1. Pre-processing techniques

1.1 Reading and displaying the original input nuclei images

First, read and display the image of StackNinja1.bmp, StackNinja2.bmp and StackNinja3.bmp, that corresponds to the respective input images of nuclei 1, 2 and 3 using MATLAB's *imread* function.

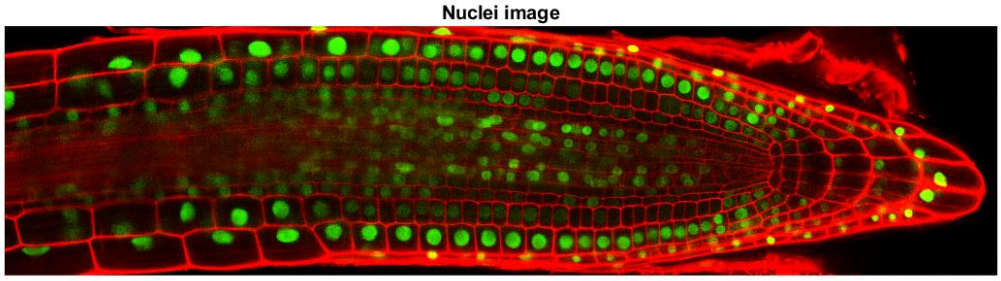

Figure 1.1 a: Original input image of nuclei 1

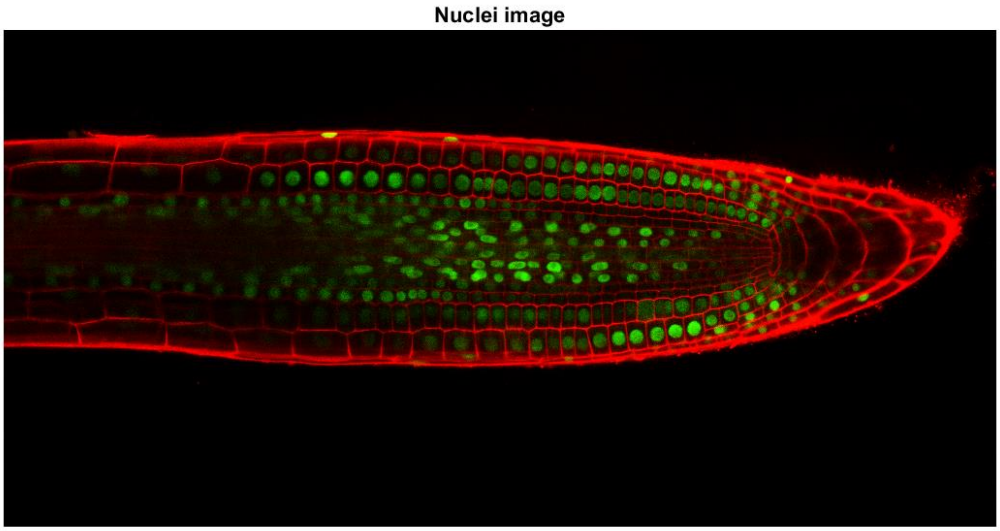

Figure 1.1 b: Original input image of nuclei 2

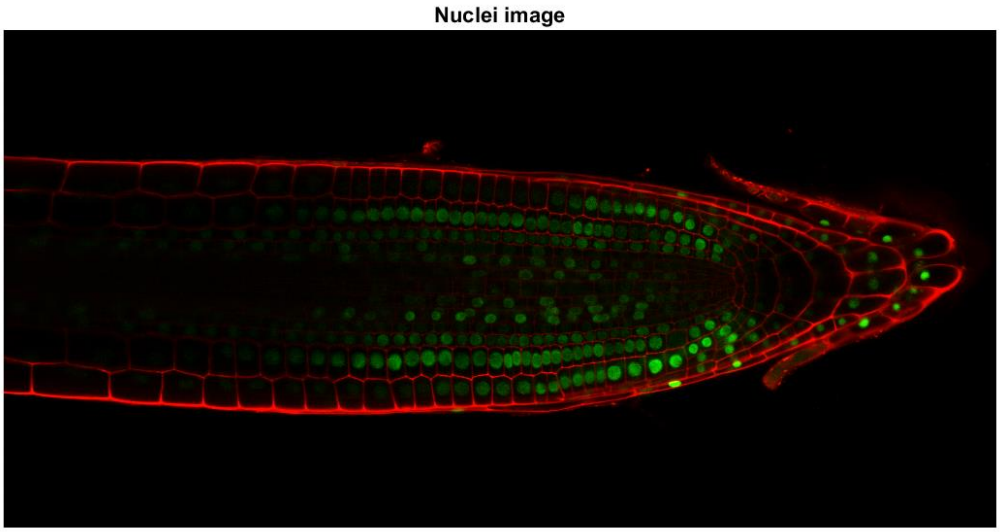

Figure 1.1 c: Original input image of nuclei 3

1.2 Contrast adjustment

The first pre-processing technique applied is MATLAB's contrast adjustment function, *imlocalbrighten*. This function enhances the brightness of an image by increasing the contrast of an image's darker regions while preserving the brighter regions. The *AlphaBlend* option is set to true to preserve content from the original image in the lightened image.

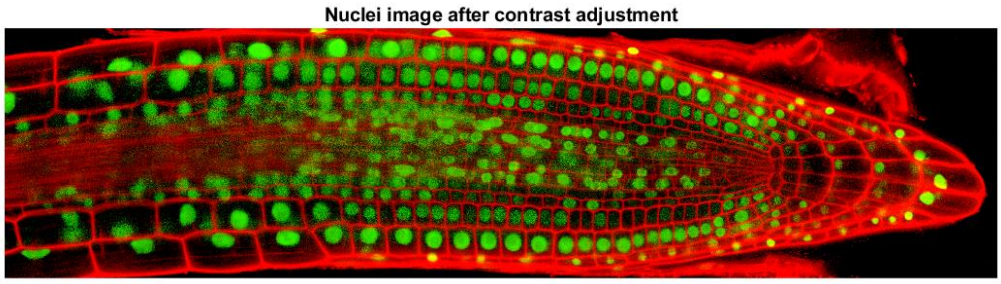

Figure 1.2 a: Nuclei 1 after contrast adjustment

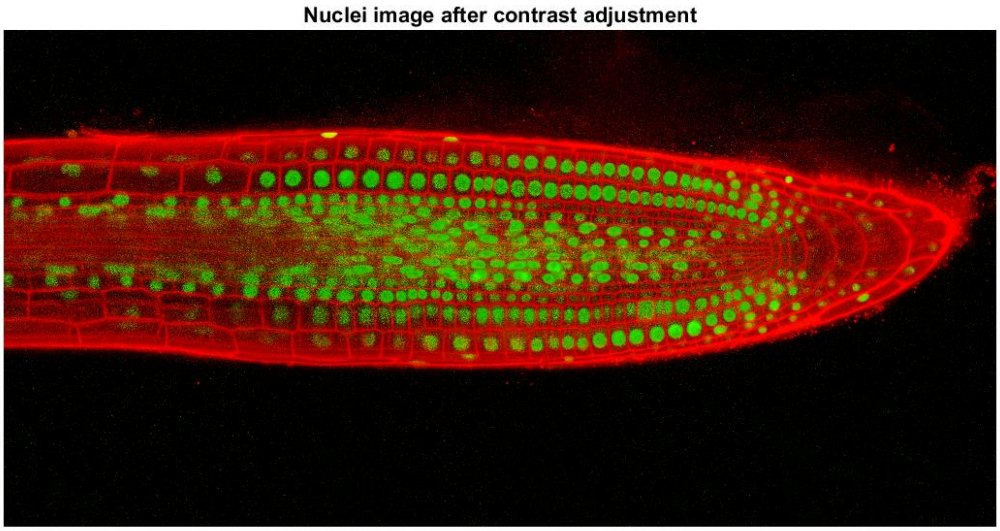

Figure 1.2 b: Nuclei 2 after contrast adjustment

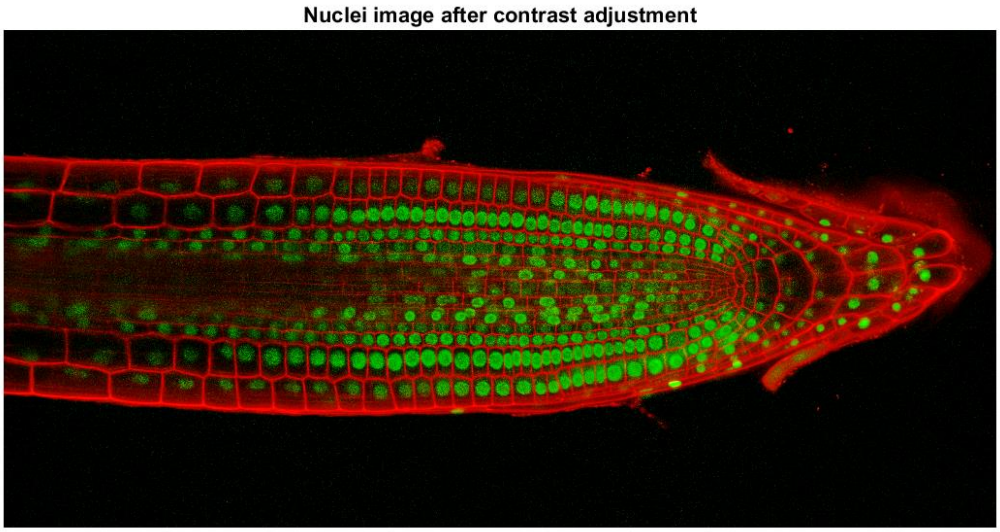

Figure 1.2 c: Nuclei 3 after contrast adjustment

1.3 Colour space conversion

The colour space is converted from RGB to CIE L*a*b*. This is done to extract the green nuclei cells from the contrast adjusted image. The extraction result is overlaid back onto the contrast-adjusted nuclei image to verify the accuracy of green cell extraction.

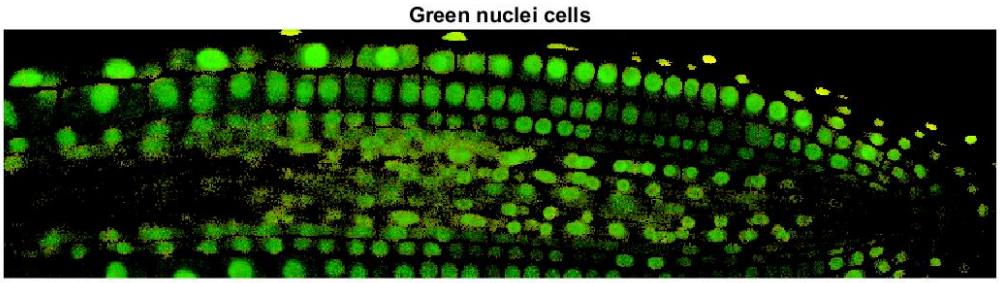

Figure 1.3 a: Nuclei 1 green cells

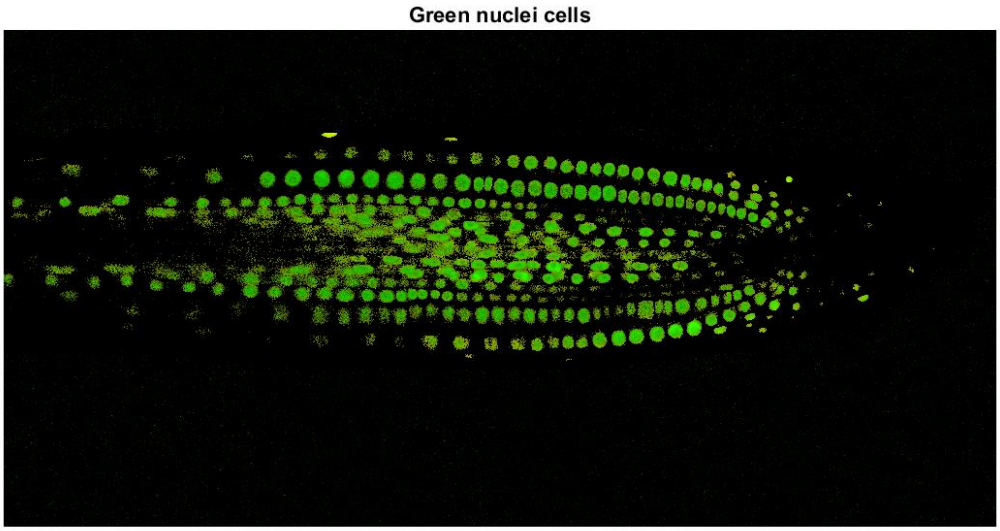

Figure 1.3 b: Nuclei 2 green cells

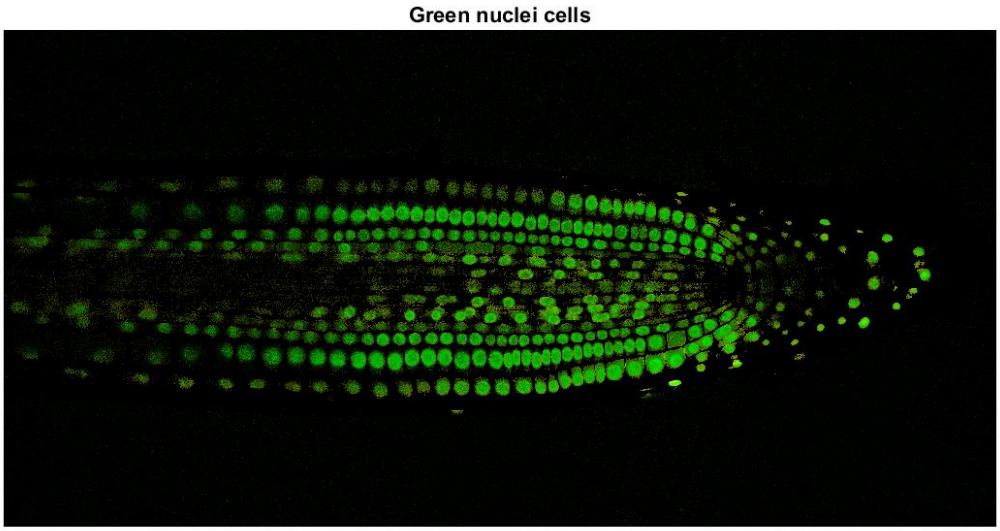

Figure 1.3 c: Nuclei 3 green cells

2. Pipeline Analysis

2.1 Grayscale conversion using greenness

Greenness is computed using the formula $Greenness = |\ G\ -\ \frac{(R+B)}{2}\ |$. The green nuclei cells are then converted into a grayscale image using the greenness formula.

