ECE-565 Image Processing and Computer Vision

Karthikeyan Kumar A20376688

Question 1

Result:

1) The magnitude spectrum of the image outlines the intensity values seen in the image. The intensities are scattered widely and to a certain degree equally, with a low frequency component in the center. The inverse transform computed using only the phasor part of the image shows us the regions where the different elements of the image is present. It is noted that the phasor gives us the inference on the location aspect of the different components. In one sense, it can be said that the phasor transform gives us a view on how the sinusoidal are positioned, while the magnitude part gives the different energy levels associated with each such sinusoids. Hence the reconstructed image looks very similar to the input image, since the phase determines the location of the image elements.

2) The magnitude transform gives us the information regarding the energy concentration at each point. Basically, details of the energy components of the sinusoids. We infer that the intensity values for the low frequency pixels are higher than the intensity values of the higher frequency pixels. We can also understand that the intensity values are nearly zero in the locations of the input image where it is black/absence of image component. And the intensity values are more prominent in the regions where the image exists.

3) As we established before, the phasor aspect of the image is responsible for determining the position of the pixels in the image. When a conjugate is employed, we are essentially inversing the pixel positions and therefore the output is the original image flipped upside down.

Fig.1.1 Magniture spectrum of original image

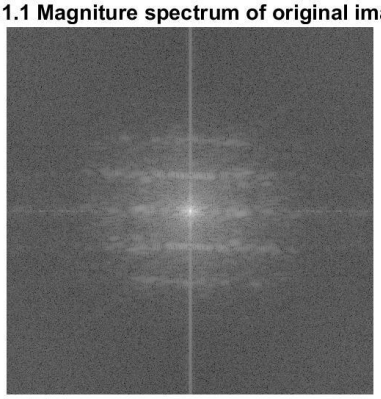


Fig.1.2 Phase spectrum of original image

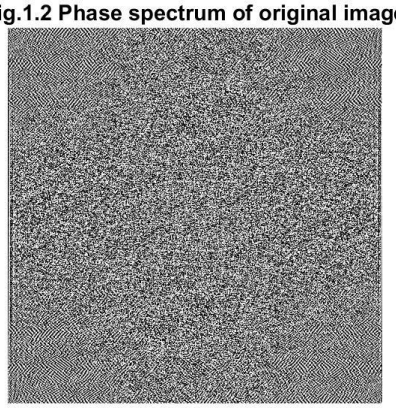


Fig.1.3 Reconstructed image using magnitude

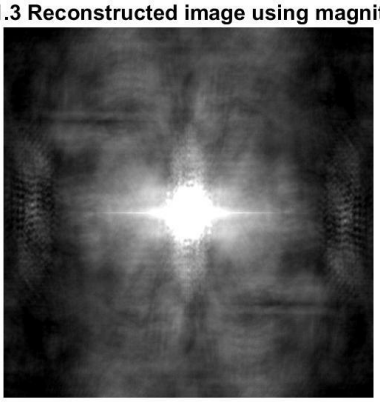


Fig.1.4 Reconstructed image using phase

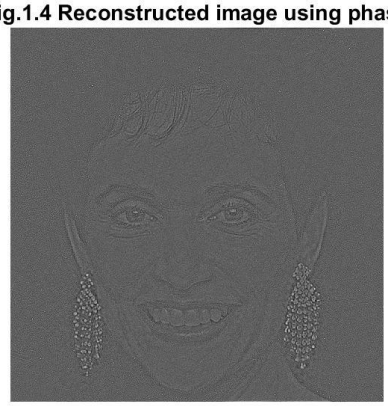
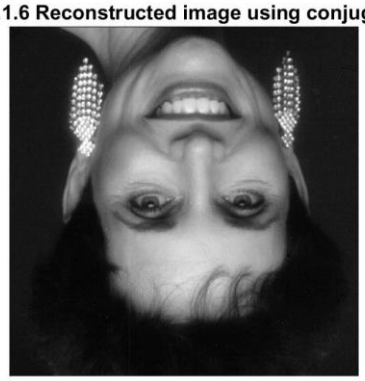


Fig.1.5 original image

Fig.1.6 Reconstructed image using conjugate



Question 2

Result:

Fig.2.1 is the original image. **Fig.2.2** shows the image using Sobel mask. It has better noise suppression characteristics. **Fig.2.3** shows the output image using 3*3 averaging filter. Before performing the gradient operation, we employ smoothing process to reduce the level of fine details which are undesirable. **Fig.2.4** and **Fig.2.5** shows the gradient image |gx|+ |gy| and histogram of the gradient image respectively. The thresholded version of the gradient image is displayed in **Fig.2.6** with Threshold value at 70. Any pixel above this value is white and below is black. We observe that there are fewer edges in the thresholded image and that the edges are much sharper, but the edges are at times discontinuous.

