EE 417 Computer Vision

Berkant Deniz Aktaş

19515

Post-Laboratory Report

November 9, 2018

Sabancı University

Faculty of Engineering and Natural Sciences

Computer Science and Engineering

In this lab, we used built-in MATLAB functions in order to detect lines and circles of given images. Most of the functions taken from MATLAB's website e.g Hough Lines :

https://www.mathworks.com/help/images/ref/houghlines.html

In this report, I will try explain the functions and try to change parameters on different images.

Line Detection

```
function img = lab4houghlines(img)
imgRGB = img;
[row, col, ch] = size(img);
if(ch==3)
    img = rgb2gray(img);
end
imgGrey = img;
img = edge(img, 'Canny');
imgEdges = img;
[H,T,R] = hough(img,'RhoResolution',0.5,'Theta',-90:0.5:89);
P  = houghpeaks(H,20);
subplot(2,2,1);
imshow(imgRGB);
title('checkers.png RGB');
subplot(2,2,2);
imshow(imgEdges);
title('checkers.png Canny');
subplot(2,2,3);
imshow(imadjust(rescale(H)),'XData',T,'YData',R,...
      'InitialMagnification','fit');
title('Hough transform of checkers.png');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
colormap(gca,gray);
subplot(2,2,4);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
title('Hough peaks of checkers.png');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
plot(T(P(:,2)),R(P(:,1)),'s','color','white');
lines = houghlines(img,T,R,P,'FillGap',10,'MinLength',40);
figure, imshow(imgGrey), hold on
max_len = 0;
min_len = 100;
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
   % Plot beginnings and ends of lines
   plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
   % Determine the endpoints of the longest line segment
   len = norm(lines(k).point1 - lines(k).point2);
   if ( len > max_len)
      max_len = len;
      xy_long = xy;
   end
   if(len < min_len)
       min_len = len;
      xy_short = xy;
   end
end
 plot(xy_long(:,1),xy_long(:,2),'LineWidth',5,'Color','cyan');
 plot(xy_short(:,1),xy_short(:,2),'LineWidth',5,'Color','red');
end
```

PS: After getting peaks, sometimes MATLAB detects lines of Hough Peaks because of computation power. I used pause(0.01) before drawing lines.

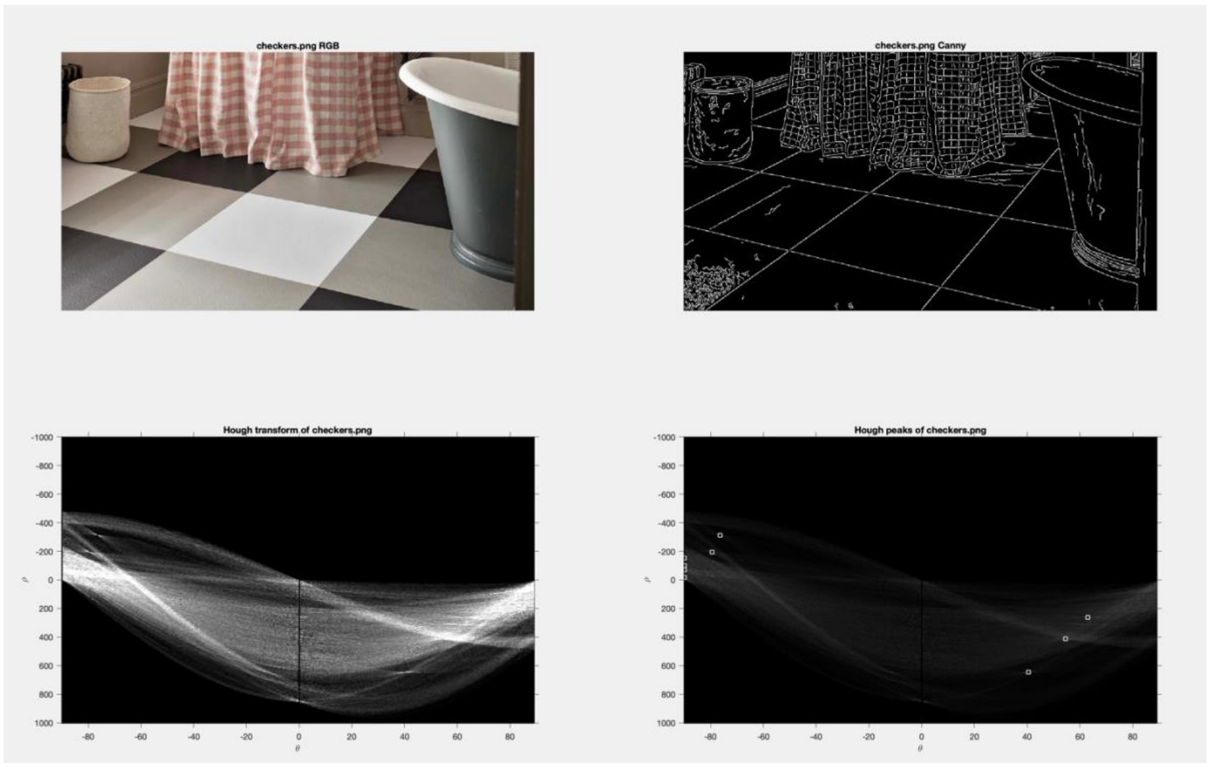

Figure 1. checkers.png edges and hough transform

First of all, I used Canny filter for detecting the edges of image. Canny image turned checkers.png to Black and White image so I was able to use hough transformation function. After implementing [H,T,R] = hough(img,'RhoResolution',0.5,'Theta',-90:0.5:89); function, MATLAB plotted the graph of hough transformation. According to MATLAB theta and rho stands for distance from coordinate origin and line rotation angles. There are also more parameters but they will be discussed in the following sections. After preprocessing image and gathering the graphs, houghlines successfully draw the lines.

