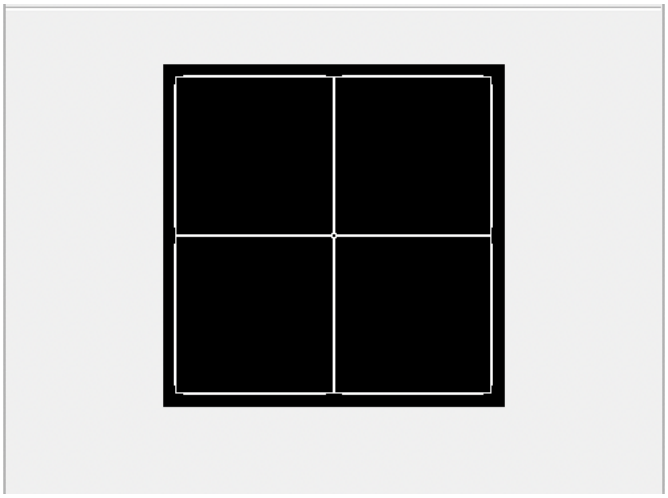
1.

a.



ps2-1-a-1.png

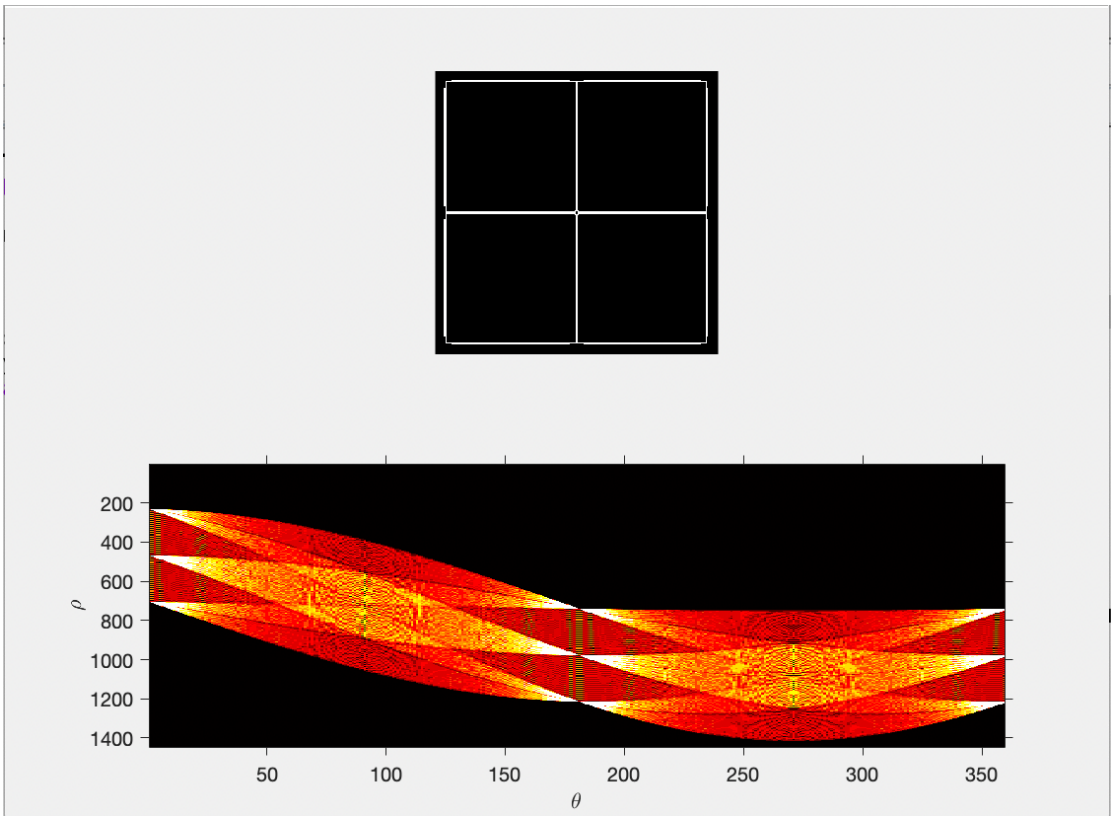
2.

a.

+2 | hough_lines_draw.m | hough_lines_acc.m | hough_peaks.m | p

```
1  function [H, theta, rho] = hough_lines_acc(img_edges);
2  BW = img_edges;
3  theta = (-90:0.5:89); %initilize theta;
4  [M,N] = size(img_edges);
5  D = sqrt((M-1)^2+(N-1)^2);
6  rhoResolution = 0.5;
7  Bound = ceil(D/rhoResolution) * rhoResolution;
8  rho = (-Bound:rhoResolution:Bound); %initilize tho;
9  H = zeros(length(rho),length(theta)); %initilize H;
10 cost = cos(theta*pi/180);
11 sint = sin(theta*pi/180);
12 %get the value of H matrix
13 for y = 1:M
14     for x = 1:N
15         if BW(y,x)
16             r = (x-1)*cost+(y-1)*sint;
17             r = round(r/rhoResolution)+ceil(D/rhoResolution)+1;
18             for k = 1:length(theta*pi/180)
19                 H(r(k),k) = H(r(k),k)+1;
20             end
21         end
22     end
23 end
24 end
```

hough_lines_acc()



ps2-2-a-1.png

b.

+8 hough_peaks.m | ps2_3_c.m | hough_lines_acc.m | hough_li

```
1  function [p] = hough_peaks(H,numpeaks);
2  H1 = H;
3  %find the max value and it's(x,y)
4  max1 = max(max(H1));
5  [x1,y1] = find(H1 == max1);
6  p = zeros(numpeaks,2);
7  l0 = 1;
8  while (l0 <= numpeaks)
9  %find the max value and it's(x,y)
10     max1 = max(max(H1));
11     [x1,y1] = find(H1 == max1);
12     l1 = length(x1);
13     l2 = l0+l1-1;
14     l4 = size(x1);
15     %if numbers of value > numpeaks, then delete the excess
16     if(l2 > numpeaks)
17         l2 = numpeaks
18         l4 = numpeaks-l0+1;
19     end
20     %put the value in matrix p
21     p(l0:l2,1) = x1(1:l4,:);
22     p(l0:l2,2) = y1(1:l4,:);
23     %make the maxvalue equal to 0 so that I can get the second largest one
24     for a = l0:l2
25         H1(p(a,1),p(a,2)) = 0;
26     end
27     l0 = l0+l1;
28 end
```

