COMP 2032 INTRODUCTION TO IMAGE PROCESSING

Academic Report
for
Extracting & Analyzing Cell Nuclei Assignment

Name: GOH XIN YEE

ID: 20093715

Username: sfyxg1

Lecturer: DR AMR AHMED

A Matlab function is created that takes in a root cell image as the input and outputs a result of type *struct* that contains all necessary information (sub-images, related variables) of the image.

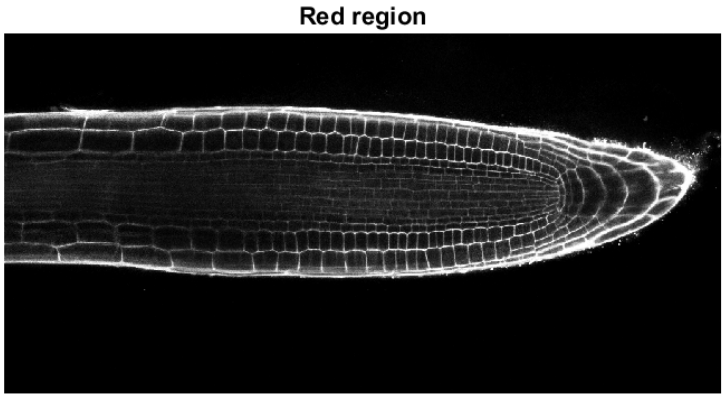
In the function,

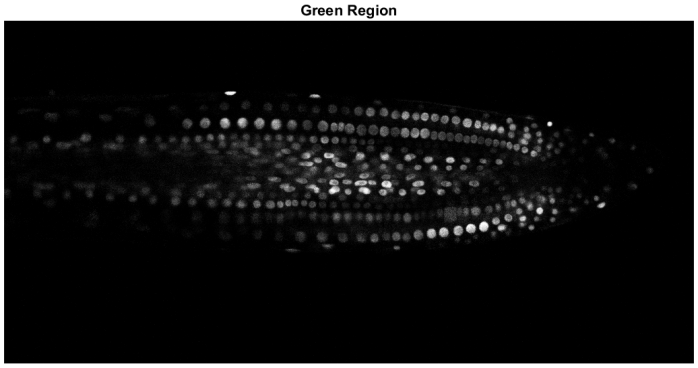
Pre-step: Make a copy of the original input image before processing.
This is to retain the original input image.

Task 1: Detect Nuclei Regions and Find Total Count of Nuclei

Step 1: Remove the RED and BLUE components from the image.

As shown, the root cell images are 3D RGB images. The nuclei are dominant in green while the cell walls are dominant in red. Below shows red and green masks extracted from the original image.

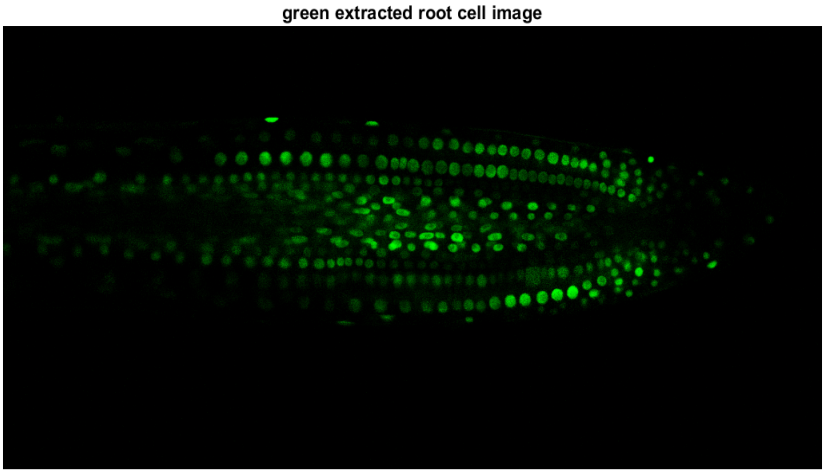




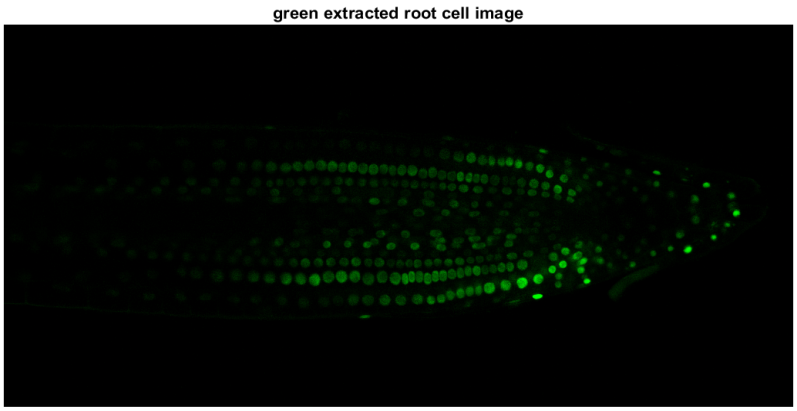
So, for easier detection of nuclei, we make all **red** ***(: , : , 1)*** and **blue** ***(: , : , 3)*** components of the image pixels black (0) and let only the green components stay.

Result:

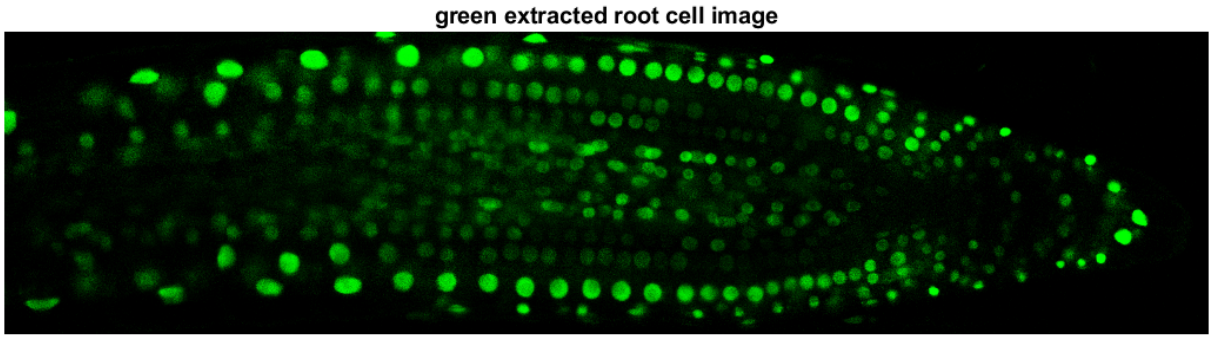
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp



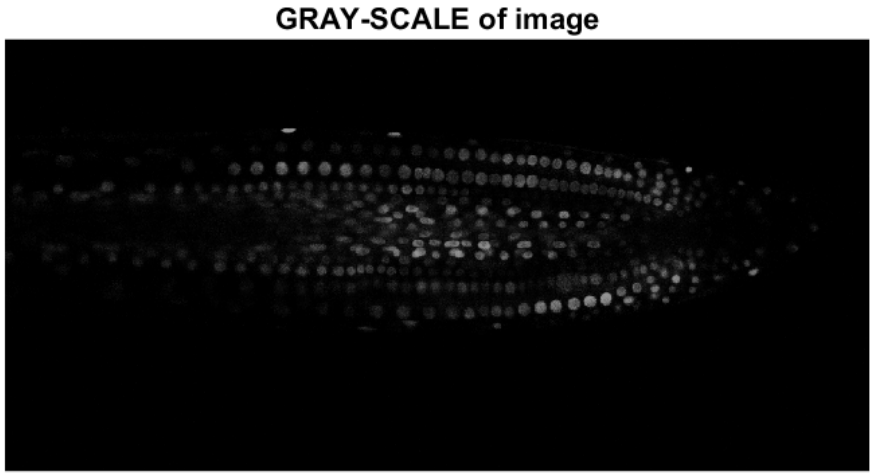
Evaluation: The red cell walls are eliminated from sight, left only the green nuclei. This allows the processing later to get less affected by the cell walls.

Step 2: Convert the green extracted image from RGB to GRAY-SCALE

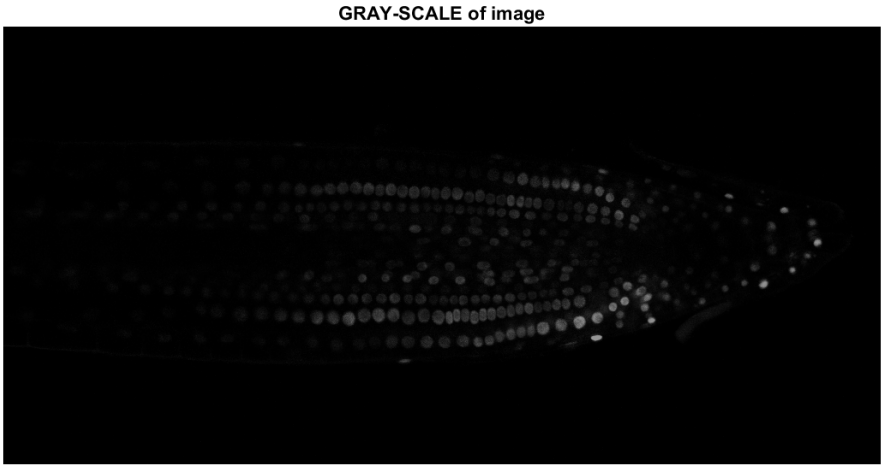
Using ***rgb2gray()***. 2D grey-scale dimension makes analysing and processing image data easier.

Result:
The green nuclei appear in different grey intensities and the surrounding will be 0 intensity.

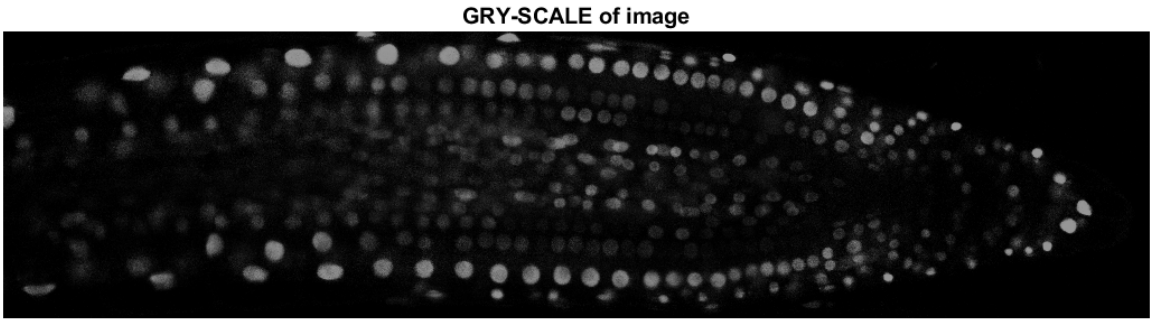
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp

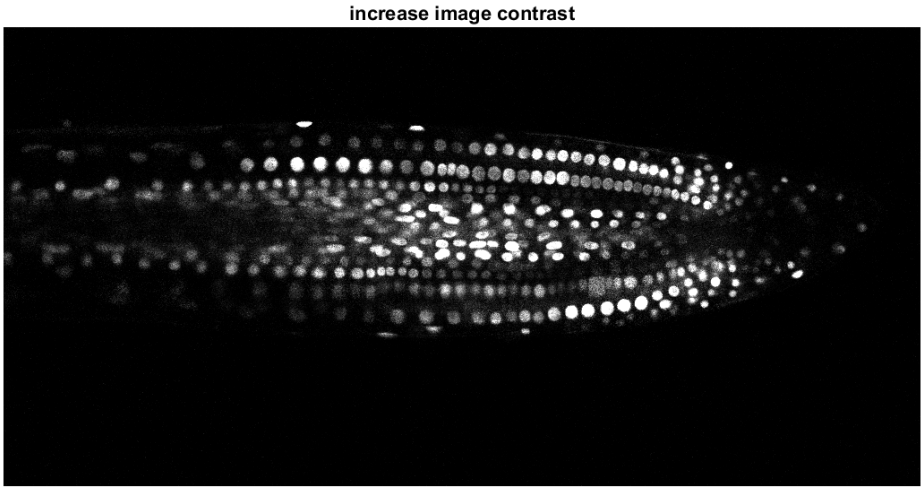


Step 3: Adjust to increase the grey-scale image contrast.