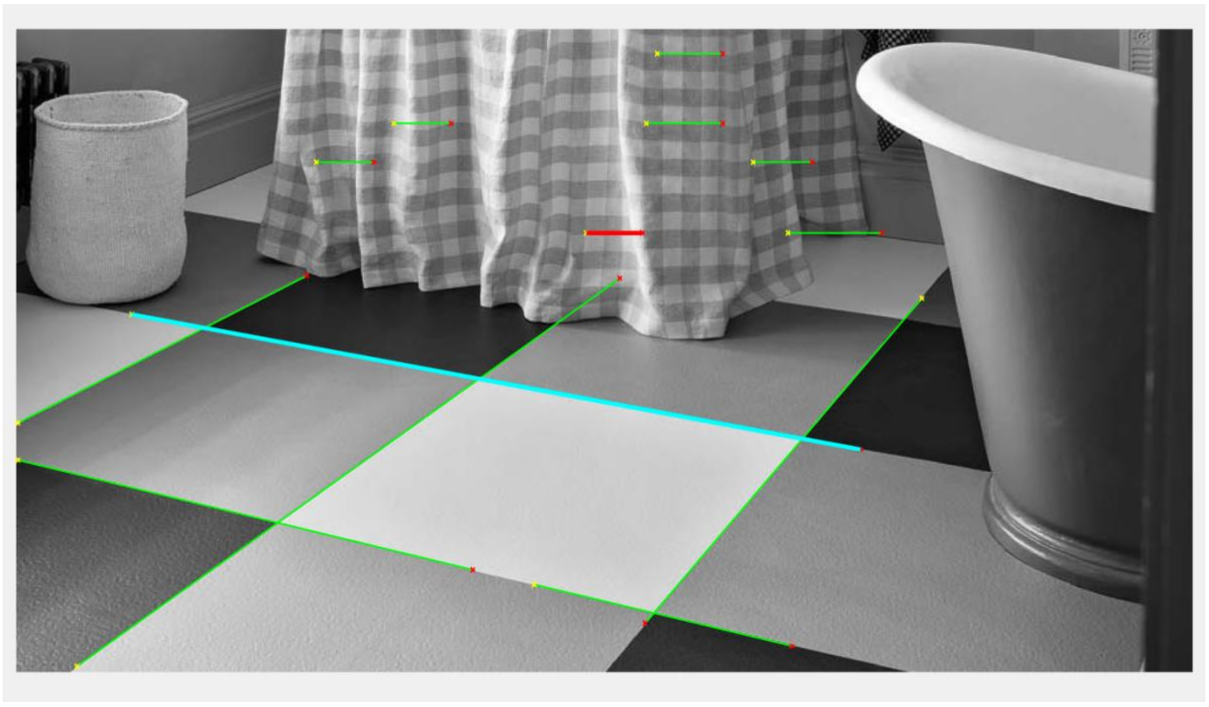

Figure 2. lines detected in checkers.png

Houghlines built-in function detected the lines and marked the longest line. According to lab instructions, we were also supposed to detect the shortest line. I used a similar trick to detect the shortest line and marked it with red color as its shown in the Figure 2.

```
if(len < min_len)
       min_len = len;
      xy_short = xy;
   end
```

With the similar reasoning with finding the longest line, I was able to find the shortest one.

Circle Detection

In this part, we were supposed to detect circles using built-in MATLAB functions.

İmfindcircles() function helped us to detect these circles. It has Object Polarity and

Sensitivity parameters but again they will be discussed in following parts.

```
function [] = lab4houghcircles()
img = imread('circlesBrightDark.png');
[row, col, ch] = size(img);
if(ch==3)
    img = rgb2gray(img);
end
Rmin = 20;
Rmax = 40;
figure;
subplot(1,3,1);
imshow(img);
title('Detected Circles using Hough Transform 20<= R <= 40');
[centersBright, radiiBright] = imfindcircles(img,[Rmin
Rmax],'ObjectPolarity','bright');
[centersDark, radiiDark] = imfindcircles(img,[Rmin
Rmax],'ObjectPolarity','dark');
viscircles(centersBright, radiiBright,'Color','b');
viscircles(centersDark, radiiDark,'LineStyle','--');
subplot(1,3,2);
imshow(img);
title('Detected Black Circles using Hough Transform 20<= R <= 60,
sensitivty 0.7 ');
Rmin = 20;
Rmax = 60;
[centersDark, radiiDark] = imfindcircles(img,[Rmin
Rmax],'ObjectPolarity','dark', 'Sensitivity', 0.7);
viscircles(centersDark, radiiDark,'Color','g');
subplot(1,3,3);
imshow(img);
title('Detected White Circles using Hough Transform 20<= R <= 60 ');
[centersBright, radiiBright] = imfindcircles(img,[Rmin
Rmax],'ObjectPolarity','bright');
viscircles(centersBright, radiiBright,'Color','y');
end
```

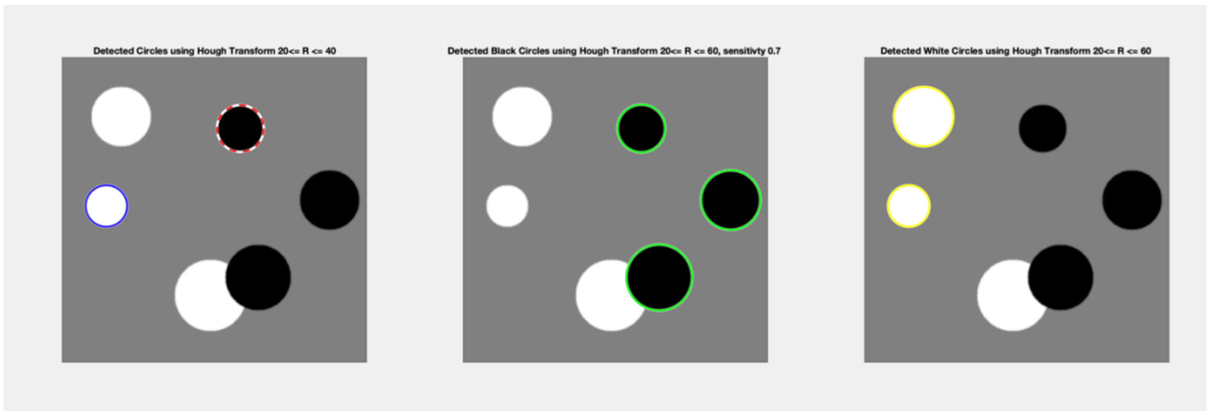

Figure 3. imfind circles with different parameters.

I tried to achieve the pictures in the lab as much as a I can. Figure 3. shows imfindcircles() on action with different parameters. First image detected both dark and bright circles which has radius betwen 20 and 40, second image detected dark circles which has radius between 20 and 60 with sensitivity set to 0.7 and third image detected bright circles which has radiues between 20 and 60.

Parameters, tic-toc and different images

I did not implement tic-toc command in the lab, but in the following sections I will try to use the command for comparing the execution times of function on different images.

Line detection parameters and discussion

There are different parameters for line detection but I will try to change ‘fill gap’ and ‘min lenght’ parameters and do not touch the parameters of Canny Edge Detector, Hough Transofrmation and Peak Threshold.

For the above code, Fill Gap is 10 and minLength is 40 and with sleep(1) elapsed time is 1.588160 seconds. I wanted to try an absurd parameter for fillGap. Since it is responsible for merging the lines setting it to a higher value will result us an interesting result.

