EMİR ALAATTİN YILMAZ

EE417

POST-LAB #5 REPORT

Data Generation for Camera Calibration

1. Intersection of two lines

In this section of the lab we will use two lines which are obtained by Hough transform. After determining these two lines, we will use their θ and ρ values to solve them together in order to find the intersection point.

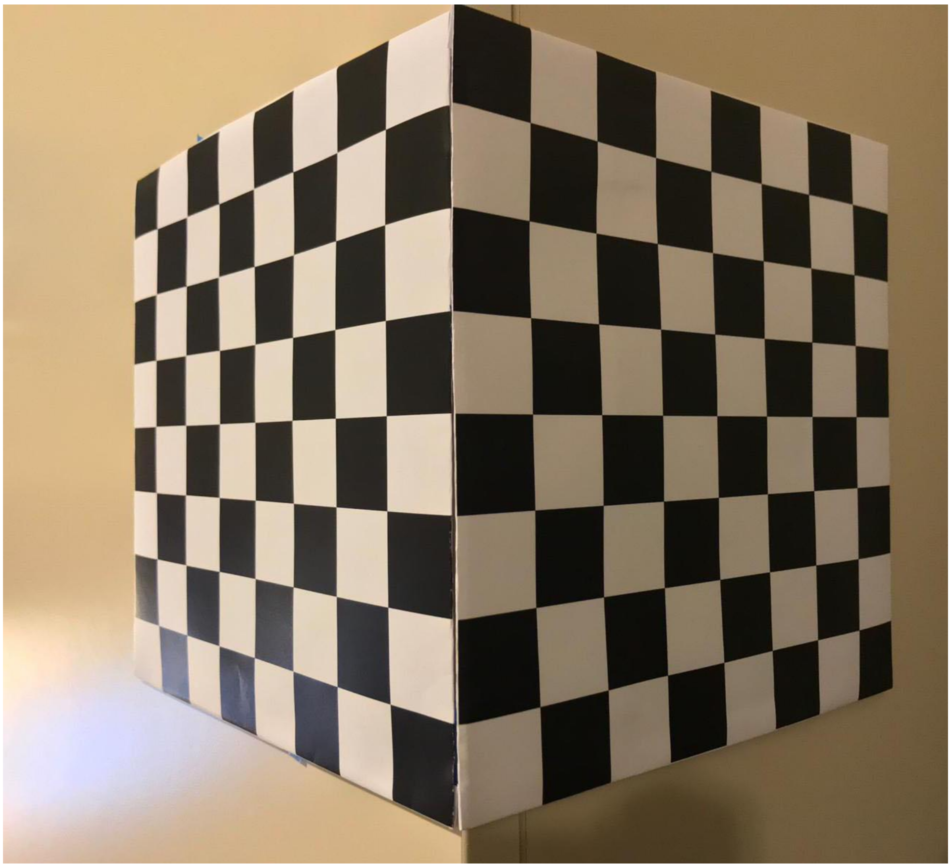
$$x \cos(\theta) + y \sin(\theta) = \rho$$

$$A = \begin{bmatrix} cos\theta_1 & sin\theta_1 \\ cos\theta_2 & cos\theta_2 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = A^{-1} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} \text{ (Intersection point)}$$

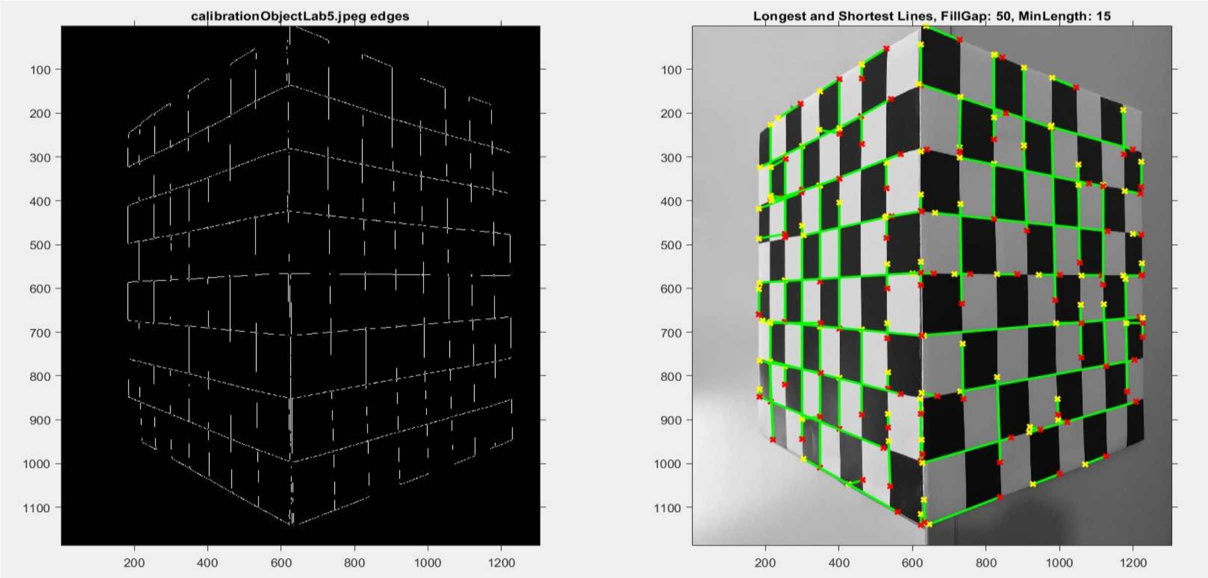
The main purpose of this lab is to find corners to calibrate our camera. In order to do that, first of all we need a calibration object. We used two pages of checker pattern sticked on the corner of the wardrobe in our dormitory which can be seen below.

