1.0 Pre-processing

1.1 Image 1 – Stack Ninja 1

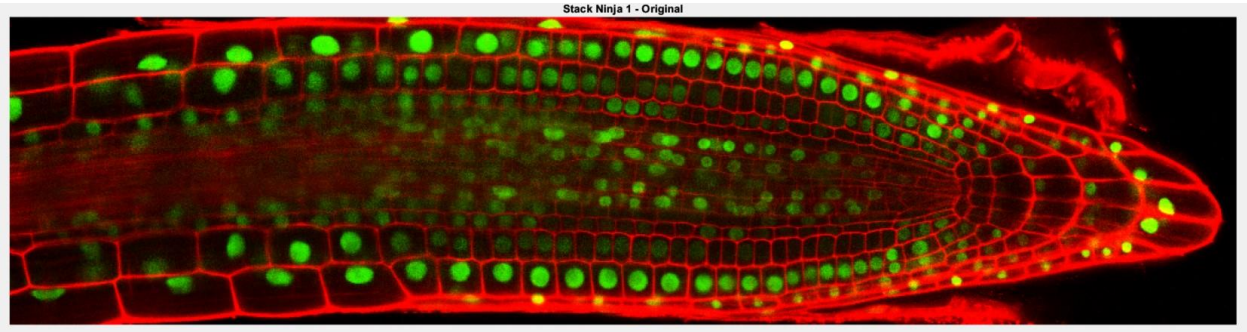

Figure 1.1.1 Original Stack Ninja Image 1

Read the original Stack Ninja 1 image at the beginning.

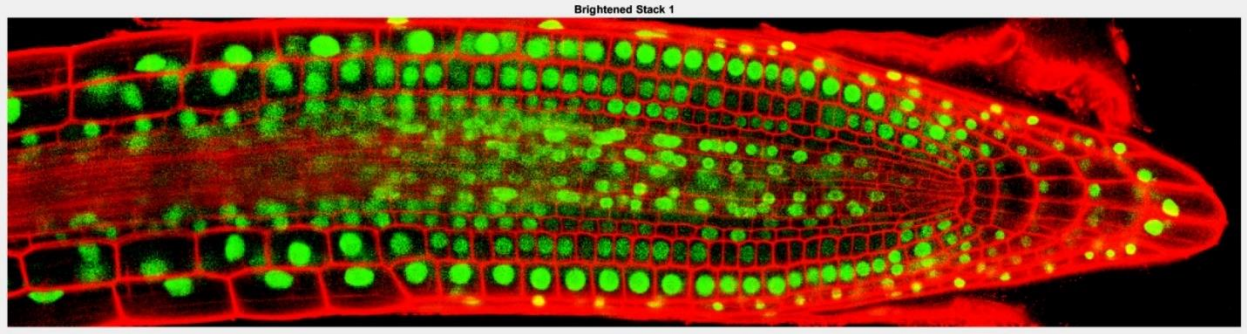

Figure 1.1.2 Brightened Stack Ninja 1

Then, the first step of pre-processing will be brightening up the image. The reason of brightening the image to ensure most of the darker cell nuclei are displayed clearly. Therefore, it is easier to extract the green colour nuclei from the darker part of the cell at the later pre-processing steps. The imlocalbrighten function is applied for brightening the image with the default value and Figure 1.1.2 is the output after applying the function.

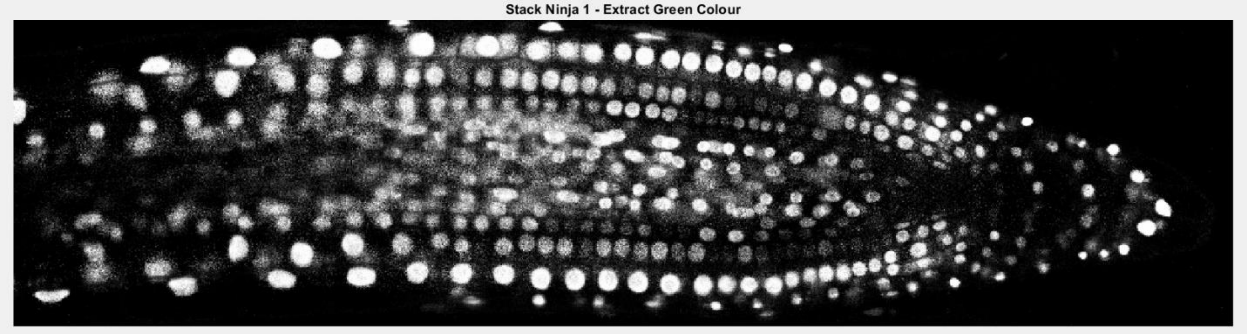

Figure 1.1.3 Green Channel of Stack Ninja 1

The next pre-processing step is extracting the green channel out from the brightened image Figure 1.1.2. The ‘greenness’ formula, Green – (Red + Blue) / 2, is applied for extracting the green channel.

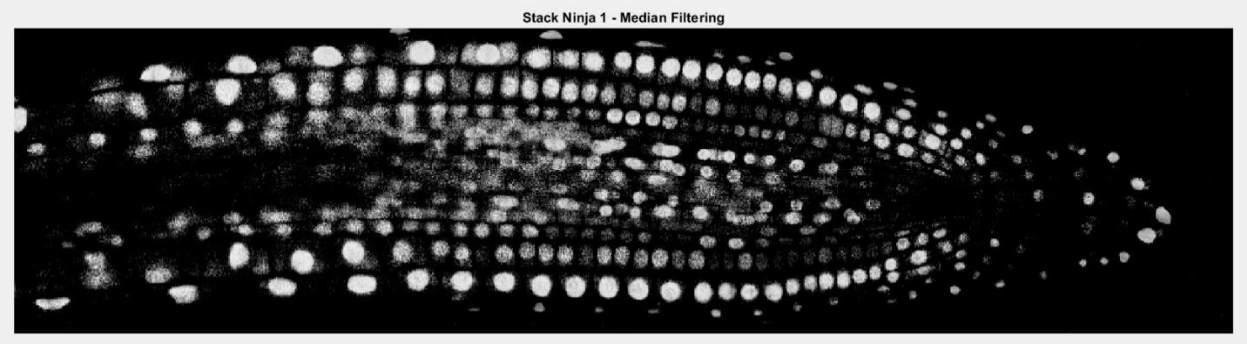

Figure 1.1.4 Median Filtering on Stack Ninja 1

Next, use median filtering function to minimise the noise of the image Stack Ninja 1 with the threshold of [1 1].

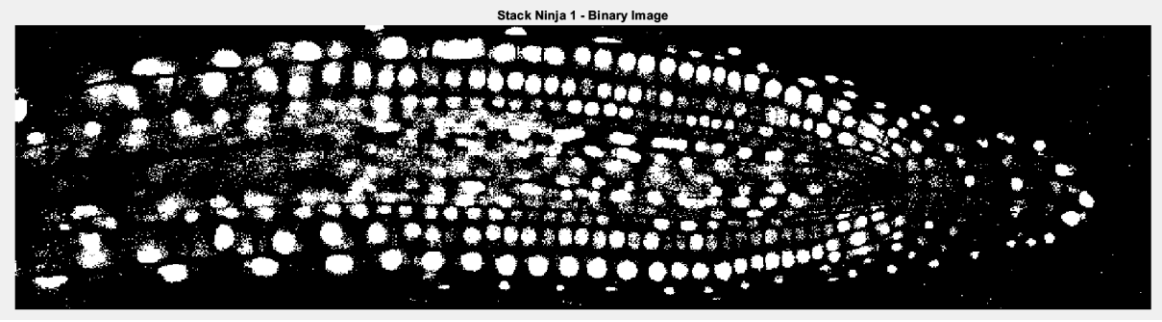

Figure 1.1.5 Binarize Stack Ninja 1

After filtering the image, binarize the filtered image using the function imbinarize.

