COMP2032 Introduction to image processing

Spring 2017 / 2018

Project title : Counting cell nuclei

Date : 23rd April 2018

Name	Student ID
Loh Jin Xian	016763

Module convener : Dr Amr Ahmed

The purpose of this coursework is to calculate the amount of cell nuclei present in the given images which are StackNinja1.bmp, StackNinja2.bmp and StackNinja3.bmp. Steps that are used in the process of obtaining the cell nuclei are given below. Explanations will be shown using StackNinja1.bmp image.

Step 1: Colour Space Conversion

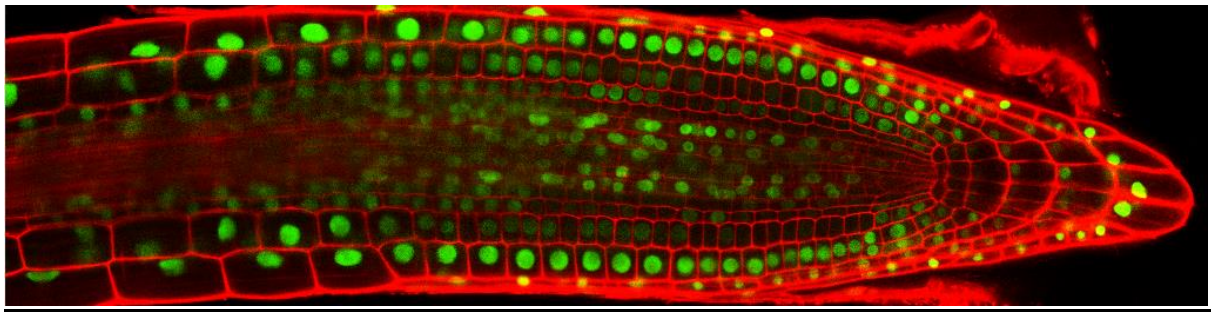


Figure 1.0 StackNinja1.bmp

By observing the original image, most of the nuclei can be considered green in colour. Therefore, it is logical by first extracting the green channel from the image. This is done by using “GreenChannel = ImageUsed(: ,: ,2);”

Figure 1.1 Green channel of StackNinja1.bmp

Explanation

By extracting the green channel from the image, most of the cell walls which are red in colour won’t be present. Other than that, this step turns the input image into a grayscale image. By working in low dimensional space, it offers a lot of benefits. For example, grayscale image has only 1 plane of image data and only need to apply the image processing operations once while image that is in RGBA need to apply more image processing operations and combine the result as it has 4 images planes [1].

Besides, the processing time is increased as the processed grayscale image only have ¼ of data due to data reduction compared to the colour image.

However, grayscale image has lower quantity of image data and decreased information (e.g. colour information) compared to its original colour image. By examining figure 1.0 again, some nuclei have red channel value, making the nuclei in figure 1.1 less noticeable with only green channel extracted. While this may be true, figure 1.1 can be processed to make it look shaper and brighter.

Step 2: Image Pre-processing and Enhancement

To improve the brightness of figure 1.1, the image is scaled by a constant of 1.4 which will multiply each pixel in the image by 1.4. A median filter is then applied to the image followed by a Gaussian filter. Afterwards, Laplacian is applied to the image and the output is subtracted from the processed image. Finally, median filter is applied again to the processed image.

Figure 2.0 Enhanced green channel image (Median & Gaussian filter)

