Activity 10

Dela Cruz, Mary Nathalie G.
2015-09114

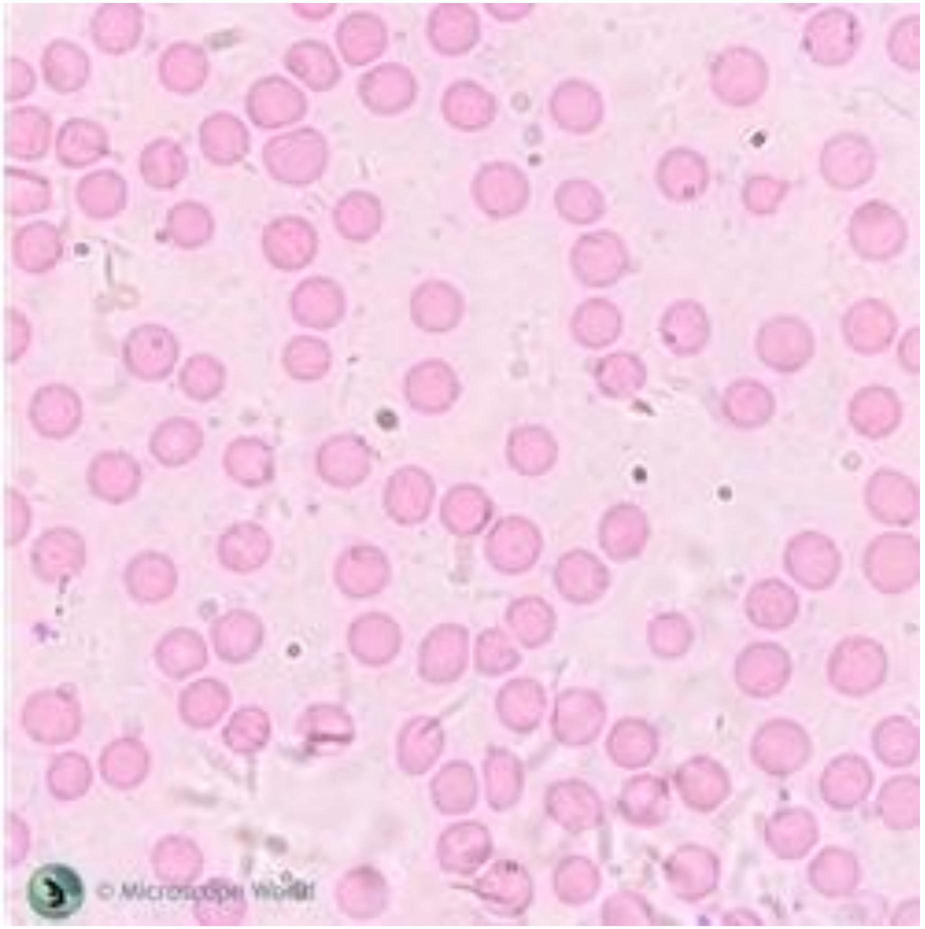

Figure 1: Original image of the cells

Objective: Use blob analysis on the image of cells shown in Figure 1 to measure the shape features of the cells in the image.

Steps done in the activity

Non-parametric image segmentation

↓

Cleaning the segmented image

↙ ↓ ↘

Fill holes | imerode() + imopen() | Watershed

↓

Blob analysis

STEP 1: Non-parametric image segmentation

The algorithm for this step is as follows:

```
close all
clear all
clc

image = imread('cell.jpg');
image = imresize(image, [600 600]);
image = imcrop(image, [12 0 555 600]);
figure(); imshow(image);
im = double(image);
imm = im;
imR = im(:,:,1); imG = im(:,:,2); imB = im(:,:,3);
imI = imR + imG + imB;
imI(imI==0) = 100000;
imr = imR./imI; img = imG./imI; imb = imB./imI;

% ROI
roi2 = imcrop(im, [496 533 19 10]);
roi = roi2;
figure(); imshow(roi);
roiR = roi(:,:,1); roiG = roi(:,:,2); roiB = roi(:,:,3
roiI = roiR + roiG + roiB; roiI(roiI==0) = 100000;
roir = roiR./roiI; roig = roiG./roiI;

