EE417 - Computer Vision – Post-Lab Report #4
Hakan Buğra Erentuğ – 23637
31 / 11 / 19 – Mehmet Keskinöz

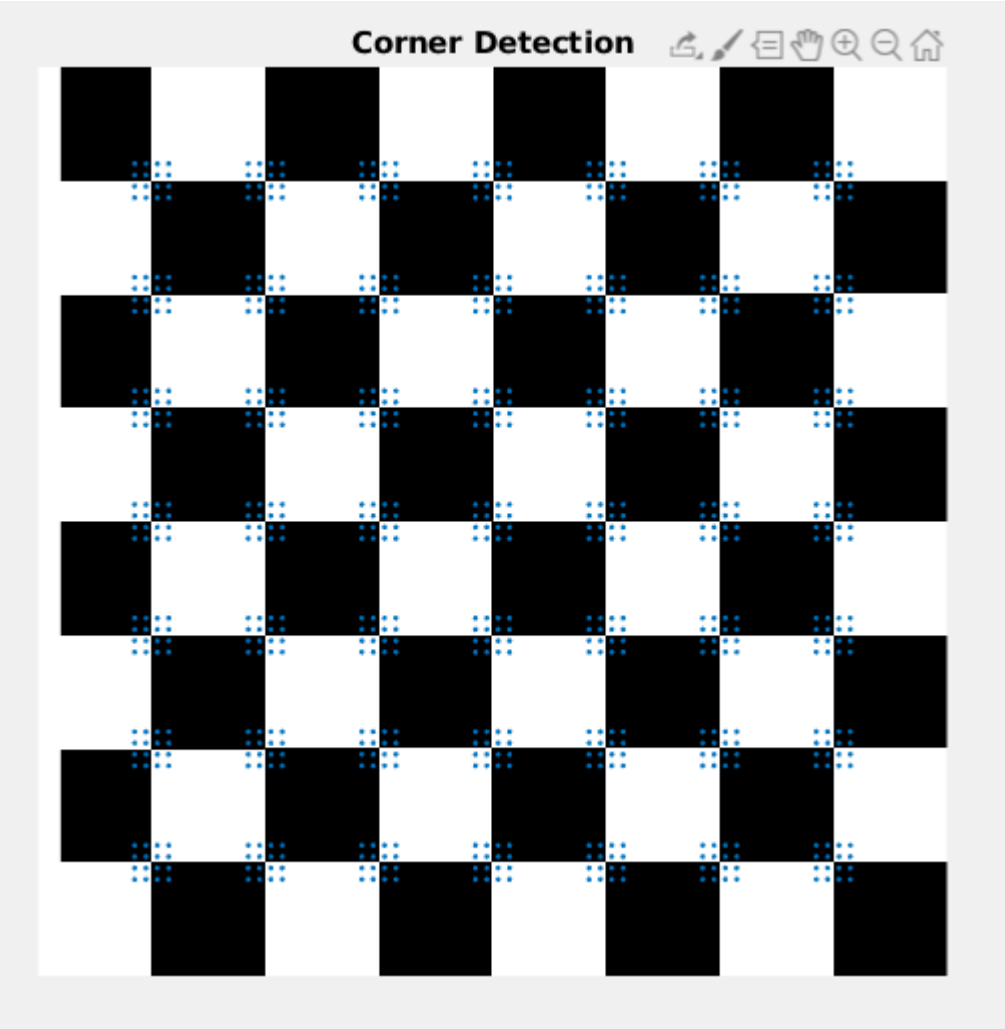
Assignment #1

In this task, we have to do corner detection with Kanade-Tomasi algoritm. The method is forming a specialized square and shifting it across the our image. While the square shifting, we may detect how the pixel values has changed by using eigen values. If the pixel count changed sufficiently we can consider this square center as a corner. To achieve that first we have to calculate the gradients Gx and Gy of the image by using gradiendtxy function which is already built in MATLAB. Then we have to create H matrix with image gradients Ix and Iy. In our implementation, the 3x3 kernel size is assumed. To calculate Ix^2 , Iy^2 and Ixy, first we multiply all values in matrix X or Y according to which image gradient is asked for. Then, we summed up all results. To illustrate following statements have been used to calculate sub-image gradients.