Fig.2.1 Input Image

Fig.2.2 Resultant image after Sobel mask

Fig.2.3 Image after smoothening

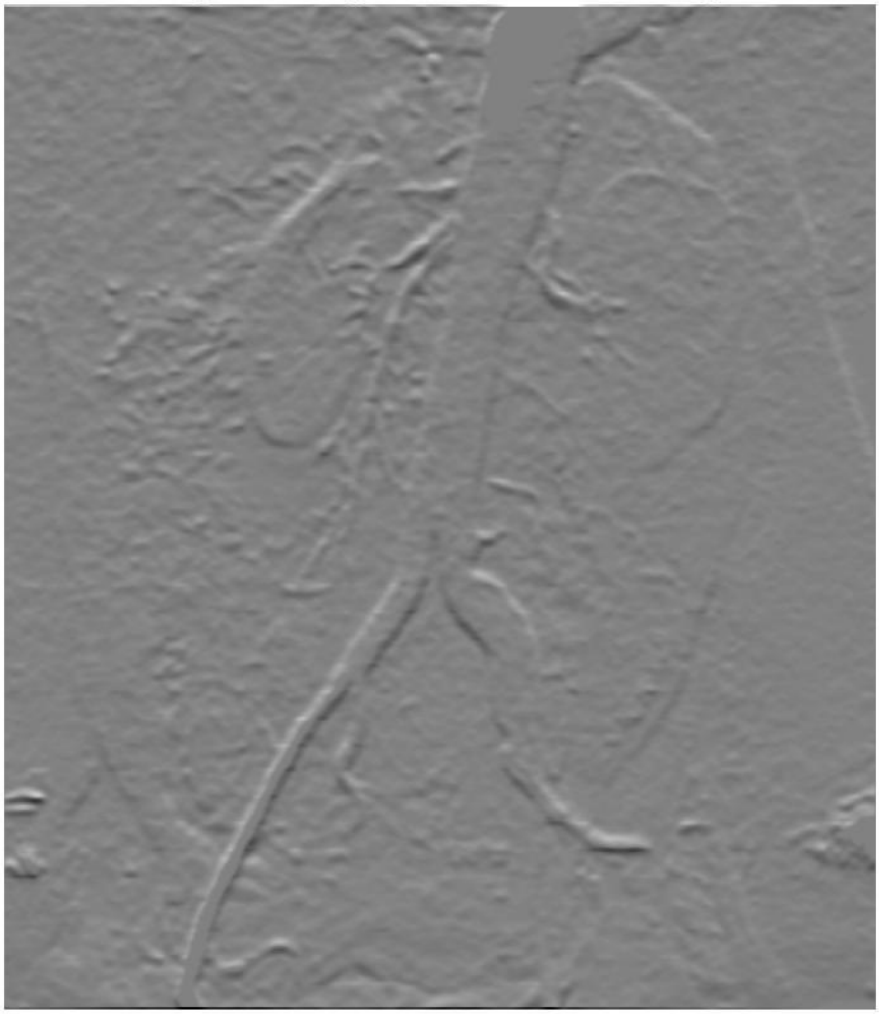


Fig.2.4 Gradiation Image

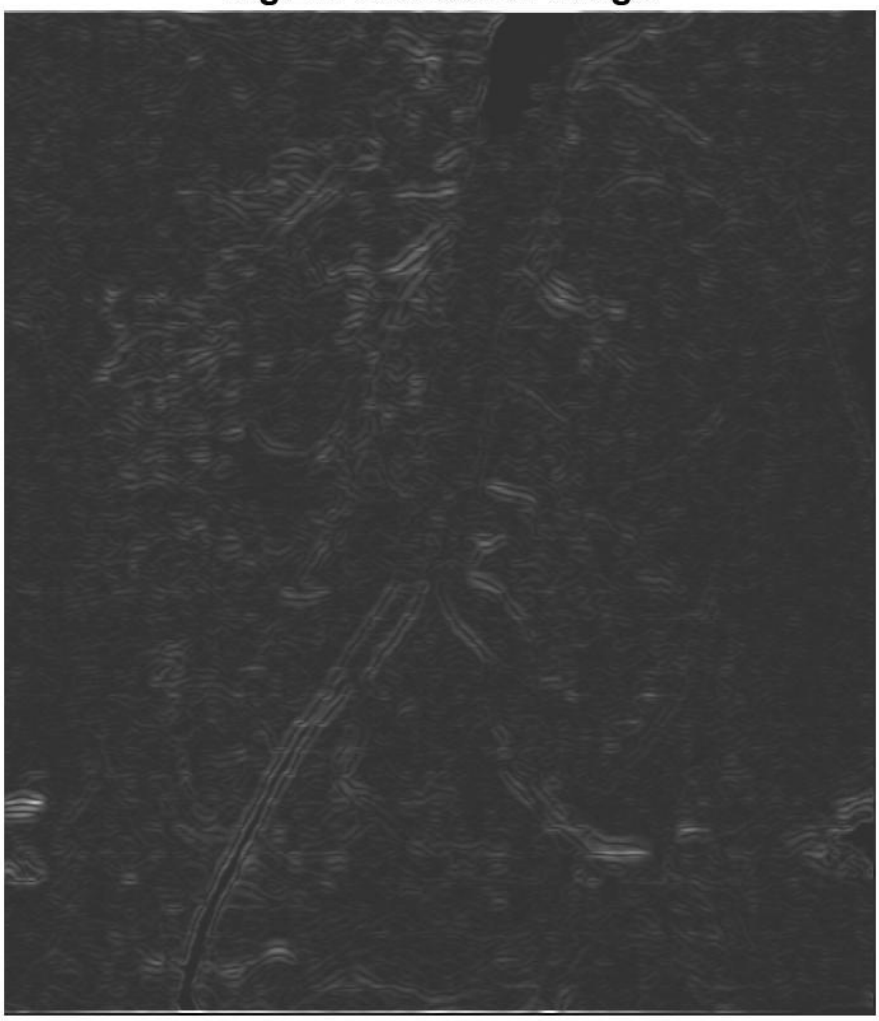


Fig.2.5 Histogram of Gradiation Image

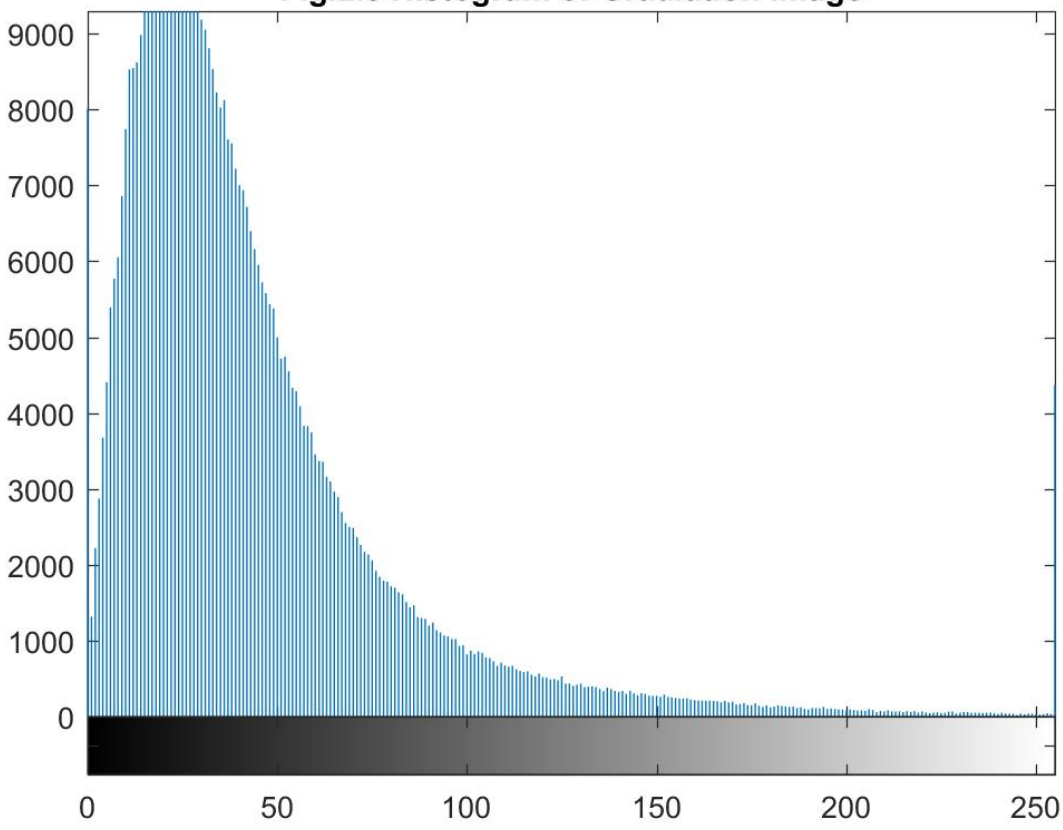


Fig.2.6 Image using Edge detection

Question 3

Result:

Fig.3.1 and **Fig.3.2** is the original image and the histogram respectively. We find that there is a distinct valley displayed in the histogram. After repeated iteration, the threshold value converges at 125.3860 owing to the iterative procedure.

150

127.6169

125.4045

125.3860

Fig.3.3 shows the result obtained using a global threshold to segment the original image. There is clear separation of peaks in the histogram, implying the segmentation between object and background was quite effective.

