DATA GENERATION FOR CAMERA CALIBRATION

NOVEMBER 26, 2021

TA's: Muhammed Zemzemoğlu and Mehmet Emin Mumcuoğlu

Name of Student: Moses Chuka Ebere
Student Number: 31491
Course: Computer Vision (EE 417) Lab

TABLE OF CONTENTS

1. Introduction

2. Explanation of Methods

3. Results

4. Multiple Results

5. Post-lab Questions

6. Discussion

7. References

8. Appendix

INTRODUCTION

A camera, in the most basic terms, is a sensor that takes images from a 3D scene. There are myriads of camera types available today; however, they all rely on the basic principle. Mathematically, a camera could be modeled as a projective camera to get an idea of the processes involved in image formation. In modern cameras, contrary to the pinhole applied in projective cameras, lenses are used to focus images on the image plane. This brings us to the notion of perspective cameras.

Today's lab focuses on the fundamental approach to calibrating a camera. Camera calibration entails estimating the intrinsic and extrinsic parameters of a camera. Intrinsic or internal parameters include the effective focal lengths, α and β, which are scaled versions of the camera's focal length, the coordinates of the principal point u_o and v_o, the skew parameter, s, and the radial distortion κ_1.

EXPLANATION OF METHODS

Line Detection Using Hough Transform

Firstly, a calibration pattern or rig is created. This could be made from a checkerboard image. In the most ideal case, 3 patterns aligned with arbitrary x-, y-, and z- axes are used. However, 2 planes are enough to form the calibration rig. A photo of the above pattern is taken and used as the input photo for the calibration process.

Whilst data generation is the focal point of this lab, a key task is to compare two methods of corner extraction – Harris corners and sub-pixel accuracy using Hough Transform for lines. Edge images are used in both cases.

For the line detection using Hough Transform, the underlying is as follows:

$$\rho = x\cos\theta + y\sin\theta \quad \text{-------} \ (1)$$

A point in the image space translates to a sinusoid in the parameter/accumulator space.

ρ is the length of the normal from a line to the origin.

θ is the orientation of the line.

The Hough Transform for line detection assumes a parameter space where the rho and theta are fixed and a point in the image space represents a sinusoid in the parameter space.

The function looks at collinear points in the image space and finds the number of times their corresponding sinusoids in the parameter space intersect. With the help of a threshold, any cumulative intersection that exceeds the threshold qualifies to be a line.

Once the lines are detected, with the aid of the naked eyes, intersecting lines are identified. These lines are highlighted alongside

their points of intersection (by solving the two line-equations resulting from equation (1) using matrices and matrix inverse).

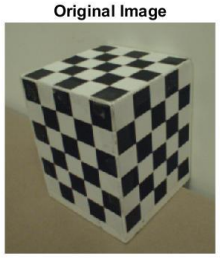
For the sake of comparison, corners are also extracted using Harris detector. The aim of this is to confirm that the sub-pixel accuracy offered by Hough Transform performs significantly better at identifying the exact location of corners.

In MATLAB, a script was written to perform the above tasks. This can be found in the appendix.