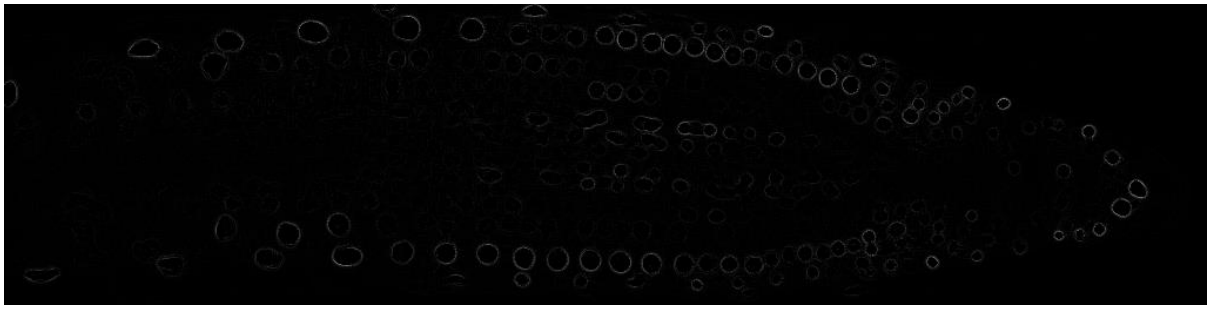


Figure 2.1 Laplacian of green channel image

Figure 2.2 Subtraction of figure 2.1 & figure 2.2

Explanation

The image is scaled by using “immultiply(GreenChannel, 1.4)” instead of “imadd” because it can preserve the blacks in the pixels. This is used to brighten up the image while maintaining the contrast of the overall image. Then, median filter is applied before the Gaussian filter as median filter is a non-linear filter than can reduce noises in an image while keeping the edges of the image relatively sharp. After that, Gaussian filter is applied to blur the additive noise. The smoothing of image from Gaussian filter helps the next process as Laplacian filter is a derivative filter that is very sensitive to noise. These 2 steps process is call the Laplacian of Gaussian (LoG) operation [2].

Here is a comparison of noise filled & noise reduced images:

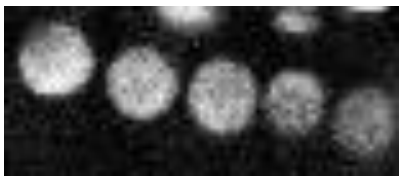


Figure 2.3 Noise filled image

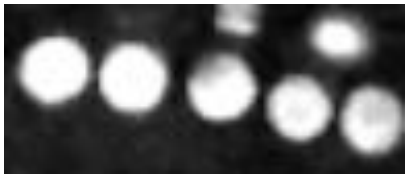


Figure 2.4 Noise reduced image

Once the edges are obtained, they are subtracted from the processed image which result in figure 2.2 that has better differences between foreground and background. Lastly, median filter is applied again to remove all the remaining impulse noise.

Step 3: Canny Edge Detection

After the image is processed properly, canny edge detection technique is used to detect the edges of the nuclei. This is done using “edge(finalGreen, 'canny')”.

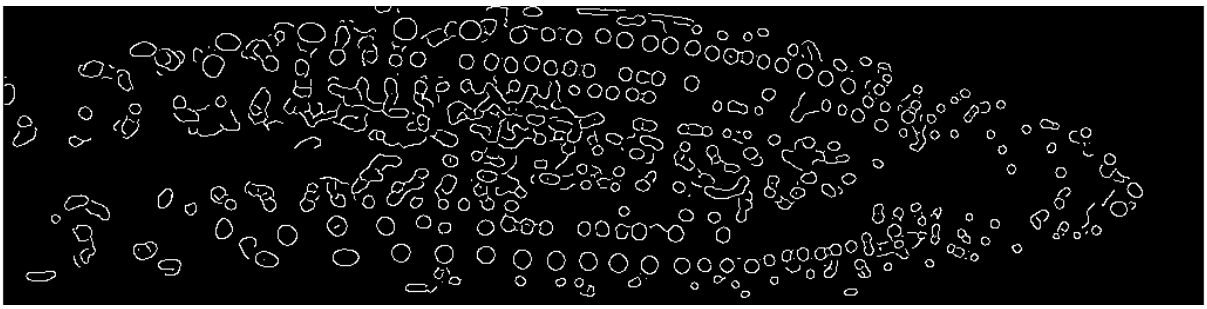


Figure 3.1 Canny edge of processed image

Explanation

Canny edge detection is used as it involves different steps in detecting the edges. For example, Canny edge detection use of Non-maximum suppression enables edge candidates which are not dominant in their neighbourhood not considered to be edges which will remove all non-nuclei pixels. Besides that, it also uses hysteresis thresholding that has 2 thresholds that can help filling in pixels that are supposed to be a part of the edges.

When compared to other edge detection techniques like Sobels or Robert’s Cross, canny edge detection works better in detecting actual nuclei edges. While it is better,

it still has its flaw as some non-nuclei edges can still be highlighted as shown below:

Figure 3.2 Non-nuclei edges

Furthermore, canny edge detection technique requires more computation time. Overall, it has better edge detection and good localisation which helps detecting the nuclei better. The use of thresholding in this technique also allow the image to be binary which aid in the incorporation of morphology processes later.