1.2 Image 2 – Stack Ninja 2

Figure 1.2.1 Original Stack Ninja 2

Read the second image in the first step of pre-processing.

Figure 1.2.2 Brightened Stack Ninja

Increase the brightness of the second image using imlocalbrighten with default value.

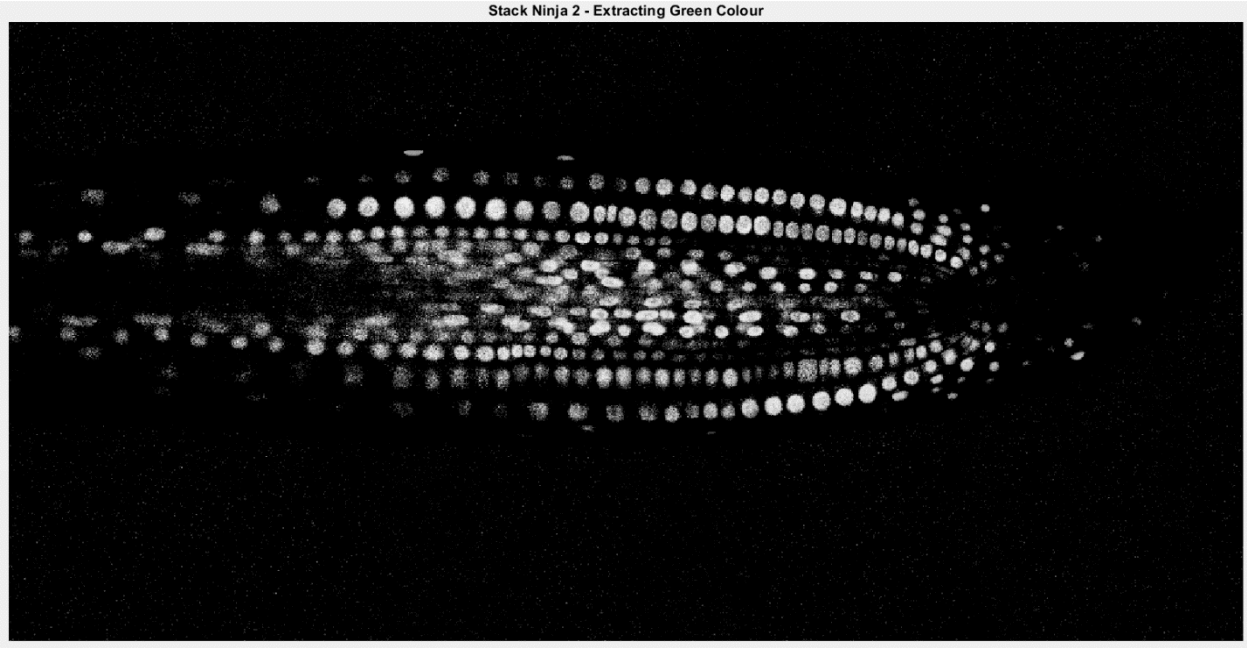

Figure 1.2.3 Green Channel of Stack Ninja 2

The next step will be extracting the green colour channel from the brightened image 2. Same as image 1, the green colour channel will be extracted using the formula, Green – (Red + Blue) / 2.

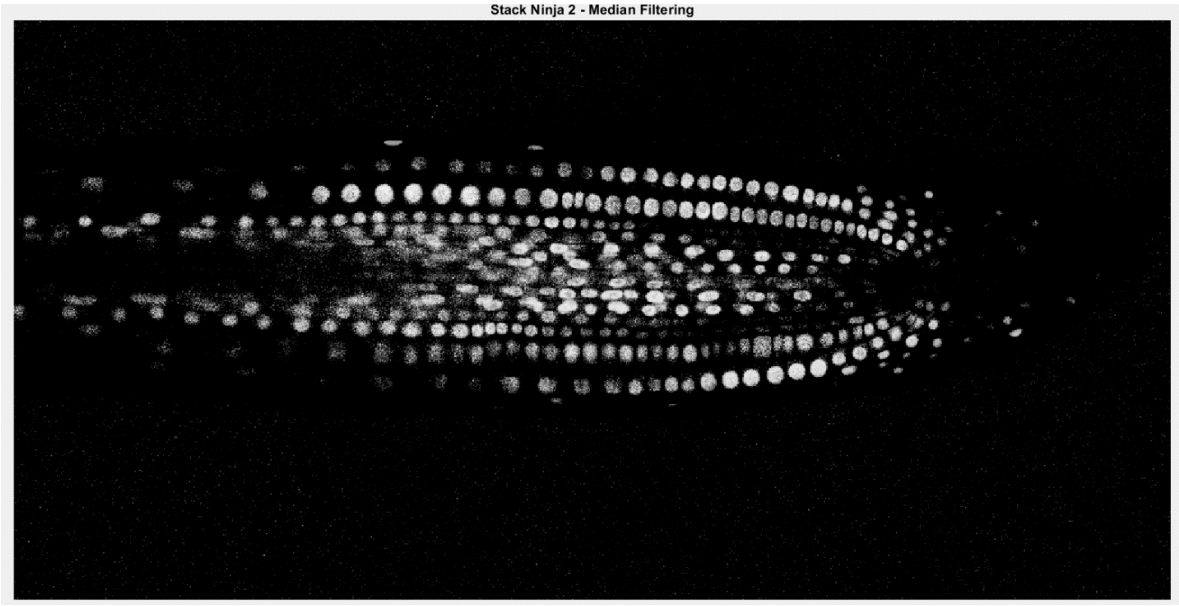

Figure 1.2.4 Median Filtering on Stack Ninja 2

After extracting the green colour channel from image 2, apply median filtering with threshold of [1 1] on image 2 to minimise the noise.