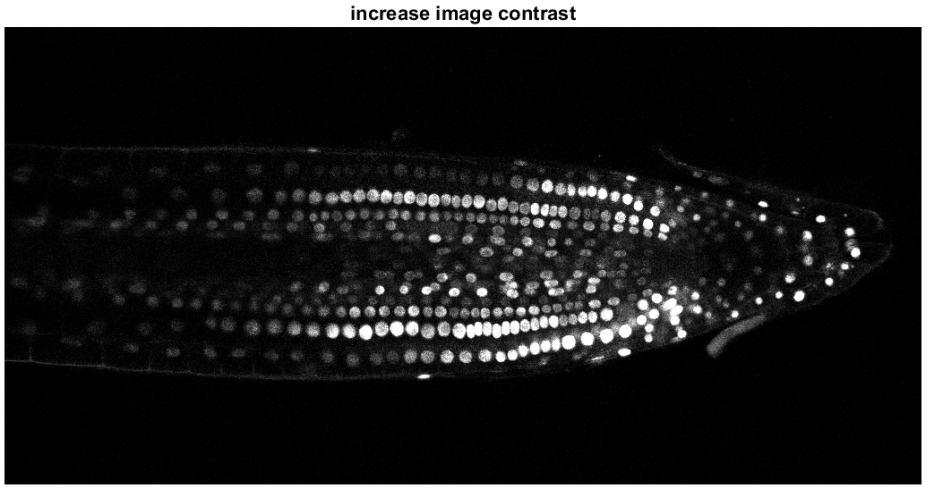
It is noticed that the image is dark, the nuclei are having low intensities close to the dark surrounding and aren't obvious. Thus, ***imadjust()*** approach is used to make the nuclei stand out even more from the dark surrounding for easier nuclei identification and extraction. We can also identify noises.

Result:

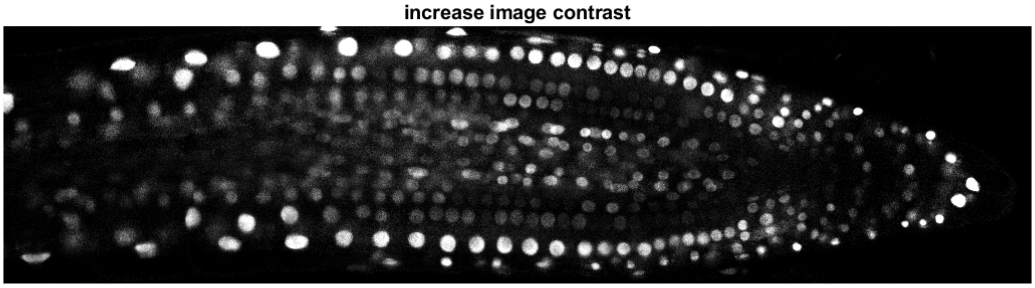
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp

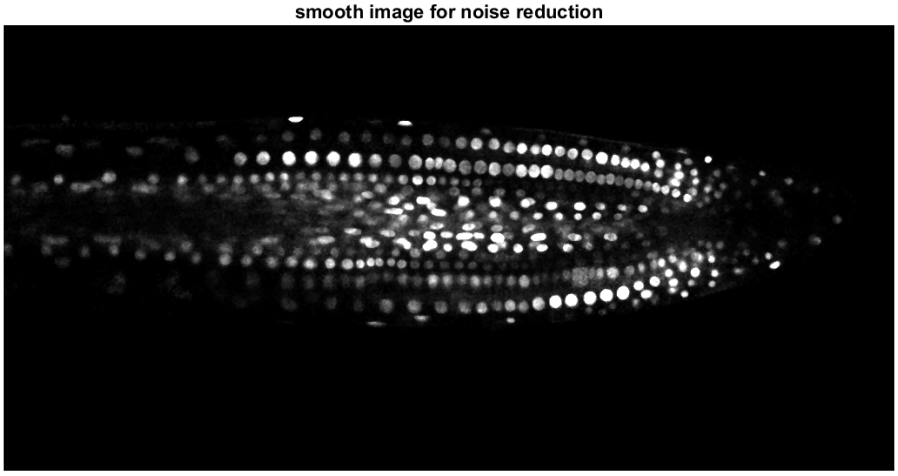


Step 4: Remove noise from the image through smoothing filter:

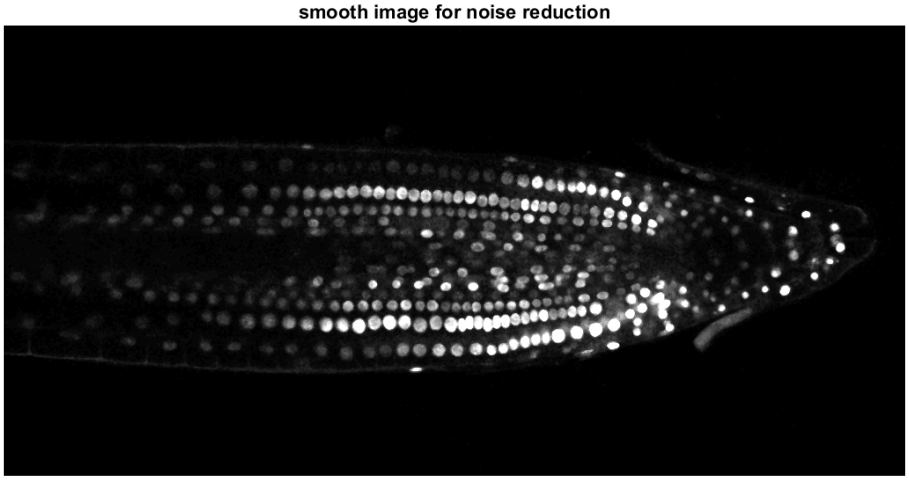
This step is important because the images in Step3 look noisy and **noise can cause false interpretation in nuclei detection.**
The smoothing filter used is the **median filter.** The pro of it is that it is good at retaining the image contrast and the original pixel values. It preserves the nuclei edges while reducing noise. Thus, the nuclei will not look dispersed after noise reduction.

Result:

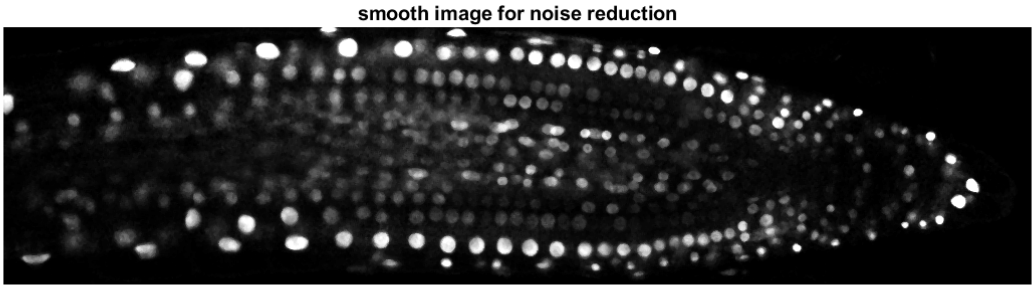
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp



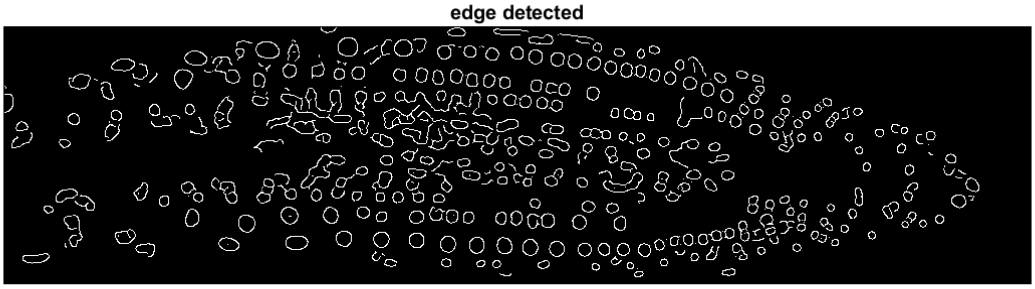
Step 5a: Edge detection on the smoothened image

We detect all possible edges of the entire image to have an overview of all possible nuclei. **Canny edge detection** is chosen because it can detect both strong and weak edges of the bright and less bright nuclei by having 2 thresholds. It is also good at not getting fooled by any remaining noise as the "nuclei edges".

Result:
StackNinja1.bmp

StackNinja2.bmp

StackNinja3.bmp

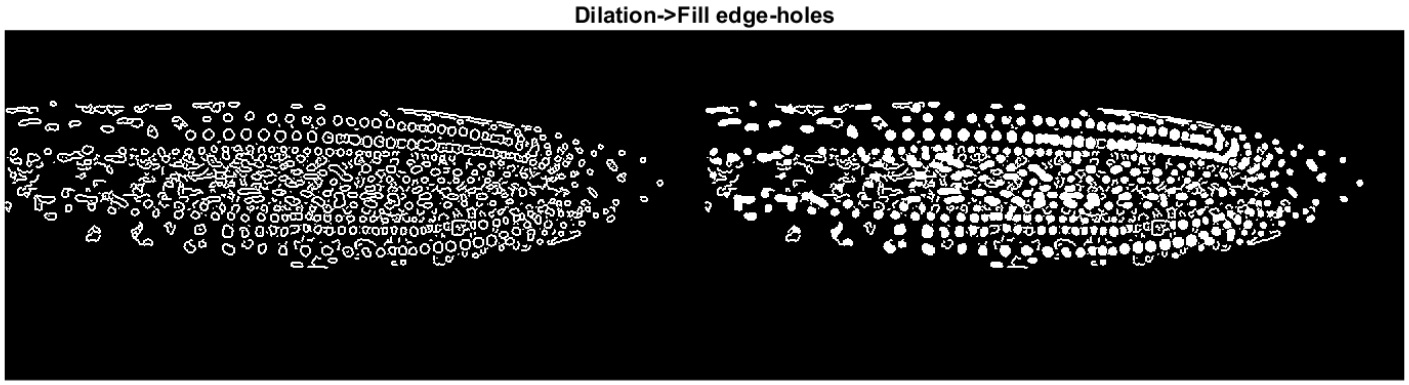


Evaluation: The problem with this is that the detected edges are not strong enough and some appear broken.

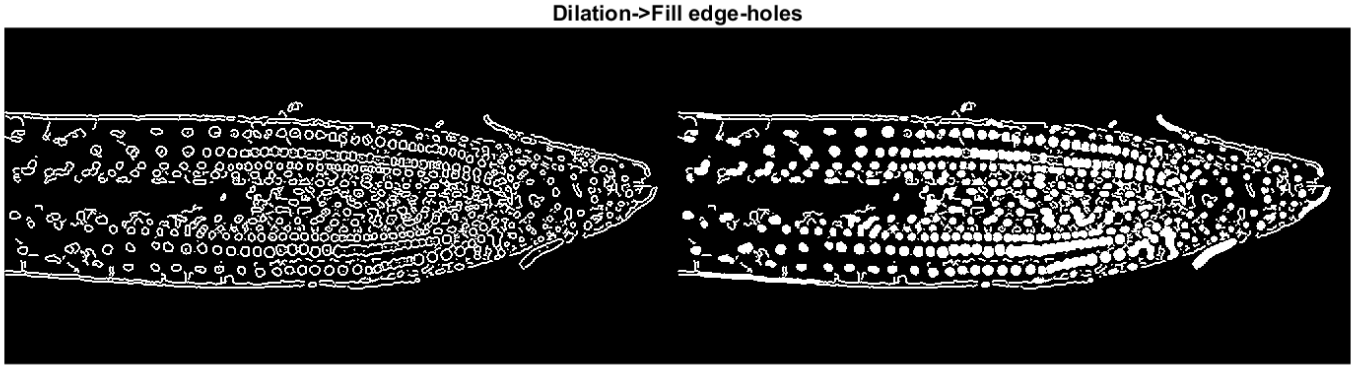
Step 5b: Process detected edges (dilate and fill)

As the detected edges are thin and broken, **morphological dilation** is applied on those edges to fill the gap and emphasize the edges. Then, using the region filling method, ***imfill()*** the inside of the edges aka the "**holes**" is **filled** to restore the look of the nuclei (simulation).

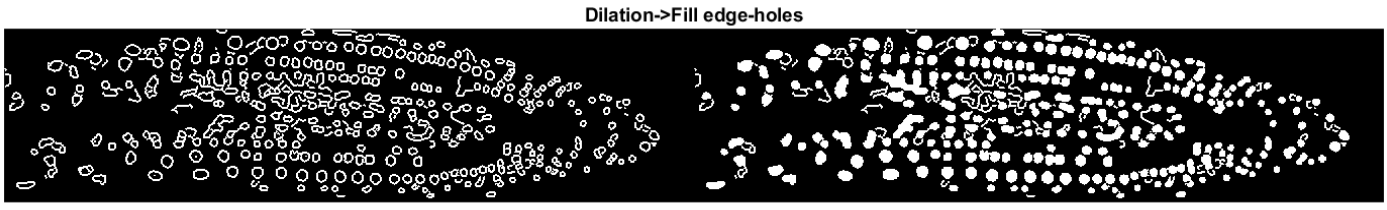
Result:
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp



Evaluation: Most of the edges got strengthened, connected after dilation and filled to simulate nuclei. The problem with this dilation approach is that some edges are connected wrongly where edges of different nuclei got connected as 1. There are also edges which do not belong to nuclei but are filled because they are detected and connected (as seen in **StackNinja2.bmp** the cell walls). There are also

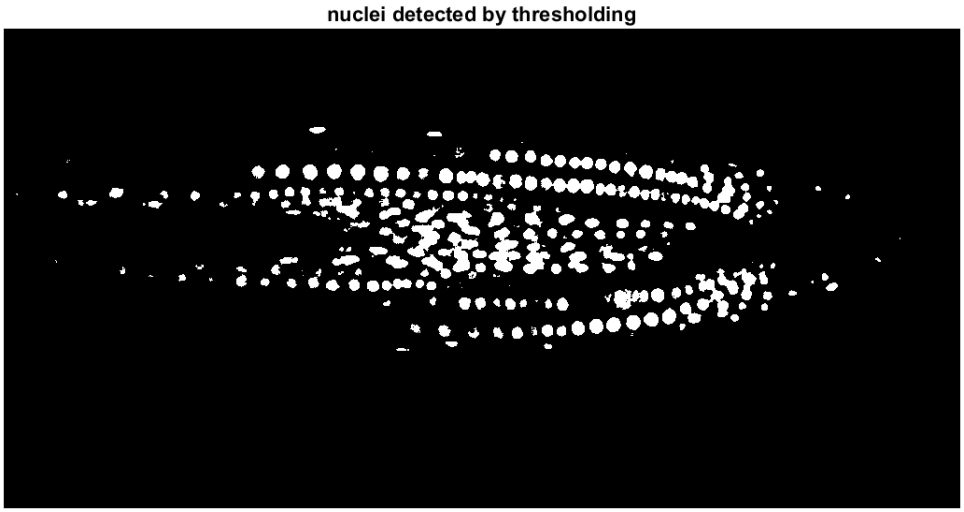
edges that seem to be nuclei edges but are not filled to simulate nuclei because they still have tiny gaps even after dilation. Thus, there is a chance that we are missing out on some nuclei.

