# EE 417 LAB 4 POSTLAB REPORT

**NAME:** *Nidanur GUNAY*

**ID:** *24231*

- **Corner Detection:** In order to detect the corners in a grayscale image by employing Kanade-Tomasi algorithm, you can apply the following steps:
  - Create a checkerboard image by calling the built-in command `checkboard` in MATLAB
  - Compute the gradients $G_x$ and $G_y$ of the image (Hint: You can use `[Gx,Gy] = imgradientxy(I);` )
  - Create $H$ matrix of each pixel in a window as follows

$$H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \quad (1)$$

    where $I_x$ and $I_y$ are the image gradients of a window along $x$ and $y$ directions, respectively.
  - Compute the eigenvalues $\lambda_1$ and $\lambda_2$ of $H$.(Hint: You can use `eig command in MATLAB` )
  - If $\min(\lambda_1, \lambda_2)$ >Threshold, add the pixel coordinates to corner list.

Now write a function which takes an image and a threshold as inputs and utilize "**Kanade-Tomasi Corner Detection Algorithm**" to return the detected corner points. Your function name should be "lab4ktcorners.m".

Detect the corners in other two images provided and your results should look as :

Fig 1.1

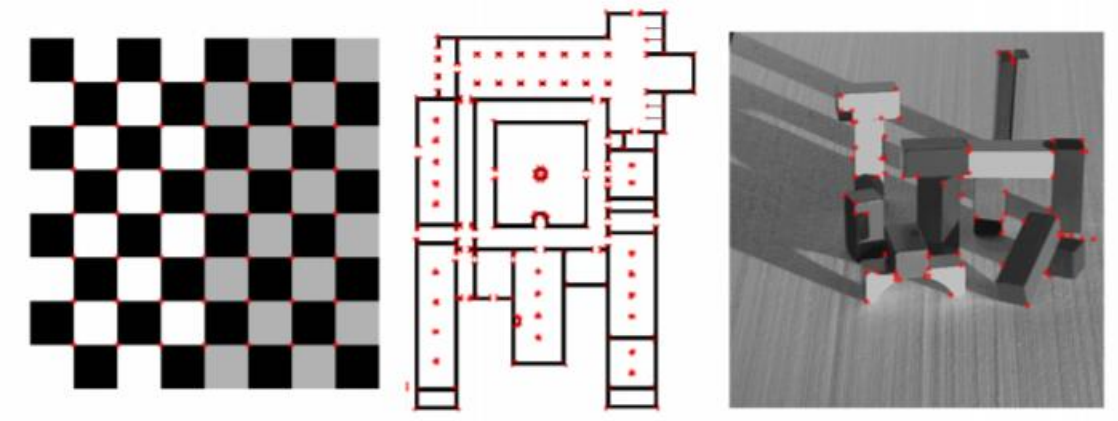


Fig 1.2

- In the first task we need to implement a corner detection algorithm. It is usually used in motion detection, image restriction, video tracking, 3D modelling and object recognition etc. Firstly, we need to create a chekerboard image. An then we need to gradient x and gradient y image. We will use that gradients in order to find the windows of an image that generate a large variation when we moved around. To get a window we created windows by inner two for loops and we've taken the windows from our gradient matrices. We can detect the how the pixels are changing with the help of the eigenvectors. If value of the change is larger than threshold that is detected by the user, we can consider that pixel as a corner. We used built in functions in MatLab.

- By the summation of value's squares in gradient X's each pixel in windows we get the summation of x derivatives and calculate the $Ix^2$, when we trace same steps for the gradient Y we get the $Iy^2$ , by the summation of each pixels gradient x and gradient y we calculated the *Ixy.* We constructed our H matrix to replace that values in the H matrix. Then we get H matrix's eigenvalue by the eig built in function in MatLab. After the all of the steps my code can detect the corners as seen in figure 1.3.

- However, during the lab I wasn't able to complete this task. Although, my algorithm was totally correct my code didn't work properly. Only reason of that I wrote eigs instead of eig built in function. This small mistake took my al lab hour and I couldn't achieve the other tasks.

