**BN2202**

# GROUP PROJECT REPORT

**Introduction to Biotransport**

Project Title: To propose a quantification method for red blood cell aggregation, based on methods acquired from literature.

**Group A03**

Chia Ji Hong, Augustine (A0183443E)

Alvin Tay Zhi Jiang (A0183486R)

Alexandra Louise Wee Rei-Min (A0190693W)

Joshua Liu Song Jae (A0183512L)

Jerome Leong (A0189863H)

**Semester 1**

**AY 2019/2020**

**National University of Singapore**

## Table of Contents

# 1. Introduction

## 1.1 Background of Red Blood Cell Aggregation

Blood is a non-Newtonian fluid containing erythrocytes, also known as Red Blood Cells (RBC), leukocytes, platelets, proteins and other components dispersed throughout the blood. RBCs aggregates primarily due to various rheological factors such as hematocrit levels, concentration of fibrinogen and blood plasma viscosity [1], and external factors such as pH levels and shear rates [2]. The higher concentrations of fibrinogen [3] and blood plasma viscosity [4], the higher the extent of RBC aggregation. While at lower shear rates up until stasis, RBCs are shown to aggregate more in two and three dimensional structures [5]. Researchers have tried to find a favourable quantification of RBC aggregation, one method being Erythrocyte Sedimentation Rate (ESR) which measures the rate of RBCs settling at the bottom of a test-tube and is associated with higher aggregation [6]. The Myrenne Aggregometer is another method, which gives indices at stasis and low shear [7] and shows a higher tendency to aggregate for a higher index [8]. Fractal Analysis displays the relation between the change in fractal dimension related to the variation of aggregation shape [9]. The aim of this study is to propose a quantification method for RBC aggregation based on comparisons and evaluations across various studies.

# 2. Materials and Methods

## 2.1 Sample Collection

Two blood samples, A and B, each of hematocrit (HCT) levels of 10%, underwent 'washing' multiple times, which involved spinning down the blood sample followed by discarding the supernatant. The product was resuspended in two different mediums, the first allowing for RBC aggregation while the second discourages aggregation. The final products were then carefully applied onto the glass slide, and not smeared. Samples A and B were viewed under a 20x and 40x objective lens of an inverted microscope. A green light filter was used to increase the colour contrast in the captured images, as RBCs absorb the most light in the blue-green range. RBC images were captured using an image acquisition software (Refer to Appendix A) and post-processing was then performed on the images, counting the number of RBCs in each image.

## 2.2 Post processing methods

We designed a computerized image analysis algorithm for post-processing, to quantify RBC aggregation in a standardised method, which produces a more consistent and accurate result, unaffected by human error. Three methods were considered:

Method 1 used Python with the gaussian filter function which reduces noise in an image. For each magnification, we set a threshold for the colour contrast between the background and RBCs. The algorithm then counts the mean number of pixels in RBCs (Refer to Appendix). Method 2 used Python with the circular hough transform function to count the number of circular figures in an image. The threshold level for the circles' radii was calibrated per magnification. Each circle identified is understood to be one RBC. Method 3 used Matlab with the same algorithm as method 2. Results of the 3 methods were cross-checked against manually counted RBC population and method 2 had the highest percentage accuracy at 95.8%, and the lowest margin of error (Refer to Appendix). Hence, method 2 was used for further processing.

Subsequently, our algorithm also identifies and isolates irregular shapes in images, which we recognised as aggregated RBCs. Using method 2, the number of circles within an irregular shape is computed and identified as RBCs involved in aggregation. Using equation 1 (Refer to Appendix), we calculated the percentage of RBCs involved in aggregation, recording the data in Table 2 (Refer to Appendix).

## 3. Results

From the images, we observed that more RBCs were involved in aggregation in sample B compared to sample A. RBCs in A were more evenly dispersed, while RBCs in B were seen to clump together. Our observations were further supported by our data. Sample B consistently had a higher percentage of RBCs involved in aggregation than in A.

Studies showed a direct correlation between high blood pressure and higher blood viscosity levels in hypertensive subjects [10]. Higher fibrinogen concentrations and hematocrit values were responsible for their elevated blood viscosity levels and hence an increase in RBC aggregability. As such, since sample B has more RBC involved in aggregation, sample B is more likely to be from a hypertensive patient as compared to sample A.

## 4. Limitations

<u>Freshness of blood samples</u>

The blood samples used were prepared about 2 weeks prior to image acquisition. RBC in blood samples that are kept for some time become more rigid and aggregate less than freshly prepared blood samples. As such the percentage of aggregation would be lower than expected and hence give inaccurate data and calculations.

Blood samples prepared without smearing in glass slides

During preparation, both blood samples were not smeared on the glass slides to allow more aggregation of blood to be observed. However, this caused RBCs to overlap each other which was captured in the images. Hence, overlapped RBCs might not be counted by the algorithm, resulting in an inaccurate mean RBC population.

Lack of blood samples

During image acquisition, multiple pictures were taken at different locations of the blood sample. The RBC population for each photo was averaged across the sample size to obtain a sample mean. However, only 5 photos were taken at each magnification for each sample. Due to the small sample size, the RBC sample mean had a higher margin of error when estimating the true population mean in the blood samples.

Evaporation of blood sample boundary

The observation of the blood samples was conducted in an air-conditioned room with lower humidity, causing faster evaporation at the outer boundaries of the blood samples. The outer boundary of the blood sample was observed to move towards the centre of the glass slide, affecting the spread of RBC in the blood samples, and causing the mean RBC population calculated to be inaccurate.

## 5. Improvements to methodology

With a larger sample size, the margin of error for counting the mean RBC population will be smaller hence accuracy of counting RBCs will be higher.

Although method 2 had the highest accuracy when counting RBCs, the algorithm still has its limitations. Our method often did not fully account for all rouleaux and RBCs that are oriented sideways. Instead of obtaining 2D images, a 3D perspective of the sample can be obtained and processed to allow all rouleaux to be observed and accounted for.

More can be done to isolate the RBCs from foreign particles or echinocyte by overlaying the blood sample in silicone oil [11]. We have also included an algorithm (Refer to Appendix B) to isolate and count these echinocytes and foreign particles.

## References

[1] M. Rabai, "In vitro hemorheological studies focusing on erythrocyte deformability and aggregation," *University of Pecs Hungary,* pp. 1-68, 2012.

[2] M. A. Elblbesy and M. E. Moustafa, "The Impact of Biophysical Properties of Erythrocytes," *International Journal of Biomedical Science,* vol. 13, no. 2, pp. 113-118, 2017.

[3] D. Lominadze and W. L. Dean, "Involvement of fibrinogen specific binding in erythrocyte aggregation," *FEBS Letters,* pp. 41-44, 2002.

[4] R. Mehri, C. Mavriplis and M. Fenech, "Red blood cell aggregates and their effect on non-Newtonian blood viscosity at low hematocrit in a two-fluid low shear rate microfluidic system," *PLos ONE,* vol. 13, no. 7, 2018.

[5] O. K. Baskurt and H. J. Meiselman, "Erythrocyte aggregation: Basic aspects and clinical importance," *Clinical Hemorheology and Microcirculation,* vol. 53, no. 1, pp. 23-37, 2013.

[6] D. A., A. Narasimha, H. K. M.L and A. K. D.R., "Significance of Erythrocyte Aggregation Test In Acute Myocardial Infarction," *International Journal of Basic and Applied Medical Sciences,* vol. 4, no. 1, pp. 45-51, 2014.

[7] B. K. Lee, T. Alexy, R. B. Wenby and H. J. Meiselman, "Red blood cell aggregation quantitated via Myrenne aggregometer and yield shear stress," *Biorheology,* vol. 44, no. 1, pp. 29-35, 2007.

[8] M. Ju, B. Namgung and S. Kim, "Application of Refutas Model to Estimate Erythrocyte Viscosity in a Dextran Solution," *Macromolecular Research,* vol. 20, no. 8, pp. 887-890, 2012.

[9] W. Wei, J. Cai, X. Hu, Q. Han, S. Liu and Y. Zhou, "Fractal analysis of the effect of particle aggregation distribution on thermal conductivity of nanofluids (Revised)," *Phys. Lett. A,* vol. 380, no. 37, pp. 2953-2956, 2016.

[10] R. L. Letcher, S. Chien, T. G. Pickering and J. H. Laragh, "Elevated blood viscosity in patients with borderline essential hypertension.," *American Heart Association,* vol. 5, no. 5, pp. 757-762, 1983.

[11] E. Horn, "Avoiding Evaporation," 29 March 2012. [Online]. Available: https://ibidi.com/img/cms/support/AN/AN12_Avoiding_evaporation.pdf. [Accessed 16 September 2019].

