EE 417
POSTLAB
REPORT #5

NAME: Nidanur GÜNAY

ID: 24231

About a half hour ago my laptop has been broken and due to that i had to bring my friends laptop and i needed to install computer vision toolbox. Unfortunately it took about 40 minute. As a result, ı had a very limited time for doing the lab tasks but i achieved to do given task. Due to the time problem, i could not only achievedistance part during the lab. However i added some code and i finally finished my all task. This report firsty explains what i've done during the lab, secondly what are the resulted image of my 3D calibration object and finally i explain which one is better algorithm and why. Since my laptop has not repaired yet, i am writing my report on my friends. Since, you may see her name on top of the screenshot figures.

- Corner detection is useful for tracking, 3D reconstruction, Stereo vision and Camera Calibratin which is the purpose of this lab.

Things to do:

Write a program ("lab5calibprep.m") to detect corner points with **two** different methods:

1. Harris corners (integer values)
2. Intersection point of two lines (sub-pixel accuracy)

Implement the following steps:

- Read the image of calibration cube given in SUCourse (Figure 1**(a)**) and convert it to a black-white edge image with your edge detector choice.

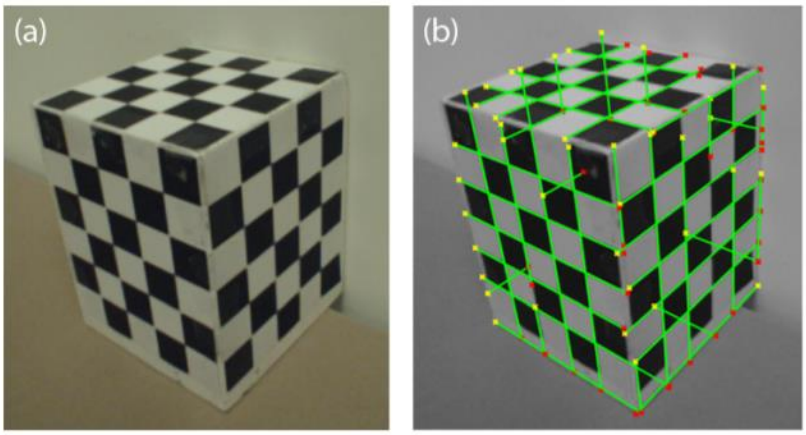

Figure 1: **(a)** Calibration Object **(b)** Result of Hough line detection

- Find the lines in the edge image by utilizing `'hough'`, `'houghpeaks'` and `'houghlines'` functions with appropriate parameters.
- Plot the beginnings of the lines with yellow cross, ends with red cross and the in-between line with green color on the gray-scale version of the original image (Figure 1**(b)**).

Figure1.1

- As a first task we needed to implement harriscorners. Thanks to built in function called `detectHarrisFeatures`, we found the Harris corners and i store them an array called H. Then i plot them into my main image. The result is shown in figure 1.2 and you can see my code in matlab code screenshots (figure 1.3).