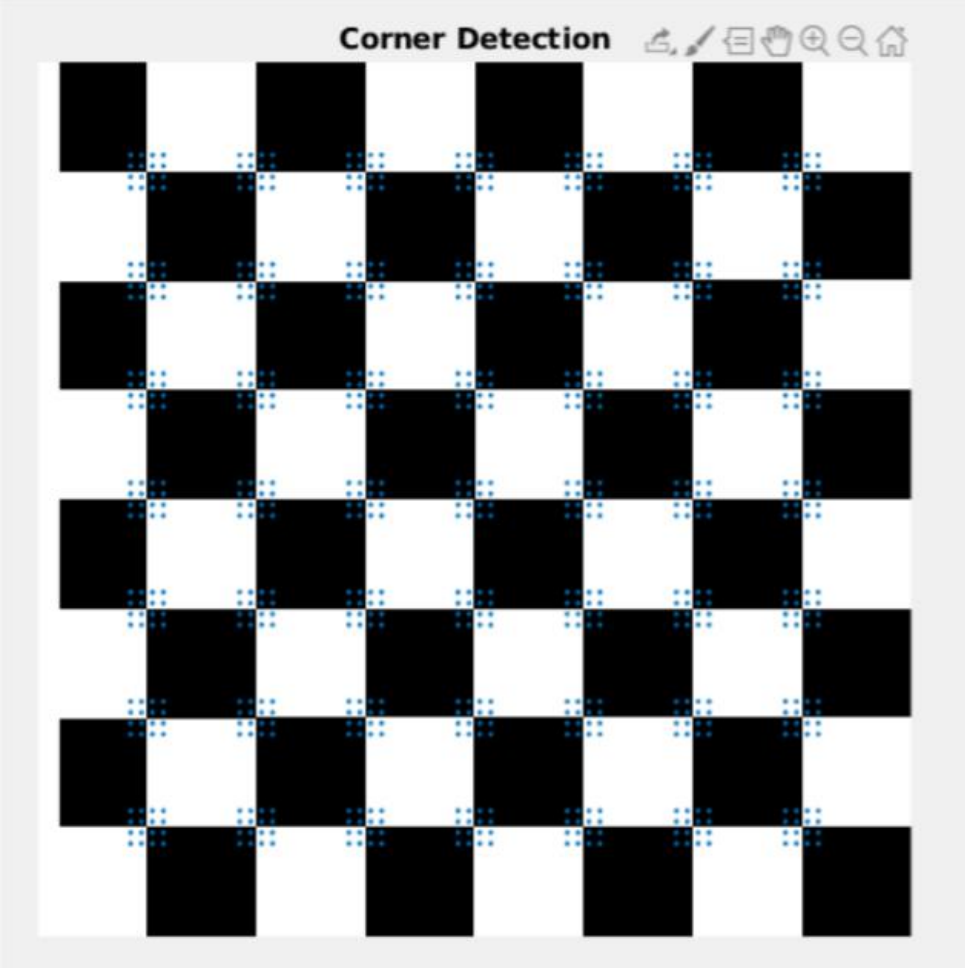
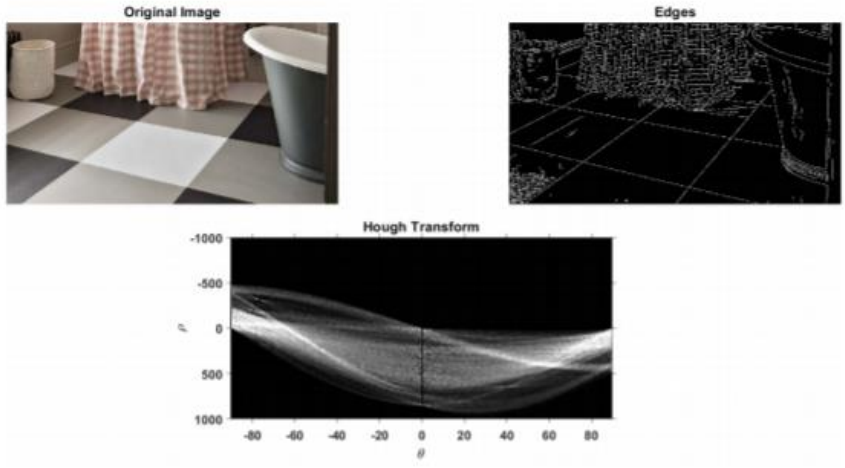