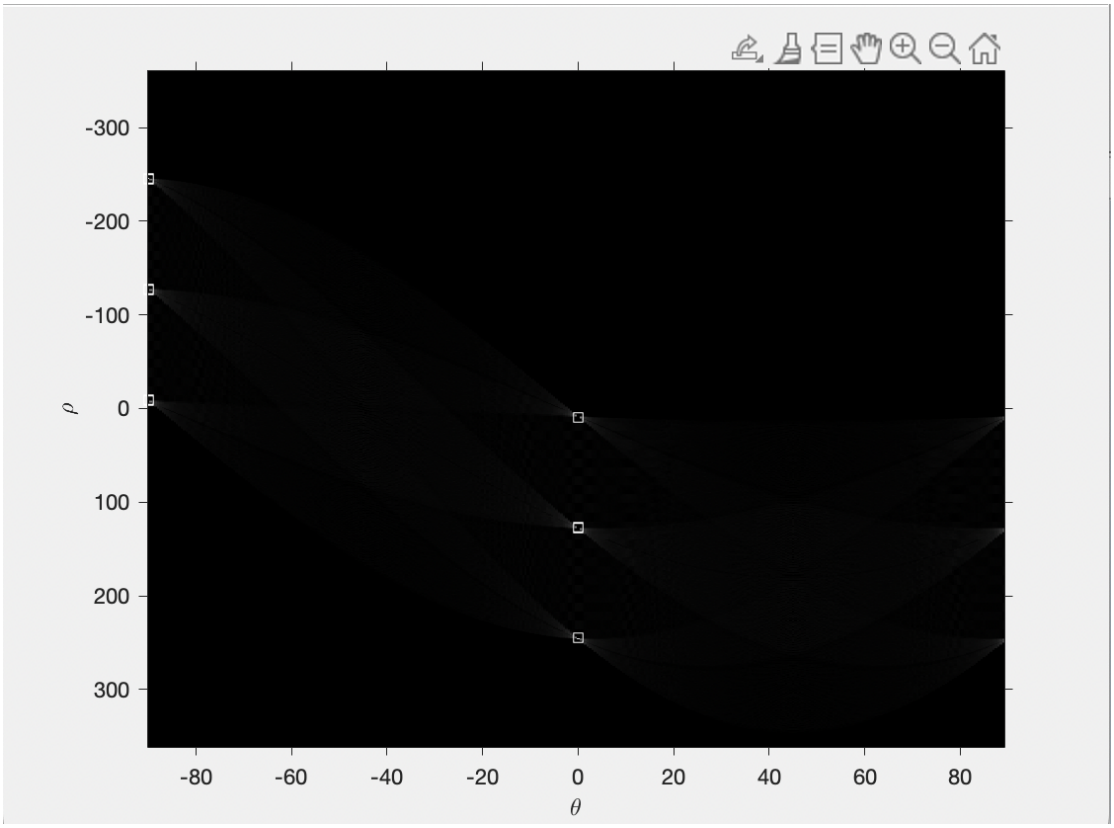
hough_peeks()

x | y | p

10x2 double

	1	2
1	233	1
2	705	1
3	741	181
4	1213	181
5	467	1
6	469	1
7	977	181
8	979	181
9	231	1
10	707	1

the position of the top10 max value



bs2-2-b-1

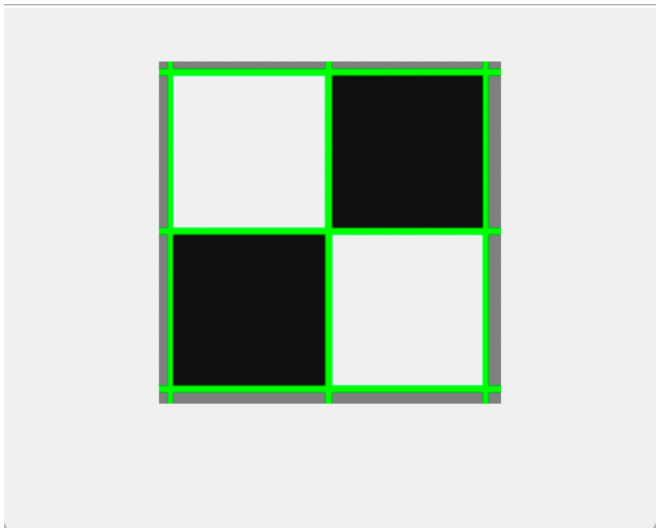
From the above data matrix, I found that some points are two close so we cannot distinguish them by human eyes.

c.

+8 | hough_lines_acc.m | hough_lines_draw.m | ps2_4_a.m | ps2_4

```
function hough_lines_draw(img,peaks,rho,theta)
figure, imshow(img), hold on
i = 1;
[X, Y] = size(peaks);
while 1
    %read the top X peeks value
    xr = peaks(i,1);
    xt = peaks(i,2);
    %find the corresponding rho and theta values
    r = rho(1,xr);
    t = theta(1,xt);
    %special condition when theta equals 0
    if t == 0
        y = [-1000,1000]
        x = [r,r]
    %common conditions
    else
        x = [-1000,1000]
        y = (r/sind(t))-x.*cosd(t)./sind(t);
    end
    plot(x,y,'LineWidth',2,'Color','green')
    i = i + 1;
    if i > X
        break
    end
end
end
```

hough_lines_draw()



ps2-2-c-1

d.

