Solution to Homework 3 (by Nafi Rashid)

1. Obtaining the histogram is fairly straight forward, I used these commands in Matlab:

```
P1 = imread('retina1.tif');
bw = rgb2gray(P1);
bw(bw<50) = 0;

[count l] = imhist(bw);
count(1) = 0; %% just to ignore the black pixels surrounding the retina
stem(count);
```

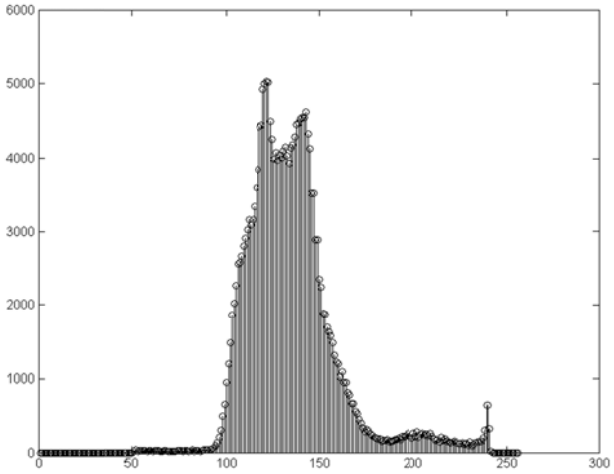


Figure 1: Histogram of P1

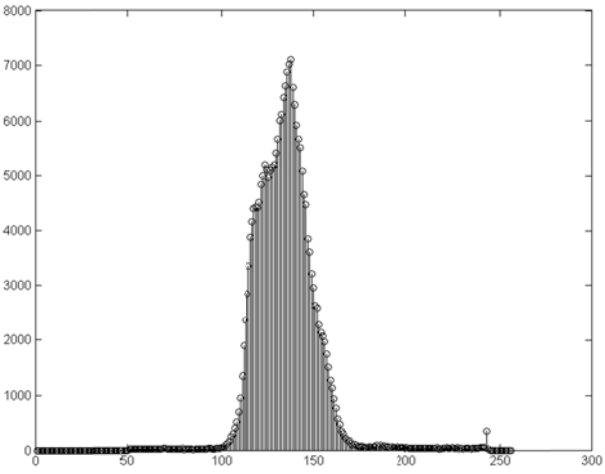


Figure 2 Histogram of P2

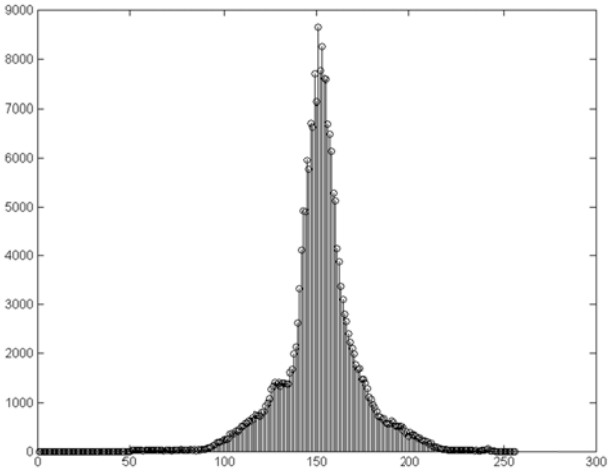
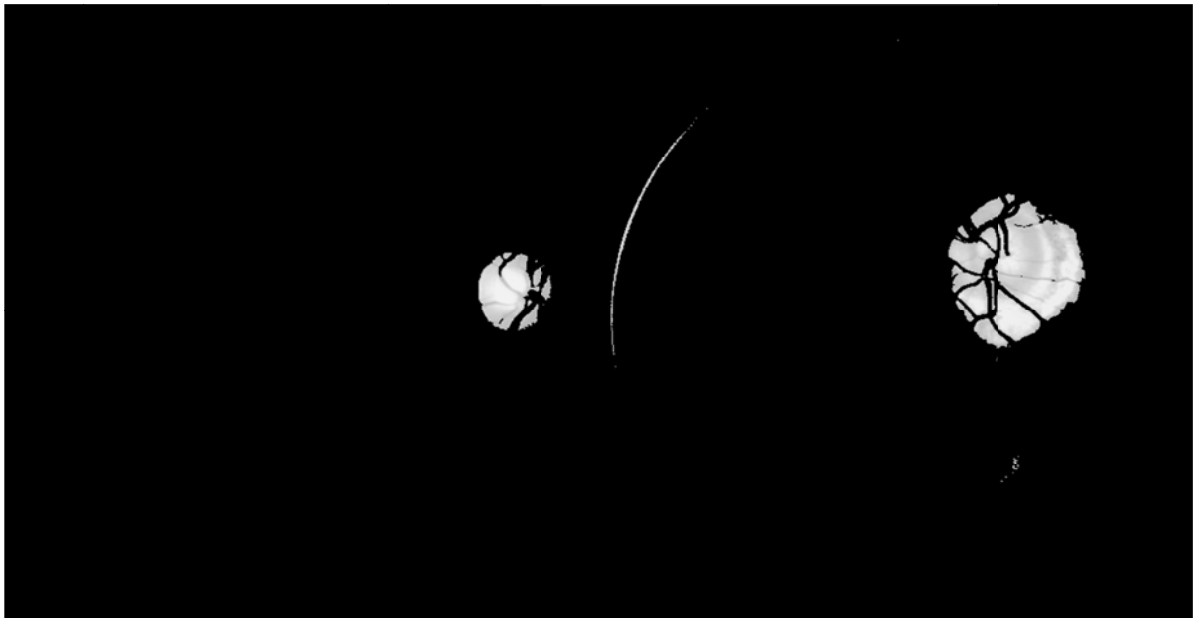