## Appendix: Equations, Parameters and Source of Data

### A. Images of Erythrocytes Under 20X and 40X Magnification

Raw image files were named based on the sample name, followed by the magnification then then picture number (e.g. first image taken of Sample A at 20X magnification will be called A_20X_1.bmp, second image taken will be A_20X_2.bmp and so on.)

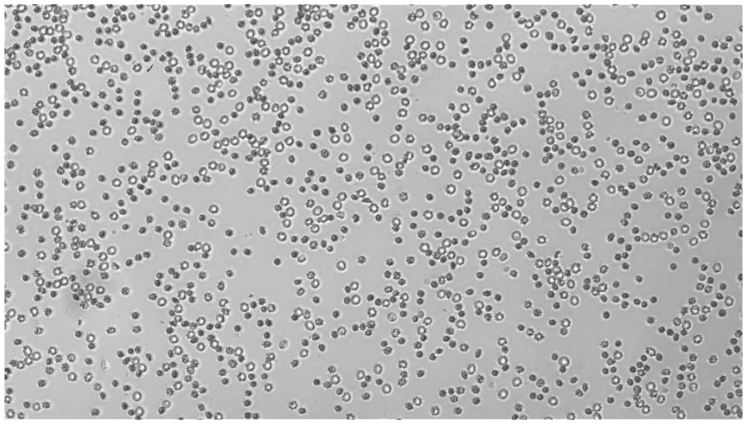


***Figure 1.*** Example of Blood of Sample A at 20X magnification, picture 1 (Filename "A_20X_1.bmp").

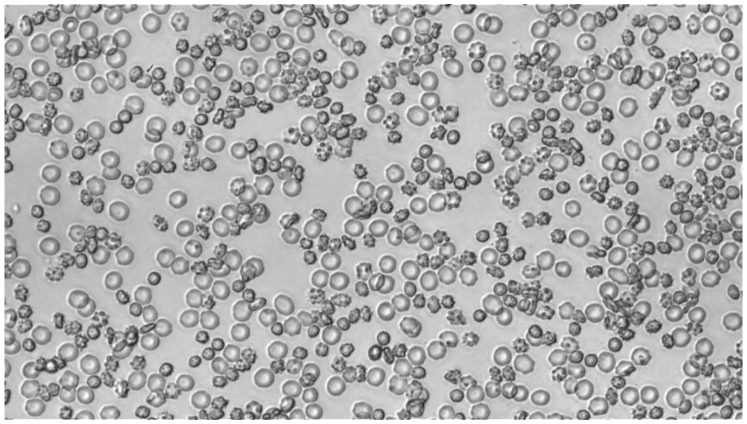


***Figure 2.*** Example of Blood of Sample A at 40X magnification, picture 3 (Filename "A_40X_3.bmp").

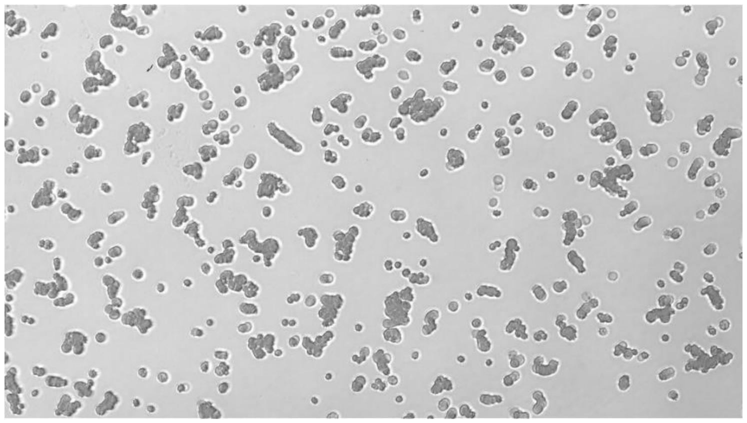


***Figure 3.*** Example of Blood of Sample B at 20X magnification, picture 2 (Filename “B_20X_2.bmp”).

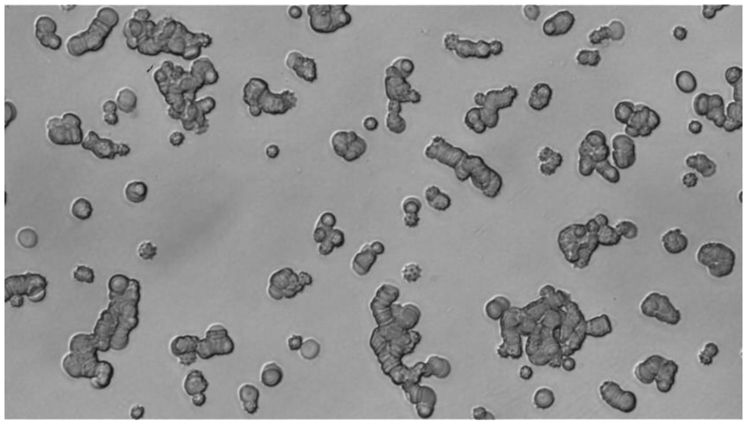
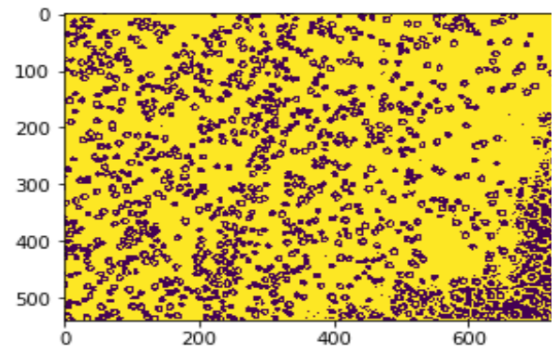


***Figure 4.*** Example of Blood of Sample B at 40X magnification, picture 3 (Filename “B_40X_3.bmp”).

**Sample A at 20X and 40X magnification**



```
Found 1138 RBC.
```

***Figure 5.*** Example of processed image using method 1 (Gaussian Filter) with sample A at 20X magnification, using image with filename “A_20X_1.bmp”).

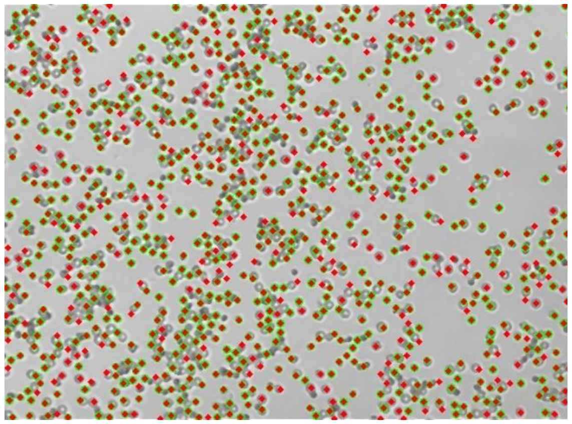


***Figure 6.*** Example of processed image using method 2 (Circular Hough transformation on Python) with sample A at 20X magnification, using image with filename “A_20X_1.bmp”).

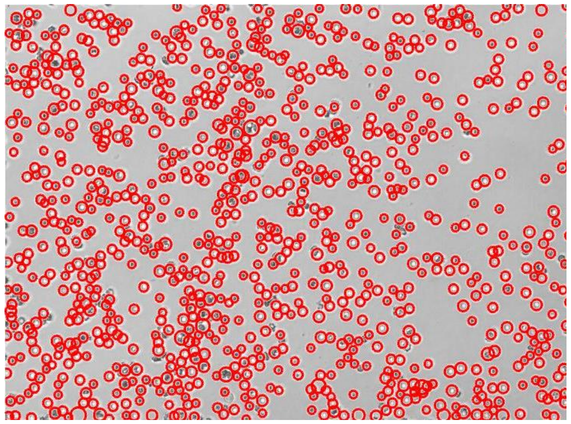
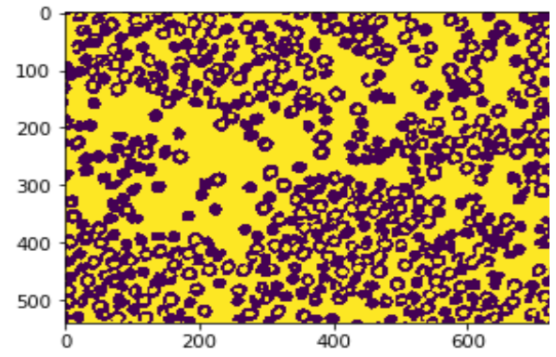


***Figure 7.*** Example of processed image using method 2 (Circular Hough transformation on MatLab) with sample A at 20X magnification, using image with filename “A_20X_1.bmp”).



```
Found 511 RBC.
```

***Figure 8.*** Example of processed image using method 1 (Gaussian Filter) with sample A at 40X magnification, using image with filename “A_40X_1.bmp”).

