University of Nottingham
UK | CHINA | MALAYSIA

COMP2032

Introduction to Image Processing

Report

Name: Tang, Liqi
Student ID: 20259468

Report

Choose The Color Space:

The first step for me is to choose the correct color space. A color space is a method for quantitatively defining and representing colors. It is a mathematical paradigm that aids in standardizing the definition, representation, and reproduction of colors across many platforms and media. The different color spaces exist because they give a more natural method to identify colors or because they provide color information in ways that facilitate specific computations [1]. Once I looked through the MATLAB, I found a 'Color Threshold' in App. I can upload images into 'Color Threshold', and it will show us Figure 1: Several Color Space. And I found Figure 2: YCbCr is convenient to process the pictures for coursework.

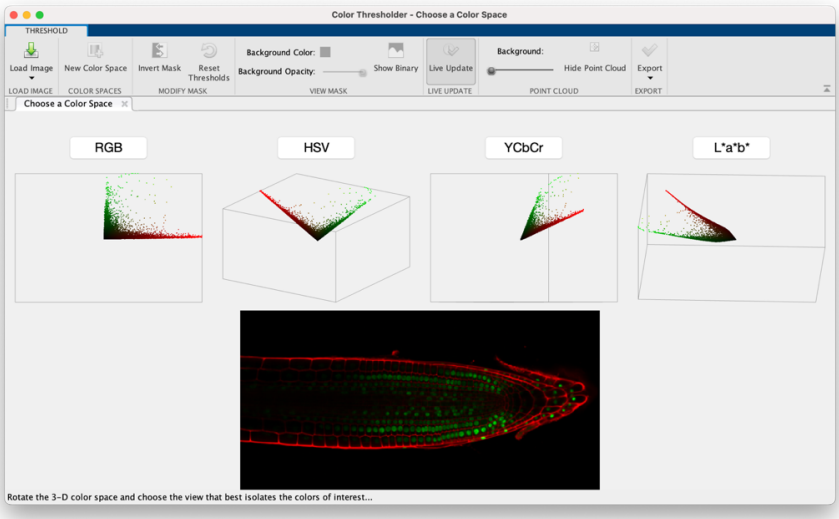


Figure 1: Several Color Space

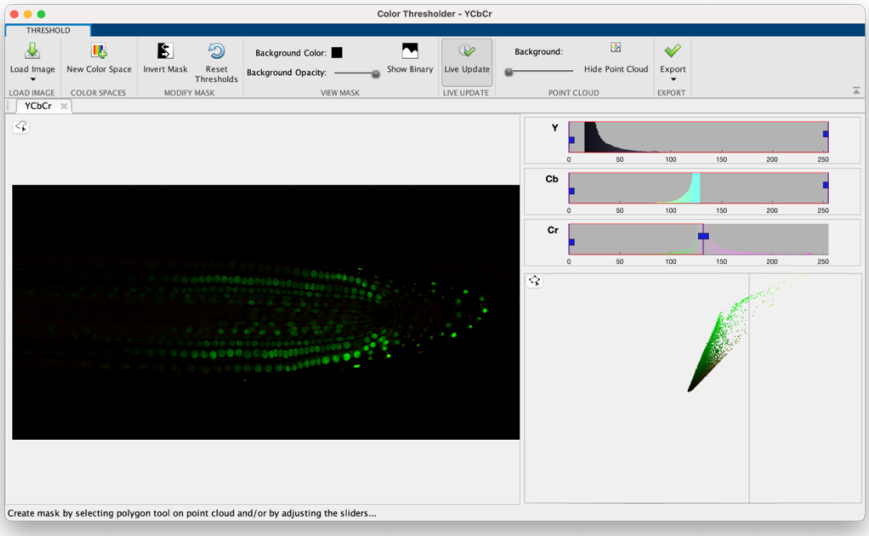


Figure 2: YCbCr

The luma part of the color is represented here by Y. The color's brightness is measured by the luma component. It refers to the color's lightness. This element is more perceptible to the human eye.
The blue and red components, Cb and Cr, are connected to the chroma component. Cb is the blue component in relation to the green component, and Cr is the red component in relation to the green component, according to this definition. These elements have a lower sensitivity to human vision [2].
Since Cr is the red component relative to the green component, I will first convert our RGB pictures into YCbCr Color Space (Figure 3: YCbCr Color Space ImageFigure 3: YCbCr Color Space Image) picture and extract the Cr from the pictures(Figure 4: Cr Image).

