Report on Introduction to Image Processing Coursework

Title: Extracting & Analysing Cell Nuclei

Module Convenor: Dr. Amr Ahmed

From:

Wong Qing Joe

10268818

Computer Science with Artificial Intelligence

Date of Submission:

24/4/2021

Function: solution.m

Input: RGB image , *im_in (3D uint8 matrix)*

Output:

i) Binary image marking regions corresponding to nuclei , *out (2D logical matrix)*

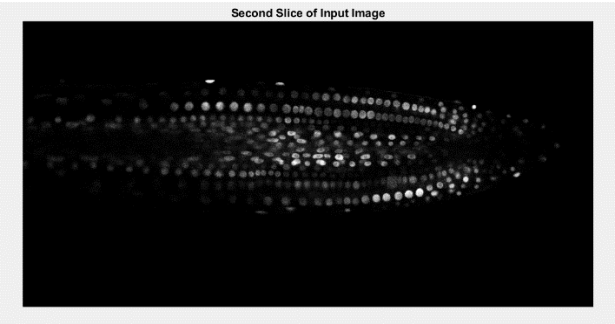
ii) Total count of nuclei detected , *nr (int)*

iii) Nuclei configuration: Sizes, shapes and brightness, *config (struct)*

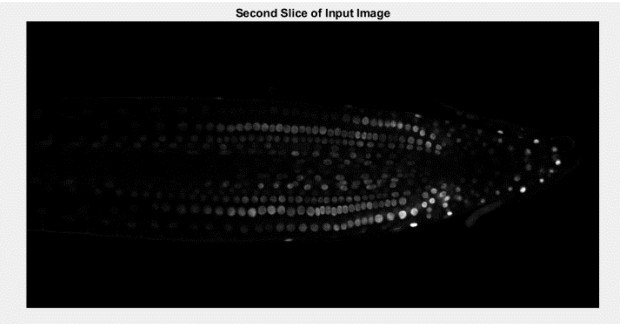
I) Getting Binary Nuclei Image

Since they are all RGB image, these images are represented as a 3D matrix with 3 slices, each slice representing Red, Green and Blue respectively. We can extract the slice (the second slice) corresponding to the green image, which contains only the nuclei. Since it is a 2D matrix, it would be a gray scale image.

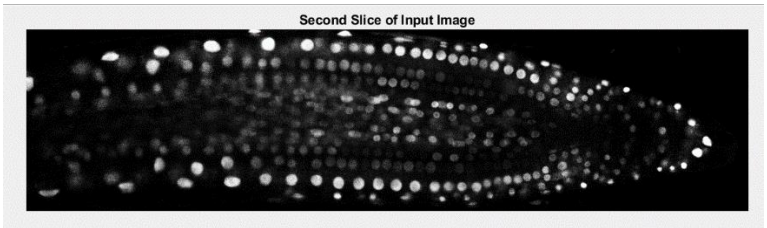
(a) Image 1:



(b) Image 2:

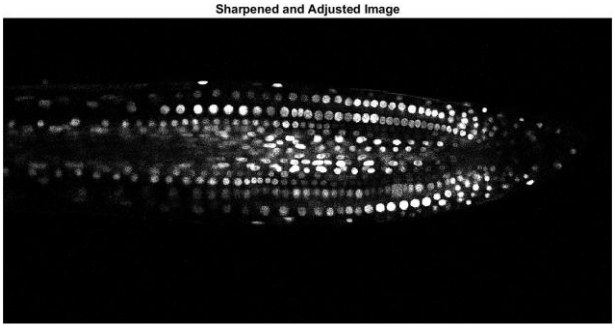


(c) Image 3:

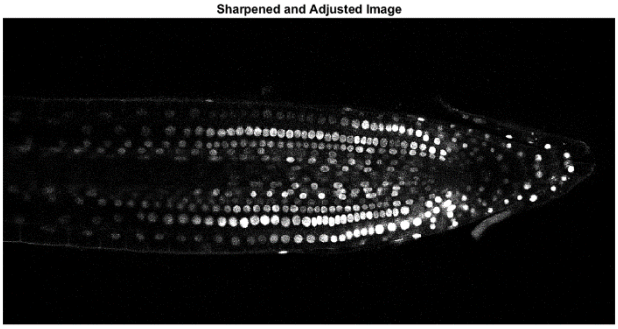


I used *imadjust* to adjusts the image intensity by saturating the bottom 1% and the top 1% of all the pixel values, hence increasing the contrast of the image. Then I used *imsharpen* to sharpen the image f1 using unsharp filtering.

