Robust Segmentation and Classification of Red Blood Cells and Bacteria in Microscopy Images

Student Name: Marcus Sanghvi

Student Number: 26944375

Module Code: CMP3108-2425

Word Count: 1244

1. Introduction and Objective

This report covers the creation of a robust method for segmentation and classification between red blood cells (RBCs) and bacteria cells (BCs) within medical microscopy images. The algorithm is designed to handle variations in lighting, object shape, and image noise using morphological operations and adaptive thresholding.

2. Methodology (Task 1-5)

2.2 Task 1

I needed to resize the image, previously converted to grayscale, to give it a height of 512. I also had to ensure that the aspect ratio stays intact. To achieve this, I started off by assigning a new height value of 512. I then calculated the aspect ratio of the original image by dividing the width of the image by its height. The new width is then calculated whilst still maintaining the aspect ratio and I have used the round function to ensure that the return value for width is an integer.

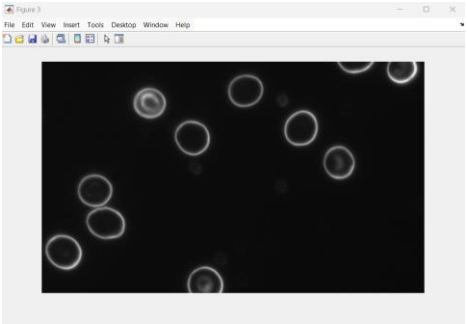

Figure 1. Resized Image Output

For the next step, I need to create a histogram with a bin size of 64. To achieve this step, I use the command “imhist” to create a histogram. It takes two arguments, the target image and the bin size.

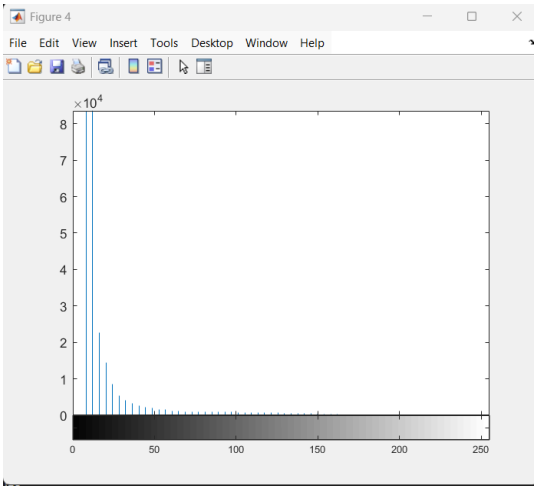

Figure 2. Image Histogram

I then needed to enhance the image using a method of my choice (i.e contrast adjustment). The reason why this is important is that contrast adjustment allows us to adjust an image in such a way that makes it a lot easier to differentiate between certain colours, in this case black and white, as the unprocessed image may be very blurred or low quality.

To adjust the contrast, I created a new variable named “I_enhanced” and then used the function “imadjust” to remap the intensity values of the image which enhances the contrast.

Figure 3. Adjusted Contrast Output

This next step is basic and wants to create another histogram after the contrast adjustment with a bin size of 64. Once again, I use the “imhist” function to create this histogram.

I needed to binarise the image. The goal is to have white pixels represent areas of interest within the image such as cells or bacteria and the black pixels represent the background of the image.

To do this I used a threshold where any pixels above a certain value are given a white colour whereas any pixels below the set value are given a black colour. To work out what value I needed to use for the threshold I used the function “graythresh” as this function computes an appropriate threshold value. “imbinarize” completes the binarization and takes in the image and threshold as arguments.