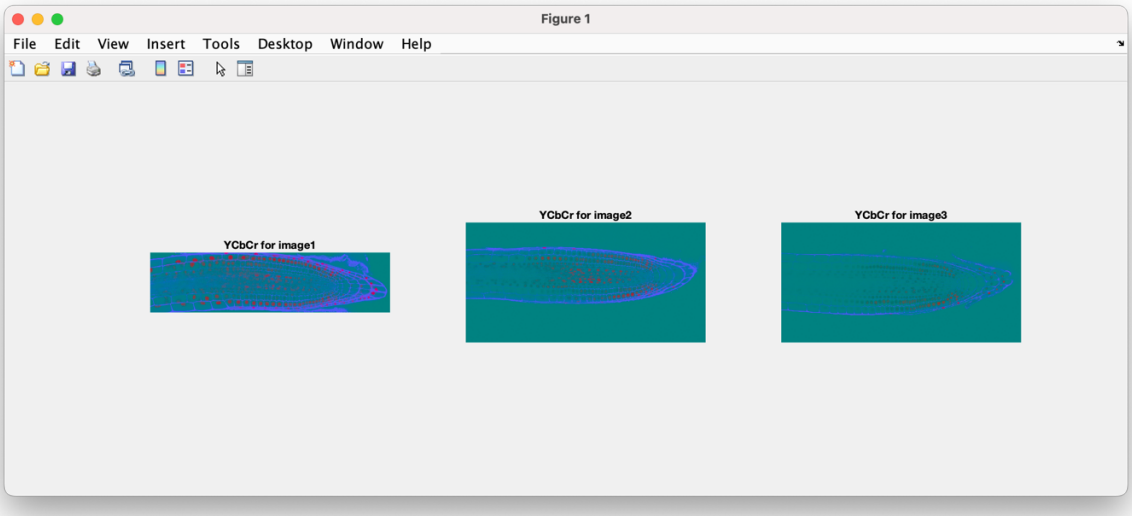


Figure 3: YCbCr Color Space Image

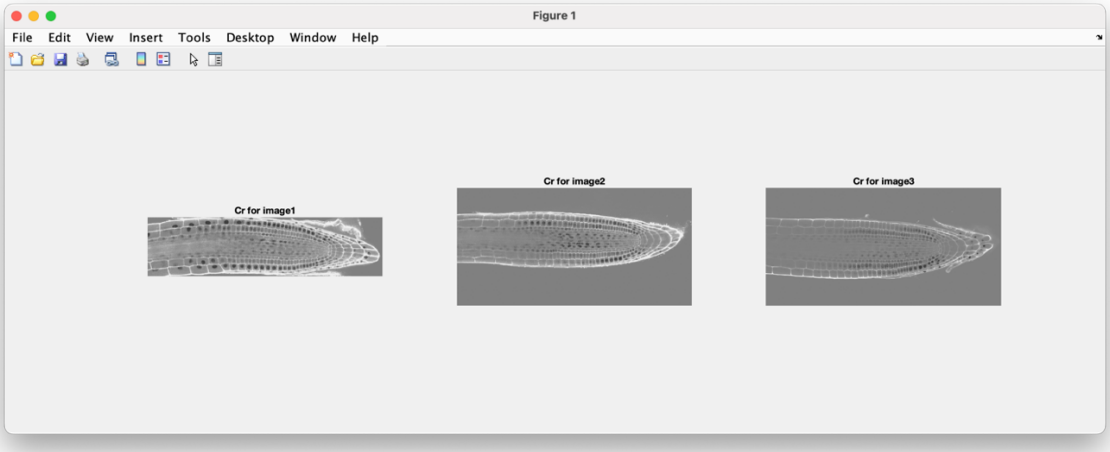


Figure 4: Cr Image

Get Binary Image and Convert to Greyscale Image:

Once I convert the image into a binary image, we can easily find there is a lot of noise there in the binary image (Figure 5: Binary Image Before Noise Reduction).

