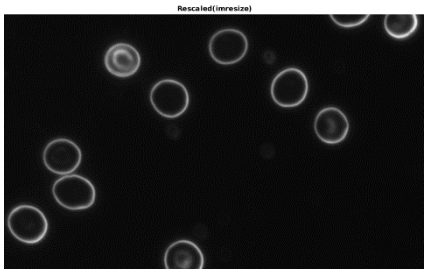
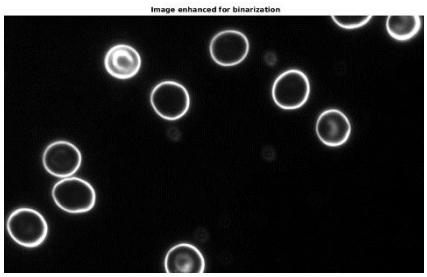
Matthew Atkins
ATK16657290
CMP3108M
University of Lincoln – School of Computer Science
Extension Code: TZPSPSDVLFWTGCF7

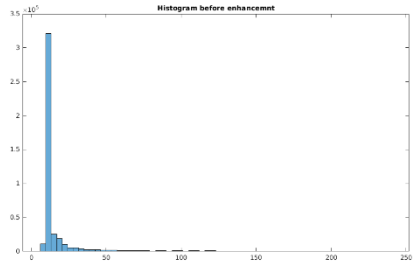
CMP3108M Image Processing

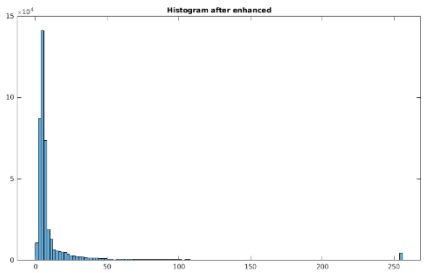
Contents

Matthew Atkins
ATK16657290
CMP3108M
University of Lincoln – School of Computer Science
Extension Code: TZPSPSDVLFWTGCF7









Task 1 – Pre-Processing

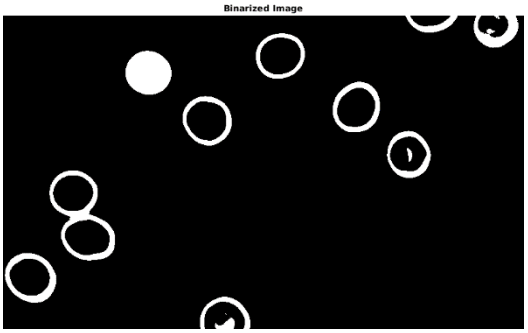
The following images are evidence of the steps taken during the pre-processing stage

'IMG_01' was loaded ('imread()') as a .png file and converted to grayscale using the 'rgb2gray()' function. To resize the image, imresize() was used and the argument [512,NaN] was added to reduce the size of the image while maintaining the aspect ratio.
A histogram of the image before enhancement was produced using the 'histogram()' function. Subsequently, the image intensity values of the image were optimised using the 'imadjust()' function.

Imadjust was implemented as it is able to increase the contrast of an image but saturating the top and bottom 1% image pixel values. This increase in contrast makes the image more optimised for use in processing techniques such as classification, edge detection, object recognition and segmentation.
The 'Histeq()' function was also tested on the dataset, however it produced undesired results.

Another histogram was created after image adjustment using 'histogram()' to compare the difference in the colormap values of the image, this visualisation allows us to prove the image has become more optimised for our use case.
IMG_01 was binarized using 'imbinarize()'. Noise on IMG_01 was fairly low, however after testing with other images on the dataset images IMG_4,7,9 and 15 were particularly noisy.