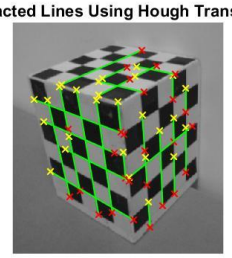
IN-LAB RESULTS

Line Detection

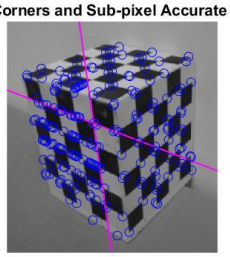
Original Image



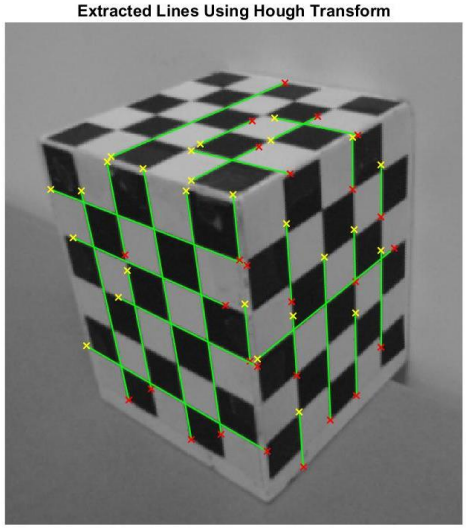
Extracted Lines Using Hough Transform



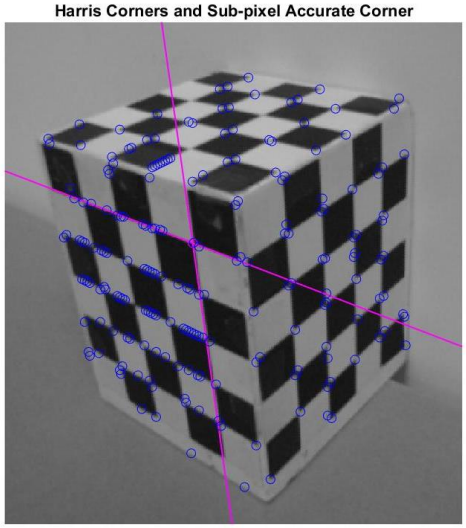
Harris Corners and Sub-pixel Accurate Corner



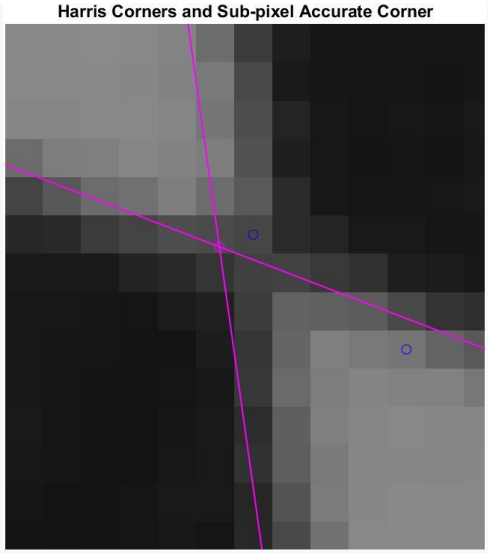
Extracted Lines Using Hough Transform



Harris Corners and Sub-pixel Accurate Corner



Harris Corners and Sub-pixel Accurate Corner



The coordinates of the lines are:

Line 1: (x,y) = (36,129) and (187,187)
Line 2: (x,y) = (140,130) and (167,317)

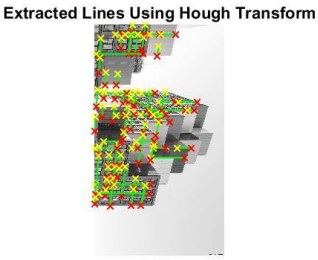
Harris corner
Chosen point: (146,170)

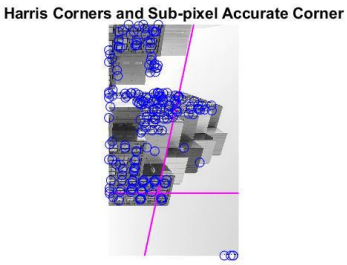
Real corner (by observation): (145.5, 170.5).

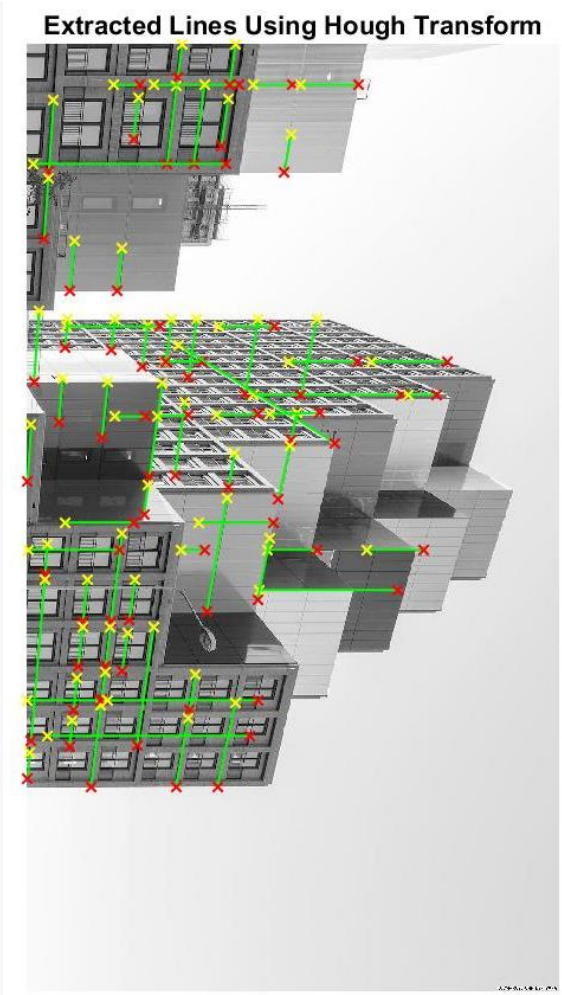
The distance between the Hough corner and the real corner is 0.4254.
The distance between the Harris corner and the real corner is 0.7071.