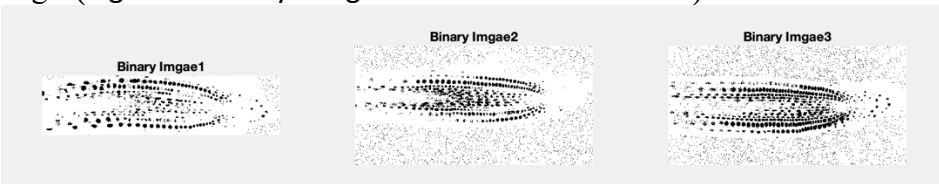


Figure 5: Binary Image Before Noise Reduction

To get the grayscale image for the next step to do the histogram equalization, I need to convert the Figure 4: Cr Image into Figure 6: greyscale image.

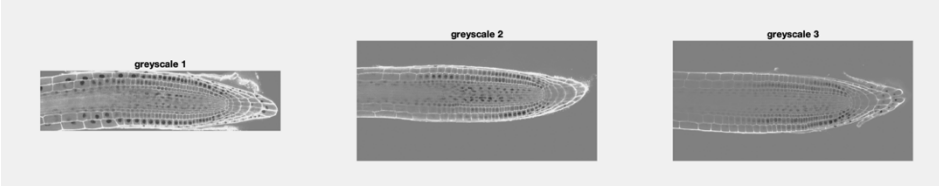


Figure 6: greyscale image

To get as clearer nuclear in the root as possible, I used the '**histqe**' function by applying to emphasize the grayscale of the nuclear in the image Figure 7: Histogram Equalization Image which is used to achieve histogram equalization on grayscale photos with pixel intensities spread more consistently throughout the whole range of potential intensities as output.

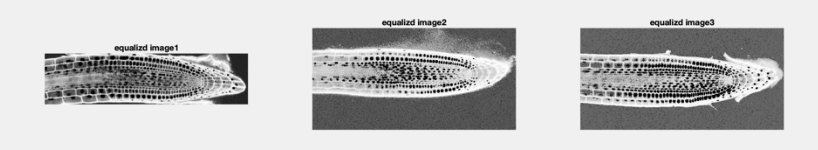


Figure 7: Histogram Equalization Image

After I got the histogram equalization image, I found that these equalized images cannot be applied with the filters directly since the background of the images are grey which means the filter can't distinguish and reduce the noises in the images clearly and for the requirements from the coursework, only the nuclear can be white. To handle this problem, I first use '**graythresh'** function to calculate the threshold value according to the histogram of Figure 7: Histogram Equalization Image, and according to the grey intensity value of Figure 7: Histogram Equalization Image pixels, the image pixels are divided into foreground and background. And apply the threshold value that gets from '**graythresh'** to the **'imbinarize**' function which comes out in Figure 8: Binary with Threshold Image.

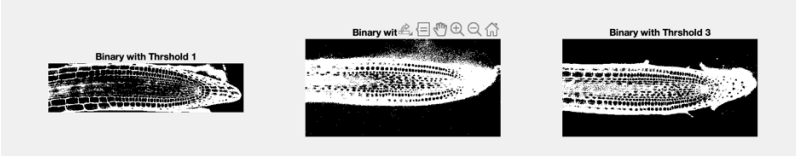


Figure 8: Binary with Threshold Image

But in Figure 8: Binary with Threshold Image, both background and nuclear are black and the outline of the root which is filled with red color in the original image is white. In this case, I add

Figure 8: Binary with Threshold Image which gets both black background and nuclear and Figure 5: Binary Image Before Noise Reduction which gets white background and black nuclear together, whereas the black background will subsequently be overlaid with white while retaining the nuclear black color. Finally get Figure 9: Final Binary Image with white background and black nuclear and then convert to Figure 10: Final Grayscale Image for edge smoothing and noise reduction.