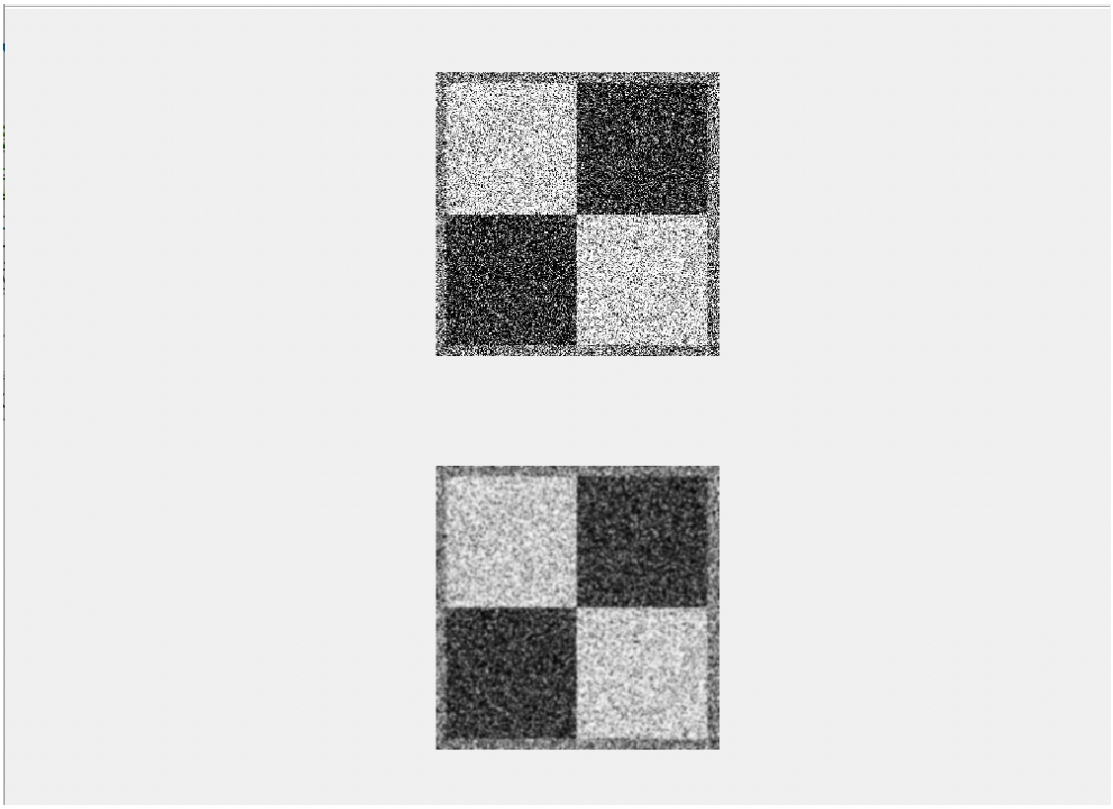
First of all, a two-dimensional array of accumulators is obtained through hough_lines_acc (), and the number of points through which the straight line passes can be obtained through the two-dimensional array, and then the n straight lines with the most passing points are obtained by hough_peeks (), so the corresponding theta and rho of the straight line can be obtained, and then the relation of the point (x) on the straight line can be obtained by theta and rho, thus the corresponding straight line can be drawn.

The accumulator bin size is default value 1, and the threshold value depends on the value of top n peeks value, and for this graph is about 213. And for finding peeks, I used the function

max() to find the max value in matrix H and get the position by for loops, then I set this max value points to 0 so that I can find the second max value positions, continue this operation until the number of max points is equals to the required number.

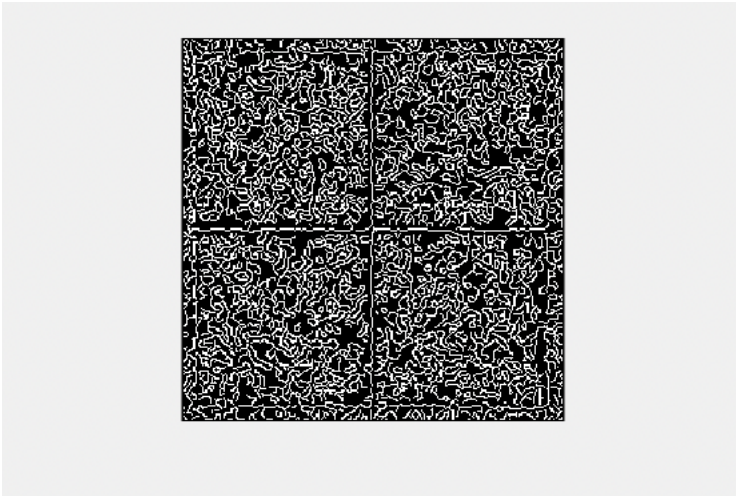
3.

σ=1



ps2-3-a-1.png

b.

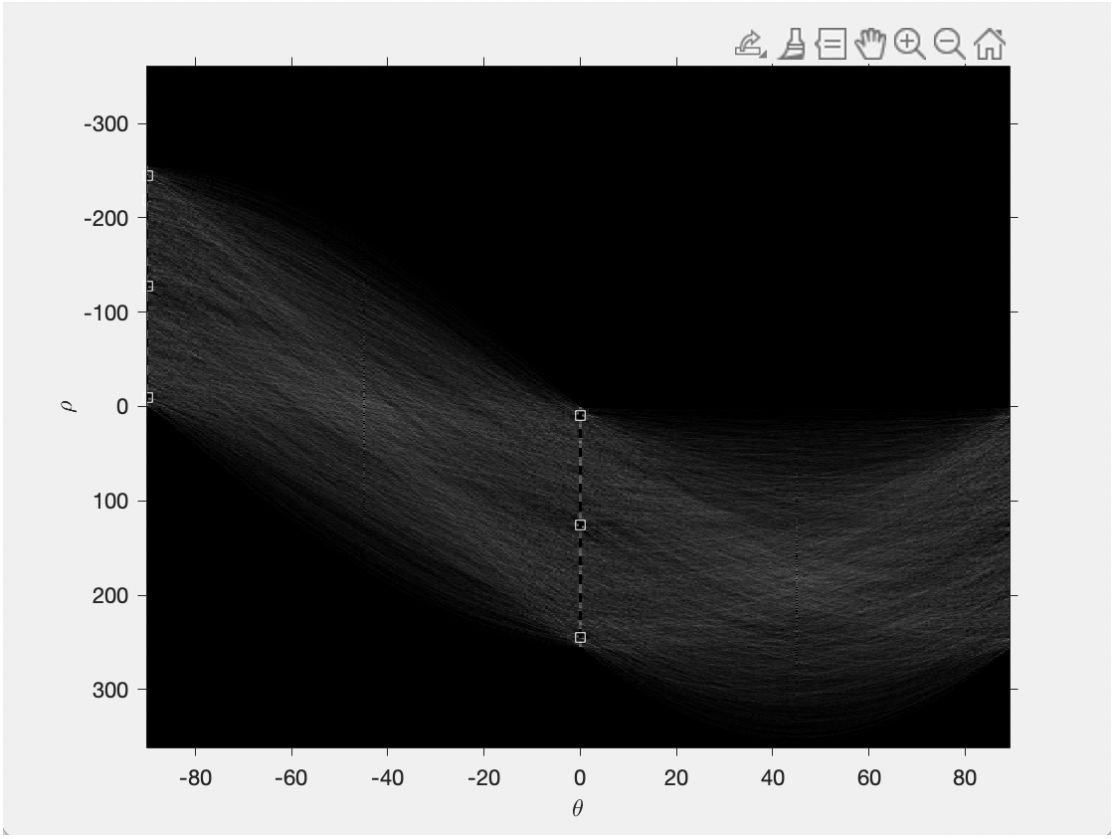


ps2-3-b-1

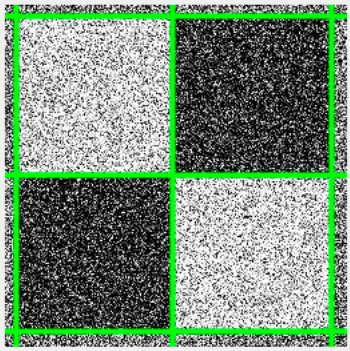
σ=2

ps2-3-b-2

c.



ps2-3-c-1



ps2-3-c-2

First, we need to smooth the image with noise as well as we can, so that when I edge the image, it will not be affected by too much noise. So that we can get a better edge image. And I found that with the increase of the σ, the edge output looks better. Besides, we need to set a good value of threshold to prevent extra lines of noise.