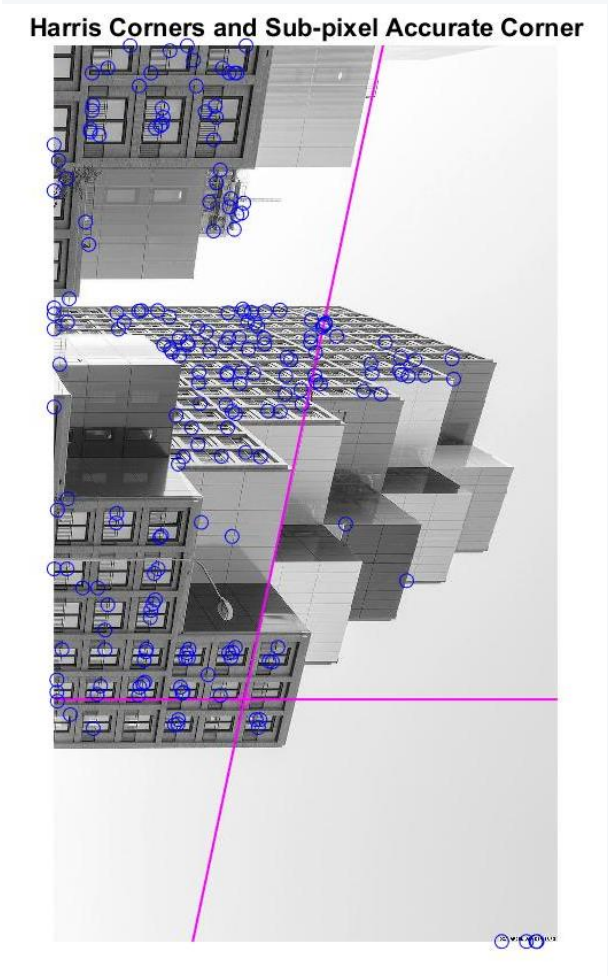
MULTIPLE RESULTS

Using Hough Transform for line detection to obtain sub-pixel accurate corners in a random image;