Fig 1.3

**Line Detection**

- Read an image named `checker.jpg` and convert it to a black-white edge image with your edge detector choice.
- Obtain the Hough transform of the black-white image and display it along $\rho$ vs $\theta$ axes. *Hint*: use `'hough'` function




- Select the peak Hough points with an appropriate threshold (e.g., half of the maximum Hough points). *Hint*: use `'houghpeaks'` function

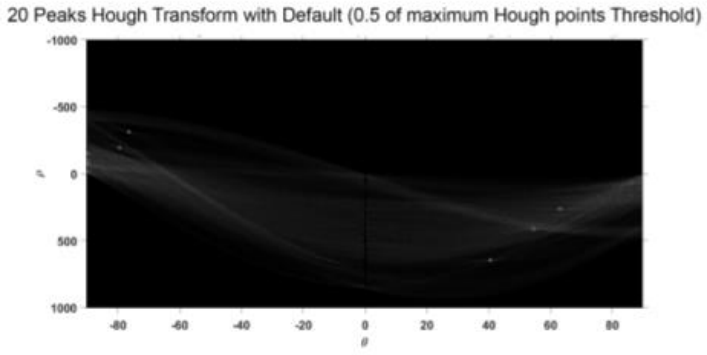



Fig2.1

- Find the lines in the image by using these peak points *Hint*: use `'houghlines'` function
- Plot all the lines that you detected with green color and highlight the longest and shortest detected lines with cyan and red color, respectively. What are the maximum and minimum lengths of the detected lines?

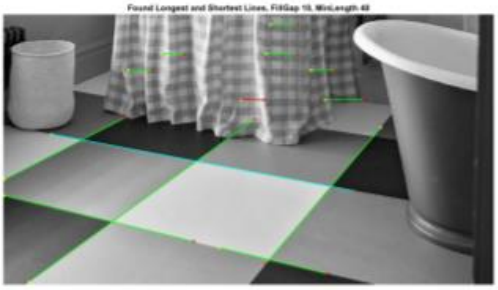


Fig 2.2

- Second task in our lab assignment is line detection. In order to detect lines, we will use the Hough transform. For Hough Transform, we need to express lines in Polar System. Each point in x and y space is line in the m and c space. However, before the applying transform we need to use edge detection algorithm. For each nonblack point, we draw a line in the m and c plane. Each intersect point are the parameter of the lines. Their position is represented using a direction theta and a distance r - instead of x and y. To compute the frequency of occurrence, theta and r are discretized. In my code, I showed first output in my main function and the code that I've searched gives us the Hough transform graphs. Detection of the intersect lines in the Hough Transform gives us the equation of the lines in image. In my lab4houghlines function I plot the lines in my main image. I stored my Hough lines in lines array and for each line I plotted a green line. In the for loop I calculated the minimum and maximum length lines and I set the maximum length lines color as cyan and minimum length of line as red. My resulted images are in Fig 2.3 and 2.4.

- Previous steps have done after the lab because I had trouble with the time. During the lab I couldn't write my code because I lacked searching skills in limited time. This is the thing that I should improve. However, after the lab I searched very well and solved how to implement the codes.