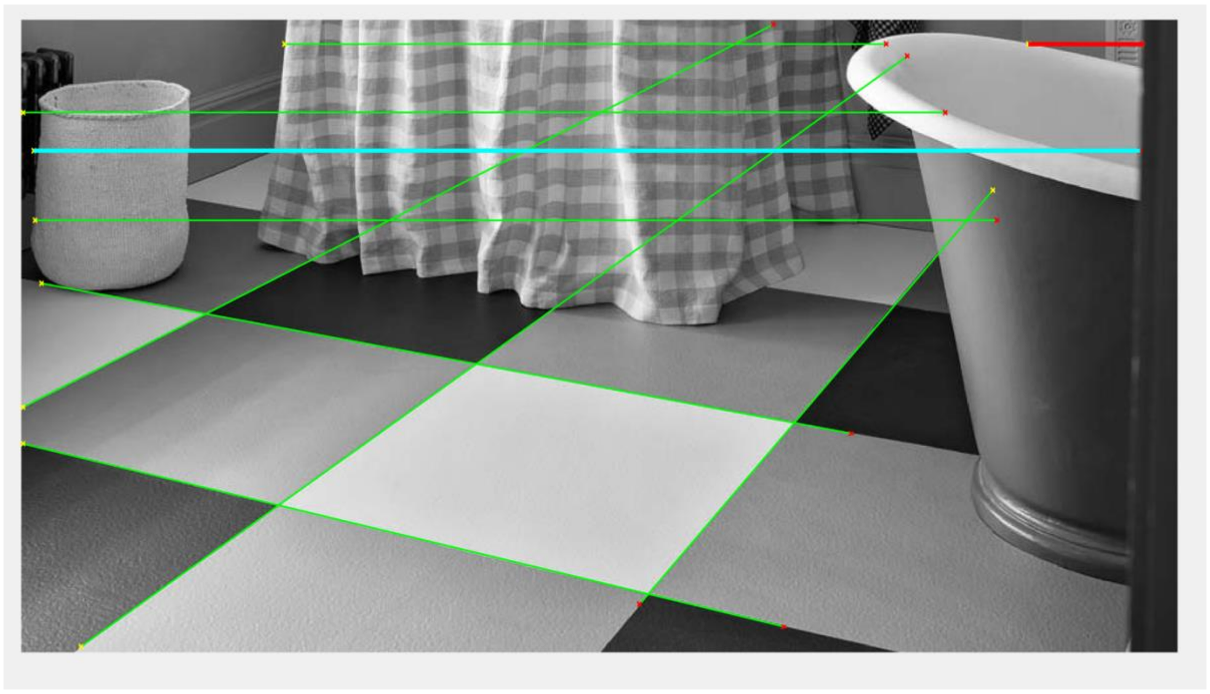

Figure 5. Fill Gap is 100

As its shown in the figure 5 we have some interesting lines detected. If we zoom the table cloth, we can see that function fills the gaps between small lines and extends the floor tiles and detect invisible lines which are under the table cloth. Now lets try to increase min length and set fill gap to 10.

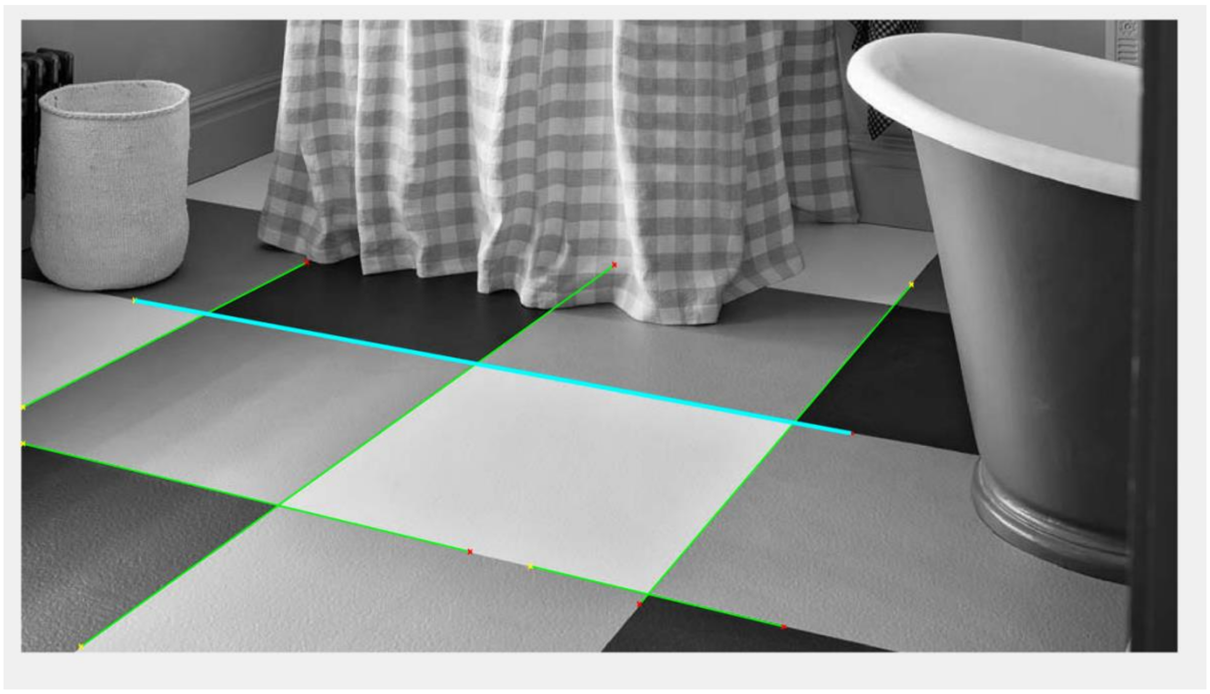

Figure 6. MinLength is 100

As it can be guessed setting a higher threshold for minlength resulted decrease of the lines detected. Also program failed to find shortest one. Now lets try another image with these parameters.