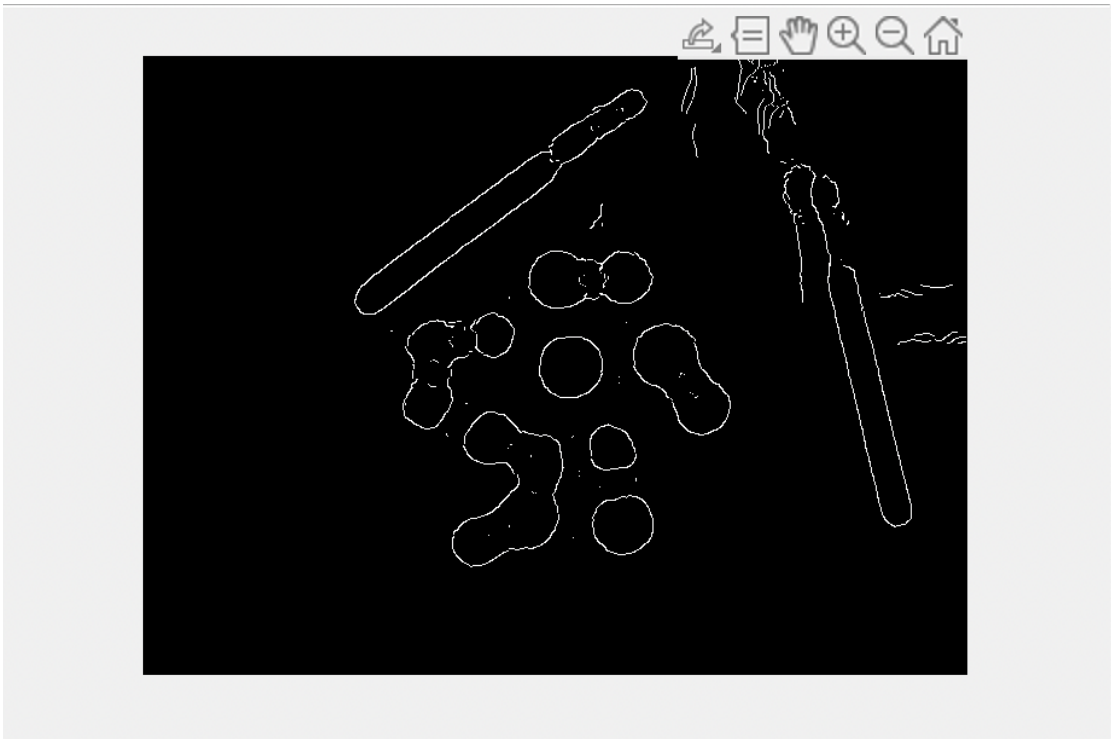
4.

a.

σ=10

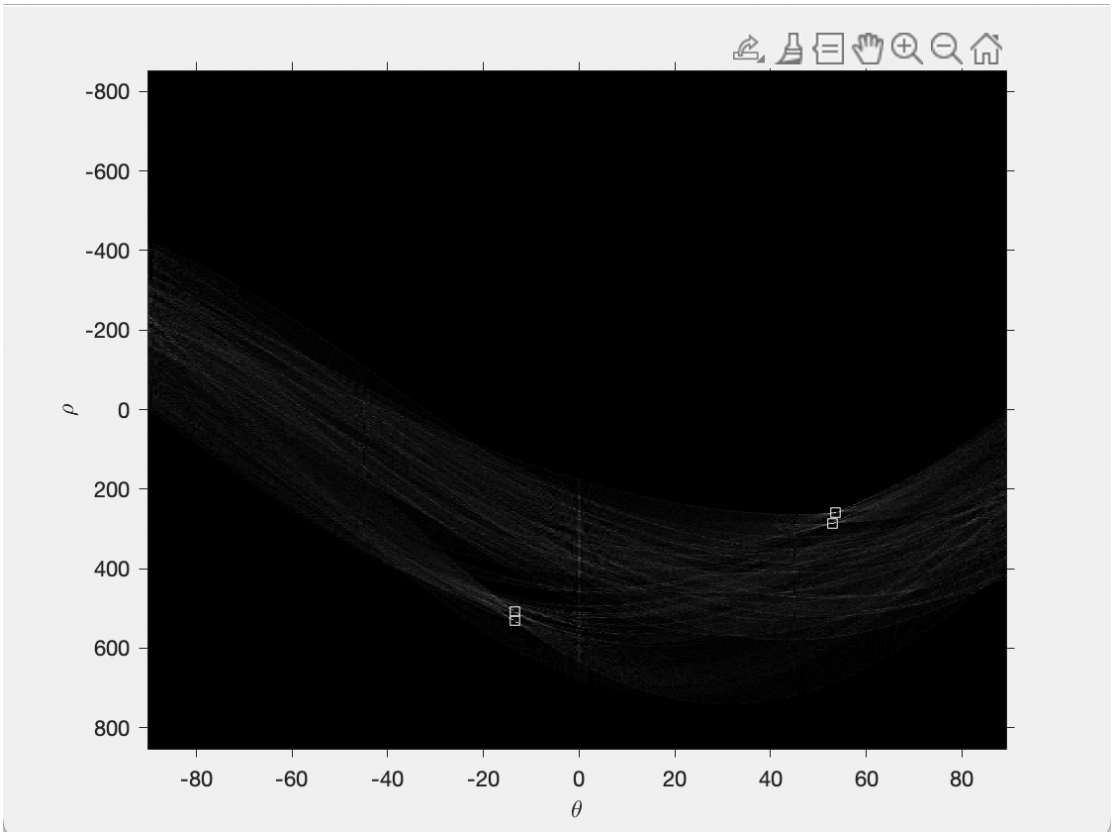
ps2-4-a-1

b.