> **Step 6:** Leave the edge image in Step5 aside, **binarize the smoothened image (in Step4) using a global threshold to obtain thresholded nuclei.**

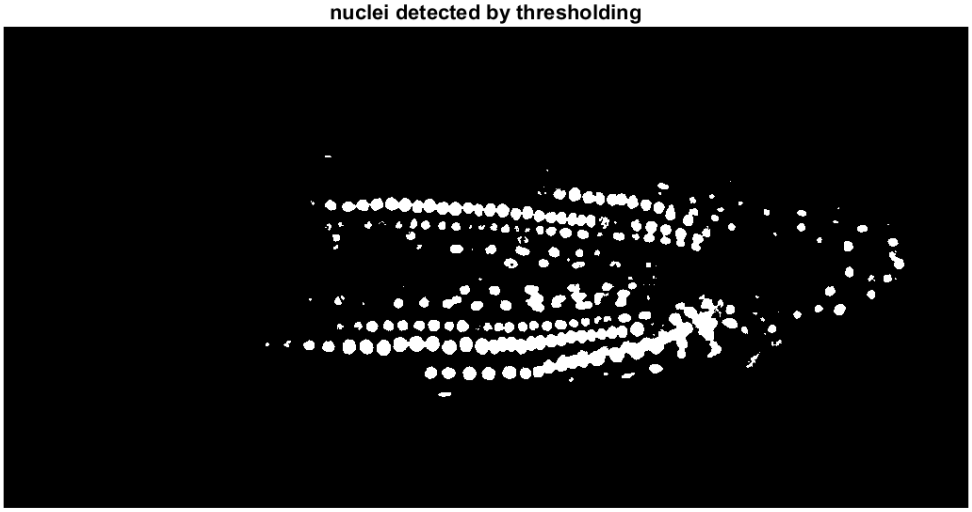
Thresholding is important in image segmentation to separate the nuclei from the surrounding. We obtain a global threshold using ***graythresh(),*** then binarize the image to get thresholded nuclei.

Result: Nuclei are shown in white, surrounding shown in black.

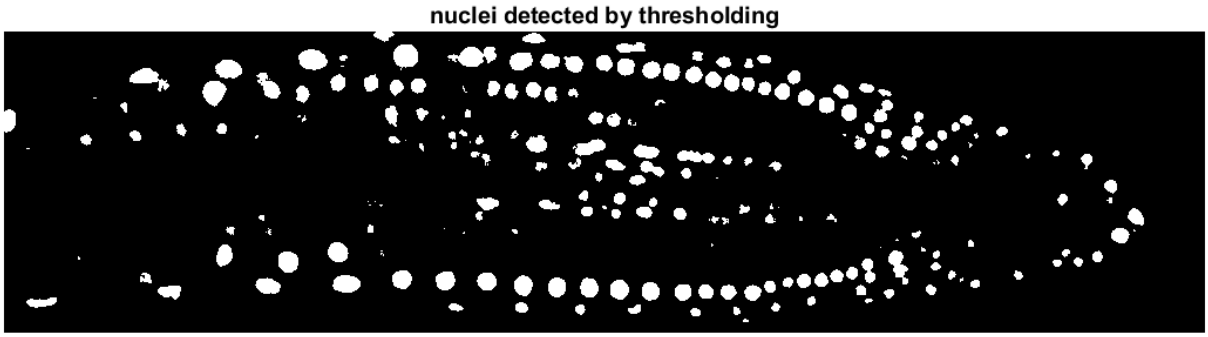
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp



Evaluation: The pro of thresholding is that we can easily extract the nuclei. However, it can't guarantee all nuclei are extracted. Only those nuclei with a higher intensity (more than the threshold) will be brought out while the lower intensity/darker nuclei will not be detected and are ignored. It will also cause overlapping thresholded nuclei to be combined as 1 when binarized, thus looking connected, as shown in all three images especially **StackNinja2.bmp.**

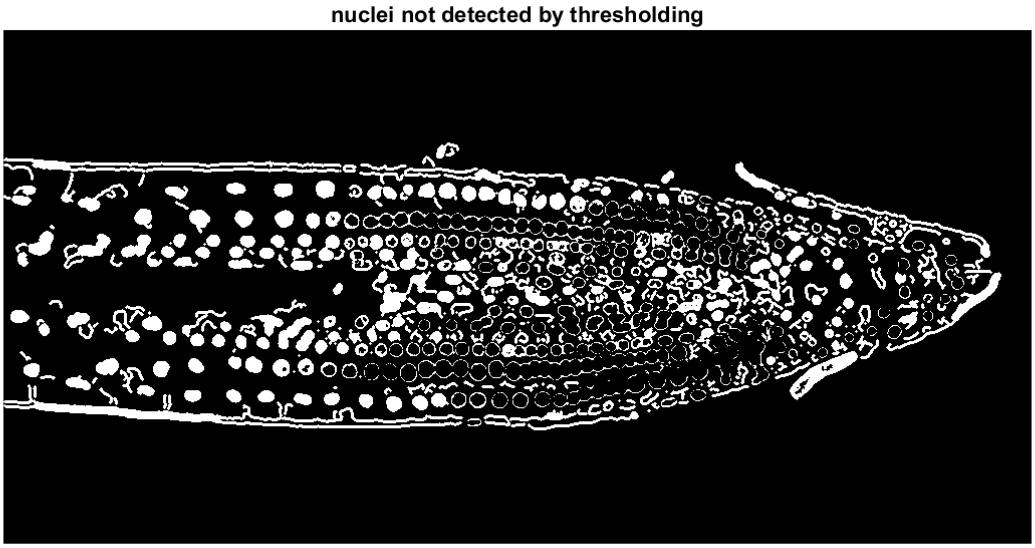
Step 7: Find the remaining nuclei undetected by thresholding

We **subtract the nuclei detected by thresholding** in ***Step6*** from **the nuclei simulated from edge detection** in ***Step5***. Since the edge detection in ***Step5*** is detecting all possible edges including the edges of those thresholded nuclei, with subtraction we hope to find the **remaining nuclei** that are left out by thresholding.

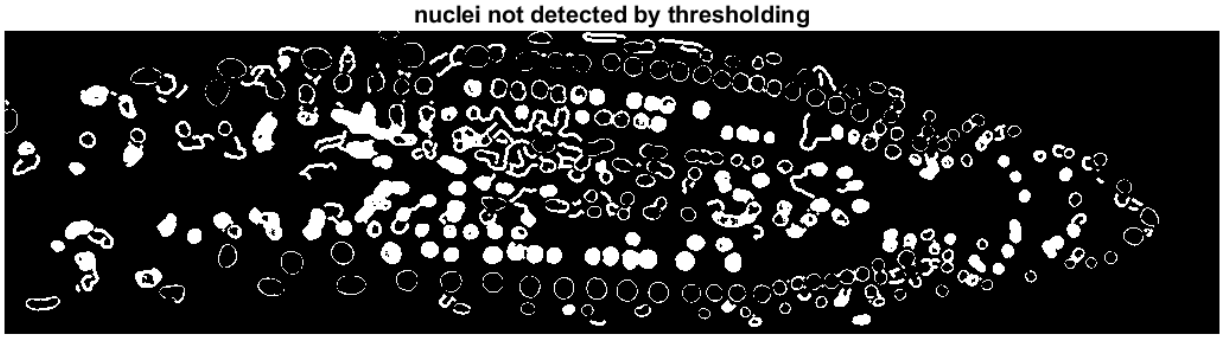
Result:

StackNinja1.bmp

StackNinja2.bmp



StackNinja3.bmp



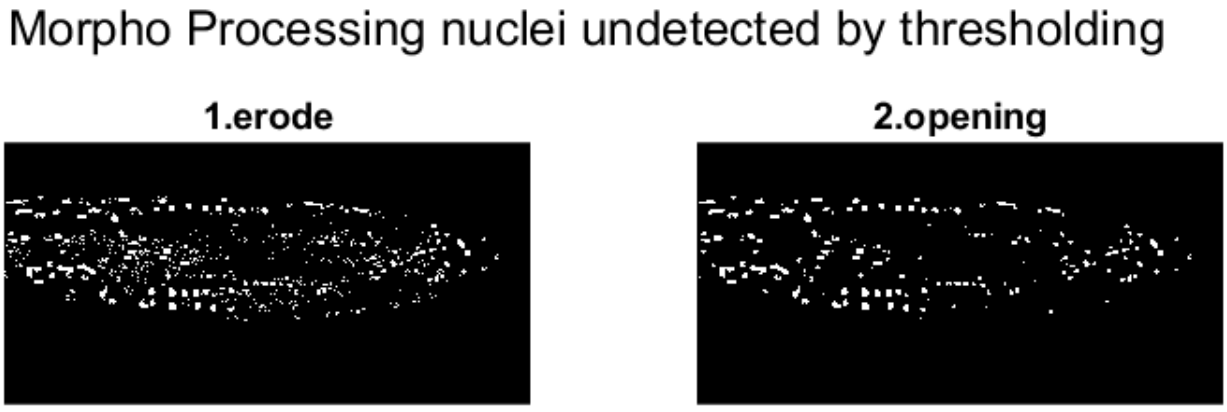
Evaluation: The pro of this is that we can extract anything that exists but is not detected by thresholding. The cons of this is that it is **merely a subtraction**. It cannot identify noises / unrelated lines and ignore them. So, as noticed in the images of the remaining nuclei, there are a lot of unrelated white dots/lines and some nuclei are connected.

Step 8a: Process the remaining nuclei

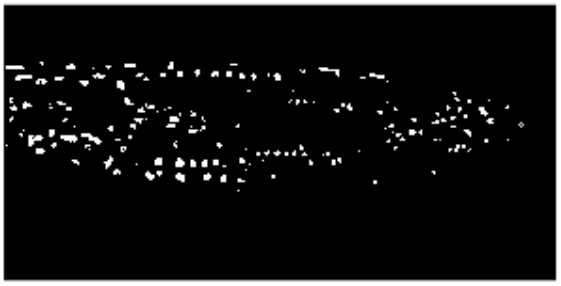
So, perform **morphological erosion and opening** on the remaining nuclei. A combination use of erosion and dilation can remove tiny white lumps/salt noise and break nuclei connections while refining the nuclei.