```

```
%% NONPARAMETRIC SEG

bin = 115;
intr = round(roir*(bin-1)+1);
intg = round(roig*(bin-1)+1);
color = intg(:) + (intr(:)-1)*bin;
hist = zeros(bin,bin);
for row = 1:bin
    for column = 1:(bin-row+1)
        hist(row,column) = length(find(color ==(((column+(row-1)*bin)))));
    end
end
a = imrotate(hist,90);
% figure(4);imshow(a);

imsize = size(imr); npsroi  = zeros(imsize(1),imsize(2));
for i = 1:imsize(1)
    for j = 1:imsize(2)
        rnew = round(imr(i,j)*(bin-1)+1);
        gnew = round(img(i,j)*(bin-1)+1);
        npsroi(i,j) = hist(rnew,gnew);
    end
end
im = npsroi;
figure(5); imshow(im);

```

STEP 1: Non-parametric image segmentation

Lines 11-14 transform the RGB coordinates of the image into normalized chromaticity coordinates (NCC) to consider the shading variation.
Lines 17-22 extract a region of interest (ROI) from the image and transform the RGB coordination of the ROI into NCC.
Lines 26-48 performs non-parametric segmentation. This method utilizes histogram backprojection where the pixels of the image were replaced with their corresponding histogram value in the chromaticity space.
I chose the non-parametric method because I realized from doing the Activity 7 that the amount of segmentation can be varied by changing the number of bins used in non-parametric method. Hence, this method is more flexible than the parametric method.

STEP 1: Non-parametric image segmentation

Figure 2 is the result of the image segmentation. As shown, the image appears grainy, and has blobs that have holes and are touching. Hence, further processing must be done to clean the image.