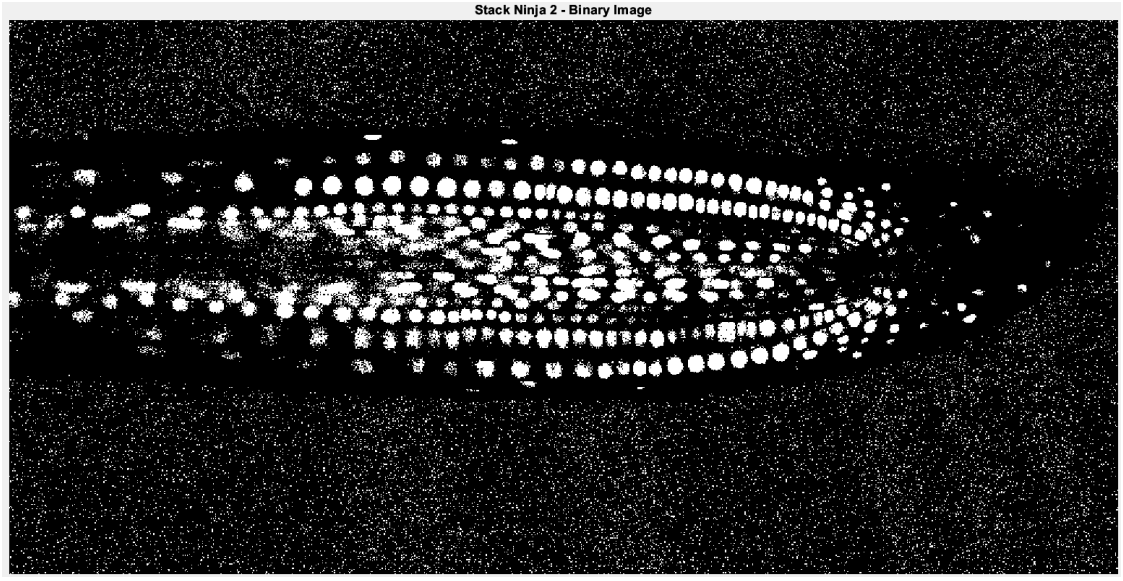

Figure 1.2.5 Binary Image of Stack Ninja 2

After applying median filter on image 2, binarize filtered image 2 with imbinarize function.

1.3 Image 3 – Stack Ninja 3

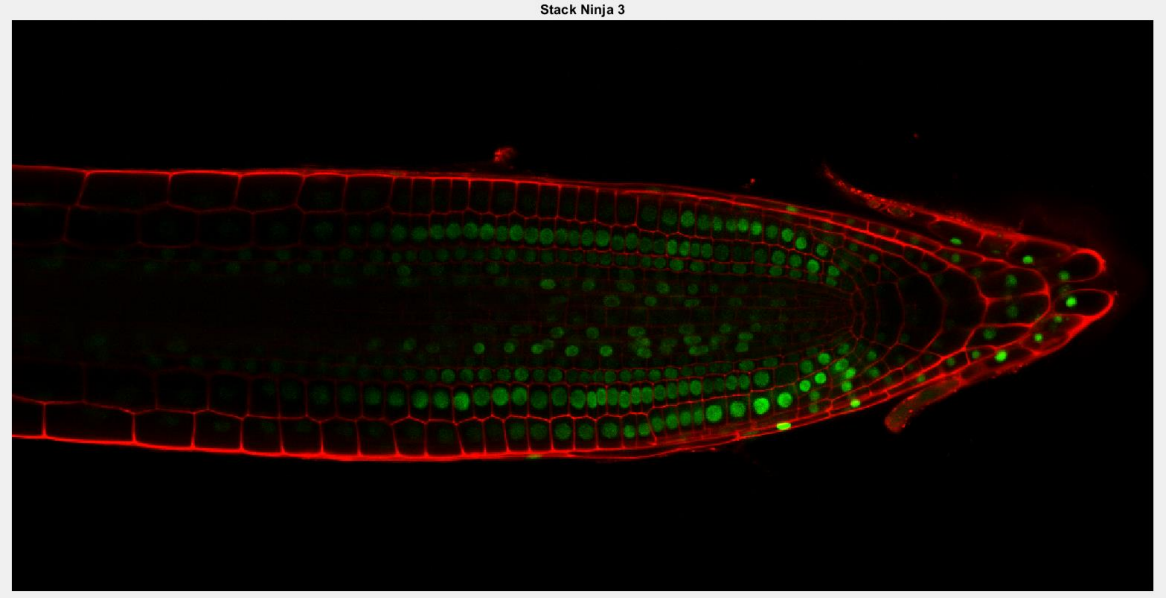

Figure 1.3.1 Original Stack Ninja 3

Read the third original image in the first step of pre-processing.

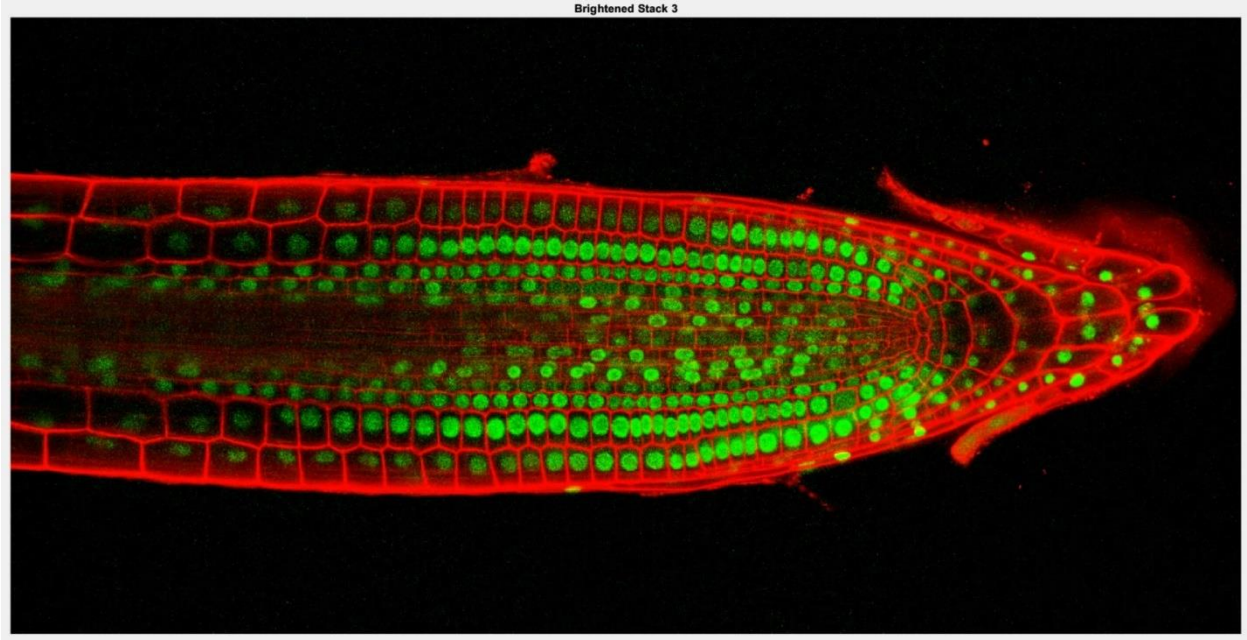

Figure 1.3.2 Brightened Stack Ninja 2

Same as image 1 and image 2, brightening image 3 is the second step of pre-processing with default value.