CalibrationObjectLab5.jpeg



To find the lines in the image, we firstly determine the edges. We used Canny Edge Detector in our lab but we need to change its parameters in order to get more proper results. We determined our threshold values as [0.1 0.4].

After edges are determined, we applied Hough transform with the parameters of FillGap:50 and MinLength:15 to get the lines.



We manually selected two lines and used their θ and ρ values to solve them together with the following formula.

$$A = \begin{bmatrix} cos\theta_1 & sin\theta_1 \\ cos\theta_2 & cos\theta_2 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = A^{-1} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} \text{ (Intersection point)}$$

lines

1x106 struct with 4 fields

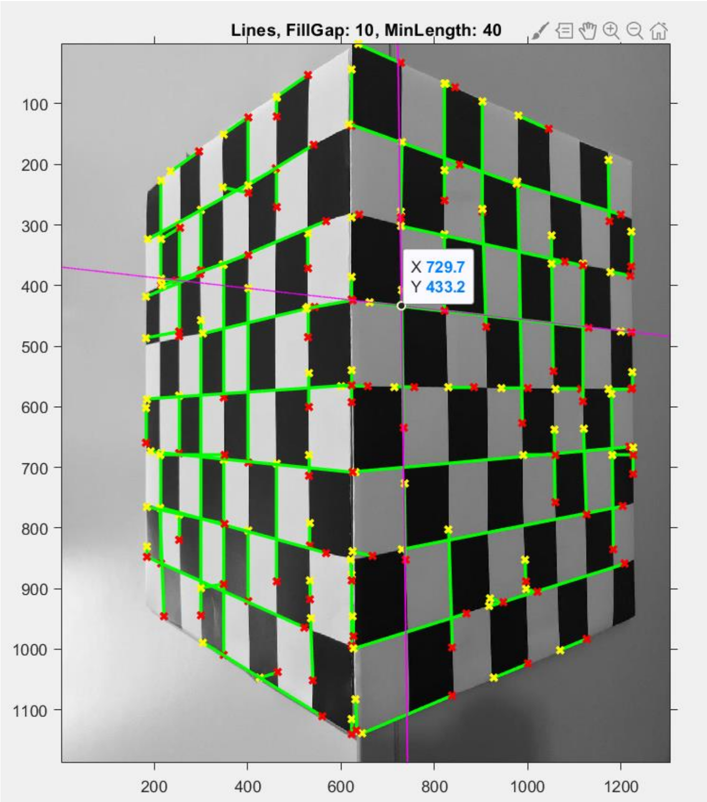
F...	point1	point2	theta	rho
31	[623,539]	[623,592]	0	622
32	[623,877]	[623,994]	0	622
33	[623,1115]	[623,1140]	0	622
34	[214,389]	[248,392]	-85	-368
35	[661,428]	[1130,469]	-85	-368
36	[1199,475]	[1222,477]	-85	-368
37	[426,1047]	[465,1038]	76.5000	1.1165e+03
38	[627,999]	[868,941]	76.5000	1.1165e+03
39	[918,929]	[1209,859]	76.5000	1.1165e+03
40	[300,899]	[349,892]	81.5000	932.5000
41	[731,835]	[1203,764]	81.5000	932.5000
42	[822,66]	[822,259]	0	821
43	[822,316]	[822,442]	0	821