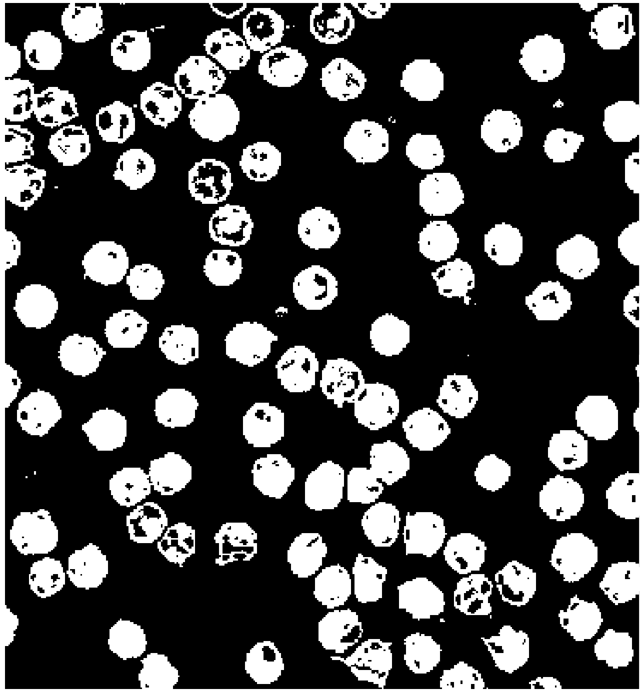

Figure 2: Image obtained after non-parametric image segmentation

STEP 2: Cleaning the segmented image

The algorithm for this step is as follows:

```
% FILL OUT CELLS
im_1 = padarray(im,[1 1],1,'pre'); %top & left
im_1 = imfill(im_1,'holes');
im_1 = im_1(2:end,2:end);

im_2 = padarray(padarray(im,[1 0],1,'pre'),[0 1],1,'post'); %top & right
im_2 = imfill(im_2,'holes');
im_2 = im_2(2:end,1:end-1);

im_3 = padarray(im,[1 1],1,'post'); %bottom & right
im_3 = imfill(im_3,'holes');
im_3 = im_3(1:end-1,1:end-1);

im_4 = padarray(padarray(im,[1 0],1,'post'),[0 1],1,'pre'); %bottom & left
im_4 = imfill(im_4,'holes');
im_4 = im_4(1:end-1,2:end);

im = im_1 | im_2 | im_3 | im_4; figure(); imshow(im);

% USE IMERODE() AND IMOPEN()
im = imerode(im, strel('diamond',1)); figure(); imshow(im);
im = imerode(im, strel('diamond',2)); figure(); imshow(im);
im = imopen(im, strel('diamond',3)); figure(); imshow(im);

% WATERSHED
D = -bwdist(~im);figure(); imshow(D,[]);
mask = imextendedmin(D,2); imshowpair(im,mask,'blend');
D2 = imimposemin(D,mask);Ld2 = watershed(D2); imshow(label2rgb(Ld2));
bw3 = im; bw3(Ld2 == 0) = 0; figure(); imshow(bw3);
im = bw3;
```

STEP 2.1: Image cleaning by filling out the holes