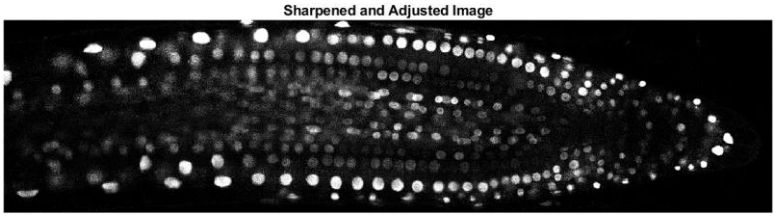
(a) Image 1:



(b) Image 2:

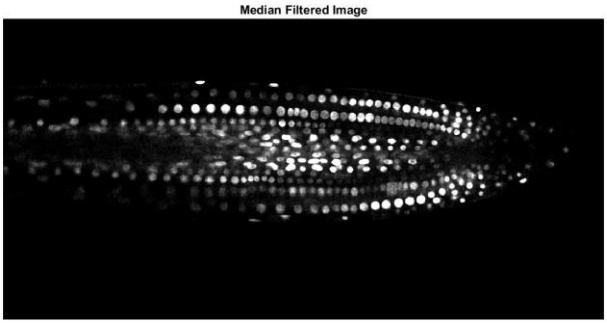


(c) Image 3:

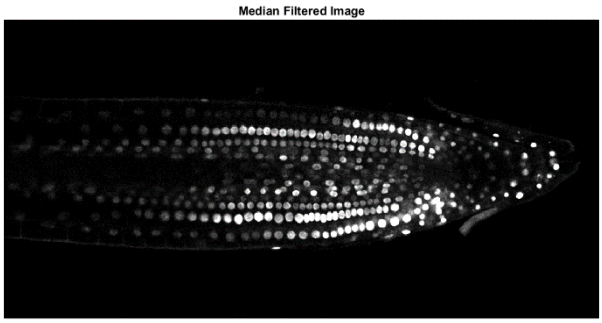


Then, using *medfilt2* filters the image by sorting the pixel values within the mask and take the middle value (median). Median filter is chosen because it is a non-linear filtering, and it seems to preserve the sharp image changes.

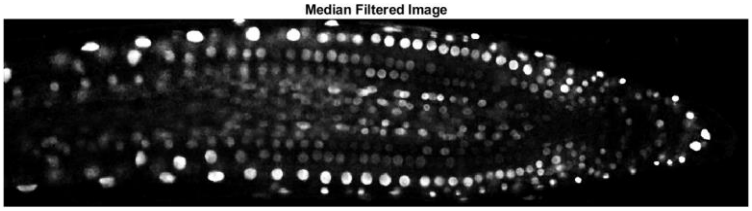
(a) Image 1:



(b) Image 2:



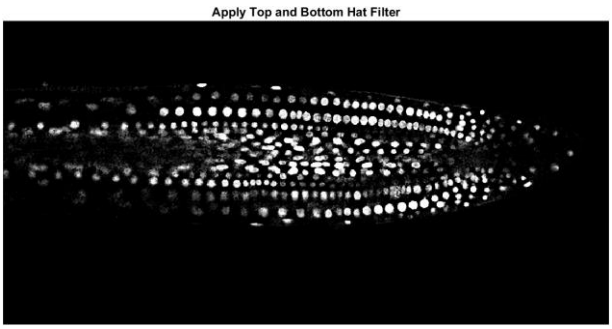
(c) Image 3:



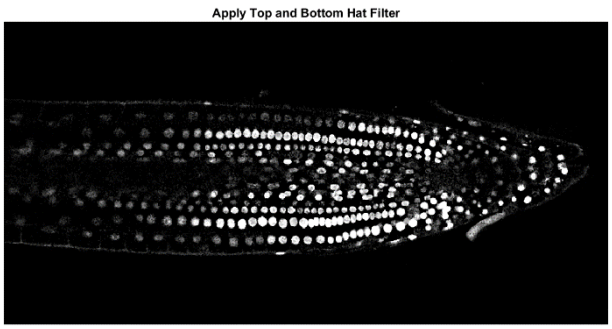
Then I did top and bottom hat filtering. The top hat filter is taking the image and subtract it with a morphological opening (erosion followed by dilation) of the image, while the bottom

hat filter is taking a morphological closing (dilation followed by erosion) of the image and subtract it with the original image, both using some structuring element. The output of the image is taking the original image, adding the top hat filter and subtracting the bottom hat filter, this way the image is more precise and the details of the nuclei are more obvious.

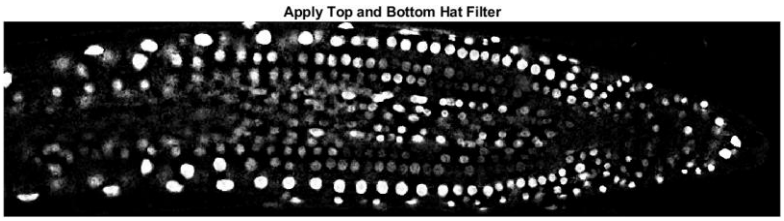
(a) Image 1:



(b) Image 2:

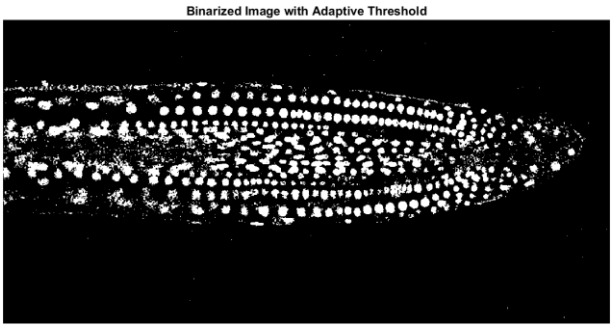


(c) Image 3:

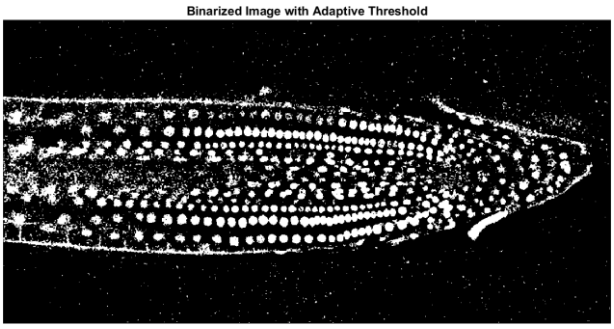


To convert to binary image, I used the function *imbinarize*, and adaptive threshold values. Threshold is chosen based on local mean intensity in the neighbourhood of each pixel. This way, some really dim nuclei can be detected.