Result:

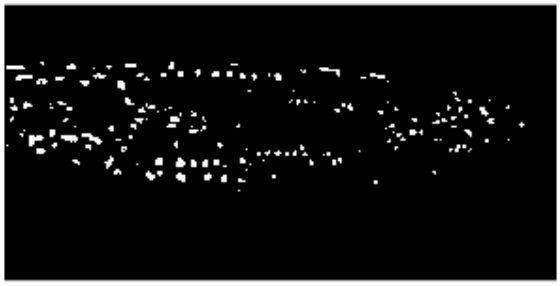
StackNinja1.bmp



3.opening



4.fill holes

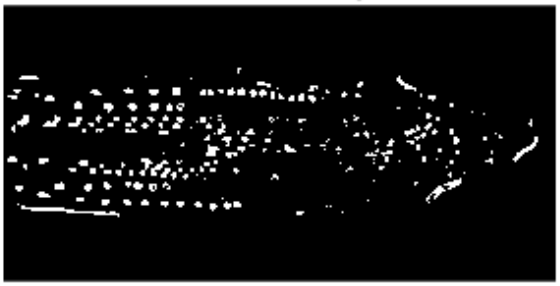


StackNinja2.bmp

Morpho Processing nuclei undetected by thresholding

1.erode

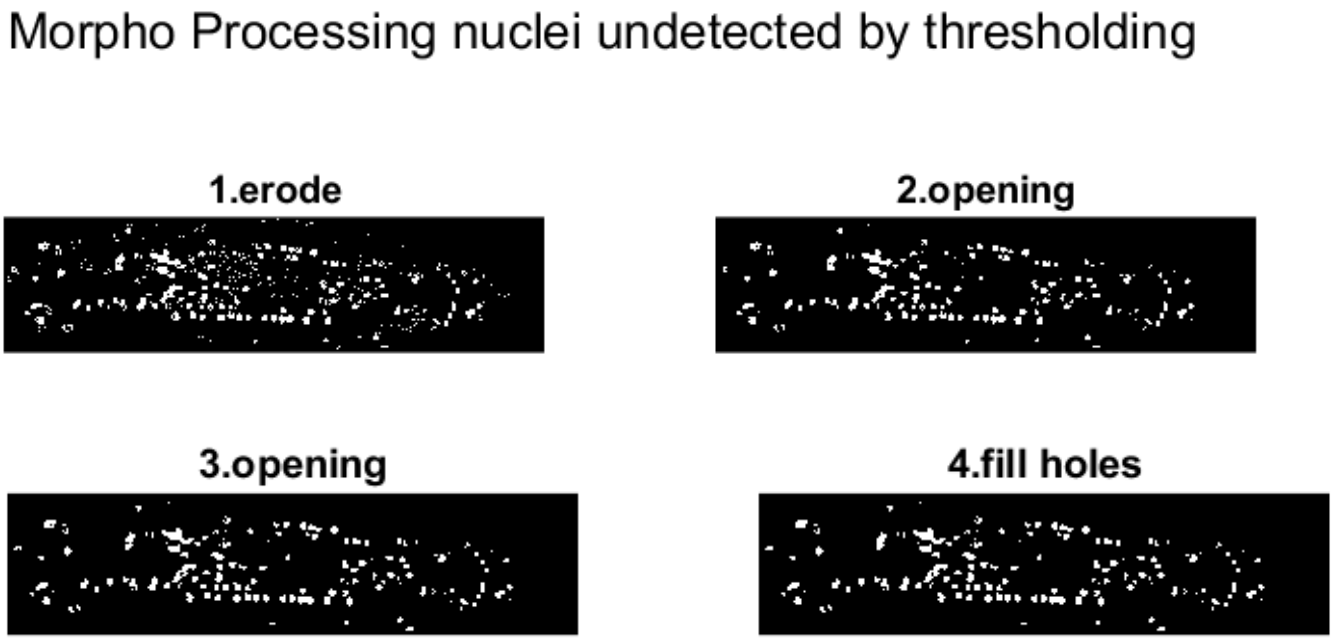
2.opening



3.opening

4.fill holes

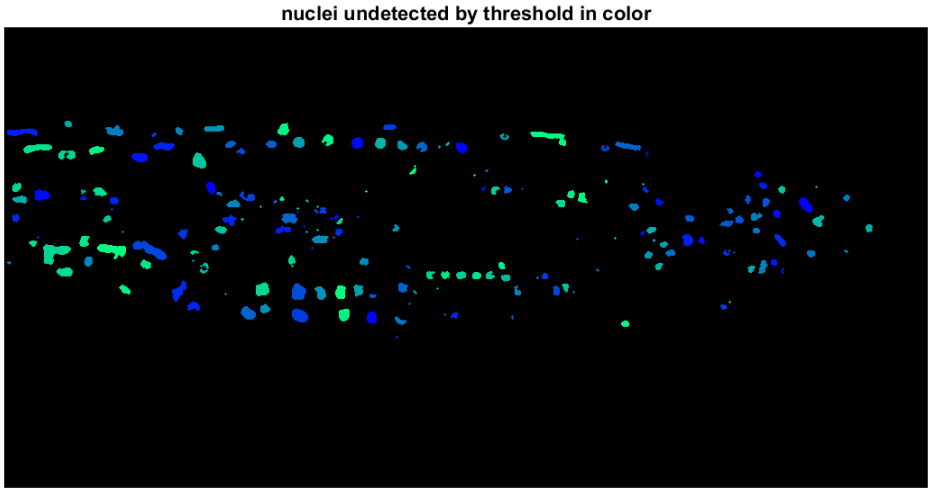
StackNinja3.bmp



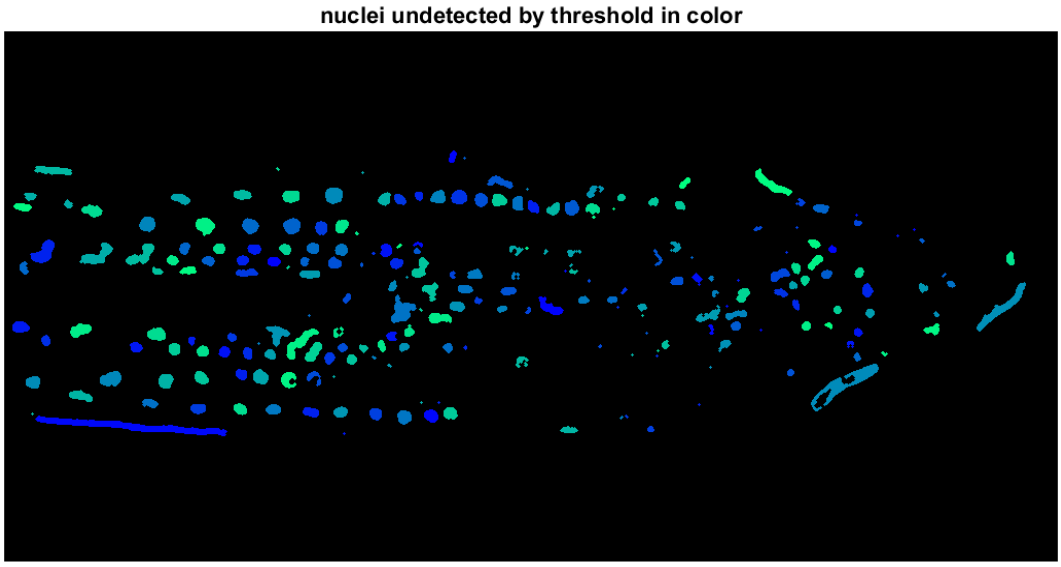
Evaluation: Morphological approach allows nuclei appear more defined, eliminating unrelated white lumps. But, not all connected nuclei are separated, especially the large ones. Also, we can't perform this operation too much as it might end up eliminating small nuclei while having successfully separated large connected nuclei.

Step 8b: color label remaining nuclei

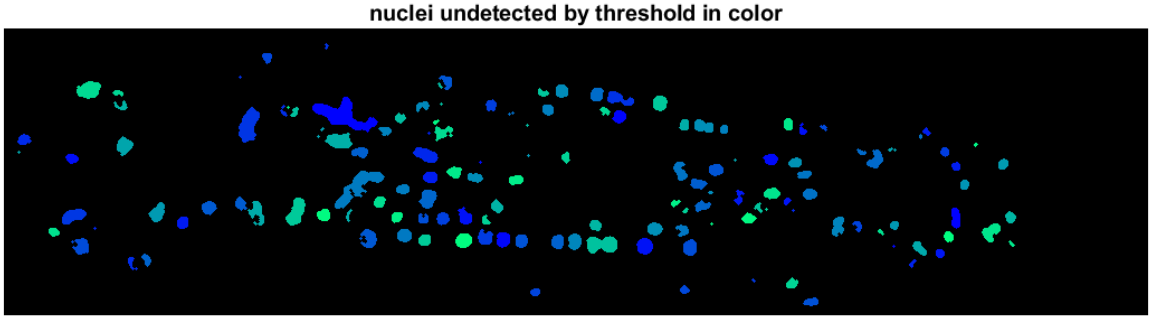
Result:
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp

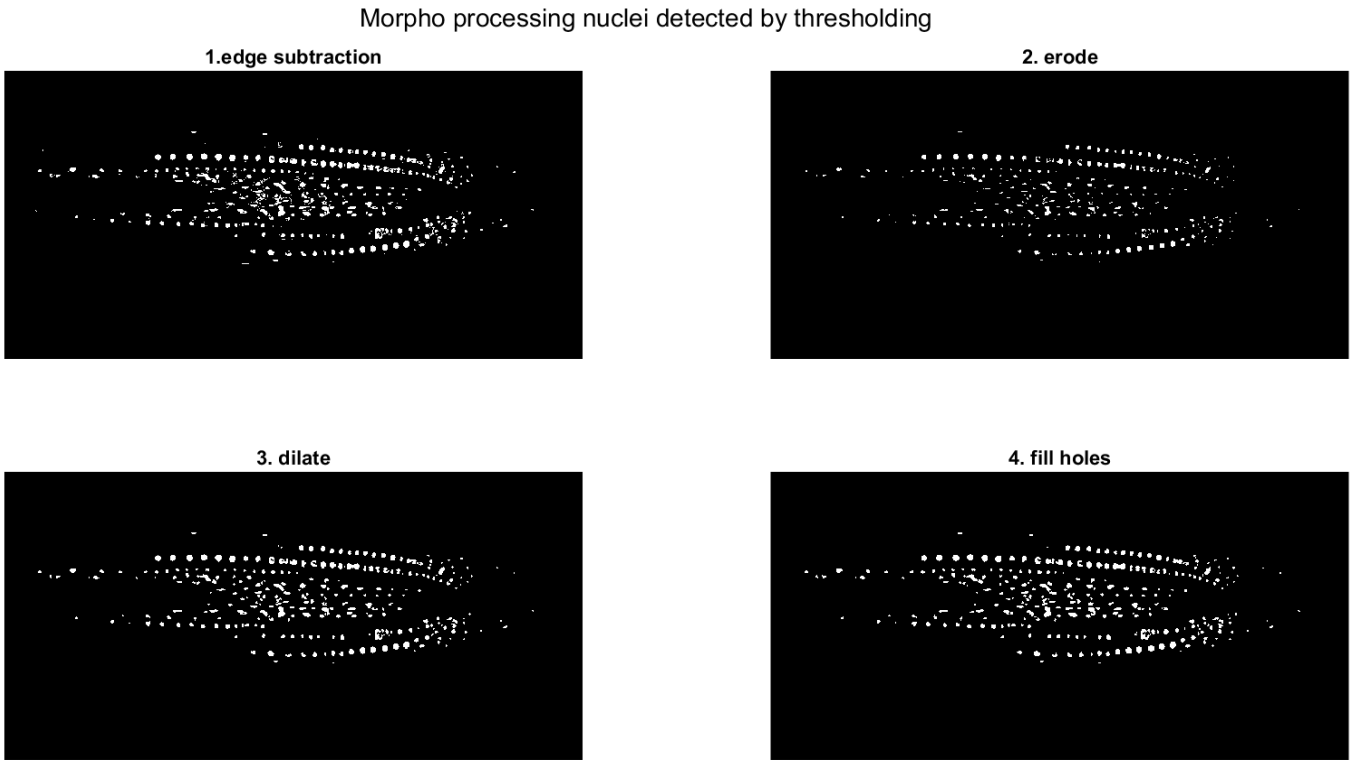


Step 9a: Process the nuclei detected by thresholding

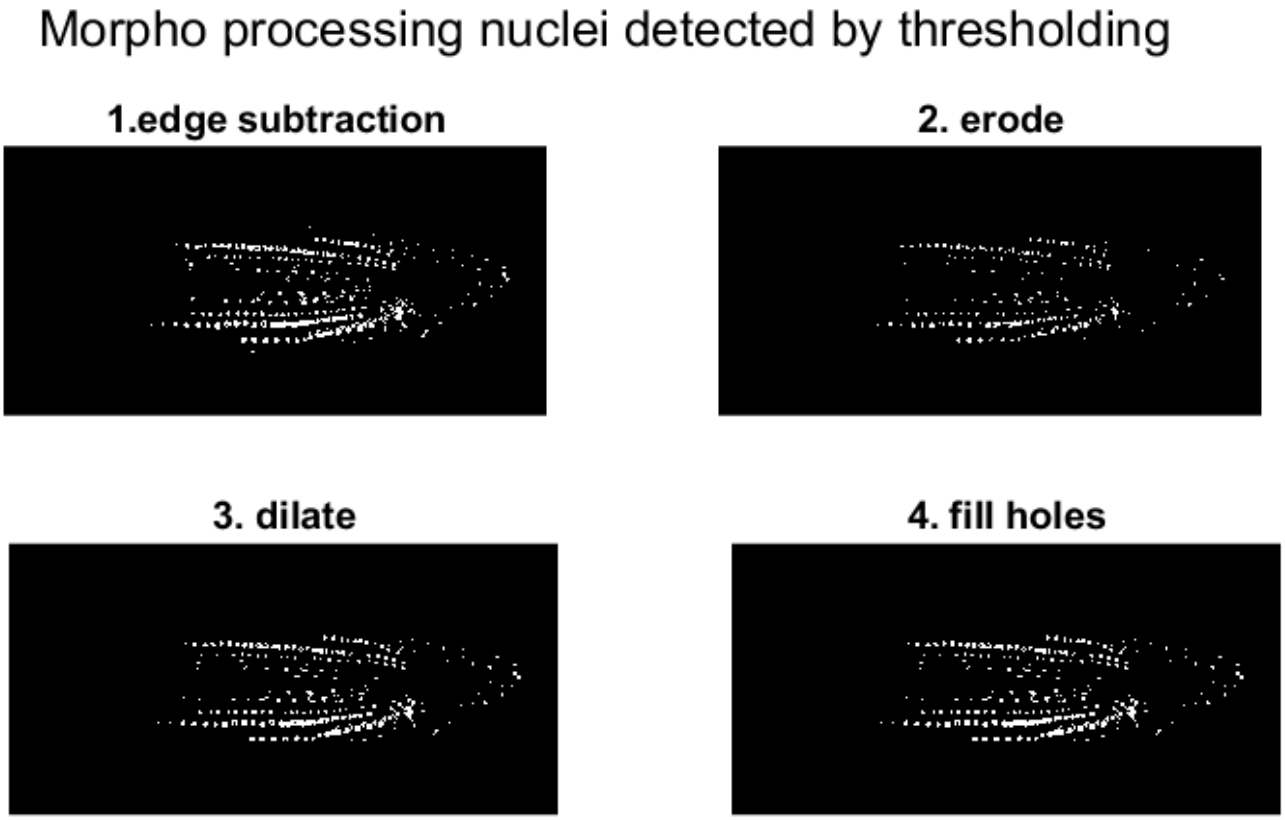
As mentioned in ***Step6***, some thresholded nuclei are connected. Hence, we will first **separate those nuclei by subtracting their edges from them**. In a way, the nuclei shrink.
Then, similar as ***Step8a***, perform **morphological erosion and opening** on thresholded nuclei to further break nuclei thin connection and remove white noise.