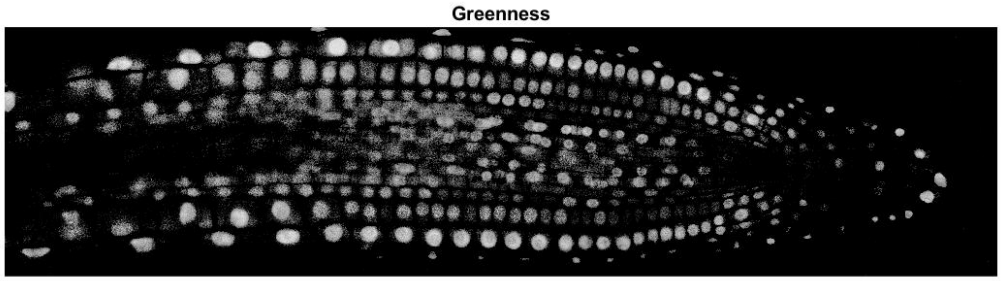

Figure 2.1 a: Nuclei 1 greenness

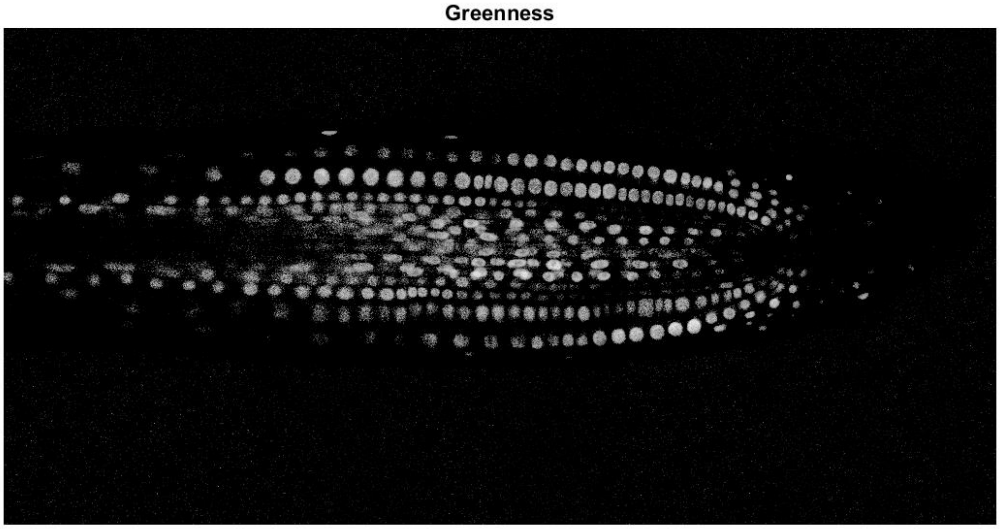

Figure 2.1 b: Nuclei 2 greenness

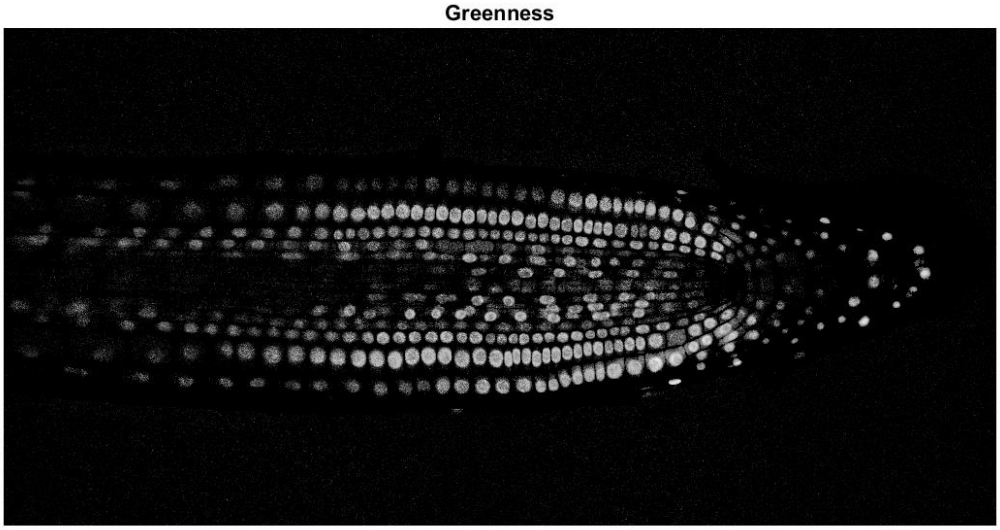

Figure 2.1 c: Nuclei 3 greenness

2.2 Bilateral Filtering

The greenness image is filtered using MATLAB's bilateral filter function *imbilatfilt*. Bilateral filter is a non-linear filtering technique that might blur the green nuclei cells while preserving the edges of the green cells.

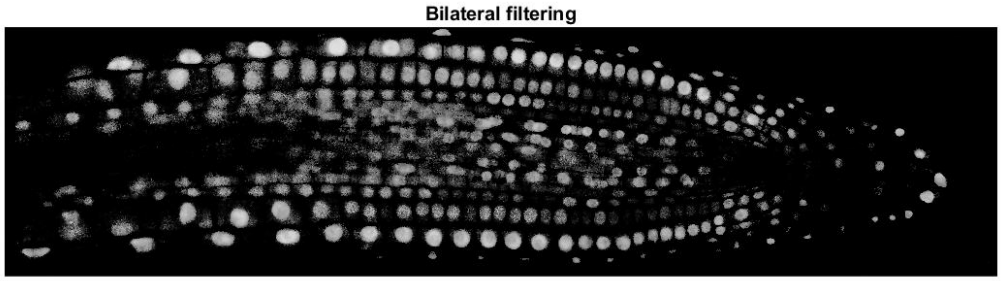

Figure 2.2 a: Nuclei 1 bilateral filtering

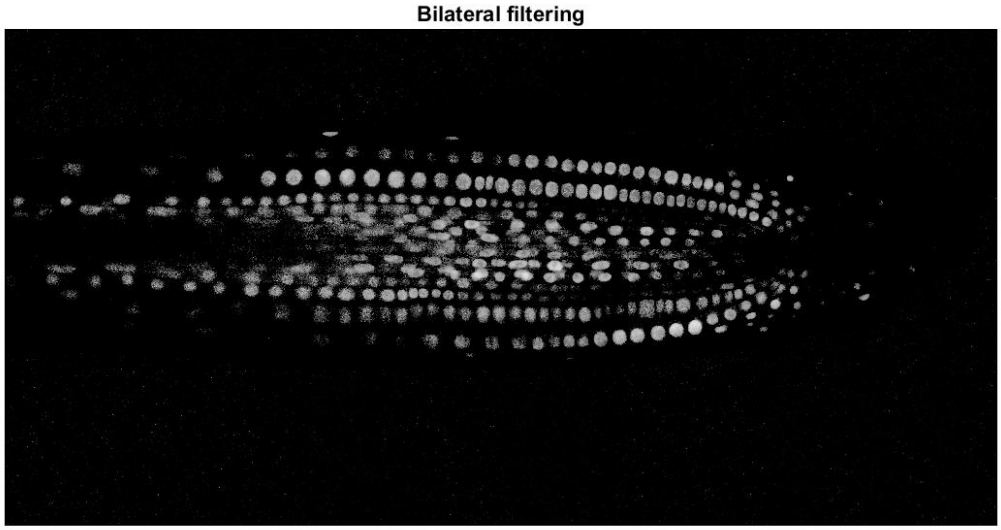

Figure 2.2 b: Nuclei 2 bilateral filtering

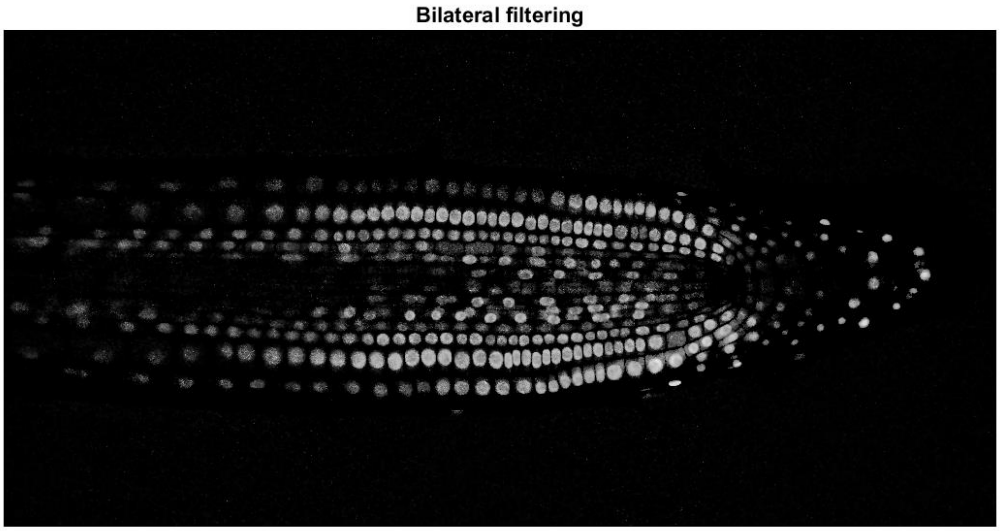

Figure 2.2 c: Nuclei 3 bilateral filtering

2.3 Thresholding and Binarization

The MATLAB adaptive thresholding function *adaptthresh* computes a locally adaptive threshold for the 2-D bilateral filtered image that varies based on local mean intensity in the neighbourhood of each individual pixel. A value of 0.03 was added as it has given the best result overall across all three nuclei images. The bilateral filtered image is then converted into a binary image using MATLAB's *imbinarize* function defined by the previous threshold value.