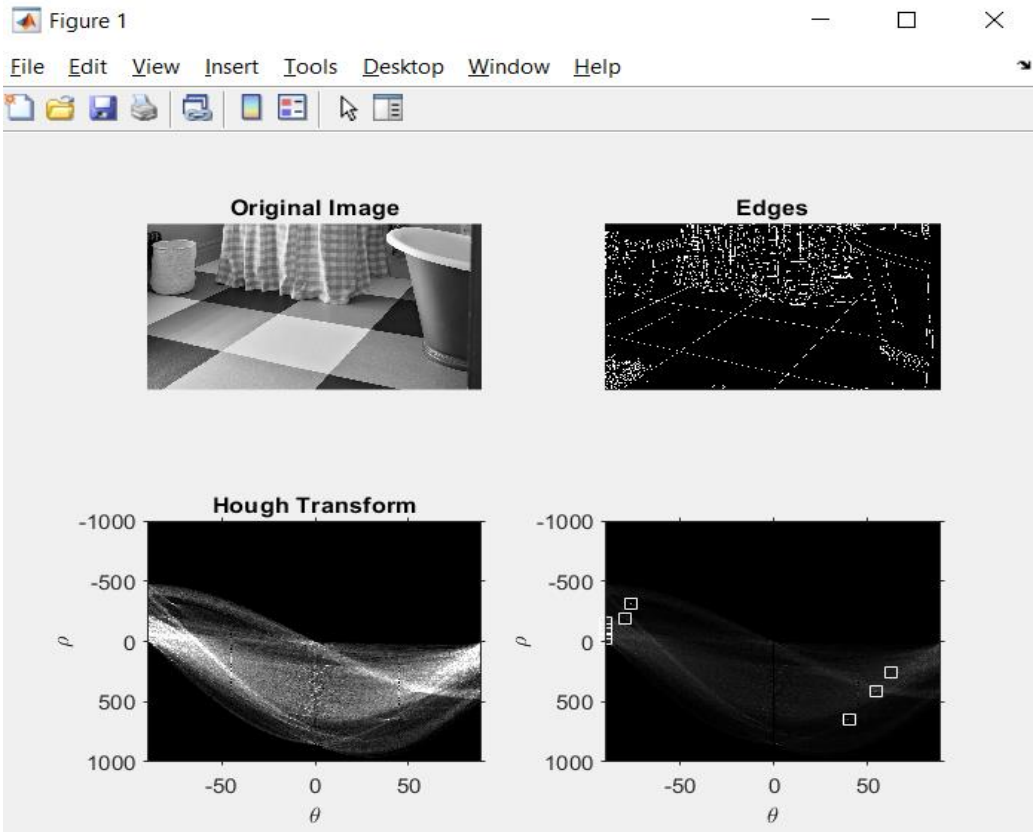


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Edges
Hough Transform
-1000
-500
0
500
1000
ρ
-50
0
50
θ