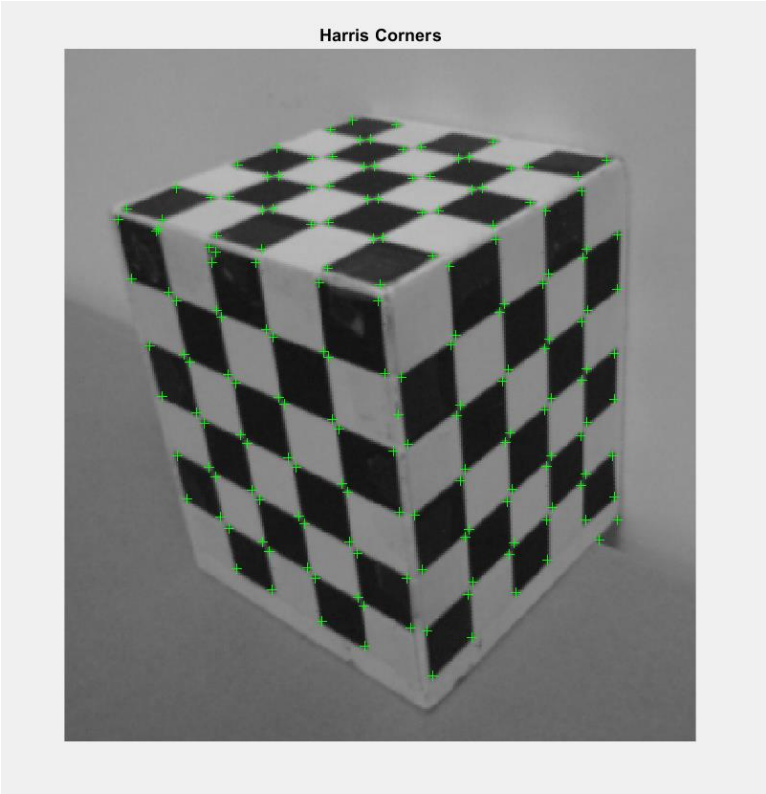

Figure 1.2

- As a second task we need to find corners by manually and to do that we found the hough lines we try to detect intersection of two houghlines which would be our corner. We detect the lines with the help of the hough, houghlines and houghpeaks built in functions. Before use the built in functions, we need to detect edges and i used Canny edge detector by edge built in function in MatLab. I took the codes from my previous lab code lab4houghlines and my output is on Figure 1.4.

```
Editor - C:\Users\IPEK GERDAN\Desktop\lab5\lab5calibprep.m
lab5calibprep.m
1  - img=imread("calibrationObject.png");
2  - img=rgb2gray(img);
3
4  - H=detectHarrisFeatures(img);
5
6    %%%%%%HARRIS CORNER DETECTION%%%%%%%%%%%%%
7  - figure
8  - imshow(img);hold on
9  - plot(H)
10 - title("Harris Corners");
11
12   %%%%%%%%%%%%INTERSECTION OF TWO LINES %%%%%%%%%%
13 -     figure;
14
15 -     BW = edge(img, 'Canny');
16 -     [H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89);
17
18 -     P  = houghpeaks(H, 20, 'Threshold',0.5*max(H(:)));
19 -     lines = houghlines(BW,T,R,P,'FillGap',10,'MinLength',40);
20
21
22 -     imshow(img), hold on
23 -     for k = 1:length(lines)
24 -        xy = [lines(k).point1; lines(k).point2];
25 -        plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
26
27 -        plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
28 -        plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
29 -     end
30
```

Fig 1.3

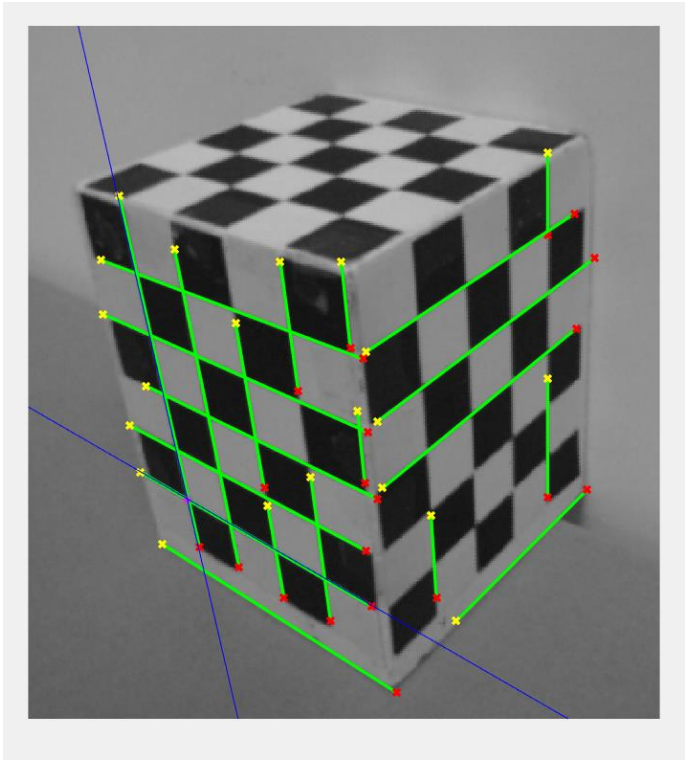

Fig 1.4

- I set the color of beginning point of the line as yellow and end of the line as red in the for loop in my code as in Fig 1.3.My houghlines output can be seen in Fig 1.4.

- Look at the output of `'houghlines'` function which returns ρ, θ and the beginning and end points of the detected lines. Select two intersecting lines manually from the plot that you obtained in previous step and extract the corresponding ρ and θ values from the output of `'houghlines'` function.
- Find the equations of these two lines by using the line equation given below and plot the lines with magenta color on the same figure:

$$x\,cos(\theta) + y\,sin(\theta) = \rho \tag{1}$$

- Solve these two line equations to find the intersection point with sub-pixel accuracy and plot that point on the same figure.
- Using the gray-scale version of the original image, find the Harris corners and plot them on the same figure with blue circles (Figure 2). Calculate and display the distance between two corner points that you obtained with two different methods.