lines

1x106 struct with 4 fields

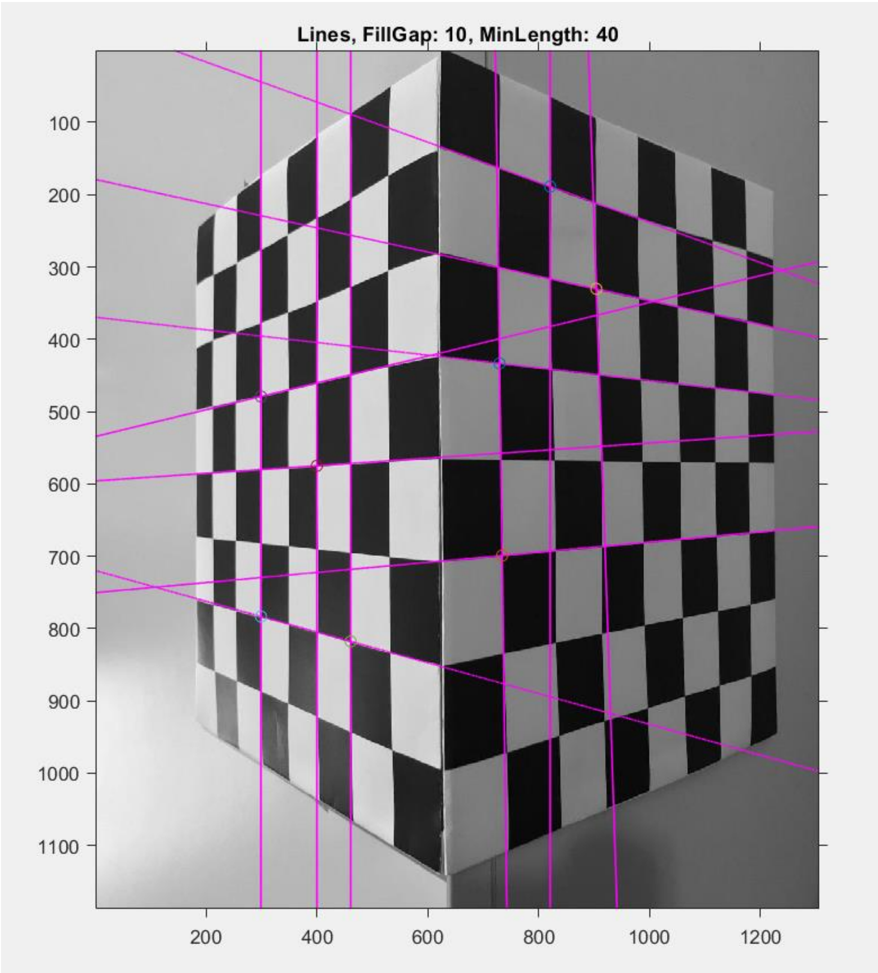
Fields	point1	point2	theta	rho
40	[300,899]	[349,892]	81.5000	932.5000
41	[731,835]	[1203,764]	81.5000	932.5000
42	[822,66]	[822,259]	0	821
43	[822,316]	[822,442]	0	821
44	[728,278]	[728,295]	-1	722
45	[730,408]	[734,634]	-1	722
46	[736,726]	[738,852]	-1	722
47	[1119,364]	[1119,591]	0	1118
48	[1173,192]	[1175,294]	-1	1.1685e+03
49	[1180,579]	[1184,835]	-1	1.1685e+03
50	[903,96]	[903,278]	0	902
51	[193,674]	[625,708]	-85.5000	-656
52	[979,229]	[989,627]	-1.5000	971.5000
53	[995,853]	[996,888]	-1.5000	971.5000

After selected these two lines we got the following result.



We found 8 corners by use of this approach

(They are in o form that can be seen in the image on the right)



2. Harris Corner Detection

In this section we use the Harris Corner Detection method which is built in method of MATLAB. The algorithm details are below:

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x.I_x \quad I_{y2} = I_y.I_y \quad I_{xy} = I_x.I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma\prime} * I_{x2} \quad S_{y2} = G_{\sigma\prime} * I_{y2} \quad S_{xy} = G_{\sigma\prime} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