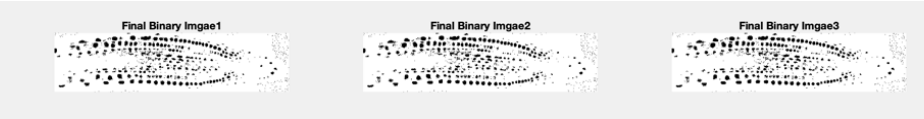


Figure 9: Final Binary Image

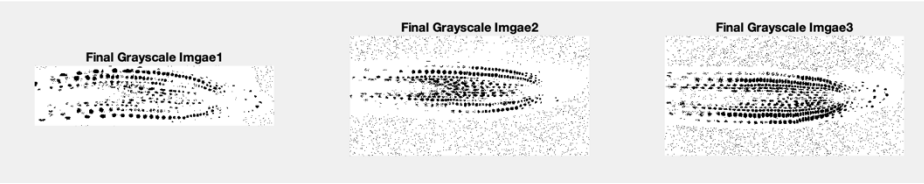


Figure 10: Final Grayscale Image

Edge Smoothing:

However, due to the histogram equalization, the edge of the nuclear is sharpened and jagged. To smooth the edge of the nuclear while keeping the space between the nuclear, I used superpixel by using the SLIC algorithm --- '**superpixels(A, N, Name, Value)'** function. Since superpixels bring together nearby pixels with similar intensity and texture, also can help smooth the borders of a picture. Since I previously employed histogram equalization in this instance to accentuate the nuclear grayscale, it is appropriate and simple to identify the nuclear from the backdrop or root in Figure 10: Final Grayscale Image. I set the number of superpixels = 1000 (since the smaller the number of superpixels is, the narrower the boundary of the nuclear edges are) and set compactness factor = 10 which determines the weighting of spatial distance and intensity difference when assigning pixels to superpixels. I also tried to decrease the number of compactness factors to 5, and it shows Figure 11: Difference between compactness factor 10 and 5 when I subtract two images with compactness factor 10 and 5, however, the difference between the two images will not affect extract the nuclear, so it does not matter whether choose 5 or 10, so I choose 10.

Figure 11: Difference between compactness factor 10 and 5

Following the superpixels Figure 12: Superpixels for Image1, Figure 13: Superpixels for Image2 and Figure 14: Superpixels for Image3, we can readily see that there is a very slight shift in all three photos, with only a little reduction in the number of nuclear, the nuclear edge has become smoother and the spacing between nuclear has also somewhat extended.