Result:
The pros and cons of this are the same as mentioned in ***Step8a***.

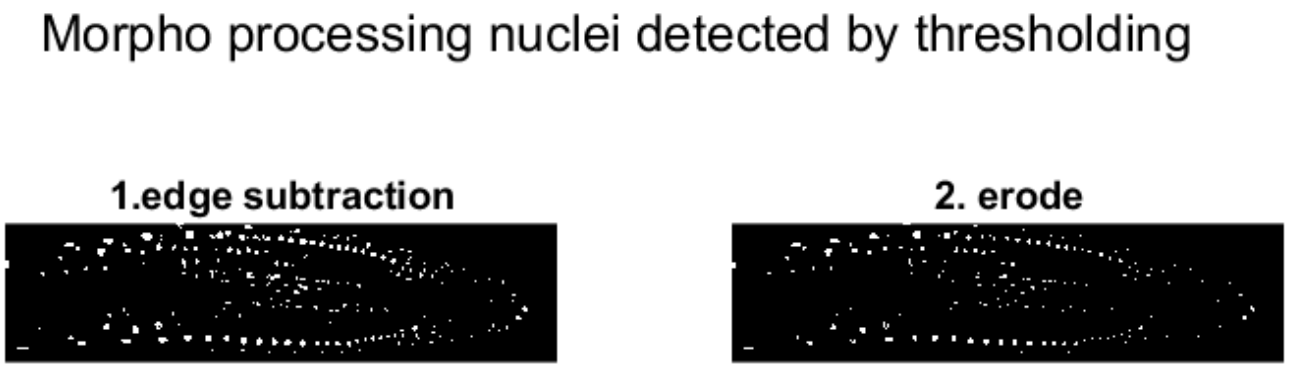
StackNinja1.bmp



StackNinja2.bmp

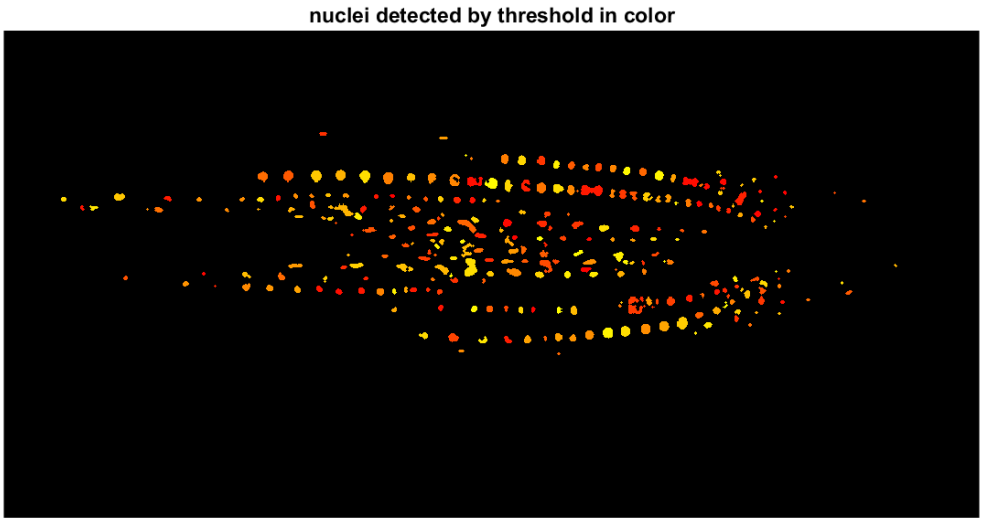


StackNinja3.bmp

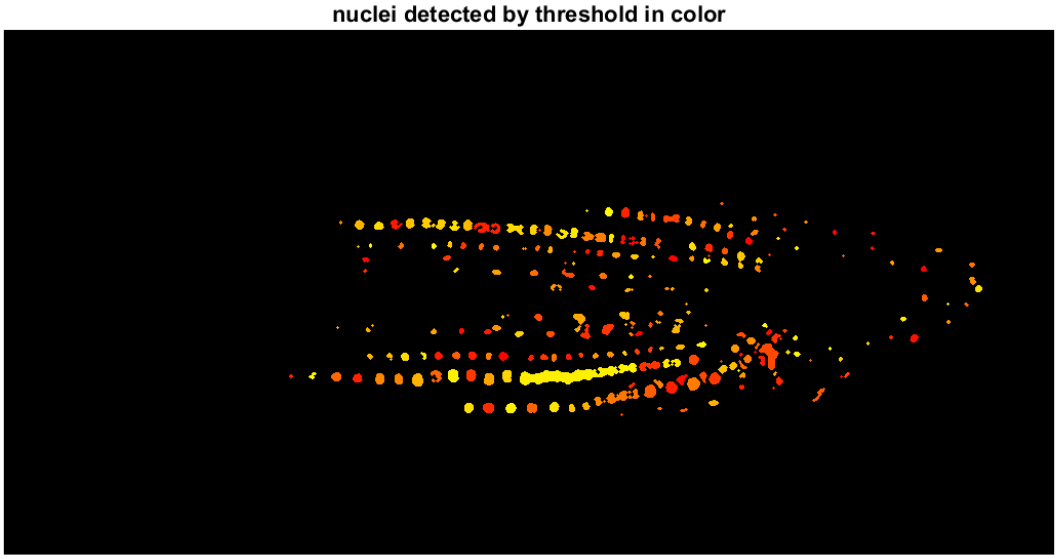


Step 9b: Color label thresholded nuclei

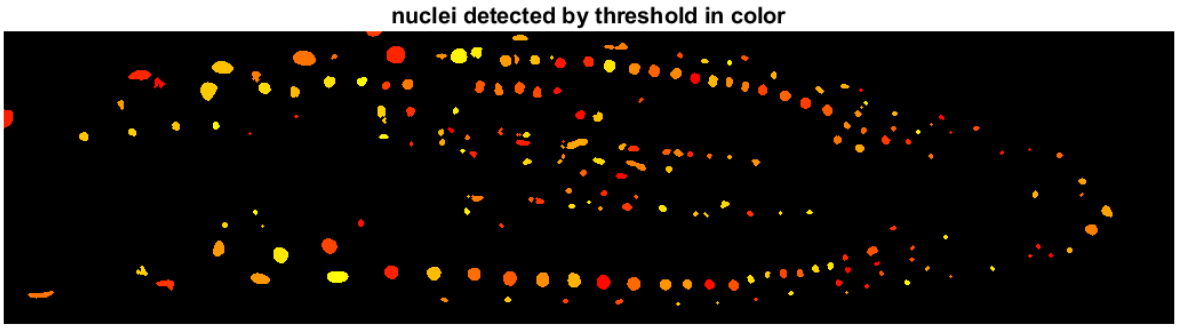
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp



Step 10: Combine all nuclei together as 1 image to restore the view of all detected nuclei.

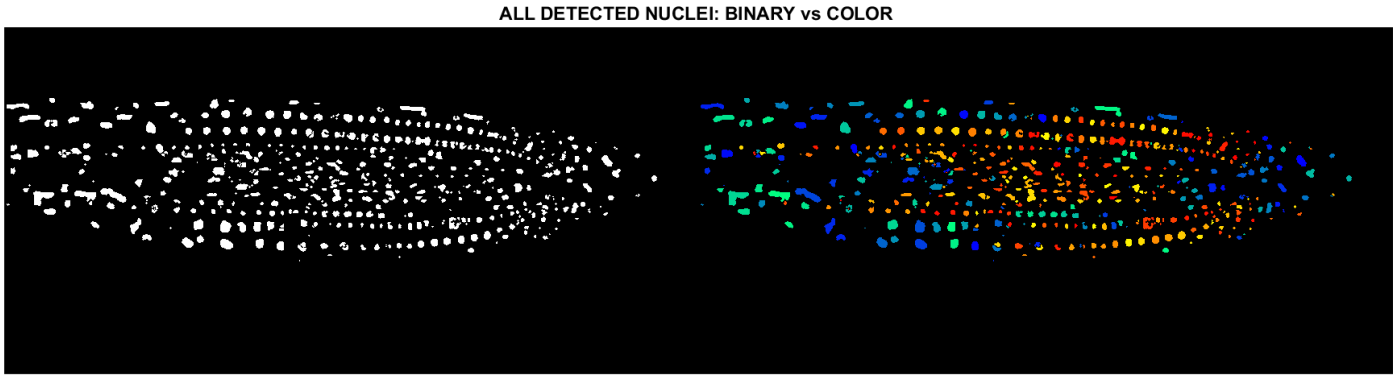
Show in each binary, colour form. In colour form, remaining nuclei will be in cool tones, nuclei detected by thresholding will be in warm tones for distinguishing..

From there, find all nuclei regions detected via finding **the connected components** with connectivity 8 and find **total count of nuclei using *NumObjects***

Result:

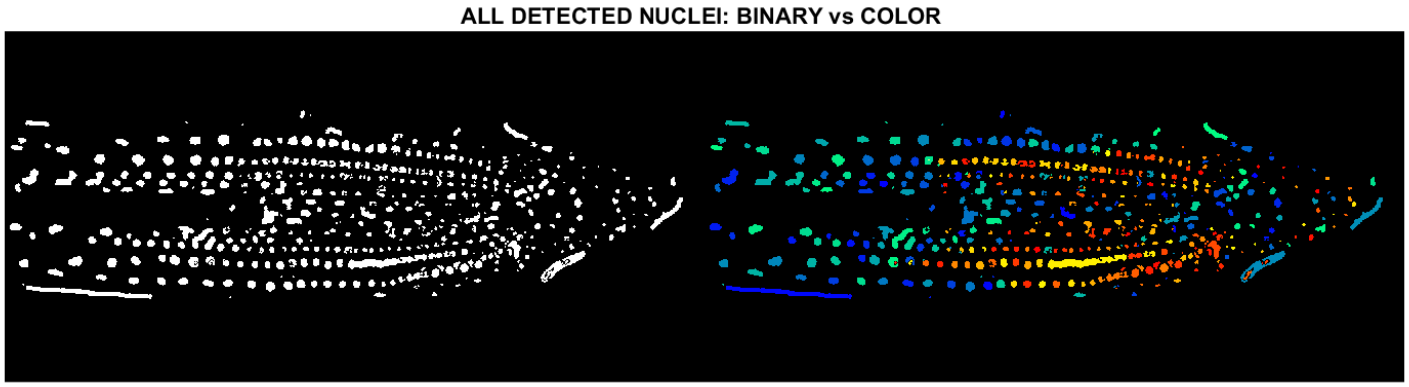
StackNinja1.bmp

sum_nuclei 445



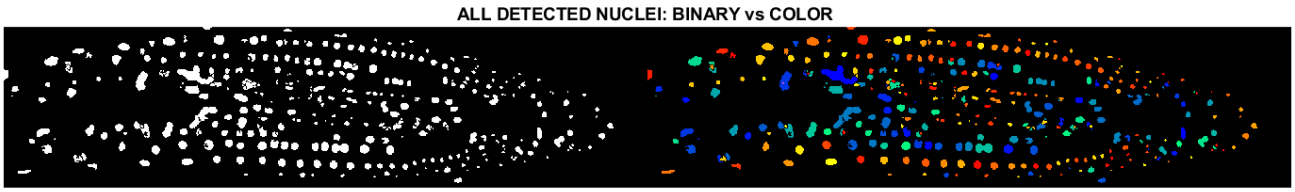
StackNinja2.bmp

sum_nuclei 429



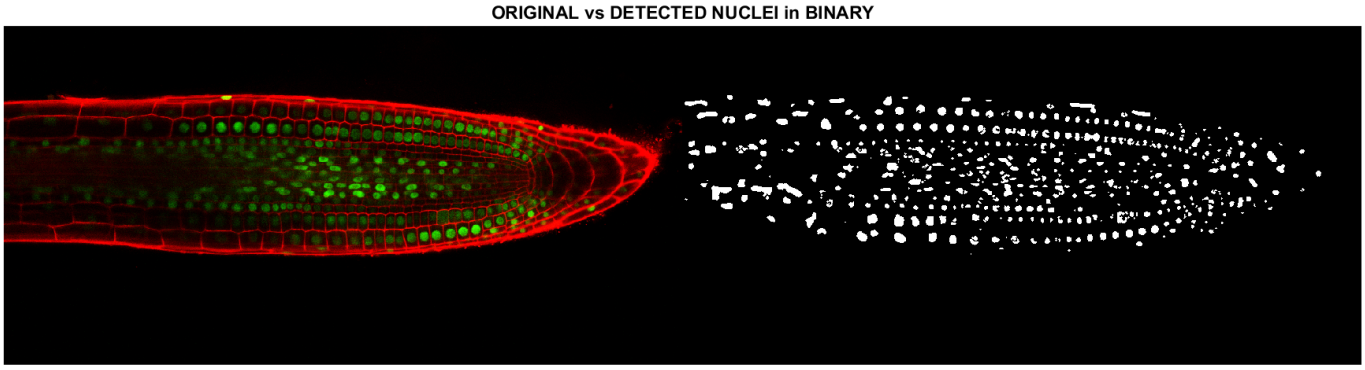
StackNinja3.bmp

sum_nuclei 315

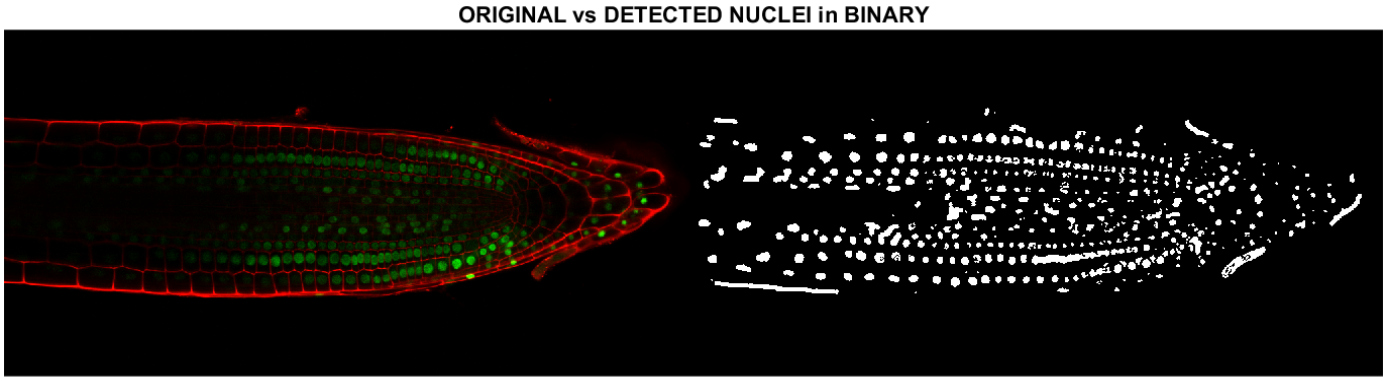


Evaluation of the final result

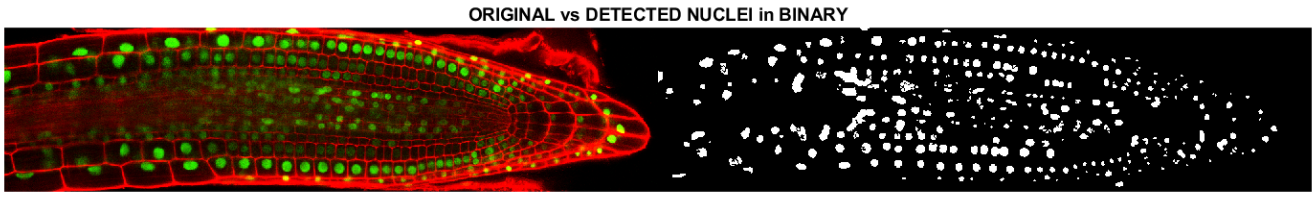
StackNinja1.bmp comparison



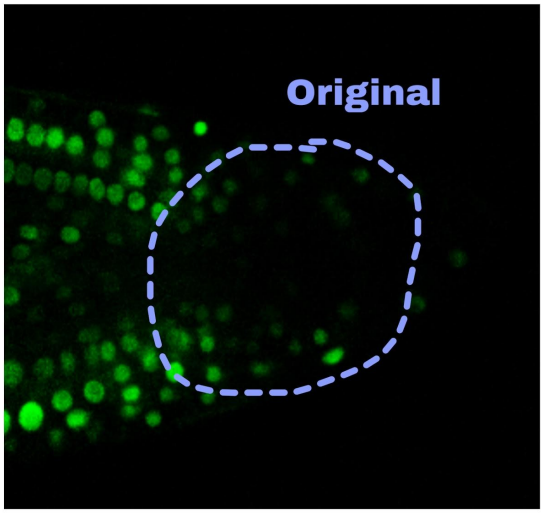
StackNinja2.bmp comparison

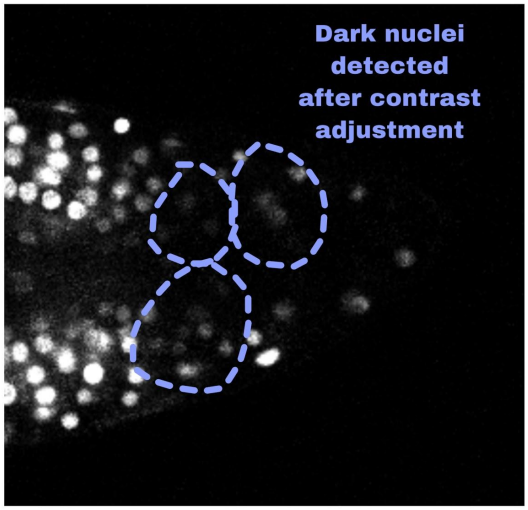


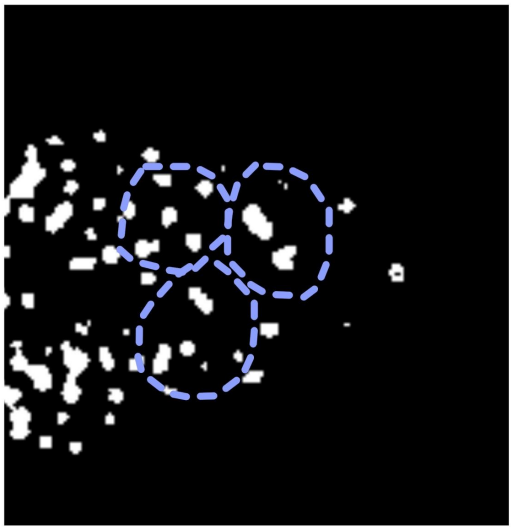
StackNinja3.bmp comparison

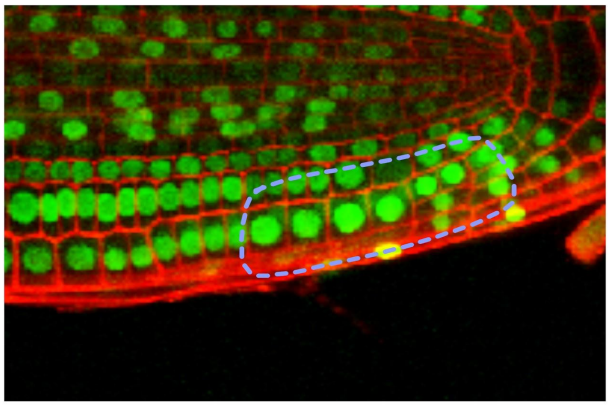


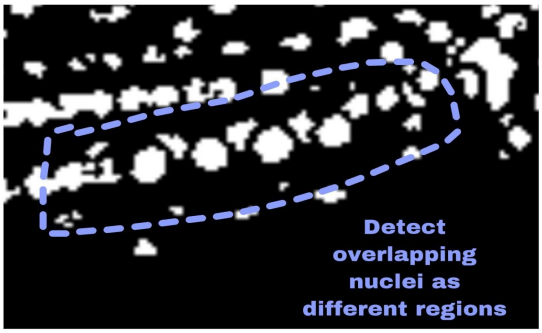
Through the approach, we have successfully **detected bright and dark nuclei.** We have also **managed to let the overlapping nuclei to be detected as different regions instead of 1 when binarized.** The shape of the root cell is retained without **false nuclei detected outside the root cell.**

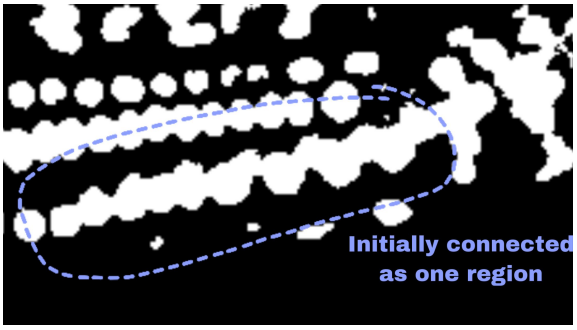




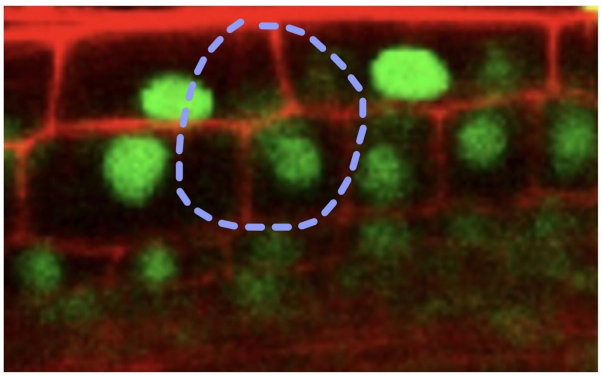


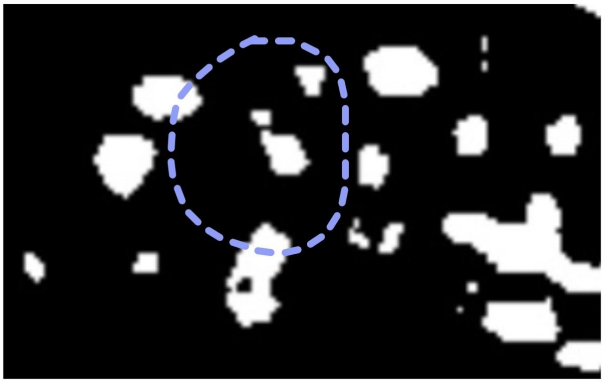






However, there are still nuclei noticed connected to each other. The final output **didn't successfully separate every nucleus.** There are also **non-nuclei being detected**, as seen in ***StackNinja2.bmp***. There are also cases where **a nucleus is being detected as multiple nuclei** due to the nucleus intersecting with the cell walls and are "splitted". Some nuclei also appear slightly **fragmented, not round.**





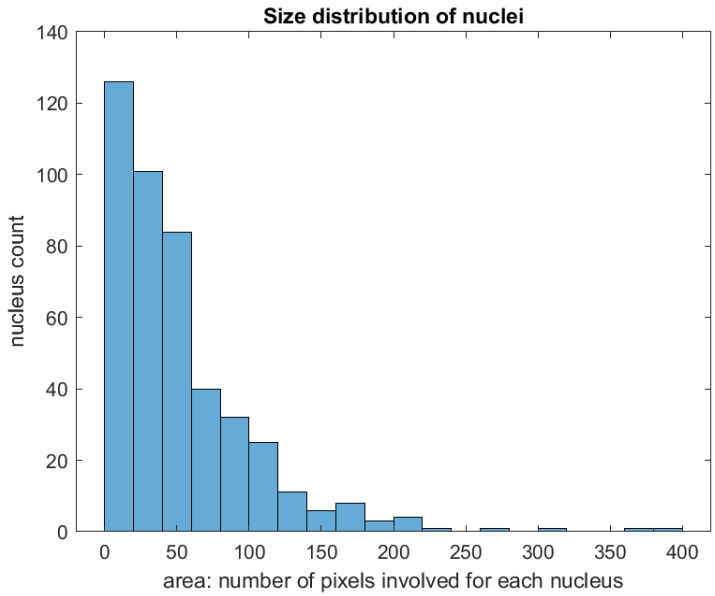
If further working is possible, I would like to research more on **a better, more flexible threshold that extracts more nuclei at once.** I'd also like to research ways that detects and **classifies tiny or extremely close regions as 1 region** to address weakness of results mentioned above. I'd probably **do more layers of extraction** for the image then process each layer differently until the nuclei at each layer look defined. Then, I'd combine them into 1. The processing for each layer needs to be taken care of as it might result in connected nuclei when all layers are combined.