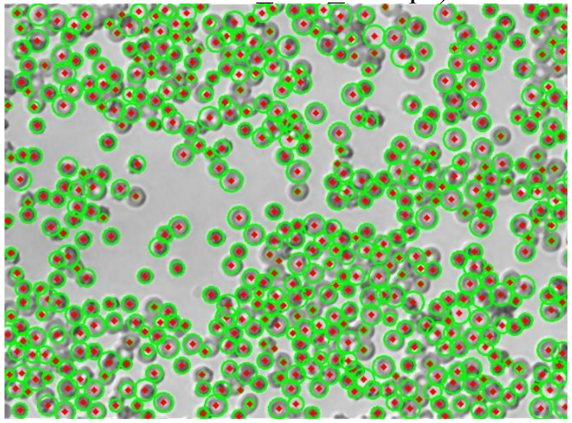


***Figure 9.*** Example of processed image using method 2 (Circular Hough transformation on Python) with sample A at 40X magnification, using image with filename “A_40X_1.bmp”).

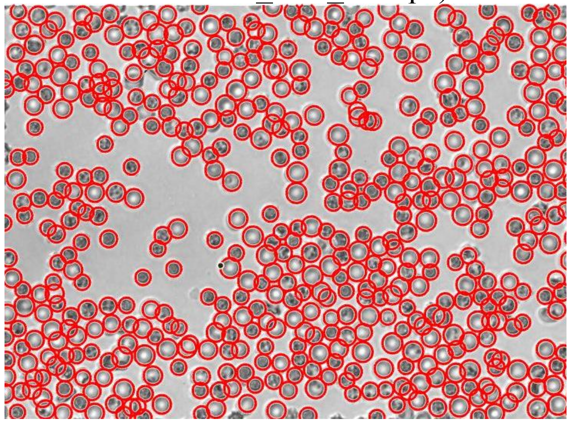
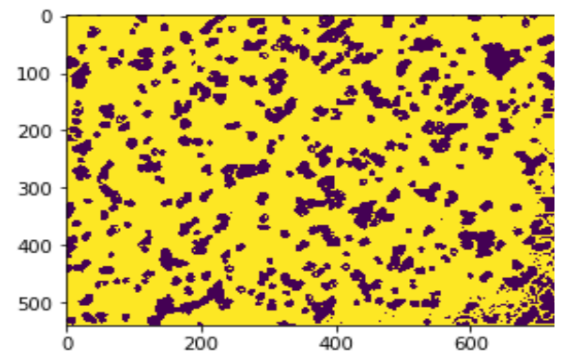


***Figure 10.*** Example of processed image using method 2 (Circular Hough transformation on MatLab) with sample A at 40X magnification, using image with filename “A_40X_1.bmp”).

**<u>Sample B at 20X and 40X magnification</u>**



```
Found 334 RBC.
```

***Figure 11.*** Example of processed image using method 1 (Gaussian Filter) with sample B at 20X magnification, using image with filename "B_20X_1.bmp").

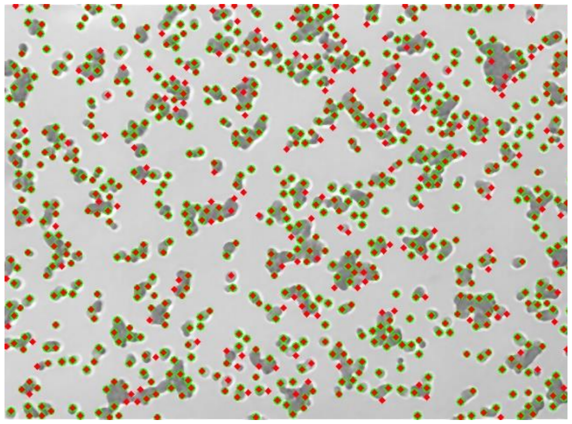


***Figure 12.*** Example of processed image using method 2 (Circular Hough transformation on Python) with sample B at 20X magnification, using image with filename "B_20X_1.bmp").

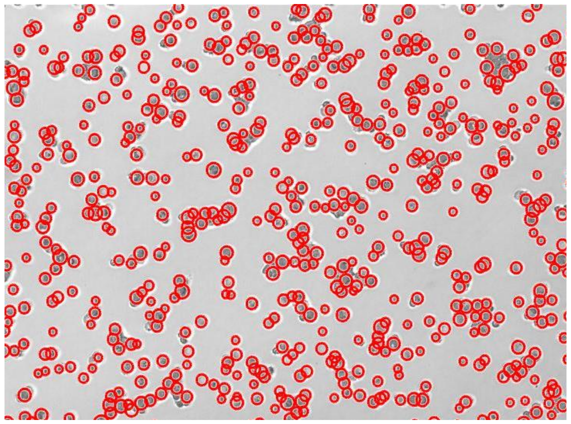
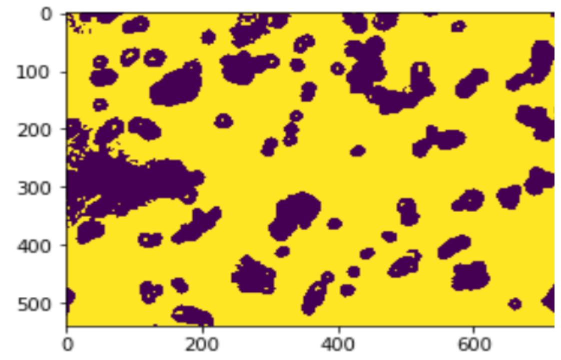


***Figure 13.*** Example of processed image using method 2 (Circular Hough transformation on MatLab) with sample B at 20X magnification, using image with filename "B_20X_1.bmp").



```
Found 102 RBC.
```

***Figure 14.*** Example of processed image using method 1 (Gaussian Filter) with sample B at 40X magnification, using image with filename "B_40X_1.bmp").

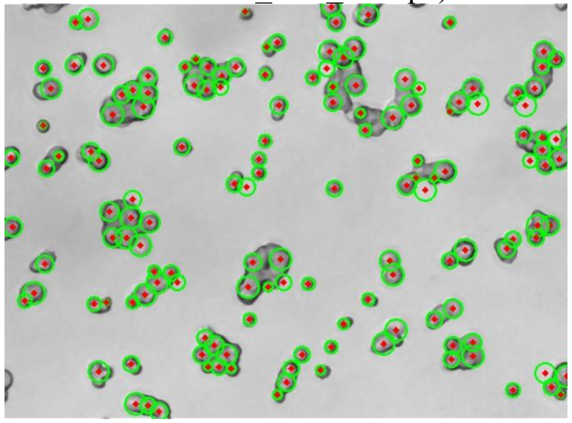


***Figure 15.*** Example of processed image using method 2 (Circular Hough transformation on Python) with sample B at 40X magnification, using image with filename "B_40X_1.bmp").

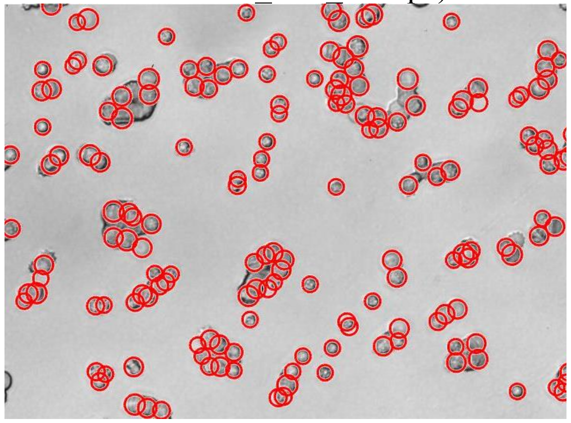


***Figure 16.*** Example of processed image using method 2 (Circular Hough transformation on MatLab) with sample B at 40X magnification, using image with filename "B_40X_1.bmp").