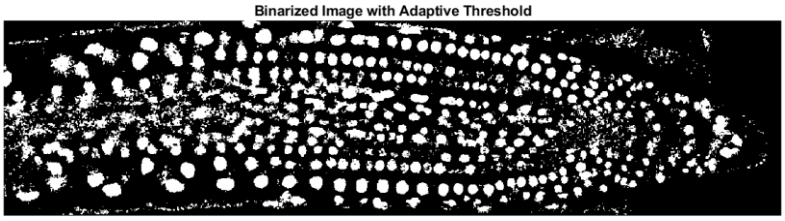
(a) Image 1:



(b) Image 2:

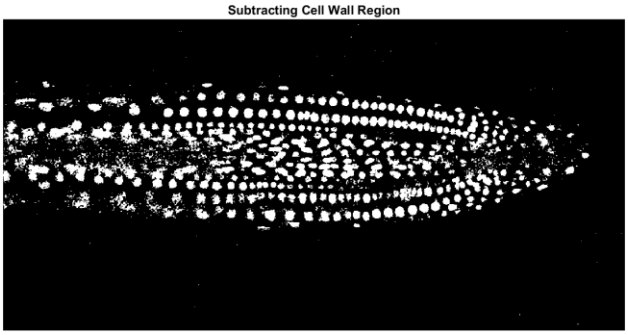


(c) Image 3:

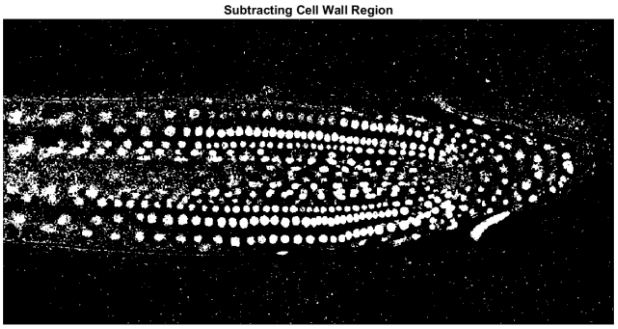


These images show the cell wall regions which we don't want (especially obvious in Image 2). We know that regions associated to the cell wall that are red in colour, which can be extracted in the first slice of the original image. But I would first remove the noise using median filtering on the first slice, then minus it with the nuclei image we got to get only the cell wall region. I then convert it to binary image using *imbinarize* with a global threshold. Then, simply minus our binarized images with the cell wall region. We also need to make sure that after performing the subtraction, we have to make pixels with negative value (when 0 - 1) to become 0 in binary image.

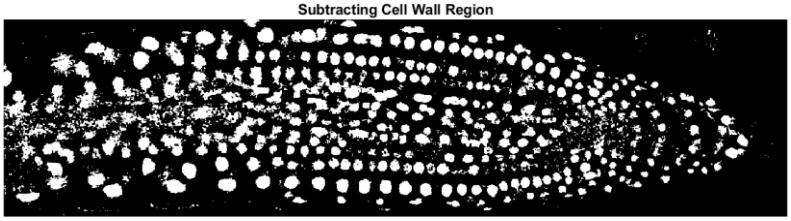
(a) Image 1:



(b) Image 2:

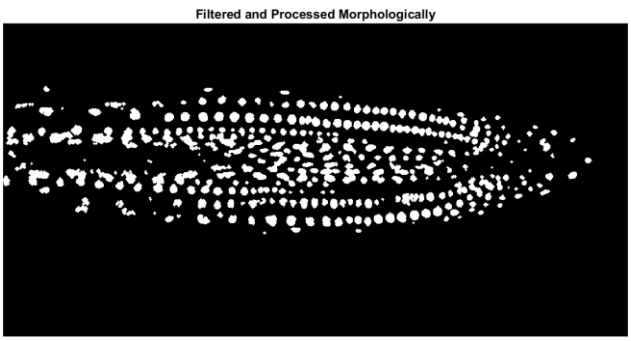


(c) Image 3:

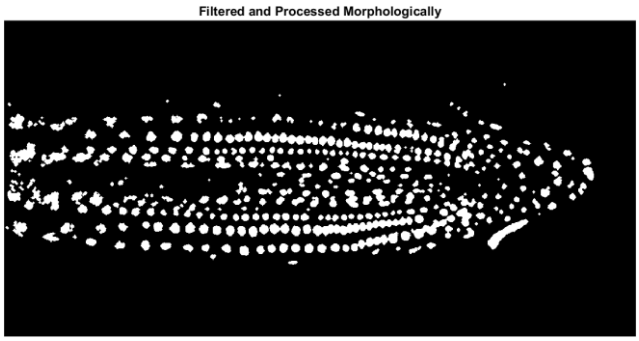


I first removed the noise using median filter, and then emphasized each of the nucleus by doing erosion then dilation (opening) on each of the pixel, with another structuring element with a shape disk and radius of 2.

