Digital Image Processing

Assignment 02

Group member:

Kuldeep Dileep Lohana (kl04008)

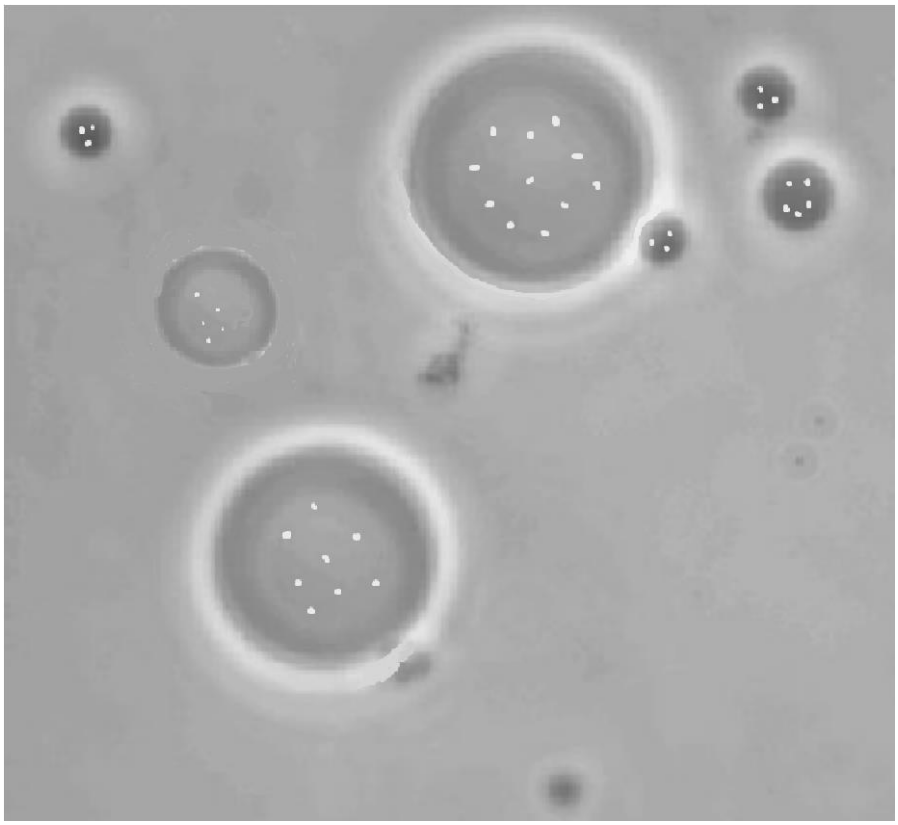
Syed Muhammad Raza Naqvi (sn03805)

Part1: Cell Image Segmentation

Task1: Take the image 'polymersomes.tif' and preprocess it, e.g., to convert it to appropriate data type, to enhance and remove any noise, and finally sharpen it.

Given Image:

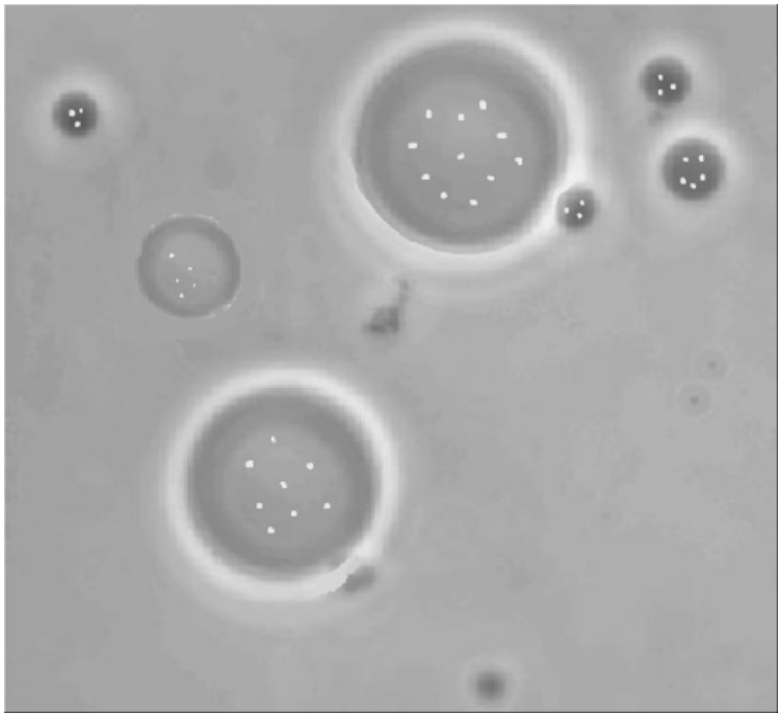
Original Image



Preprocessing: We tried different filters (results shown below) to achieve the best result. We got equally good results from both averaging and 2D adaptive wiener filter but we decided to carry on our experiment with adaptive filter's output because it's adaptive and hence, it preserved the information inside each region i.e. dots whereas, averaging filter blurred them out which reduces the accuracy of cell segment analysis.