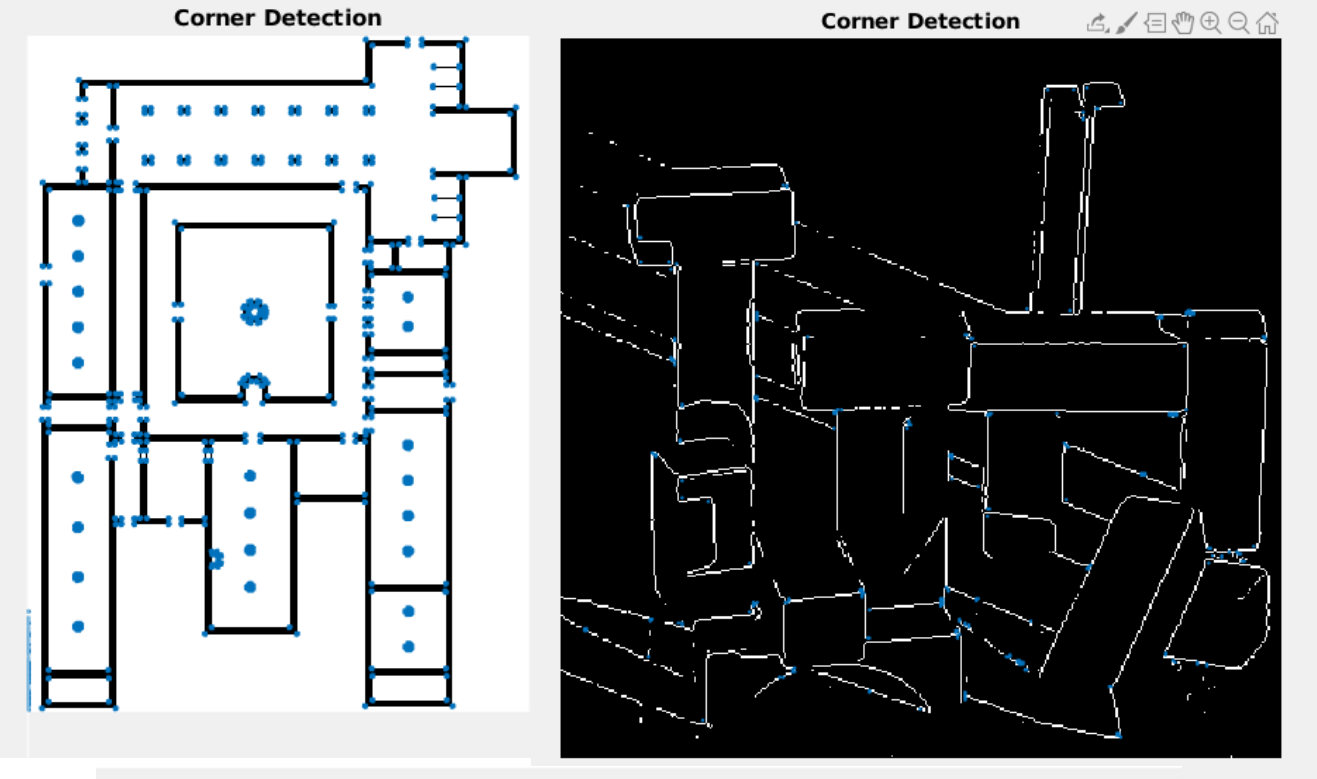
```
Ix2 = sum(sum(subImageofGx.*subImageofGx));
Iy2 = sum(sum(subImageofGy.*subImageofGy));
Ixy = sum(sum(subImageofGx.*subImageofGy));
```

After that step, we were available to form the H 2x2 matrix with [Ix2 Ixy; Ixy Iy2] values. If the two eigen value of that H matrices I larger than threshold, it is likely to specify that pixel as a corner. Which means, to eliminate fake corners we should pick small threshold. In our example, we assigned out threshold to 20. Unfortunately, there was no time to visualize corner in proper way. Therefore, by using plot function we have drawen some blue dots by looking at the array that we assigned pixels. Nevertheless, the corner representation is clear to persuade. Please check figures below.

F.T : In some example like ‘blocks.png’ which you can find in Appendix, have better results if we do edge detection first.