Fig.3.1 original image

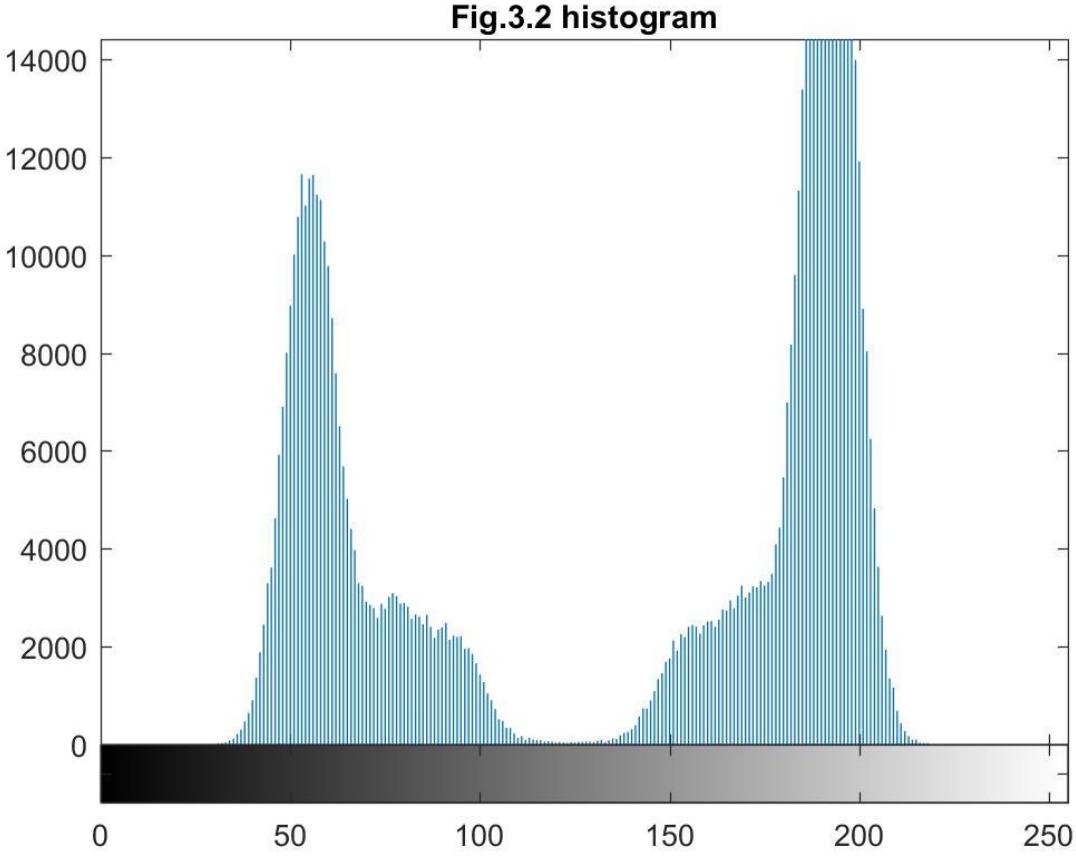


Fig.3.2 histogram
14000
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250

Fig.3.3 Segmented result using a global threshold

Question 4a and 4b:

Result:

Fig.4a.1 and **fig.4a.2** is the original image and the histogram respectively. The objective is to segment the molecules from the background. **Fig.4a.3** shows the result obtained using Otsu's method. The threshold value computed by Otsu's method was 182.

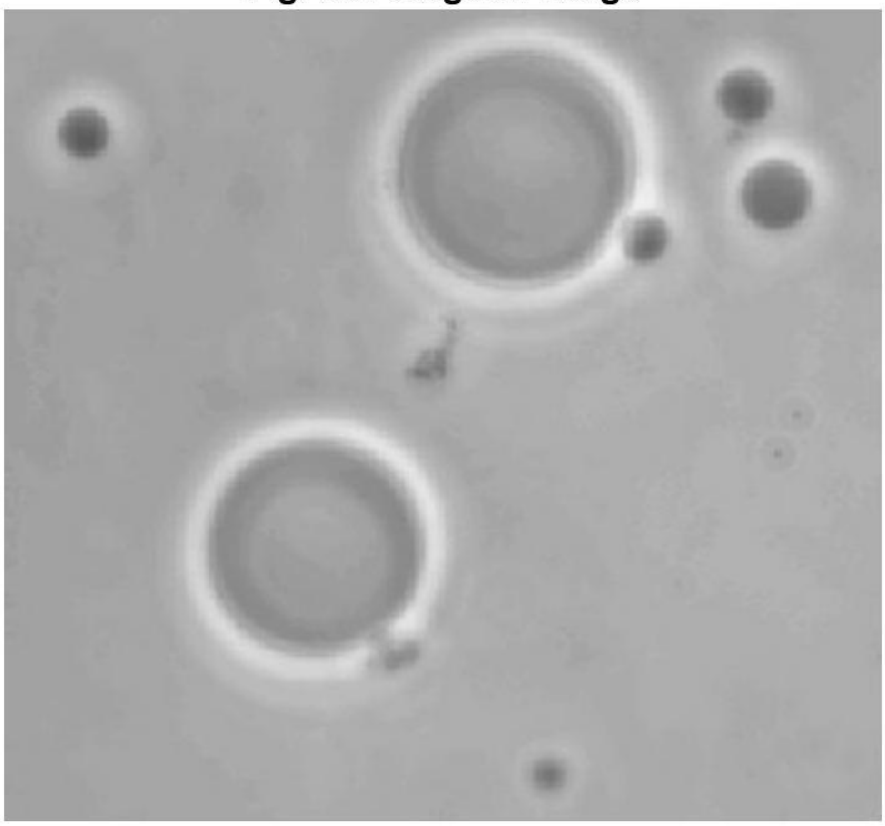
Fig.4b.1 and **Fig.4b.2** is the original image and image histogram respectively. The objective is to segment the molecules from the background. **Fig.3b.3** is the result of using the basic global thresholding algorithm from problem 3. After repeated iteration, the threshold value converges at 169.3950 owing to the iterative procedure.