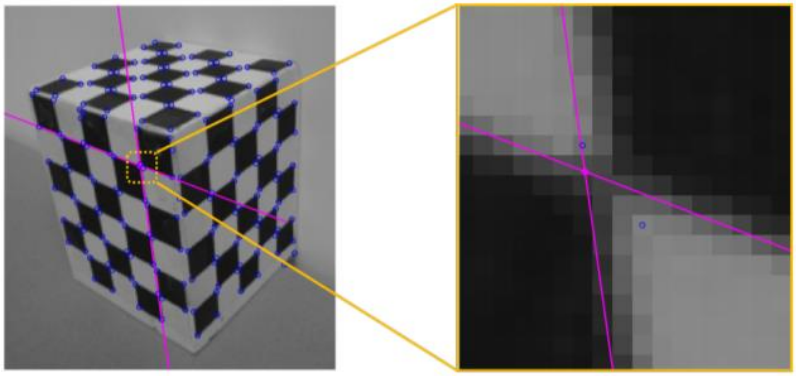

Figure 2: Corners extracted by the intersection point of two lines and Harris corner detection algorithm

Fig 1.5

- After i detect the houghlines as in the output figure 1.4, i picked 2 intersection lines manually and store their initial and end point as l1p1, l1p2, l2p1, l2p2 as you can see the my code in Fig 1.6. I cretated a for loop that iterates lines array's length times. If the lines are equal to lines that have been picked by me, i stored the arrays indexes in n1 and n2. Then i cretaed b vector by the line1's and line2's rho values and i created a matrix by cos and sin of their theta angle. In order to find tehir intersection point, we need to inverse of A and multiply it by b array ad as aresult i foind the coordinates of my intersection point.