Lines 53-69 fills the holes of the blobs in the image. Note that the imfill() function does not fill the holes of the blobs that are touching the border. To solve this, padarray() function was used to add a row or a column of pixels at the borders of the image so that when the imfill() function was used, it includes the blobs that were initially touching the border. To revert back to the original binary image, the image matrix was indexed such that the rows or columns pixels that were added by padarray() function were removed. In particular, **lines 53-55** fill the blobs connected to the top & left borders, **lines 57-59** fill the blobs connected to the top & right borders, **lines 61-63** fills the blobs connected to the bottom & right borders, and **lines 65-59** fill the blobs connected to the bottom & left borders. Finally, logical or was operated to get combine them.

Reference: https://blogs.mathworks.com/steve/2013/09/05/defining-and-filling-holes-on-the-border-of-an-image/

STEP 2.1: Image cleaning by filling out the holes

Figure 3 is the result of image cleaning by filling out the holes. As shown, the holes in the blobs of the image were all removed.

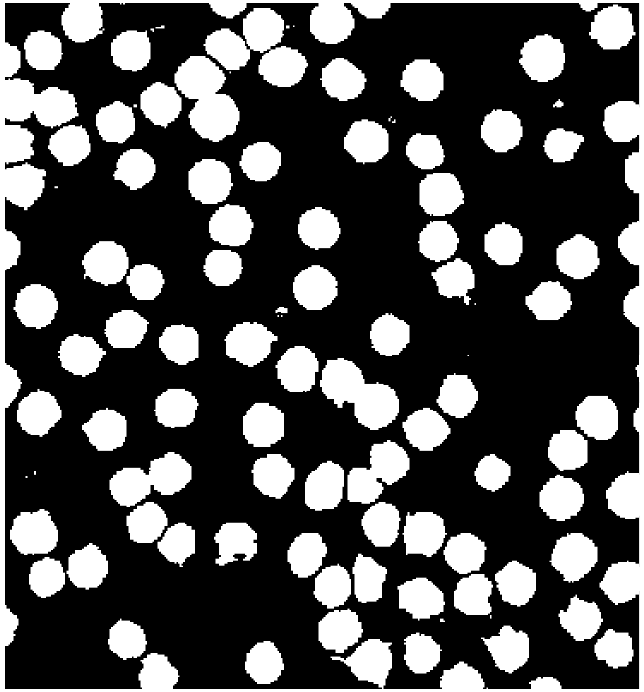

Figure 3: Image obtained after cleaning by filling out holes

STEP 2.2: Image cleaning by morphological processing

Lines 73-75 clean the segmented by using morphological operations. Imerode() function and Imopen() function were used to reduce the size of the blob and/or to remove the stray pixels.