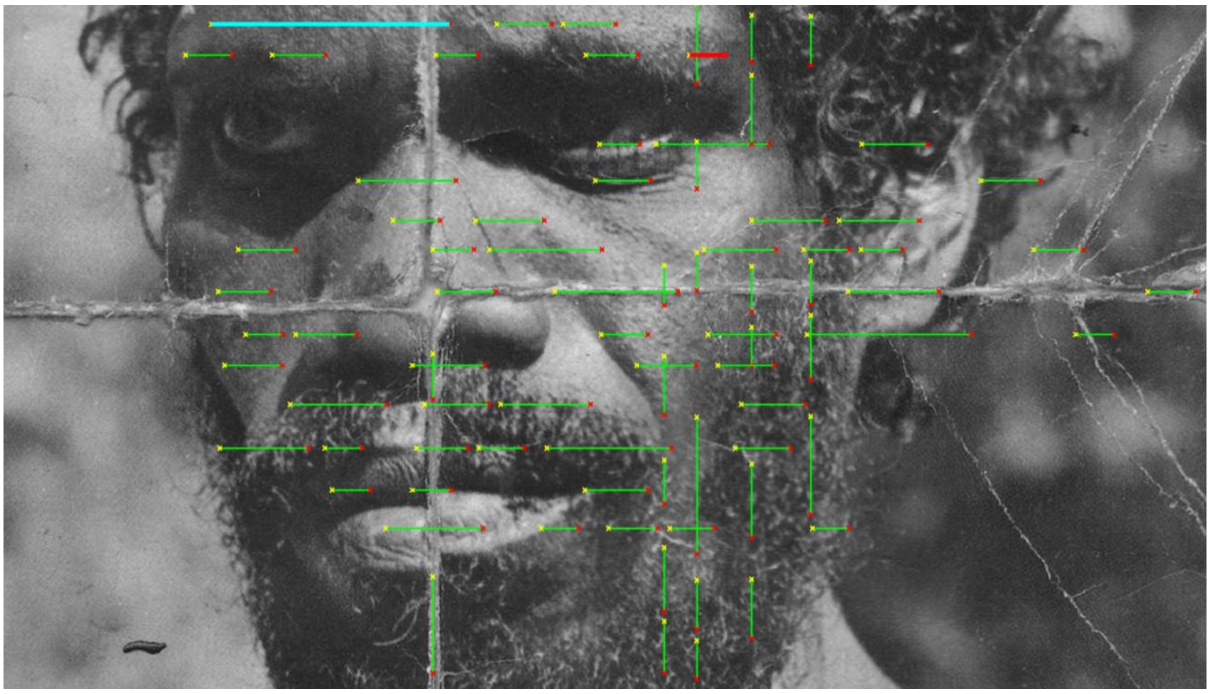

Figure 7. minLength 40 fillGap 10

It looks like it does not detecting lines properly. Lets try to change the parameters of Canny edge detector to see its effects on the image. I changed Canny thresh holds to 0.16 and 0.17 and set FillGap 80, MinLength 50 to see new results.

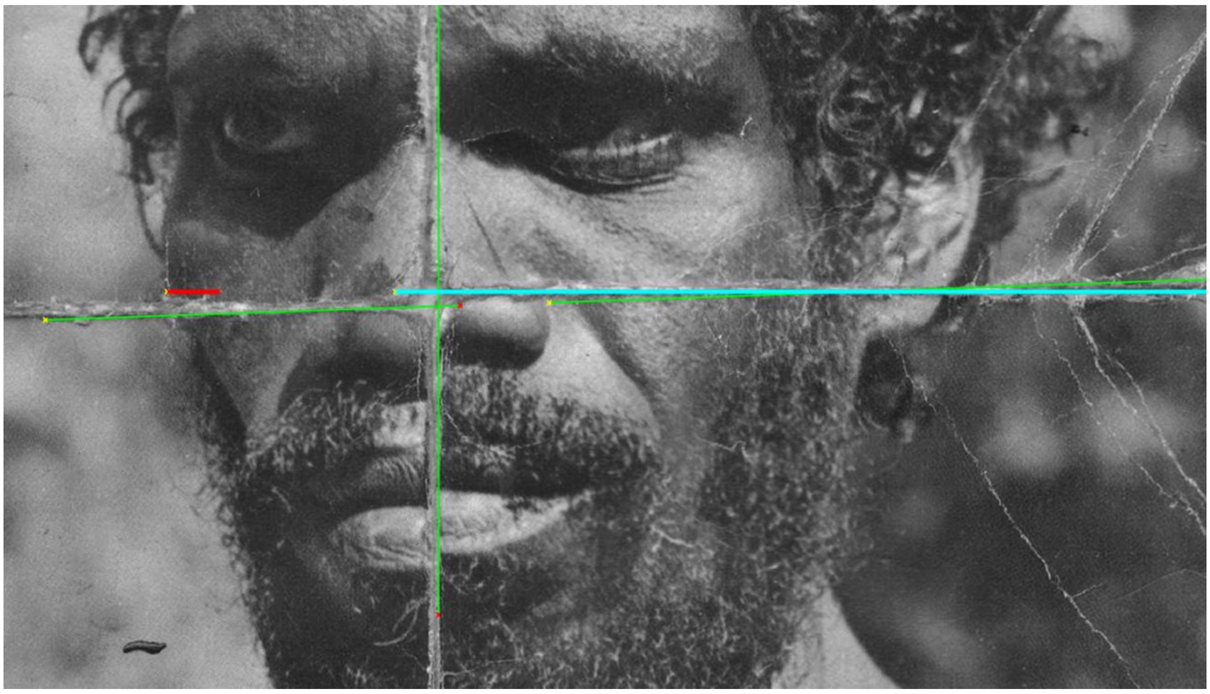

Figure 8. New Canny parameters, minLength 50 FillGap 80

As it can be guessed increasing fill gap and detecting better edges will resulted detecting the strong lines. As you can see the picture has folded 2 times. This folding created lines on the image. With new parameters line detector successfully detected folding lines. I realized that elapsed time is not changing that much when playing with paramaters with Sleep(1) the result is between 1.6 and 1.8 seconds for man.png. Conclusion is FillGap creates longer lines with merging tiny lines. Increasing it will cause less number of lines but they will be longer and merged if they are co linear. minLength is a threshold for line length. Increasing it will decrease the number of lines and setting a small minLength will cause lots of small lines detected.

Circle detection parameters and discussion

According to MATLAB’s function page, imfindcircles() has different kind of parameters but I will try to play with sensitivity, object polarity , radius ranges.

As its shown on the Figure3. changing radius parameter will result detecting circles in the range e.g first image only small circles detected because of threshold. Object polarity is for detecting ‘dark’ or ‘bright’ circles depending on background. Sensitivity is a parameter between [0,1] which denotes the threshold of hough transform. Increasing sensitivity will detect more circles but according to MATLAB’s page, it may result false detections. Lets try those on different images and test parameters.

I tried 1920x1080 images and setting minRadius = 10 and maxRadius = 1000. MATLAB gave me warning about radius parameters. But as one can guess increasing the gap between maxRadius and minRadius increases the elapse time.