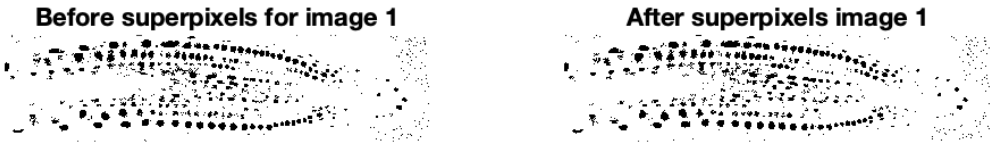


Figure 12: Superpixels for Image1

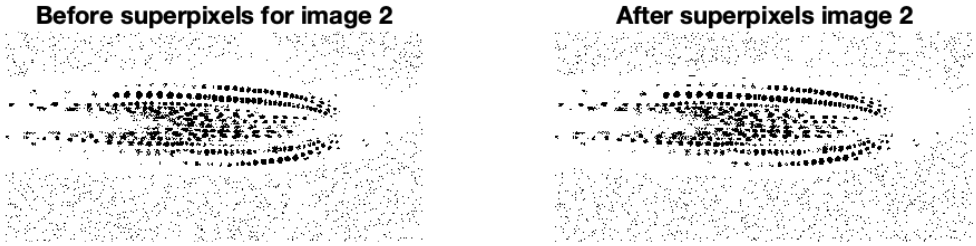


Figure 13: Superpixels for Image2

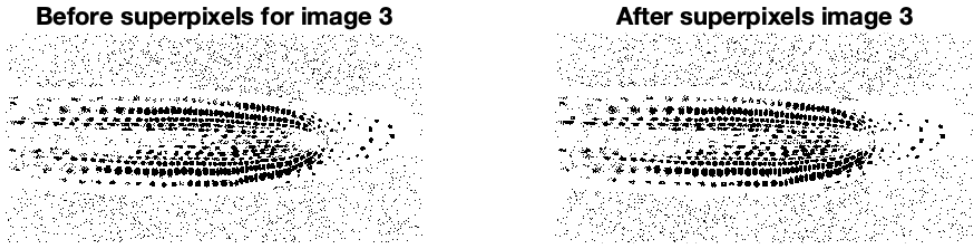


Figure 14: Superpixels for Image3

Noise Reduction:

For noise reduction, I need to utilize filters to minimize the noise in the greyscale image. The filtering technique can be used to improve or modify a photograph. For example, you might filter an image to accentuate some parts while deleting others. Filtering is used in image processing to perform tasks such as edge enhancement, sharpening, and smoothing [3]. But to ensure the balance between the noise reduction and the completeness of the nuclear, I used two filters --- wiener filter and the gaussian filter.

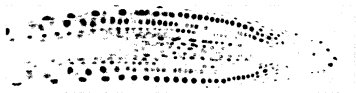
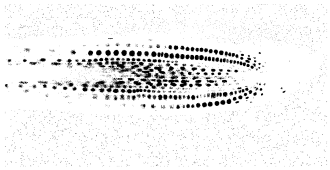
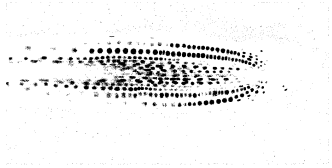
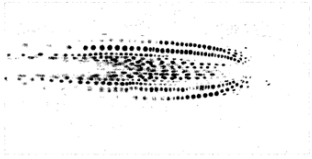
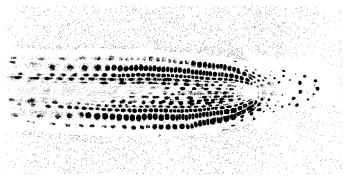
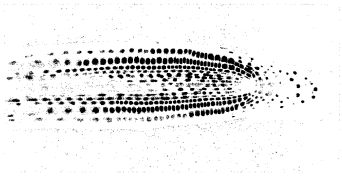
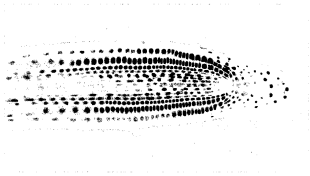
A. Wiener Filter

The Wiener filter is a linear filter that removes noise from an image by minimizing the mean square error between the original image and the filtered image. It is effective in reducing Gaussian noise in an image while preserving the edges.

The grayscale picture is filtered by Wiener A pixel-wise adaptive low-pass. The neighbourhood size (m-by-n) needed to calculate the local image mean and the standard deviation is specified by the expression [m n]. The input image has been damaged by additive noise with constant power.

'Wiener2' employs the pixel-wise adaptive Wiener technique based on data extrapolated from each pixel's immediate surroundings [4].