Fig2.3

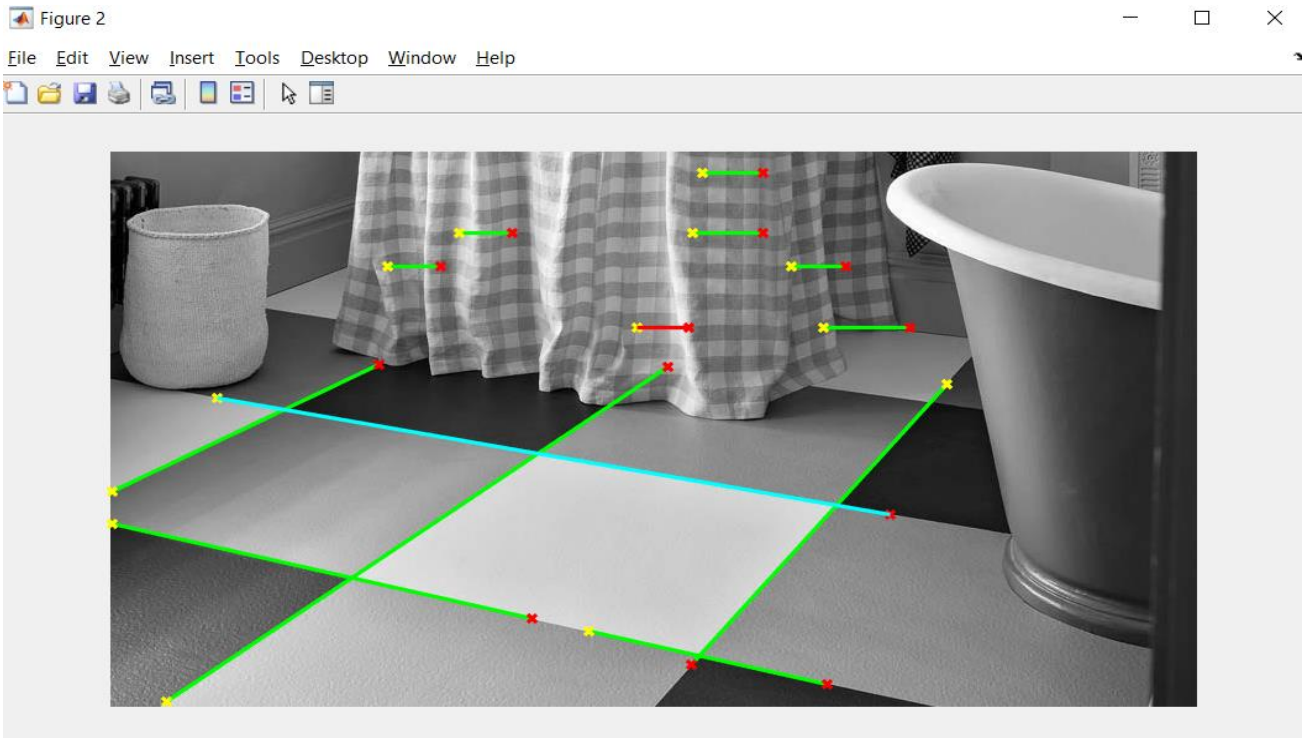


Figure 2
File Edit View Insert Tools Desktop Window Help

Fig 2.4

**Circle Detection**

- Load a demo image of MATLAB named `'circlesBrightDark.png'` , which contains several circles with different sizes.
- Convert it to a black-white image.
- Detect all the circles with radius $r$ such that $20 \leq r \leq 60$ pixels by using Hough transform.
  *Hint*: use `'imfindcircles'` function
- Change `'Sensitivity'` factor and test the performance of circle detection
- Change `'ObjectPolarity'` parameter to detect `'bright'` and `'dark'` circles separately

  You can call the function as `imfindcircles(I,[Rmin, Rmax],'Parameter',value)` where parameter can be `'Sensitivity'` and value is a number between 0 and 1, or `'ObjectPolarity'` and value is either `'bright'` or `'dark'`.