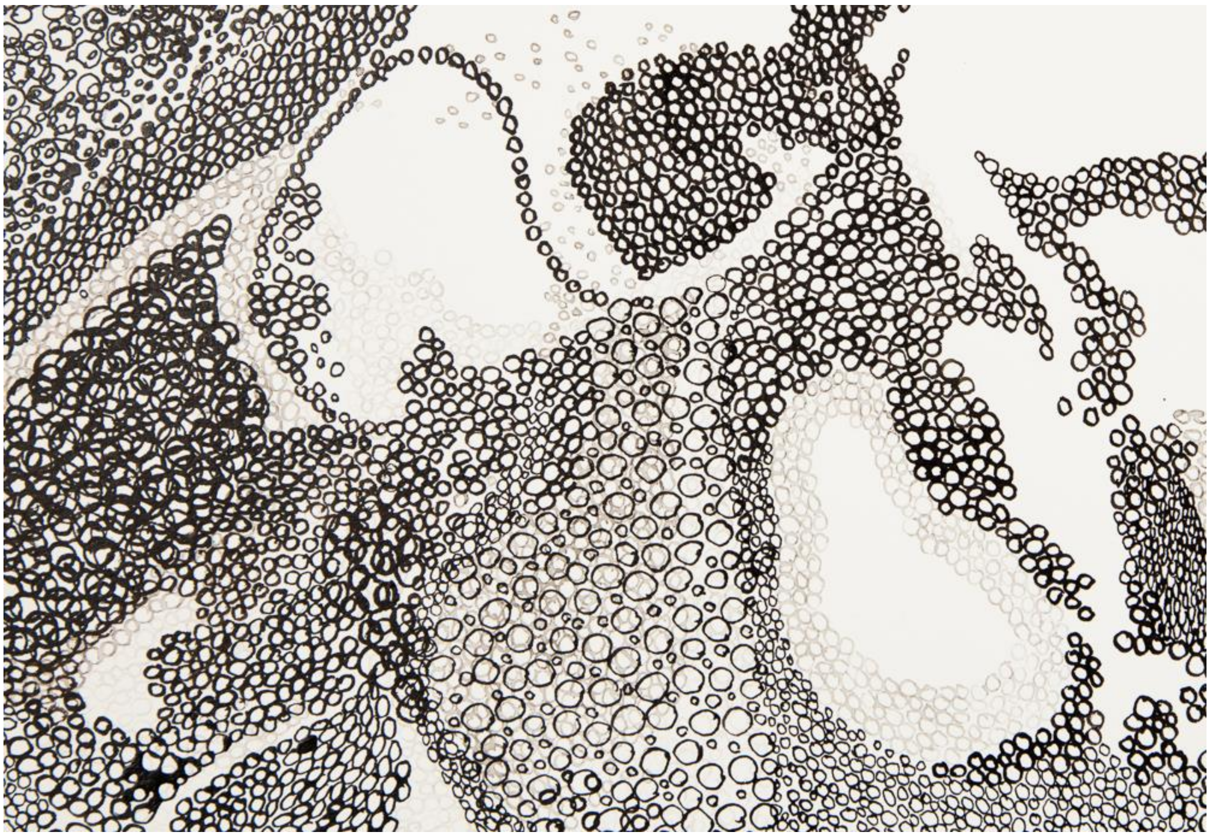

Figure 9. image for circle detection.

I will try to detect circles in this image with changing sensitivity and radius parameters.

Magenta circles stands for light and yellow circles stands for dark circles.

