have low contracts and also low light conditions as shown in *Figure 2.2.1* and *Figure 2.2.2* for image 3 (StackNinja3.bmp). Figure 2.2.3 will be the result.

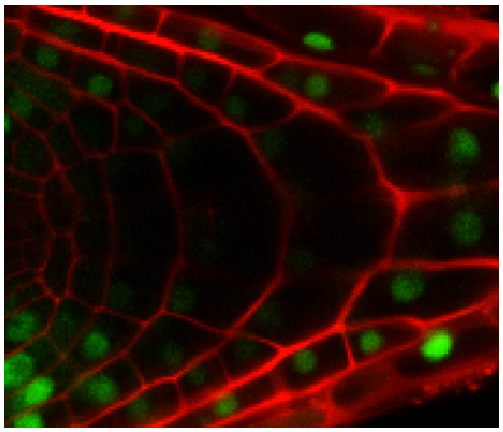


Figure 2.2.1

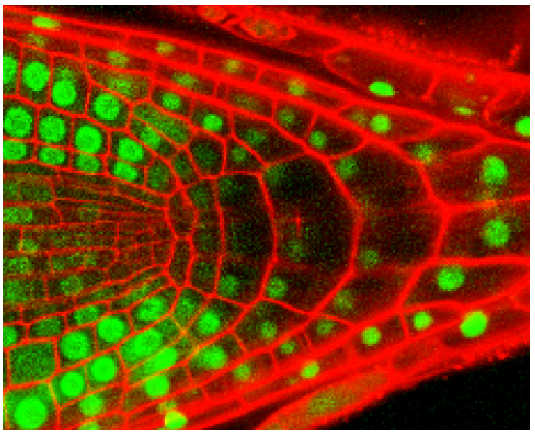


Figure 2.2.2

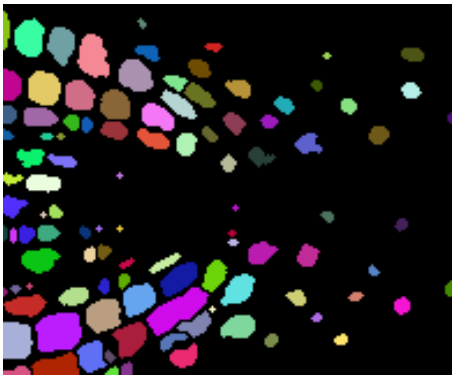


Figure 2.2.3

Limitation of contrast stretching is the user is required to use trial and error to get the desired and optimum contract values of the image.

2.3. Median Filtering (medfilt2)

Median filtering effectively removed the 'salt and pepper' noise. These three original Images, of them, have a lot of 'salt and pepper noise' as shown below in *Figure 2.3.1, Figure 2.3.3,* and *Figure 2.3.5.* After using median filtering, most of the 'salt and pepper' noise is removed. *Figure 2.3.2* is image 1 which has mostly removed all the 'salt and pepper noise' However, *Figure 2.3.4* and especially in Figure *2.3.6* show the limitation of median filtering which is not

all the salt and pepper noise is removed from the image. The solution for this is using the *Opening* method to remove all the other noise. *Figure 2.3.7* is the result after Image 3 used the opening method, all the ‘salt and pepper’ noise is removed.