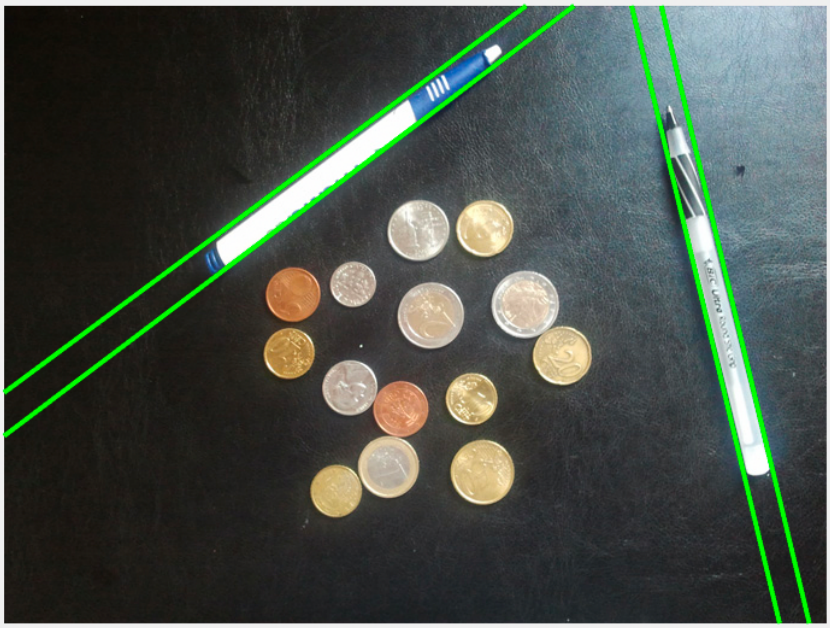


ps2-4-b-1

c.



ps2-4-c-1



ps2-4-c-2

First, when I set σ=2 and I found that the edge image of it is a lit bit mass, because of the gradient of the brightness, so to reduce the effect of the light, I increased σ up to 10 so that the edge image looks much clearer so that computer can better recognize the line. Then I set the peak number to 10 and I found that only 4 points of the Hough accumulator array image are highlighted and only 4 lines are be recognized, I think it because some lines are too close so that we cannot be distinguished by human eyes.

5.

a.

+8 | hough_lines_draw.m | ps2_4_a.m | ps2_4_c.m | hough_circles_

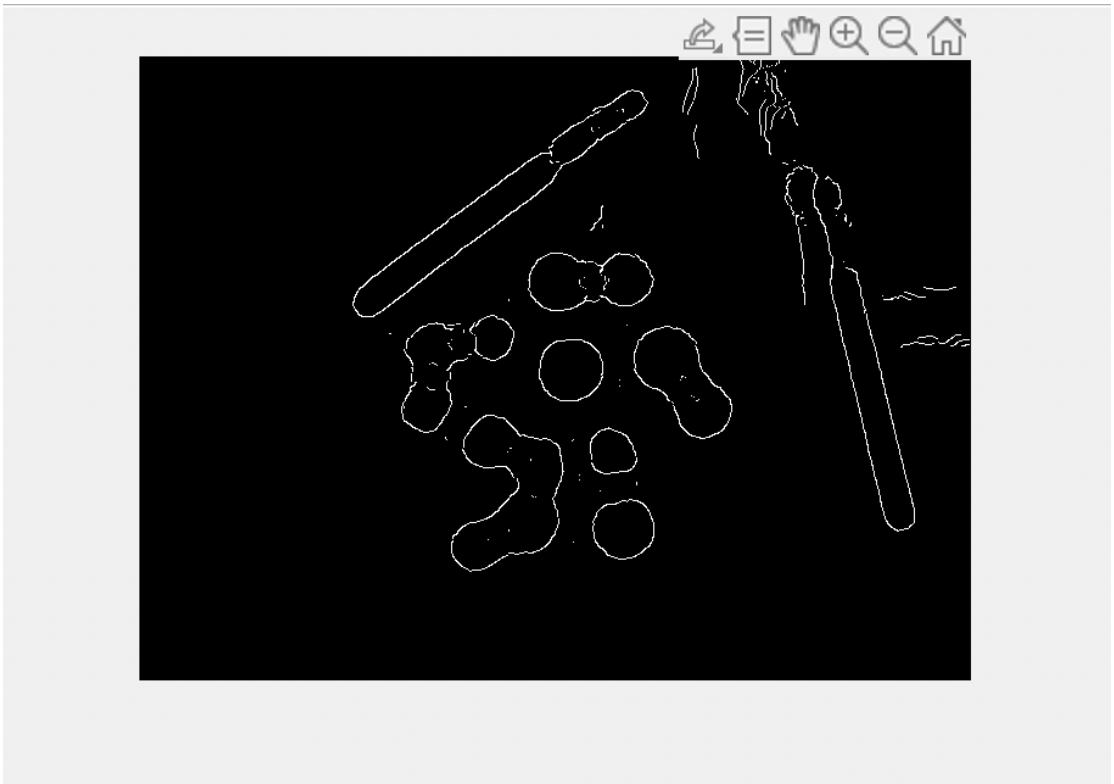
```
function [H] = hough_circles_acc(BW,r)
H = zeros(size(BW));
[x y] = find(BW); %find all pixels which val = 1
%to get the value of H matrix
for i = 1:length(y)
    low = y(i)-r;
    high = y(i)+r;
    if (low < 1)
        low = 1;
    end
    if (high > size(BW,2))
        high = size(BW,2);
    end
    for y0 = low:high
        x1 = round(x(i)-sqrt(r^2-(y(i)-y0)^2));
        x2 = round(x(i)+sqrt(r^2-(y(i)-y0)^2));
        %if the circles dont beyond the boundry of image, acc+1
        if x1 < size(BW,1) & x1 > 1
            H(x1,y0) = H(x1,y0) + 1;
        end
        if x2 < size(BW,1) & x2 > 1
            H(x2,y0) = H(x2,y0) + 1;
        end
    end
end
end

```

hough_circles_acc()

We can also use the same hough_peaks() function to get the max points, and these points can be considered as the centers of circles.