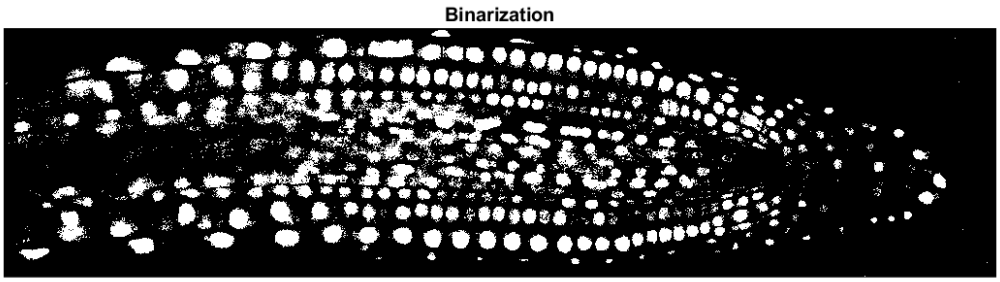

Figure 2.3 a: Nuclei 1 binarization

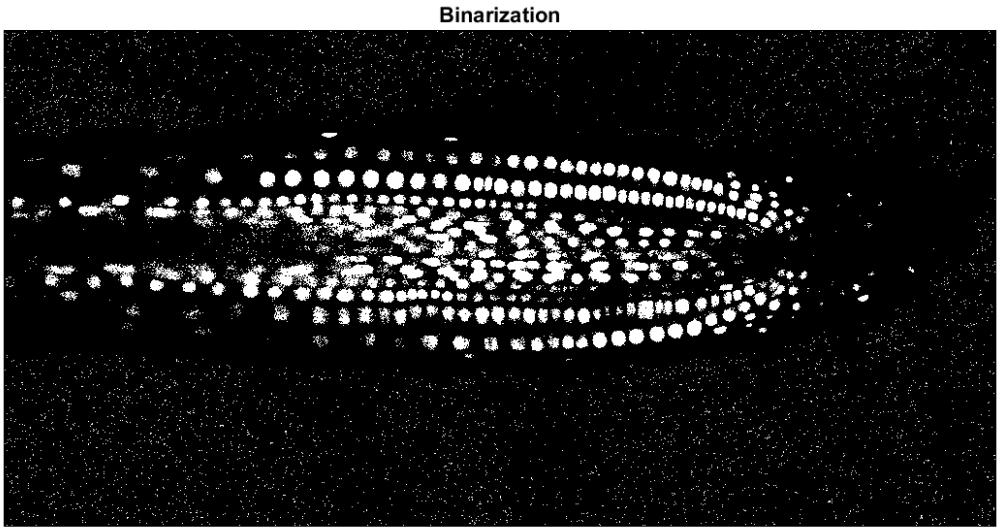

Figure 2.3 b: Nuclei 2 binarization

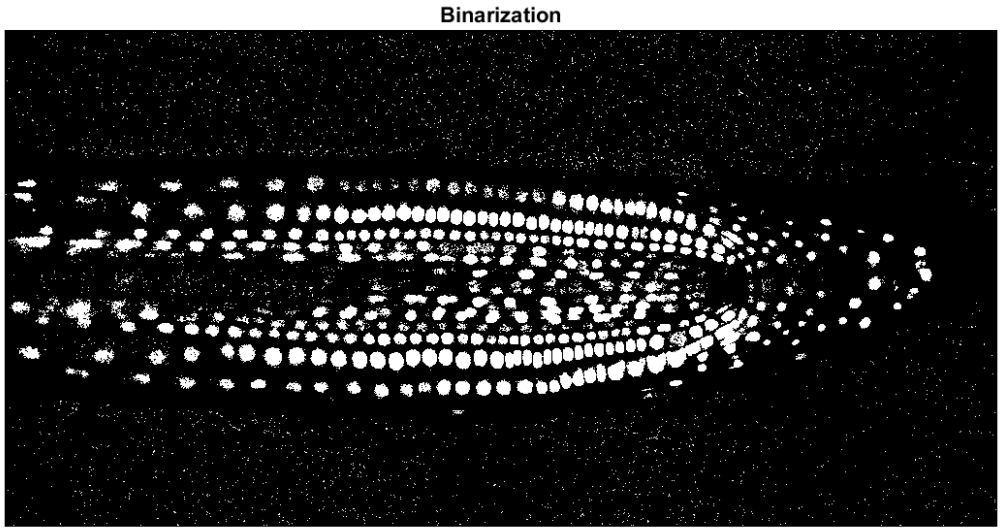

Figure 2.3 c: Nuclei 3 binarization

2.4 Connected Component Filtering

The MATLAB function *bwareafilt* is incorporated to filter out connected components based on the area of its connected components. The function returns a logical matrix with objects that has an area larger or equal to the specified value of 4. The filtered binary image retains connected components with area of 4 and above. This is necessary to further reduce the ambient noise that were previously retained from the binary image.

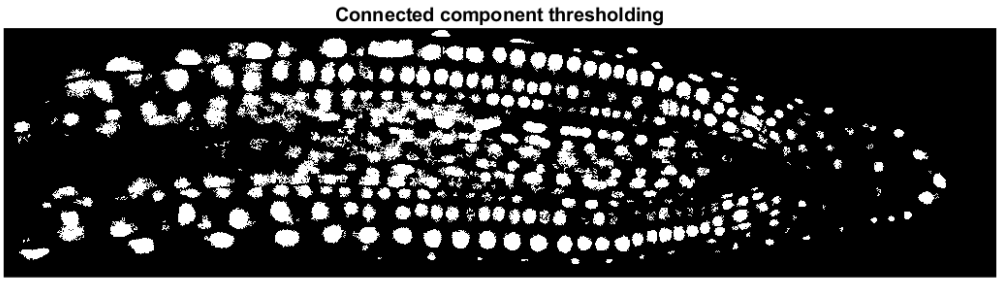

Figure 2.4 a: Nuclei 1 thresholding for connected components

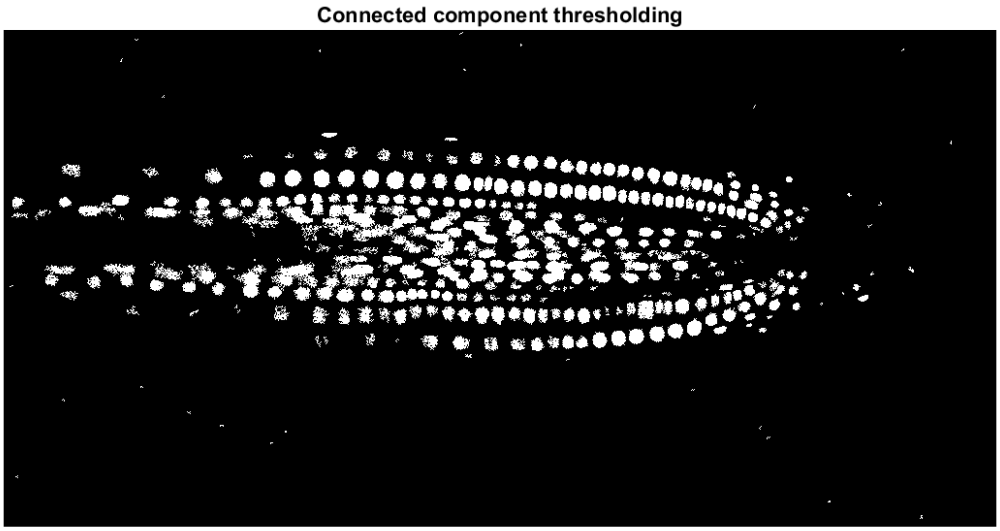

Figure 2.4 b: Nuclei 2 thresholding for connected components

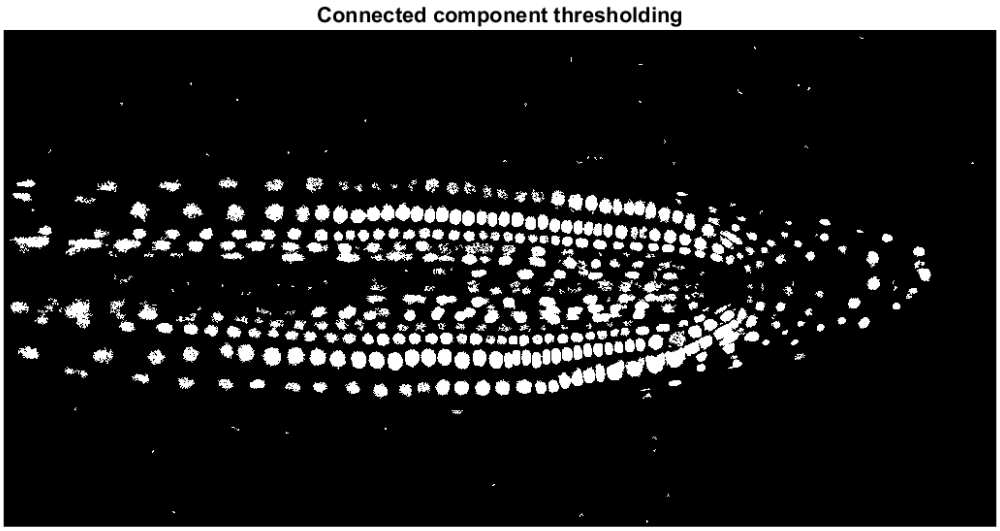

Figure 2.4 c: Nuclei 3 thresholding for connected components

2.5 Morphological operation on larger connected components

2.5.1 Defining larger connected components

The MATLAB function *bwareafilt* extracts the 10 largest connected components from the threshold binary image, producing a separate binary image containing only the 10 largest connected components defined by the pixel connectivity argument of 8. Pixels are considered connected if their edges or corners touch. Resulting in 2 adjoining pixels are part of the same neighbourhood if they are both on and are connected along the horizontal, vertical, or diagonal direction.

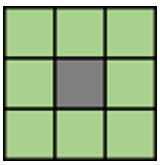

Figure 2.5.1 a: 8-neighbourhood pixel connectivity

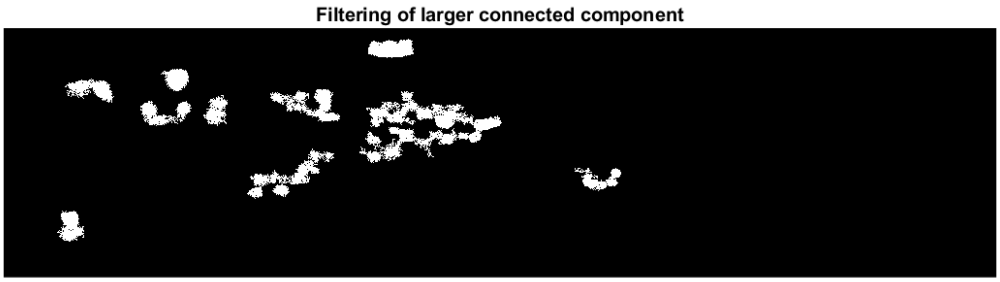

Figure 2.5.1 b: Nuclei 1 larger connected components

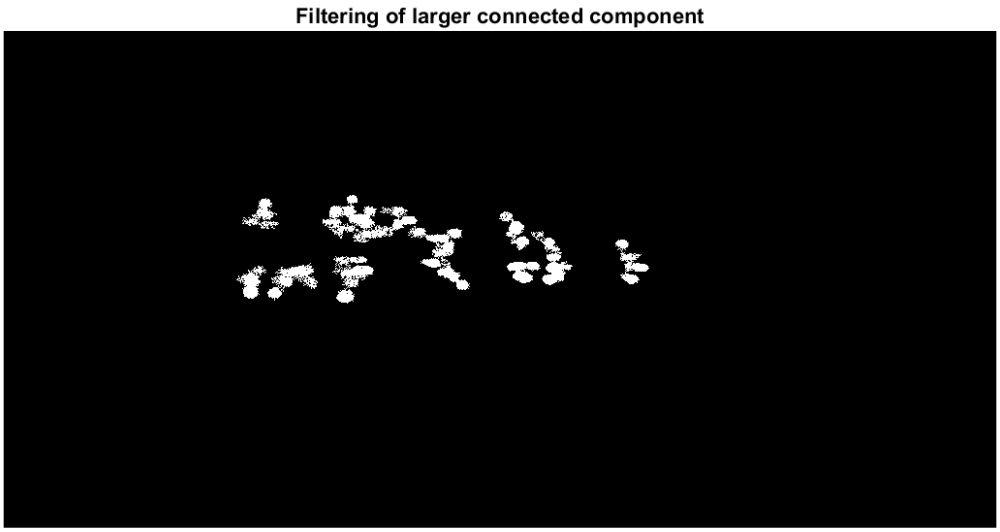

Figure 2.5.1 c: Nuclei 2 larger connected components

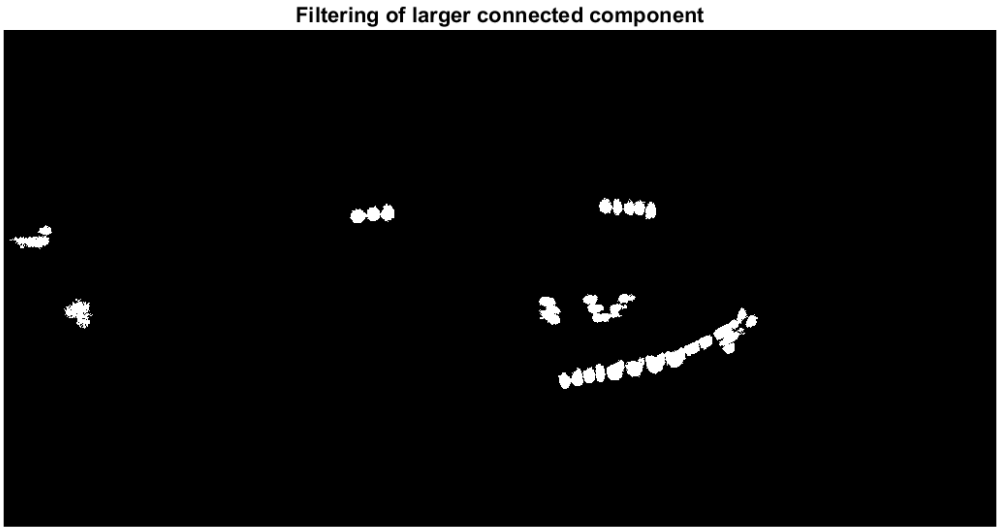

Figure 2.5.1 d: Nuclei 3 larger connected components

2.5.2 Structuring element

The MATLAB's *strel* function is defined with the *disk* shape with a radius size of 2 measured from the origin to create a flat 2-D morphological structuring element. This matrix is a special form of 5-by-5 processing matrix that shows the pixel of an image being processed and defines the neighbourhood used in the processing of each pixel.

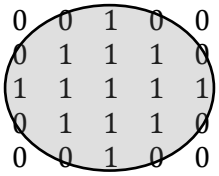

Figure 2.5.2: Disk shape overlaying on the structuring element with radius of 2

2.5.3 Morphological opening

MATLAB's *imopen* function is paired with the structuring element defined above to smooth the contour of connected components, break narrow isthmuses to reduce thin noise protrusions. The opening morphological operation performs morphological erosion followed by dilation on the eroded image using the same structuring element defined in the previous section. The overall shape of the nuclei cells is preserved while some connected components are detached. This results in removal of smaller connected components and noise while preserving the overall shape of the nuclei cells.