To apply the wiener filter, I can use the built-in function **'wiener2 (I, [m n])'** and change the neighbourhood size (m-by-n). In this coursework, I will apply three different neighbourhood size (m-by-n) which is 3*3, 4*4, and 5*5 in these three images. And we can see the output for 3*3 neighbourhood Figure 15: Wiener Filter 3*3 here; and the output for 4*4 neighbourhood size is Figure 16: Wiener Filter 4*4; for 5*5 neighbourhood size is Figure 17: Wiener Filter 5*5, and can easily discover that the bigger the neighbourhood size is, the lesser the noise and the number of nuclear are. However, no matter whether the neighbourhood is up to 4*4 or 5*5, there are still some noises in image2 and 3, especially at the border of the image. To improve this problem, I applied the gaussian filter after the wiener filter.

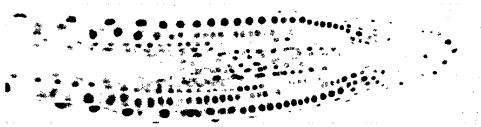
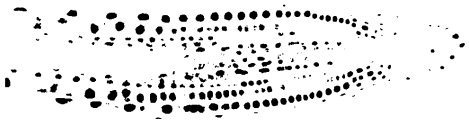
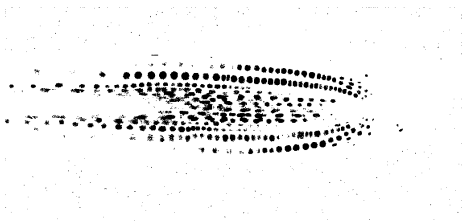
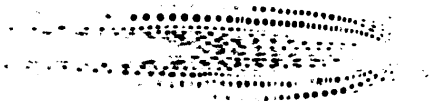
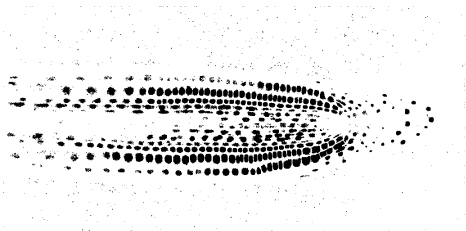
*Figure 15: Wiener Filter 3*3*	*Figure 16: Wiener Filter 4*4*	*Figure 17: Wiener Filter 5*5*
 *Figure 14a: 3*3 for image1*	 *Figure 15a: 4*4 for image1*	 *Figure 16a: 5*5 for image1*
 *Figure 14b: 3*3 for image2*	 *Figure 15b: 4*4 for image2*	 *Figure 16b: 5*5 for image2*
 *Figure 14c: 3*3 for image3*	 *Figure 15c: 4*4 for image3*	 *Figure 16c: 5*5 for image3*

B. Gaussian Filter

The Gaussian filter is a linear filter that replaces each pixel value with the weighted average of its neighbouring pixels. A Gaussian function is used to calculate the weights, which gives adjacent pixels more weight and distant pixels less weight. It successfully lowers Gaussian noise in a picture while keeping the edges sharp.

To apply the gaussian filter, I can use a built-in function in MATLAB which is **B = imgaussfilt(A, sigma)**, that by changing the value sigma --- Gaussian distribution's standard deviation, either as a positive integer or a two-element vector of positive values. And in this coursework, I will apply two values for sigma which are 0.5 and 1.0.

According to all the results shown above which apply different sigma values on three images Figure 18: Gaussian Filter (Sigma=0.5) and Figure 19: Gaussian Filter(Sigma=1.0) ---- I discovered that applying a sigma value of 1.0 within a 3*3 neighbourhood when using a Wiener filter can effectively reduce noise in all three images. This suggests that there is no need to experiment with larger neighbourhood sizes such as 4*4 or 5*5. We can plainly see that the higher the sigma value, the less noise there is around the border and in the centre of the images, however, the number of nuclear is also reduced.