Figure 3: Histogram of P3

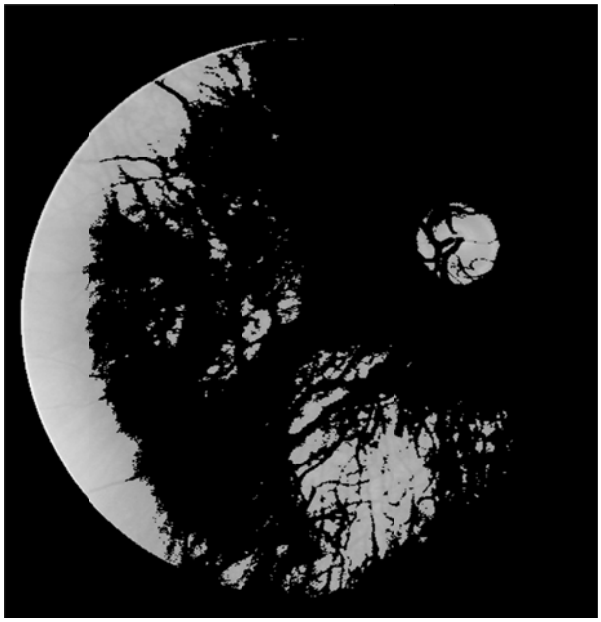
We can also observe that Obj2 has higher pixel values in the grayscale image. So, it is safe to assume that the peak near 240 in histogram of P1 and P2 corresponds to Obj2. In contrast, the histogram of P3 doesn't have a distinct peak at higher pixel values. By looking at the histogram, it can be said that same threshold will not work for every image. Moreover, for P3 it would be harder to segment Obj2. This fact is verified in figure 4 wherein the segmented Obj2 of these 3 images is shown. It is apparent that only pixel-value based segmentation didn't work well for P3. This is because, in P3, the background has pixels that have values closer to Obj2.