**<u>Sample A and B with aggregated cells being isolated.</u>**

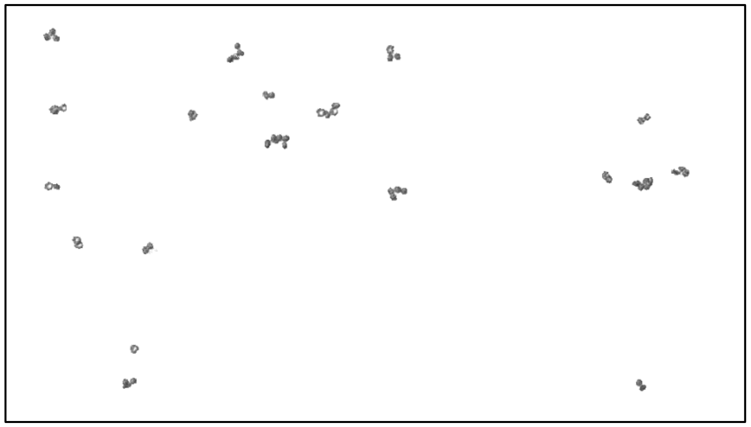


***Figure 17.*** Example of post-processed image after using the aggregated cells isolation method with Sample A at 20X magnification. ("A_20X_1.bmp")

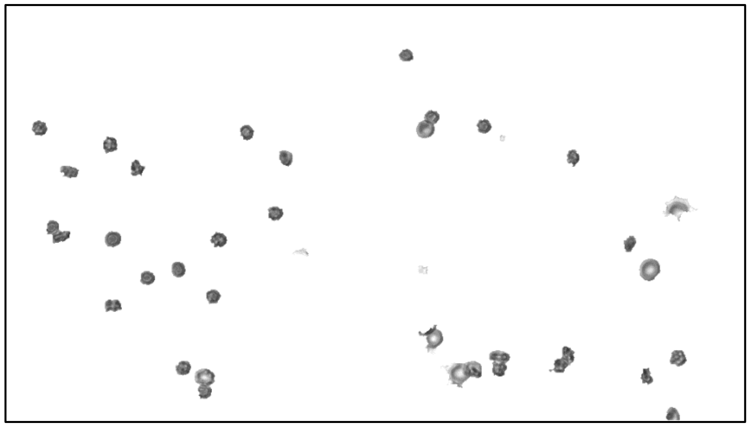


***Figure 18.*** Example of post-processed image after using the aggregated cells isolation method with Sample A at 40X magnification. ("A_40X_1.bmp")

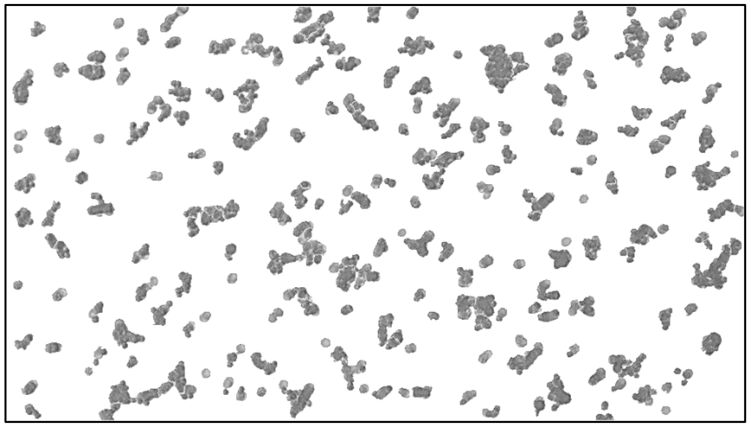


***Figure 19.*** Example of post-processed image after using the aggregated cells isolation method with Sample B at 20X magnification. (“B_20X_1.bmp”)

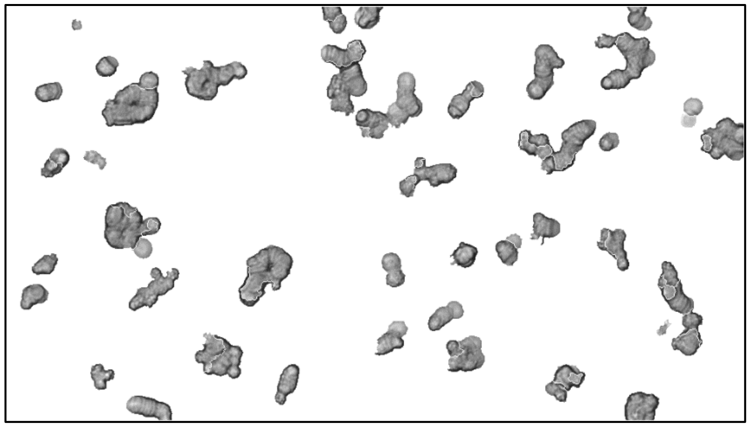


***Figure 20.*** Example of post-processed image after using the aggregated cells isolation method with Sample B at 40X magnification. (“B_40X_1.bmp”)

## B. Table of Comparisons

**Table 1.** Raw data for RBC count for each Method.

| | | **RBC Count** | | |
|---|---|---|---|---|
| **Sample & Magnification Type** | **Image Number** | **Method 1 Count (Gaussian Filter)** | **Method 2 Count (Hough Transform)** | **Method 3 Count (Hough Transform)** |
| Sample A, 20X | 1 | 1189 | 1138 | 1294 |
| | 2 | 1817 | 1434 | 1768 |
| | 3 | 1618 | 949 | 1667 |
| | 4 | 2196 | 1805 | 1990 |
| | 5 | 1210 | 872 | 1149 |
| Sample A, 40X | 1 | 715 | 511 | 757 |
| | 2 | 493 | 448 | 483 |
| | 3 | 671 | 627 | 683 |
| | 4 | 883 | 948 | 951 |
| | 5 | 628 | 608 | 627 |
| Sample B, 20X | 1 | 735 | 334 | 1039 |
| | 2 | 541 | 427 | 776 |
| | 3 | 752 | 641 | 950 |
| | 4 | 805 | 553 | 1014 |

| | | | | |
|---|---|---|---|---|
| | 5 | 676 | 666 | 853 |
| | | | | |
| Sample B, 40X | 1 | 222 | 102 | 193 |
| | 2 | 214 | 103 | 191 |
| | 3 | 248 | 84 | 182 |
| | 4 | 216 | 106 | 180 |
| | 5 | 155 | 119 | 142 |

**Table 2.** RBC count for each Method (95%CI).

| | **RBC Count**[1] | | |
|---|---|---|---|
| **Sample & Magnification Type** | **Method 1 (Gaussian Filter)** | **Method 2 (Hough Transform)** | **Method 3 (Hough Transform)** |
| **A_20X RBC Count (95% CI)** | 1606 [1233.221088 - 1978.778912] | 1239.6 [903.7103218 - 1575.489678] | 1573.6 [1270.475624 - 1876.724376] |
| **A_40X RBC Count (95% CI)** | 678 [553.9086246 - 802.0913754] | 628.4 [459.288683 - 797.511317] | 700.2 [549.0404084 - 851.3595916] |
| **A_20X RBC Count (95% CI)** | 701.8 [613.2795776 - 790.3204224] | 524.2 [400.0816942 - 648.3183058] | 926.4 [829.4669521 - 1023.333048] |
| **B_40X RBC Count (95% CI)** | 211 [181.0818985 - 240.9181015] | 102.8 [91.82750056 - 113.7724994] | 177.6 [159.4808627 - 195.7191373] |

[1] 95% Confidence interval was calculated using the following formula: $\bar{x} \pm z\frac{s}{\sqrt{n}}$ where $\bar{x}$ is the sample mean, $z$ is the confidence coefficient (fixed at 1.96 in this case, since we want a 95% confidence interval), $s$ is the standard deviation, and $n$ is the sample size (fixed at 5 for each category since we only took 5 images of each sample size of a magnification type).

**Table 3.** Average Accuracy of total number of cells counted for each Computational Method compared to the Manual Counting Method.

| | **Method 1 (Gaussian Filter)** | **Method 2 (Hough transform in Python)** | **Method 3 (Hough transform in MatLab)** | **Manual Counting Method** |
|---|---|---|---|---|
| **Mean Number of Cells for counted slides** | 851.1666667 | 1108.666667 | 1083.833333 | 1157.833333 |
| **Percentage Accuracy when Compared to Manual Counting** | 73.51374694% | 95.75356269% | 93.60875198% | N.A. |