169.6083

169.3950

Because the histogram has no distinct valley and the intensity difference between background and desired objects is small, the desired result was not achieved. **Fig.4a.3** shows the result obtained from Otsu's method, we can infer that the result is better than **fig.4b.3**. The threshold value computed by the basic global algorithm is 169.3950, while the threshold computed by Otsu's method is 182.

Fig.4a.1 Original Image



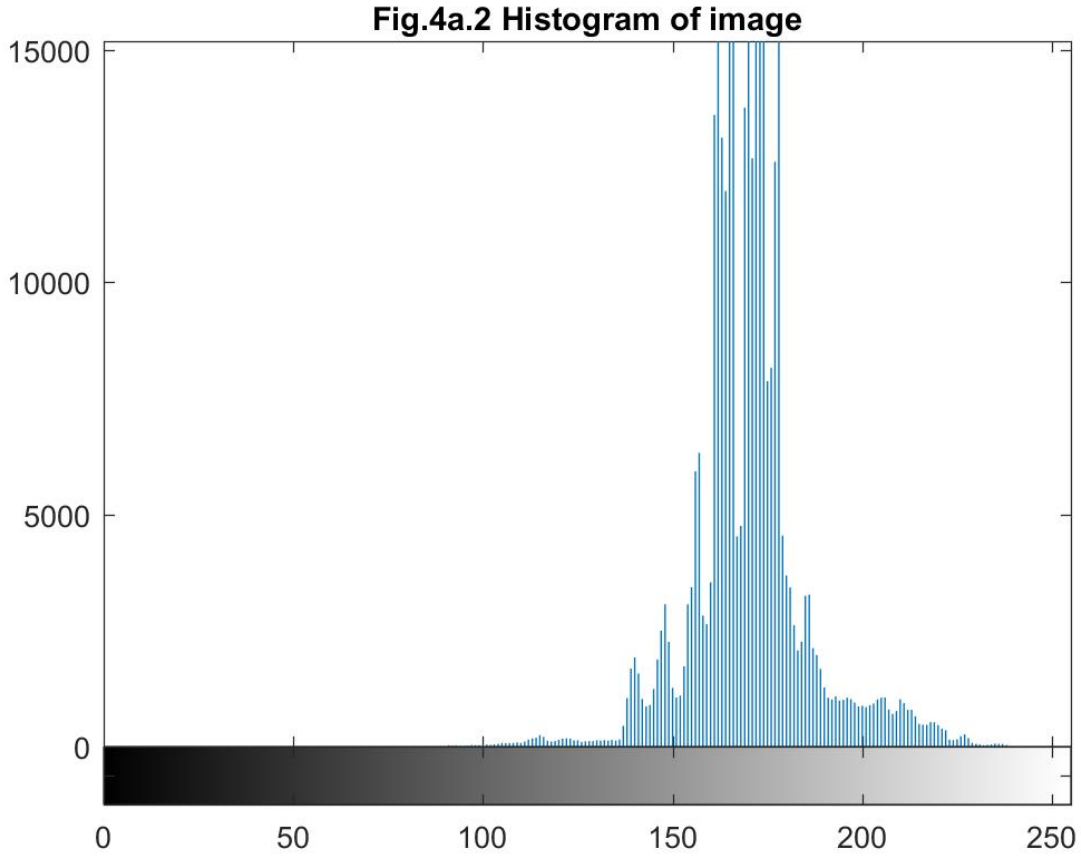


Fig.4a.2 Histogram of image
15000
10000
5000
0
0
50
100
150
200
250

Fig.4a.3 Otsu method output

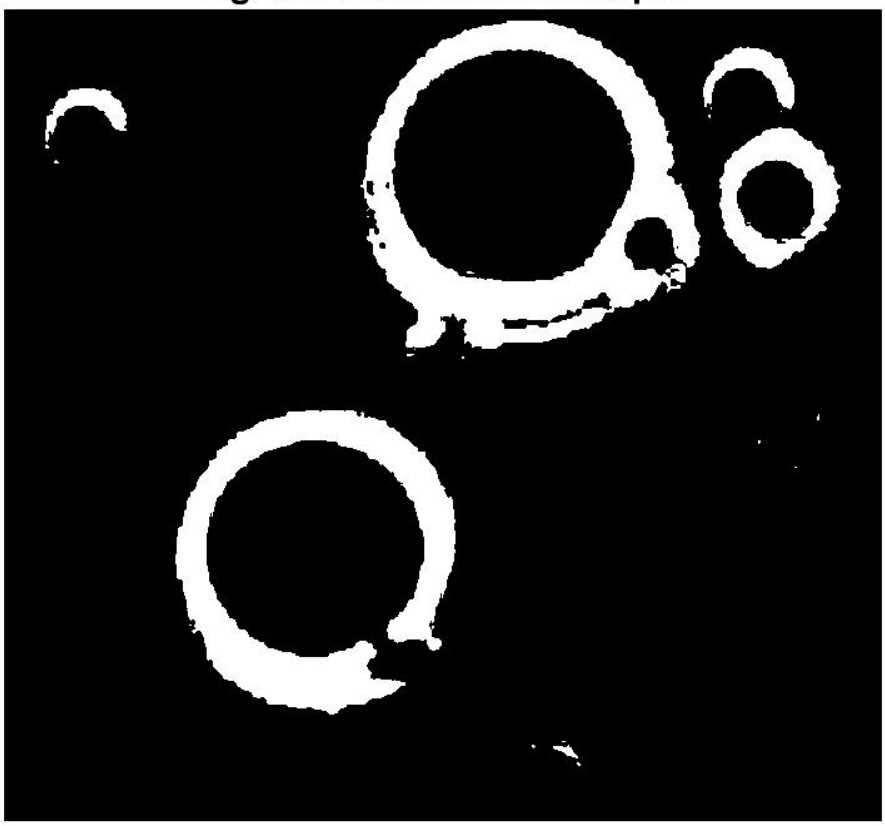
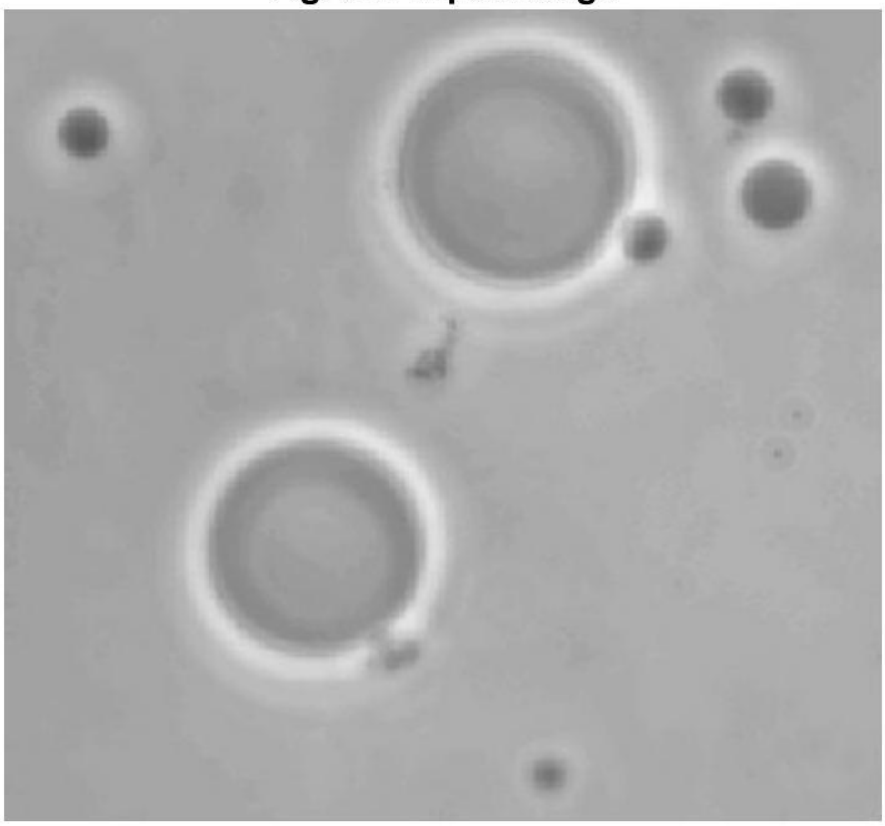