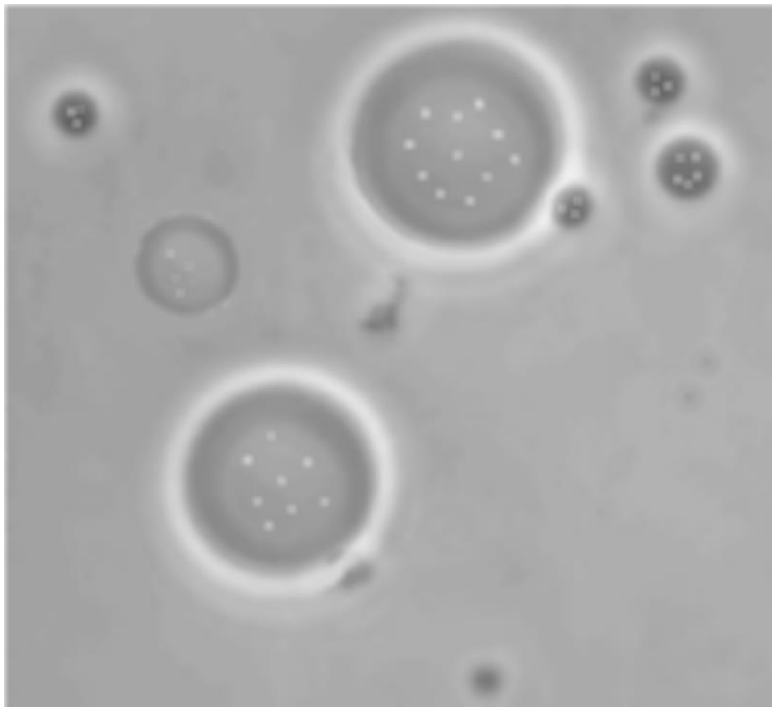
1) Gaussian Filter:

Gaussian Filtered Image



2) Averaging Filter:

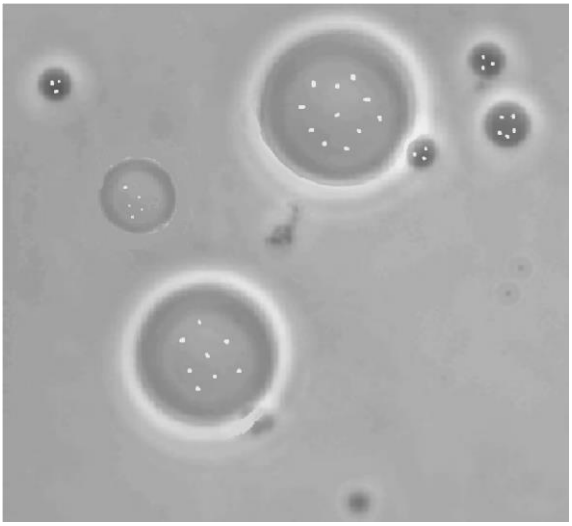
Averaging filtered Image



3) Gaussian Bilateral Filtered Image:

Patch

Gaussian bilateral filtered Image



4) 2-D Wiener Filter:

2-D adaptive noise-removal filtering

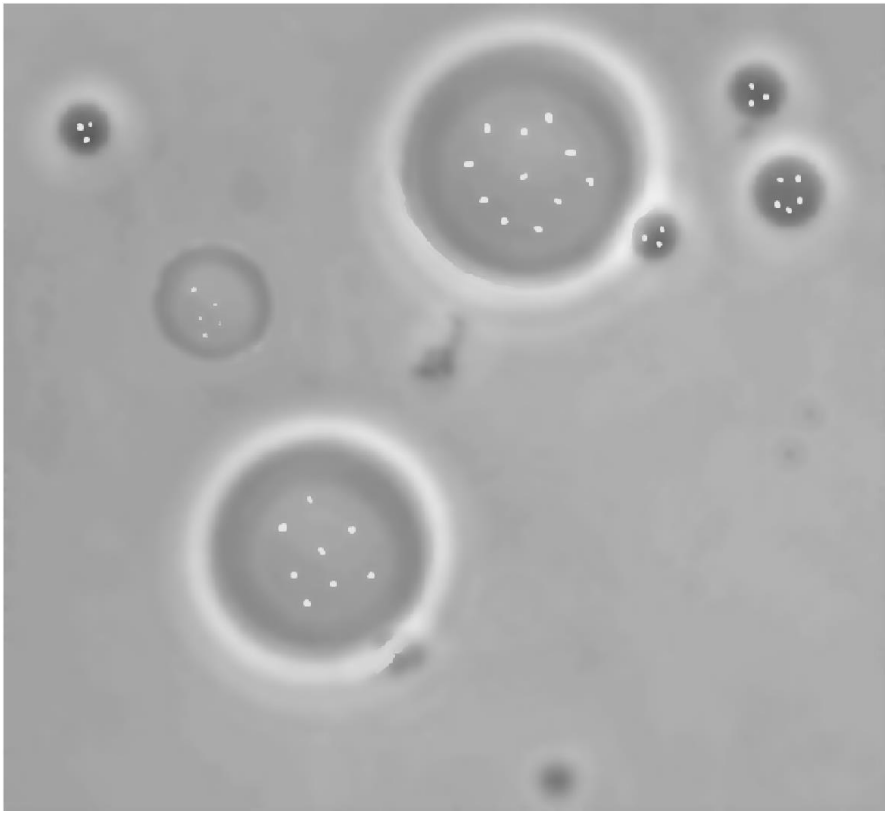
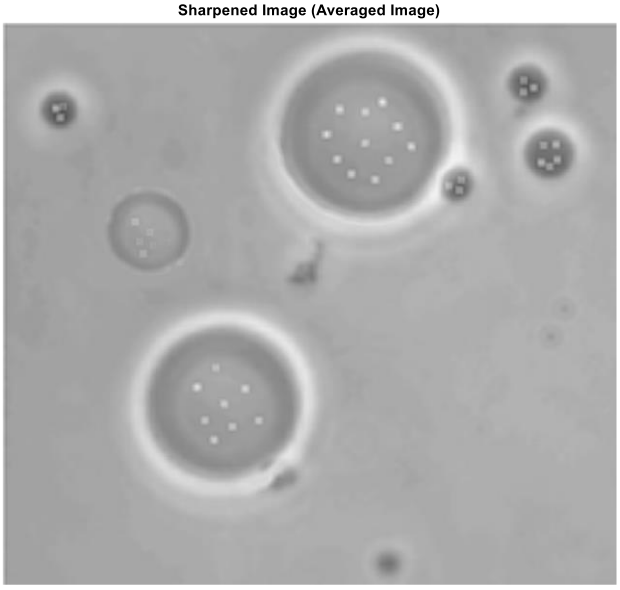
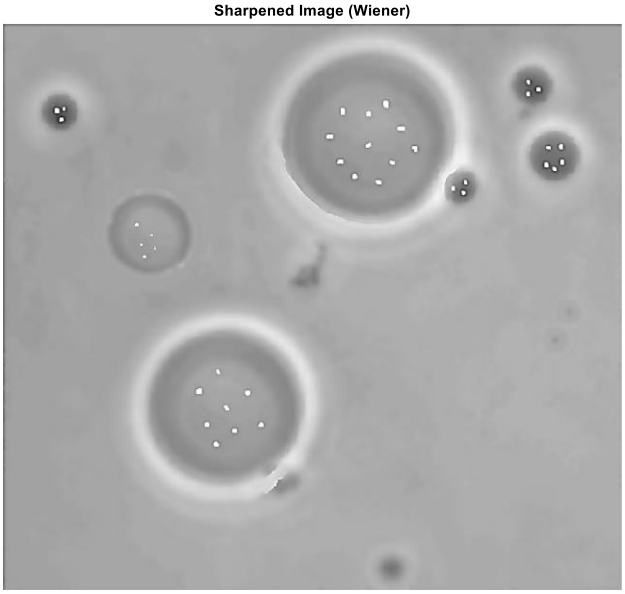