σ=10

ps2-5-a-1

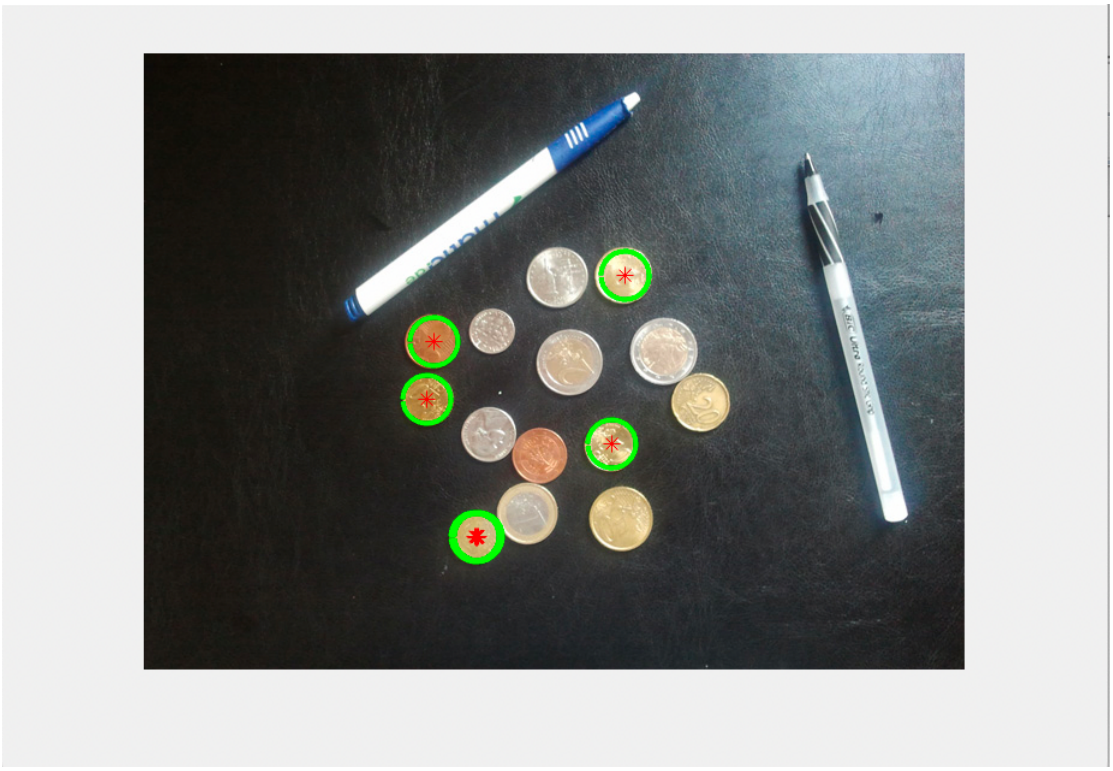


ps2-5-a-2

10x2 double

	1	2
1	402	277
2	403	277
3	402	278
4	402	279
5	325	390
6	288	236
7	402	276
8	403	278
9	240	242
10	185	401

10 centers of the circles



ps2-5-a-3

b.

14x2 double

	1	2
1	287	236
2	243	243
3	402	278
4	233	288
5	317	289
6	257	355
7	325	389
8	388	398
9	184	400
10	291	464
11	185	344
12	115	554
13	249	433
14	381	317