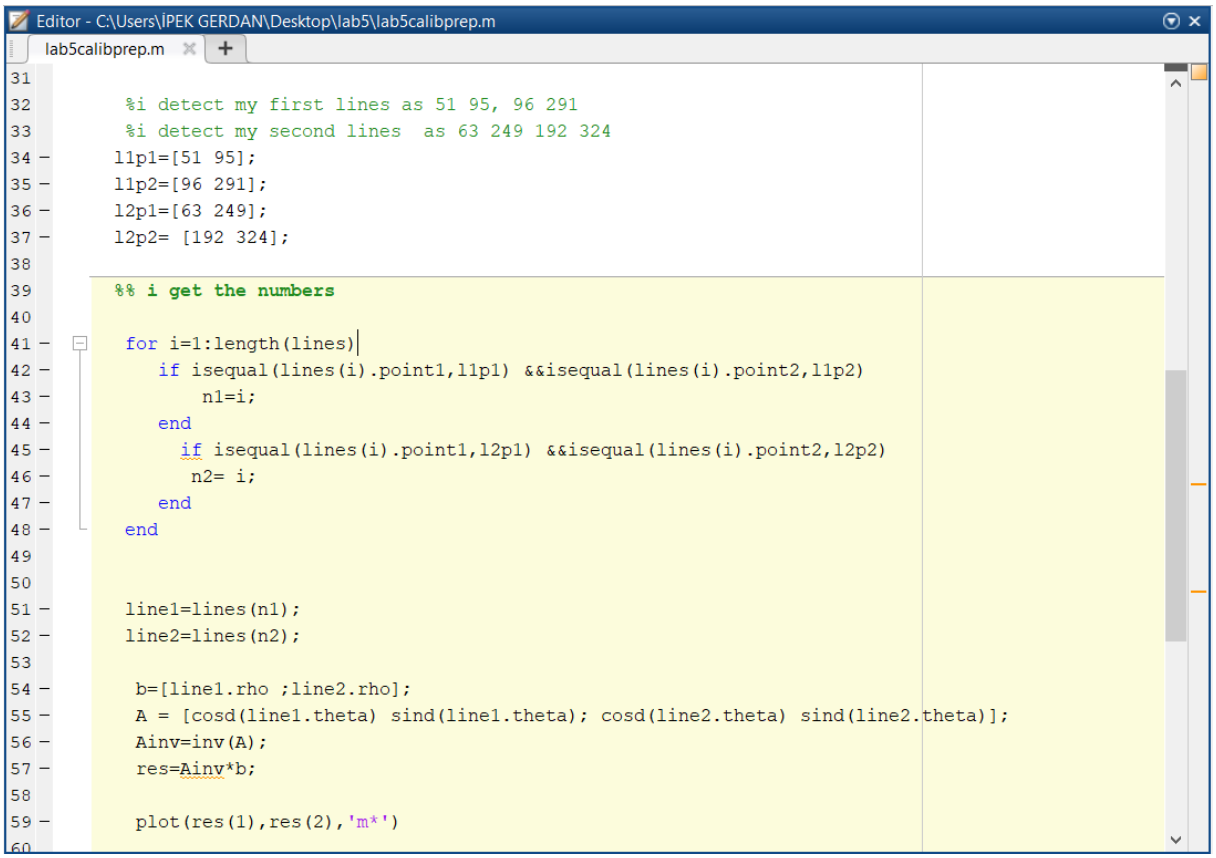

Fig 1.6

- After detecting the intersection point, i plotted point to my figure. After that i constructed my lines and i calculated to destance beteen the intersection point and Haris corner detected corner points in Fig 1.7.

Editor - C:\Users\İPEK GERDAN\Desktop\lab5\Untitled.m

lab5calibprep.m | Untitled.m

```
60
61 -        x=0:size(img,1);
62 -        y1=(line1.rho-x*cosd(line1.theta))/ sind(line1.theta);
63 -        y2=(line2.rho-x*cosd(line2.theta))/ sind(line2.theta);
64
65
66 -        plot(x,y1,"b");
67 -        plot(x,y2,"b");
68
69 -    figure
70 -    imshow(img);hold on
71 -    H=detectHarrisFeatures(img);
72 -    plot(x,y1,"b");
73 -    plot(x,y2,"b");
74 -    plot(H);
75 -    title("Harris Corners and Intersection Lines");
76
77 -    interpoint=[89.0 263.8];
78 -    harrisp1=[87.22 261.3];
79 -    harrisp2=[92.01 267.8];
80 -    dif1x=interpoint(1)- harrisp1(1);
```

Command Window

```
dist1 =

    3.0689


dist2 =

    5.0060
```

Fig 1.7

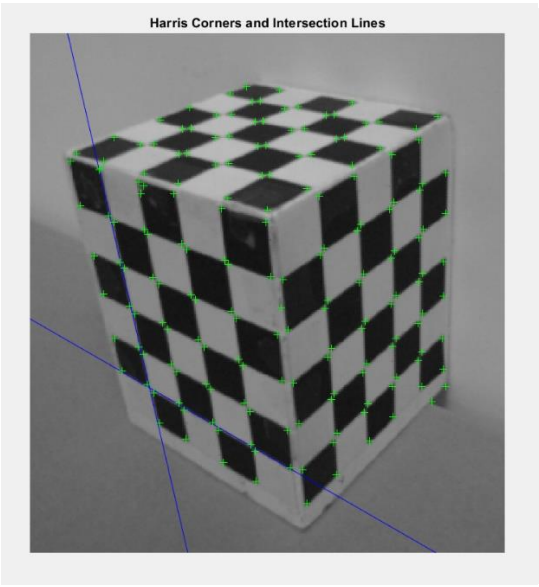

Fig 1.8

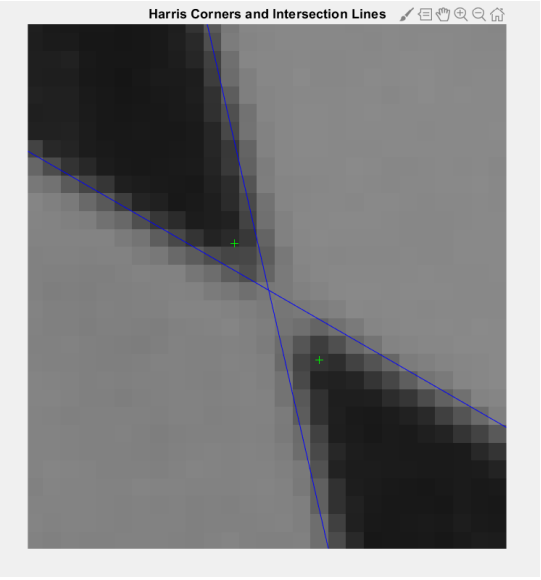

Fig 1.9

- I’ve executed the same code for the image that has been taken by me. Although, Harris Corner detection’ve worked properly, Houghlines have not give us the proper solutions. Result of Harris corner detected objet is shown in figure 2.1 and detected hough lines image and manually chosen lines intersection point is on figure 2.2. Harris Corner detected my objects corner properly and put the x signs in the accurate locations. However, since my objects line’s are smooth and even finding intersection lines were very difficult, detecting corners manaully from hough lines were unsuccesfull.