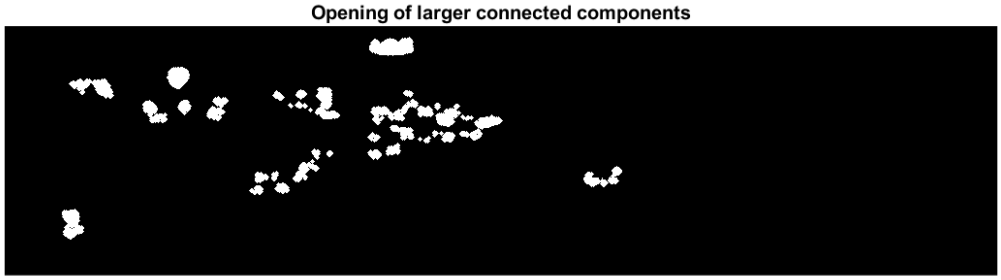

Figure 2.5.3 a: Nuclei 1 opening of larger connected components

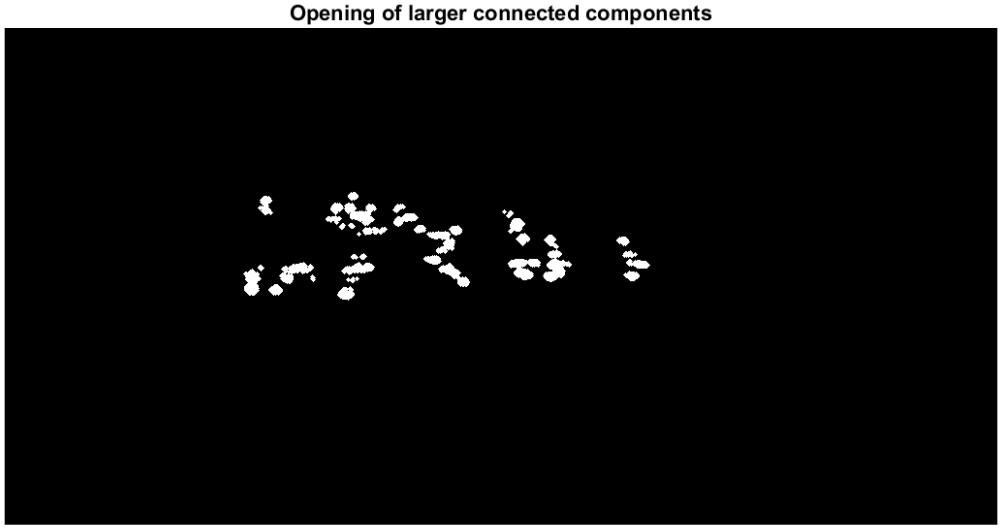

Figure 2.5.3 b: Nuclei 2 opening of larger connected components

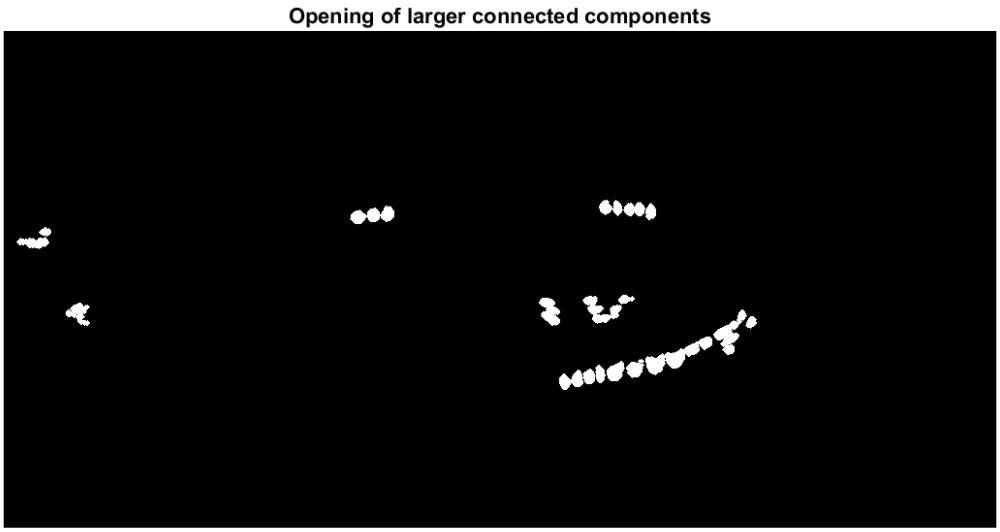

Figure 2.5.3 c: Nuclei 3 opening of larger connected components

2.5.4 Watershed segmentation

A series of functions was used to perform watershed segmentation taken from the previous image. The steps taken in chronological order calculates the distance transformation of the complemented binary image. Followed by segmenting the previous image using watershed ridge lines. MATLAB's *imextendedmin* function with h-value of 2 is used for minima suppression whose depth is less than *h* to filter out tiny local minima. The function modifies the distance transformation and minimises over-segmentation. The MATLAB's *imimposemin* function is applied to modify the initial distance transformation to ensure that it only has minima at desired regions. Finally, fill any holes in between the nuclei cells using MATLAB's *imfill* function.

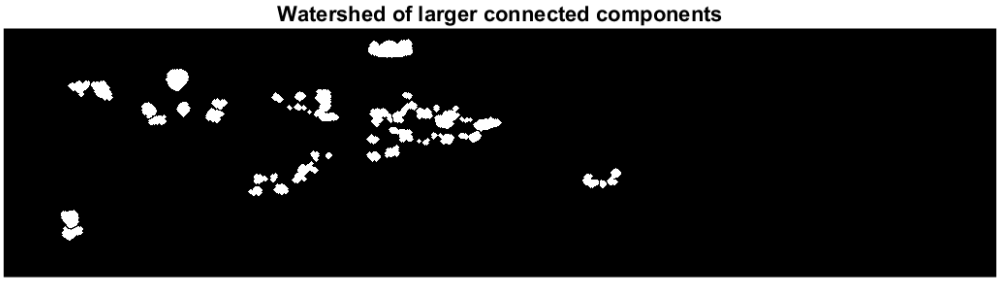

Figure 2.5.4 a: Nuclei 1 watershed segmentation of larger connected components

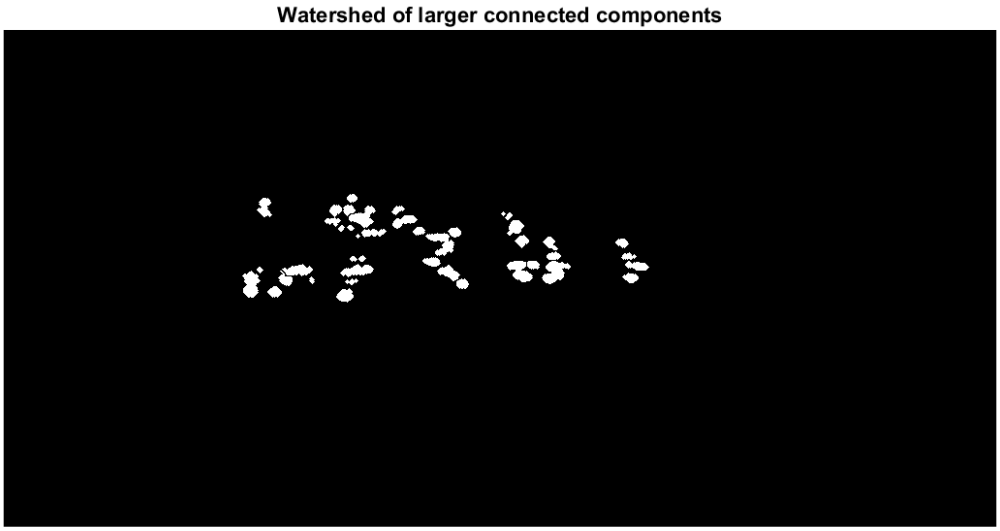

Figure 2.5.4 b: Nuclei 2 watershed segmentation of larger connected components

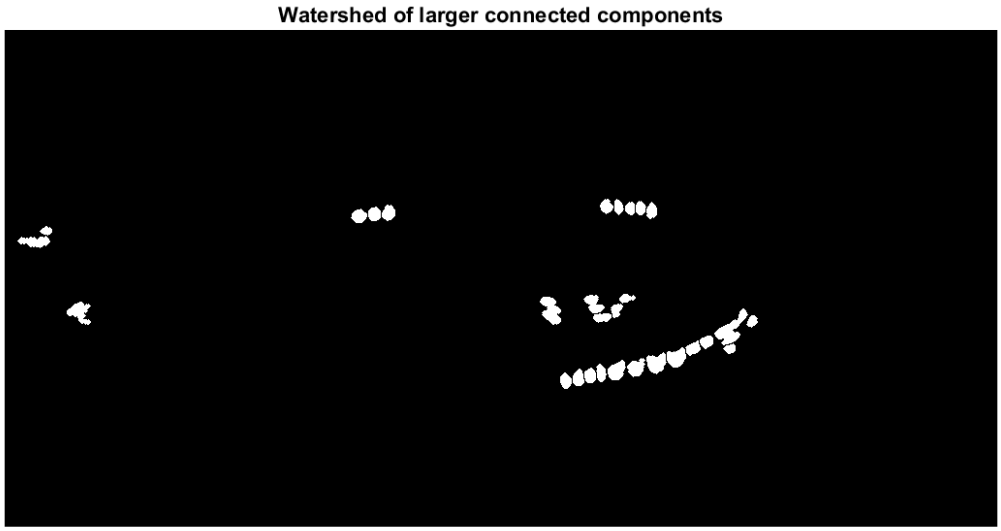

Figure 2.5.4 c: Nuclei 3 watershed segmentation of larger connected components

2.6 Morphological operation on smaller connected components

2.6.1 Defining smaller connected components

The larger connected components are subtracted from the pre-morphological binary image MATLAB function. Pixels are considered connected if their edges or corners touch. Resulting in 2 adjoining pixels are part of the same neighbourhood if they are both on and are connected along the horizontal, vertical, or diagonal direction.

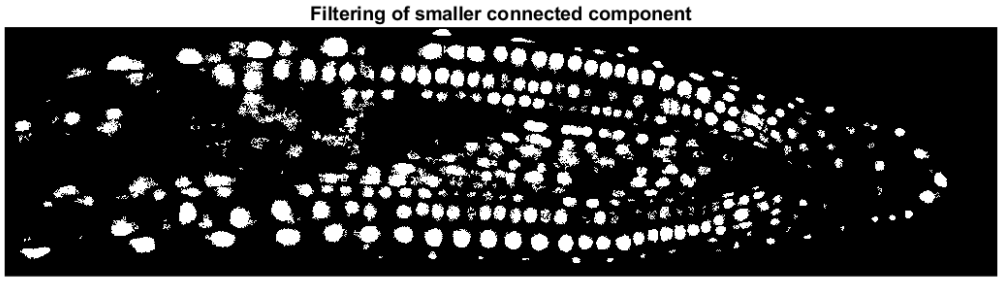

Figure 2.6.1 a: Nuclei 1 smaller connected components

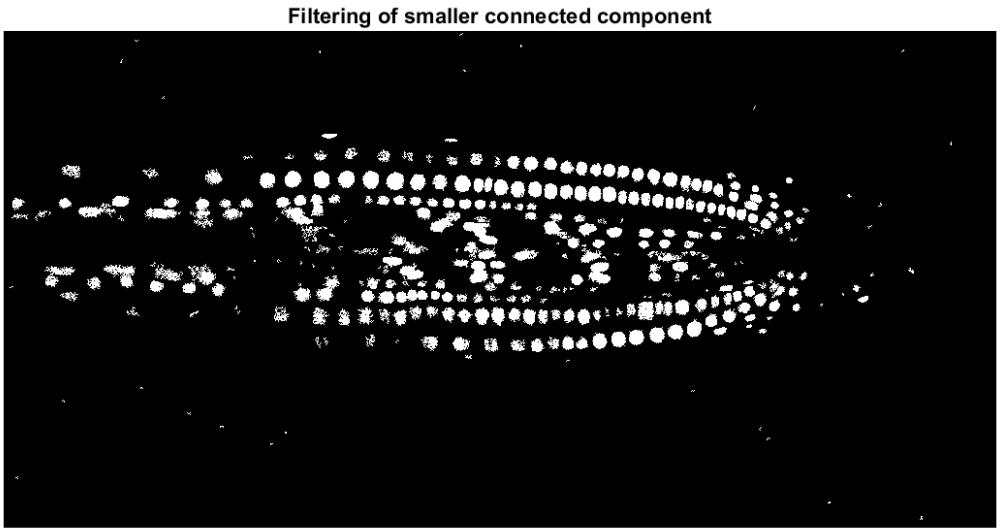

Figure 2.6.1 b: Nuclei 2 smaller connected components

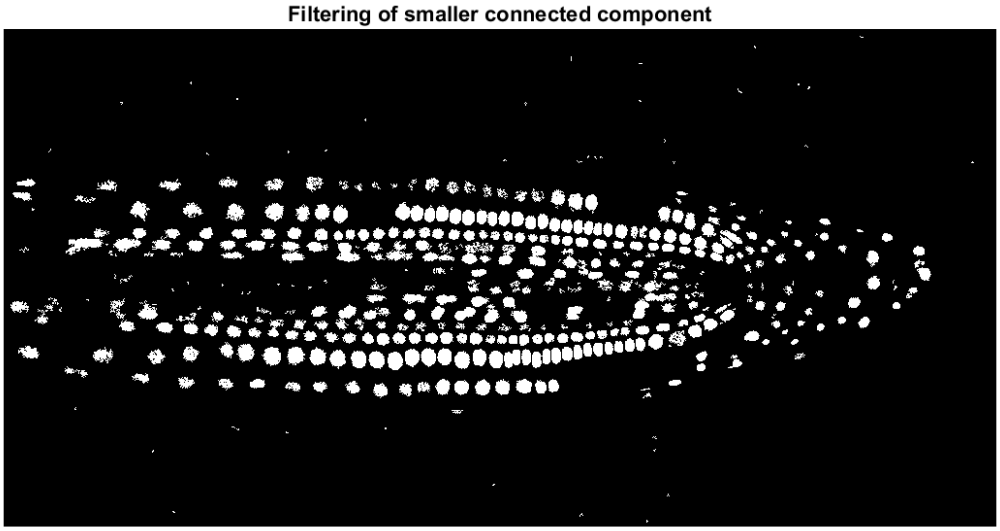

Figure 2.6.1 c: Nuclei 3 smaller connected components

2.6.2 Structuring element

MATLAB's *strel* function is defined with the *disk* shape with a radius size of 1 measured from the origin to create a flat 2-D morphological structuring element. This matrix is a special form of 3-by-3 processing matrix that shows the pixel of an image being processed and defines the neighbourhood used in the processing of each pixel.