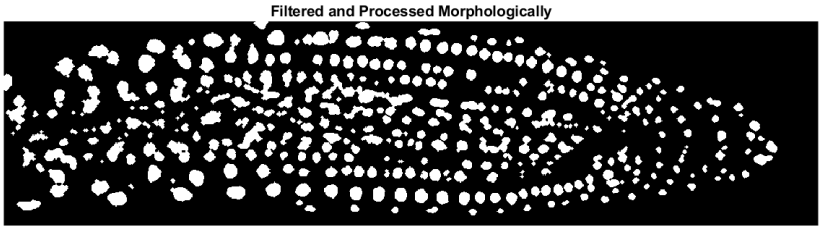
(a) Image 1:



(b) Image 2:



(c) Image 3:



Some of the nuclei are merged together. So I implemented the watershed algorithm to separate them. I first invert the logical value of the images (1 becomes 0, 0 becomes 1), now the nuclei are black, and the background is white. Then, I calculated the distance of each black pixel to the nearest white pixel, using distant transform function *bwdist*. The further away the black pixel is to a nearest white pixel, the larger the value. Then I put a negative sign on the values to create a basin for every object, with the centre being the lowest point of the basin.

(a) Image 1:

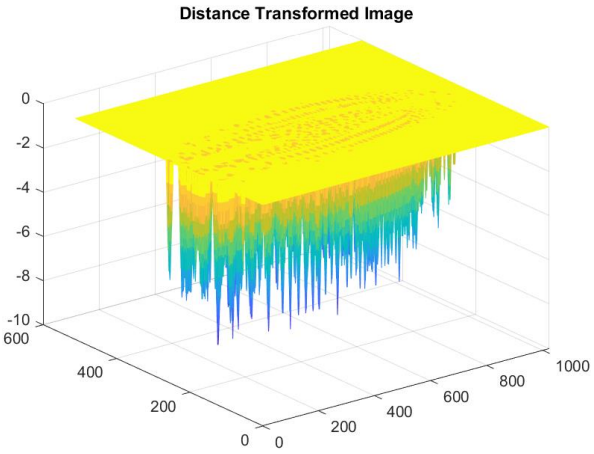
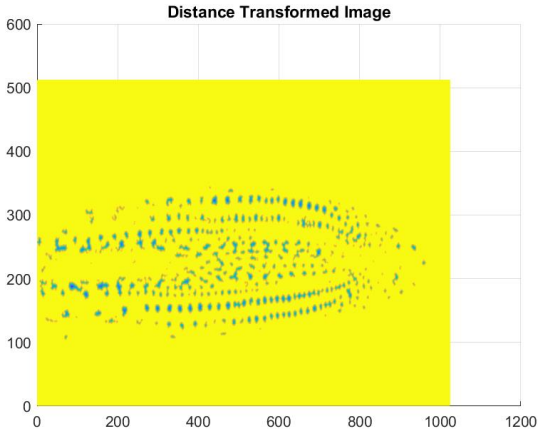


Image 1 (top view) :



(b) Image 2:

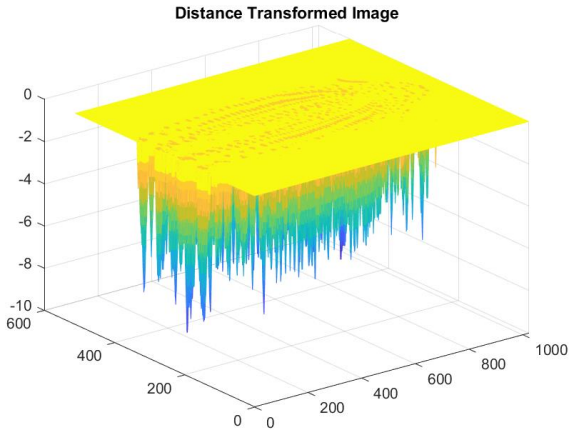
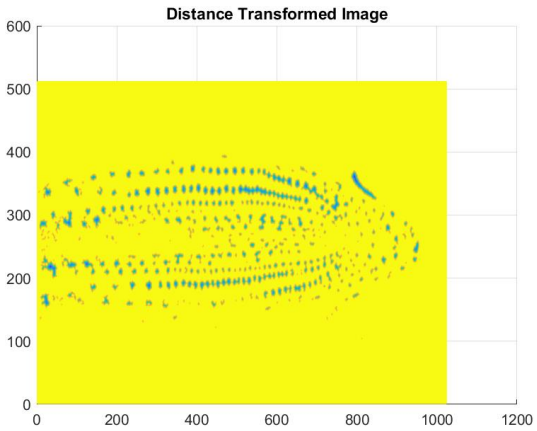


Image 2 (top view) :



(c) Image 3:

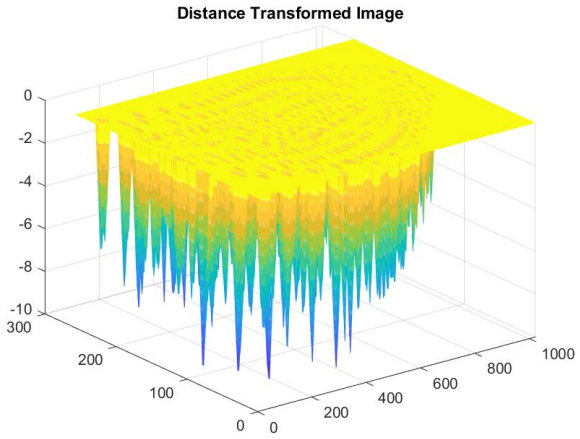
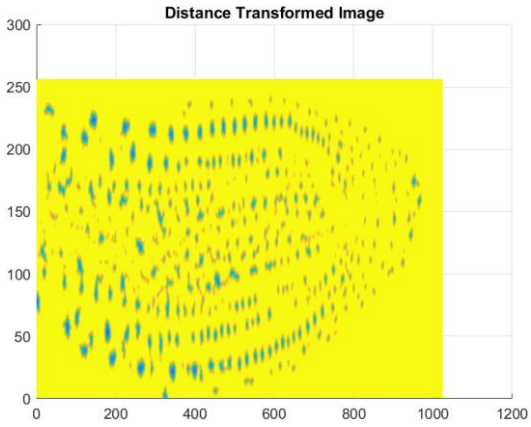


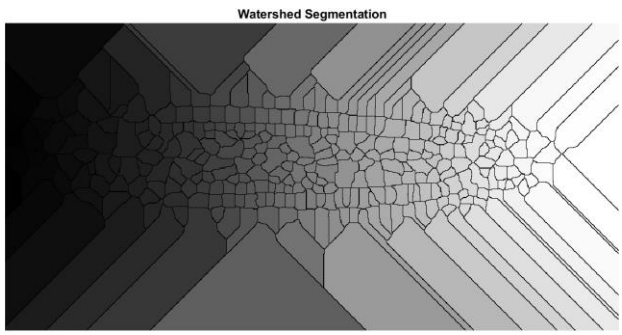
Image 3 (top view) :



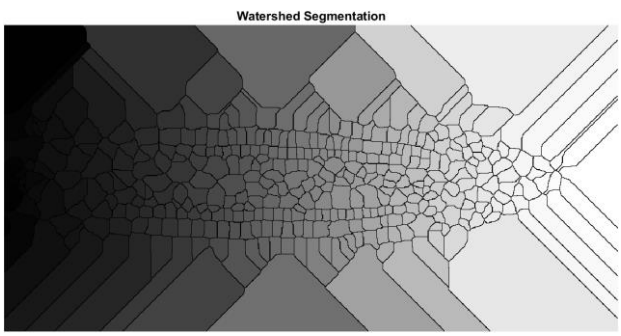
We can imagine filling these “basins”, each corresponding to an object (nucleus), with water in order to separate them. But this is not perfect, there are small intensity local minima in the basin that will over-separate regions after water-shedding. To remove the minima, we need to suppress the minima in D whose depth is less than some value (2 for example). This can be done using the function *imhmin*.

We can then apply the watershed algorithm using the function *watershed* to create different segment of region.

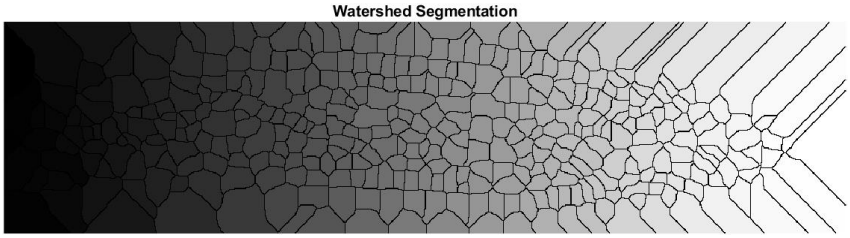
(a) Image 1:



(b) Image 2:



(c) Image 3:



After getting the segment image, I use logical indexing to compare the segment image to our nuclei image. If there is a 0 in the segment image (a ridge), then the pixel in our nuclei image should be 0 as well. Then there are some small holes within a nuclei that should be filled, so I use *imfill* to fill up the holes in the nuclei image, and here are the results:

(a) Image 1 (binary):

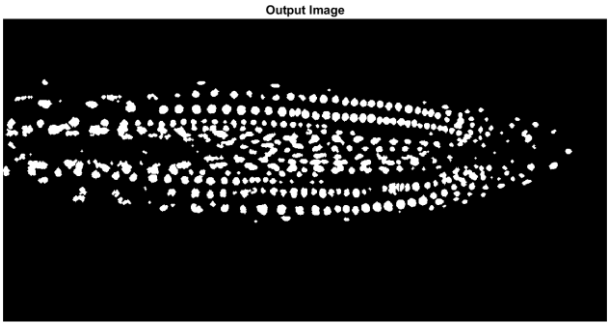
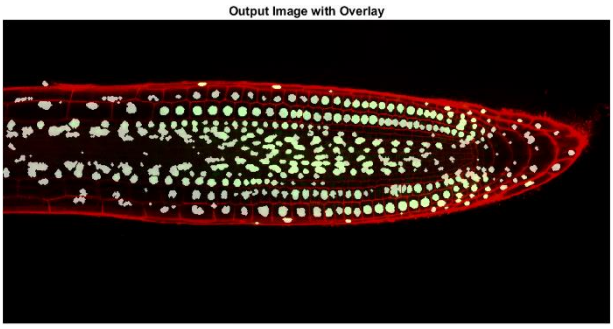