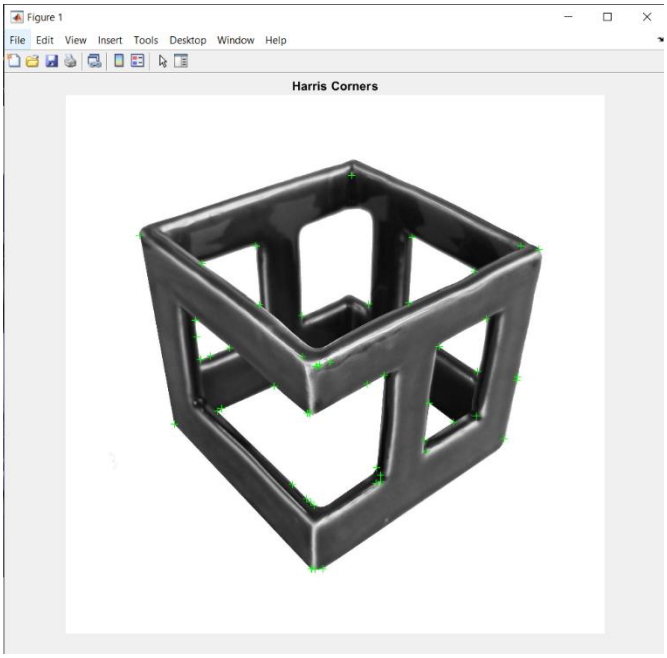

Fig 2.1

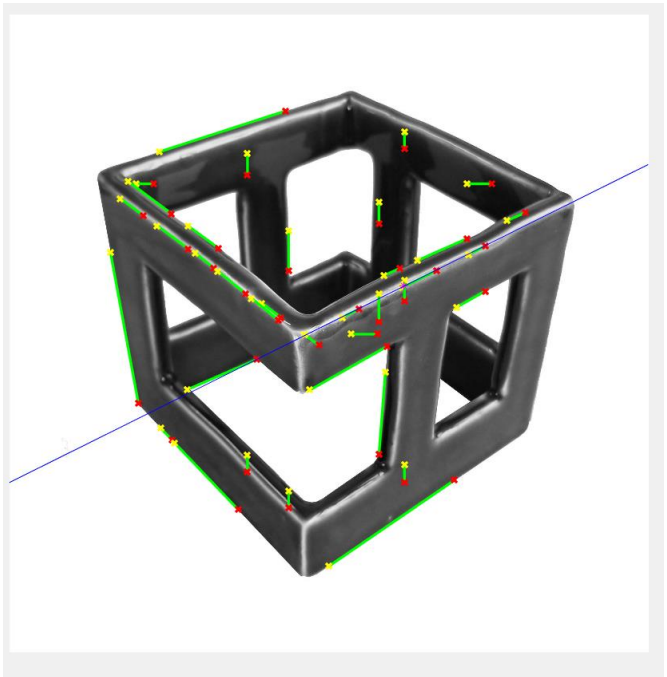

Fig 2.2

- Since our 3D object was unsucsessful, i've searched another 3D object image from the internet and i found that pyramid 3D image. You can see the Harris corner detected image in figure 2.3 and hough line's detected image in 2.4. I also picked 2 intersection lines manually and found their intersection point. I also find 8 different intersection points as shownin figure 2.7 and i calculated distance between harris corners and one of that points.

Discussion: As a result of 3 corner detected image, i can say that harris corner detection gives more solution than the subpixel-accuracy method. For instance, in figure 1.9 , it is the secreenshot of the 1 corner and harris displays 2 points as a corner. Main reason of that is Harris Corner Detection technique is based on subpixel property. However, Sub-pixel accuracytechniqu displays 1 solution for the corner and its localization is more accurate rather than the Harris Corner Detection.So i would prefer Sub-pixel accuracy technique for camera calibration.