EE417 - Computer Vision – Post-Lab Report #4
Hakan Buğra Erentuğ – 23637
31 / 11 / 19 – Mehmet Keskinöz



E
H
3

```
function corners = lab4ktcorners(img, t)


if(length(size(img))==3)
    img = rgb2gray(img);
end
img=double(img);
[r,c]=size(img);
I=img;
img=imgaussfilt(img);
%img=edge(img,'Prewitt');
corners=[];
[Gx, Gy] = imgradientxy(img);

for i =2:1:r-2
    for j=2:1:c-2

        subImageofGx=Gx(i-1:i+1, j-1:j+1);

        subImageofGy=Gy(i-1:i+1, j-1:j+1);
        Ix2 = sum(sum(subImageofGx.*subImageofGx));
        Iy2 = sum(sum(subImageofGy.*subImageofGy));
        Ixy = sum(sum(subImageofGx.*subImageofGy));

        H=[Ix2 Ixy; Ixy Iy2];
        eigs = eig(H);

        if min(eigs) > t
            corners=[corners; i, j];

        end

    end

end


img=uint8(img);
I=uint8(I);


hold on; plot(corners(:,2),corners(:,1),'.');

title('Corner Detection')

end
```

EE417 - Computer Vision – Post-Lab Report #4
Hakan Buğra Erentuğ – 23637
31 / 11 / 19 – Mehmet Keskinöz

Unfortunately, there is no command line to explain how code works due to lack of time. Edge detection and Gaussfiltering is preferable so fell free to remove and add that statements.

Assignment #2

In the assignment, we have to detect lines with hough transform. Hough transform is basically a method which turns the points to lines and lines to points. In more detail, to detect a edge which has equation ax+b=y, you can find corresponding point in new hough space by looking b = -xa+y which is derived and suitable for our new space. In our image, if two point share same edge, their lines in hough space should intercept in one point. The point is also corresponding of the shared line. Lucklily, we were able to use built-in functions in MATLAB.

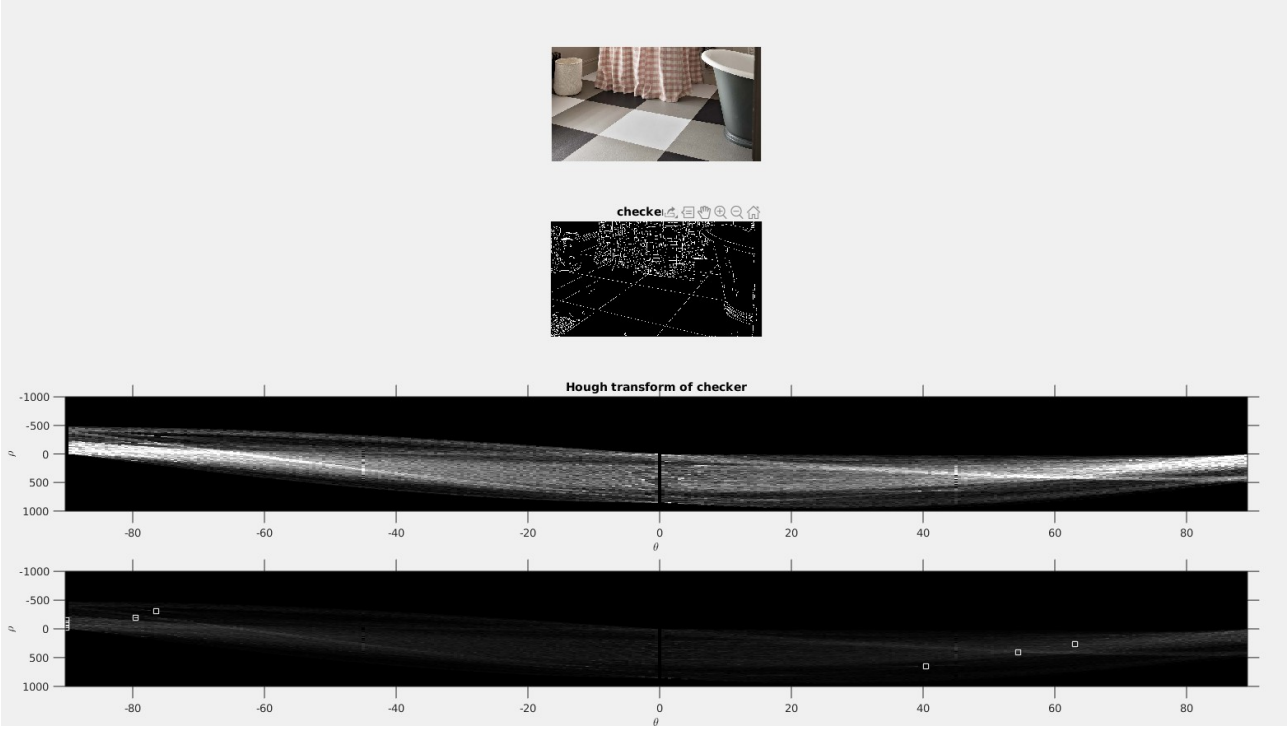
First, we have to detect edges via some built-in functions. In put implementation we have used the Canny Edge Detection. Then we applied hough function to that image with parameters 0.5 rho resolution and [-90,0.5,89] there value. In hough space, it is better to show lines in rho format because the length and angle is limited. Angle is limited with 180 degree and length is limited by image size. With hough function we obtained three result, H T and R where H is hough graph, T is theta and R is rho value. Then we have to draw a graph of hough transformation. Unfortunately, I could not manage to do that because there were many parameters and function that I did now learn yet. Therefore, I found a code block in matlab documentation in "https://www.mathworks.com/help/images/ref/hough.html" and draw the hough graph properly.