Figure 18: Gaussian Filter (Sigma=0.5)	*Figure 19: Gaussian Filter(Sigma=1.0)*
 *Figure 18a: 3*3, Sigma=0.5 for image1*	 *Figure 19a: 3*3, Sigma=1.0 for image1*
 *Figure 18b: 3*3, sigma=0.5 for image2*	 *Figure 19b: 3*3, sigma=1.0 for image2*
 *Figure 18c: 3*3, sigma=0.5 for image3*	 *Figure 20c: 3*3, sigma=1.0 for image3*

In summary, the Wiener filter is good for reducing Gaussian noise while preserving edges and details, while the Gaussian filter is good for reducing Gaussian noise while retaining edges. First, use a wiener filter to reduce most of the noises in the images while preserving edges and details, then, use the gaussian filter to reduce noise that can't be reduced by the wiener filter, meanwhile, retaining edges. And the best output for noise reduction in my algorithm is first to apply a 3*3 neighbourhood by using the Wiener filter with a sigma value of 1.0 using the gaussian filter.

Random Color Fill:

Since the noise reduction image here is the binary image for three images Figure 19: Gaussian Filter(Sigma=1.0) by using 3*3 neighbourhood wiener filter and Gaussian filter where sigma =1.0, the next step for disposing of the binary image is to convert the background color to black

and the nuclear to white since I have to fill the random color in the binary image. Since these are binary images, I can easily use **'1-binary image'** to convert the color like Figure 21: Convert Image.

Figure 21: Convert Image

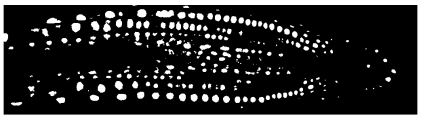


Figure 21a: Convert Image1

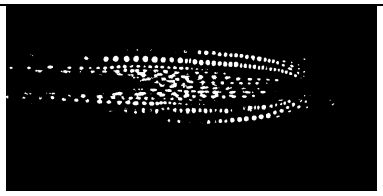


Figure 21b: Convert Image2

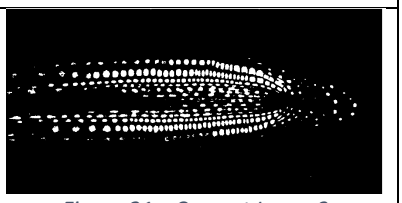


Figure 21c: Convert Image3

Then I used the **'bwconncomp'** function first to identify the connected components of the binary image which is white nuclear here. And the number of nuclear is determined by using the **'NumObetcts'** property of the binary nuclear image. Generating a matrix of random RGB colors by using the 'rand' function where the number of colors randomly generated is equal to the number of nuclear. The **'label2rgb'** function is used to generate a new picture in which each linked component is allocated a color at random from the color's matrix. and the **'shuffle'** argument is used to randomize the sequence in which the colors are given to the components. And the results are like Figure 22: Final Output for Image1, Figure 23: Final Output for Image2 and Figure 24: Final Output for Image3 showed.