In despite of the result in image that is used in lab, in the image that i've taken by myself, there is a few intersection points. Houghlines were unseccesful to detect lines. The main reason of that is my 3D object's edges are smooth and there is luminance on the edges due to light. So for 3D object Harris is more accurate and i ould prefer Harris corner detection in images like in instance.

In the image which is taken from the from the internet Harris corner points are more accurately localized, hough lines are shifted a little bit as seen in figure 2.6, 2.7 and figure 2.8. In the pyramid image i would prefer Sub-pixel accurecy tecnique. Althgouh harris points are more localized, it displayed more points which are not corners.

If i needed to do camera calibration, i would make hybrid of both technique. Firstly i would display the Harris corners then i would have picked the harris corners which were closer to the points that have been calculated by sub-pixel accuracy technique.

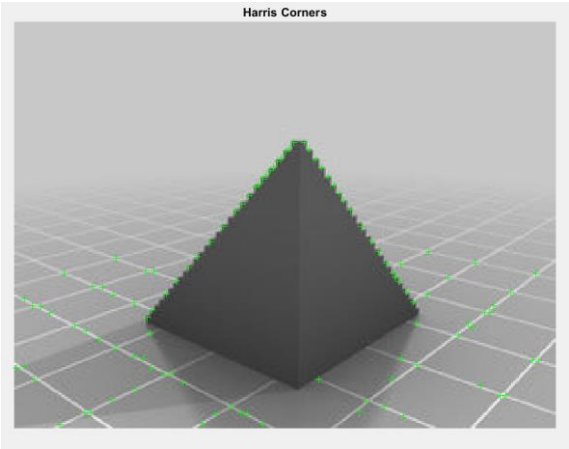

Fig 2.3

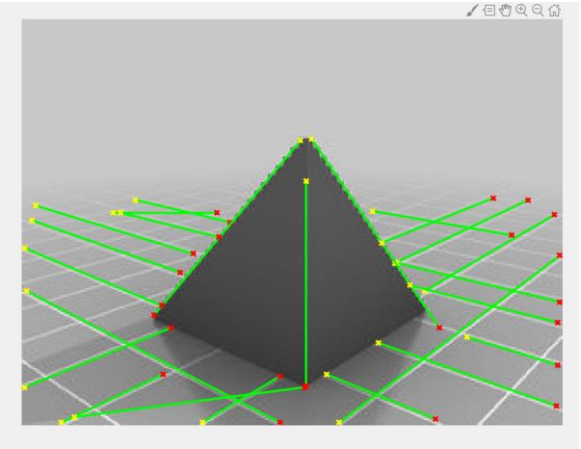

Fig 2.4

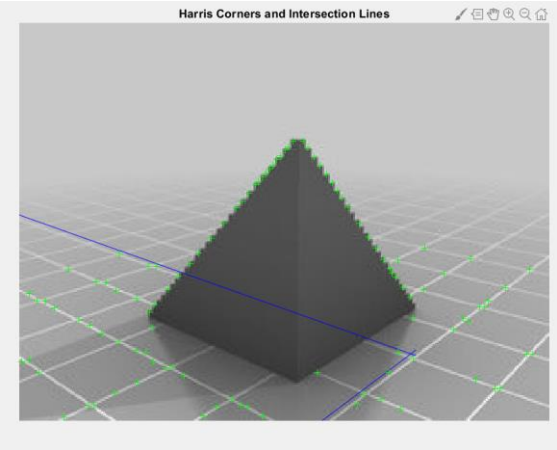

Fig 2.5

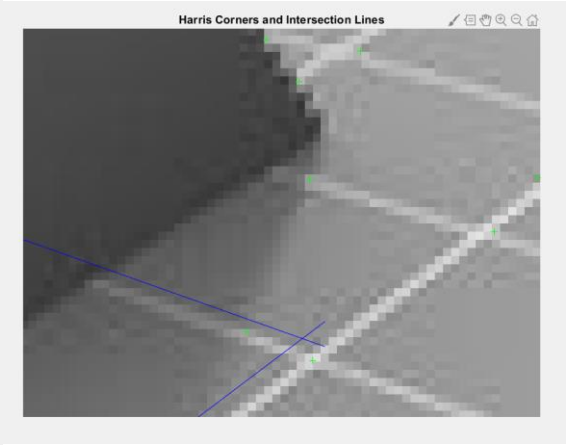

Fig 2.6

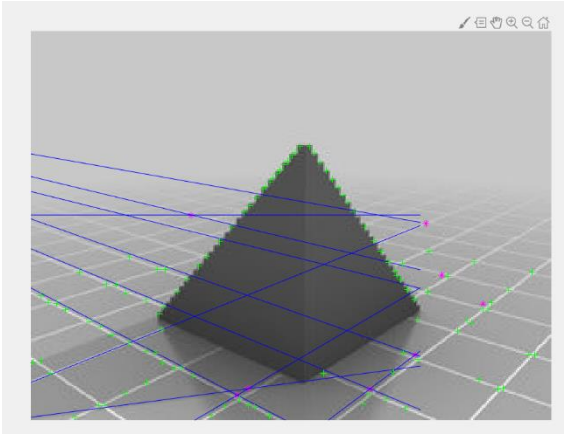

Fig 2.7

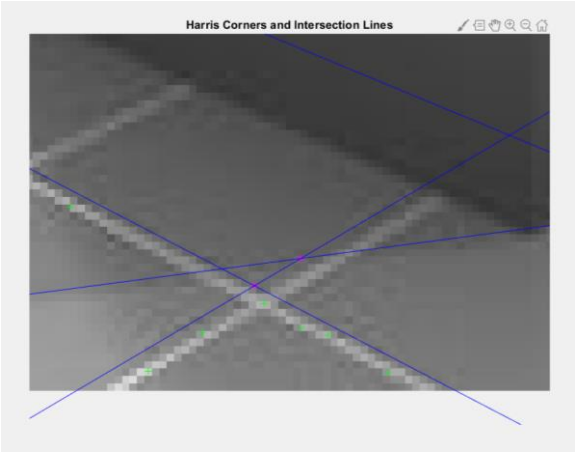

Fig 2.8

APPENDIX: *LAB5CALIBPREP*

```
img=imread("calibrationObject.png");
img=rgb2gray(img);

H=detectHarrisFeatures(img);

%%%%%%HARRIS CORNER DETECTION%%%%%%%%%%%%
figure
imshow(img);hold on
plot(H)
title("Harris Corners");

%%%%%%%%%%%%INTERSECTION OF TWO LINES %%%%%%%%%%