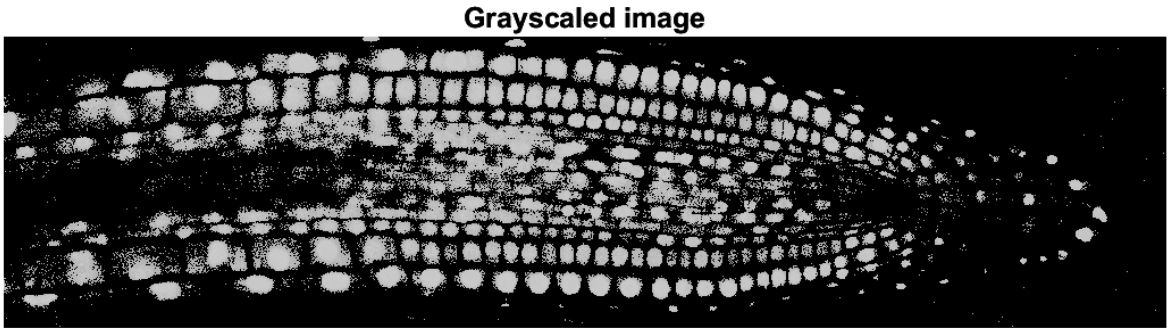


Figure 2.3.1

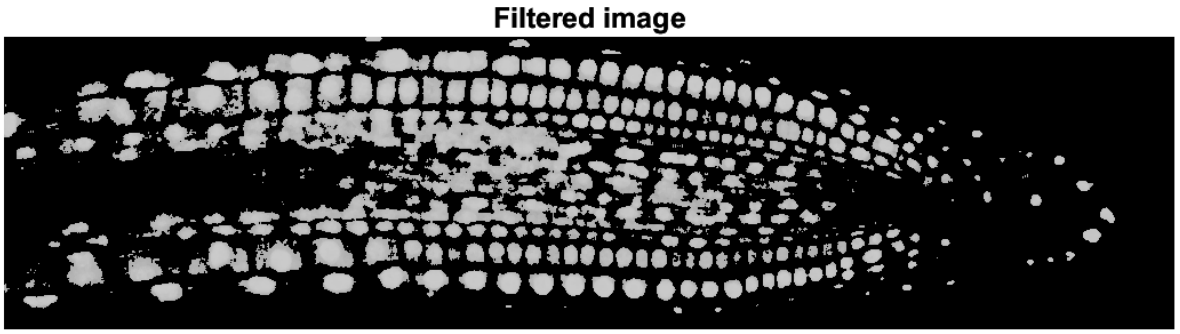


Figure 2.3.2

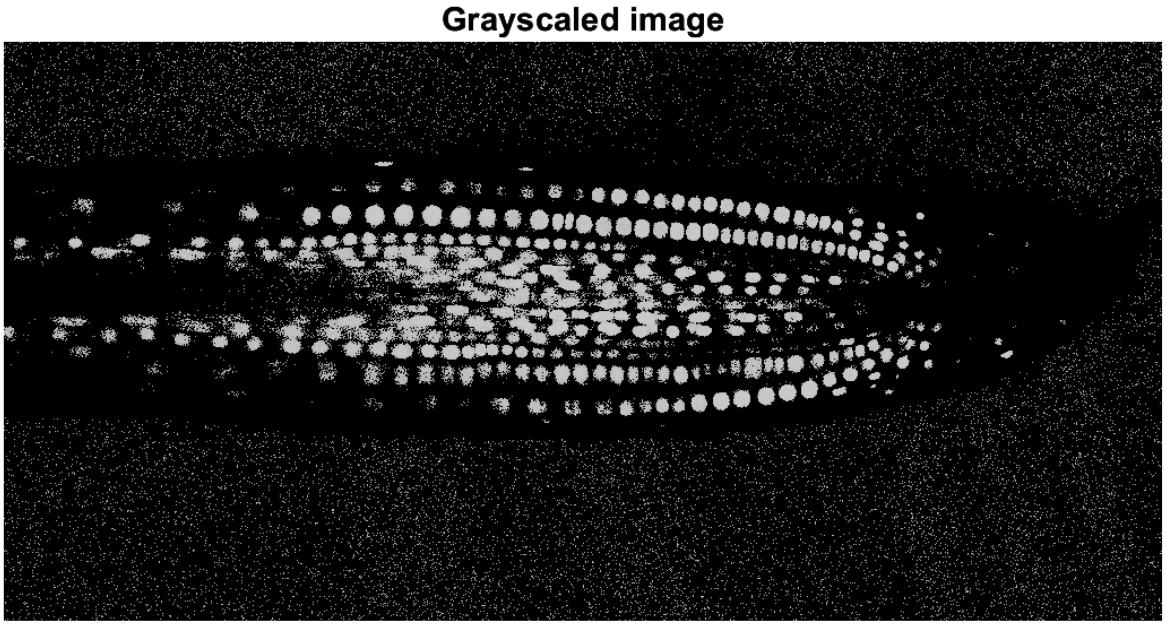


Figure 2.3.3

Filtered image

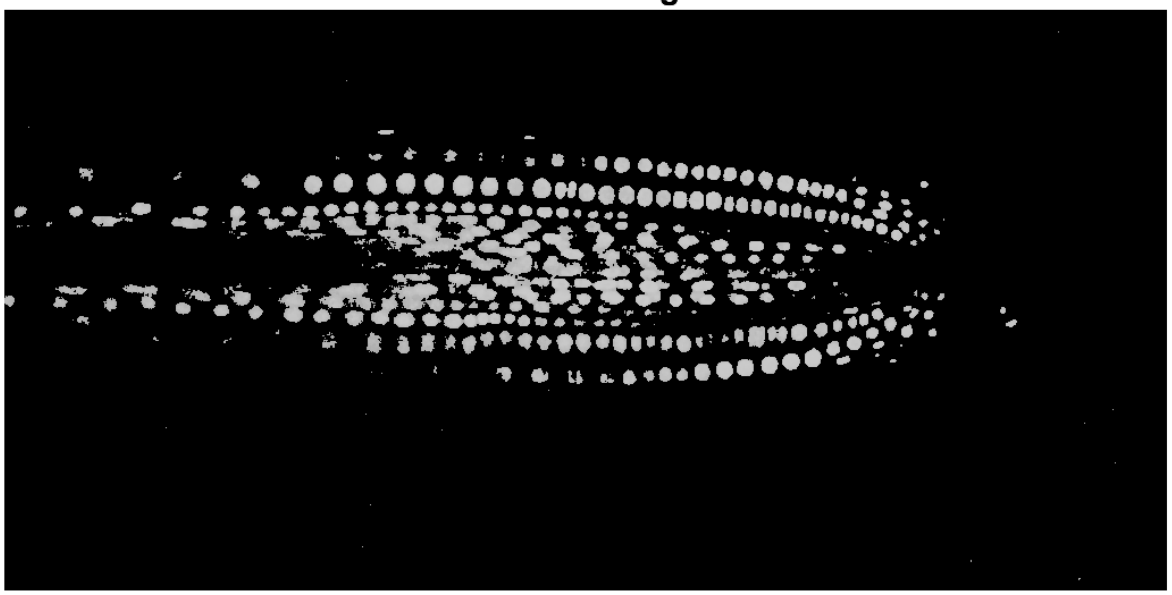


Figure 2.3.4

Grayscaled image

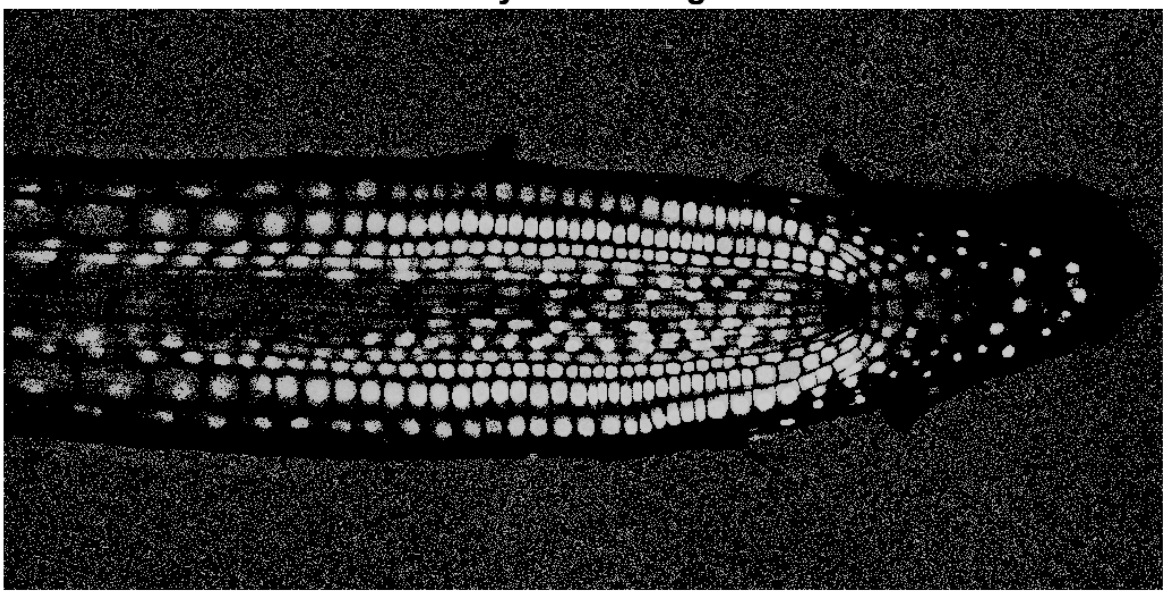