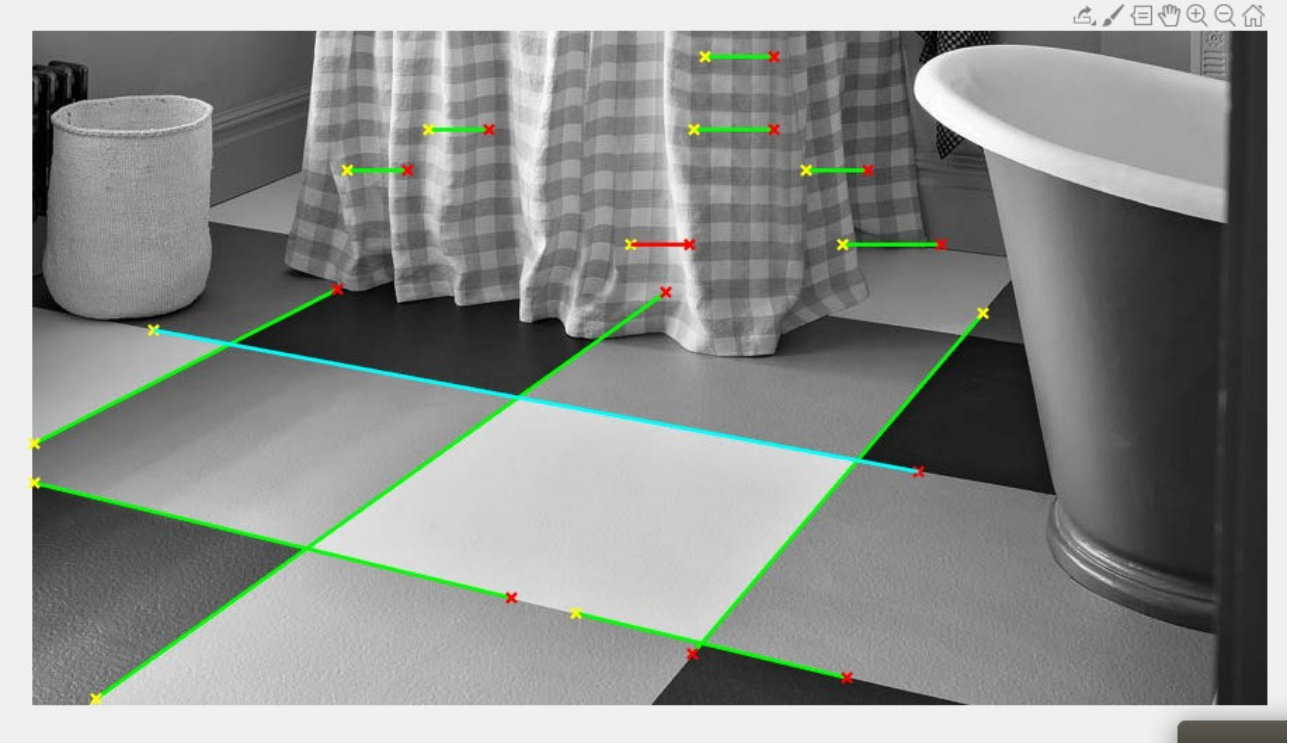
The next thing we have to do is counting the intersecting points in hough graph because each intersecting point means, some pixels share a line. First, I think implement a accumulotor but fortunately, there is already built in function in MATLAB which called houghpeaks and houghlines. Again, there were many function that I do not how to use, so I used some designed codes in matlab page. The results of houghpeaks, are our shared lines and houghlines is able to calculate them easily.