Task 2: Sizes of Detected Nuclei and Distribution

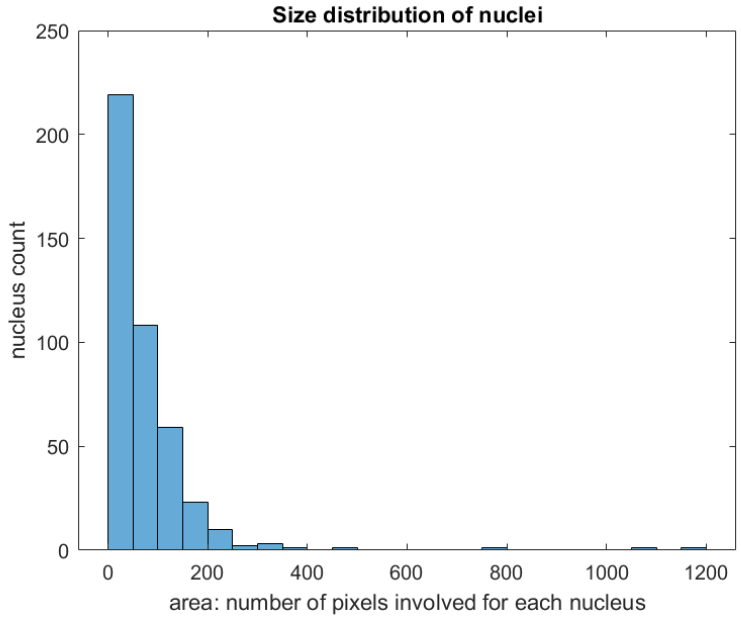
Finding sizes is by **computing the area of each nucleus region.**
Get the area property of all nuclei regions using ***regionprops()***. Display the area distribution in the **histogram**.

Result:
Evaluation: Most of the nuclei detected are small in size.

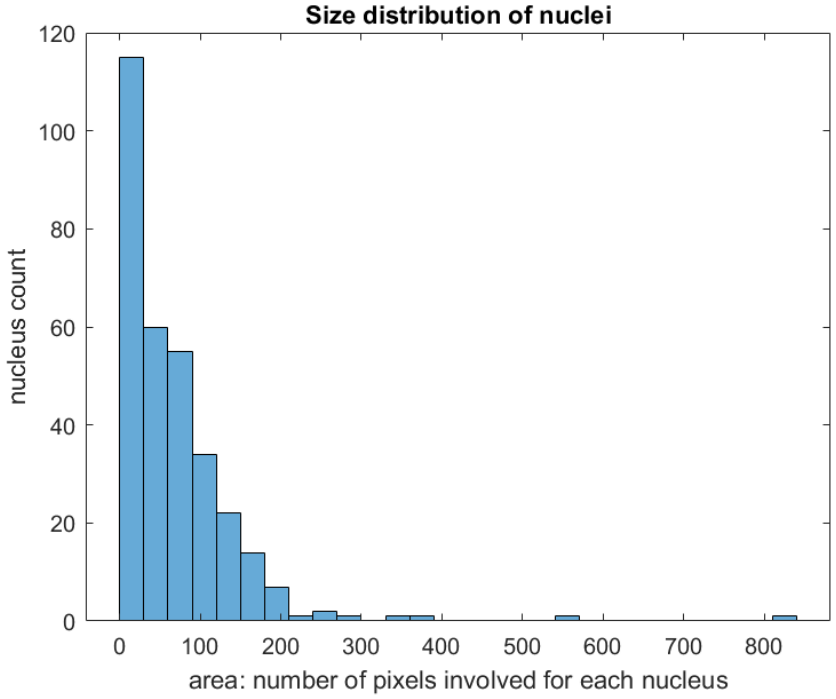
StackNinja1.bmp



StackNinja2.bmp



StackNinja3.bmp

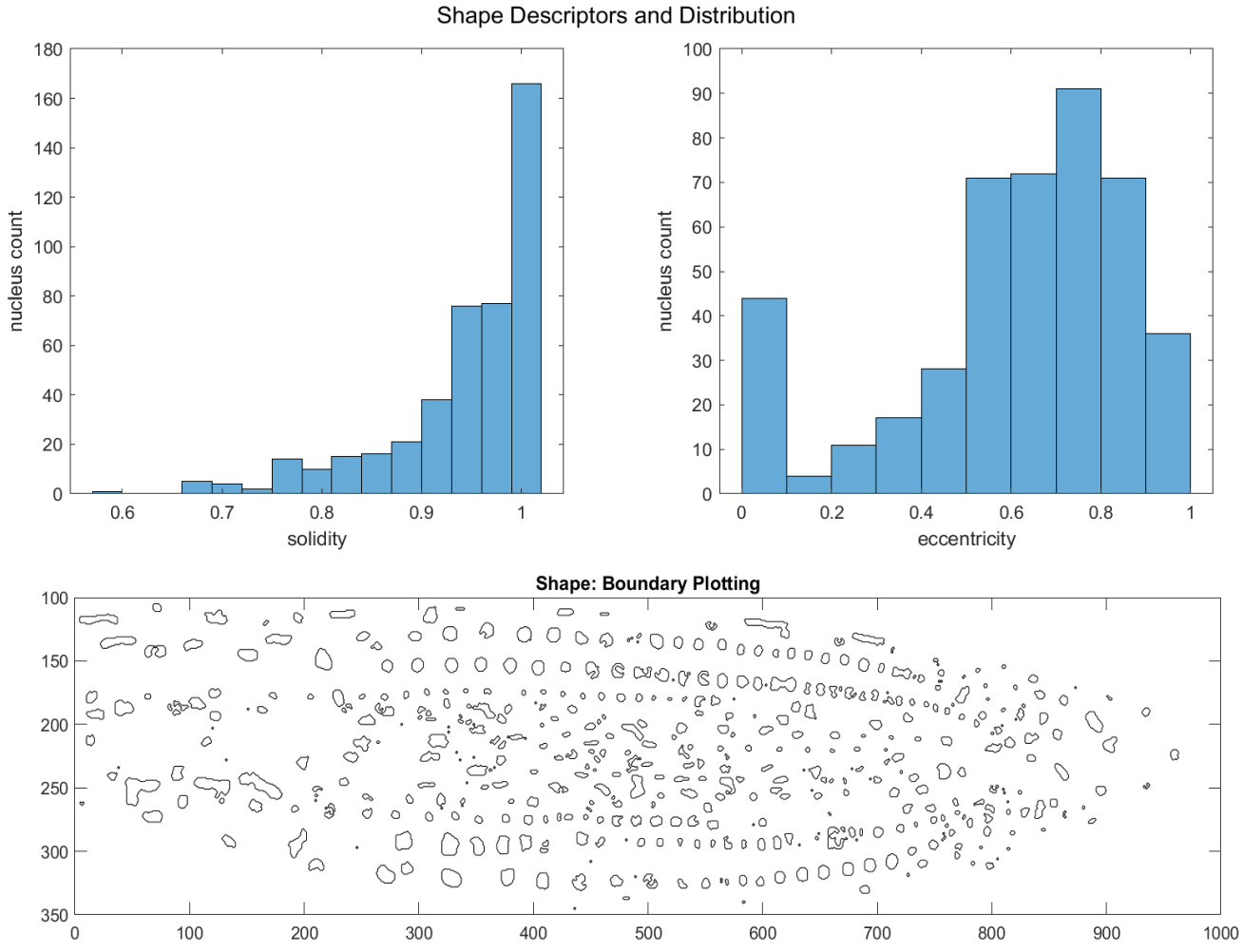


Task 3: Shapes of Nuclei, Distribution

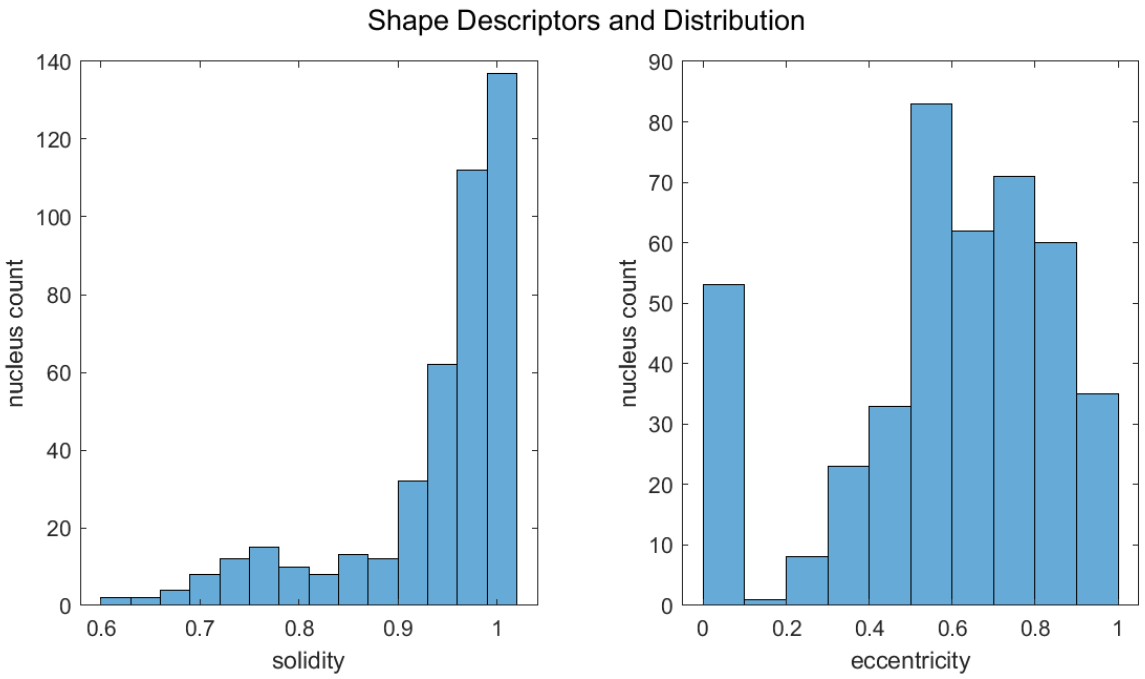
To see how round the nuclei are, get the **eccentricity**, **solidity** properties of the nuclei regions using ***regionprops()***.
Solidity defines how convex a region is. Eccentricity defines how elongated a region is.
Circles have solidity=1, so the higher solidity of a nucleus, the rounder it is.
Circles have eccentricity=0, so the higher the eccentricity of a nucleus, the less circular it is.
Display in histogram for each descriptor.

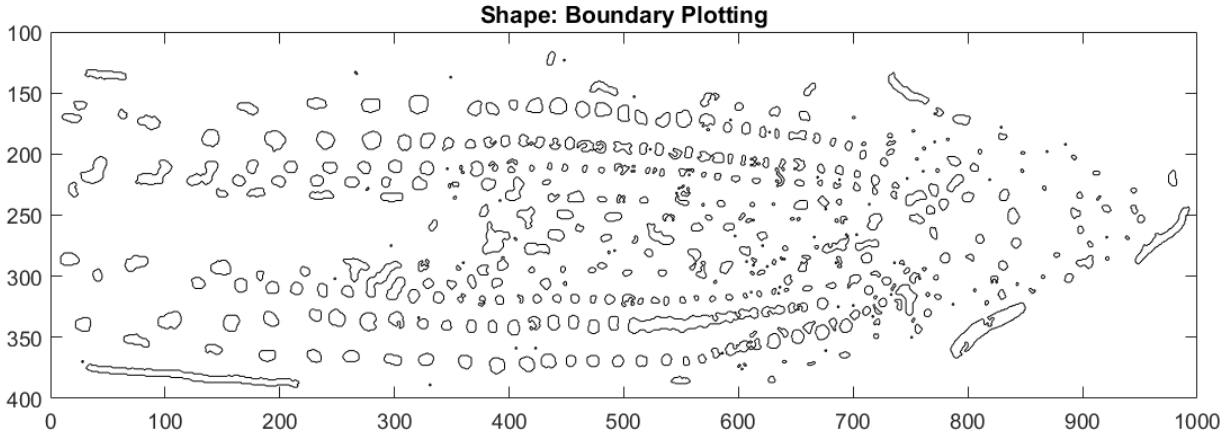
To directly observe the nuclei shapes, extract the **boundary** of each nucleus region using ***bwboundaries()***. Access each boundary data and **plot**. The boundary data of a nucleus will be the pixel coordinates of its boundary. The accuracy of nuclei extraction in ***Task1*** will also reflect on this boundary plotting. Then, generate **chaincode** for each nucleus from its boundary data using the ***fchcode.m*** file provided in the Lab. We get to know the shape of a nucleus by tracing through the chaincode.