Fig.4b.1 Input Image



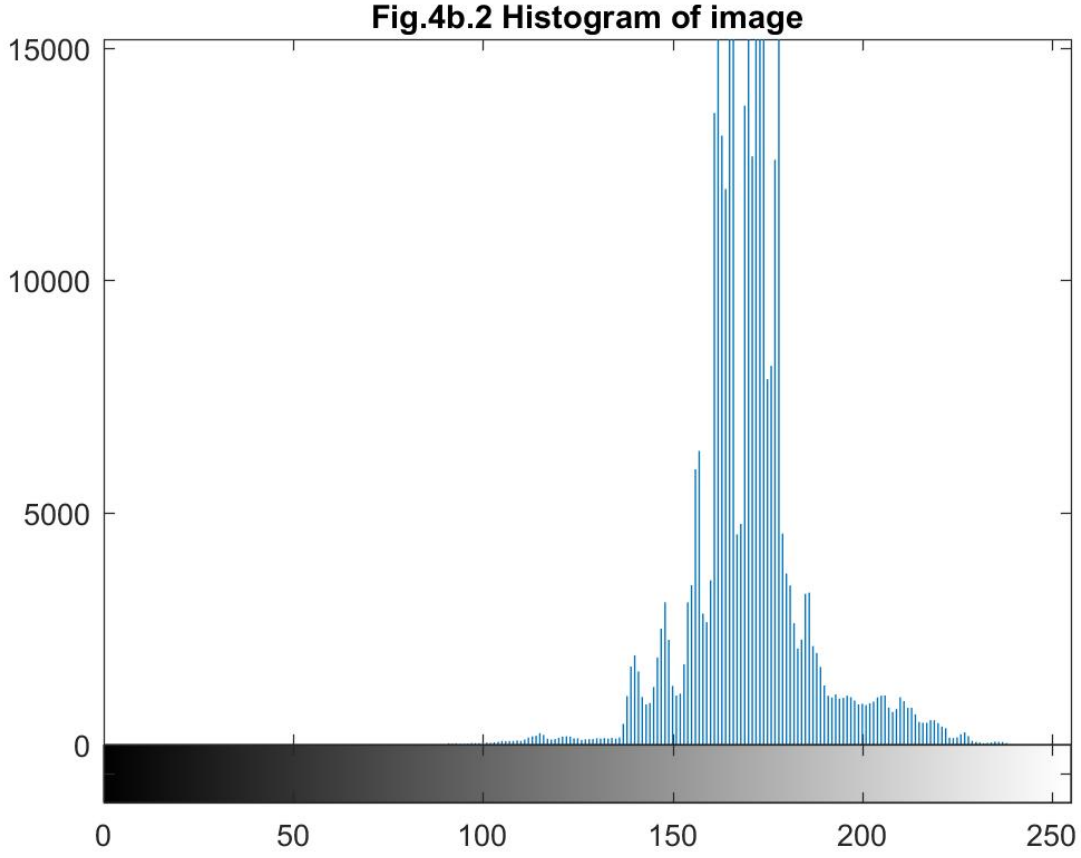


Fig.4b.2 Histogram of image
15000
10000
5000
0
0
50
100
150
200
250

Fig.4b.3 Result of the Global threshold

Appendix

```
%Loading the image
clc;clear;
i=imread(('E:\Coll work\Image processing\ece565-s18-
project1\woman.tif'),'tif');
F=fft2(double(i));  %compute the FFT of the image
mag=(abs(F));       %Separating the magnitude and phase part
mag1=fftshift(mag); %To shift the values to the enter
mag1=log(1+mag1);   %Purpose of scaling the values
phase=exp(1i*angle(F));
imag=ifftshift(ifft2((mag))); %Reconstruct the image from magnitude
iphase=ifft2(phase);          %Reconstructing using phase
inverseff=mag.*exp(-1i*angle(F)); %Reconstruction using conjugate
inverseff1=ifft2(inverseff);
%Image display
figure, imshow((mag1),[]),title('Fig.1.1 Magniture spectrum of
 original image');
figure,imshow((phase),[]),title('Fig.1.2 Phase spectrum of original
 image');
figure,imshow(uint8(imag),[]),title('Fig.1.3 Reconstructed image using
 magnitude');
figure,imshow((iphase),[]),title('Fig.1.4 Reconstructed image using
 phase');
figure,imshow(i),title('Fig.1.5 original image');
figure,imshow(inverseff1,[]),title('Fig.1.6 Reconstructed image using
 conjugate');
```