14x1 double

	1
1	21
2	23
3	21
4	20
5	22
6	25
7	20
8	24
9	22
10	23
11	25
12	22
13	26
14	27

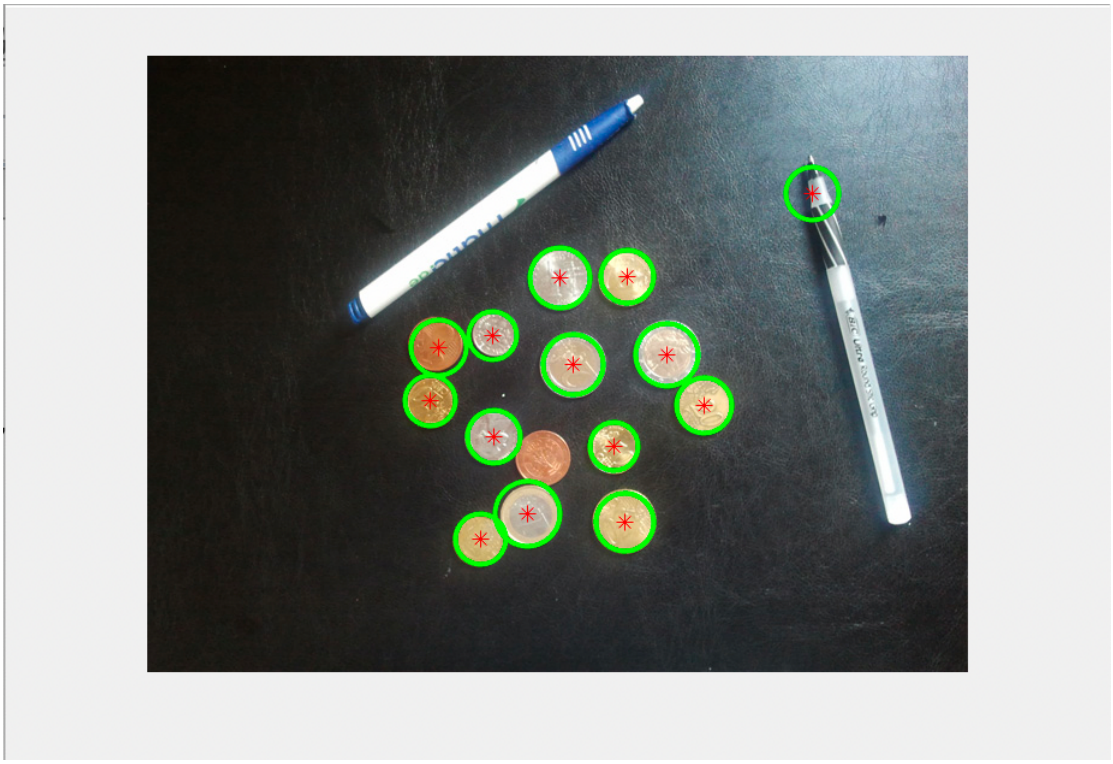
centers and radii

+10 ps2_4_c.m | hough_circles_acc.m | ps2_5_a.m | ps2_5_b.m

```
function [centers,radii] = find_circles(img_edges,r_min,r_max)
step_angle = 0.1
p = 0.4;
circle=[];
centers=[];
radii=[];
[m,n] = size(img_edges);
rs = r_max-r_min+1;
ra = round(2*pi/step_angle);
hough_space = zeros(m,n,rs);
[rows,cols] = find(img_edges);%find the rows and cols of val = 1;
count = size(rows);
%Correspond the image space(x,y) to the parameter space (a,b,r)
%because the value of r is not fixed.
for i=1:count
    for r=1:rs
        for k=1:ra
            a = round(rows(i)-(r_min+(r-1))*cos(k*step_angle));
            b = round(cols(i)-(r_min+(r-1))*sin(k*step_angle));
            if(a > 1 & a < m & b > 1 & b < n)
                hough_space(a,b,r) = hough_space(a,b,r)+1;
            end
        end
    end
end
% Search for points that exceed the threshold.
max_para = max(max(max(hough_space)));
% In a matrix, find the position where it is greater than the threshold
sat = find(hough_space>=max_para*p);
length = size(sat);
%get the paremeters of different circles which are satisfied
for k=1:length
    p3 = floor(sat(k)/(m*n))+1;
    p2 = floor((sat(k)-(p3-1)*(m*n))/m)+1;
    p1 = sat(k)-(p3-1)*(m*n)-(p2-1)*m;
    circle = [circle;p1,p2,p3];
end
```

```
%The points concentrated at the center of each circle are averaged to get
%the exact center and radius for each circle.
while size(circle,1) >= 1
    num = 1;
    newcircle = [];
    temp1 = circle(1,1);
    temp2 = circle(1,2);
    temp3 = circle(1,3);
    c1 = temp1;
    c2 = temp2;
    c3 = temp3;
    temp3 = r_min+temp3-1;
    if size(circle,1) > 1
        for k = 2:size(circle,1)
          if (circle(k,1)-temp1)^2+(circle(k,2)-temp2)^2 > temp3^2
             %Save the center and radius position of the remaining circle
             newcircle = [newcircle;circle(k,1),circle(k,2),circle(k,3)];
          else
            c1 = c1+circle(k,1);
            c2 = c2+circle(k,2);
            c3 = c3+circle(k,3);
            num = num+1;
          end
        end
    end
    c1 = round(c1/num);
    c2 = round(c2/num);
    c3 = round(c3/num);
    c3 = r_min+c3-1;
    centers = [centers;c1,c2];
    radii = [radii;c3];
    circle = newcircle;
end
end
```

find_circles()

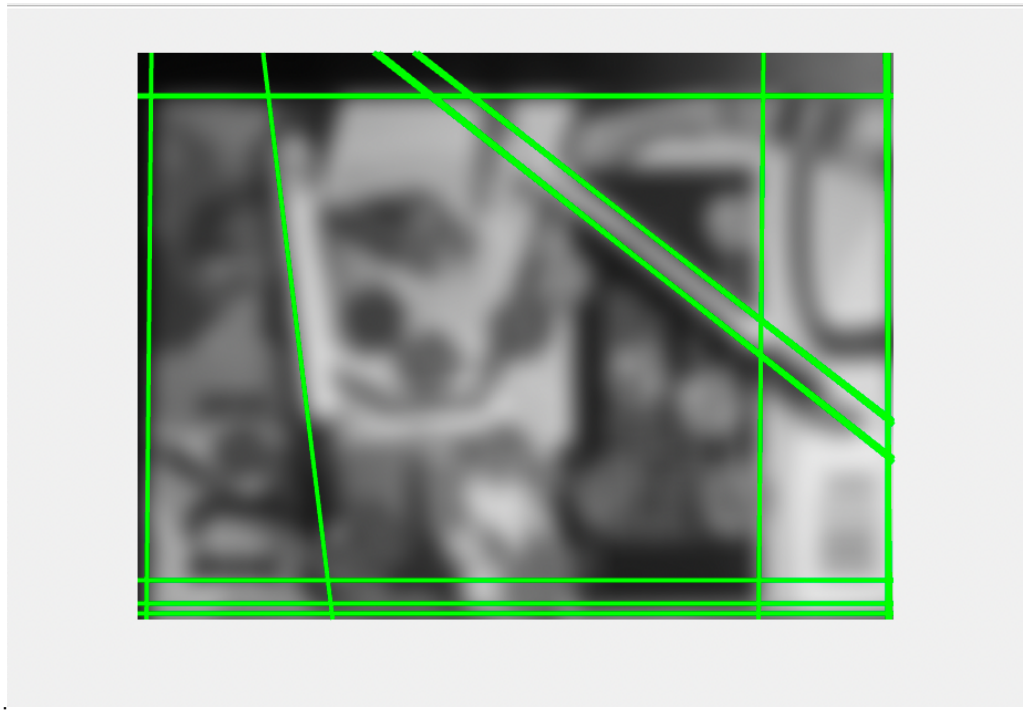


ps2-5-b-1

Because the radius of the circle we need to find is in a certain range, so I convert a two-dimensional array to a three-digit array and then use the find function to find the point that exceeds the threshold. Then convert 3D to 2D array, where the first two columns are the position of the center of the circle, and the third column is the radius. Since there is more than one point near the center of the circle, the effect of drawing it directly will be very poor, so I averaged the points near the center of the circle to get an accurate position of the center and radius. Therefore, I can more accurately detect the circle in the picture that meets the requirements.

6.

a.

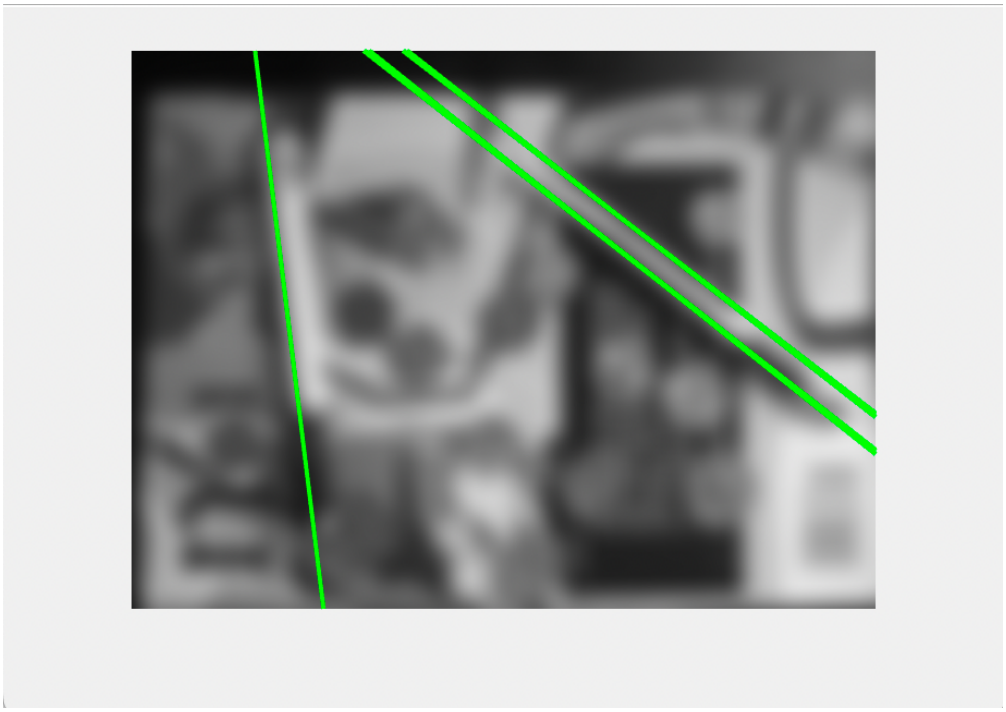


ps2-6-a-1

b.