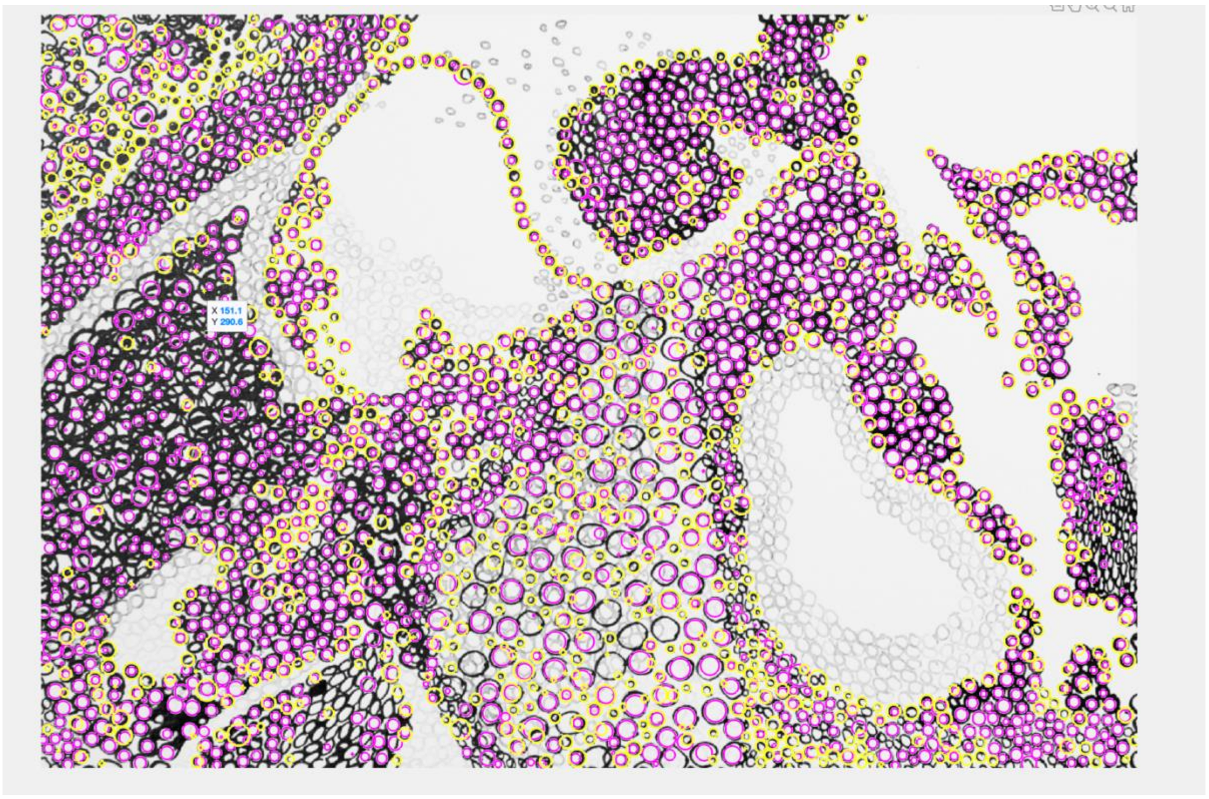

Figure 10. min radius 1 max radius 10

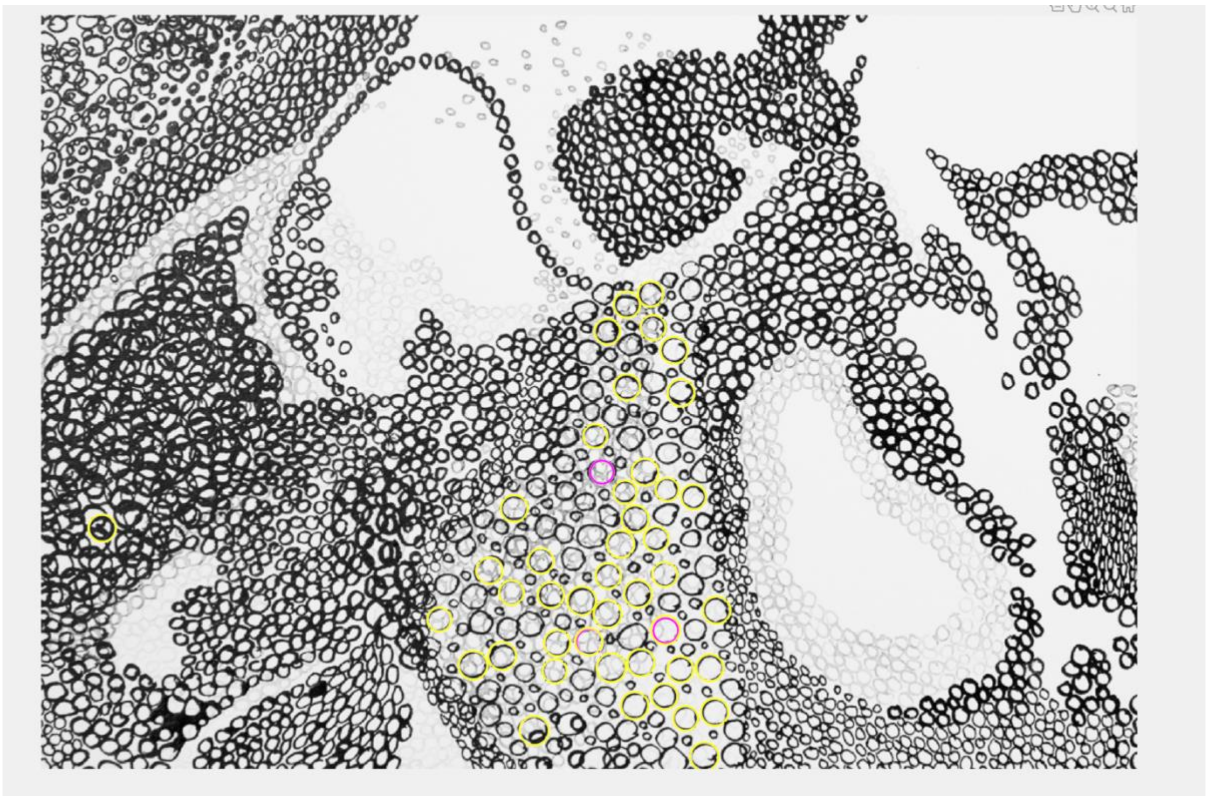

Figure 11. min radius 10 max radius 100

As it can be guessed increasing max radius and min radius grant us bigger circles detected.

Note that Figure 11. Has the circles which Figure 10 failed to detected and vice versa.

Not lets try to play with sensitivity.

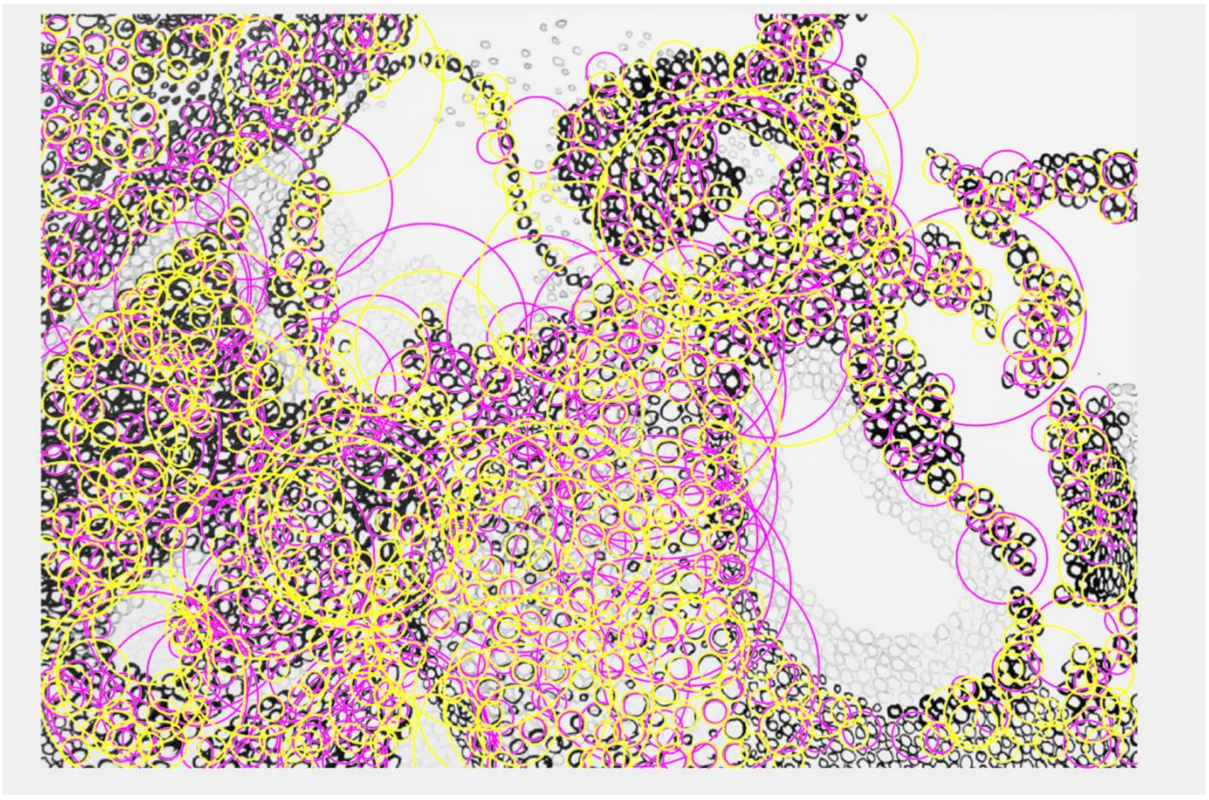

Figure 12. Sensitivity set to 0.95

As its stated in the MATLAB’s website, setting sensitivity high resulted false circles detected.

Now lets try to decrease it. Default value is 0.85

Setting it 0.65 detected no circles for this image.