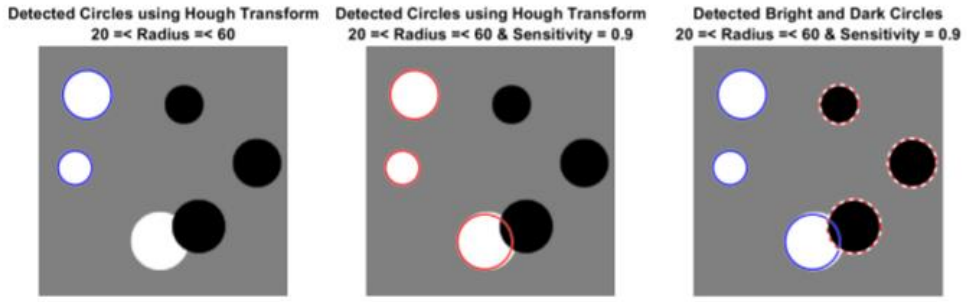


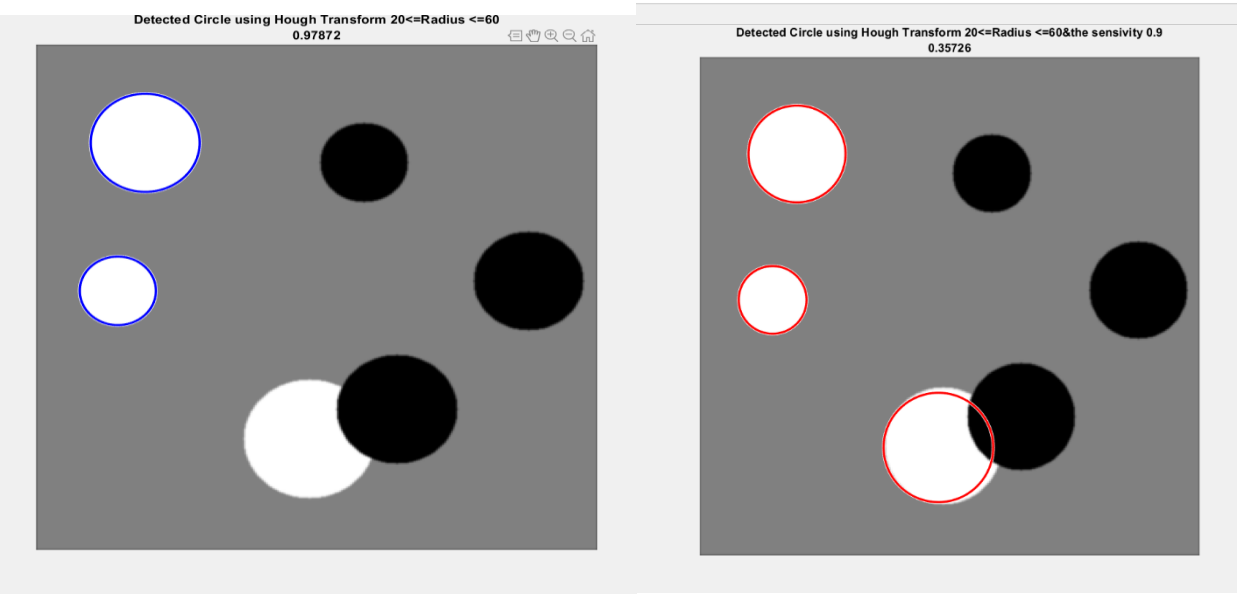

Fig 3.1

- Third task that we supposed to do is detecting circles. Since circlesBrightDark.png is a grayscale image in my main function I get the error while I was converting my image. With the help of the imfindcircle built in function in MatLab, I detect the circles that have a radius between 20 and 60. When we change the sensitivity as 0.9, we get 1 more circle in our output. Different circle is the circle that intersect with the other dark circle and its shape is not exactly like circle when we decrease the sensitivity, we can detect that circle. After that steps we add the dark circles in the detected circles with the help of the ObjectPolarity parameter.

- My after-lab searches are in above. Unfortunately, I wasn't able to fulfill that task due to lack of time during the lab. First task took my all lab hours. Thus, I couldn't even start that task. Time management is the main gain from the 4$^{th}$ lab.

- Time for the calculation of the code for detecting the circles that have a radius between 20 and 60 is 0.97872, for the same circles but with the 0.9 sensivity 0.3526, detecting bright and dark circles took 0.84657 sec as you can see in Fig 3.2.

- When I use another image for the same code my results are 3.028 and 1.2738 second for the detection of bright and dark circles and the circles with sensivity 0.9. I conclude that result as when we increase the radius difference for the detected circles, algorithm works slower.



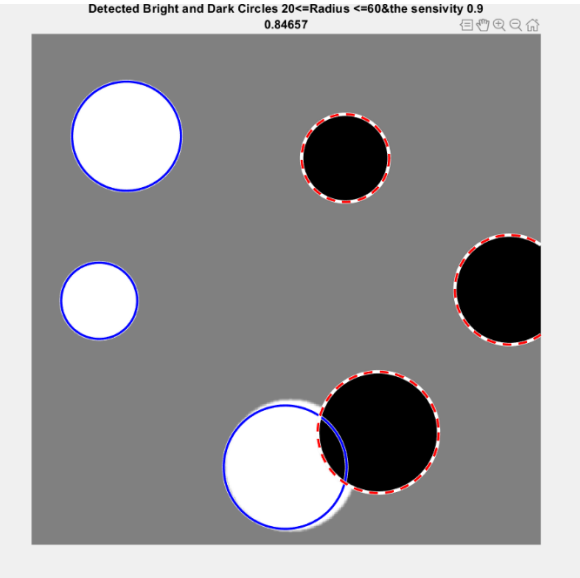

Detected Bright and Dark Circles 20<=Radius <=60&the sensivity 0.9
0.84657

Fig 3.2


Detected Bright and Dark Circles 60<=Radius <=1000&the sensivity 0.9
3.028
Detected Circle using Hough Transform 60<=Radius <=1000&the sensivity 0.9
1.2738