This is the command I used for thresholding:

```
 %% thresholding
mask2 = im2bw(P1, threshold /255);
imshow(P1.*uint8(mask2));
```



(a)



(b)

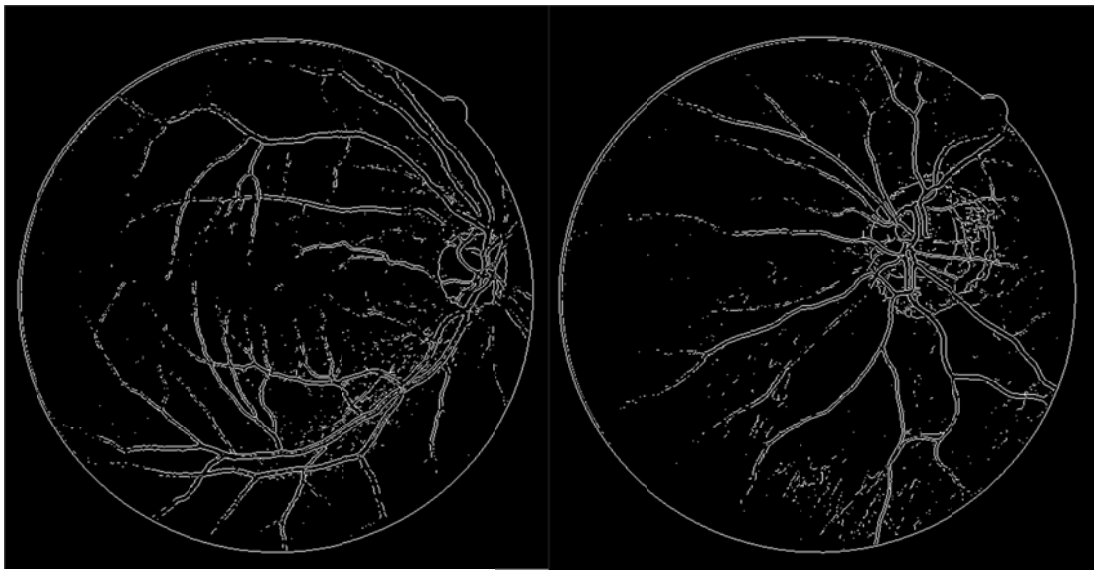
(c)

Image	threshold
P1	180
P2	180
P3	160

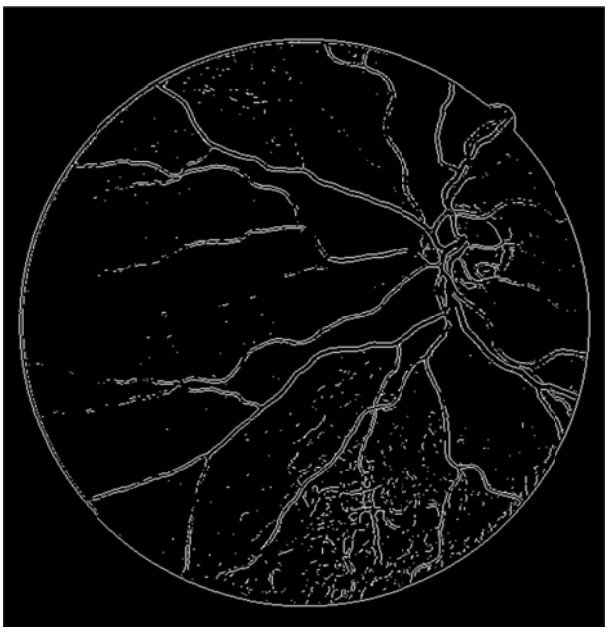
Figure 4: (a) P1 (b) P2 (c) P3 after thresholding

2. To obtain the sobel edge map I used the following command:

```
sobelE = edge(bw, 'sobel', edgeThreshold);
figure,imshow(sobelE);
```



(a) (b)



(C)

Figure 4: Binary edge map of (a) P1 (b) P2 (c) P3

The same threshold doesn't work for getting these three edge maps. This is dependent on the texture of the image. As for P3, the pixels are closely distributed, so we need to use a smaller threshold.