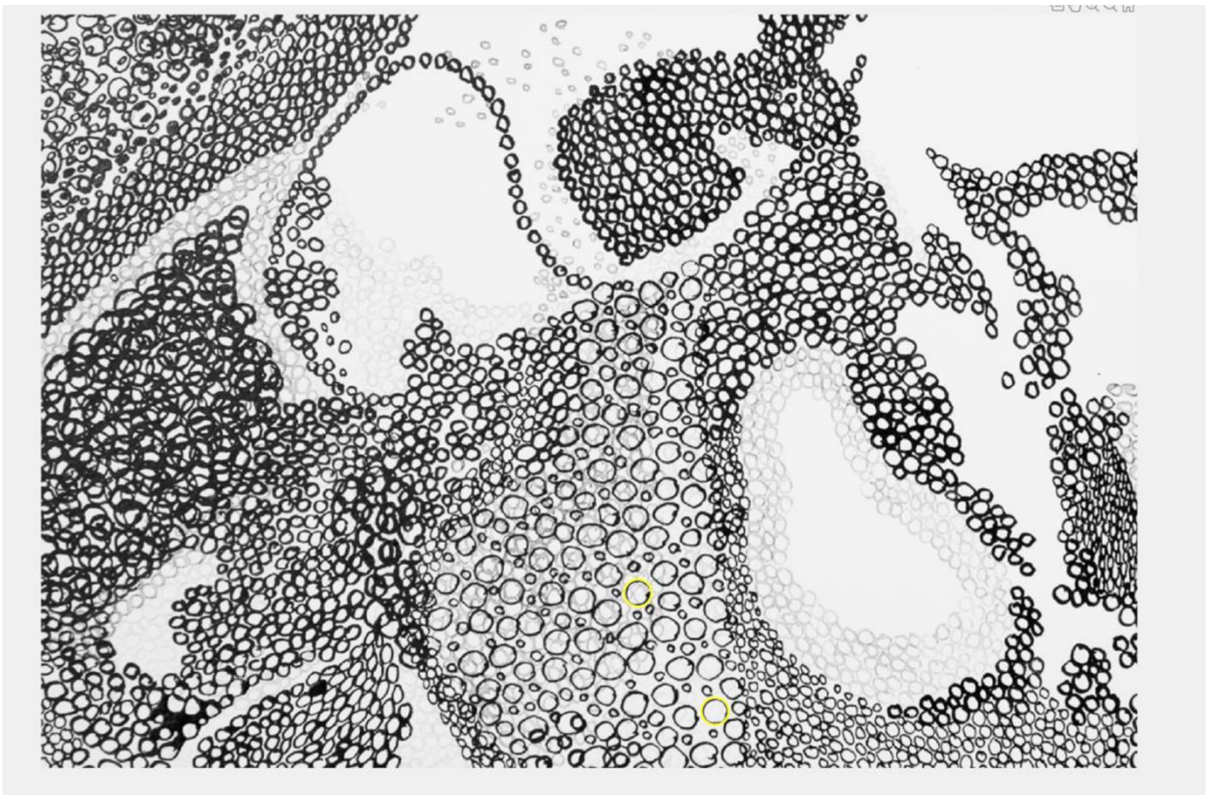

Figure 13. Sensitivity is 0.75

Increasing sensitivity from 0.65 to 0.75, we can see 2 circles detected which are better drawn circles than others. Now lets try to edge parameter of imfindcircle().

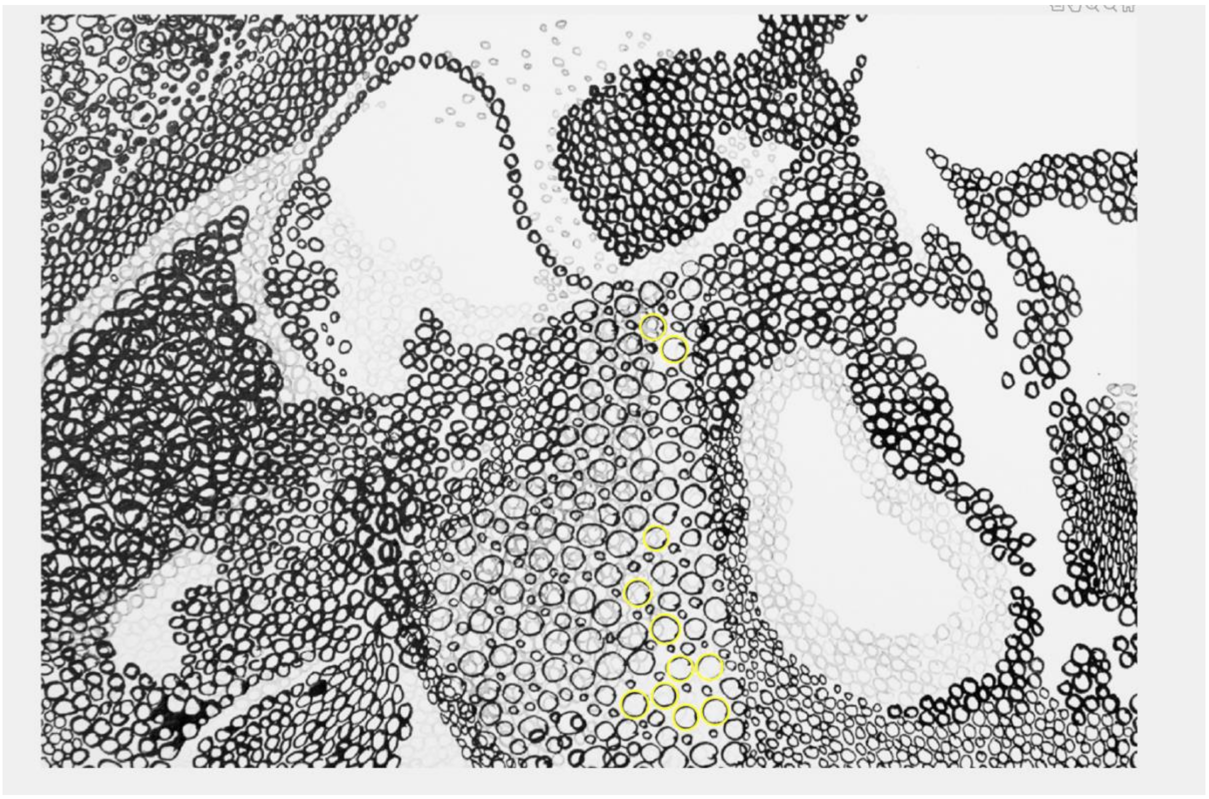

Figure 14. Sensitivity is 0.75 and Edge Threshold is 0.1

Setting edge threshold to 0.1 resulted detecting more weak edges which resulted detecting circles which has weak edges. When we set it to 0.7 no edges will be detected for Rmin = 10 and Rmax = 100. In Figure 10 we saw lots of circles for Rmin = 1 and Rmax = 10. Not lets to try to set an high edge threshold for it to see results.