Step 4: Morphological Image Processing

After getting the edge of the image, the function “imdilate” with disk-shaped structuring element is used to fill in some missing pixels that prevent the edges from connecting. Then, the edges are filled by using “imfill(final4, 'holes')”. Next, the image is eroded to reduce its noise seen in figure 3.2 followed by “bwareaopen(final6 , 10)” that will remove all objects that have pixel value lower than 10. The image is once again dilated to get back to its original size.

Once everything is done, watershed technique is used to separate some of the connected nuclei in the image. This is done by computing the distance transform of the image's complement and negate the distance transform to turn the two bright areas into catchment basins using “-bwdist(~final8)”. Then, “imextendedmin(D,2)” and “imimposemin(D,mask)” is applied to filter out tiny local minima and modify the distance transform. Lastly, “bw3(Ld2 == 0) = 0” to separate the connected objects in the original image.

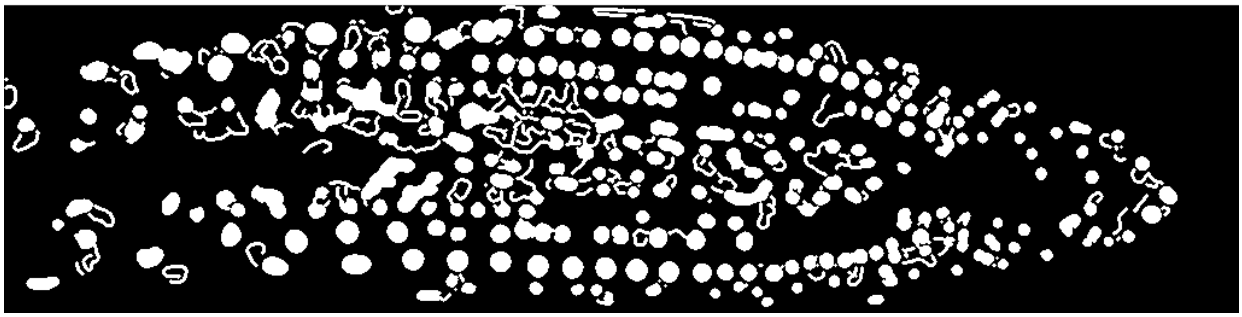


Figure 4.0 Dilated and filled canny edge

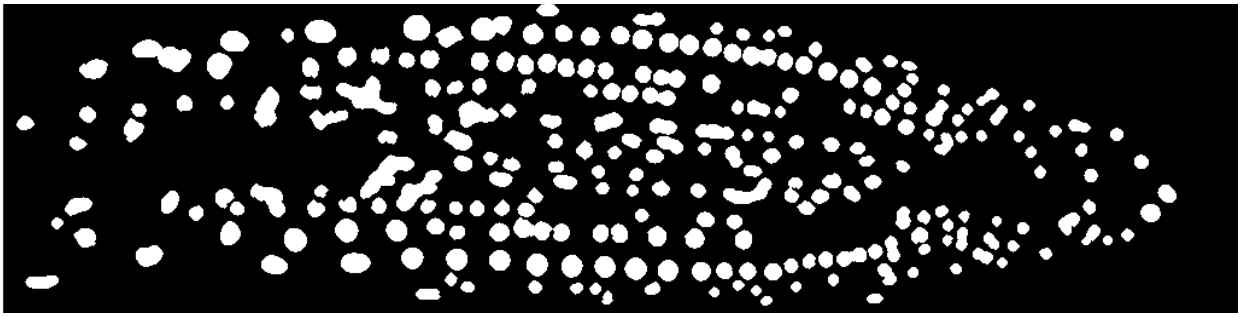


Figure 4.1 Clean-up of figure 4.0 with morphological image processing

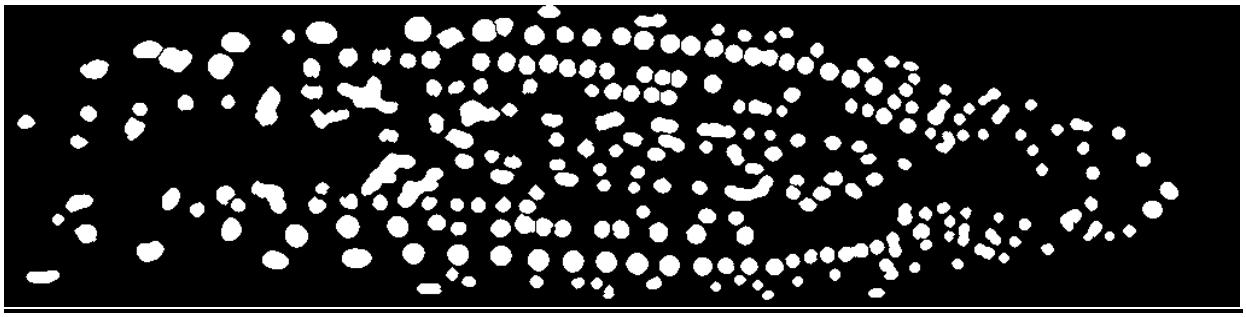


Figure 4.2 watershed image of figure 4.1

Explanation