3. Separating these two objects in the Edge map can be done in a few ways. I used the edge map to check the boundary pixel in the grayscale image.

```
E2 = sobelE;
[dimy dimx] = size(sobelE);
[row col] = find(sobelE ==1);
for y = 1:length(row)
    x=y;
    if(bw(row(y),col(x))<153)
        E2(row(y),col(x)) = 0;
    end
end
E1 = xor(sobelE,E2);
subplot(121),imshow(E2)
subplot(122),imshow(E1);
```

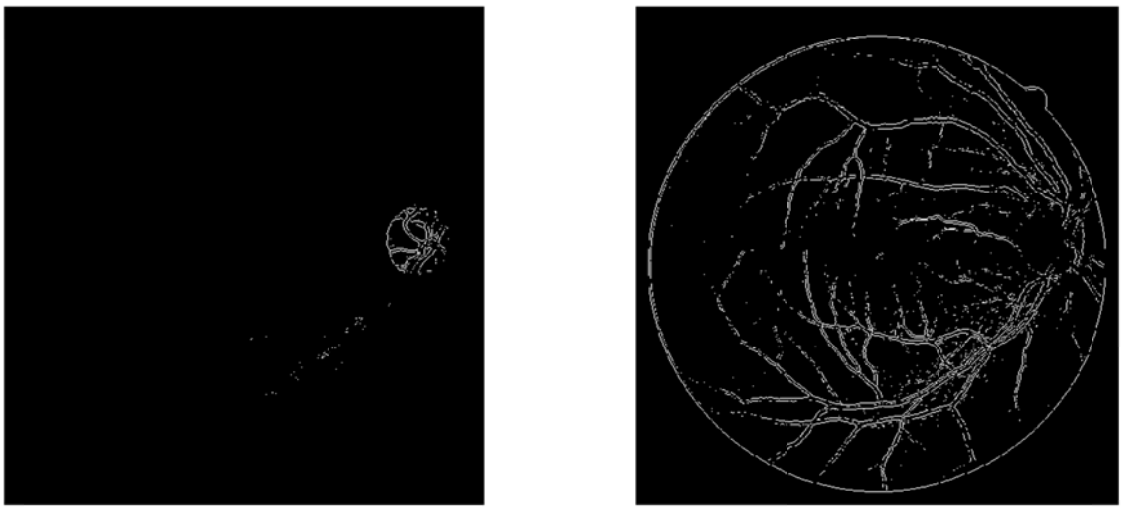


Figure 5: E1 and E2 of P1

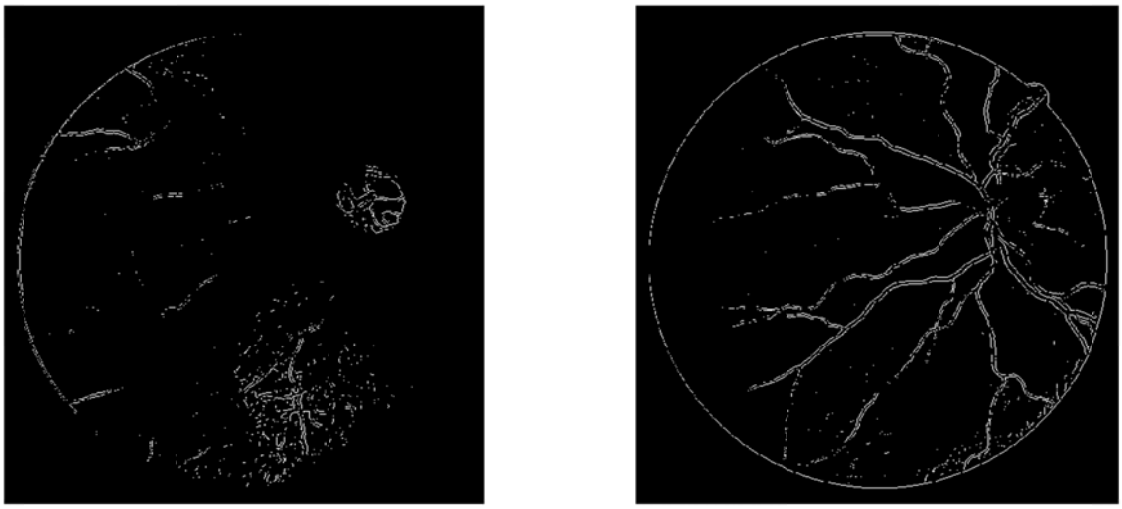


Figure 6: E1 and E2 of P2

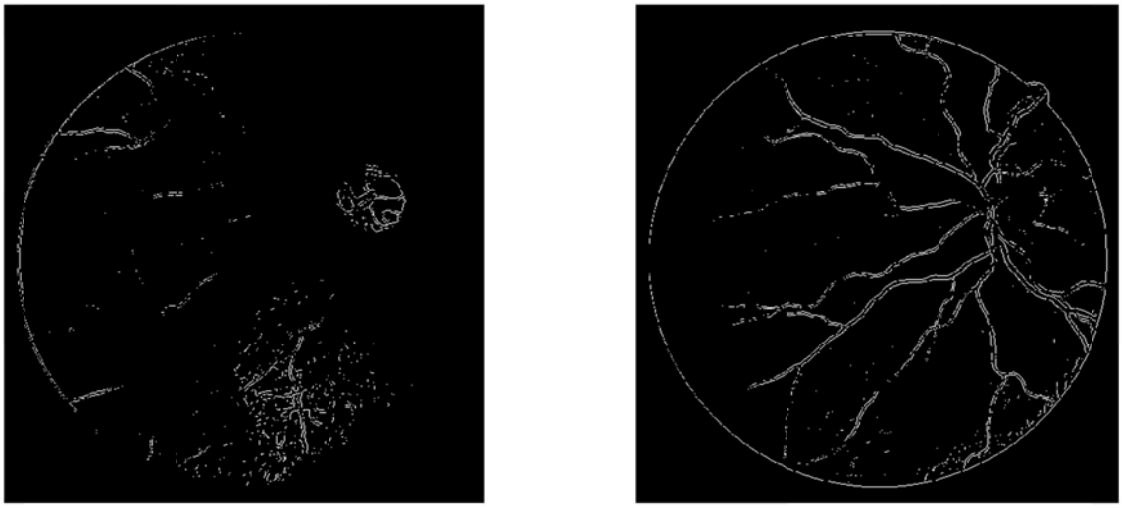
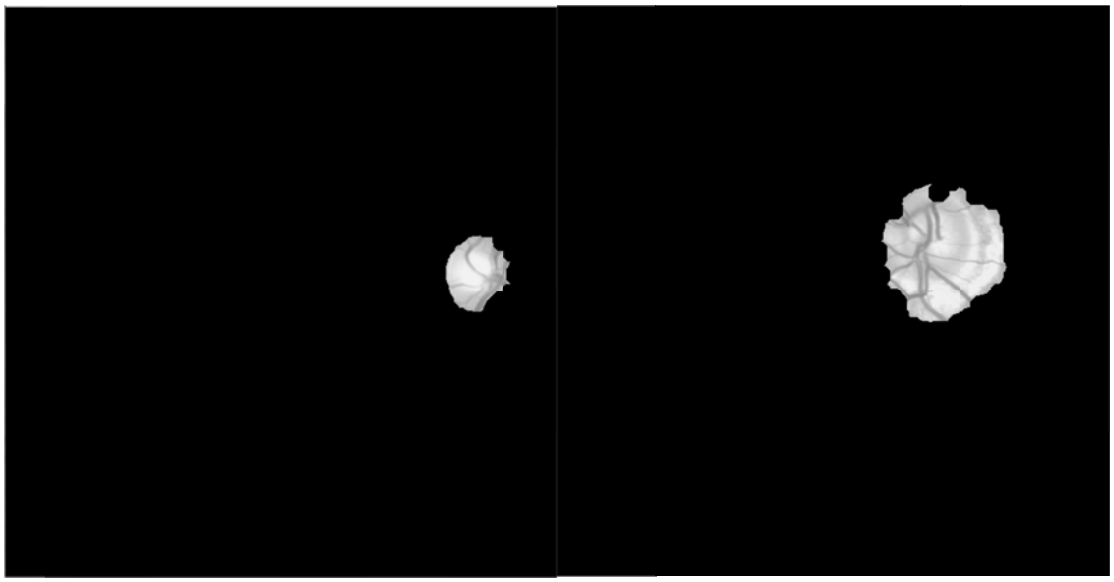