Because for the edge image, the computer can not recognize which line is the boundary of pens or boundaries of others. Computers can only detect the lines, but only humans can distinguish different lines by looking at the original images.

c.

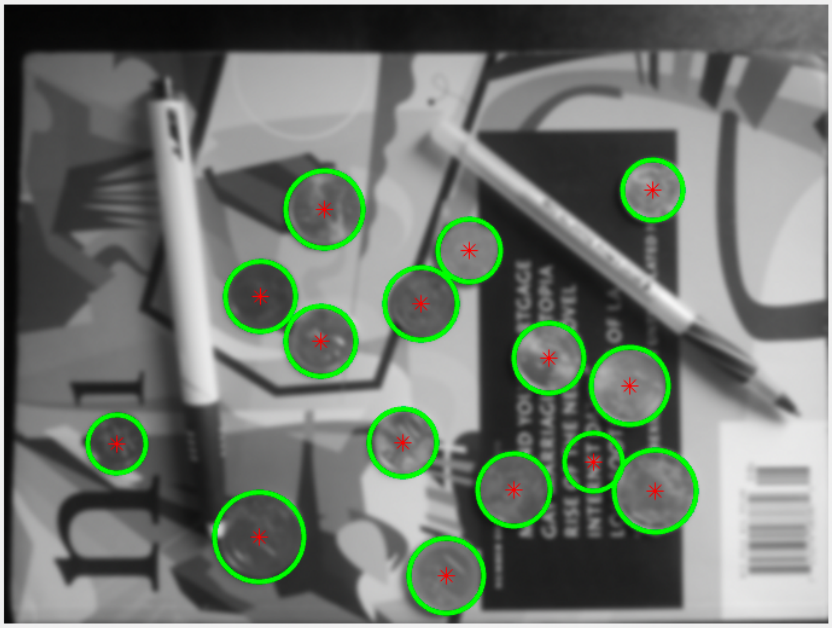


ps2-6-c-1

I delete the line which is almost horizontal or vertical because I found that the theta of the lines of pens is different from other lines.

7.

a.



ps2-7-a-1

b.

Yes, there is some false alarm that exists. I think it is because the value of the threshold is hard to set.

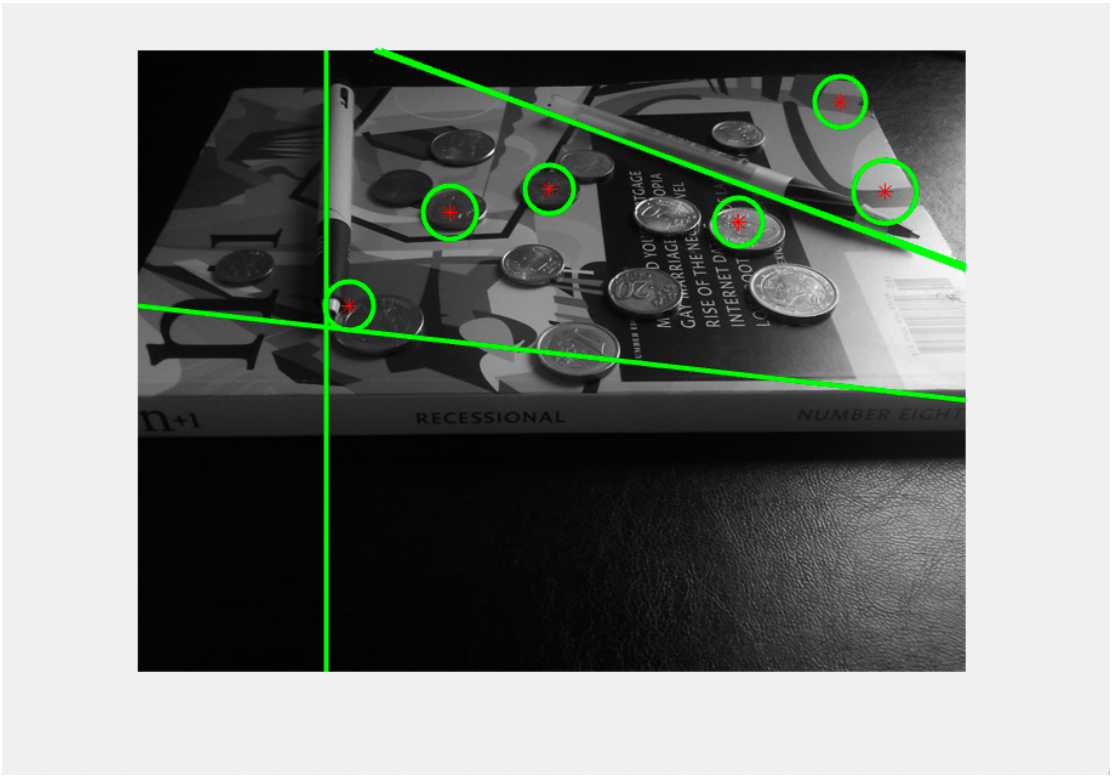
For code, in the find_circles.m lin4, I set the value of threshold equal to 7 and I found that when at this value, the false is the least.

```
% Search for points that exceed the threshold.
max_para = max(max(max(hough_space)));
% In a matrix, find the position where it is greater than the threshold
sat = find(hough_space>=max_para*p);
length = size(sat);
```

And we can see the code here, the threshold decides which points are satisfied with what I need. Even though, there still exists some mistakes because the satisfying points of the edge image may confuse the computer because the accumulator on these points may also be big enough.

8.

a.



ps2-8-a-1

b.

Because the coin changes from a circle to an ellipse under the change of angle of view.

I think there are two ways to solve this problem.

The first method is to change the shape of the picture and stretch the picture so that the view of the picture becomes the top view so that the ellipse can become a circle by stretching.

The second method is to use the function to identify the ellipse, through the method of identifying the ellipse to get the center of the ellipse, and then draw the outline of the ellipse.

The first method needs to know what the angle of the change of perspective is, which is not easy to implement, so I choose the second method.

c.

I try my best to solve this problem, but I don't know how to deal with an ellipse, which has two focus and two parameters of radius. It takes four parameters to decide on an ellipse. And different ellipse has different parameters. I haven't come up with a good solution for this question. I am sorry for that.