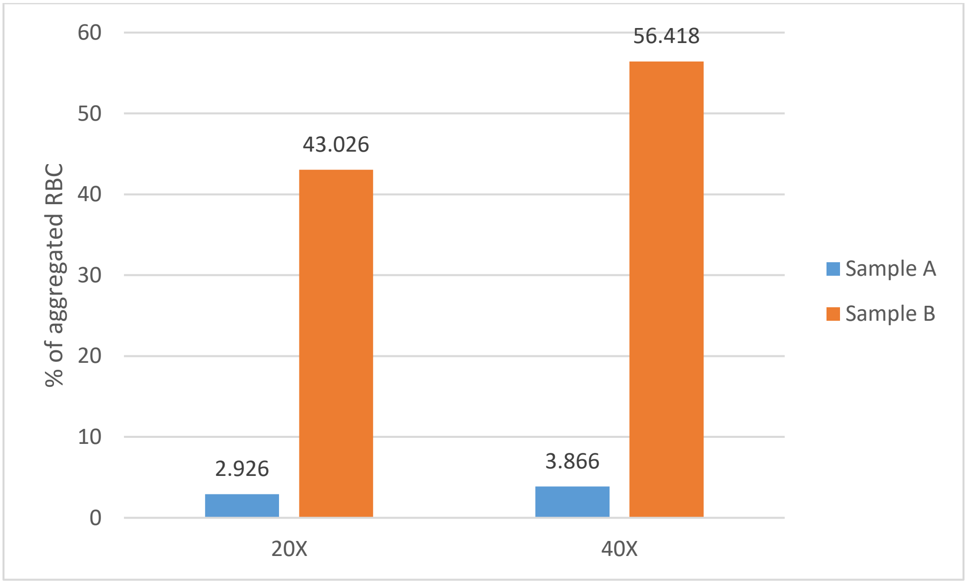
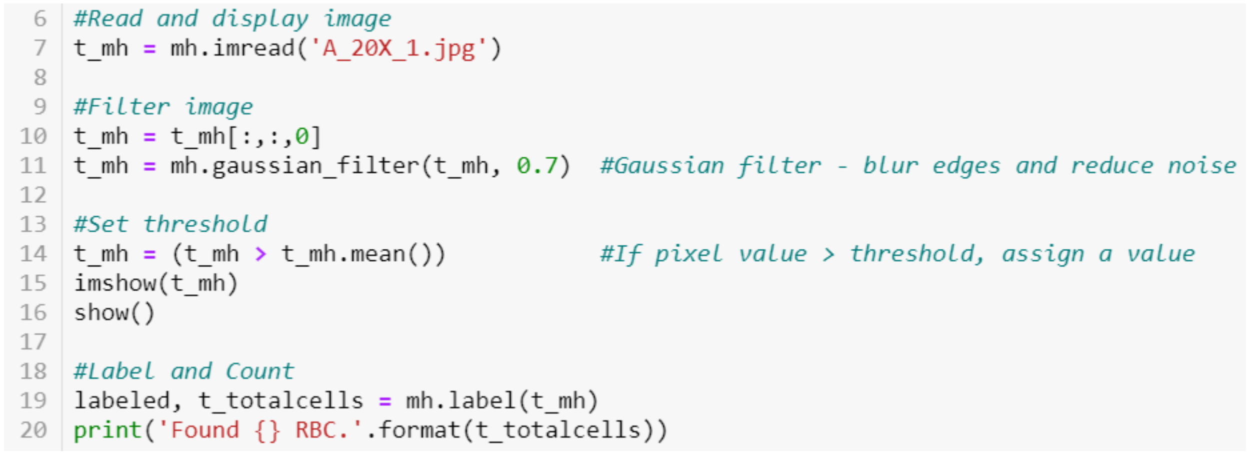


***Figure 21.*** Average percentage of aggregated RBCs in sample A and sample B.

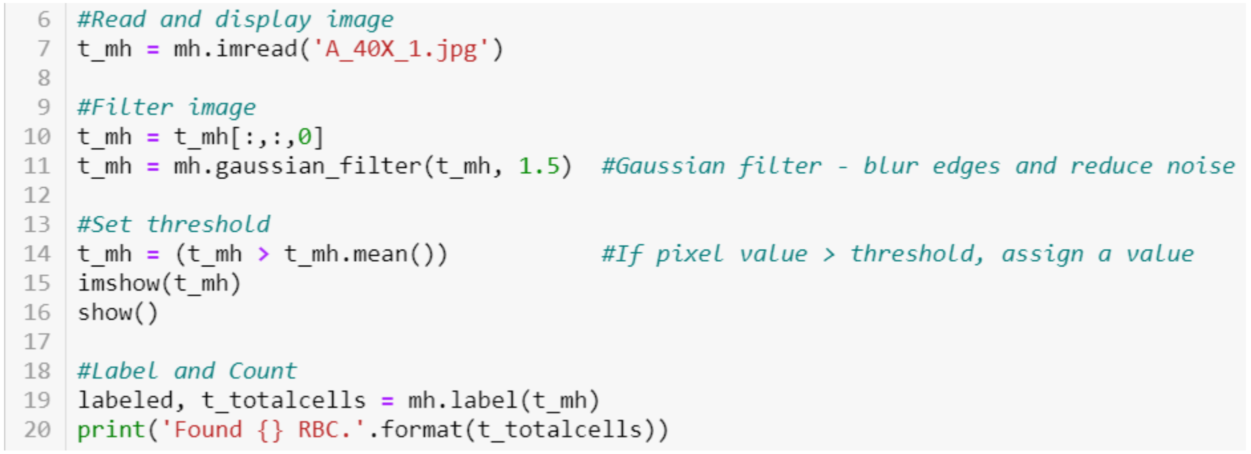
## C. Matlab and Python Source Codes

**Method 1 (Gaussian Filter) Source Code:**



```python
#Read and display image
t_mh = mh.imread('A_20X_1.jpg')

#Filter image
t_mh = t_mh[:,:,0]
t_mh = mh.gaussian_filter(t_mh, 0.7)  #Gaussian filter - blur edges and reduce noise

#Set threshold
t_mh = (t_mh > t_mh.mean())           #If pixel value > threshold, assign a value
imshow(t_mh)
show()

#Label and Count
labeled, t_totalcells = mh.label(t_mh)
print('Found {} RBC.'.format(t_totalcells))
```

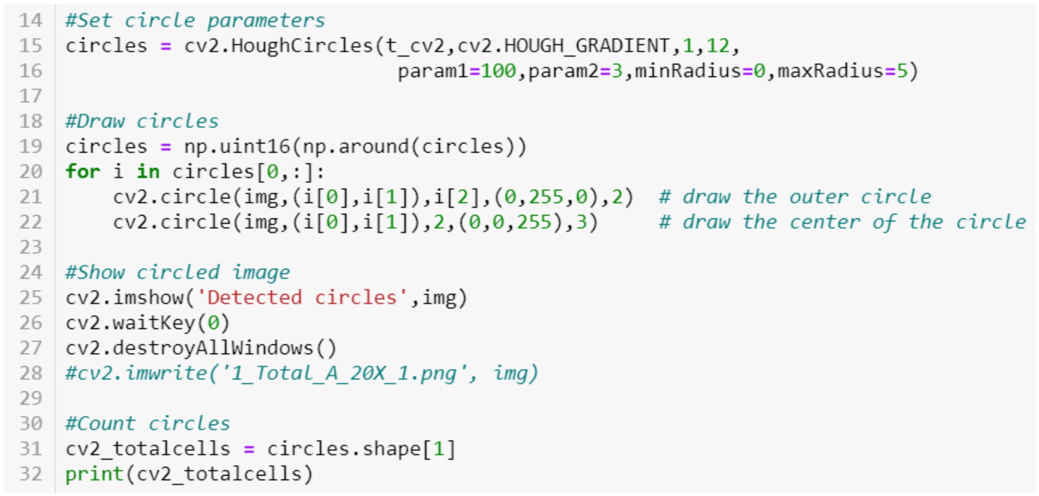
***Figure 22.*** Source Code for Method 1 (Gaussian Filter) with threshold levels set for <u>20X</u> magnification.



```python
#Read and display image
t_mh = mh.imread('A_40X_1.jpg')

#Filter image
t_mh = t_mh[:,:,0]
t_mh = mh.gaussian_filter(t_mh, 1.5)  #Gaussian filter - blur edges and reduce noise

#Set threshold
t_mh = (t_mh > t_mh.mean())           #If pixel value > threshold, assign a value
imshow(t_mh)
show()

#Label and Count
labeled, t_totalcells = mh.label(t_mh)
print('Found {} RBC.'.format(t_totalcells))
```

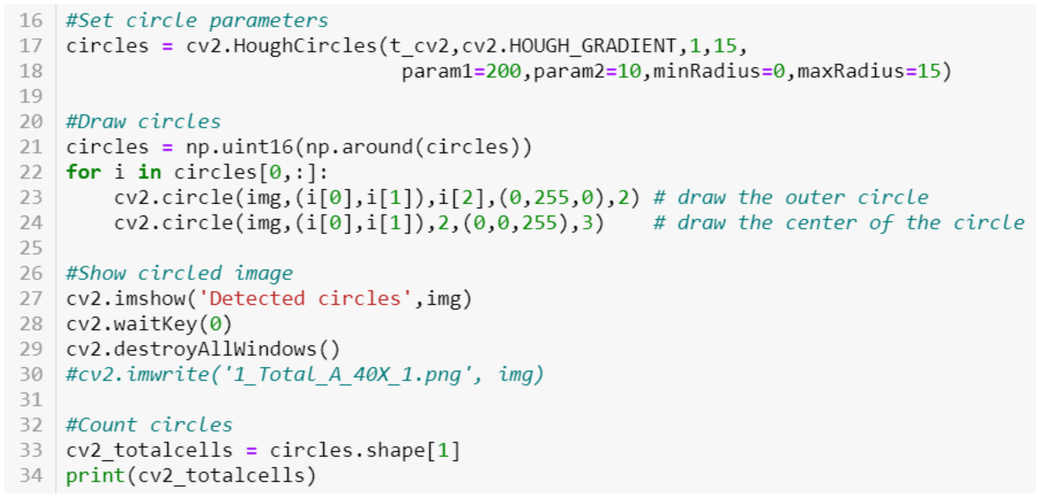
***Figure 23.*** Source Code for Method 1 (Gaussian Filter) with threshold levels set for <u>40X</u> magnification.