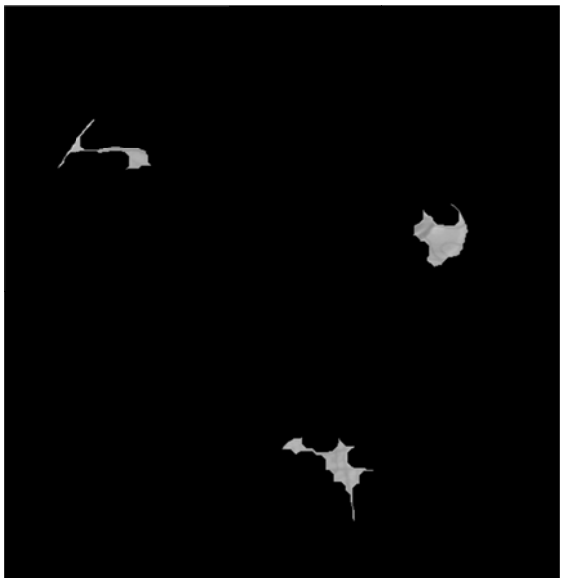
Figure 7: E1 and E2 of P3

Obtaining segmentation mask from E2 can be done in many ways. I used the following operations:

```
Bw1 = bwareaopen(E2,10);
se = strel('disk',10);
bw2 =imclose(bw1,se);
bw3 = imfill(bwRemove2,'holes');
figure, imshow(bw.*uint8(bw3));
```



(a) (b)



(c)

Figure 7: Segmenting Obj2 using the edge map E2 of (a) P1 (b) P2 (c) P3

Comparing figure 7 with figure 4, we can see that for P3, edge-based segmentation gave better result although the segmentation is far from perfect. Note that in contrast to pixel-value based segmentation edge extraction tend to produce the boundary of the object.

4. For the last part of this homework, I used matlab function hough() to find the lines in E1. The strongest 5 lines correspond to highest 5 peaks in the accumulator array return by the hough() function. Here is the code I used:

```
[H,T,R] = hough(E1);
P  = houghpeaks(H,5);
imagesc(H,'XData',T,'YData',R );
```