Image Sharpening: We enhanced the features of the filtered image with the help of MATLAB's built-in function IMSHARPEN where 'Radius' which signifies the standard deviation of the Gaussian lowpass filter used as part of unsharp masking was set equal to 2. It controls the size of the region around the edge pixels that is affected by sharpening. A large value sharpens wider regions around the edges, whereas a small value sharpens narrower regions around edges. On the other hand, 'Amount', which Specifies the strength of the sharpening effect, was also set equal to 2. A higher value leads to larger increase in the contrast of the sharpened pixels.

1) **Result of Averaging Filter sharpened:**



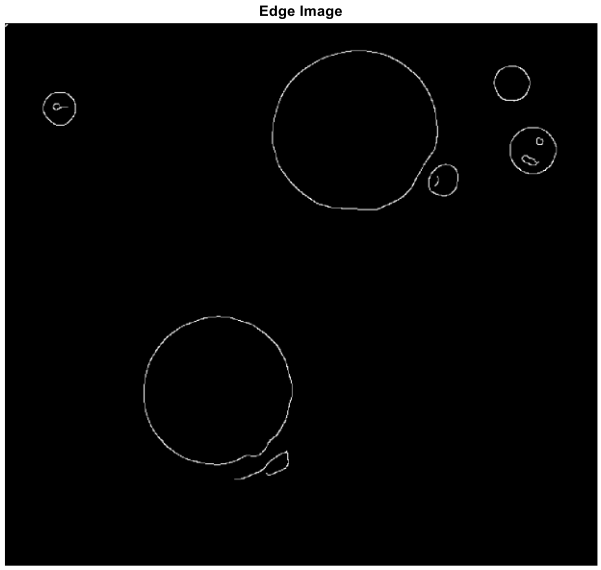
2) **Result of Wiener Filter sharpened:**



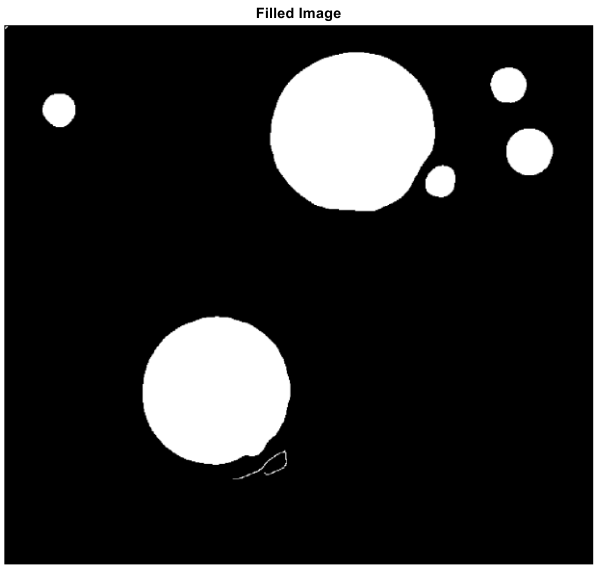
Wiener gives better result where the information within the regions of interest is preserved.

Task2: Apply appropriate edge detection method to extract the edges of the cell body as is depicted in the Figure 1b. Once the edge detection is performed, we can use it to segment the regions of interest, that is, the cell body. The edges can be connected if they are not already and flood fill operation (use help to study function for it) on the resulting image gives the segmented cells regions as shown in Figure 1c.

Edge Detection: We performed canny edge detection on the sharpened image of Wiener filter to extract the edges of the objects in the image. To achieve this, we used MATLAB's built-in function 'EDGE' where the threshold is 0.28 as per our image's sensitivity and directionality was set as 4.