    figure;

    BW = edge(img, 'Canny');
    [H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89);

    P  = houghpeaks(H, 20, 'Threshold',0.5*max(H(:)));
    lines = houghlines(BW,T,R,P,'FillGap',10,'MinLength',40);


    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');
    end


   %i detect my first lines as 51 95, 96 291
   %i detect my second lines  as 63 249 192 324
  l1p1=[51 95];
  l1p2=[96 291];
  l2p1=[63 249];
  l2p2= [192 324];

  %% i get the numbers

   for i=1:length(lines)
      if isequal(lines(i).point1,l1p1) &&isequal(lines(i).point2,l1p2)
          n1=i;
      end
        if isequal(lines(i).point1,l2p1) &&isequal(lines(i).point2,l2p2)
         n2= i;
      end
   end


   line1=lines(n1);
   line2=lines(n2);

    b=[line1.rho ;line2.rho];
    A = [cosd(line1.theta) sind(line1.theta); cosd(line2.theta) sind(line2.theta)];
    Ainv=inv(A);
    res=Ainv*b;

    plot(res(1),res(2),'m*')

    x=0:size(img,1);
    y1=(line1.rho-x*cosd(line1.theta))/ sind(line1.theta);
    y2=(line2.rho-x*cosd(line2.theta))/ sind(line2.theta);


    plot(x,y1,"b");
    plot(x,y2,"b");

figure
imshow(img);hold on
H=detectHarrisFeatures(img);
plot(x,y1,"b");
plot(x,y2,"b");
plot(H);
```

```
title("Harris Corners and Intersection Lines");

interpoint=[89.0 263.8];
harrisp1=[87.22 261.3];
harrisp2=[92.01 267.8];
dif1x=interpoint(1)- harrisp1(1);
dif1y=interpoint(2)-harrisp1(2);
dif2x=harrisp2(1)-interpoint(1);
dif2y=harrisp2(2)-interpoint(2);
dist1=sqrt(dif1x^2+dif1y^2);
dist2=sqrt(dif2x^2+dif2y^2);
dist1
dist2
```