The last task is plotting the lines in different colors properly. We have used the regular and stardart max min algoritm to find longest and smallest lines and captured these values in min_len and max_len variable to plot them later. For the rest of the lines, MATLAB provided some codes to how to draw lines in different colors. In lab, TA's give permission to use these codes. Finally, we draw max and min length with different colors to obtain output. Please verify the results and matlab function which attached below. *F,N: I used Ubuntu 18.04 for the test environment. I believe operating system causes some errors in matlab code and draw the lines to last figure. However, some times instead of drawing lines to main image, it draw colored lines to houghpeaks graph. Please run the code multiple time to obtain correct result. Also, it is highly recommended to execute each tasks individually.*

The elapsed time have been calculated with tic-toc commends to see the performance of hough transform. The elapsed time with all plotting and drawing functions is 1.879779 second. The test environment is Lenova-Y520- 15IKBN, 15.% GiB memory, Intel® CoreTM i7-7700HQ CPU @ 2.80GHz × 8, Intel® HD Graphics 630 (Kaby Lake GT2) ,Ubuntu 18.04.3 LTS ,251 GB free disk space and '9.7.0.1190202 (R2019b)' MATLAB version.

EE417 - Computer Vision – Post-Lab Report #4
Hakan Buğra Erentuğ – 23637
31 / 11 / 19 – Mehmet Keskinöz





```
function returnValue = lab4houghlines(img)
    figure;
    subplot(4,1,1)
    imshow(img);
    [r,c, ch] = size(img);
    if ch == 3
       img = rgb2gray(img);
    end
    BW = edge(img, 'Canny');
    subplot(4,1,2);
    imshow(BW);
    [H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89);

    title('checker.png');
    subplot(4,1,3);
    imshow(imadjust(rescale(H)),'XData',T,'YData',R,...
          'InitialMagnification','fit');
    title('Hough transform of checker');
    xlabel('\theta'), ylabel('\rho');
    axis on, axis normal, hold on;

    subplot(4,1,4)
    P  = houghpeaks(H, 20, 'Threshold',0.5*max(H(:)));
    imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
    xlabel('\theta'), ylabel('\rho');
    axis on, axis normal, hold on;
    plot(T(P(:,2)),R(P(:,1)),'s','color','white');

    lines = houghlines(BW,T,R,P,'FillGap',10,'MinLength',40);

    max_len = 0;
    min_len = 200;

    figure, imshow(img), hold on
    for k = 1:length(lines)
       xy = [lines(k).point1; lines(k).point2];
       plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');


       plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
       plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');


       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',2,'Color','cyan');
    plot(xy_short(:,1),xy_short(:,2),'LineWidth',2,'Color','red');
end
```

EE417 - Computer Vision – Post-Lab Report #4
Hakan Buğra Erentuğ – 23637
31 / 11 / 19 – Mehmet Keskinöz

Self-Critisism about Assignment #2

I consider myself unsuccessful for this task. Even though, I managed to obtain some results, most of the code was not my own work. I am satisfied with understanding how line detection works and I am sure I can imperilment this code by my own next time. Still, my design have many bugs, referenced statements and lack of readability. Most of the codes taken from MATLAB page and forms. In addition, there is no proper visualization commend lines and consistency. I could not manage the time successfully and spend many minutes on assignment 1. That is why, I cannot give proper design. I am aware of my deficient and any point break is acceptable.