Result:
StackNinja1.bmp

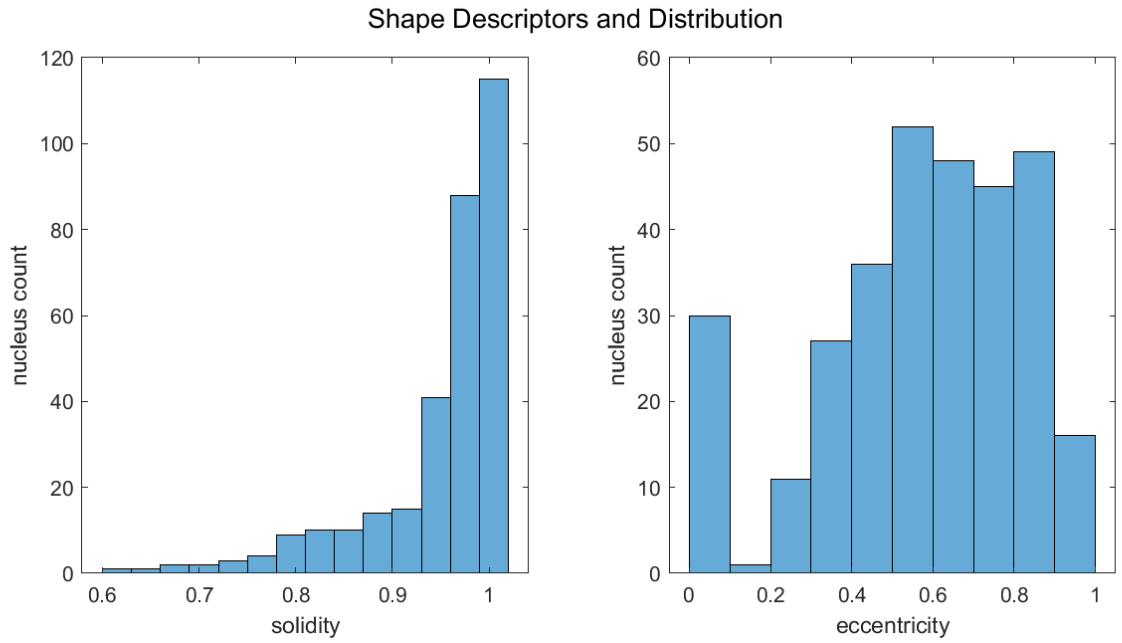


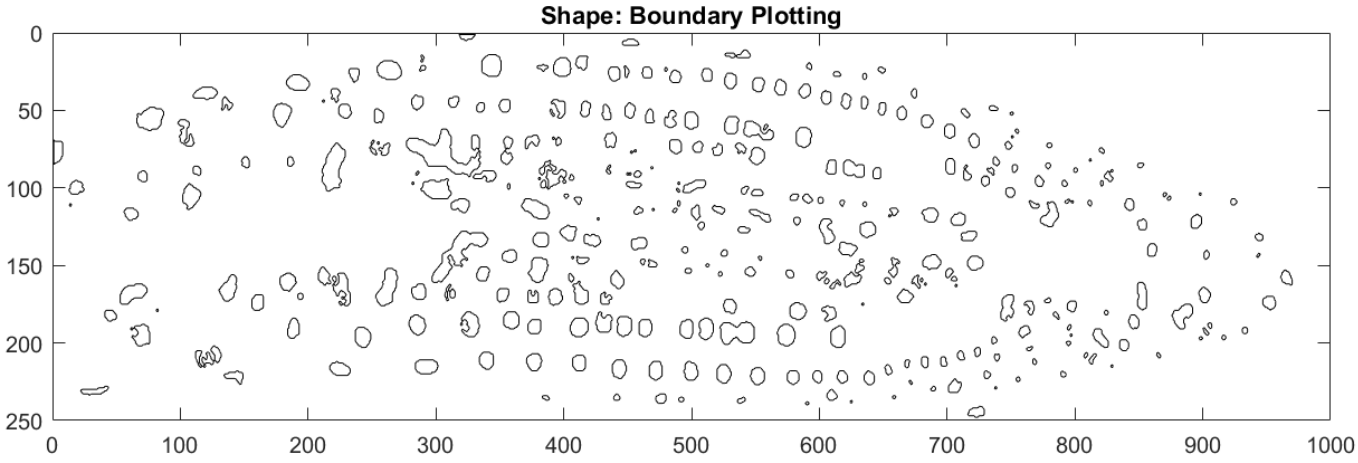
StackNinja2.bmp





StackNinja3.bmp





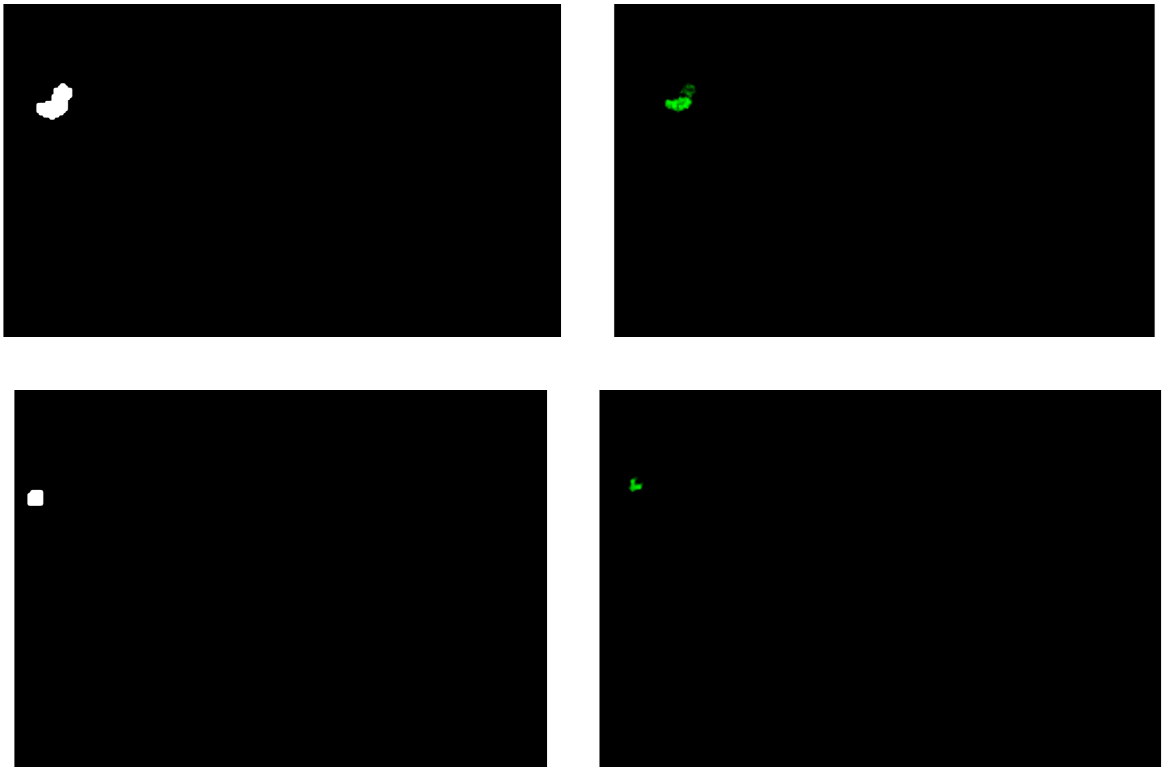
Task 4:Average Brightness of Detected Nuclei and Distribution

Step 1: Calculate the brightness of each detected nucleus region.

This is done by **calculating the average brightness of the image pixels in a nucleus region.** This is because 1 nucleus region may involve more than 1 image pixel and each image pixel may have a different brightness value. So, a good way to define the brightness of a nucleus is to find the average brightness of all its image pixels.

We go through each nucleus region 1 by 1 with looping and compute the brightness of each nucleus 1 by 1.

This means at each loop, we will have a current nucleus. Based on its position, we extract its region from the original RGB green image and have it mapped to **a temporary RGB image**. So, the temporary RGB image will have the current nucleus region shown in green and the surrounding in black (as shown below).



Then, we **convert the temporary RGB image to HSV** and **extract the brightness data *(: , : , 3)*** of only the current nucleus region without the black surrounding. The brightness data of a nucleus region is a list of brightness values of all image pixels in the region.

From the data, we compute
brightness of the current nucleus region
= sum of brightness of all image pixels in the nucleus region / total count of image pixels in the nucleus region

Step 2: Sum up the brightness of each nucleus region

At the end of each loop, we will **append the current nucleus brightness to the current *sum of nuclei brightness***, before moving on to the next nucleus region in the next loop.

Step 3: Calculate the average brightness

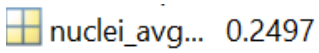
After the looping, we calculate
Average brightness = sum of nuclei brightness / total count of nuclei

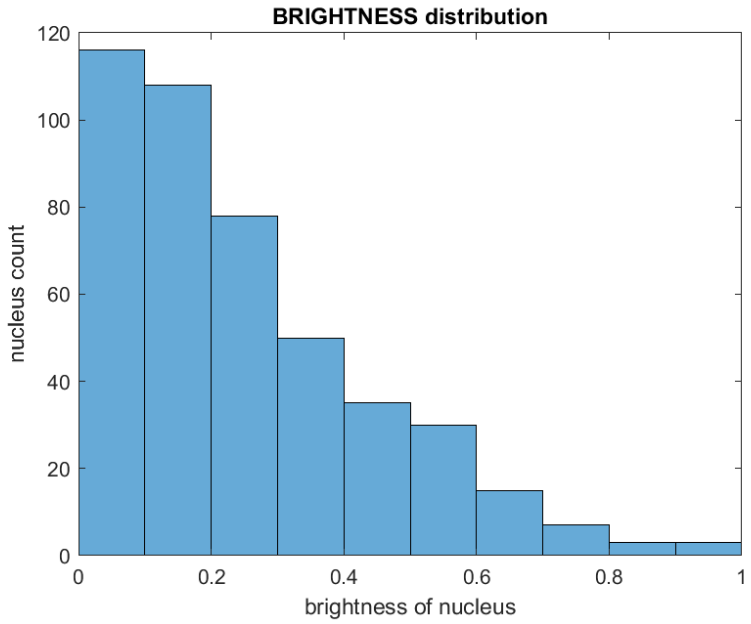
Step 4: Display brightness distribution of nuclei using Histogram

Display the nucleus count for each brightness (*ex*: *How many nuclei having brightness range from 0.4 to 0.5*).

Result:
Evaluation: Most of the nuclei detected inclined towards being dark except for StackNinja3.bmp where the nuclei brightness are more equally distributed.

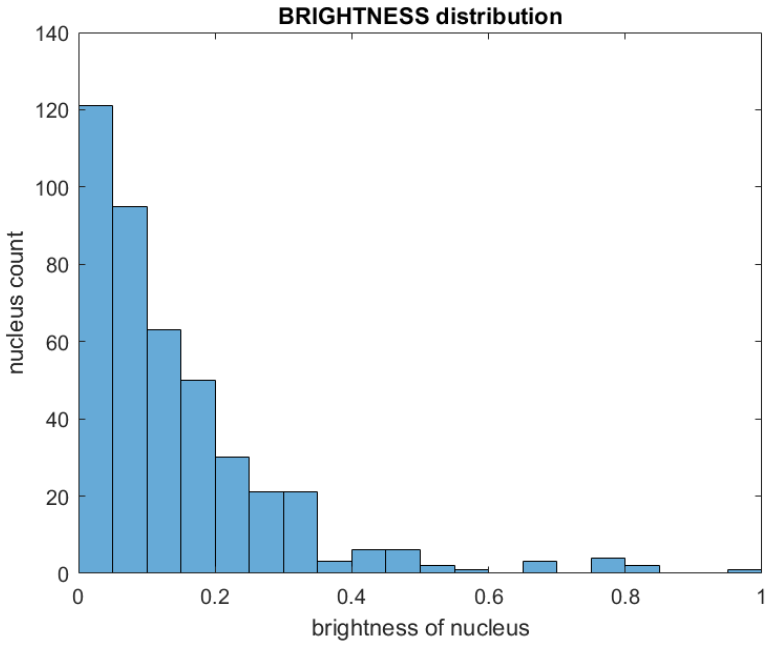
StackNinja1.bmp





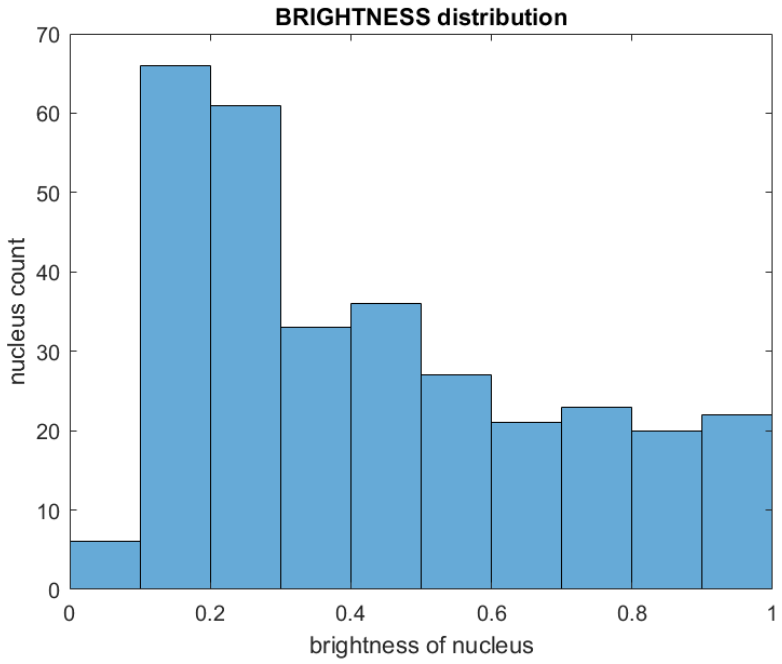
StackNinja2.bmp

nuclei_avg... 0.1443



StackNinja3.bmp

nuclei_avg... 0.4356



Link to Video: https://youtu.be/Ye9fALlzvnY

References

Correct Nonuniform Illumination and Analyze Foreground Objects. (n.d.). Retrieved April 23, 2021, from https://www.mathworks.com/help/images/correcting-nonuniform-illumination.html

Imfill. (n.d.). Retrieved April 23, 2021, from https://www.mathworks.com/help/images/ref/imfill.html#d123e142310