LAB5CALIBPREP/(TO DETECTING 8 CORNER)

```
l1p1=[193 175]; %984 685
  l1p2=[289 209];%1070 642
  l2p1=[172 218];%989 667
  l2p2= [291 128];%989 719

  l3p1=[29 215]
  l3p2=[153 199]
  l4p1=[98 218]
  l4p2=[140 193]

  l5p1=[3 147]
  l5p2=[140 218]
  l6p1=[98 218]
  l6p2=[140 193]

  l7p1=[172 218]
  l7p2=[291 128]
  l8p1=[165 192]
  l8p2=[224 213]

  l9p1=[172 218]
  l9p2=[291 128]
  l10p1=[210 144]
  l10p2=[290 164]

  l11p1=[195 121]
  l11p2=[255 97]
  l12p1=[190 104]
  l12p2=[265 117]

  l13p1=[203 132]
  l13p2=[291 153]
  l14p1=[218 147]
  l14p2=[288 106]

  l15p1=[50 105]
  l15p2=[106 105]
  l16p1=[62 98]
  l16p2=[113 110]



  % i get the numbers

   for i=1:length(lines)
      if isequal(lines(i).point1,l1p1) &&isequal(lines(i).point2,l1p2)
          n1=i;
      end
      if isequal(lines(i).point1,l2p1) &&isequal(lines(i).point2,l2p2)
         n2= i;
      end
      if isequal(lines(i).point1,l3p1) &&isequal(lines(i).point2,l3p2)
         n3= i;
      end
      if isequal(lines(i).point1,l4p1) &&isequal(lines(i).point2,l4p2)
         n4= i;
      end
      if  isequal(lines(i).point1,l5p1) &&isequal(lines(i).point2,l5p2)
```

```
        n5= i;
      end
      if  isequal(lines(i).point1,l6p1) &&isequal(lines(i).point2,l6p2)
        n6= i;
      end
      if isequal(lines(i).point1,l7p1) &&isequal(lines(i).point2,l7p2)
        n7= i;
      end
      if isequal(lines(i).point1,l8p1) &&isequal(lines(i).point2,l8p2)
        n8= i;
      end
      if isequal(lines(i).point1,l9p1) &&isequal(lines(i).point2,l9p2)
        n9= i;
      end
      if isequal(lines(i).point1,l10p1) &&isequal(lines(i).point2,l10p2)
        n10= i;
      end
      if isequal(lines(i).point1,l11p1) &&isequal(lines(i).point2,l11p2)
        n11= i;
      end
      if isequal(lines(i).point1,l12p1) &&isequal(lines(i).point2,l12p2)
        n12= i;
      end
      if isequal(lines(i).point1,l13p1) &&isequal(lines(i).point2,l13p2)
        n13= i;
      end
      if isequal(lines(i).point1,l14p1) &&isequal(lines(i).point2,l14p2)
        n14= i;
      end
      if isequal(lines(i).point1,l15p1) &&isequal(lines(i).point2,l15p2)
        n15= i;
      end
      if isequal(lines(i).point1,l16p1) &&isequal(lines(i).point2,l16p2)
        n16= i;
      end
   end


   line1=[lines(n1), lines(n3), lines(n5), lines(n7), lines(n9), lines(n11), lines(n13),
lines(n15)];
   line2=[lines(n2), lines(n4),lines(n6), lines(n8), lines(n10), lines(n12),lines(n14),
lines(n16)];
   figure
    imshow(img);hold on
H=detectHarrisFeatures(img);
   for i=1:8
    b=[line1(i).rho ;line2(i).rho];
    A = [cosd(line1(i).theta) sind(line1(i).theta); cosd(line2(i).theta)
sind(line2(i).theta)];
    Ainv=inv(A);
    res=Ainv*b;

    plot(res(1),res(2),'m*')

    x=0:size(img,1);
    y1=(line1(i).rho-x*cosd(line1(i).theta))/ sind(line1(i).theta);
    y2=(line2(i).rho-x*cosd(line2(i).theta))/ sind(line2(i).theta);



    plot(x,y1,"b");
    plot(x,y2,"b");
    plot(H);
   end
title("Harris Corners and Intersection Lines");

interpoint=[216 182.8];
harrisp1=[209.3 181.9];
harrisp2=[217.5  185.4];
dif1x=interpoint(1)- harrisp1(1);
dif1y=interpoint(2)-harrisp1(2);
dif2x=harrisp2(1)-interpoint(1);
dif2y=harrisp2(2)-interpoint(2);
dist1=sqrt(dif1x^2+dif1y^2);
dist2=sqrt(dif2x^2+dif2y^2);
dist1
```

dist2