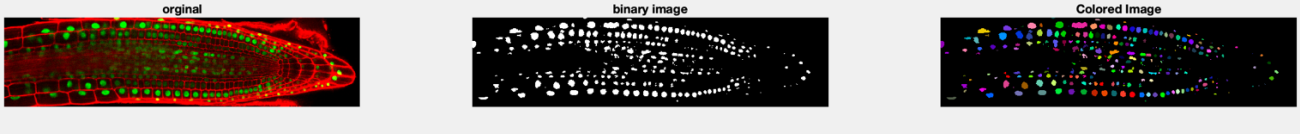


Figure 22: Final Output for Image1

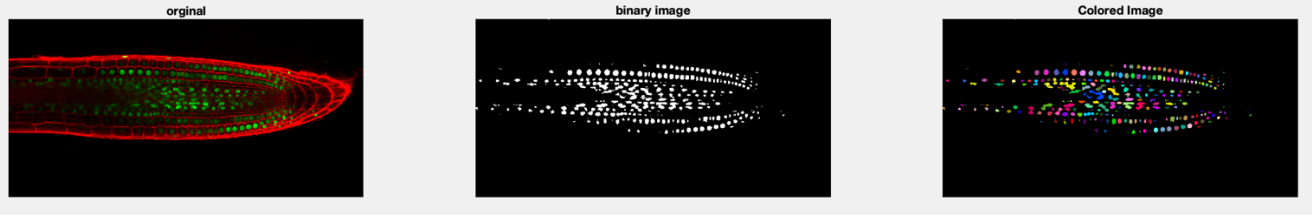


Figure 23: Final Output for Image2

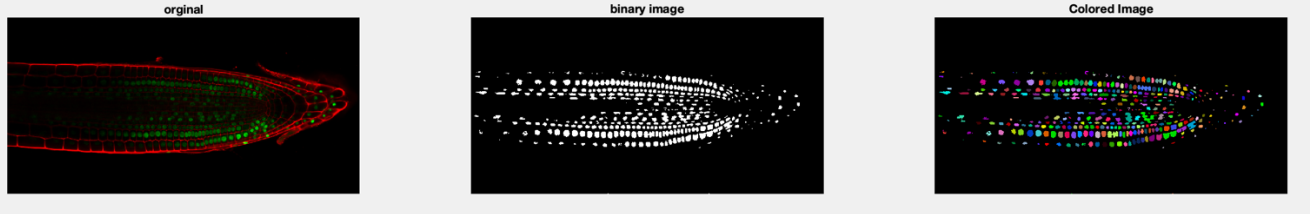


Figure 24: Final Output for Image3

Strength:

To summarize, the program's strength is that it can balance the output of these three images only by modifying the input source image; no hardcoded parameters or human intervention are required for the program to provide a successful outcome. The approaches are employed to guarantee that the best results are obtained for all three images. To get the best outcomes for all three photos, much consideration was given to establishing the ideal ratio between the spacing between each nuclear and the number of the nuclear.

Weakness:

The weakness of this program is also very easy to be discovered. As I mentioned before while using the histogram realization to enhance the grayscale of the nuclear, the edge of the nuclear is sharpened and jagged. To keep the internal space between the nuclear, no matter what I do with the smooth edge function, the edge of the nuclear can never get back to as same as the original image and due to the use of superpixels to do the edge smoothing, the number and size of the nuclear also reduced.

One more thing is, since I applied the filter, while reducing the noise, some of the dark green nuclear also reduced as noise, so it is impossible for me to extract the whole nuclear in the original image.

Conclusion:

Based on the rationale and outcomes I presented before, I consider combining the histogram equalization method and superpixels with the wiener filter by using 3*3 neighbourhood size and Gaussian filter with sigma=1.0 as the ideal strategy for this course. However, it is difficult to fully extract the nuclear in the images clearly while remaining the edge between each nuclear. Also, the illumination of nuclear also can greatly affect image processing.

It is challenging to achieve a flawless solution in image processing. As a result, we must always maintain a balance and understand what is providing and what is taking. In conclusion, image processing is a complicated area that needs careful consideration of lots of elements like image noise, illumination, image resolution, image segmentation etc. to provide accurate and practical outcomes.

Reference:

1. “Select a web site,” *Understanding Color Spaces and Color Space Conversion - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/help/images/understanding-color-spaces-and-color-space-conversion.html. [Accessed: 12-Mar-2023].

2. D. Dias, “What is YCbCr? (color spaces),” *Medium*, 04-May-2017. [Online]. Available: https://medium.com/breaktheloop/what-is-ycbcr-964fde85eeb3. [Accessed: 12-Mar-2023].

3. “Image filtering,” *Image Filtering - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/help/images/linear-filtering.html. [Accessed: 13-Mar-2023].

4. “wiener2,” *2-D adaptive noise-removal filtering - MATLAB*. [Online]. Available: https://www.mathworks.com/help/images/ref/wiener2.html#d124e319051. [Accessed: 13-Mar-2023].