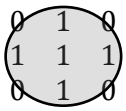

Figure 2.6.2: Disk shape overlaying on the structuring element with radius of 1

2.6.3 First iteration of morphological opening

MATLAB's *strel* function defines a disk shape with a radius size of 1 is used in conjunction with MATLAB's *imopen* function. Any holes present in the processed binary image is then filled using MATLAB's *imfill* function. The filled image is then thresholded using MATLAB's *bwareaopen* function to remove any noise and disjoint connected components smaller than 6 based on 4-neighbourhood argument.

Figure 2.6.3 a: 4-neighbourhood pixel connectivity

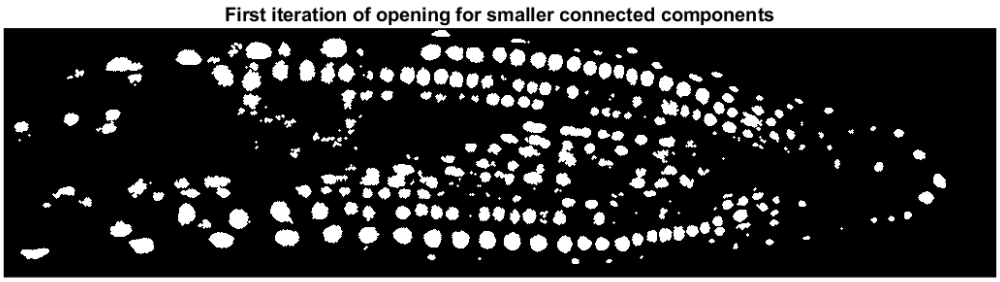

Figure 2.6.3 b: Nuclei 1 first iteration of opening for smaller connected components

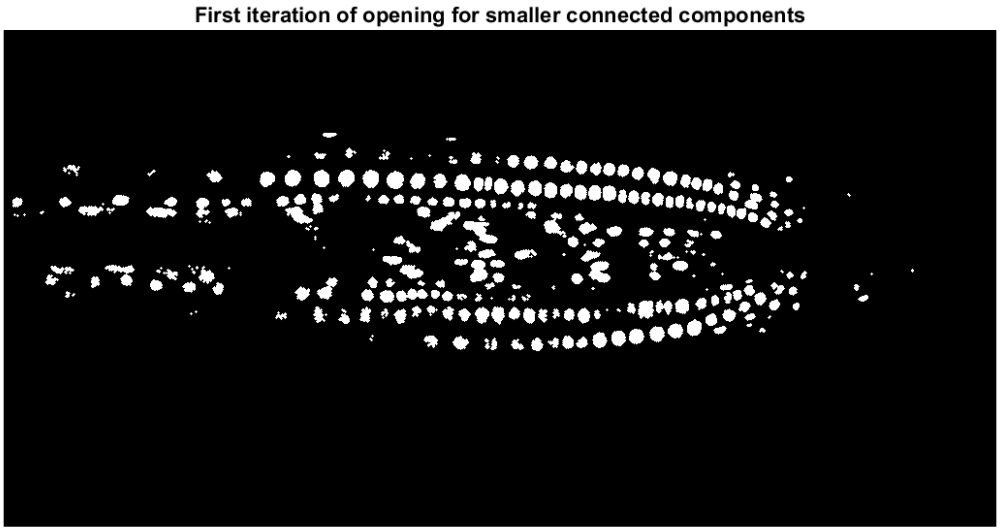

Figure 2.6.3 c: Nuclei 2 first iteration of opening for smaller connected components

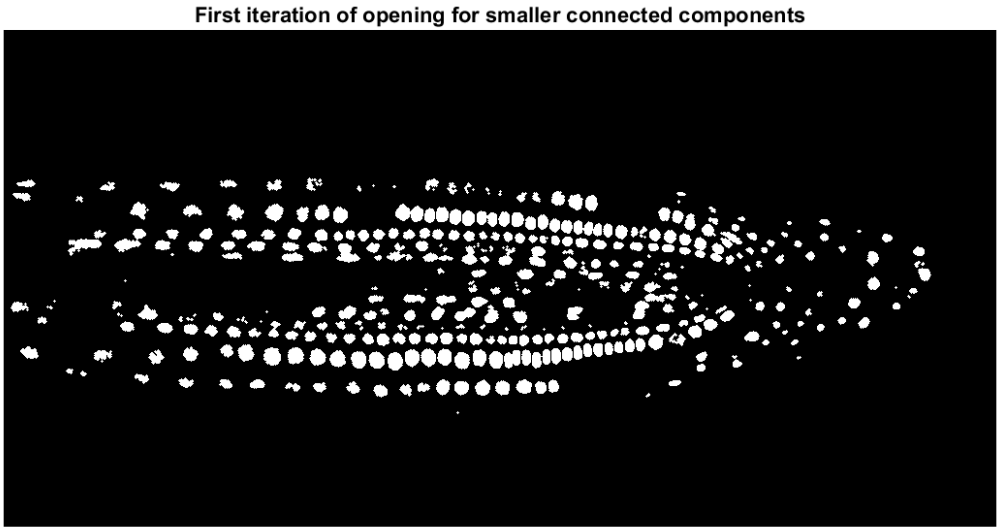

Figure 2.6.3 d: Nuclei 3 first iteration of opening for smaller connected components

2.6.4 Second iteration of morphological opening

The result derived from the first iteration is not satisfactory enough as its visible that some of the nuclei cells are breaking apart. Therefore, a second approach is done using MATLAB's *strel* function defines a disk shape with a radius size of 2 is used in conjunction with MATLAB's *imopen* function. This is done to smoothen out the edges of nuclei cells. Any holes present in the processed binary image is then filled using MATLAB's *imfill* function.

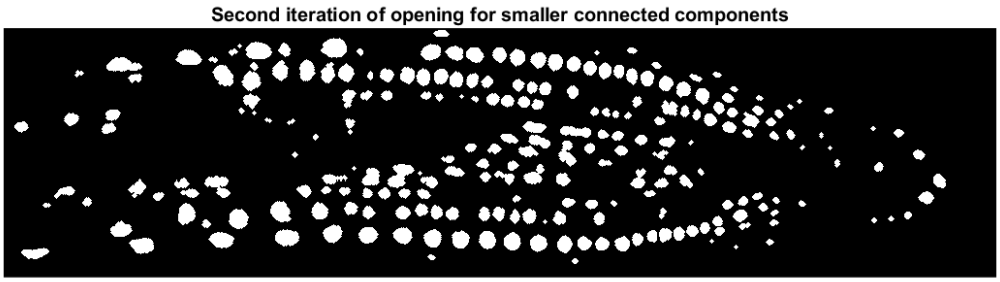

Figure 2.6.4 a: Nuclei 1 second iteration of opening for smaller connected components

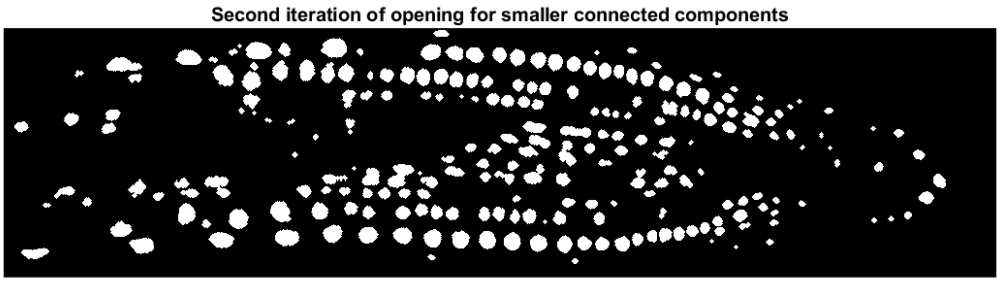

Figure 2.6.4 b: Nuclei 2 second iteration of opening for smaller connected components

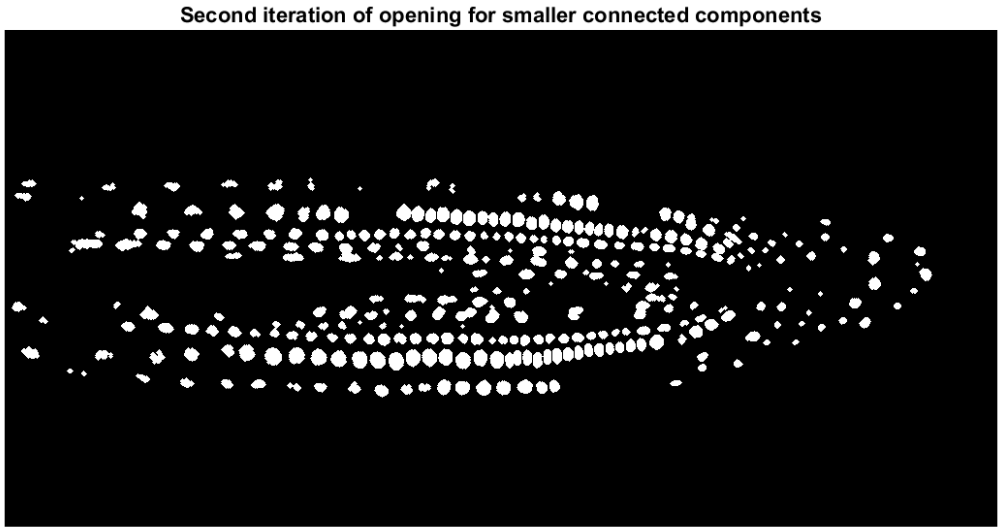

Figure 2.6.4 c: Nuclei 3 second iteration of opening for smaller connected components

2.7 Finalisation for morphological operation

Finalised the morphological operations by concatenating larger and smaller connected components to produce a final binary image before colourisation.

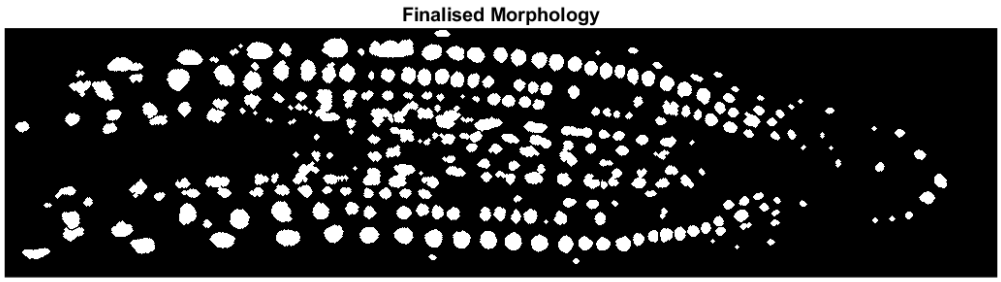

Figure 2.7 a: Nuclei 1 finalised binary image

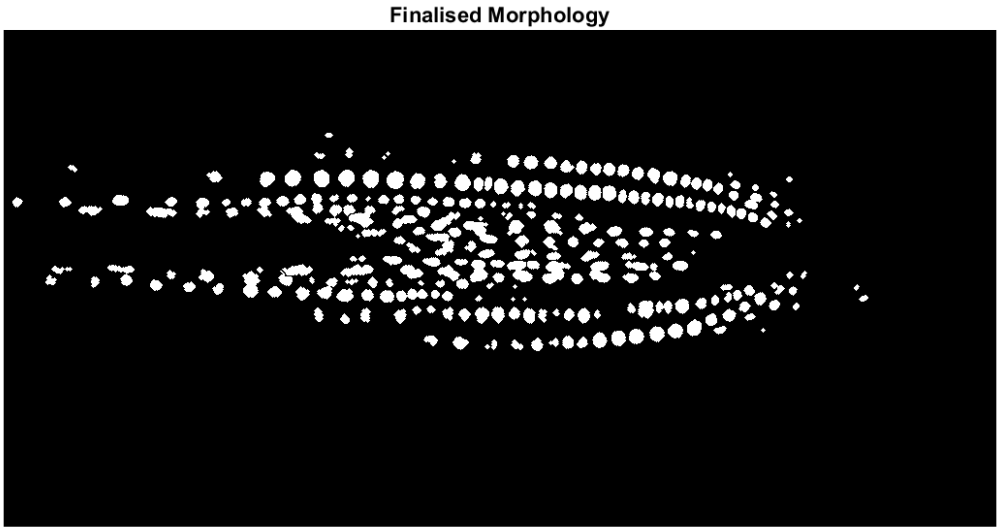

Figure 2.7 b: Nuclei 2 finalised binary image

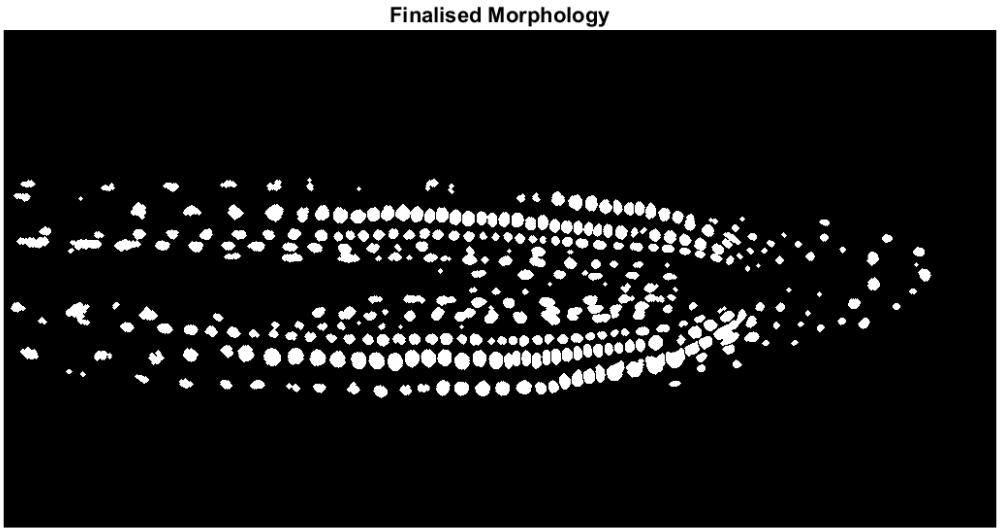

Figure 2.7 c: Nuclei 3 finalised binary image

2.8 Colourisation of nuclei cells

The colourised image is done by labelling connected components in the binary image and a binary image is returned that contains labels from objects of the connected neighbourhood using MATLAB's *bwlabel* function. In order to generate randomised colourised image, MATLAB's *max, rng* and *rand* function generates a random colour map with as many label rows in the label matrix. The generated label matrix is overlaid back onto the finalised binary image in which the colourmap is chosen at random to produce random colour for the nuclei cells after every running iteration.