Assignment #3

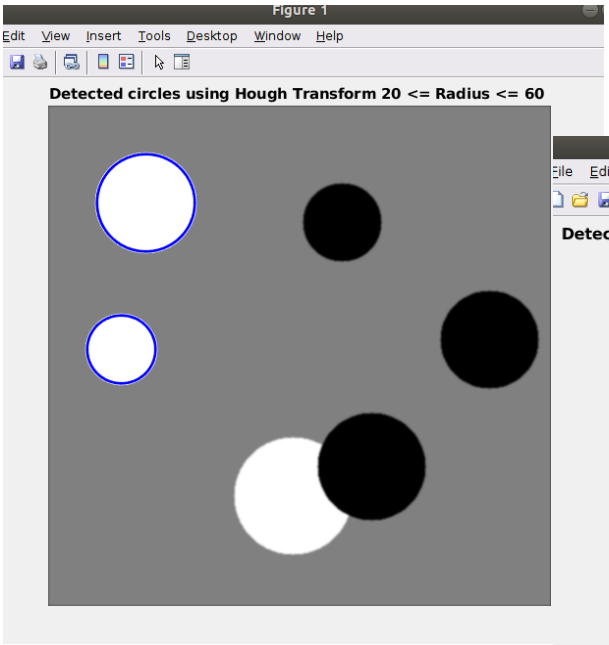
In this task, we supposed to do circle detection via Hough transformation in general. However, the TA's ease the job and gave permission to use built in MATLAB functions "imfindcircles". In general, we would use the same method that we just did above. Unlike the line detection, we would use 3D space due to number of parameters. For line two parameter a and b is enough. However, for circles we need three parameter, x and y value of the center and radius. Due to hard implementation of accumulator, imfindcirles function is the best choice. Also another built in function "viscircles" has been used. Viscircles's function is plotting circle on the image.

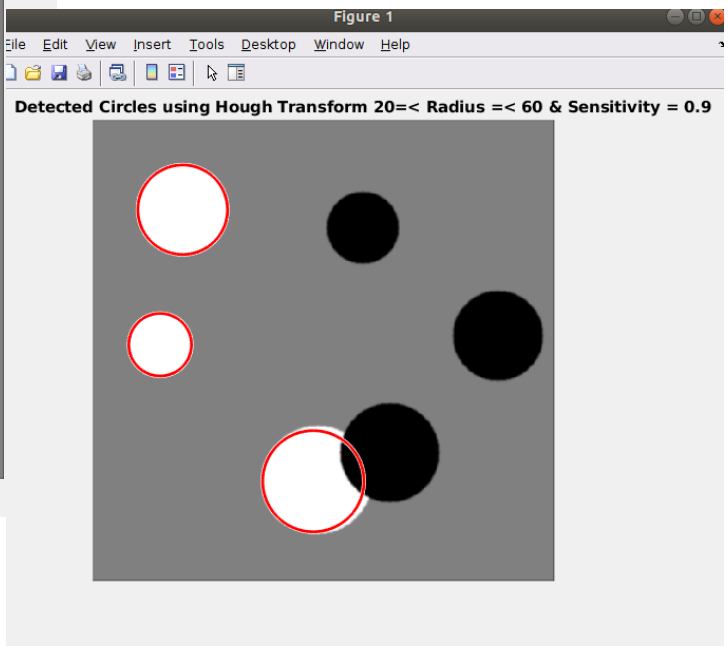
Firstly, we called the imfindcircles by proper parameters for first task which has 100% sensitivity by default. Sensitivity means how much displace will be tolerated, so for first figure, it can only find entirely correct circles. Below, the general statement has placed to see in detail.