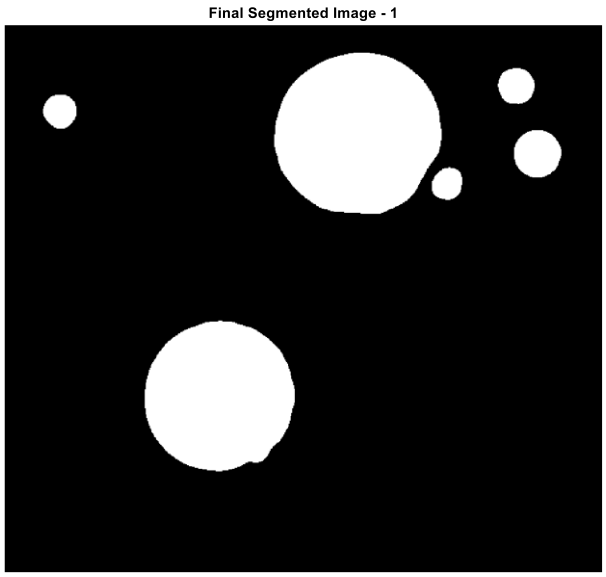


Region Filling: Next we perform flood-fill operation on holes in background pixels from the edge of the image that cannot be reached by filling in the background. For this purpose, we used MATLAB's built-in function 'IMFILL'.



Task 3: The resulting image has non-cellular components and false edges due to imperfect segmentation as visible in Figure 1c. Use a combination of morphological operations to perform filtering to get an image as shown in Figure 1d. Use help to study functions like 'BWAREAOPEN'', 'IMCLEARBORDER' for getting an idea if they could be applicable here. For example, consider removing objects with pixel area less than say 200-400 pixels.

Segmentation (Method 1): As you can observe in our last result that there still is a non-cellular region in the image so to further filter our image, such that we only get the desired cellular region we used MATLAB's built-in function 'BWAREAOPEN' which removes from a binary image all the connected components (objects) that have fewer than P pixels, where in our case P is 300 along with 8 connectivity.



Task4: Apply cell image segmentation method described in the attached paper page 4. Compare the segmentation results from the two methods.

Given Image:

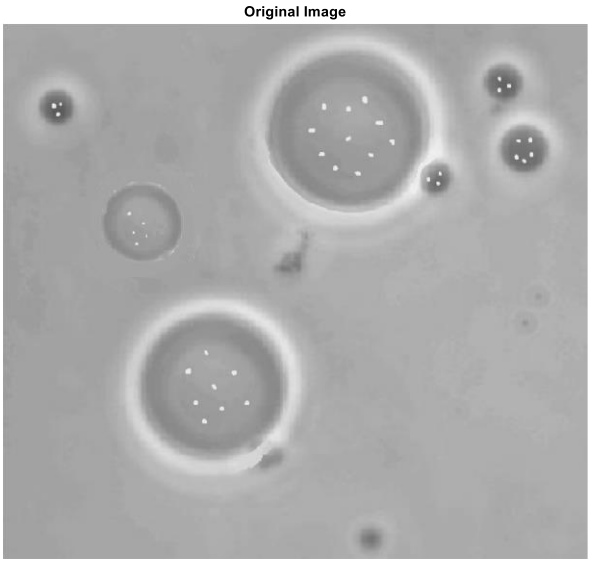
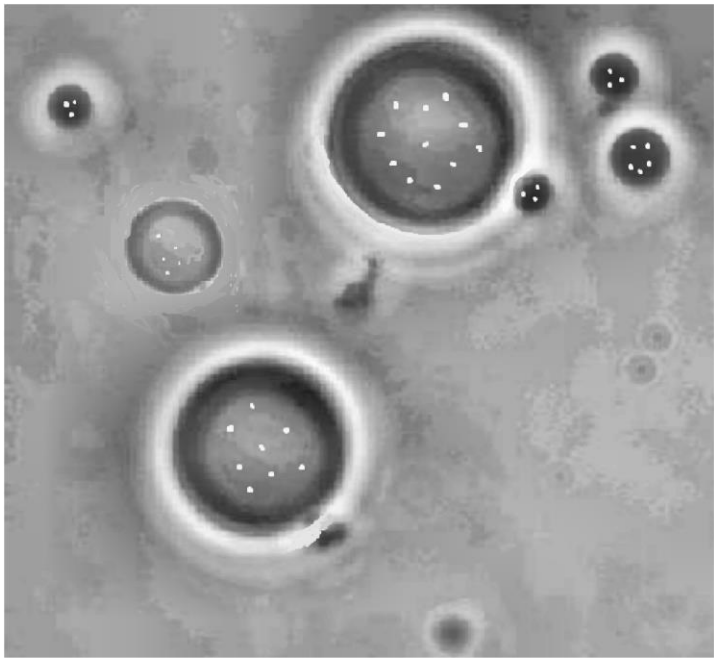