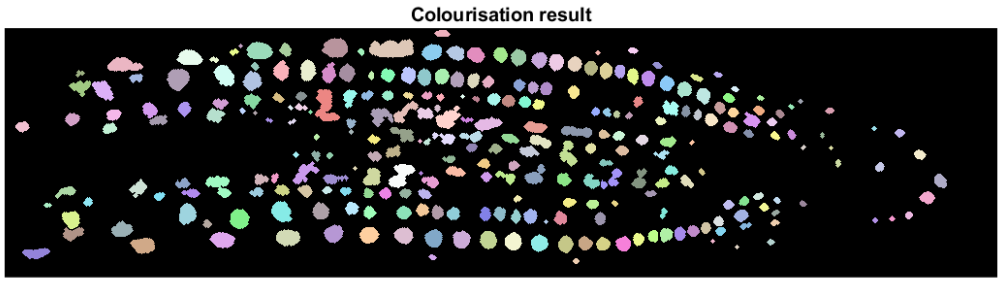

Figure 2.8 a: Nuclei 1 colourisation result

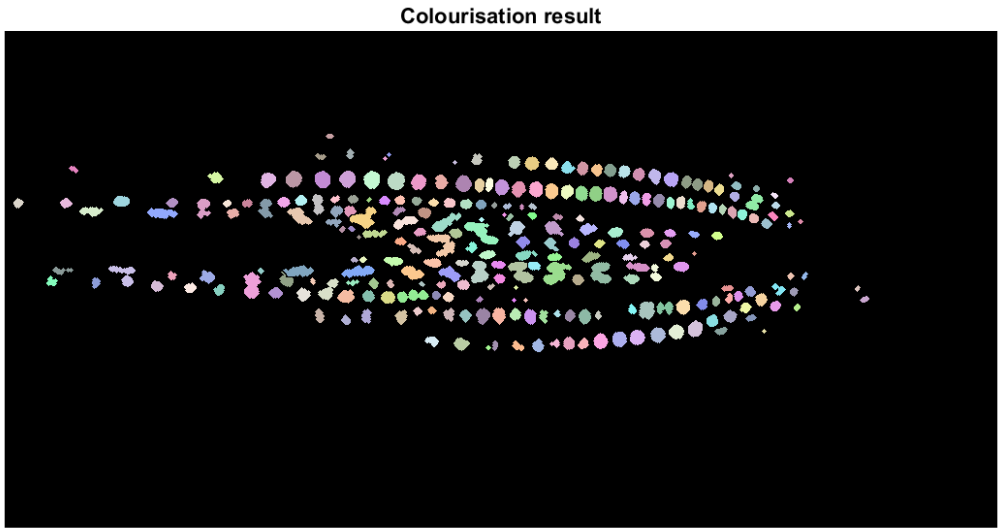

Figure 2.8 b: Nuclei 2 colourisation result

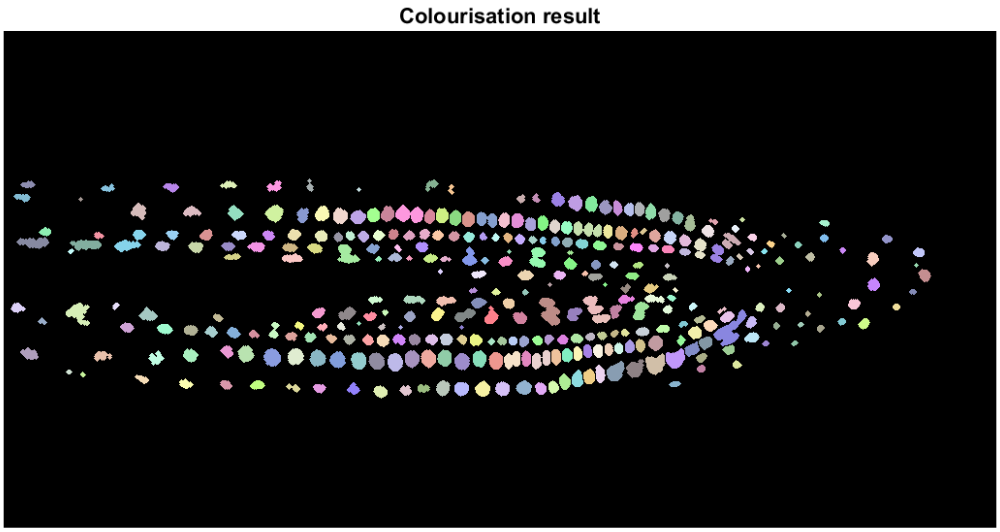

Figure 2.8 c: Nuclei 3 colourisation result

3. Critical Analysis and Value Optimality

The following analysis will be done solely on the original input image of nuclei 1 except for section for the overlaying results, the output results for nuclei 2 and 3 are similar as nuclei 1.

3.1 imlocalbrighten

Several MATLAB functions such as *imadjust*, *histeq*, *adapthisteq* and *imlocalbrighten* was compared. *imadjust* performs the worst amongst all due to lack of enhancement for the green nuclei cells, *histeq* performs histogram equalisation so that the histogram of the output image approximately matches uniformly distributed histogram but in return a much poorly contrasted image is obtained, *adapthisteq* performs contrast-limited adaptive histogram equalisation as it does not operate on the entire image, the result turns out to be worse than *histeq* as some noise is amplified. *imlocalbrighten* is suitable in performing contrast adjustment function comparing against its counterparts. This is due to *imlocalbrighten* accomplishes the resulting output image by returning the proportionality of the estimated darkness intensity for each pixel in the input image. This technique performs exceptionally well at enhancing low-light images.

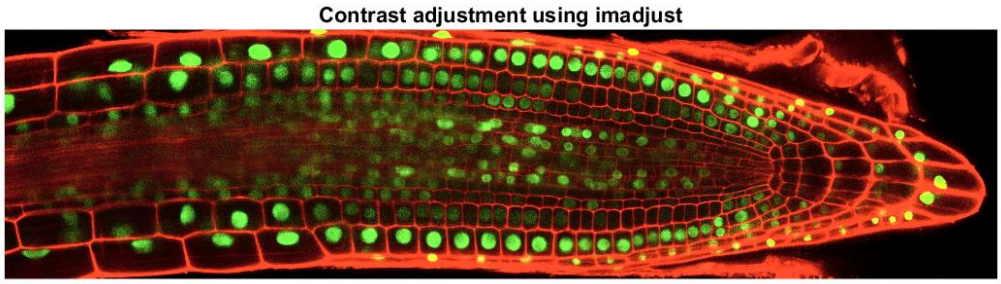

Figure 3.1 a: Contrast adjustment using imadjust function

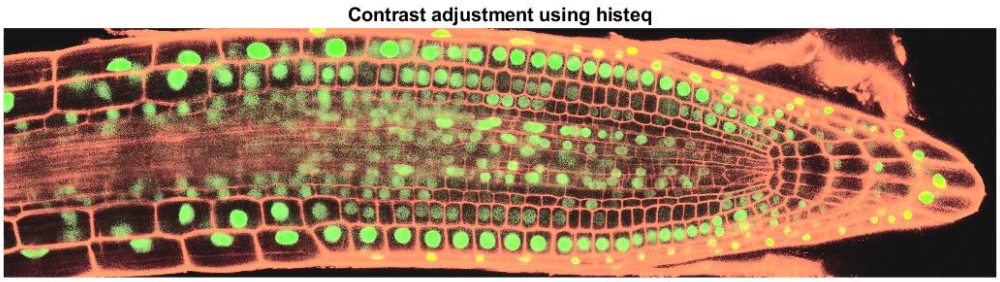

Figure 3.1 b: Contrast adjustment using histeq function

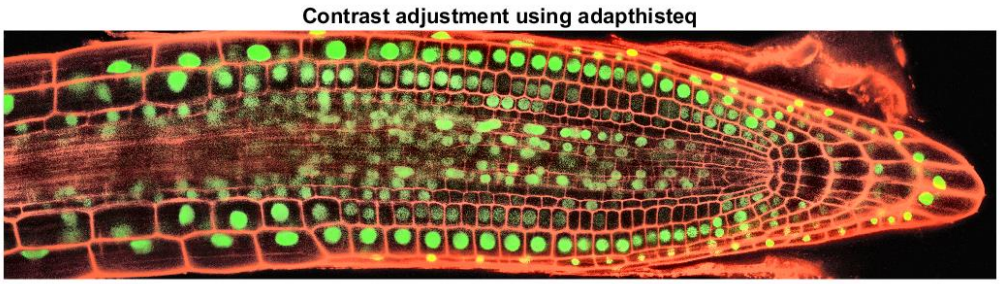

Figure 3.1 c: Contrast adjustment using adapthisteq function

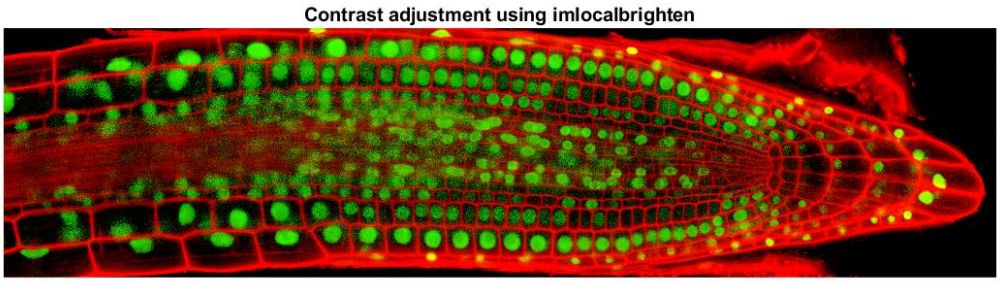

Figure 3.1 d: Contrast adjustment using imlocalbrighten function

3.2 CIE L*a*b*

CIE L*a*b* is compared against the likes of RGB, HSV and Y'CbCr to select the best-performing colour space that separates the green nuclei cells from the red nuclei cell wall. Studies have shown that the CIE L*a*b* colour space is perceptually uniform and device independent, results are consistent across display regardless of its colour gamut accuracy. This means that it is easier to extract the green nuclei cells from the red nuclei cell wall. CIE L*a*b* treats black and white as own opponent channels, more commonly interpreted as separating contrast from colour. It also has a wider colour space than other colour spaces which allows the programmer to perform greater colour shifts by not affecting the overall contrast which most colour spaces lacks. The biggest advantage that CIE L*a*b* equips which other colour spaces lack is the high accuracy of estimations for the output colour even of the slightest colour differences.

3.3 Greenness

The green channel is emphasized in order to distinguish the green nuclei cells apart from the surrounding speckle noise by verifying each individual pixel's green channel values. A comparison against the weighted average formula for grayscale is done to show that greenness is the better representing image.