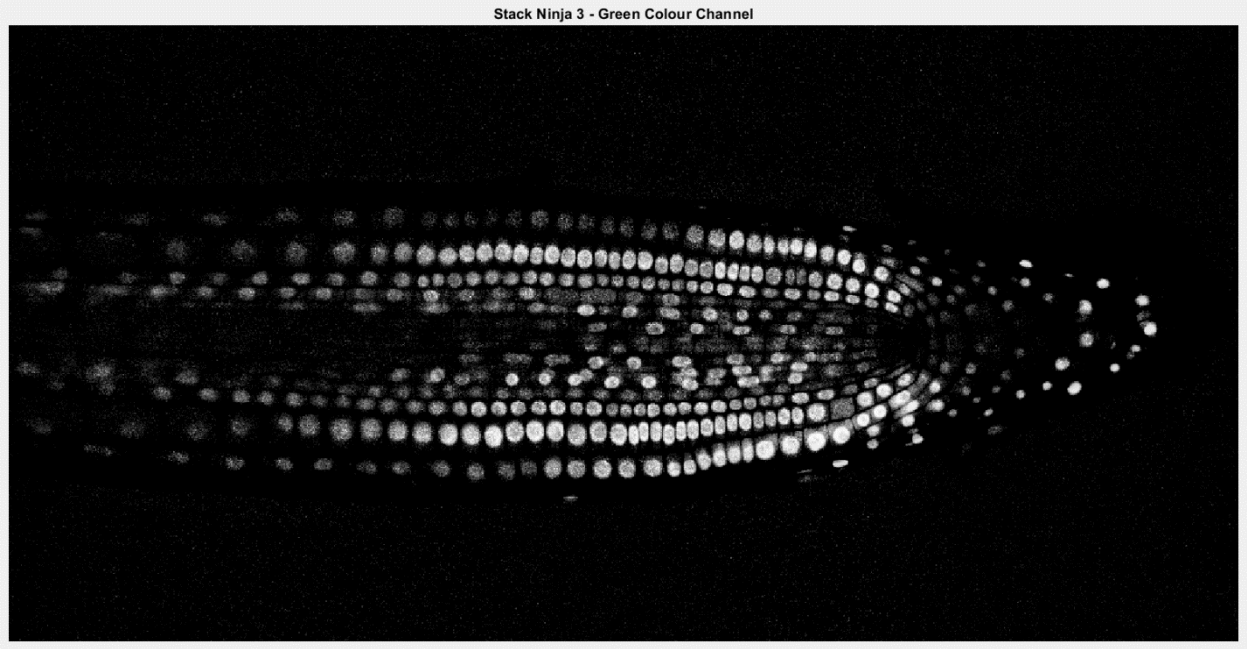

Figure 1.3.3 Green Channel of Stack Ninja 3

The next pre-processing step is extracting the green colour channel from image 3. Same as image 1 and image 2, the formula, Green – (Red + Blue) / 2 is applied for extracting green colour channel from image 3.

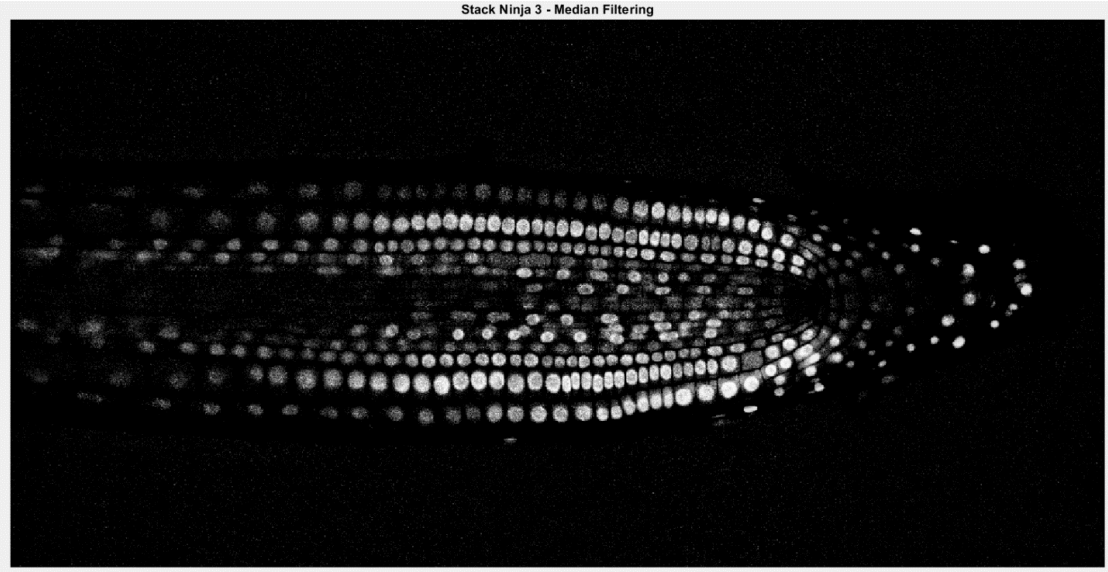

Figure 1.3.4 Median Filtering on Stack Ninja 3

After extracting the green channel from image 3, apply median filtering with the threshold same as image 1 and image 2, which is [1 1].

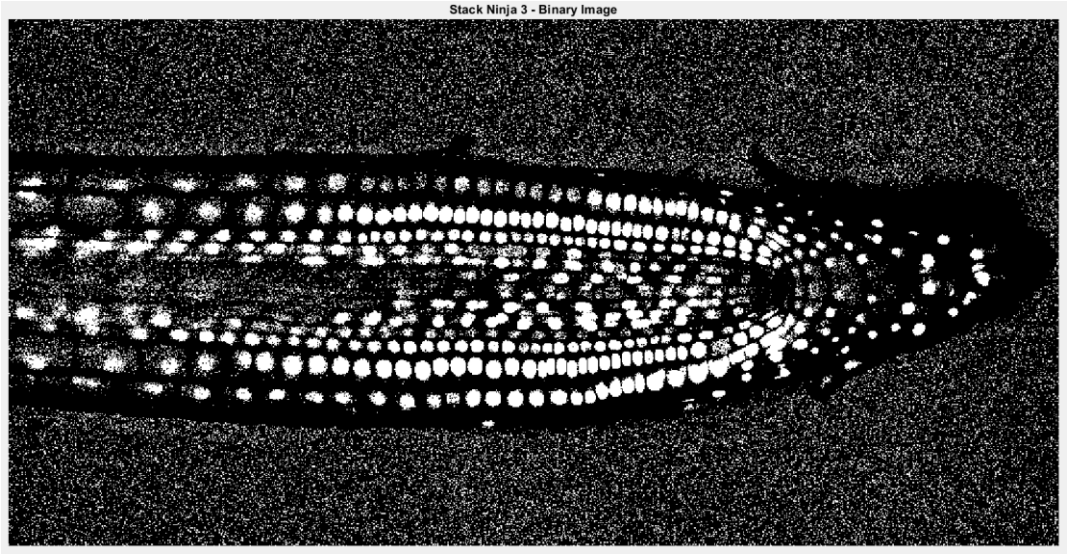

Figure 1.3.5 Binary Image of Stack Ninja 3

Next, use imbinarize function to binarize the filtered image 3.

2.0 Pipelining (After binarization)

2.1 Image 1 – Stack Ninja 1

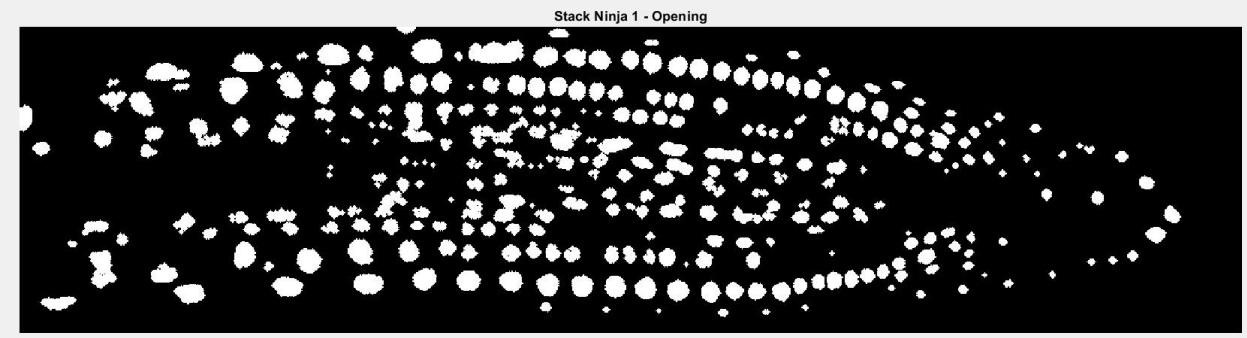

Figure 2.1.1 Opening Stack Ninja 1

After all the pre-processing is done, the first step of pipelining will be opening the binarized image using the imopen function. The value of structuring element or known as SE for opening the image is ‘disk’ and integer value of 2.