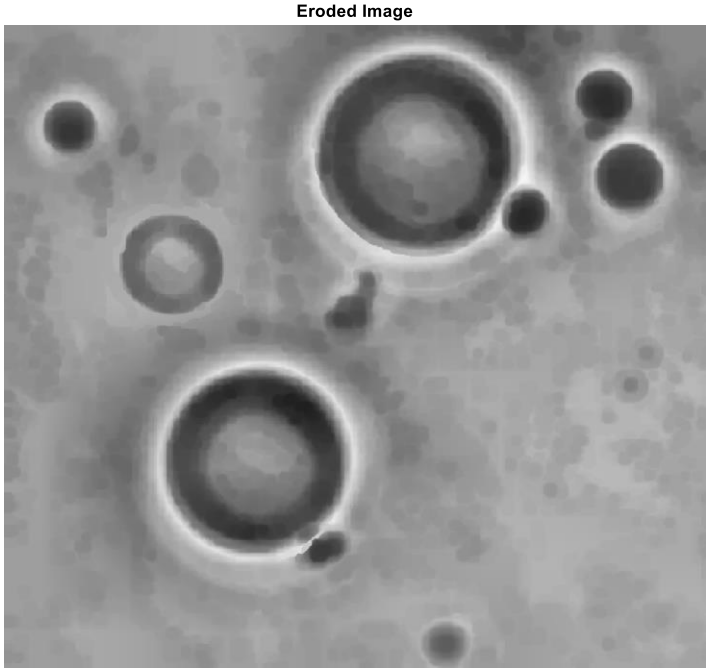
Image Histogram Equalization: To perform adaptive histogram equalization, as a preprocessing step, we use 'ADAPTHISTEQ' which enhances the contrast of the given image by transforming the values in the intensity image.

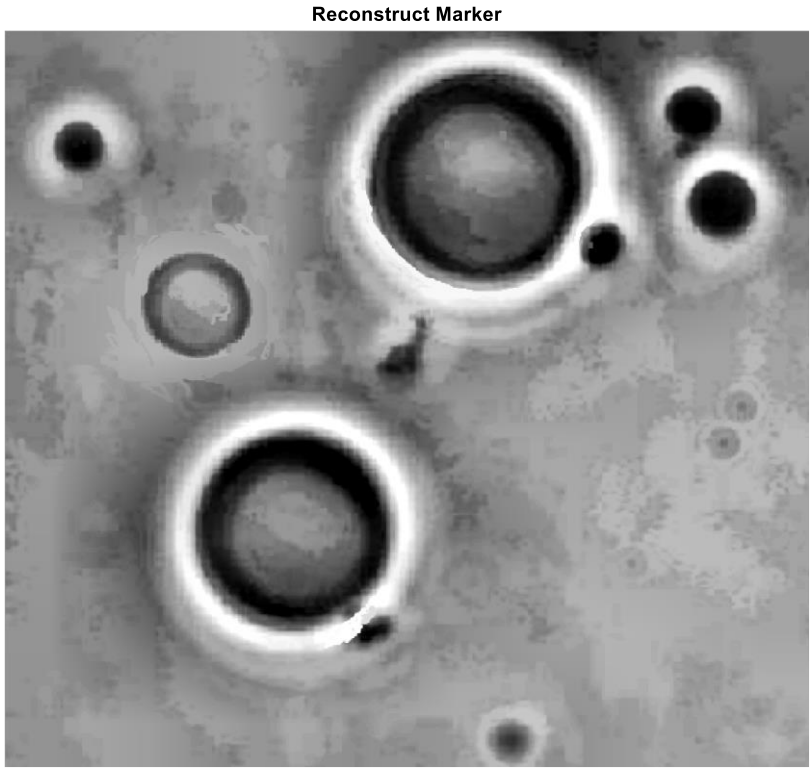
Contrast-limited adaptive histogram equalization (CLAHE)



Morphological Reconstruction: We create a marker by eroding the equalized image using MATLAB's built-in function 'IMERODE' which takes an image and a structuring element. We used MATLAB's built-in function 'STREL' to define a disk-shaped structuring element with a size of 5. We then used MATLAB's built-in function 'IMRECONSTRUCT' to perform morphological reconstruction of the image marker under the equalized image. Then we used MATLAB's built-in function 'IMADJUST' to enhance the contrast of the reconstructed image.