Image 1 (overlay):



(b) Image 2 (binary):

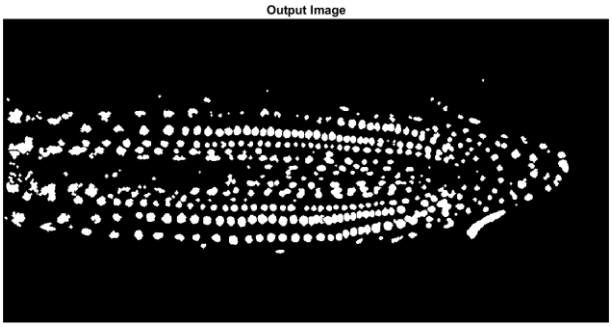
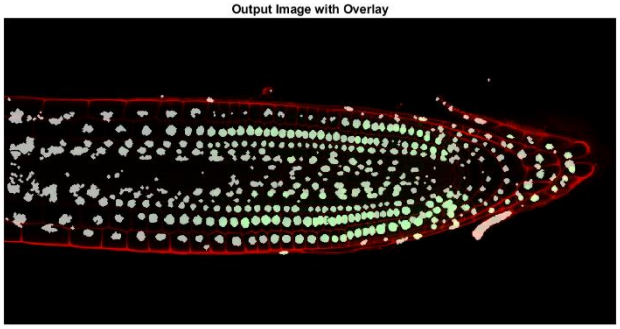


Image 2 (overlay):



(c) Image 3 (binary):

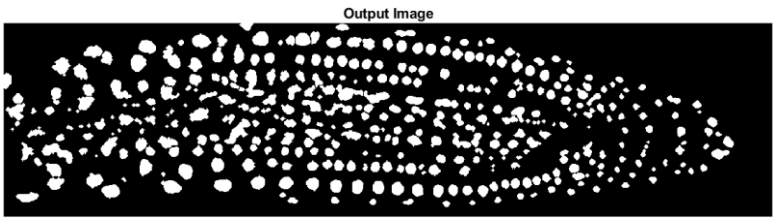
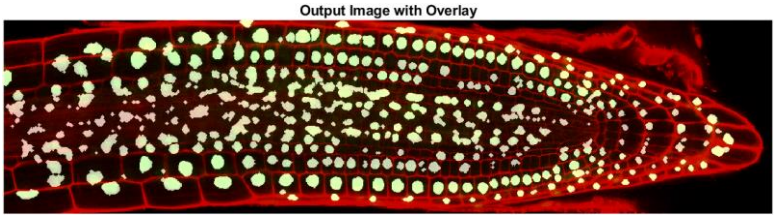


Image 3 (overlay):



II) Counting Nuclei

I use the function *bwlabel* to count the objects in the binary image. *bwlabel* function returns 2 output, the label matrix that contains the 8-connected objects found in the binary image, and the total number of connected objects found, which in this case is the total number of nuclei we have. We will have to assume that connected objects are counted as one nucleus even there should be more than 1.

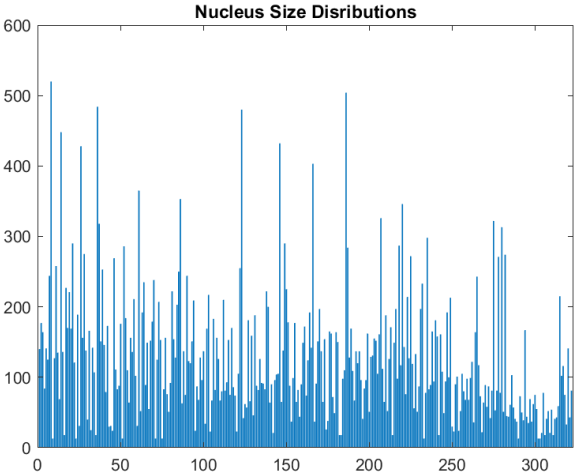
III) Nuclei Analysis

Now that we have the label matrix for our image, we can easily analyse the nuclei and get the properties for each of the nuclei using the function called *regionprops*. This function returns the measurements for the set of properties for each 8-connected object in the image.

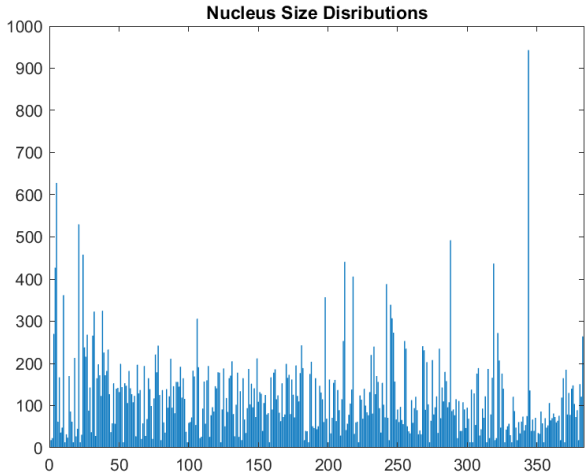
1) Size

To calculate the size of each of the nucleus, the property we want is the area, or the number of white pixel of an object(nucleus). It calculates the total number of white pixels in each label and store it in a form of *struct*. In the following distribution, x-axis represents a particular nuclei and y-axis represents the corresponding size.