**<u>Method 2 (Hough Transform using Python) Source Code:</u>**



```python
#Set circle parameters
circles = cv2.HoughCircles(t_cv2,cv2.HOUGH_GRADIENT,1,12,
                            param1=100,param2=3,minRadius=0,maxRadius=5)

#Draw circles
circles = np.uint16(np.around(circles))
for i in circles[0,:]:
    cv2.circle(img,(i[0],i[1]),i[2],(0,255,0),2)  # draw the outer circle
    cv2.circle(img,(i[0],i[1]),2,(0,0,255),3)       # draw the center of the circle

#Show circled image
cv2.imshow('Detected circles',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
#cv2.imwrite('1_Total_A_20X_1.png', img)

#Count circles
cv2_totalcells = circles.shape[1]
print(cv2_totalcells)
```

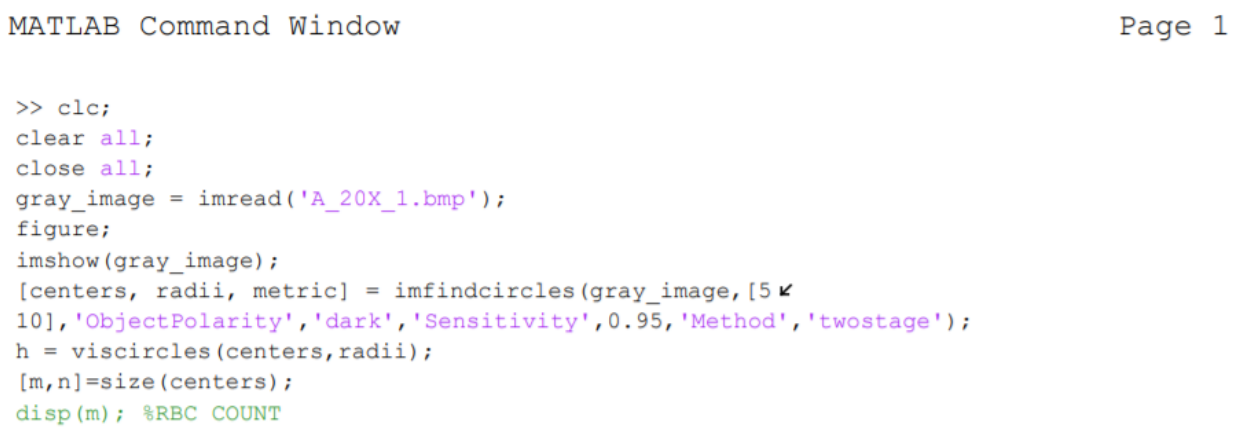
***Figure 24.*** Source Code for Method 2 (Hough Transform using Python) with threshold levels set for <u>20X</u> magnification.



```python
#Set circle parameters
circles = cv2.HoughCircles(t_cv2,cv2.HOUGH_GRADIENT,1,15,
                            param1=200,param2=10,minRadius=0,maxRadius=15)

#Draw circles
circles = np.uint16(np.around(circles))
for i in circles[0,:]:
    cv2.circle(img,(i[0],i[1]),i[2],(0,255,0),2) # draw the outer circle
    cv2.circle(img,(i[0],i[1]),2,(0,0,255),3)      # draw the center of the circle

#Show circled image
cv2.imshow('Detected circles',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
#cv2.imwrite('1_Total_A_40X_1.png', img)

#Count circles
cv2_totalcells = circles.shape[1]
print(cv2_totalcells)
```

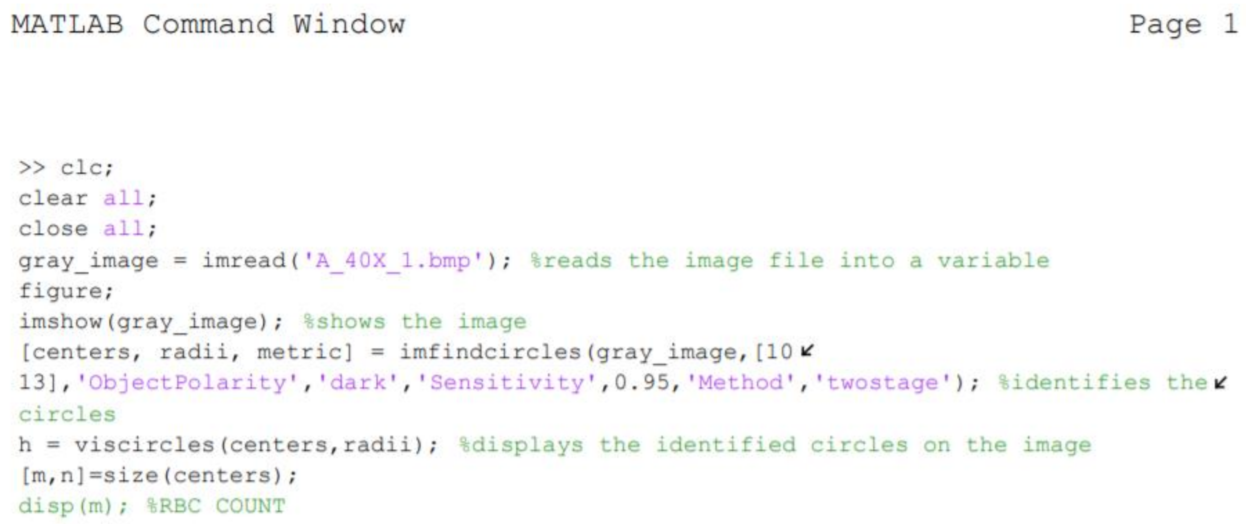
***Figure 25.*** Source Code for Method 2 (Hough Transform using Python) with threshold levels set for <u>40X</u> magnification.

**Method 3 (Hough Transform using MatLab) Source Code:**



```
MATLAB Command Window                                           Page 1

>> clc;
clear all;
close all;
gray_image = imread('A_20X_1.bmp');
figure;
imshow(gray_image);
[centers, radii, metric] = imfindcircles(gray_image,[5 ↙
10],'ObjectPolarity','dark','Sensitivity',0.95,'Method','twostage');
h = viscircles(centers,radii);
[m,n]=size(centers);
disp(m); %RBC COUNT
```

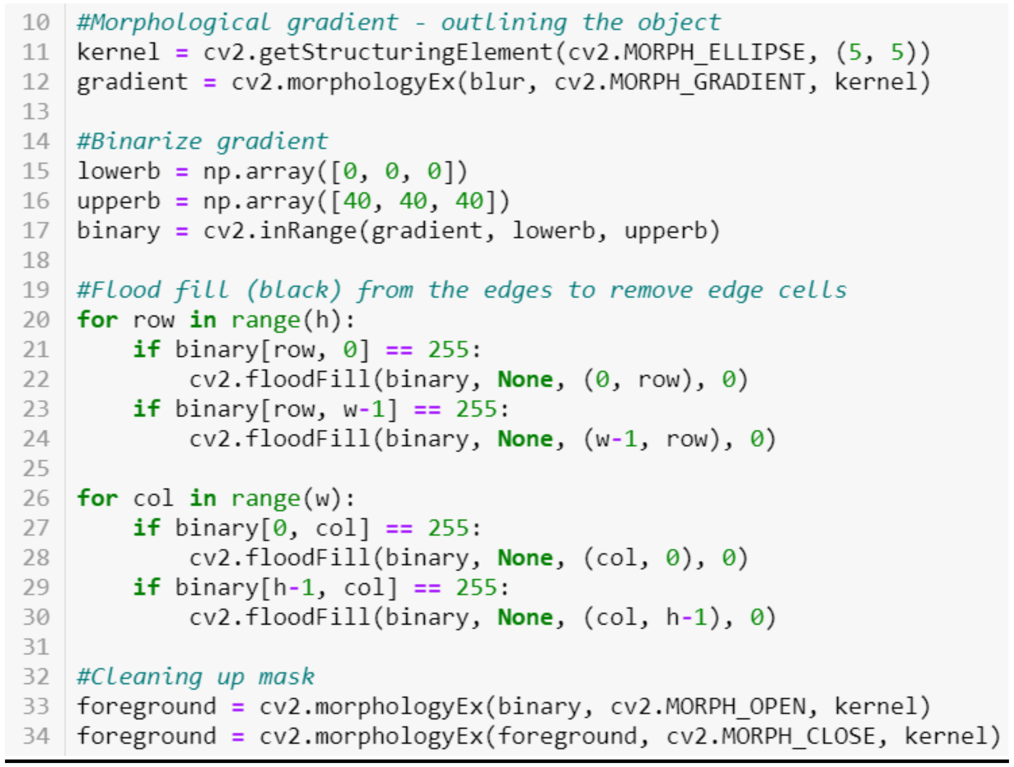
***Figure 26.*** Source Code for Method 3 (MatLab Hough transform) with threshold levels set for 20X magnification.



```
MATLAB Command Window                                           Page 1

>> clc;
clear all;
close all;
gray_image = imread('A_40X_1.bmp'); %reads the image file into a variable
figure;
imshow(gray_image); %shows the image
[centers, radii, metric] = imfindcircles(gray_image,[10 ↙
13],'ObjectPolarity','dark','Sensitivity',0.95,'Method','twostage'); %identifies the ↙
circles
h = viscircles(centers,radii); %displays the identified circles on the image
[m,n]=size(centers);
disp(m); %RBC COUNT
```