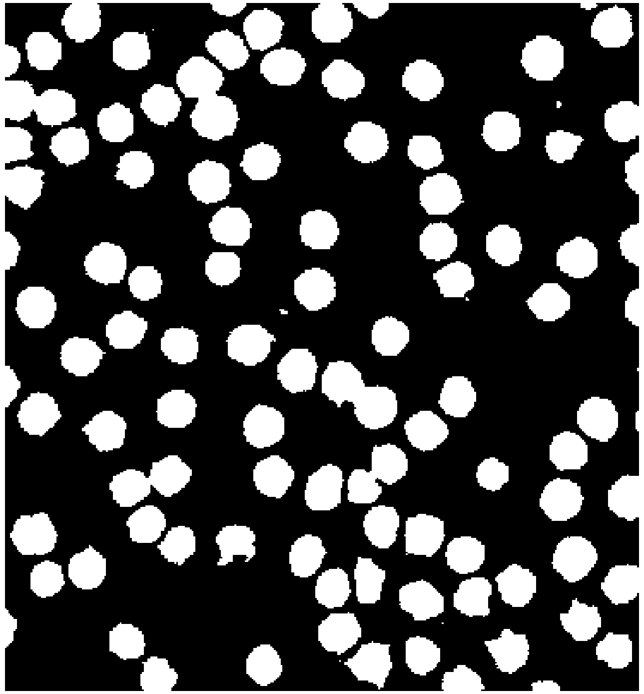

Figure 4.1: Image obtained after the 1st imerode() function

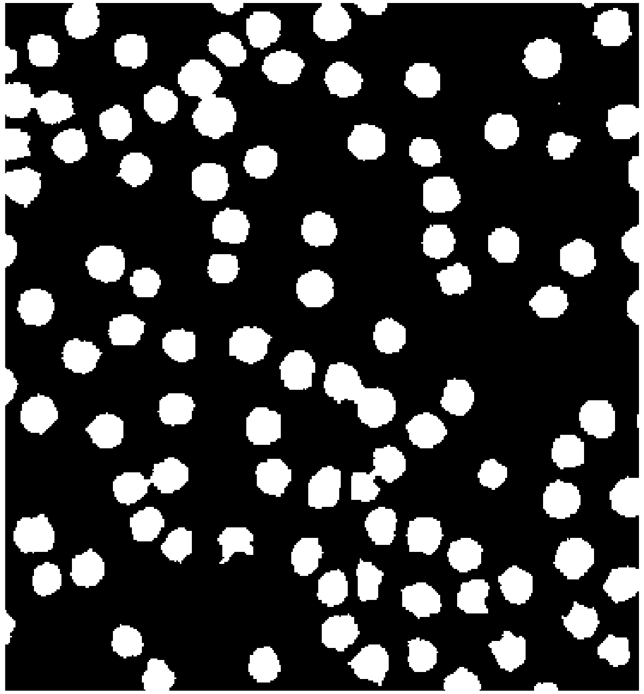

Figure 4.2: Image obtained after the 2nd imerode() function

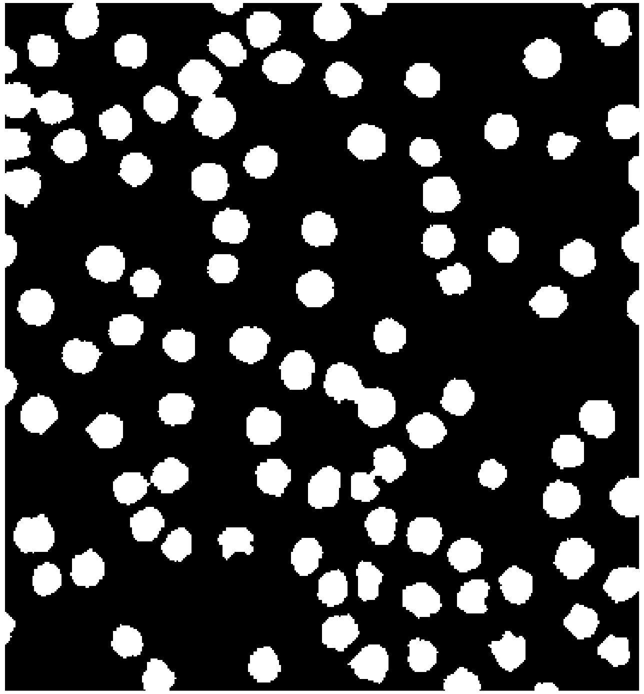

Figure 4.3: Image obtained after the imopen() function

STEP 2.3: Image cleaning by watershed segmentation

Lines 77-81 perform watershed segmentation on the image to separate touching blobs. Here, the watershed() function treats the image as a surface with high elevations at the white pixels and low elevations at the dark pixels. To use this, we filter out tiny maxima that are in the middle of the blobs in line 78 and 79. Then, we compute its watershed transform in line 80. The ridge lines of the watershed was then used to segment the original image in line 81.

Reference:
https://blogs.mathworks.com/steve/2013/11/19/watershed-transform-question-from-tech-support/

STEP 2.3: Image cleaning by watershed segmentation

Figure 5.2 is the result of the watershed segmentation. As shown, the connected blobs are separated. As shown, the blobs are have similar shapes and sizes to their cells counterpart which is shown in Figure 5.1. We can now perform blob analysis to get the shape features of the cells in the image.