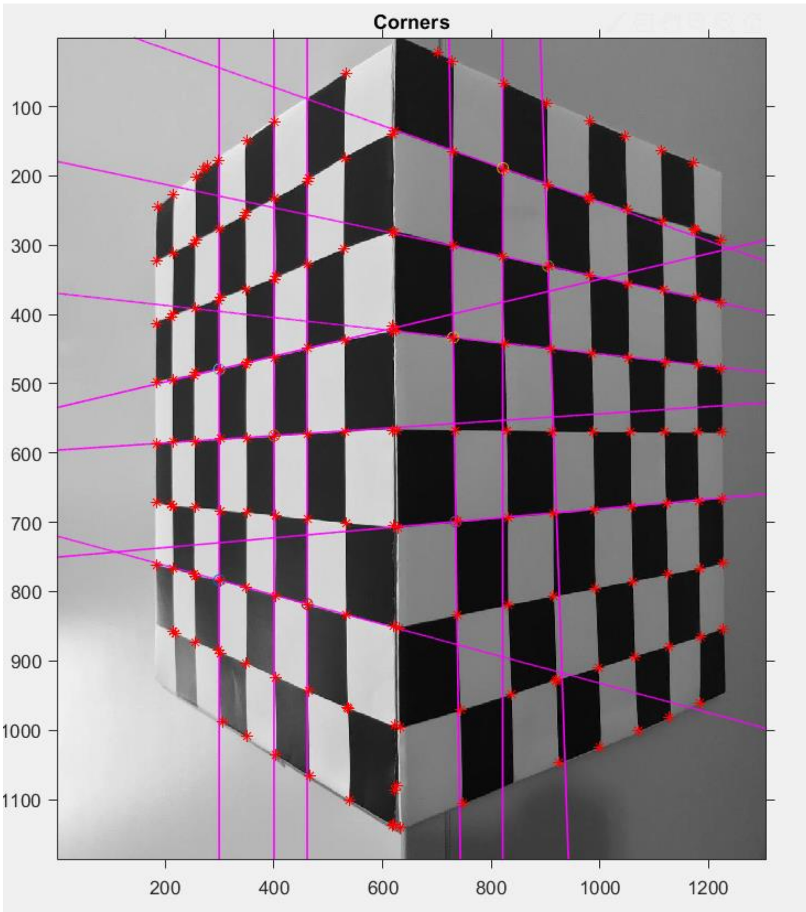
$$H(x,y) = \begin{bmatrix} S_{x2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