Figure 2.3.5

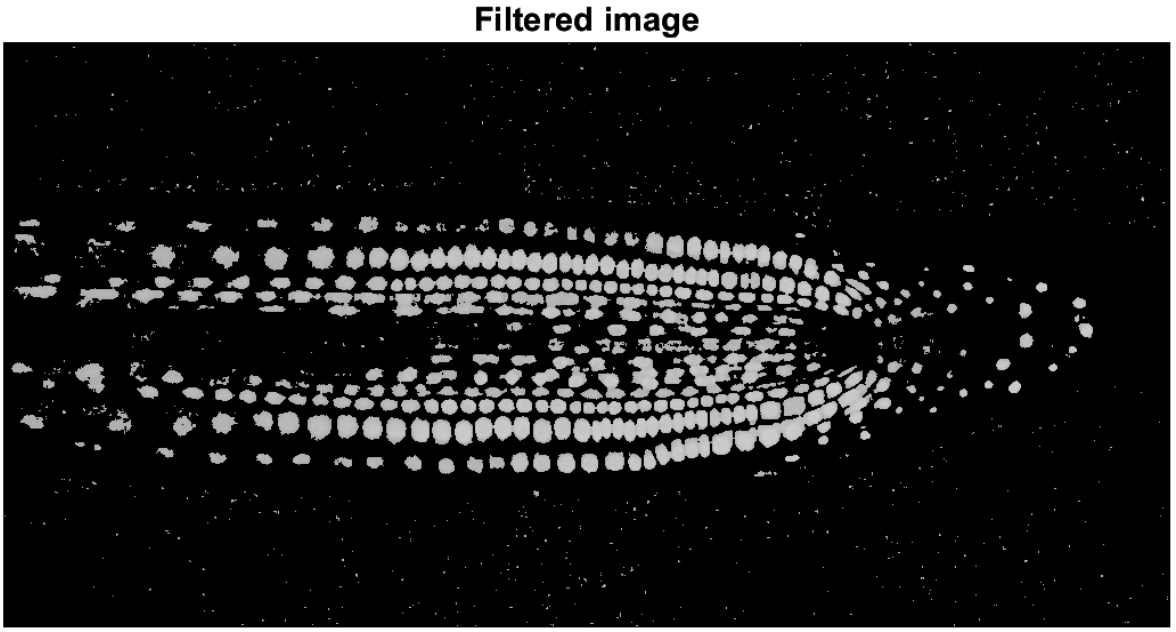


Figure 2.3.6

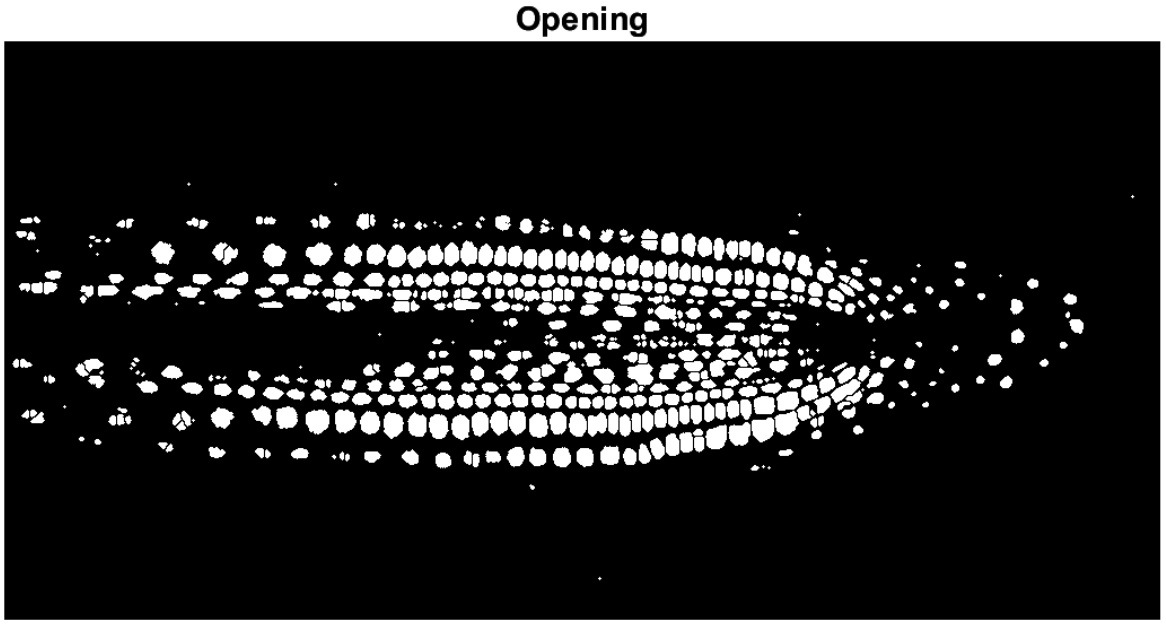


Figure 2.3.7

2.4. Distance Transform

Using image 3 (StackNinja3.bmp) as an example (*Figure 2.4.1),* the distance method that I used is the Euclidean method as shown in *Figure 2.4.4*. This is because the Euclidean distance method works better than other distance methods like city block and chessboard. The Chessboard method does not separate the nuclei based on the image as shown in *Figure 2.4.2* City blocks separate too many green objects and a lot of unnecessary separation as shown in *Figure 2.4.3*.