Published with MATLAB® R2016a

```
%Edge detection and Thresholding
clc;clear;
im=imread(('E:\Coll work\Image processing\ece565-s18-
project1\kidney.tif'),'tif');
figure,imshow(im);title('Input Image');
%[Row,Column]=size(im);
padding=zeros(794,690);
padding(3:end-2,3:end-2)=im; %zero padding
[paddedR,paddedC] = size(padding);
paddedTL = (3-1)/2;paddedBR = (3-1)/2;
sobelmask=[-1 -2 -1;0 0 0;1 2 1]; %sobel gradient mask
outputimage=zeros(paddedR,paddedC);
% spatial filtering
for i=3:792
for j=3:688
outputimage(i,j)=sum(sum(padding((i-paddedTL):(i+paddedBR),(j-
paddedTL):(j+paddedBR)).* sobelmask));
end
end
outputimage = outputimage(3:end-2,3:end-2);
figure,imshow(outputimage,[]);title('Resultant image after Sobel
 mask');
%[Row1,Column1] = size(outputimage);
padded1=zeros(794,690);
padded1(3:end-2,3:end-2)=outputimage;
[paddedR1,paddedC1] = size(padded1);
padTL1 = (3-1)/2;padBR1 = (3-1)/2;
smoothingfilter=input('Enter the filter weights as array');
sm1=sum(smoothingfilter);
sm2=sum(sm1);
smoothingfilter=smoothingfilter*(1/sm2); %3*3 smoothing filter
outputimage1=zeros(paddedR1,paddedC1);
% spatial filtering
for i = 3:792
for j = 3:688
outputimage1(i,j)=sum(sum(padded1((i-padTL1):(i+padBR1),(j-padTL1):(j
+padBR1)).* smoothingfilter));
end
end
outputimage1 = outputimage1(3:end-2,3:end-2);
figure,imshow(outputimage1,[]);title('Image after smoothening');
for i=1:size(outputimage1,1)-2
for j=1:size(outputimage1,2)-2
%Sobel mask for x-direction:
Gx=((2*outputimage1(i+2,j+1)+outputimage1(i+2,j)+outputimage1(i+2,j
+2))-(2*outputimage1(i,j+1)+outputimage1(i,j)+outputimage1(i,j+2)));
%Sobel mask for y-direction:
Gy=((2*outputimage1(i+1,j+2)+outputimage1(i,j+2)+outputimage1(i+2,j
+2))-(2*outputimage1(i+1,j)+outputimage1(i,j)+outputimage1(i+2,j)));
%The gradient of the image
outputimage1(i,j)=sqrt(Gx.^2+Gy.^2);
end
```

```
end
figure,imshow(outputimage1,[]);title('Gradiation Image' );
outputimage1=uint8(outputimage1);
figure,imhist(outputimage1);title('Histogram of Gradiation Image');
Thresh=70;
outputimage1=max(outputimage1,Thresh);
outputimage1(outputimage1==round(Thresh))=0;
figure,imshow(outputimage1,[]);title('Image using Edge detection');
```

Published with MATLAB® R2016a

```
%Global thresholding
clc;clear;
i=imread(('E:\Coll work\Image processing\ece565-s18-
project1\noisy_fingerprint.tif'),'tif');
figure,imshow(i);title('Fig.3.1 original image');
[hist,x]=imhist(i);
figure,imhist(i);title('histogram');
T=150; %initial estimate for global threshold
while true
disp(T);
value1=i(i>T);value2=i(i<=T);
mean1=sum(value1)/length(value1);mean2=sum(value2)/length(value2);
newthresh=(mean1+mean2)/2;
    if T==newthresh
        break;
    else
        T=newthresh;
    end
end
i(i<newthresh)=0;i(i>=newthresh)=1;
figure,imshow(i,[]);title('Fig.3.2 Segmented result using a global
 threshold');
```

Published with MATLAB® R2016a

```
%Otsu’s optimum thresholding
clc;clear;
I=imread(('E:\Coll work\Image processing\ece565-s18-
project1\polymersomes.tif'),'tif');
figure,imshow(I);title('Fig.4a.1 Original Image');
[hist,x]=imhist(I);
figure,imhist(I);title('Fig.4a.2 Histogram of image');
threshold=256;
counts=imhist(I,threshold);
mean=counts / sum(counts);
mean1=0;mean2=0;mu1=0;
for t=1:threshold
mean1(t)=sum(mean(1:t));
mean2(t)=sum(mean(t+1:end));
mu1(t)=sum(mean(1:t).*(1:t)');
end
sigma =(mu1(end).*mean1-mu1).^2./(mean1.*(1-mean1));
[row,column]=max(sigma);
I(I<column)=0;I(I>=column)=1;
figure,imshow(I,[]);title('Fig.4a.3 Otsu method output');
```

Published with MATLAB® R2016a

```
%Basic global thresholding
clc;clear;
i=imread(('E:\Coll work\Image processing\ece565-s18-
project1\polymersomes.tif'),'tif');
[row1,column1]=size(i);
figure,imshow(i);title('Fig.4b.1 Input Image');
[hist,x]=imhist(i);
figure,imhist(i);title('Fig.4b.2 Histogram of image');
s=row1*column1;
threshold=sum(sum(i))/s; %initial estimate for global threshold
while true
disp(threshold);
value1=i(i>threshold);value2=i(i<=threshold);
mean1=sum(value1)/length(value1);
mean2=sum(value2)/length(value2);
newthresh=(mean1+mean2)/2;
    if threshold==newthresh
        break;
    else
        threshold=newthresh;
    end
end
i(i<newthresh)=0;i(i>=newthresh)=1;
figure,imshow(i,[]);title('Fig.4b.3 Result of the Global threshold');
```

Published with MATLAB® R2016a