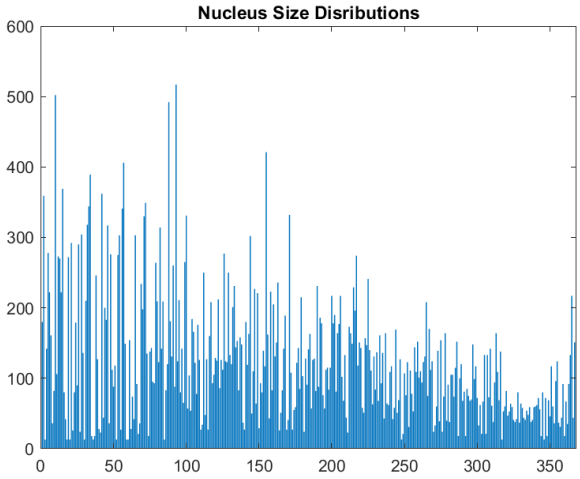
(a) Image 1:



(b) Image 2:



(c) Image 3:



As we can see, the nucleus sizes for 3 images have some deviations and anomalies, those are mainly due to the nuclei merging after converted to binary image. I calculated the maximum, minimum and average size of the nuclei in each image:

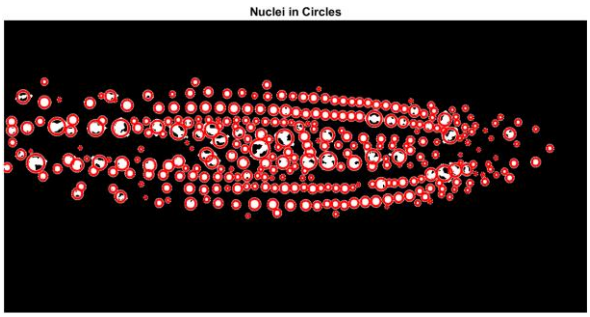
(a) Image 1:	(b) Image 2:	(c) Image 3:
Maximum : 520	Maximum : 943	Maximum : 517
Minimum : 13	Minimum : 13	Minimum : 13
Average : 125.65	Average : 119.16	Average : 122.5

2) Shapes

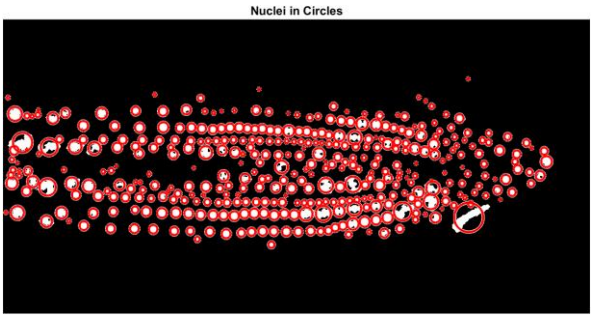
As for shape of the nuclei, that can be seen are mostly circles, but not perfect. With the function *regionprops*, I can find the circularity of each nucleus. The circularity value is computed as (4*Area*pi)/(Perimeter^2). For a perfect circle, the circularity value is 1. So the farther away the circularity of a nucleus is to 1, the less circular it is.

I also used *regionprops* to find the centroid (the centre of mass of object), the major axis length and minor axis length of an object (nucleus). Assuming each nucleus is a circle, the centre of that circle is the centroid, and the diameter can be found by taking the average of major axis length and minor axis length. Here we have the image with nuclei labelled in circles, using the formulae.

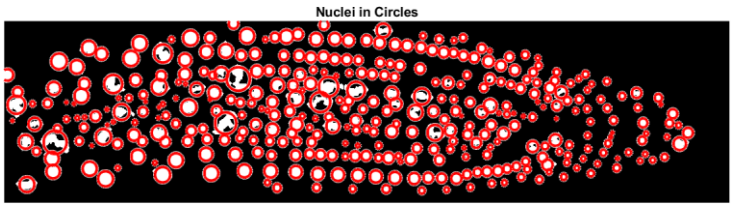
(a) Image 1:



(b) Image 2:



(c) Image 3:

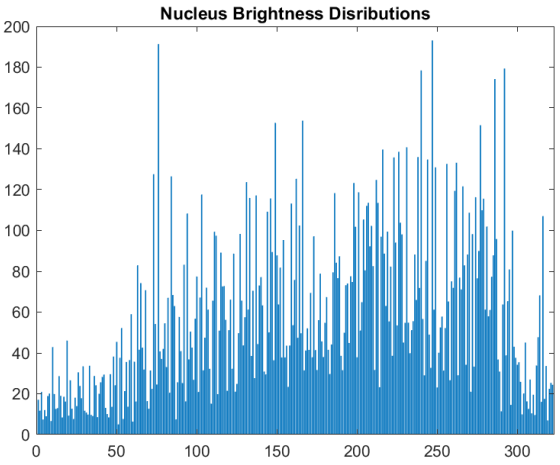


3) Brightness

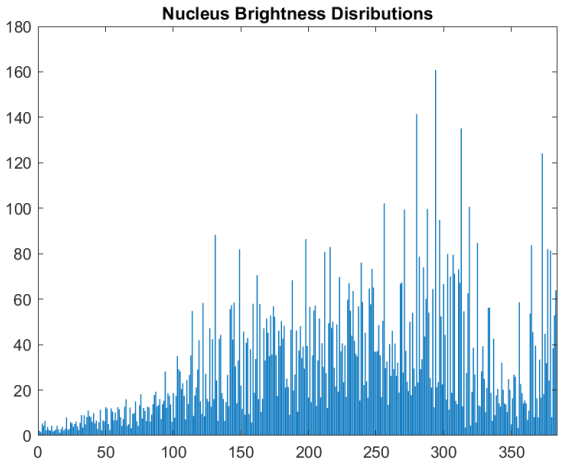
For brightness, we can get the intensity of the pixel of each nucleus in the original gray scale image of the nuclei. In this case, the pixel value is in between 0 and 255, with 0 being the darkest and 255 being the brightest.

I can find the pixel index list for each nucleus using *regionprops*. With the pixel list, I can find the nucleus position in the original image, analyse its brightness by taking the average pixel value (0~255) of all the pixel in a single nucleus and store them in an array. The following distributions shows the average brightness of each detected nucleus.

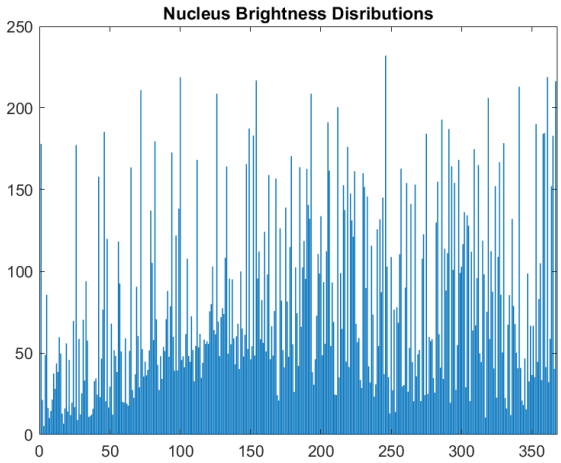
(a) Image 1:



(b) Image 2:



(c) Image 3:



I then also calculated the overall average brightness and the maximum and minimum brightness of the nuclei in each image.

(a) Image 1:	(b) Image 2:	(c) Image 3:
Maximum : 193.06	Maximum : 160.8	Maximum : 232.09
Minimum : 5	Minimum : 1.2	Minimum : 5.23
Average : 57.24	Average : 29.57	Average : 78.43

As we can see that image 3 is having the highest brightness of detected nuclei and image 2 is having the lowest brightness of detected nuclei.

IV) Evaluations

1) Strengths

i. Median filter

Strengths: Doing well in filtering image spatially. It removes a lot of ‘salt’ noise of the image especially after we converted it to binary.

Weaknesses: Some of the ‘salt’ noise in the background is too big. It is not enough to remove just by using median filter, so they will be mistakenly taken as nuclei.

ii. Top and bottom hat filter

Strengths: It is a very good filter in enhancing the image, in this case the nuclei. It helps to enhance the details of the nuclei and make it more visible.

iii. Binary image conversion

Strengths: *imbinarize* has a features in which the threshold can be chosen adaptively. That is, threshold value is based on the mean intensity of the neighbourhood of that pixel. This way, some really dim nuclei can be seen after converting to binary image.

Weaknesses: In the gray scale image, it can be clearly seen that some nuclei behind are dimmer than the one in front , some are even stacked on top of each other. After converting it to binary image, they are taken as one. So clearly that by just counting connected objects in the binary image, we missed out a lot of nuclei that are hidden behind.

Improvements: Needs to be able to separate merged nuclei to individual nuclei after converted to binary image.

iv. Watershed algorithm

Strengths: It is a very good algorithm to separate nuclei. Some nuclei that are merged together can be clearly seen separated after applying this algorithm.

Weaknesses: Some nuclei are still merged after applying this algorithm. Using distance transform watershed is not good enough to separate the nuclei as they might be too close to each other.

v. Region props

Strengths: Region props is a very good function to analyse labelled binary image. It allows us to find the area, the shape (the centroid and axis lengths), and the pixel index of an object.

Weaknesses: Every connected objects are taken into account, which means some small noise will be considered as nuclei and some of the nuclei are also merged together due to the weakness in the watershed algorithm. This will greatly affect the analysis we do on the nuclei.

Improvements: Ignore small objects and anomalies.

v. Circularity

Weaknesses: An object's circularity should not be larger than 1, but we are getting some objects having circularity larger than 1, this is due to some of the objects are too small and the way *regionprops* find the circularity is inconsistent.

Improvements: We can ignore the objects that are too small (3 by 3) as those are mainly noises, or not significant to be counted as nuclei. This way, the circularity will not be larger than 1.

IV) VIDEO DEMONSTRATION

https://youtu.be/MQabLJkp8c0

* English subtitle is available in case for un-clear speech