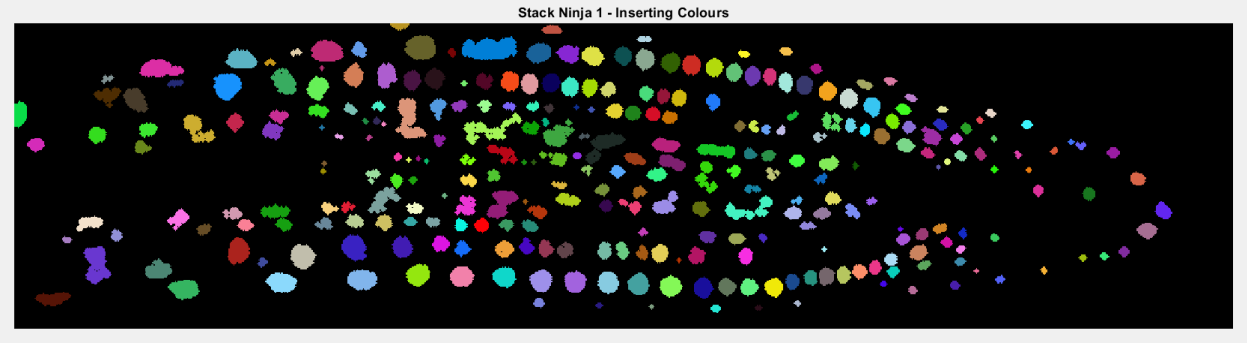

Figure 2.1.2 Colourised Binary Stack Ninja 1

The last step of pipelining will be inserting colours into the white nuclei. The insertion of randomised colours involves in few steps. Firstly, use bwconncomp function to count the numbers of connected components within the binary image. Secondly, use labelmatrix function to extract the matrix value from the connected components and add a unique label to each individual connected component. Thirdly, to display the number of components within the binary image using the NumObjects property. Fourthly, create a colour randomiser with RGB colours and lastly, insert random colours into the binary image with random colour using label2rgb function. A black background is assigned in the parameters of label2rgb function.

2.2 Image 2 – Stack Ninja 2

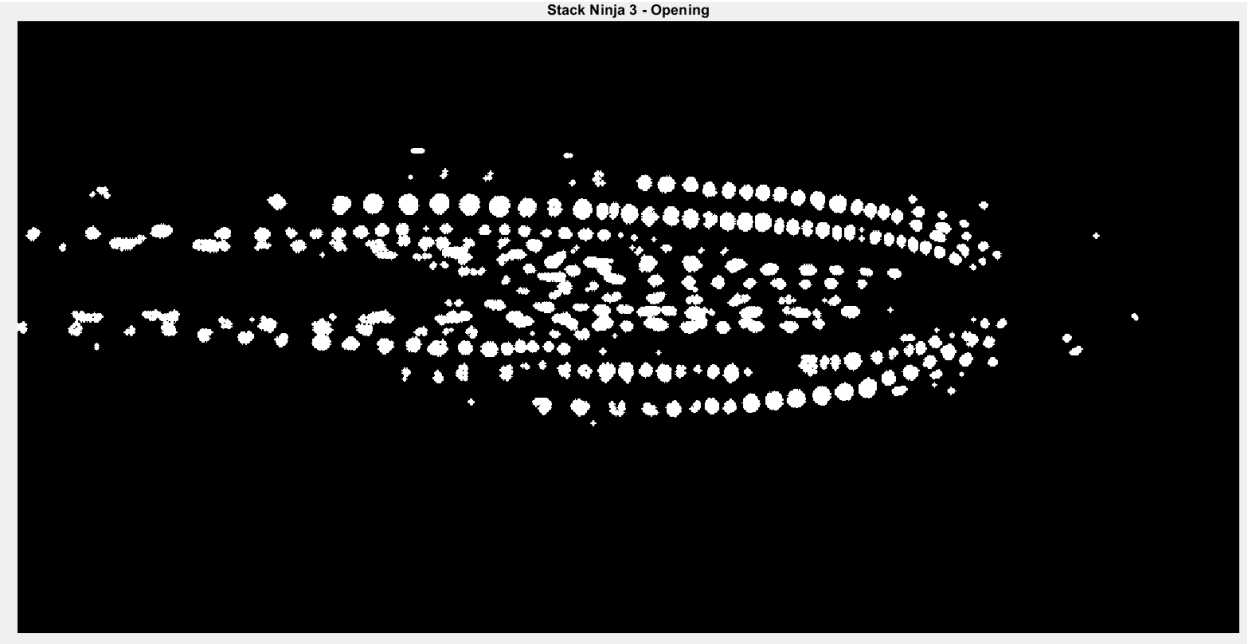

Figure 2.2.1 Opening Stack Ninja 2

As the pre-processing is completed, the first step is the same as image 1. The first step is opening the binary image 2 using imopen function. The structuring element will be the same as image 1, which are ‘disk’ and integer value of 2.

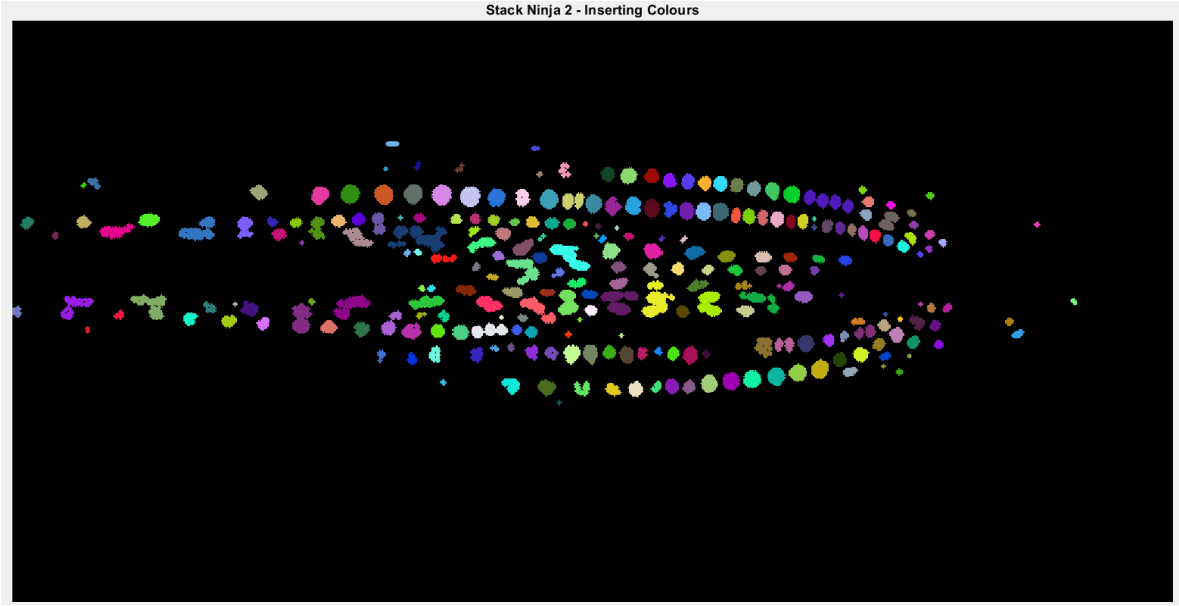

Figure 2.2.2 Colourised Stack Ninja 2

The method of inserting random colours is the same as image 1. In short, counting the number of components, extract the matrix and add unique label to each component. Then create a colour randomiser with RGB colours and insert the colours into the binary image. The colour of the background is black, same as image 1.