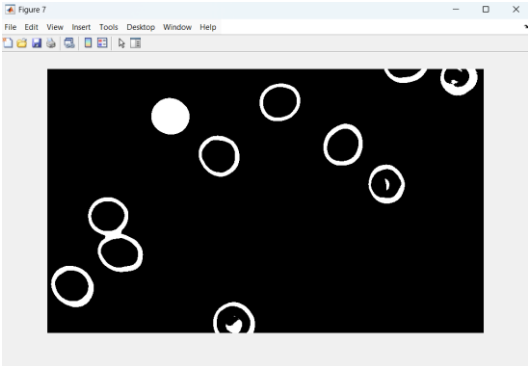

Figure 4. Binarized Image

2.3 Task 2 – Edge Detection

The edge detection algorithm I have chosen to use for this task is the ‘Canny’ edge detection algorithm. The reason for this choice of algorithm is that it is considered one of the most tested, reliable and effective algorithms for detecting edges in image processing.

It consists of several steps to achieve the end goal of detecting and showing edges. First of all, it uses a gaussian filter to reduce the amount of noise in the image. It then calculates the intensity of the gradient of the image and then uses two thresholds to differentiate between weak and strong edges.

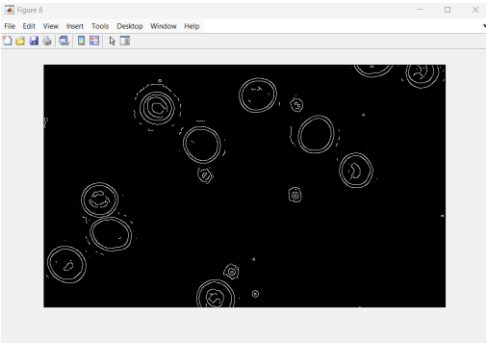

Figure 5. Edge Detection Output

2.4 Task 3 – Simple Segmentation

The objective for task 3 is to display the blood cells as white circles on a black background by the process of segmentation. This allows doctors to clearly identify and analyse these blood cells.

The first thing that I did was clean up the binirised image from task 1. I created a structuring element in the shape of a disk with a radius of 2 pixels. This structure will be used to allow for small objects such as noise to be removed from the image during cleanup.

After the image has been cleaned, I then use “imfill” to fill in the circles.

A black image is then initialised with the same size as the previous image however all the values are set to false. This image will be used later to draw the blood cells.

Then I moved on to labelling connected components. The function “bwconncomp” finds all white regions within the image (binirised image makes this process a lot easier) and then the function “regionprops” returns properties that are useful for drawing these circles.

A for loop is then used to iterate over each of the identified blood cells. Within the loop it calculates the radius of the cell, creates coordanites for the image dimensions, creates a circular mask at the center of the cell and then sets the pixels in the cell to true to give them the white colour.

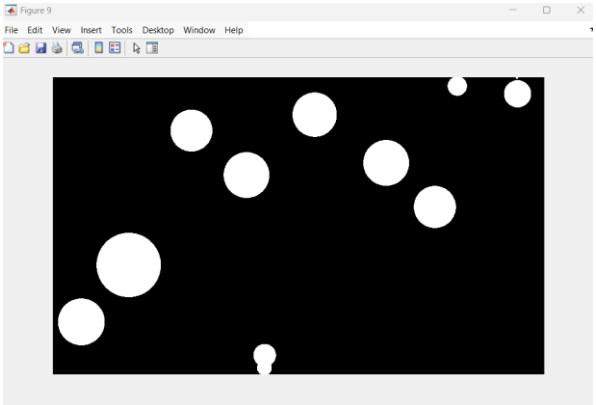

Figure 6. Segmentation Output

2.5 Task 4 – Object Recognition

After segmenting the image, I now need to create an overlay that clearly labels different types of cells. To begin with I loaded and pre-processed the image using techniques such as binarization, grayscale and image feature extraction. I then created empty masks and then classified each cell based on set threshold values.