By using morphological image processing, figure 4.0 can be turned into a cleaner and clearer image as shown in figure 4.1. Nuclei that have broken edges can finally be connected and edges noise can be removed effectively. While this is desirable, morphological image processing like dilation may connect some of the unconnected nuclei unintentionally.

This is where watershed segmentation technique comes in handy. Watershed is used to find the watershed lines in an image in order to separate the distinct regions. However, the "raw" watershed transform is known for its tendency to "over segment"

an image due to tiny local minima becoming a catchment basin. That is why “imextendedmin” is used to filter out tiny local minima and “imimposemin” to modify the distance transform so that no minima occur at the filtered-out locations [3].

Once the catchment basins are identified, the zero-valued elements of label matrix Ld2, which are located along the watershed lines, can be used to separate the connected nuclei in the image [4].

.

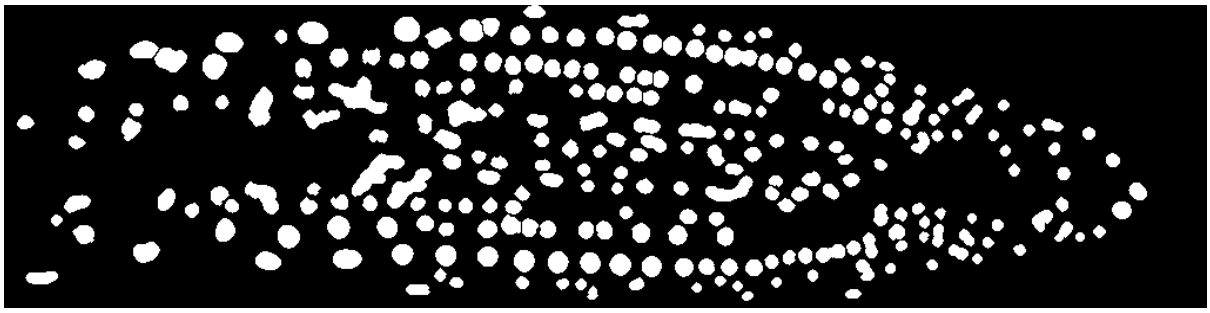
Figure 4.3 Nuclei without watershed

Figure 4.4 Nuclei with watershed

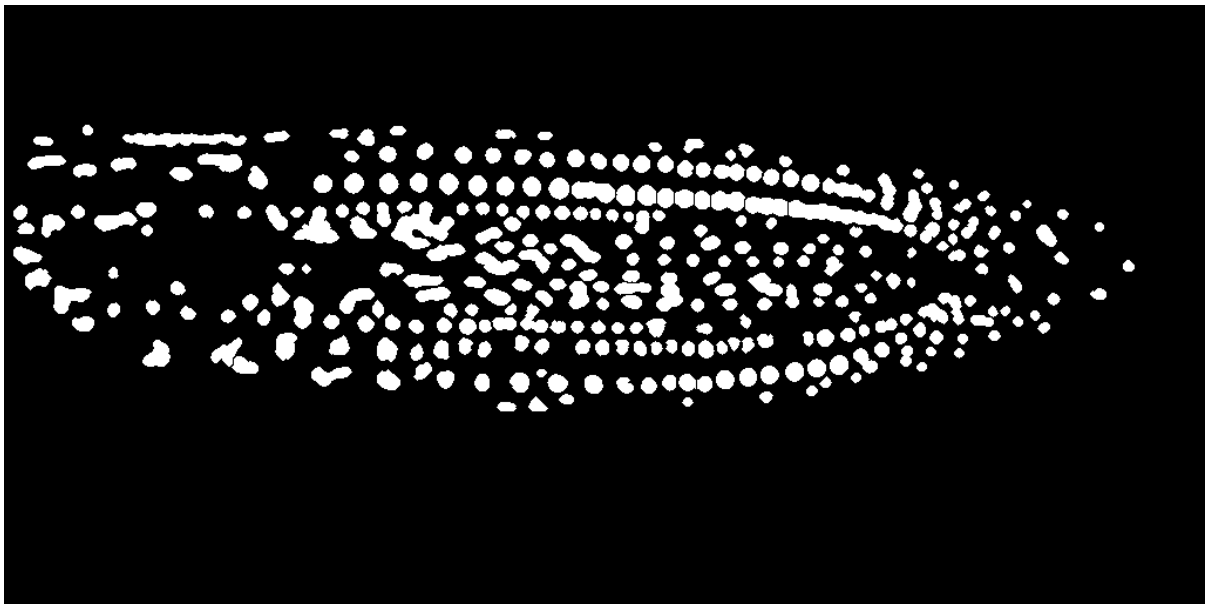
Although the watershed segmentation didn’t successfully separate all connected nuclei, but it enables more nuclei to be spotted which satisfy our final objective to count as many nuclei as possible.