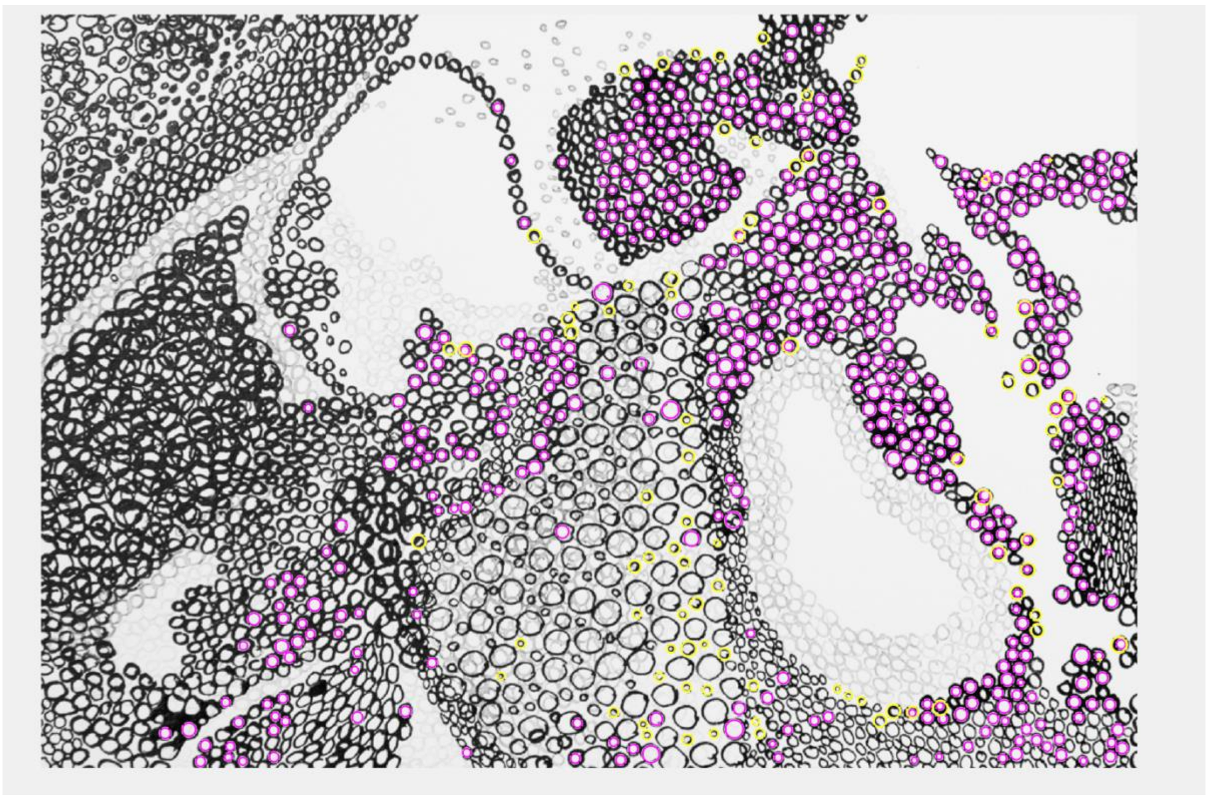

Figure 15. Edge Threshold 0.7 of Figure 10

Comparing Figure 15. and Figure 10, we can see decrease of the number of circles detected. Figure 15 has the circles which has strong edges.

Comparing the elapsed times we can see that, higher radius interval and bigger radius causes increase in elapsed time. Also decreasing threshold will increase elapsed time because it needs to traverse more edges to find circles.

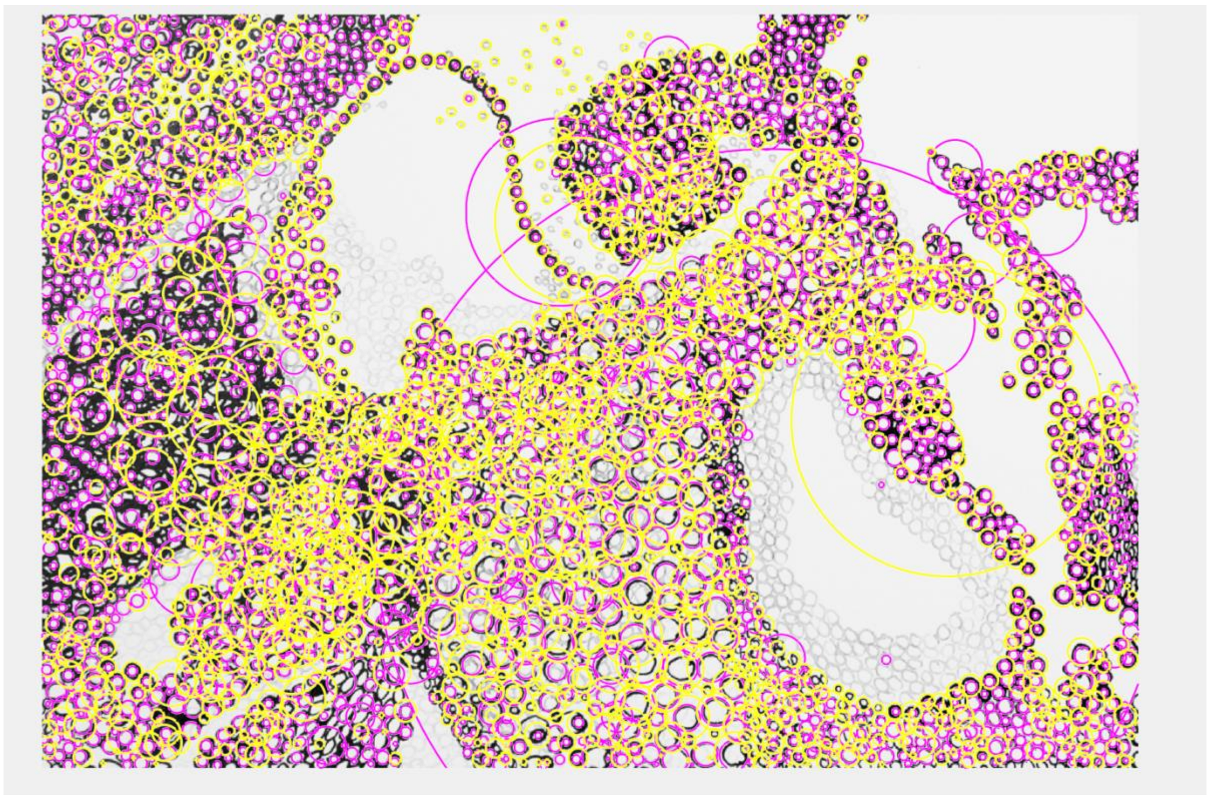

Figure 16. Tried crazy parameters

Figure 16 shows minumum radius = 1, maximum radius = 1000 and threshold set to 0.1. It took more than one minute to detect these circles