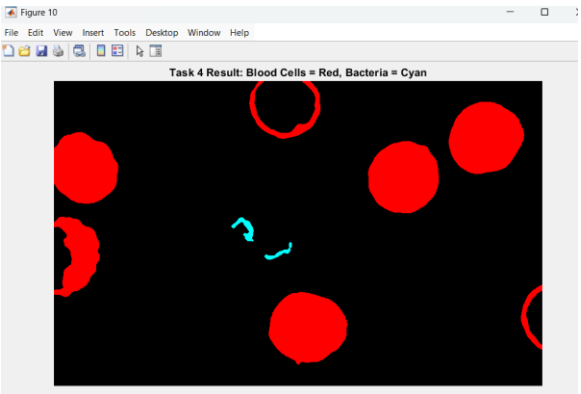

Figure 7. Task 4 Result

2.6 Task 5 – Robust Method

For this task I created a robust method for segmenting and classifying cells within a large dataset of images. To begin with I loaded the image, converted it to grayscale and applied a 5x5 median filter. This removes noise without blurring any edges. A morphological top hat was then applied to yield a “flat” image that will make thresholding work a lot better.

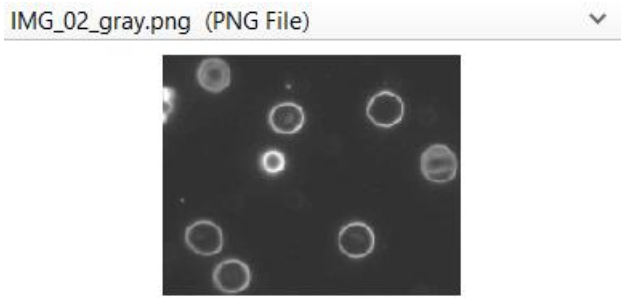

Figure 7. Output after grayscale

I then used Otsu's method to find the optimal global threshold automatically from the histogram. Then all holes were filled into a solid blob making them easier to read and see. I then attempted to use 'bwareaopen' to try and remove any small specs that may interfere.

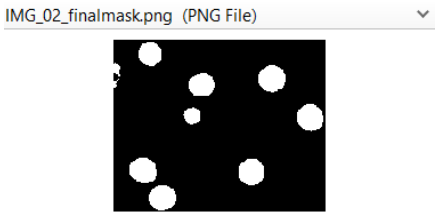

Figure 8. Output after Otsu's Method

After this step I then measured the features of the shapes such as area, circularity, eccentricity, and solidity. Using this data I was then able to compare it to set parameters for the acceptable ranges for each of the features. From this I was able to classify between RBCs and BCs.

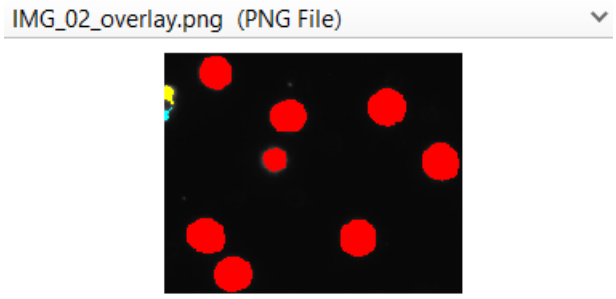

Figure 9. Final Output

3. Evaluation (Task 6)

In this task I have compared my automatic segmentations against the provided assignment ground-truth labels (1 = RBC, 2 = bacteria,3 = background) and report three standard metrics—Dice score, precision and recall—computed on a per-pixel basis and then averaged over all 15 images.

The first step I have taken is to read all GT images and convert them into a single channel label image. From there I have extracted the predicted label overlay masks to isolate the exact pixels my algorithm has classified as BCs or RBCs.

For each image processed each false positive, true positives and false negatives are counted and then using this I can compute the metrics needed for the comparison.

```
--- Final Evaluation over 15 images ---

RBC Performance:
Dice Score    = 0.667
Precision     = 0.868
Recall        = 0.619

Bacteria Performance:
Dice Score    = 0.112
Precision     = 0.143
Recall        = 0.108
```

Figure 10. Metric Data Output