Fig 3.3

# ***APPENDIX: USED CODES:***

## **MAIN FUNCTION:** *lab4.m*

```
clc;
clear all;
close all;
i=imread("Monastry.bmp");
k=imread("checker.jpg");
k=rgb2gray(k);
l=imread("blocks.png");


check=checkerboard;
lab4ktcorners(l,50);



%%%%%%%%%%%%%%%%%%%%%%%%
figure
subplot(2,2,1);
imshow(k);
title("Original Image");
subplot(2,2,2);
k1=edge(k,"Canny");
imshow(k1);
title("Edges");
subplot(2,2,3);
[H,T,R] = hough(k1,'RhoResolution',0.5,'Theta',-90:0.5:89);
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;
```

```
title("Hough Transform");
subplot(2,2,4);
   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');

lab4houghlines(k);

%%%%%%%%%%%%KANADE-TOMASI CIRCLE DETECTION%%%%%%%%
 circle=imread("circlesBrightDark.png");
%circle=imread("images2.png");
  [r,c] = size(circle);

  tic;
   figure;
       imshow(circle)
       [centersBright, radiiBright] = imfindcircles(circle,[20 60],'ObjectPolarity','bright');
       viscircles(centersBright, radiiBright,'Color','b');
     m=toc;
     title(["Detected Circle using Hough Transform 20<=Radius <=60", num2str(m)]);

       tic;
   figure;
       imshow(circle)
   [centersBright, radiiBright] = imfindcircles(circle,[20 60],'ObjectPolarity','bright', 'Sensitivity',0.9);
   viscircles(centersBright, radiiBright,'Color','r');
   d=toc;
    title(["Detected Circle using Hough Transform20<=Radius <=60 & the sensivity 0.9", num2str(d)]);
   figure;
   imshow(circle)
   [centersBright, radiiBright] = imfindcircles(circle,[20 60],'ObjectPolarity','bright', 'Sensitivity',0.9);
   [centersDark, radiiDark] = imfindcircles(circle,[20 60],'ObjectPolarity','dark');
```

```
viscircles(centersBright, radiiBright,'Color','b');
viscircles(centersDark, radiiDark,'LineStyle','--');
h=toc;
title(["Detected Bright and Dark Circles 20<=Radius <=60& the sensivity 0.9", num2str(h)]);
```

## LAB4KTCORNERS:

```
function lab4ktcorners (check, t1)

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

[r1,c1]=size(check);

check=double(check);
corners=[];

[gx,gy]=imgradientxy(check);

for  i=2:1:r1-2
   for j=2:1:c1-2
      subimgx=gx(i-1:i+1,j-1:j+1);
      subimgy=gy(i-1:i+1,j-1:j+1);

      ix=sum(sum(subimgx.*subimgx));
      ixy=sum(sum(subimgx.*subimgy));
      iy=sum(sum(subimgy.*subimgy));

      H=[ix,ixy;ixy,iy];
      eigvalue=eig(H);
      if ( min(eigvalue) > t1)
        corners= [corners; i,j];
```

```
    end
  end
end

figure;
check=uint8(check);
imshow(i);
hold on; plot(corners(:,2),corners(:,1),'r*');
title("Corner Detection");
```

## LAB4HOUGHLINES:

```
function  lab4houghlines(I)

  [r,c,ch] = size(I);

   if ch == 3
    I = rgb2gray(I);
   end
    K=edge(I,"canny");
   [H,T,R] = hough(K,'RhoResolution',0.5,'Theta',-90:0.5:89);

 xlabel('\theta'), ylabel('\rho');
   axis on, axis normal, hold on;
   P  = houghpeaks(H, 20, 'Threshold',0.5*max(H(:)));

  xlabel('\theta'), ylabel('\rho');
   axis on, axis normal, hold on;
   plot(T(P(:,2)),R(P(:,1)),'s','color','white');

  %%%%%%%%%%%%%%%%%%%
   lines = houghlines(K,T,R,P,'FillGap',10,'MinLength',40);
```

```
    figure, imshow(I), hold on
    max = 0;
    min = 200;
    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)
        max = len;
        xy_long = xy;
      end
      if( len < min)
         min = 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
```