Eroded Image

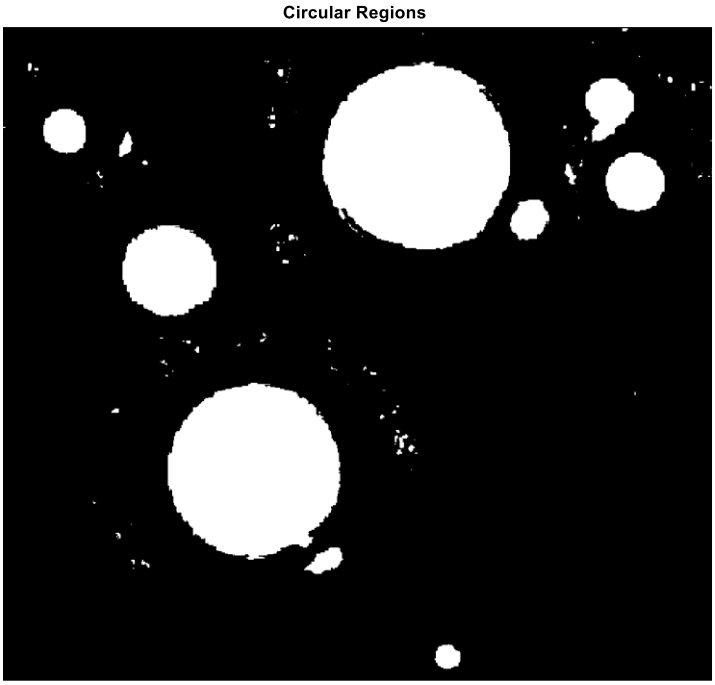




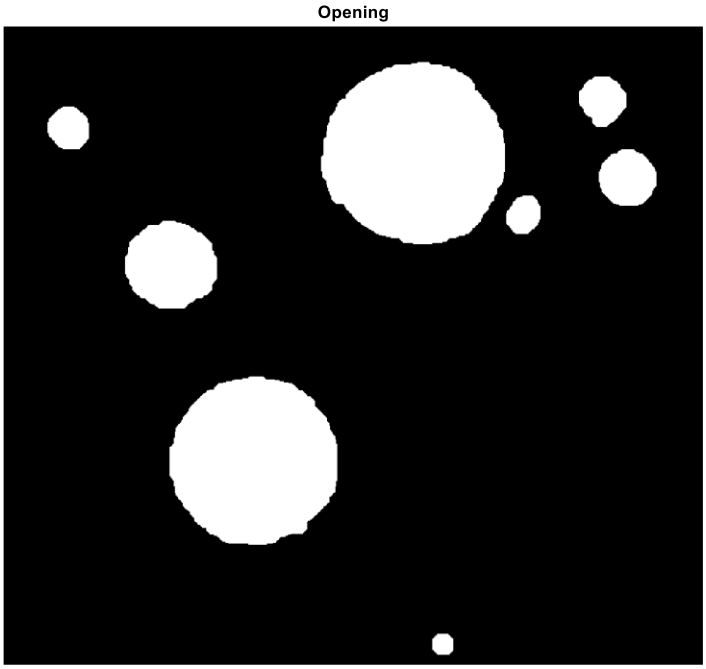
Binary Image: Resulting reconstructed image is then stacked with 6 more channels filtered at different sigma. The result image was then thresholded at threshold acquired from MATLAB's built-in function 'GRAYTHRESH' and then we took complement to set background black and foreground white. We then used MATLAB's built-in function 'IMFILL' to fill empty regions in the image.

Post-processing:

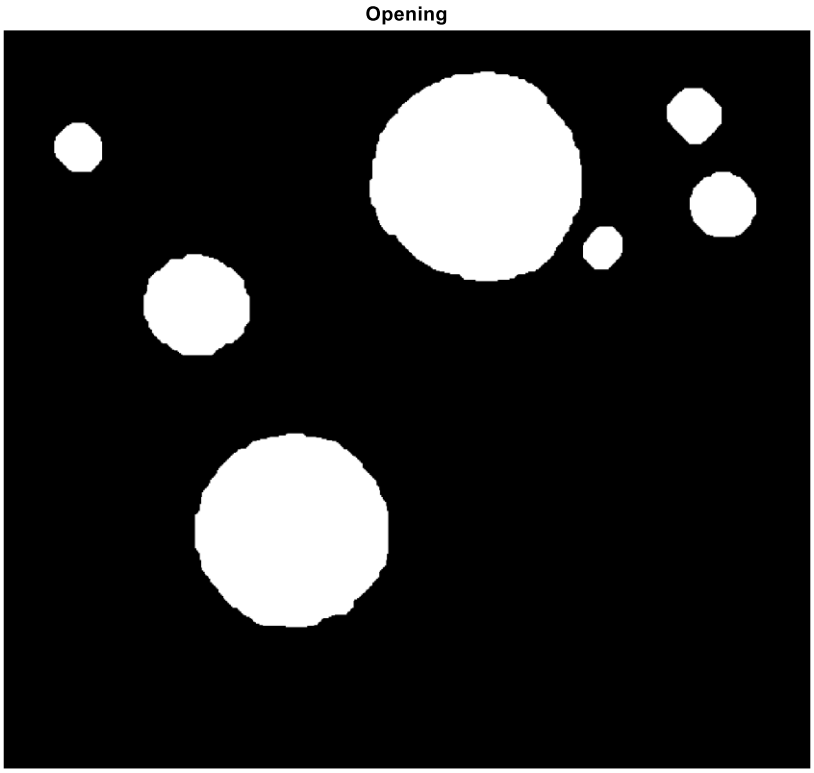
Circular Bodies: Under post-processing, as evident from our last result, with the help of MATLAB's built-in function 'REGIONPROPS' we were able to extract/segment circular bodies and eliminated the remaining objects.



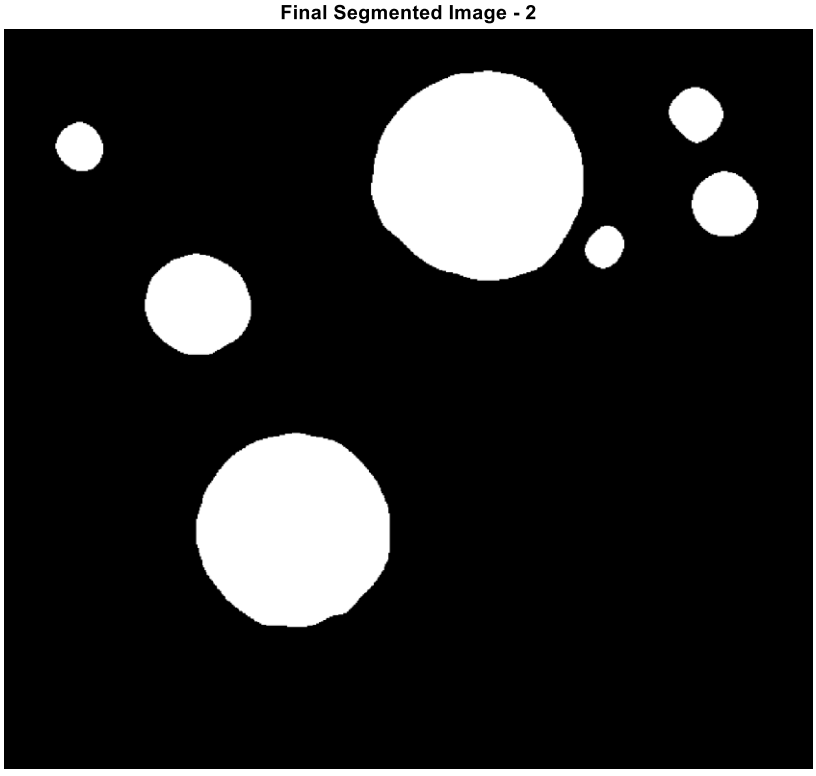
Morphologically Opening: As we had studied in our lecture that opening operation performs erosion followed by dilation where erosion first shrinks the non-cellular regions and then dilation expands the cellular regions to give the following results. We had defined a disk-shaped structuring element with the size of 10.