2.3 Image 3 – Stack Ninja 3

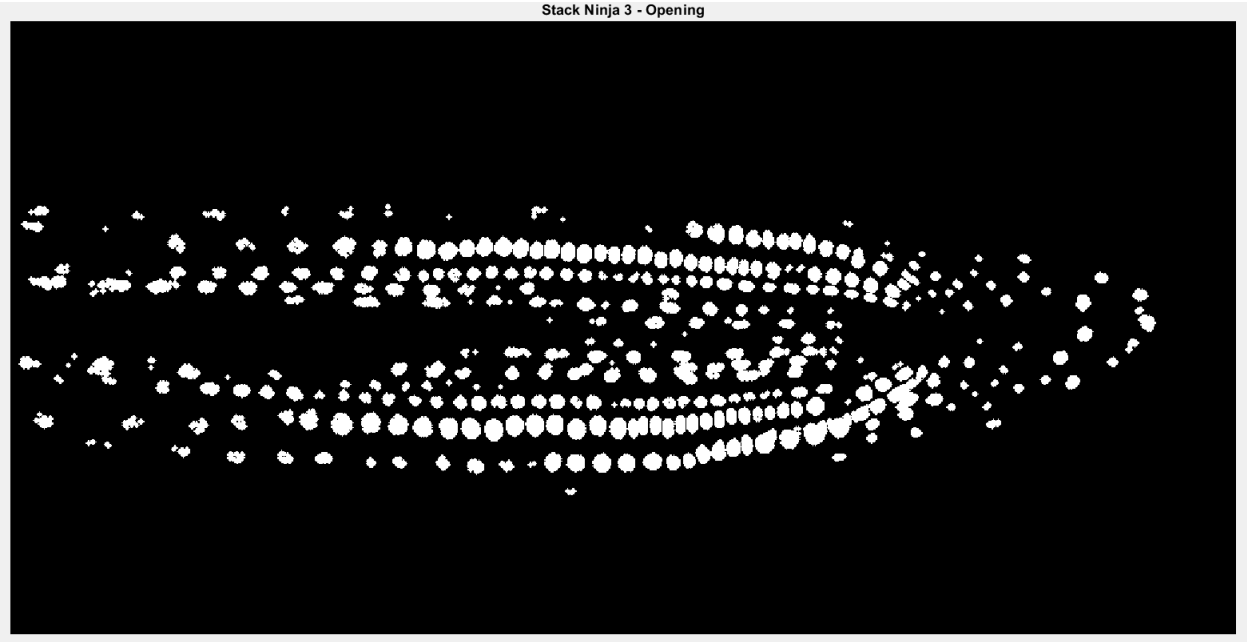

Figure 2.3.1 Opening Stack Ninja 3

The pre-processing is completed for image 3. For the first step of pipelining on image 3, opening the image using the function imopen with the same structural element as image 1 and image 2. The structural element will be ‘disk’ and integer value of 2.

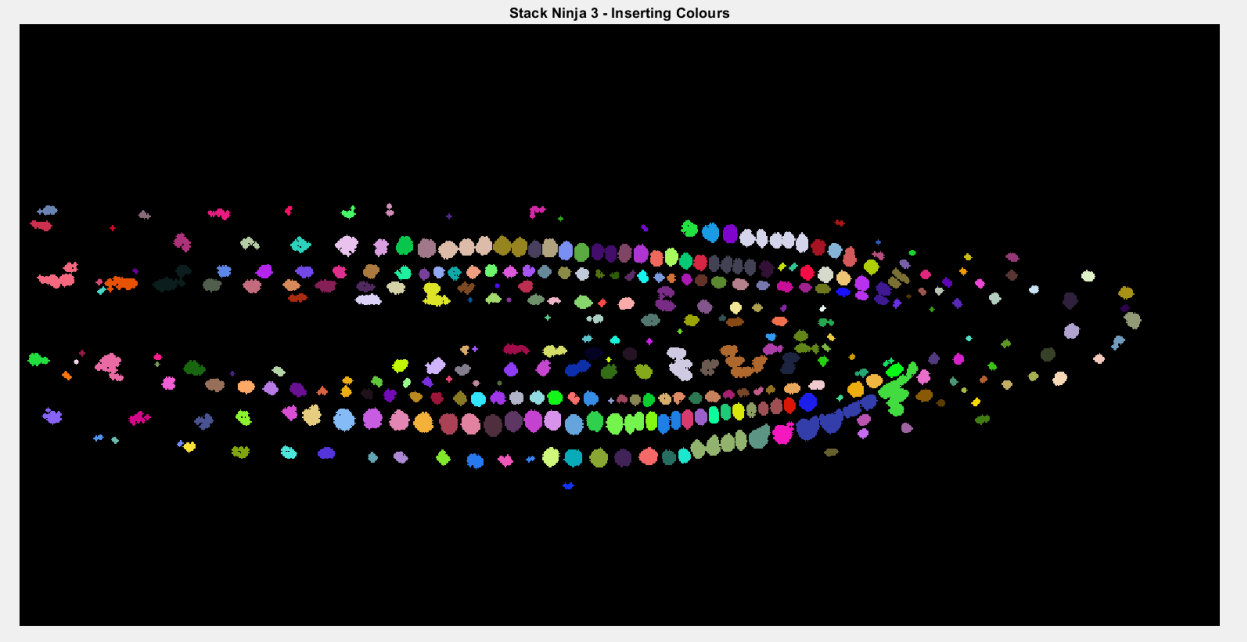

Figure 2.3.2 Colourised Stack Ninja 3

The insertion of colour is the same as image 1 and image 2. First, count the components using bwconncomp function. Next, extract the matrix of the components and label each component with unique label. Then, make a random colour generator with RGB colours and insert colour into the binary image 3. The background colour will be the same as image 1 and image 2, which is black.

3.0 Critical Analysis

The images are displayed as binary images to show more detailed comparison between two or more images.