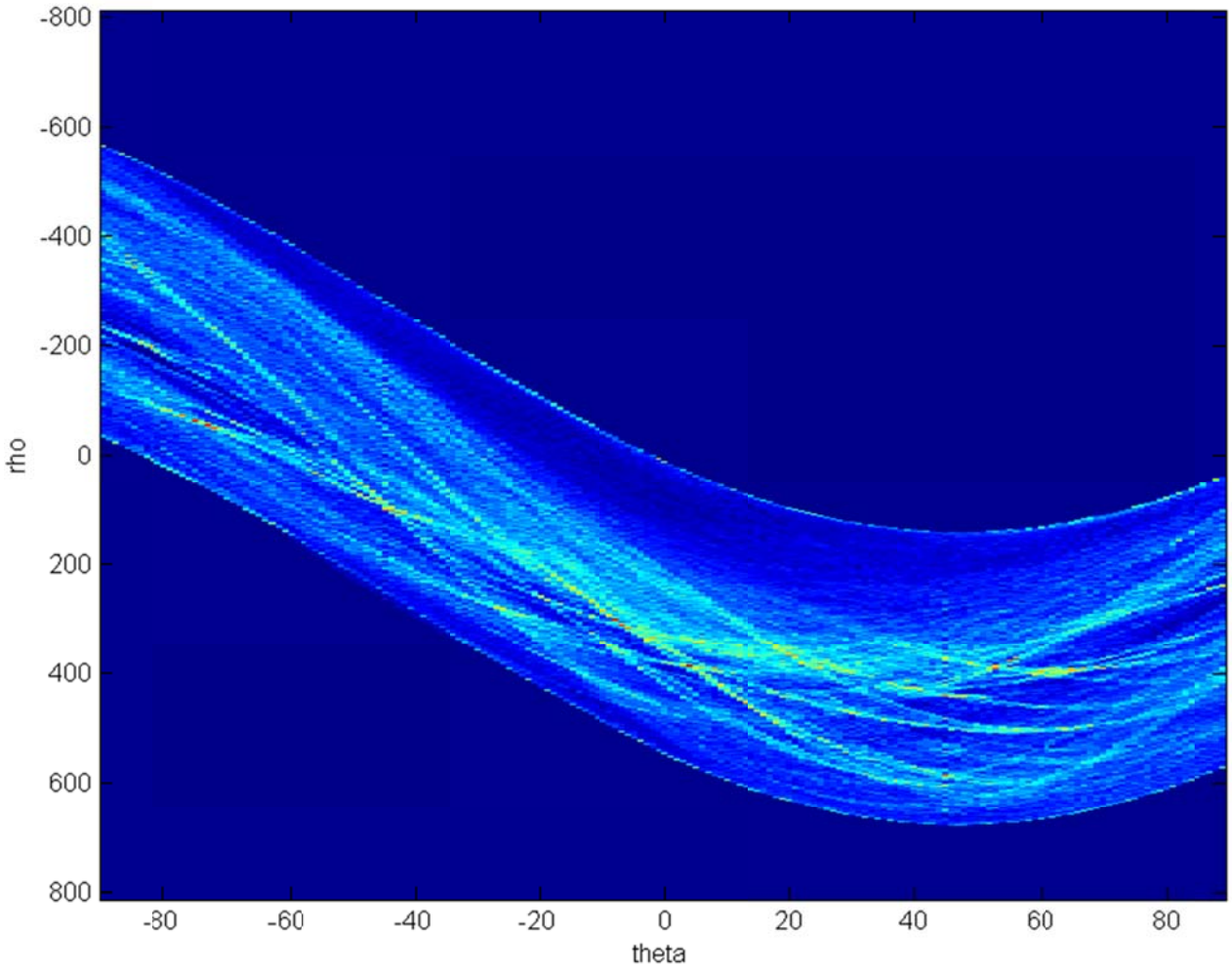


Figure 8: Accumulator array returned by the hough transform

In figure 8, we can see how the accumulator array of hough transform looks like. The red pixels indicate peaks. To obtain the 5 strongest peaks, I used the function houghpeaks(). And finally, I used houghlines() to translate the lines back to the original image:

```
P  = houghpeaks(H,5);
lines = houghlines(E1,T,R,P,'FillGap',40,'MinLength',10);
figure, imshow(bw), hold on

for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
end
```

Figure 9: 5 strongest lines drawn on the image

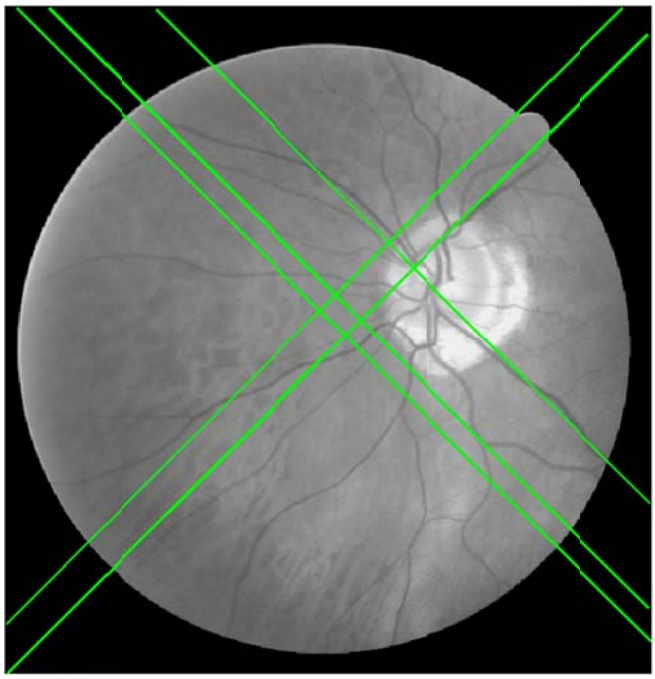


Figure 10: 5 strongest lines drawn on the image

In figure 9, we can see that the lines indicated by Hough represent the blood vessels fairly accurately. But if we add 15% salt and pepper noise, the Hough transform completely fails. This is because there are many random outliers in the edge map due to noise. These outliers do not let the accumulator to have distinct peaks and thus lines cannot be detected accurately.