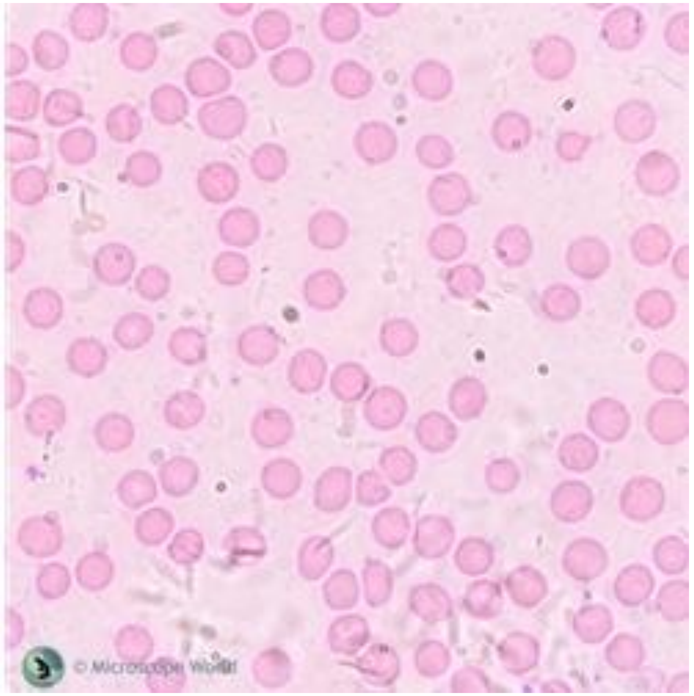

Figure 5.1: Original Image

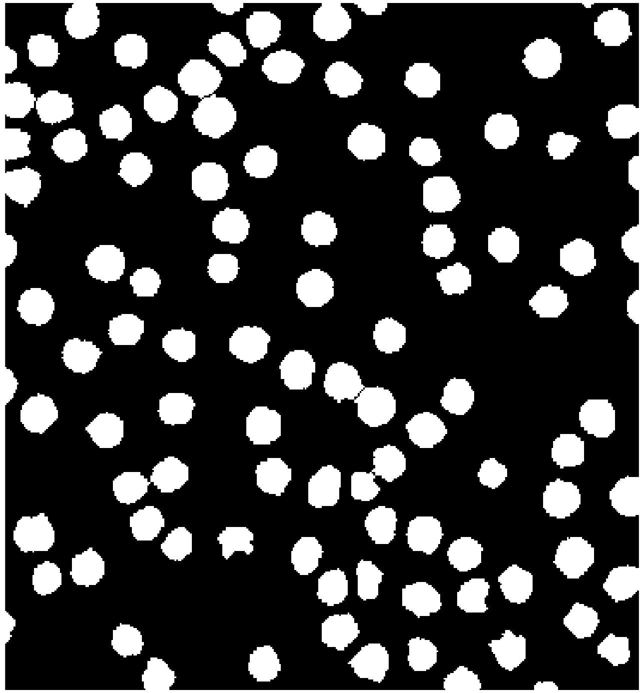

Figure 5.2: Image obtained after cleaning

STEP 3: Blob analysis

Lines 85-96 performs labelling and displays the label on the centroid of the corresponding blob. By using numel(regionprops()) function, I found that there are 104 cells in the image.

```matlab
83      %% with label
84
85      [label num] = bwlabel(im);
86      cent = regionprops(label, 'all');
87      figure(); imshow(im);
88      hold on
89      for i = 1:numel(cent)
90          c = cent(i).Centroid;
91          text(c(1),c(2),sprintf('%d',i),...
92              'HorizontalAlignment','center',...
93              'VerticalAlignment','middle',...
94              'FontSize',6);
95      end
96      hold off
97
```

Figure 6.1: Algorithm for this step

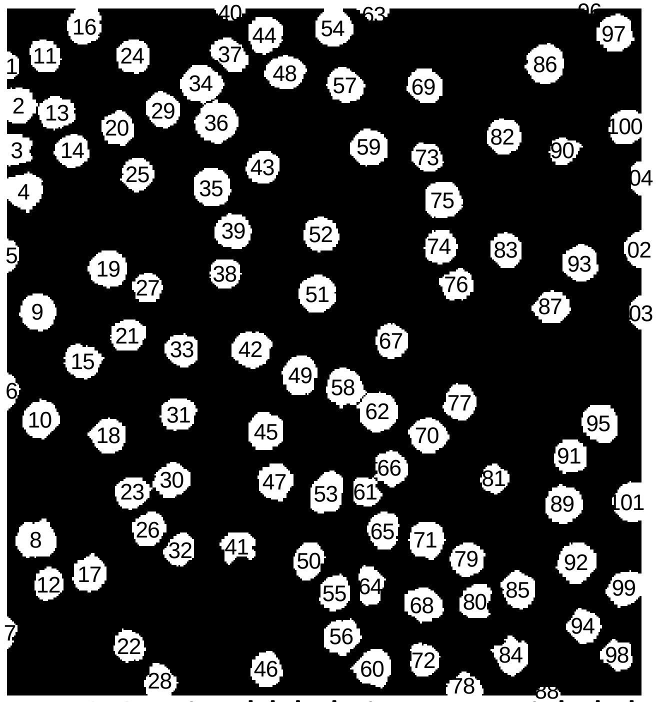

Figure 6.2: Final blob image with labels

STEP 3: Blob analysis

I did **lines 100-105** for fun. These lines result to an image that displays the centroids and the borders of the blobs.

```
98      %% marked at centroid
99
100 -   bw4 = bwperim(im);
101 -   figure(); imshow(bw4);
102 -   centroids = cat(1,cent.Centroid);
103 -   hold on
104 -   plot(centroids(:,1),centroids(:,2),'b*');
105 -   hold off
106
```