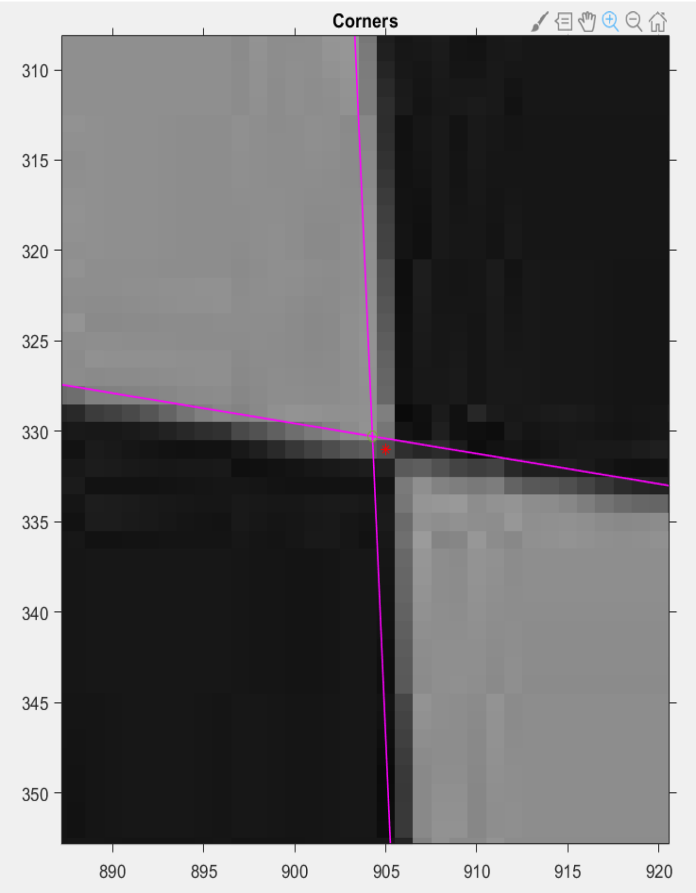
$$R = Det(H) - k(Trace(H))^2$$

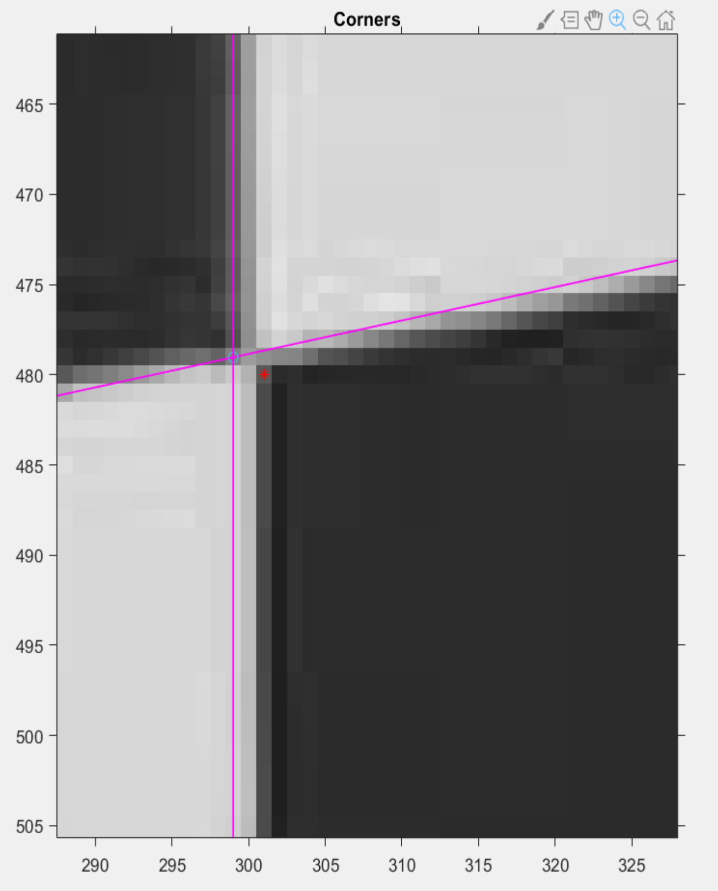
6. Threshold on value of R. Compute nonmax suppression.

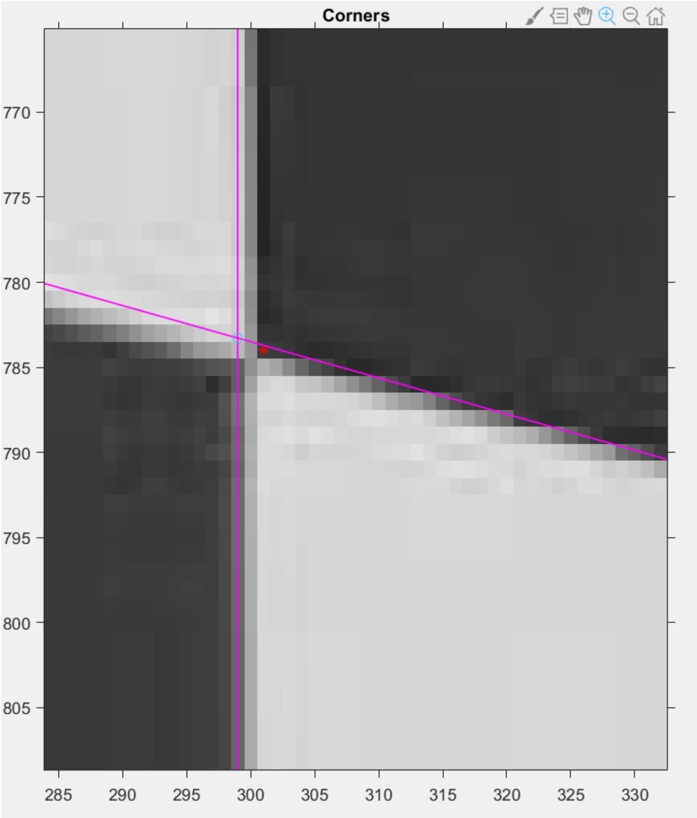
We applied Harris Corner Detection method and we get the following result. (Red spots are the Harris method results, o forms are obtained by intersection method)