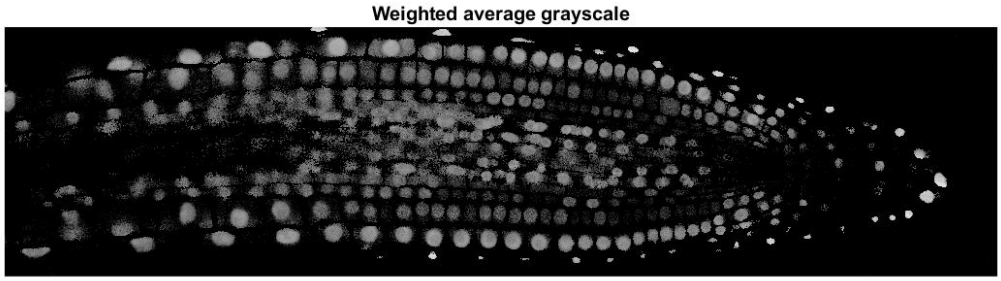

Figure 3.3 a: Grayscale computed using the weighted average formula

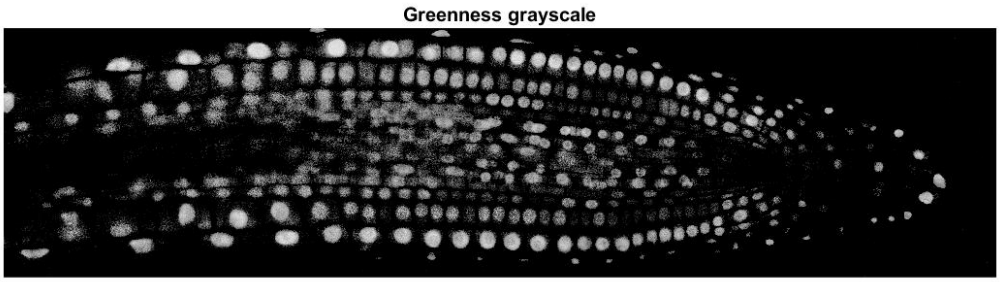

Figure 3.3 b: Grayscale computed using the greenness formula

3.4 Bilateral filtering

As stated by Sylvain Paris, Pierre Kornprobst, Jack Tumblin and Frédo Durand (Paris et al. 2008), bilateral filtering is capable for the decomposition of an image into different scales without causing haloes after modification has made it ubiquitous in image processing. The advantage for choosing bilateral filtering is it is a non-linear filtering technique commonly used in the removal of noise that are not additive and smoothens the image while preserving its strong edges. The result shows a sharpen edge contour and denoising of the overall image. The filtered image result is adaptive in which the size of the kernel surrounding the initial image varies according to which better output result is obtained. It does not replace all the pixel values from the greenness image, i.e., retaining finer details in the edges of the nuclei cells and improving the visual quality of the nuclei cells. Comparisons are done against median, Gaussian, and fast local Laplacian filtering techniques for justification.

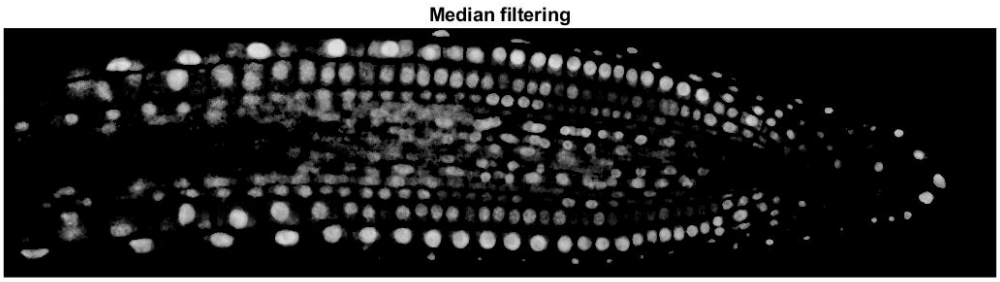

Figure 3.4 a: Median filtering

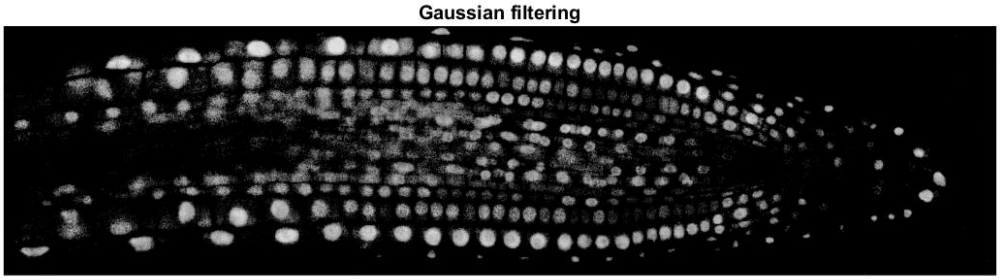

Figure 3.4 b: Gaussian filtering

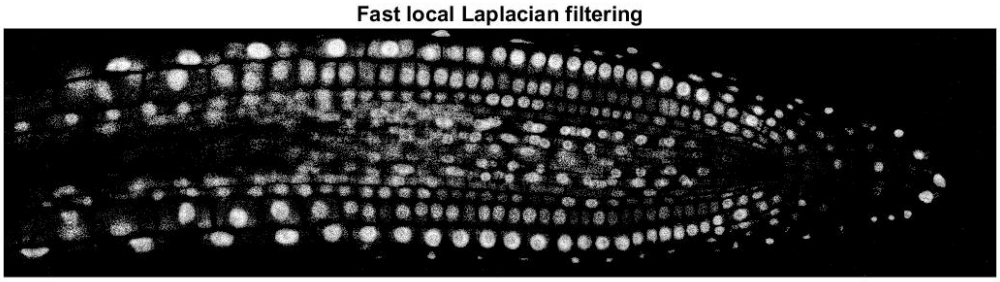

Figure 3.4 c: Fast local Laplacian filtering

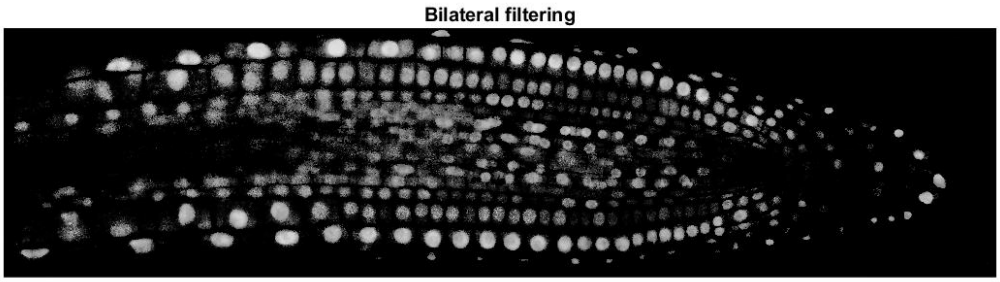

Figure 3.4 d: Bilateral filtering

3.5 adaptthresh

MATLAB's *adaptthresh* function computes a locally adaptive threshold for the bilateral filtered image. The threshold optimum value is adding 0.03 to the *adaptthresh* function after comparing against no addition and adding 0.06 .The function chooses the threshold based on the local mean intensity in the neighbourhood of each pixel. It provides a more accurate representation of the image by preserving details in both and bright and dark regions. Comparisons are done against *graythresh* and *otsuthresh*.

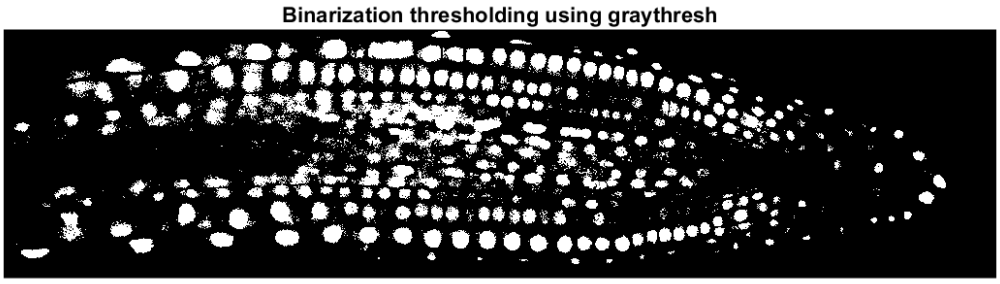

Figure 3.5 a: Thresholding using graythresh

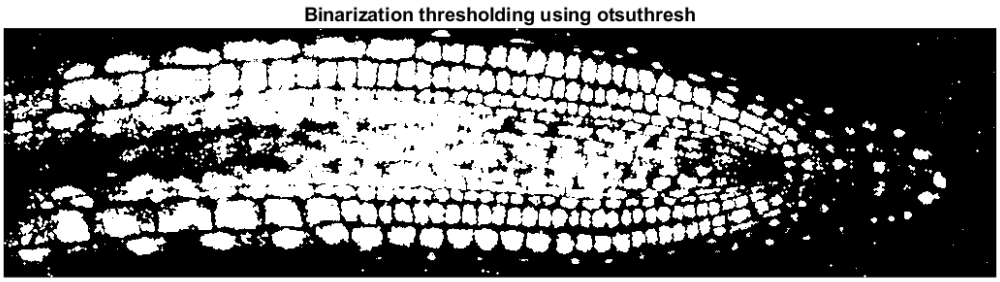

Figure 3.5 b: Thresholding using otsuthresh

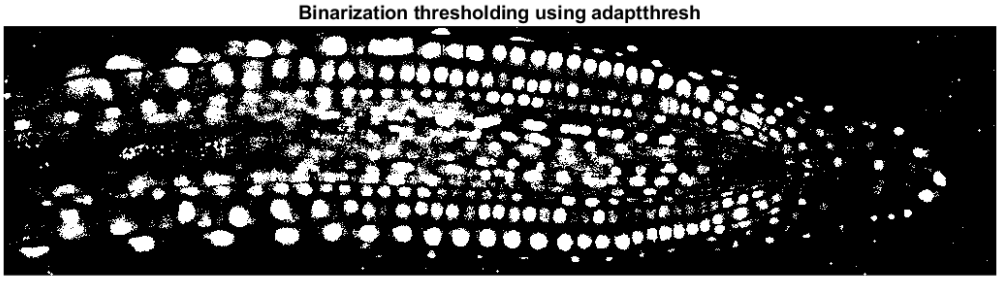

Figure 3.5 c: Thresholding using adaptthresh

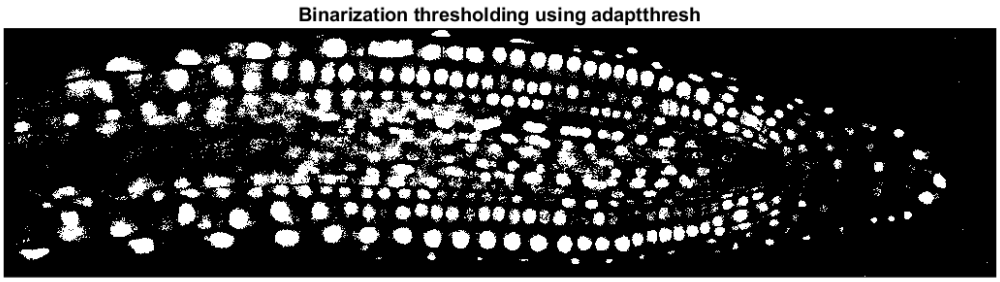

Figure 3.5 d: Thresholding using adaptthresh + 0.03

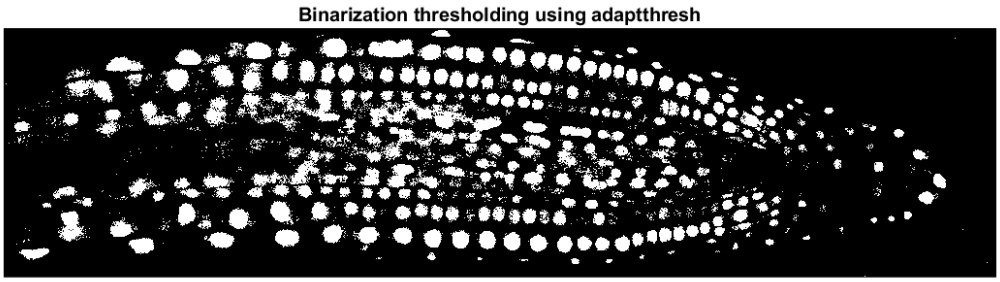

Figure 3.5 e: Thresholding using adaptthresh + 0.06

3.6 Morphological opening

The circular and ellipsoid like objects are being regarded as green nuclei cells as shown in Figure 2.5.3 a, Figure 2.6.3 a and Figure 2.6.4 a. Based on the article published, opening smoothens the edges of the objects in the binary image, making it easier to detect, analyse and segment (Said, Jambek, & Sulaiman, 2016). The optimal sizes for the structuring element used is both 1 and 2. This is crucial as any larger area of opening is excessive and results in a decrease of the number of nuclei cells.

3.7 bwareafilt

Strengths of *bwareafilt* allow programmers to extract region of interest to perform operations while not interfering with the non-interested regions. The downside of using *bwareafilt* is the amount of trial and error in order to obtain the optimum value.

3.8 bwareaopen

The MATLAB function *bwareaopen* is performed on the smaller objects to remove disjoint connected components from a binary image that have fewer than a specified number of pixels in the defined neighbourhood. This effectively removes noise while preserving the larger objects. The optimum value for the area size is 6 after comparing against 10 and 15.