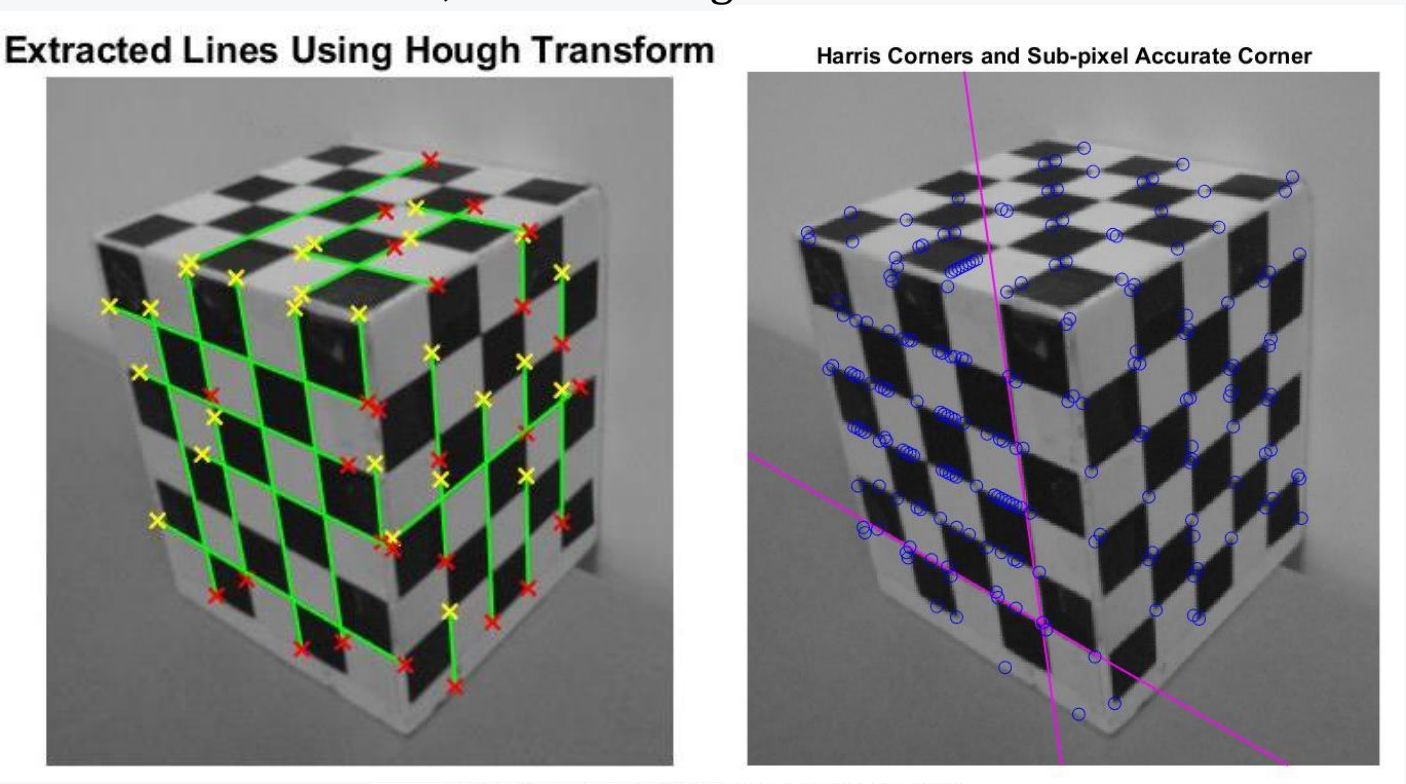


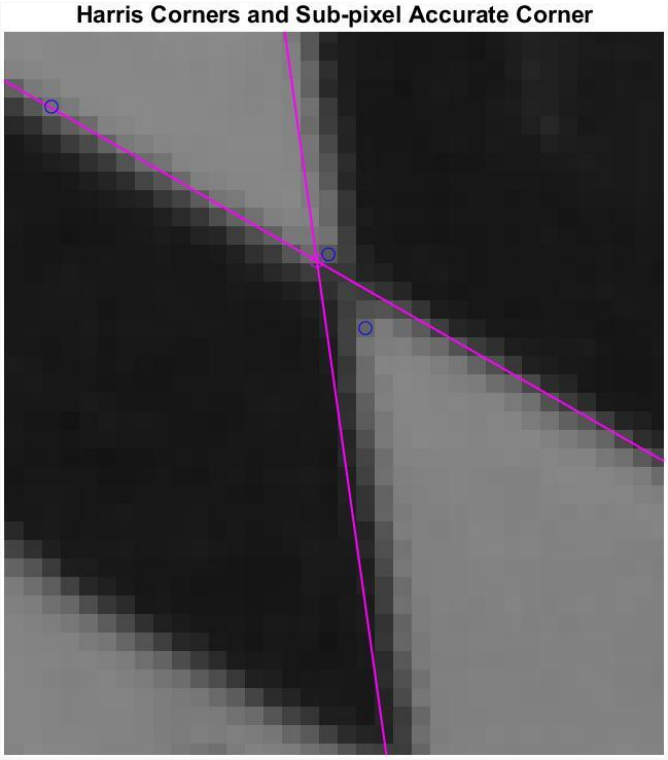


The coordinates of the lines are:

Line 1: (x,y) = (353,1113) and (323,1255)
Line 2: (x,y) = (35,1169) and (378,1169)

Going back to our original calibration image and changing the edge detector to ‘zerocross’, the following new corner was extracted:





The coordinates of the lines are:
Line 1: (x,y) = (140,130) and (167,317)
Line 2: (x,y) = (63,249) and (202,330)

Harris corner
Chosen point: (165,307)

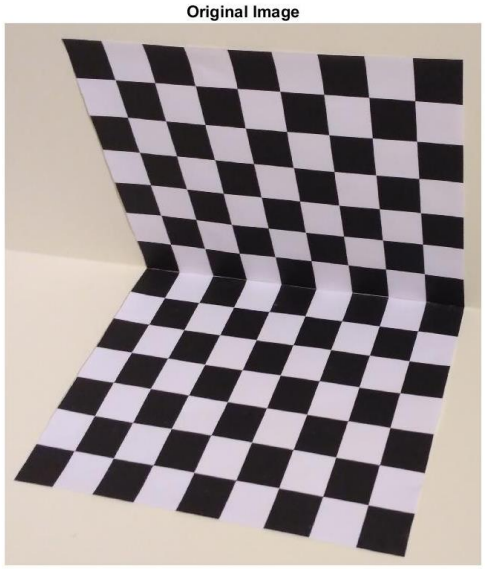
Real corner (by observation): (165.5, 309.5).

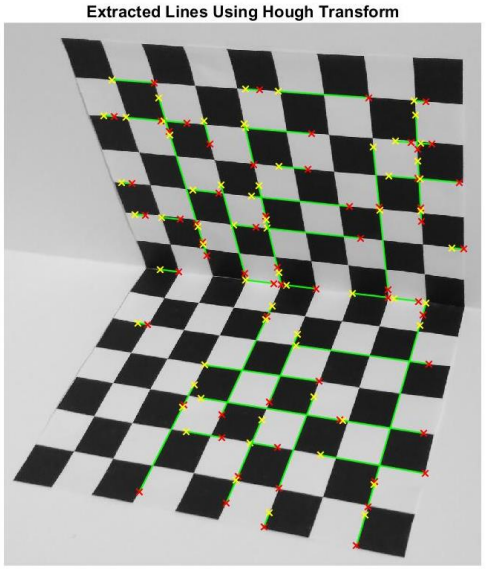
The distance between the Hough corner and the real corner is 2.4096.
The distance between the Harris corner and the real corner is 2.5495.

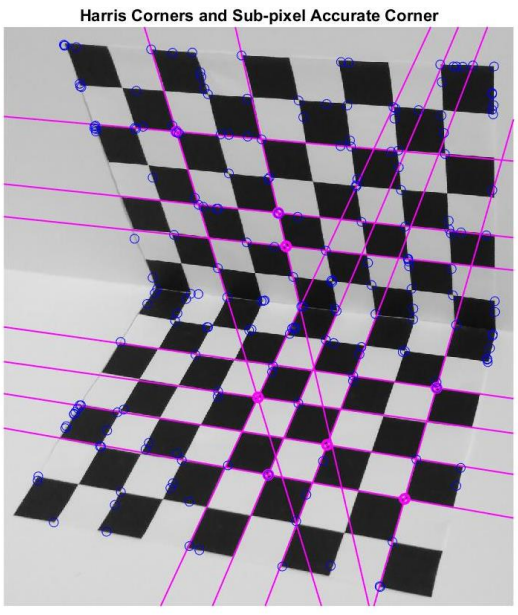
POST-LAB QUESTIONS

For the post-lab, the following calibration rig was created:

To extract the corners, the Laplacian of gaussian edge detector was applied first:







Eight corners were extracted using the Hough Transform. The equations of the 16 lines are as follows:

$$x_1.\cos(-1.4137) + y_1.\sin(-1.4137) = -1486$$
$$x_2.\cos(0.3665) + y_2.\sin(0.3665) = 1853$$
$$x_3.\cos(-1.43117) + y_3.\sin(-1.43117) = -1360$$
$$x_4.\cos(0.4363) + y_4.\sin(0.4363) = 1.588e + 03$$
$$x_5.\cos(-1.396) + y_5.\sin(-1.396) = -1621$$
$$x_6.\cos(0.27925) + y_6.\sin(0.27925) = 2115$$
$$x_7.\cos(-1.4661) + y_7.\sin(-1.4661) = -641$$
$$x_8.\cos(-0.2269) + y_8.\sin(-0.2269) = 926$$
$$x_9.\cos(-1.48353) + y_9.\sin(-1.48353) = -366$$
$$x_{10}.\cos(-0.2967) + y_{10}.\sin(-0.2967) = 552$$
$$x_{11}.\cos(-1.4312) + y_{11}.\sin(-1.4312) = -1219$$
$$x_{12}.\cos(0.279) + y_{12}.\sin(0.279) = 2115$$
$$x_{13}.\cos(-1.396) + y_{13}.\sin(-1.396) = -1621$$
$$x_{14}.\cos(0.4014) + y_{14}.\sin(0.4014) = 1.717e + 03$$
$$x_{15}.\cos(-1.4661) + y_{15}.\sin(-1.4661) = -774$$
$$x_{16}.\cos(-0.2269) + y_{16}.\sin(-0.2269) = 926$$

The coordinates of the end points of the above lines are:
Line 1: (x,y) = (863,1642), (1472,1739)
Line 2: (x,y) = (1357,1638), (1204,2037)
Line 3: (x,y) = (1386,1891), (1845,1972)
Line 4: (x,y) = (1822,1325), (1626,2008)
Line 5: (x,y) = (880,1498), (1380,1568)
Line 6: (x,y) = (1147,1300), (954,1714)
Line 7: (x,y) = (1086,760), (1513,804)
Line 8: (x,y) = (1116,712), (1146,846)
Line 9: (x,y) = (700,430), (803,439)
Line 10: (x,y) = (679,329), (723,476)
Line 11: (x,y) = (1279,1412), (1861,1493)
Line 12: (x,y) = (1822,1325), (1626,2008)

Line 13: (x,y) = (1080,1837), (1200,1858)
Line 14: (x,y) = (1129,1739), (1025,1984)
Line 15: (x,y) = (1153,900), (1563,943)
Line 16: (x,y) = (1150,863), (1199,1072)

- The intersection of lines 13 and 14 was compared with the closest Harris corner in terms of their distance to the real corner:

Harris corner
Chosen point: (1086,1838)

Real corner (by observation): (1085.5, 1837.5).

The distance between the Hough corner and the real corner is 0.1836.
The distance between the Harris corner and the real corner is 0.7071.

Discussion
The discussion can be found in the following section.

DISCUSSION

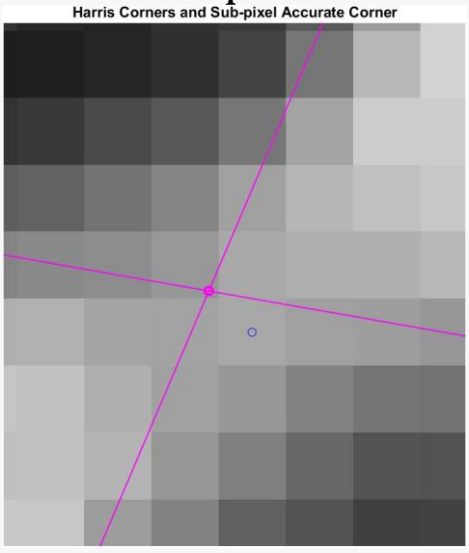
General Comments on Hough Transform

- The main idea is that we map a difficult pattern problem into a simple peak detection problem.
- Hough Transform is a really robust algorithm as it could be used to detect any curve that can be expressed in parametric form.

$$Y = f(x, a1, a2, \ldots ap)$$

a1, a2, … ap are the parameters

- I noticed that the number of lines extracted is really dependent on the edge detection algorithm used and also on the orientation of the image.
- In some cases, editing the photo and changing its orientation changed my results significantly.
- Corner extraction using Hough lines is significantly more accurate than using Harris corners. This can be credited to the fact that while Harris corners is restricted to integer corners, intersection of Hough lines allows for sub-pixel accuracy.



- As seen in the above picture, the real corner falls on the border (to the right) of the within which the Hough intersection corner lies, while the Harris corner is at half a pixel diagonal distance away.

- **Hough transform is robust to undesirable features like noise, occlusion, etc.**

Which Corner Extraction Method Would I Prefer for Camera Calibration?

- I would opt for the corners extracted using the intersection of Hough lines due to the sub-pixel accuracy it affords us.