Figure 7.1: Algorithm for this step

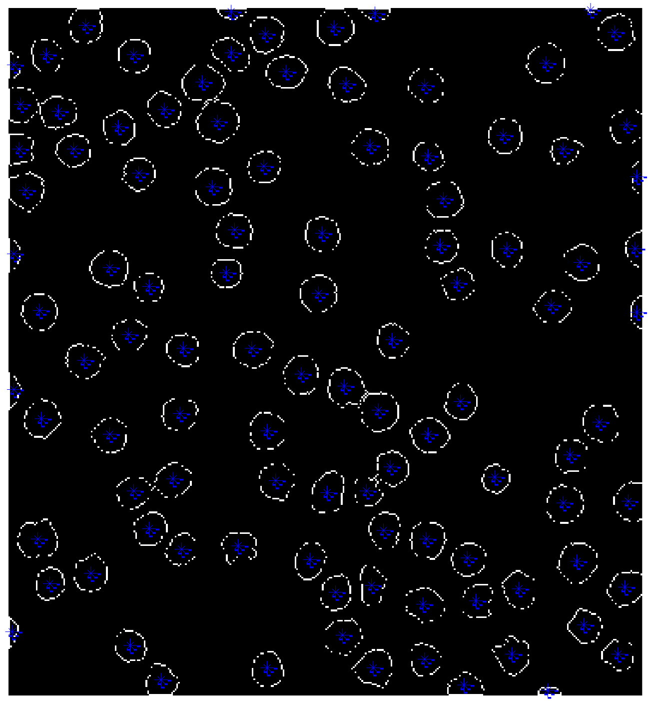

Figure 7.2: Final blob image displaying the centroid and the border of each blob

STEP 3: Blob analysis

Lines 109-118 marks the centroid of each blob and encloses each blob in a bounding box. It results to Figure 8.2. In the figure, the centroid of each blob is represented by the blue asterisk and the bounding box is represented by the red box.

```
107      %% marked at centroid and bounding box
108
109 -    figure(); imshow(im);
110 -    hold on
111 -    box=[cent.BoundingBox];
112 -    box=reshape(box,[4 num]); %reshape to 1D array
113 -    for i=1:num
114 -        rectangle('position',box(:,i),'edgecolor','r');
115 -    end
116 -    centroids = cat(1,cent.Centroid);
117 -    plot(centroids(:,1),centroids(:,2),'b*');
118 -    hold off
119
```

Figure 8.1: Algorithm for this step

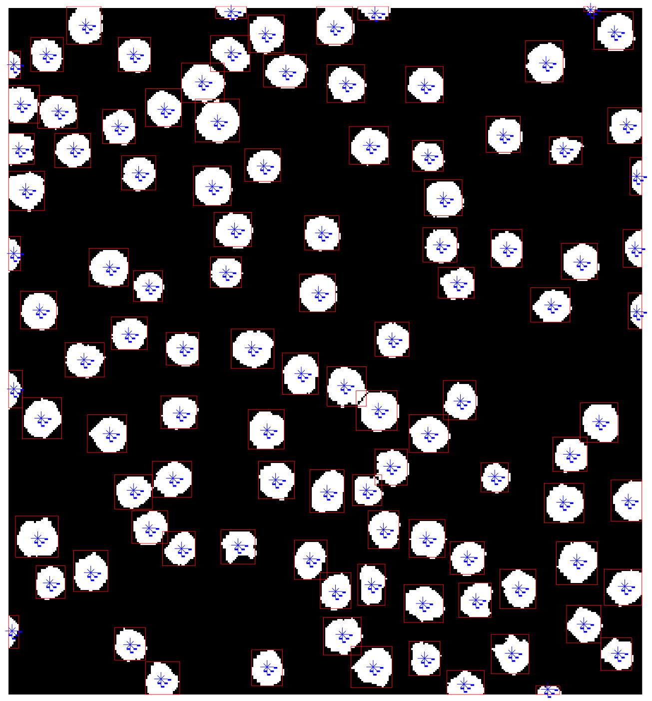

Figure 8.2: Final blob image displaying the centroid and the bounding box of each blob