Figure 3.8 a: bwareaopen with area size of 6

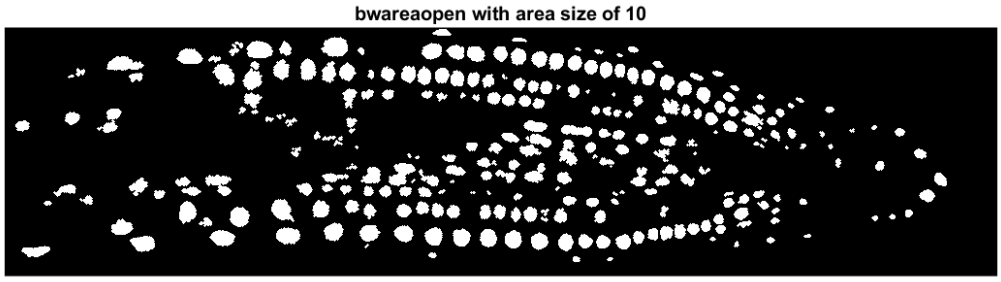

Figure 3.8 b: bwareaopen with area size of 10

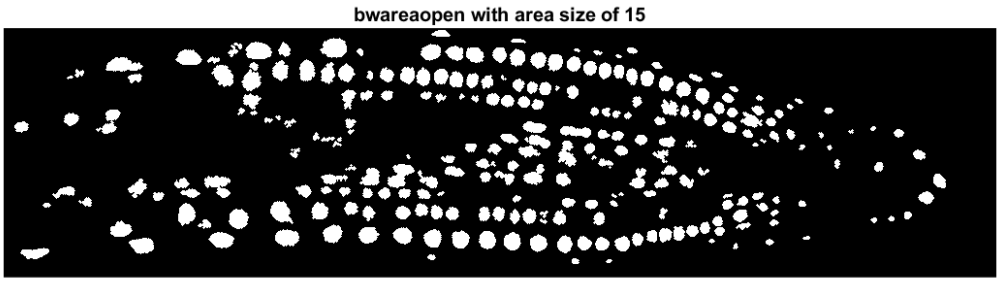

Figure 3.8 c: bwareaopen with area size of 15

3.9 Watershed segmentation

Kornilov, Safonov and Yakimchuk had stated that one of the uses of watershed is separation of touching or overlapping objects in a binary image to employ an instance segmentation. (Kornilov et al., 2022). The algorithm treats pixels values as a local topography to produce closed region, which is useful for separating connected components in a binary image. (Preim & Botha, 2014). Watershed segmentation locate and separates most of the larger connected nuclei cells accurately.

As mentioned on Peter's article (Eggleston, 1998), over-segmentation which the objects being segmented from the background are being fractured into subcomponents. This results in an increase of chance that boundaries of importance creating many insignificant boundaries. However, this is minimised as watershed segmentation is applied solely to the larger connected components of the binary image. The MATLAB's *imextendedmin* and *imimposemin* functions are applied to reduce the chance of over-segmentation. It is important to note that certain connected components are left unsegmented to prevent over-segmentation as shown in Figure 2.5.4 a, Figure 2.5.4 b and Figure 2.5.4 c.

3.10 Distance transformation

As stated Grevera it is important to develop a morphological methodology to verify and produce results with minimal errors (Grevera, 2007). Therefore, the morphological operation performed before implementing watershed segmentation has to yield a good result. The distance transformation is computed using Euclidean's method. The distance transform method assigns each pixel in the foreground of the binary image to the nearest background pixel. Euclidean distance performs better than other distance transformation argument methods such as *cityblock*, *chessboard* and *quasi-euclidean* using the same parameters as in *euclidean.*

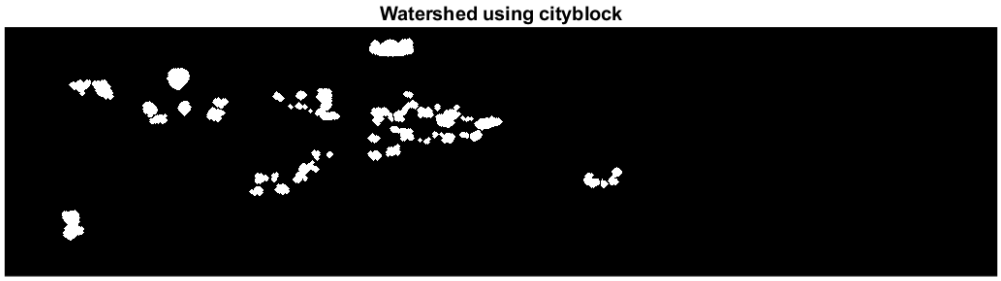

Figure 3.10 a: Distance transformation using cityblock

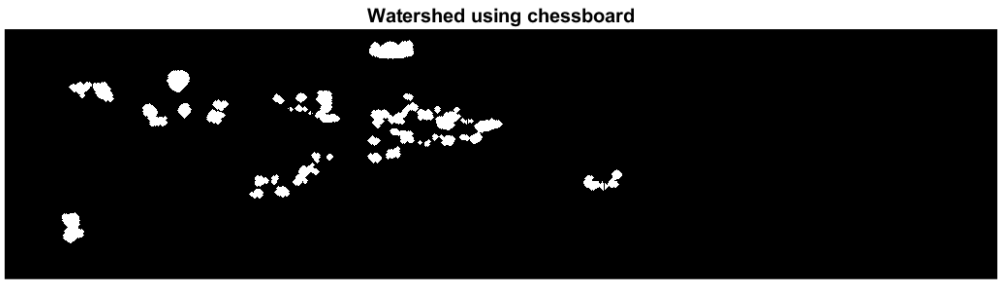

Figure 3.10 b: Distance transformation using chessboard

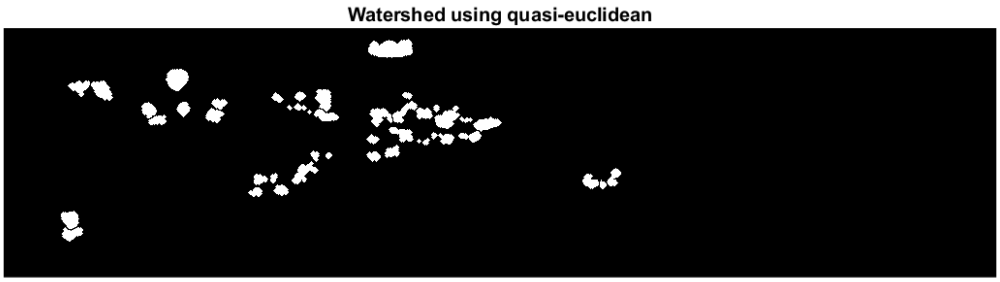

Figure 3.10 c: Distance transformation using quasi-euclidean

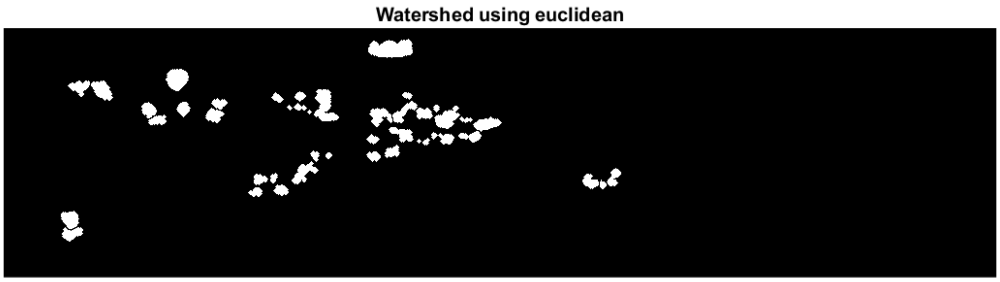

Figure 3.10 d: Distance transformation using euclidean

3.11 Overlaying of morphological operation

The contrast adjusted images shown under section 1.2 is overlaid onto the finalised binary images shown under section 2.7 to verify the accuracy of processing performed. The accuracy of detection is highly accurate as objects dominated by non-green colours are insignificant and mostly appearing on the edge which is a good performing trade-off for using morphological operations.

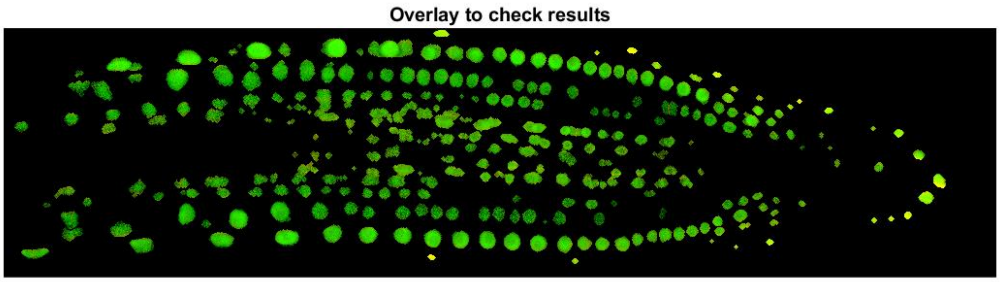

Figure 3.11 a: Nuclei 1 nuclei cell detection

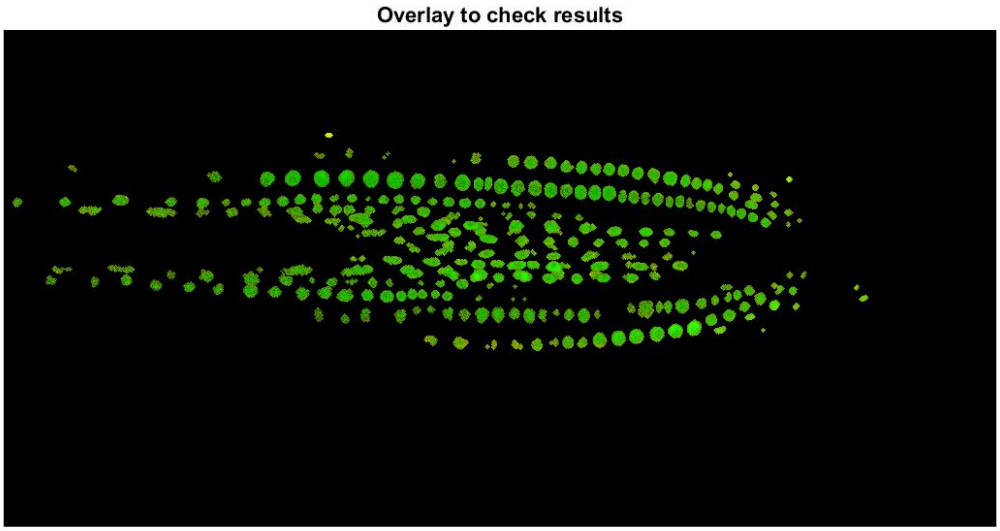

Figure 3.11 b: Nuclei 2 nuclei cell detection

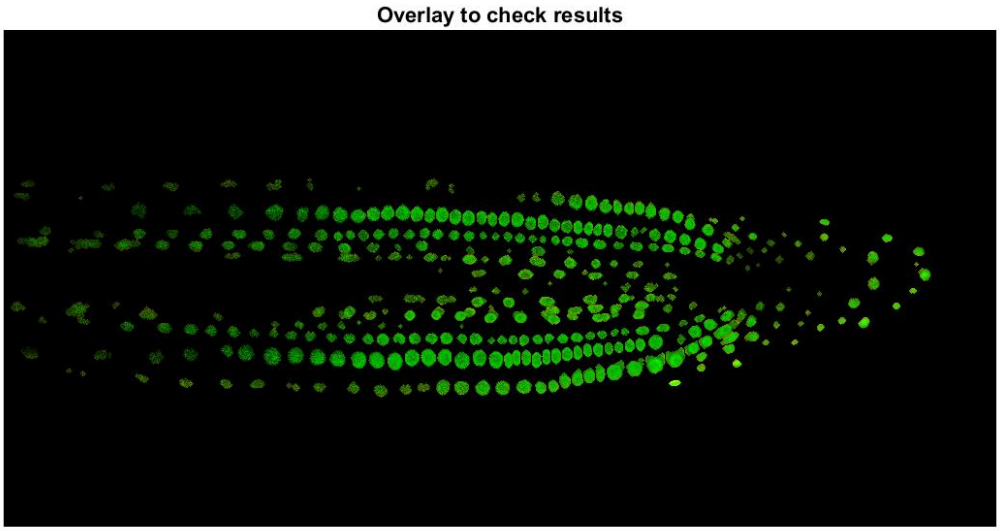

Figure 3.11 c: Nuclei 3 nuclei cell detection

4. References

Eggleston, P. (1998). Understanding Over segmentation and Region merging. *Vision Systems Design*. https://www.vision-systems.com/non-factory/security-surveillance-transportation/article/16739494/understanding-oversegmentation-and-region-merging

Grevera, G. J. (2007). Distance Transform Algorithms and Their Implementation and Evaluation. *Springer EBooks*, 33–60. https://doi.org/10.1007/978-0-387-68413-0_2

Kornilov, A., Safonov, I. V., & Yakimchuk, I. (2022). A Review of Watershed Implementations for Segmentation of Volumetric Images. *Journal of Imaging*, *8*(5), 127. https://doi.org/10.3390/jimaging8050127

Novak, C. L., Shafer, S. A., & Willson, R. G. (1992). Obtaining accurate color images for machine vision research. *Physics-Based Vision: Principle and Practice-Color*, 13-27. https://www.researchgate.net/publication/270956937_Understanding_Color_Image_Processing_by_Machine_Vision_for_Biological_Materials

Paris, S., Kornprobst, P., Tumblin, J., & Durand, F. (2008). Bilateral Filtering: Theory and Applications. *Foundations and Trends in Computer Graphics and Vision*, *4*(1), 1–75. https://doi.org/10.1561/0600000020

Preim, B., & Botha, C. P. (2014). Image Analysis for Medical Visualization. *Elsevier EBooks*, 111–175. https://doi.org/10.1016/b978-0-12-415873-3.00004-3

Said, K. A. M., Jambek, A. B., & Sulaiman, N. (2016). A Study of Image Processing Using Morphological Opening and Closing Processes. *International Journal of Control Theory and*

Applications, 9(31), 15-21. https://asral.unimap.edu.my/wp-content/uploads/2018/04/2016_IJCTA_opening_closing_final_manuscript.pdf