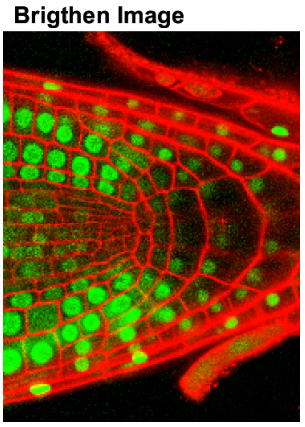
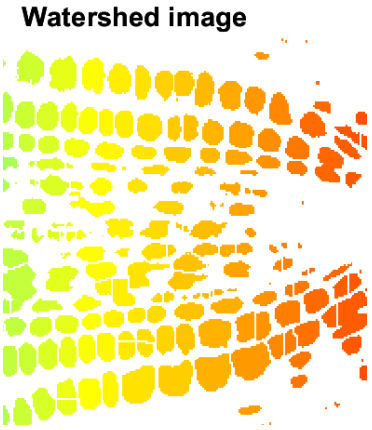
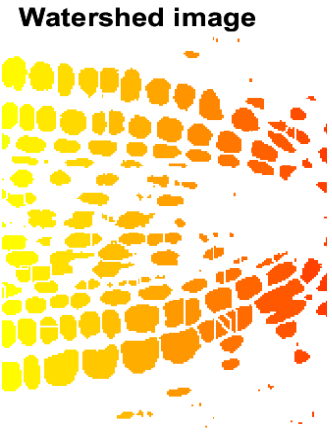


Figure 2.4.1

Chessboard (*Figure 2.4.2)*



City Block (*Figure 2.4.3)*



Euclidean(*Figure 2.4.4)*

So, comparison with visualizing the original image as the example of *Figure 2.4.1,* Euclidean is the best distance method among the other two other methods. However, simple thresholding may not always produce satisfactory results, and additional image processing may be required before applying the distance transform. Hence, if the preprocessing steps are insufficient, the outcome of the distance transform may not be optimal.

2.5. Watershed Segmentation

From the final images shown above (*Figure 1.1.14, Figure 1.1.15,* and *Figure 1.1.16*), we can see that watershed segmentation is able to locate and separate most of the distinct leaves. This shows that the watershed algorithm is high in precision and efficiency. Watershed segmentation is able to separate out regions that are connected together in the binary image of the nuclei cell. *Figure 2.5.1* shows that before image 3 (StackNinja3.bmp) and after the watershed, the individual cell nuclei have their own region and are not connected to each other.