Task 2 – Edge Detection

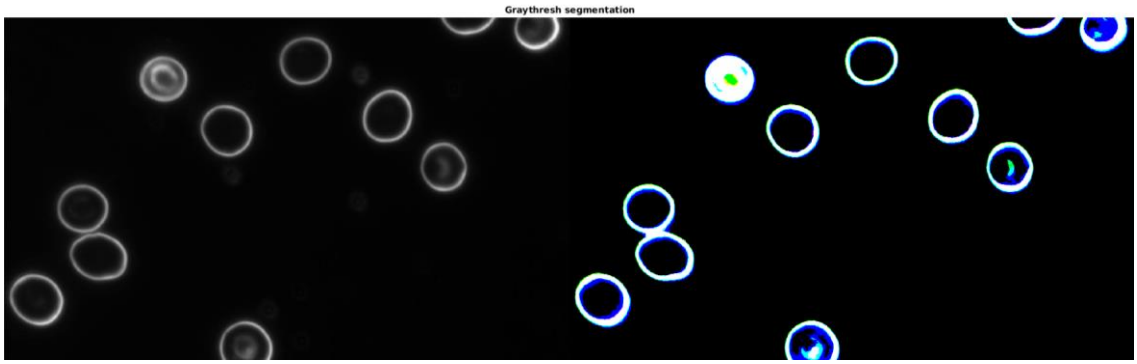


Edge detection was achieved through the utilisation of the 'edge()' function using argument 'Canny'. This implements Canny edge detection onto the binarized image.
Canny provided the most desirable response to unique edges: This is because canny first removes noise from an image using smoothing, Then suppresses pixels that are not at the maximum of the image gradient.
Other methods of edge detection, such as the "Sobel" or "fuzzy logic" methods could have been used, but due to their nature they are unsuitable for this task as they are not designed to detect circular objects, meaning data would almost certainly be lost. (Joshi, 2014)

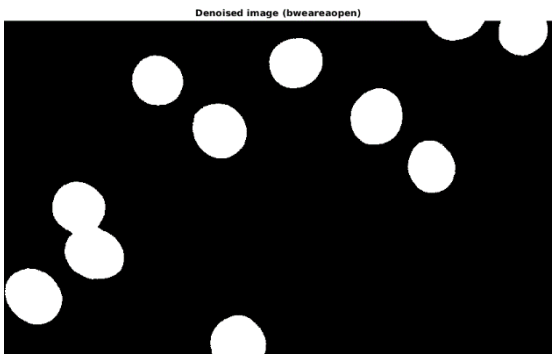
Matthew Atkins
ATK16657290
CMP3108M
University of Lincoln – School of Computer Science
Extension Code: TZPSPSDVLFWTGCF7

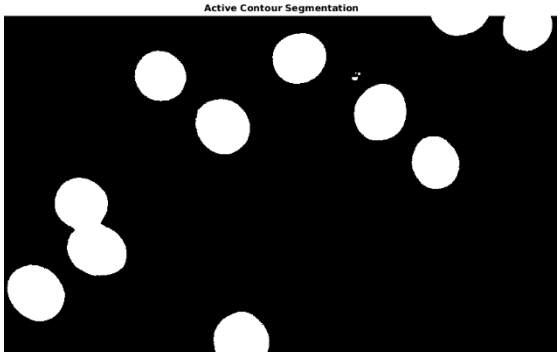
Task 3 – Simple Segmentation

'graythresh()' was used to determine an appropriate threshold value and segment the image. This method was effective. However, noise still remained.
Because of this, the active contour segmentation method was also used to further segment the image using a region growing technique. (Chen,2019)



This removed much more of the noise, and completed many of the circles. As the image had been binarized, 'bwareopen()' was used to remove all object with a pixel area value of under 170. This meant red blood cells and bacteria would stay, but noise would automatically be removed.

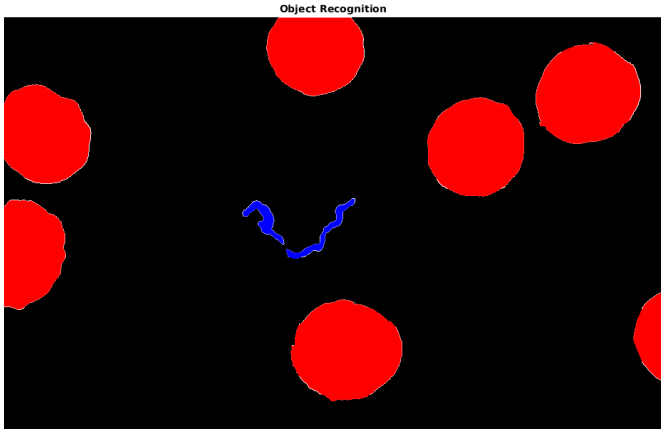




Task 4 – Object Detection

In the initial, three objects must be classified from on another, red blood cells, bacteria cells, and anomalous cells (noise). By using a segmentation approach to remove the noise. We are able to implemented an object detection algorithm to differentiate these two objects.

To do this, I differentiated the objects based on primary key features (shape). Using the permanent logic that red blood cells are **always** round, and bacteria cells are **always** long, I was able to write an object detection algorithm which scanned all object boundaries in an image, determined their **circularity** value, and based upon this, determined whether the object was a red blood cell, or a bacteria.