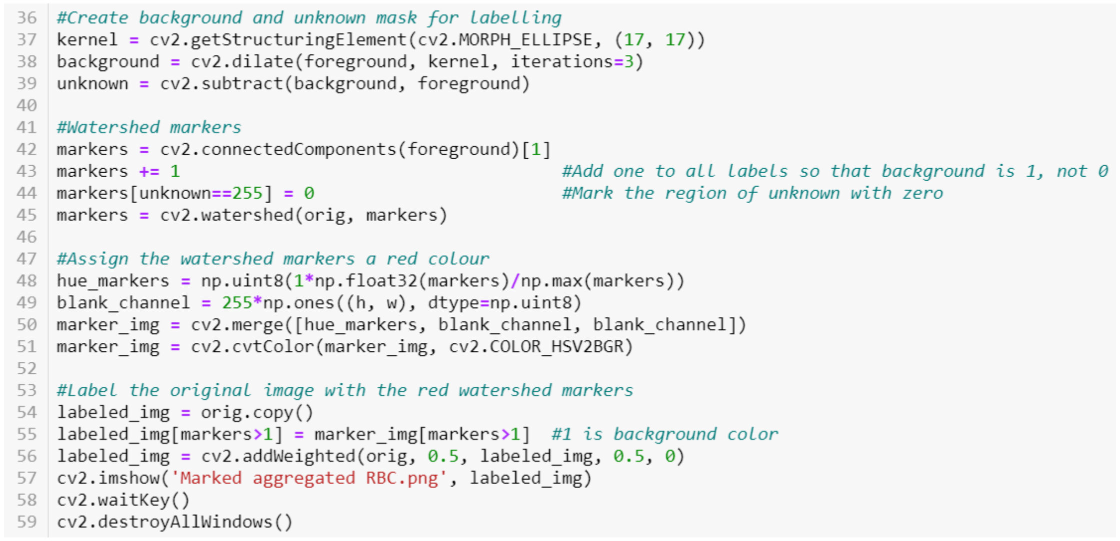
***Figure 27.*** Source Code for Method 3 (MatLab Hough transform) with threshold levels set for 40X magnification.

**Part 2: Source codes for isolating and couting RBCs to determine extent of aggregation.**



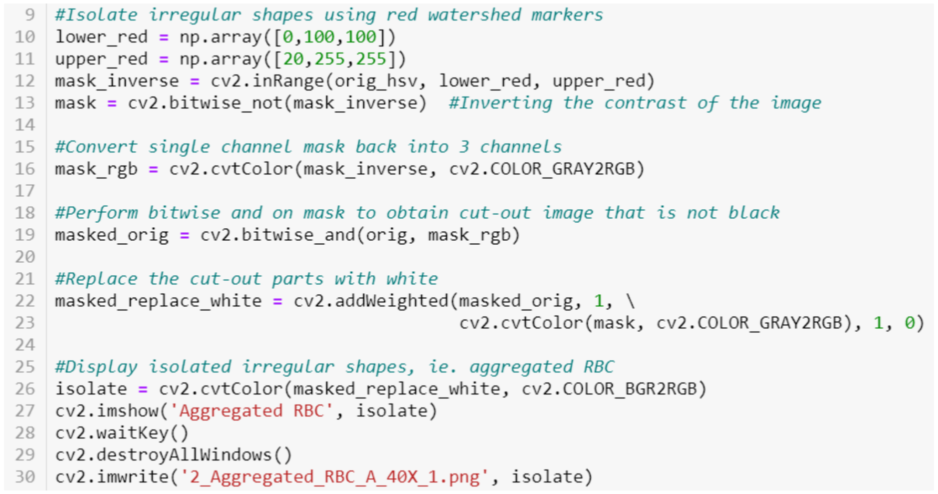
```
#Morphological gradient - outlining the object
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
gradient = cv2.morphologyEx(blur, cv2.MORPH_GRADIENT, kernel)

#Binarize gradient
lowerb = np.array([0, 0, 0])
upperb = np.array([40, 40, 40])
binary = cv2.inRange(gradient, lowerb, upperb)

#Flood fill (black) from the edges to remove edge cells
for row in range(h):
    if binary[row, 0] == 255:
        cv2.floodFill(binary, None, (0, row), 0)
    if binary[row, w-1] == 255:
        cv2.floodFill(binary, None, (w-1, row), 0)

for col in range(w):
    if binary[0, col] == 255:
        cv2.floodFill(binary, None, (col, 0), 0)
    if binary[h-1, col] == 255:
        cv2.floodFill(binary, None, (col, h-1), 0)

#Cleaning up mask
foreground = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)
foreground = cv2.morphologyEx(foreground, cv2.MORPH_CLOSE, kernel)
```

***Figure 28.*** Source Code for outlining of irregularly shaped objects.



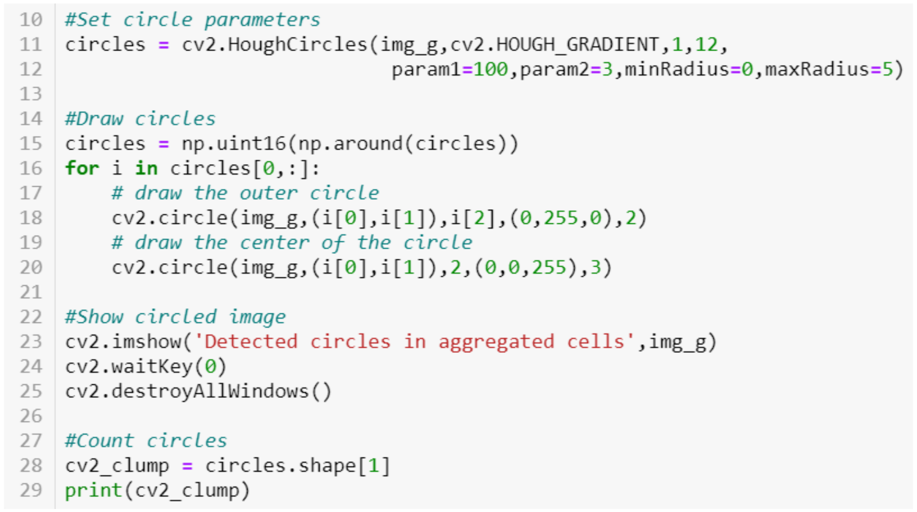
```python
#Create background and unknown mask for labelling
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (17, 17))
background = cv2.dilate(foreground, kernel, iterations=3)
unknown = cv2.subtract(background, foreground)

#Watershed markers
markers = cv2.connectedComponents(foreground)[1]
markers += 1                                     #Add one to all labels so that background is 1, not 0
markers[unknown==255] = 0                        #Mark the region of unknown with zero
markers = cv2.watershed(orig, markers)

#Assign the watershed markers a red colour
hue_markers = np.uint8(1*np.float32(markers)/np.max(markers))
blank_channel = 255*np.ones((h, w), dtype=np.uint8)
marker_img = cv2.merge([hue_markers, blank_channel, blank_channel])
marker_img = cv2.cvtColor(marker_img, cv2.COLOR_HSV2BGR)

#Label the original image with the red watershed markers
labeled_img = orig.copy()
labeled_img[markers>1] = marker_img[markers>1]  #1 is background color
labeled_img = cv2.addWeighted(orig, 0.5, labeled_img, 0.5, 0)
cv2.imshow('Marked aggregated RBC.png', labeled_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

***Figure 29.*** Source Code for creating background and masking detected aggregated RBCs in red for labelling. This part of the code also accounts for overlaying the marked aggregated RBCs on the original image.



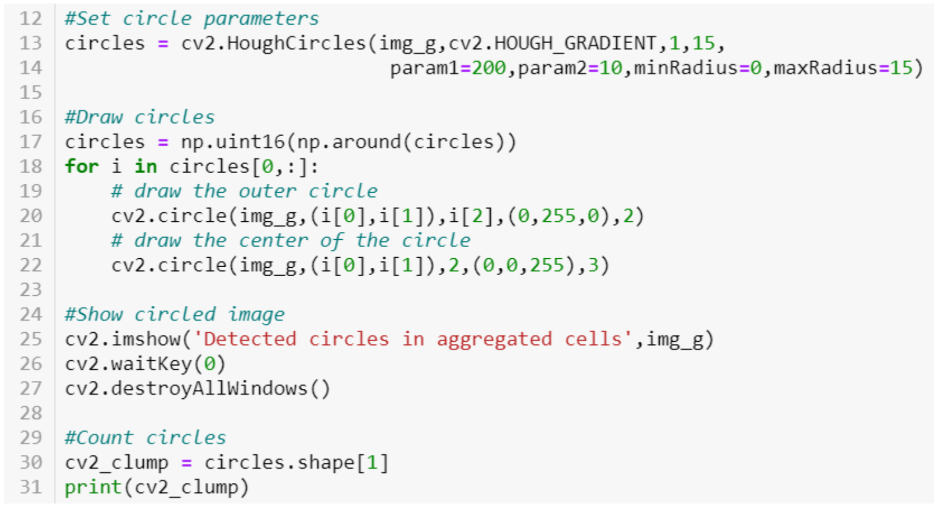
```python
#Isolate irregular shapes using red watershed markers
lower_red = np.array([0,100,100])
upper_red = np.array([20,255,255])
mask_inverse = cv2.inRange(orig_hsv, lower_red, upper_red)
mask = cv2.bitwise_not(mask_inverse)  #Inverting the contrast of the image