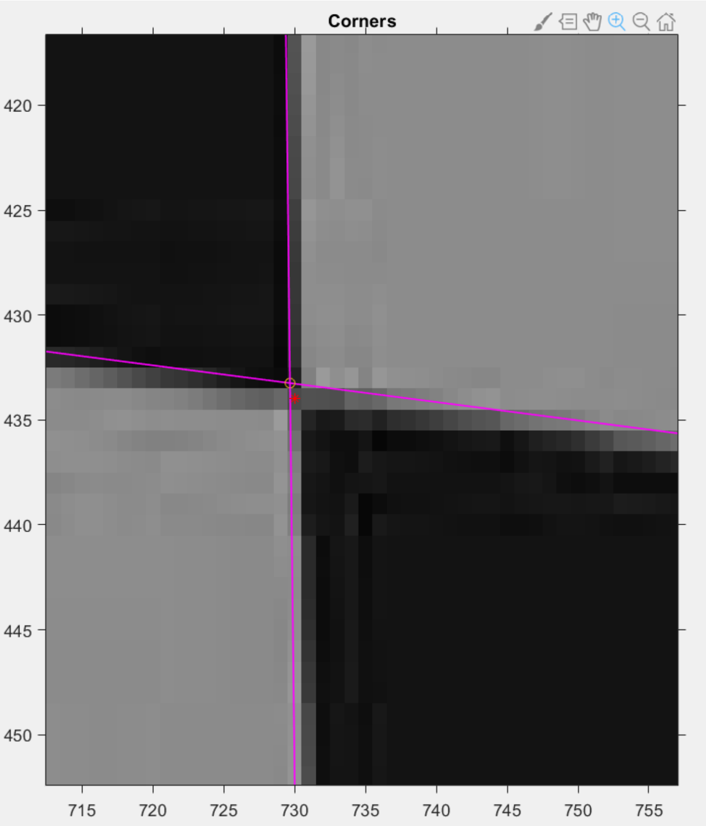


A Closer Look (Intersection point is shown in green circle, it is hard to see)









DISCUSSION

If we examine the results above, we can say that intersection method gives more precise results in comparison with Harris corner detection. On the other hand, use of intersection method requires selecting of lines manually. It takes time to examine which line is crossing with which one. But, I would still use intersection method for more precise calibration because it gives subpixel accuracy rather than Harris corner detection which gives integer values. If I would suggest another method to detect the corners, this would be Kanade-Tomasi method.

CODES

lab5.m

```
img = imread('calibrationObjectLab5.jpeg');

 [row,col,ch] = size(img);
 if(ch==3)
    img = rgb2gray(img);
 end

img2 =  edge(img,'Canny',[0.1 0.4]);

[H,T,R,edges,P,lines] = lab5calibprep(img);

subplot(1,2,1);
imshow(img2);
title('calibrationObjectLab5.jpeg edges');
axis on, axis normal, hold on;

subplot(1,2,2);
imshow(img);
title('Corners');
axis on, axis normal, hold on;

lab5drawlines(lines,45,35);
lab5drawlines(lines,45,14);
lab5drawlines(lines,81,60);
lab5drawlines(lines,2,71);
lab5drawlines(lines,17,69);
lab5drawlines(lines,2,69);
lab5drawlines(lines,4,68);
lab5drawlines(lines,66,42);

axis on, axis normal, hold on;
corners = corner(img,'Harris');
plot(corners(:,1),corners(:,2),'r*');
```

lab5calibprep.m

```
function [H,T,R,edges,P,lines] = lab5calibprep(img)

    [row,col,ch] = size(img);
    X = zeros(size(img));
    k = 1;

    if(ch==3)
        img = rgb2gray(img);
    end

    edges = edge(img,'Canny',[0.1 0.4]);

    [H,T,R] = hough(edges,'RhoResolution',0.5,'Theta',-
90:0.5:89);

    P = houghpeaks(H,500,'Threshold',0.1*max(H(:)));

    lines =
houghlines(edges,T,R,P,'FillGap',50,'MinLength',15);

end
```

lab5drawlines.m

```
function [intersection, y1,y2] = lab5drawlines(lines,L1,L2)

     tetha1 = lines(L1).theta;
     tetha2 = lines(L2).theta;

     if tetha1 == 0
         tetha1 = 0.001;
     end
     if tetha2 == 0
         tetha2 = 0.001;
     end

     rho = [lines(L1).rho; lines(L2).rho];
     A = [cosd(tetha1) sind(tetha1); cosd(tetha2)
sind(tetha2)];
     intersection = inv(A)*rho;

     x1 = 0:0.1:2000;
     x2 = 0:0.1:2000;
     y1 = (rho(1)-x1*cosd(tetha1))/sind(tetha1);
     y2 = (rho(2)-x2*cosd(tetha2))/sind(tetha2);

     plot(x1(1,:), y1(1,:), 'm','LineWidth',1);
     plot(x2(1,:), y2(1,:), 'm','LineWidth',1);
     plot(intersection(1),intersection(2),'-o')
end
```