STEP 3: Blob analysis

The following lines calculate and display the shape features of the image:

```
%% Calculations

area_mean = mean([cent.Area]); area_std = std([cent.Area]);
eccen_mean = mean([cent.Eccentricity]); eccen_std = std([cent.Eccentricity]);
major_mean = mean([cent.MajorAxisLength]); major_std = std([cent.MajorAxisLength]);
minor_mean = mean([cent.MinorAxisLength]); minor_std = std([cent.MinorAxisLength]);
peri_mean = mean([cent.Perimeter]); peri_std = std([cent.Perimeter]);

disp(['mean area = ' num2str(area_mean) ', std area = ' num2str(area_std)]);
disp(['mean eccen = ' num2str(eccen_mean) ', std eccen = ' num2str(eccen_std)]);
disp(['mean major = ' num2str(major_mean) ', std major = ' num2str(major_std)]);
disp(['mean minor = ' num2str(minor_mean) ', std minor = ' num2str(minor_std)]);
disp(['mean peri = ' num2str(peri_mean) ', std peri = ' num2str(peri_std)]);
```

They show the best estimate of the average cell area, average eccentricity, average major axis length, average minimum axis length and the average perimeter.

STEP 3: Blob analysis

Here's the data obtained. Note that this is in number of pixels. Since I resized the image in line 6 to 600 pixels by 600 pixels, my result may differ from other people's result.

Best estimate of the average area: 682.25 $\pm$ 195.0396
Best estimate of the average eccentricity: 0.45763 $\pm$ 0.20872
Best estimate of the average major axis length: 31.7978 $\pm$ 3.643
Best estimate of the average minor axis length: 27.1882 $\pm$ 6.3117
Best estimate of the average perimeter: 93.1047 $\pm$ 14.2225

Further investigation

When I did not erode the image in step 2.2 (image cleaning by morphological processing) and only used imopen(), it results to larger blobs, as shown in Figure 9.2. Hence, the resulting measurements for the cell shape features are all larger, except for the eccentricity.

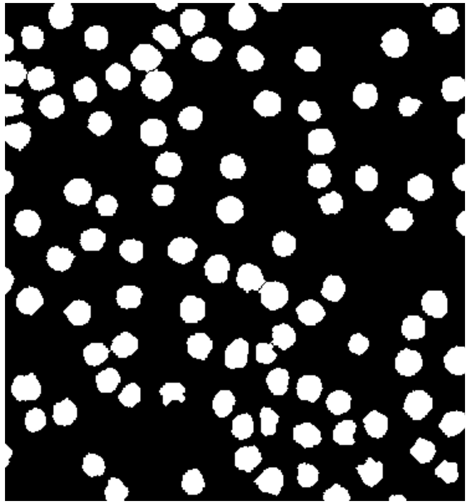

Figure 9.1: Final blob image from previous algorithm when imerode() was used twice.

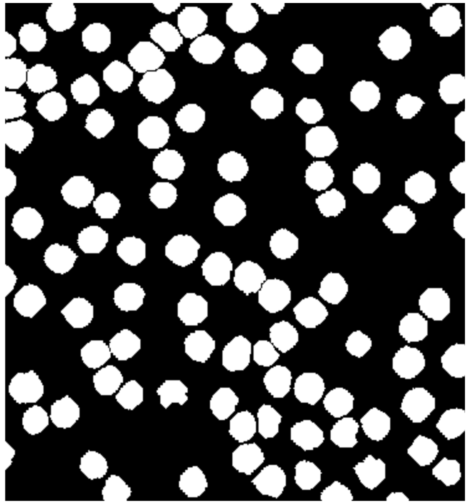

Figure 9.2: Final blob image for this investigation when imerode() is not used.

Best estimate of the average area: 934.2404 $\pm$ 248.1446
Best estimate of the average eccentricity: 0.44029 $\pm$ 0.20471
Best estimate of the average major axis length: 37.0271 $\pm$ 3.6981
Best estimate of the average minor axis length: 32.0566 $\pm$ 6.9017
Best estimate of the average perimeter: 109.2104 $\pm$ 14.6806

Self-evaluation: 12/10