#Convert single channel mask back into 3 channels
mask_rgb = cv2.cvtColor(mask_inverse, cv2.COLOR_GRAY2RGB)

#Perform bitwise and on mask to obtain cut-out image that is not black
masked_orig = cv2.bitwise_and(orig, mask_rgb)

#Replace the cut-out parts with white
masked_replace_white = cv2.addWeighted(masked_orig, 1, \
                                        cv2.cvtColor(mask, cv2.COLOR_GRAY2RGB), 1, 0)

#Display isolated irregular shapes, ie. aggregated RBC
isolate = cv2.cvtColor(masked_replace_white, cv2.COLOR_BGR2RGB)
cv2.imshow('Aggregated RBC', isolate)
cv2.waitKey()
cv2.destroyAllWindows()
cv2.imwrite('2_Aggregated_RBC_A_40X_1.png', isolate)
```

***Figure 30.*** Source Code for isolating the marked aggregated RBCs and displaying it on a white background to observe the isolated cells.



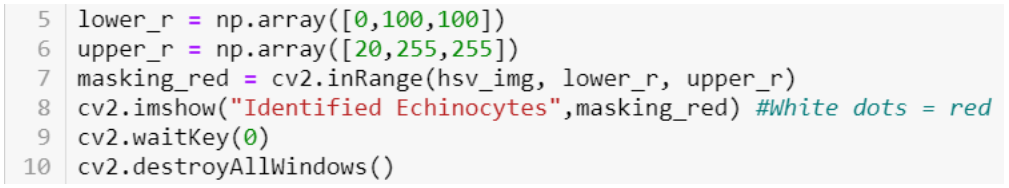
```python
#Set circle parameters
circles = cv2.HoughCircles(img_g,cv2.HOUGH_GRADIENT,1,12,
                            param1=100,param2=3,minRadius=0,maxRadius=5)

#Draw circles
circles = np.uint16(np.around(circles))
for i in circles[0,:]:
    # draw the outer circle
    cv2.circle(img_g,(i[0],i[1]),i[2],(0,255,0),2)
    # draw the center of the circle
    cv2.circle(img_g,(i[0],i[1]),2,(0,0,255),3)

#Show circled image
cv2.imshow('Detected circles in aggregated cells',img_g)
cv2.waitKey(0)
cv2.destroyAllWindows()

#Count circles
cv2_clump = circles.shape[1]
print(cv2_clump)
```

***Figure 31.*** Source Code for counting the isolated RBCs in the sample image that are involved in aggregation, under 20X magnification.



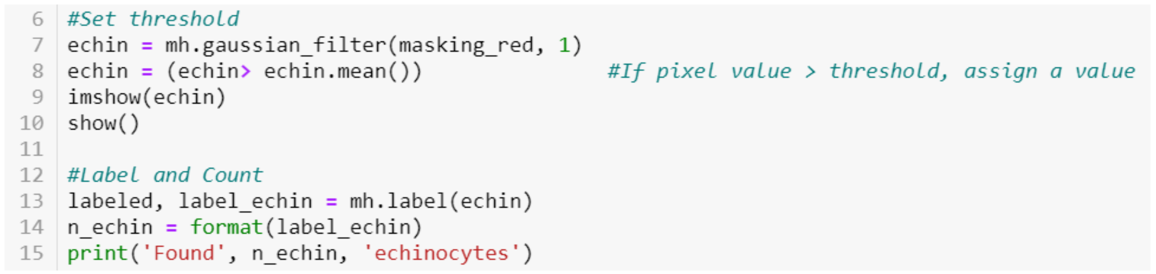
```python
#Set circle parameters
circles = cv2.HoughCircles(img_g,cv2.HOUGH_GRADIENT,1,15,
                            param1=200,param2=10,minRadius=0,maxRadius=15)

#Draw circles
circles = np.uint16(np.around(circles))
for i in circles[0,:]:
    # draw the outer circle
    cv2.circle(img_g,(i[0],i[1]),i[2],(0,255,0),2)
    # draw the center of the circle
    cv2.circle(img_g,(i[0],i[1]),2,(0,0,255),3)

#Show circled image
cv2.imshow('Detected circles in aggregated cells',img_g)
cv2.waitKey(0)
cv2.destroyAllWindows()

#Count circles
cv2_clump = circles.shape[1]
print(cv2_clump)
```

***Figure 32.*** Source Code for counting the isolated RBCs in the sample image that are involved in aggregation, under 40X magnification.



```python
lower_r = np.array([0,100,100])
upper_r = np.array([20,255,255])
masking_red = cv2.inRange(hsv_img, lower_r, upper_r)
cv2.imshow("Identified Echinocytes",masking_red) #White dots = red
cv2.waitKey(0)
cv2.destroyAllWindows()
```

***Figure 33.*** Source Code for identifying echinocytes (Part 1)



```python
#Set threshold
echin = mh.gaussian_filter(masking_red, 1)
echin = (echin> echin.mean())                #If pixel value > threshold, assign a value
imshow(echin)
show()

#Label and Count
labeled, label_echin = mh.label(echin)
n_echin = format(label_echin)
print('Found', n_echin, 'echinocytes')
```

***Figure 34.*** Source Code for identifying echinocytes (Part 2)

## D. Equations

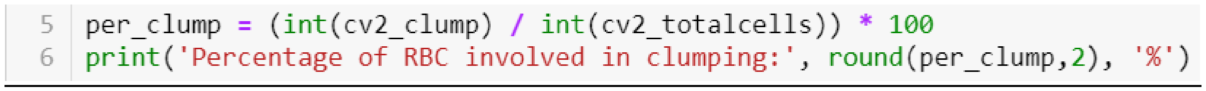
**Equation 1: calculation for 95% confidence interval used in Table 2.**

$$\bar{x} \pm t \frac{s}{\sqrt{n}}$$

***Equation 1:*** Formula used to calculate a 95% confidence interval for RBCs in the sample.

Where $\bar{x}$ is the sample mean, $s$ is the standard deviation, $n$ is the sample size and $t$ is the coefficient fixed at 1.96 for a 95% confidence interval. The purpose of calculating a 95% Confidence Interval is to determine how wide our margin of error is when trying to estimate the population mean from the sample mean.

**Equation 2: Calculating the percentage of aggregated RBCs in the sample.**



```python
per_clump = (int(cv2_clump) / int(cv2_totalcells)) * 100
print('Percentage of RBC involved in clumping:', round(per_clump,2), '%')
```

***Figure 35.*** Source code in Python to calculate the percentage of aggregated RBCs in the sample to determine the extent of aggregation.

$$\frac{Number\ of\ RBCs\ invovled\ in\ aggregation\ in\ the\ sample\ image}{Total\ number\ of\ RBCs\ in\ sample\ image} \times 100\%$$

***Equation 2.*** Equation to calculate the percentage of aggregated RBCs in the sample to determine the extent of aggregation.