I loaded the image 'IMG_11.png' using 'imread()', then I reproduced the steps from Task 1 to 4 to pre-process my image

in order to prepare it for binarization, then I post-processed the image using 'bwareaopen()' to remove noise, and implemented my algorithm to differentiate objects based on circularity.

Firstly, I make sure all holes are filled using the imfill() function alongside the argument 'holes'. This fills any images containing holes in them.

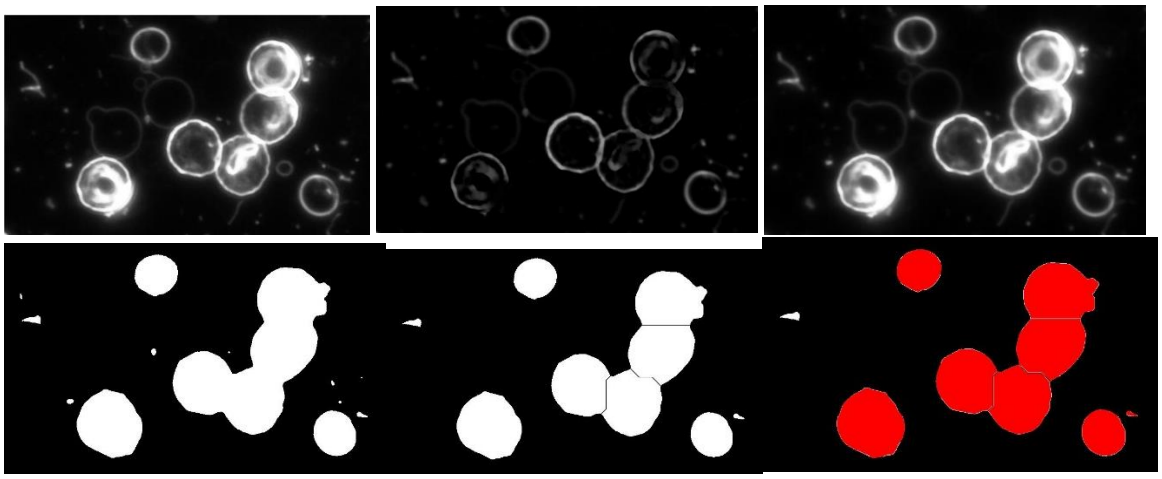
Secondly, I traced the boundaries and returned them to a matrix of objects 'B' and holes 'L'. Then I used a for loop to run through every value in B and calculate the boundaries.
Finally, for each item in B, I calculated an estimate of the object's perimeter and retrieved the area in order to compute the circularity value.

If the circularity value of the object was below the threshold, it would fill the boundary blue, as it was not circular enough to be a red block cell.

If the circularity value of the object was above the threshold, the fill() function would be called to fill the boundary red.

Task 5 – Robust Method

The following images are evidence of the processing stages on 'IMG_05', which demonstrate image pre-processing, edge detection, object separation, and object recognition.



Firstly, a loop was created to run through every file and perform image-processing, this was stored in the function *loopfiles()*.
The goal of the robust method was to implement multiple image processing techniques in order to standardised / normalise the image background, shapes and objects in the image.

The image was resized, maintaining aspect ratio. This was so measurements of all images tested would be of equal proportion.
Conglomerated objects touching a blood cell will always have a longer length than the blood cell, these statistics can be utilised and separate them. A skeleton was generated using the bwmorph() and imerode() functions, and branches were reduced using the spur() function.
Tophat filtering was employed to remove harsh contrasted noise particularly found in images with harsh noise and contrasting.
Holes were filled using imfill, and distorted or 'broken' circular objects were corrected using active contouring segmentation.
Bwareaopen() was used to denoise the image, the watershed method was used to separate overlapping circular red blood cells and conglomerated objects such as bacteria touching blood cells.

Next, the boundaries of each the objects in the image were determined and their circularity metric was calculated. This was compared against a circularity threshold values determined over the entire dataset, which allowed the program to accurately detect a red blood cell against bacteria. The red blood cells were coloured red using the fill() function.

Bibliography

Joshi, M. (2014). *Comparison of Canny edge detector with Sobel and Prewitt edge detector using different image formats.* INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) ETRASCT.

Kothari, J. D. (2021). Medical image segmentation using advanced machine learning algorithms (learning active contour models). *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3794007