3.1 Extracting Green Channel

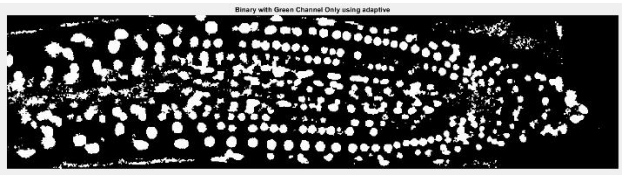

Figure 3.1.1 Image 1

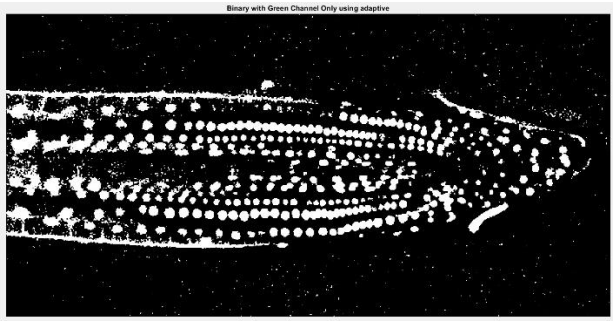

Figure 3.1.2 Image 3

One of the pre-processing steps is extracting the green colour channel from the images using RGB. The reason of using the formula Green – (Red + Blue) / 2 is to ensure all the green colour only components are extracted out from the image, as adaptive binarization is applied. If extracting green colour channel only, some non-green components are extracted as well. For example, Figure 3.1.1 and Figure 3.1.3 display red colour components after adaptive binarization.

Figure 3.1.3 Image 1

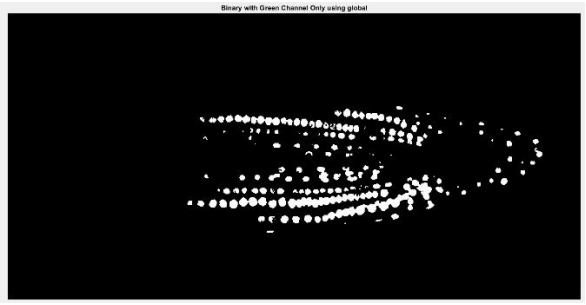

Figure 3.1.4 Image 3

Changing into global binarization will provide different results. The non-green components do not appear after global binarization, compared to adaptive. However, the major downside is the number of cells is reduced, especially comparing Figure 3.1.2 and Figure 3.1.4. A lot of cell nuclei do not appear after global binarization.

3.2 Difference between adaptive and global binarization

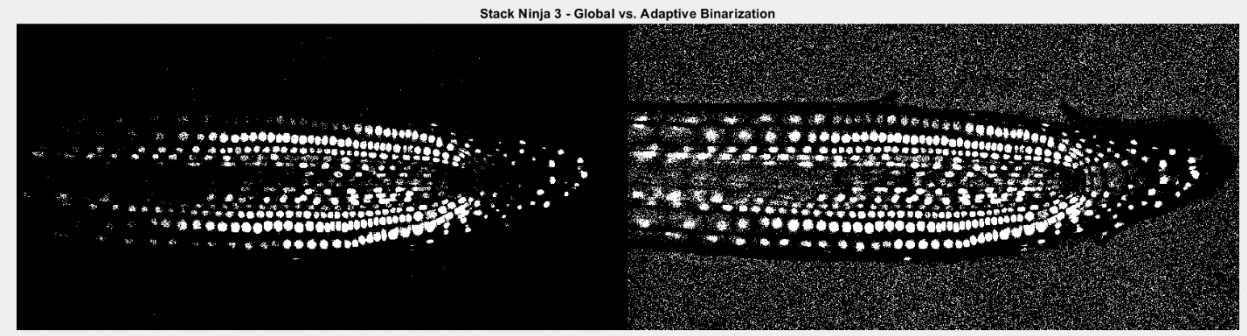

Figure 3.2.1 Difference between global and adaptive binarization on image 3

After extracting the green colour channel using Green – (Red + Blue) / 2, there are two methods of binarizing the image. One of them is global binarization and another method is adaptive binarization. The reason of applying adaptive binarization is because it shows more detail and higher number of cells displayed. Image 3 has the biggest difference when binarize methods are applied. Figure 3.2.1 is the comparison between global and adaptive binarization. Left binary image is binarized using global binarization, while the right binary image is binarized using adaptive binarization. The global binarized image on the left shows less noise compared to the right image, but it suffers from displaying a smaller number of cell nuclei. Despite the right image is noisier, it is not a big deal as it can be removed by applying median filtering and opening afterwards.

The process of adaptive binarization, or known as local binarization, involves in dividing the images into tiny areas. Each pixel will be calculated after the division of the image during when adaptive binarization is started (Khakimov, 2021). The threshold value is determined by the intensity of the neighbouring pixels. The calculation is based on the pixel intensity of the image and the threshold value. As this method is calculating each pixel across the image with a threshold value, this could be the reason why the right image in Figure 3.2.1 is noisier than the left image. The noisy pixels are calculated in the method as well.

The other method is global binarization or known as thresholding binarization. This method unlike adaptive binarization, calculating each pixel with a threshold value. It calculates the entire image with a pre-defined threshold value (Khakimov, 2021). If the intensity of pixels is lower than the threshold, it will turn into black when transform into binary image. This could suggest

that some of the cells in the darker side is not shown in the binary image due to low pixel intensity.

3.3 Filtering

Figure 3.3.1 Image 1 - Comparison between Median Filtering and Gaussian Filtering

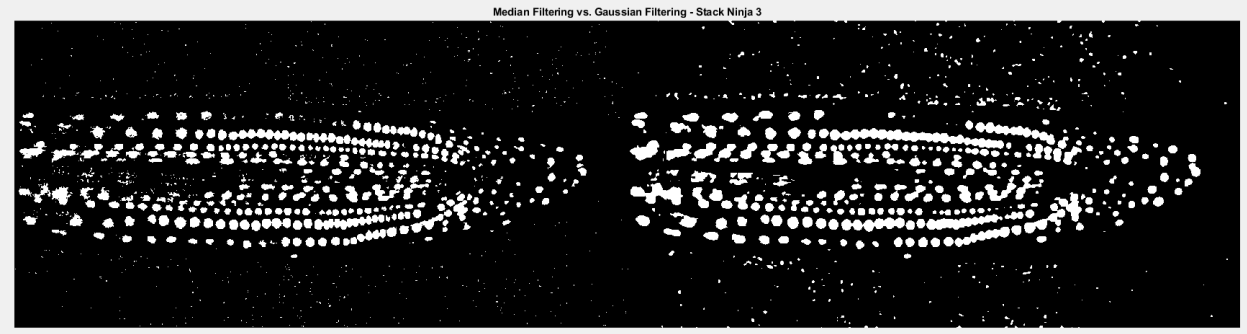

Figure 3.3.2 Image 3 - Comparison between Median Filtering and Gaussian Filtering

There are two image filtering technique applied to reduce the noise of the three images, which are median filtering and Gaussian filtering. The purpose of applying both technique is to reduce noise. However, these filtering techniques showed two different results.

Gaussian filtering is a linear filter, and it blurs out the image. The pixel replaces their neighbouring pixels which are the noisy pixels using the Gaussian formula (Kumar & Nachamai, 2017). This is the reason why the right Figure 3.3.1 and Figure 3.3.2, there are number of cell nuclei are combined with their neighbouring cell nuclei. The image became blurry before the images are binarized. Besides that, the tiny bits outside of the cell are not removed and they become larger. Therefore, this is the main reason of not using Gaussian filtering for noise reduction.

Median filtering is a non-linear filter that using the median value of each neighbouring pixel. The neighbouring area of a pixel is usually in square, such as 3x3 or 5x5 (Felicity, 2022). Median

filtering scans though pixel by pixel to determine the intensity value of the pixels. After that, the pixels will be replaced with the median value of the neighbouring pixels (Kumar & Nachamai, 2017). One of the advantages of median filtering is preserving the edges while removing the noises from an image. Therefore, this is the main reason of applying median filtering for noise reduction.