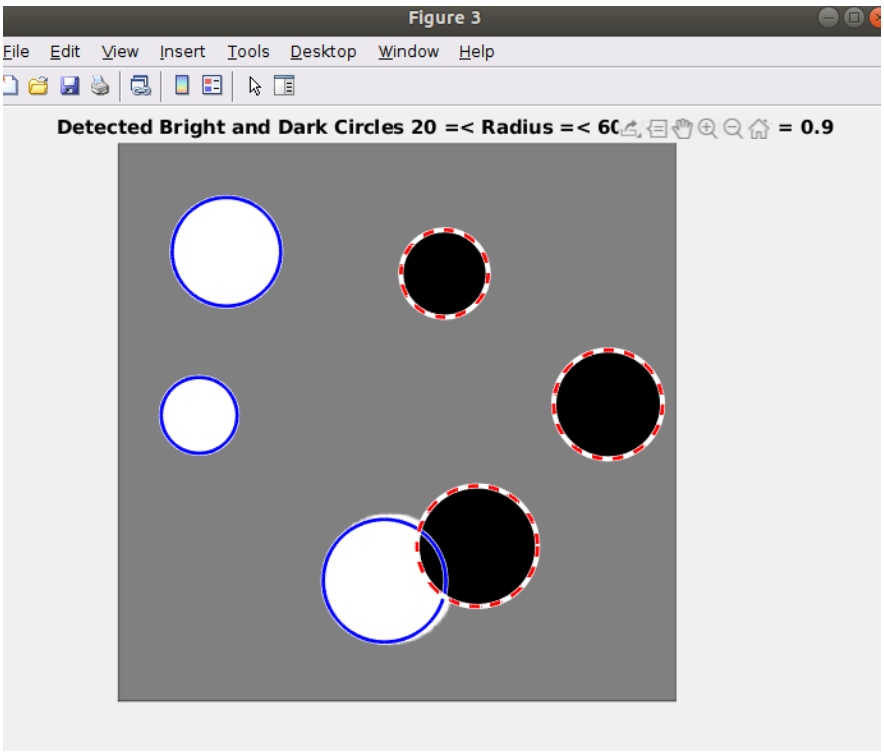
```
[whites_center, whites_radi] = imfindcircles(img,[20 60],'ObjectPolarity','bright',
'Sensitivity',0.9);
```

The function returns two variable, centers of the circles and radius of the centers. Respectively, parameters are the variable of image, radios limits, pixel value bias (tenderness to detect white or black), and toleration level. Luckily, viscircles and imfindcircles can coopearte very easily. To utilize viscircles only two parameter is required, centers of circles and radious of circles. Optionally you can input line style or color. Please verify the results and check the code that designed. Unfortunately, there was no time to put titles on figures. Therefore I did two updates to my previous code that I already sent during lab assignment. First, I put titles on the figures and secondly, I removed double casting of image, which is unnecessary and eliminate gray pixels. Please take these updates into consider while compare two codes.

EE417 - Computer Vision – Post-Lab Report #4
Hakan Buğra Erentuğ – 23637
31 / 11 / 19 – Mehmet Keskinöz







EE417 - Computer Vision – Post-Lab Report #4
Hakan Buğra Erentuğ – 23637
31 / 11 / 19 – Mehmet Keskinöz

```
function [returnValue] = lab4houghcircles(img)

if(length(size(img))==3)
    img = rgb2gray(img);
end

[r,c]=size(img);

    figure;
        imshow(img)
        [whites_center, whites_radi] = imfindcircles(img,[20 60],'ObjectPolarity','bright');
        viscircles(whites_center, whites_radi,'Color','b');
        title("Detected circles using Hough Transform 20 <= Radius <= 60 ");
%     figure;
%         imshow(img)
%     [whites_center, whites_radi] = imfindcircles(img,[20 60],'ObjectPolarity','bright',
'Sensitivity',0.9);
%     viscircles(whites_center, whites_radi,'Color','r');
%     figure;
%     imshow(img)
%     [whites_center, whites_radi] = imfindcircles(img,[20 60],'ObjectPolarity','bright',
'Sensitivity',0.9);
%     [darks_center, darks_radi] = imfindcircles(img,[20 60],'ObjectPolarity','dark');
%     viscircles(whites_center, whites_radi,'Color','b');
%     viscircles(darks_center, darks_radi,'LineStyle','--');


end
```

```
clc;clear all;close all;

% Hakan Bugra Erentug

% Task 1
% I = checkerboard;
% I=imread('blocks.png')
% imshow(I);
% I=rgb2gray(I);
%
% imshow(lab4ktcorners(I,20))

% Task 2
% tic
% I2= imread('checker.jpg');
% lab4houghlines(I2);
% toc

% Task 3
I=imread("circlesBrightDark.png")

lab4houghcircles(I);
```