We performed opening once again to remove the non-cellular object at the bottom right which was first shrunk such that it was eliminated and then the remaining objects were dilated/expanded.



Boundary Smoothing: Once all the cellular regions were extracted then we smoothened the boundaries.



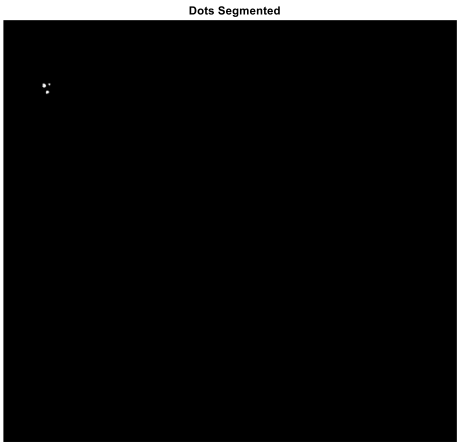
Part2: Cell Image Analysis (Spots detection, counting and density)

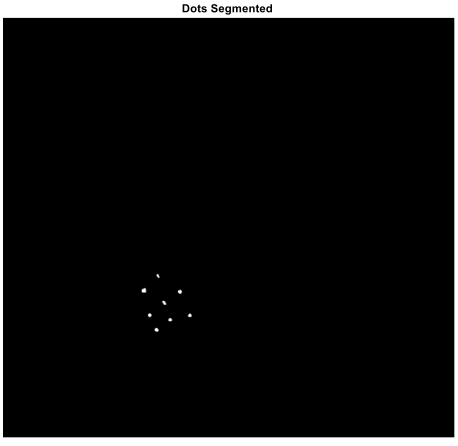
Task5: Label (bwlabel) the cell regions for individually accessing them in performing single cell analysis. Using the value of label, create a row vector to contain number of spots on each cell. Also create cell array to hold regional properties 'structure' for each cell. Now, in a loop (using labels), access/extract the individual cell regions from the labeled image such that the image contains only that region (use Logical operator). Extract regional properties (regionprops) of the extracted region. Now, use extracted object image as mask to extract the corresponding region/object of the original intensity image. Threshold that image, setting a threshold, e.g., between 75%-90% of maximum value, will help extract further features, e.g., spots within the cells. Labeling the thresholded image (which is actually spots in cells) would give the number of spots in that particular cellular region. Calculate the size(or area)-number of spots ratio of the cells, e.g., to highlight their health.

Methodology:

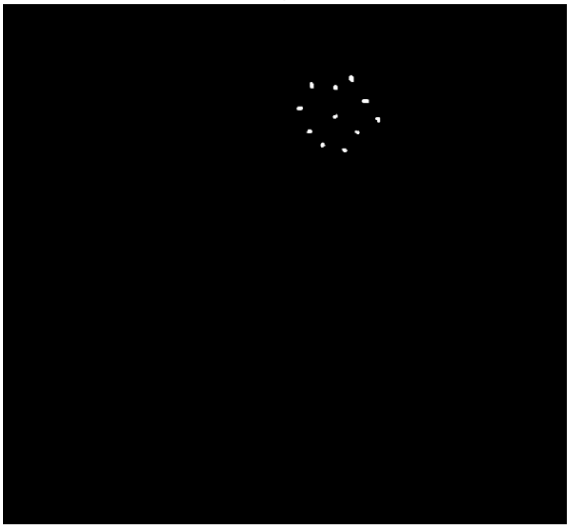
Firstly, we label all the connected bodies(cells), segmented in the image, by the help of MATLAB's built-in function 'BWLABEL'. We loop through the image to generate masks for each region which is then applied on the original image. The image is then converted to binary form with the help of MATLAB's built-in function 'IMBINARIZE' where the threshold is 0.8. The resulting binary image is then labeled and its regional(area) properties are acquired with the help of MATLAB's built-in function 'REGIONPROPS'. When the dots are detected in the region the number of dots in that region and the ratio between the area of dot and the total area of cell is computed and stored in cell_array.

Results:





Dots Segmented



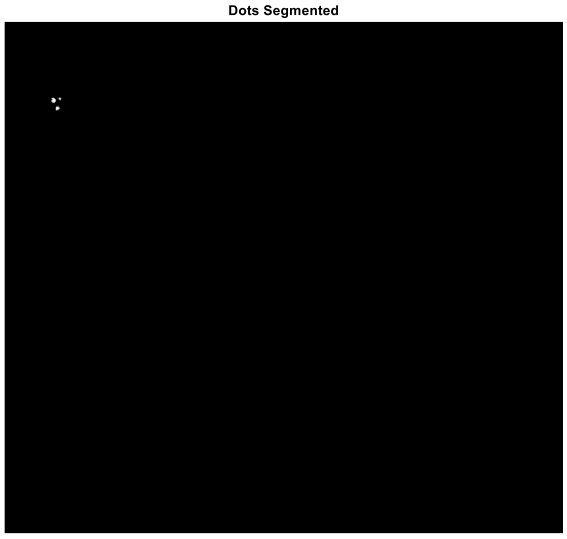
Dots Segmented

Dots Segmented

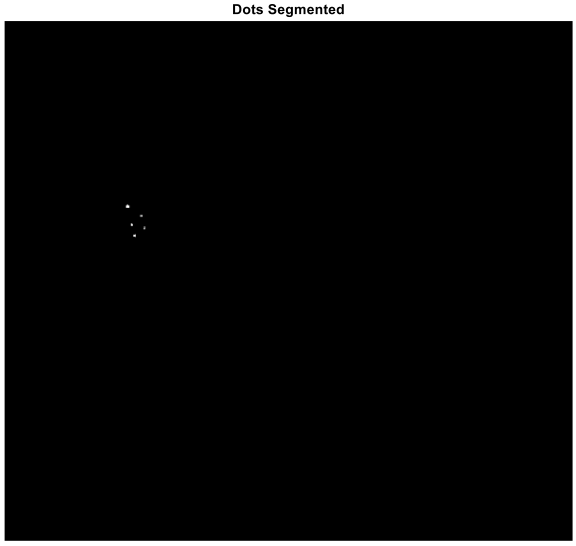
Number of cells in the original image = 6
Number of dots in each cell = 3 8 11 3 3 5
Ratio between the area of dot and cell: 0.0400 0.0081 0.0115 0.0410 0.0335 0.0390

Task6: Perform Task5 using the other segmentation result and comment on the accuracy of the results.

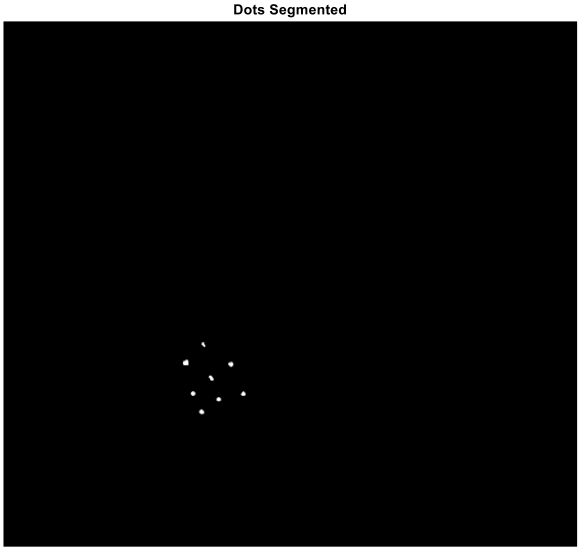
Results:



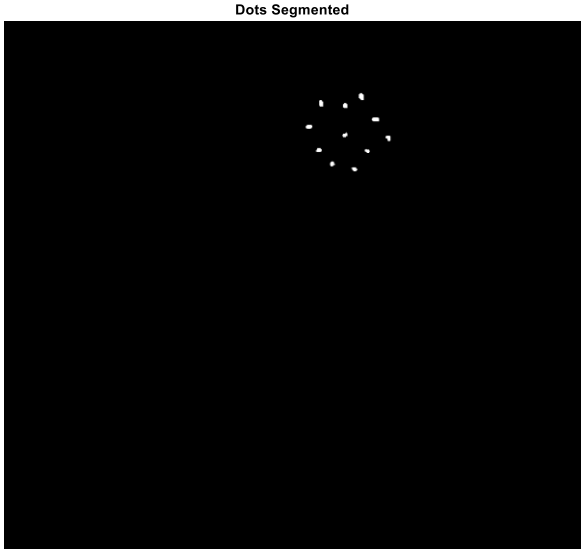
Dots Segmented



Dots Segmented

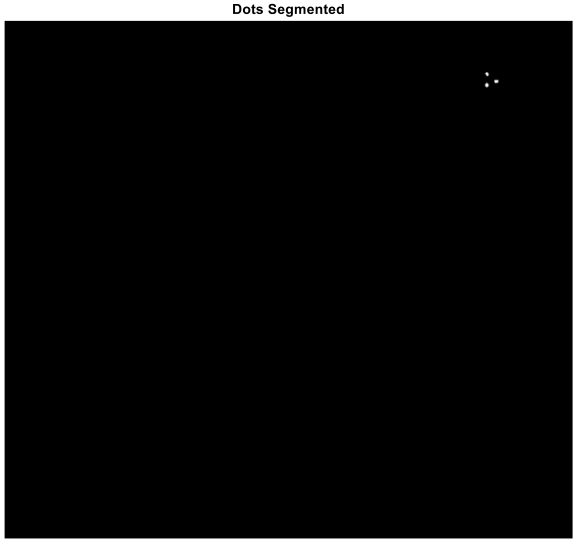


Dots Segmented



Dots Segmented

Dots Segmented



Dots Segmented

Number of cells in the original image = 7
Number of dots in each cell = 3 5 8 11 3 3 5
Ratio between area of dot and cell: 0.0368 0.0059 0.0089 0.0129 0.0470 0.0294 0.0380

Comment on accuracy:
Firstly, the number of regions/cells identified from the segmented image of task 5 is accurate since, there are total of 7 cells in the original image as well whereas, task 4 had only identified 6 cells. Secondly, the number of dots inside each region are also accurately identified that is the original image also has 3, 5, 8, 11, 3, 3, and 5 dots in 7 regions. Lastly, the ratio between the area of dot and cell acquired from both tasks' segmented image is approximately equivalent the difference is due to the difference in the pipeline of the segmentation.