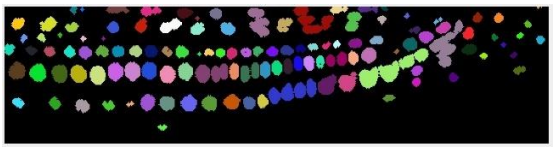

Figure 3.3.3

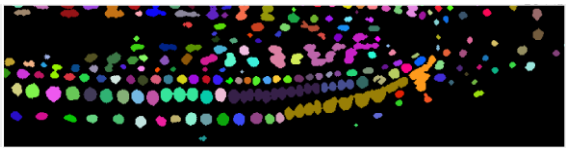

Figure 3.3.4

However, the threshold of median filtering applied is low. Initially, the threshold of median filtering was set at [3 3], but the problem is some of the cell nuclei are merged. Take image 3 as an example. Figure 3.3.3 is the median filtering threshold of [1 1] and Figure 3.3.4 is the threshold of [3 3]. Both images had applied opening function. In Figure 3.3.3, most of the cells are not sticked together as the threshold is lower. In Figure 3.3.4, as the threshold is higher, cells are combined. At the end, threshold [1 1] was chosen for median filtering as the individual cells are clearly shown after binarization and opening.

3.4 Opening

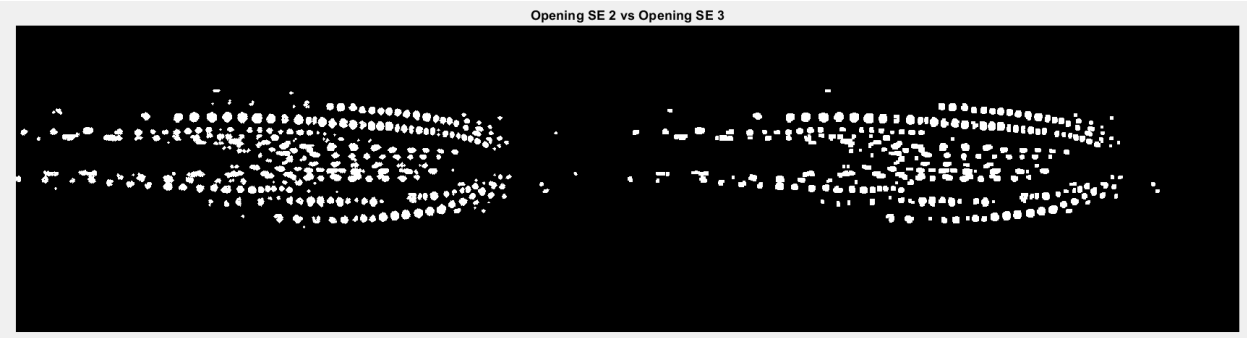

Figure 3.4.1 Opening with SE value 2 and SE value 3

Opening is an image processing technique for smaller objects in an image, and it also used for internal noise reduction, while reserving the edges (*Difference Between Opening And Closing in Digital Image Processing*, n.d.).

The function imopen is used for opening the images with a structural element defined. The structural element (SE) contains two parameters which are ‘disk’ and integer value of 2. The reason of using ‘disk’ is because the cell nuclei are in circular or disk shape, so ‘disk’ is suitable for opening the image. In addition, the radius is set to 2. The reason of not using higher radius is to preserve the cell nuclei as many as possible. The Figure 3.4.1 shows the comparison of using structural value of 2 and 3. Comparing the two images, the difference is minimal. Some of the cell nuclei are removed when the threshold increased. The advantage of higher threshold value is it will produce a smoother edge of the cell nuclei. As the main goal is preserving the cell nuclei as many as possible, structural element with ‘disk’ and 2 will be the final decision.

3.5 Removed functions

3.5.1 Gaussian Filtering

As mentioned in 3.3, the reason of removing Gaussian Filtering function imgaussfilt is the noise are grown bigger as shown in the left image of Figure 3.3.2. Therefore, Gaussian filtering is replaced by median filtering for noise reduction.

3.5.2 Laplacian Filtering

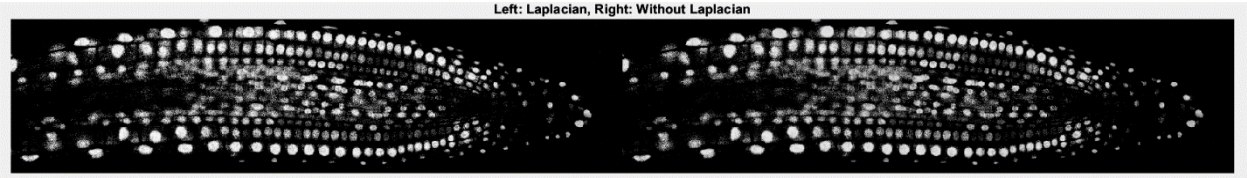

Figure 3.5.2.1 Left: with Laplacian filtering, Right: without Laplacian filtering

The initial purpose of applying Laplacian filtering is to smooth the edge of the cell nuclei so it will show a smoother edge after binarization.

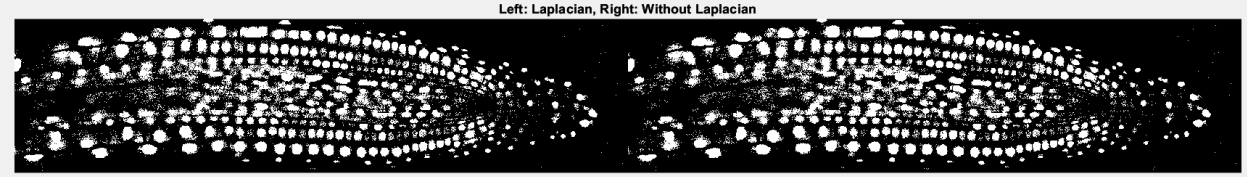

Figure 3.5.2.2 Left: with Laplacian filtering, Right: without Laplacian filtering

The fast local Laplacian filtering function, locallapfilt was applied previously in the code. It was removed because of the differences between before and after applying the function. Initially, the sigma value and alpha value was set as 0.3 and 0.5 respectively. However, the differences are not noticeable with naked eyes. Even though the sigma value was increased to 0.5 and alpha value was increased to 0.7. The differences are still not noticeable as it shows in Figure 3.5.2.1 and Figure 3.5.2.2.

4.0 References

Difference Between Opening And Closing in Digital Image Processing. (n.d.). Viva Differences. Retrieved April 7, 2023, from https://vivadifferences.com/difference-between-opening-and-closing-in-digital-image-processing/

Felicity. (2022). *The Median Filter: An Effective Solution For Removing Noise From Images*. Picozu. https://www.picozu.com/the-median-filter-an-effective-solution-for-removing-noise-from-images

Khakimov, E. (2021). *Image binarization methods*. LinkedIn. https://www.linkedin.com/pulse/image-binarization-methods-erik-khakimov

Kumar, N., & Nachamai, M. (2017). Noise Removal and Filtering Techniques used in Medical Images. *Oriental Journal of Computer Science and Technology*, *10*(1), 103–113. https://doi.org/10.13005/ojcst/10.01.14