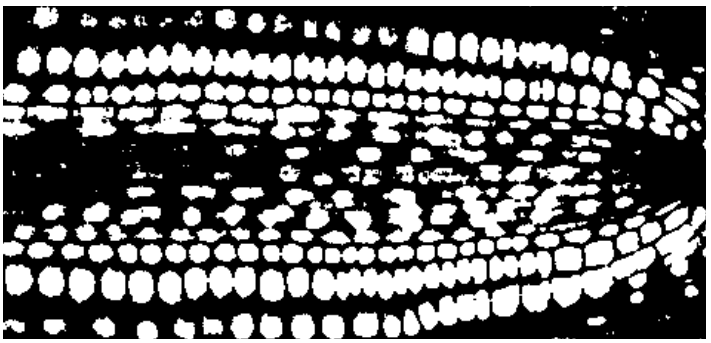


Figure 2.5.1

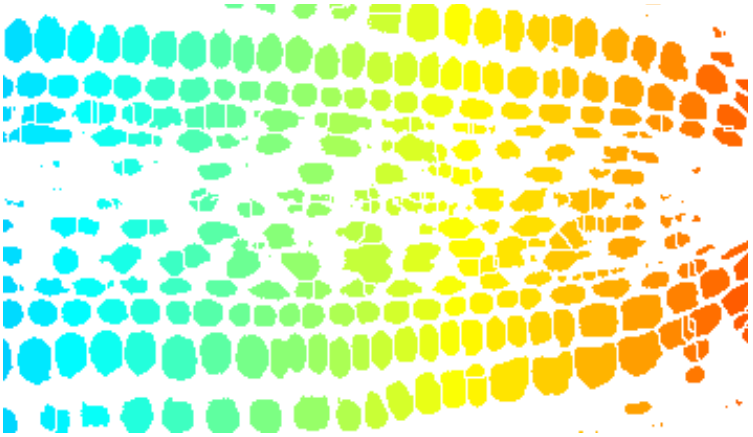


Figure 2.4.2

According to Nicket & Rameh (2013), the Watershed Transform has a major drawback which is its tendency to produce excessive over-segmentation in most natural images. This is because the algorithm can't classify the green area correct objects, resulting in every regional minimum forming its own catchment basin, even if it is tiny and insignificant. As shown in Figure 2.5.3, Figure 2.5.4, and Figure 2.5.5, some colors on the nuclei cell appear disconnected and are segmented into 2 or 3 different regions by the Watershed Transform due to over-segmentation.

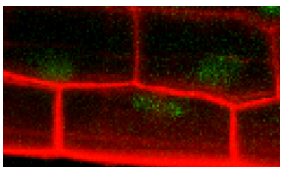


Figure 2.5.3

Figure 2.5.4

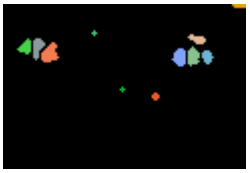


Figure 2.5.5

While most of the leaves can be accurately segmented, there are some cases where the edges between leaves do not match the true outline of the leaves in the original image. This can be observed in Figure 2.5.6. This issue may occur when the cell nuclei have low contrast or brightness, or when they are not well-defined. When the image is binarized, the lighter green cell nuclei may be incorrectly considered as noise and filtered out. Consequently, when the watershed algorithm is applied to the binary image, there may be some distortion in the edges of the leaves, as demonstrated in Figure 2.5.7.

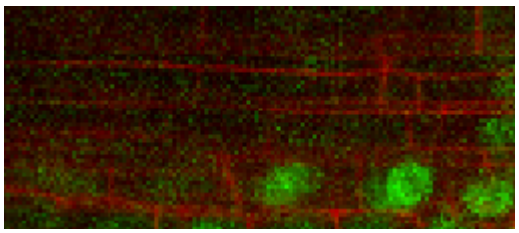


Figure 2.5.6

Figure 2.5.7

There can be more improvement will a better segmentation algorithm to use in this project.

3. Reference list

1. Bandara, R. (n.d.). *Image Segmentation using Unsupervised Watershed Algorithm with an Over-segmentation Reduction Technique*. [online] Available at: https://arxiv.org/ftp/arxiv/papers/1810/1810.03908.pdf [Accessed 9 Apr. 2023].
2. Bora, D., Kumar Gupta, A. and Khan, A.F. (2008). Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation International Journal of Emerging Technology and Advanced Engineering Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation. *Certified Journal*, 9001(2).
3. Niket Amoda & Ramesh K Kulkarni. (2013).Efficient Image Segmentation Using Watershed Transform. Retrieved from http://ijcst.com/vol42/2/niket.pdf
4. MathWorks (2019). *Brighten low-light image - MATLAB imlocalbrighten*. [online] www.mathworks.com. Available at: https://www.mathworks.com/help/images/ref/imlocalbrighten.html.
5. MathWorks (2022). *2-D median filtering - MATLAB medfilt2*. [online] www.mathworks.com. Available at: https://www.mathworks.com/help/images/ref/medfilt2.html.
6. MathWorks (n.d.). *Marker-Controlled Watershed Segmentation - MATLAB & Simulink Example*. [online] www.mathworks.com. Available at: https://www.mathworks.com/help/images/marker-controlled-watershed-segmentation.html.
7. MathWorks (2023). *Segment Image and Create Mask Using Color Thresholder App - MATLAB & Simulink*. [online] www.mathworks.com. Available at: https://www.mathworks.com/help/images/image-segmentation-using-the-color-thresholder-app.html [Accessed 9 Apr. 2023].
8. Yang, W., Wang, S., Zhao, X., Zhang, J. and Feng, J. (2015). Greenness identification based on HSV decision tree. *Information Processing in Agriculture*, [online] 2(3-4), pp.149–160. doi:https://doi.org/10.1016/j.inpa.2015.07.003.