Final output of all images

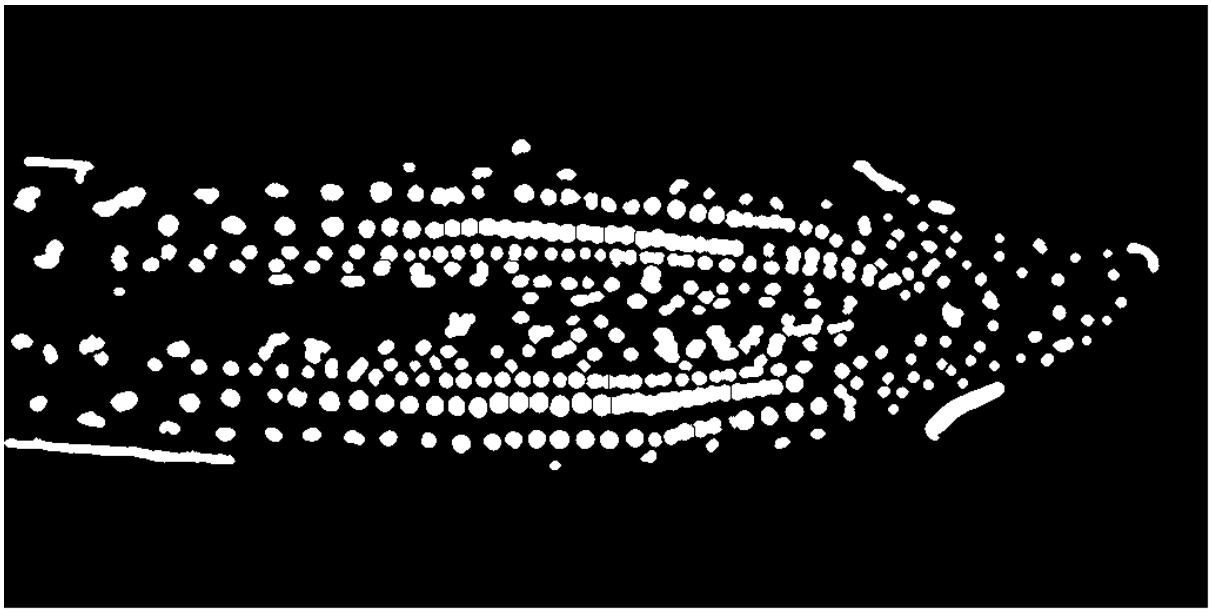
1. StackNinja1.bmp



2. StackNinja2.bmp



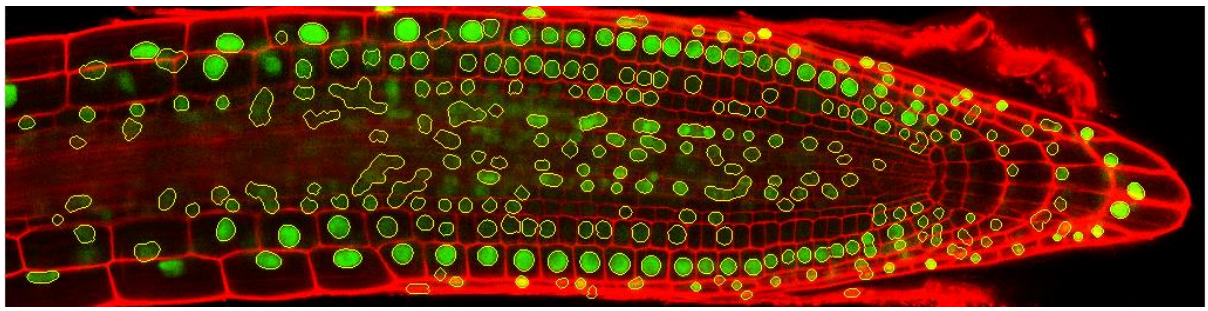
3. StackNinja3.bmp



Comparison of cells detected with the original input image

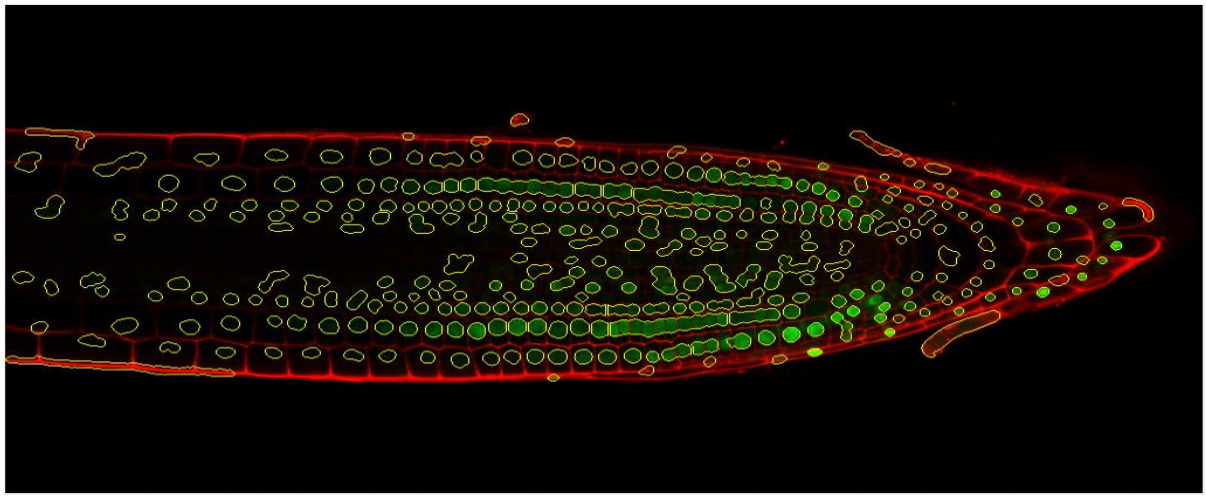
By using “bwperim(bw3)”, the perimeter pixels of the cells in the image can be obtained. Then, overlay the perimeter pixels on top of the original input image with “imoverlay(ImageUsed, bwperim(bw3))” to get a visualization on how many of them are successfully detected.

1. StackNinja1.bmp



2. StackNinja2.bmp

3. StackNinja3.bmp



Conclusion

The processes used in detecting and calculating the nuclei are successful but not perfect. In summary,

- working in a low dimensional space,
- using image processing techniques to improve the image
- using Canny edge detection to obtain the edges
- using morphological processes to make the image clearer

are just one of the many ways to complete this objective.

Therefore, it is important to innovate and find better ways of detecting the nuclei in the future. Although the methods listed above are not entirely perfect, but I am satisfied with it because it takes a lot of trials and errors to get these results.

Video Link to YouTube video

Link given below is a demonstration of the Matlab function CountCells running in real time.

https://youtu.be/s3XldbGspWg

References

[1] "Is conversion to gray scale a necessary step in Image preprocessing?" [Online]. Available: https://stackoverflow.com/questions/20473352/is-conversion-to-gray-scale-a-necessary-step-in-image-preprocessing. [Accessed: 21-Apr-2018].

[2] "LoG filter," 2001. [Online]. Available: http://academic.mu.edu/phys/matthysd/web226/Lab02.htm. [Accessed: 21-Apr-2018].

[3] Steve Eddins, "Watershed transform question from tech support," 2013. [Online]. Available: https://blogs.mathworks.com/steve/2013/11/19/watershed-

transform-question-from-tech-support/. [Accessed: 21-Apr-2018].

[4] S. Eddins, “The Watershed Transform: Strategies for Image Segmentation,” 2002. [Online]. Available: https://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html. [Accessed: 21-Apr-2018].