Better Methods

- As found in the literature, we can apply certain methods to already existing corner detectors (like Harris, Kanade-Tomasi, etc.) to achieve sub-pixel accuracy.
- **These methods can be classified into interpolation method, classification method, and fitting method.**
- *The fitting method is widely used in the field of digital image processing. There are various kinds of surface fitting, such as Gaussian surface fitting, polynomial fitting method, ellipse fitting method. Gaussian surface fitting algorithm is particularly common in achieving positioning precision at level of the sub-pixel.*

REFERENCES

Reinhard Klette. 2014. Concise Computer Vision: An Introduction into Theory and Algorithms. Springer Publishing Company, Incorporated.

https://www.bigrentz.com/blog/how-to-start-construction-company

https://edition.cnn.com/style/article/anticipated-architecture-2021/index.html

https://www.designtrends.com/graphic-web/patterns/circle-pattern.html

https://www.dreamstime.com/seamless-abstract-pattern-black-overlapping-circles-dots-bubbles-bulbs-background-vector-illustration-calm-classic-design-image185529825

https://download.atlantis-press.com/article/25866.pdf

APPENDIX

CAMERA CALIBRATION

A single m-file was created for this lab:

In-Lab Code

```
clear all; close all; clc;

% Read the image to be preprocessed
im = imread('calibrationObject.png');

tic
% The row, column, and channels of the image are obtained
along with the cardinality of the image.
[r, c, ch] = size(im);
Card = r*c;

img = im;

% This is added in case the image introduced is an RGB
image.
% It functions to convert it to a gray-scale image.
if ch == 3
    img = rgb2gray(img);
end

% Create a Black-white Edge Image
I = edge(img, 'log');

% Find the space matrix, theta, and rho parameters.
[H,T,R] = hough(I);

% Threshold
t = 0.5*max(H(:));

% Find the peaks
P = houghpeaks(H, 20, 'threshold', t);

% Extract the line segments based on Hough Transform
lines = houghlines(I,T,R,P,'FillGap',10,'MinLength',40);

figure %1
```

```
subplot(2,2,1)
imshow(im)
title('Original Image')

subplot(2,2,2)
imshow(img), hold on
title('Extracted Lines Using Hough Transform')

for k = 1:length(lines)
    % Obtain the endpoints of each line found earlier.
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',1,'Color','green');

   % Plot beginnings and ends of lines

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

   % r = lines(k).theta
   %a = xy(1), b = xy(2), c = xy(3), d = xy(4)

   if xy(1) == 140 && xy(3) == 130 && xy(2) == 167 && xy(4)
== 317
       k1 = k;
       rho1 = lines(k).rho;
       theta1 = lines(k).theta*pi/180;
   end

   if xy(1) == 36 && xy(3) == 129 && xy(2) == 187 && xy(4)
== 187
       k2 = k;
       rho2 = lines(k).rho;
       theta2 = lines(k).theta*pi/180;
   end

end

x1 = 0:1:386;
y1 = (rho1 - x1*cos(theta1))/sin(theta1);
x2 = 0:1:386;
y2 = (rho2 - x2*cos(theta2))/sin(theta2);

subplot(2,2, [3,4])
```

```
imshow(img), hold on
title('Harris Corners and Sub-pixel Accurate Corner')
plot(x1, y1,'LineWidth',1,'Color','magenta');
plot(x2, y2,'LineWidth',1,'Color','magenta');

% Find the intersection point of the two lines
intersection = [cos(theta1) sin(theta1); cos(theta2)
sin(theta2)] \ [rho1; rho2];
% Plot the intersection point
plot(intersection(1), intersection(2), 'mo')

% Find the Harris corners
C_H = corner(I, 'Harris');
plot(C_H(:,1),C_H(:,2),'bo');

% Find the distance between the two points
len = norm(intersection - [145.5; 170.5]);
len2 = norm([146; 170] - [145.5; 170.5]);

fprintf('The distance between the Hough corner and the real
corner is %.4f.\n', len)
fprintf('The distance between the Harris corner and the
real corner is %.4f.\n', len2)

toc
```

Post-Lab Code

```
clear all; close all; clc;

% Read the image to be preprocessed
im = imread('3D_Calibration_Object.jpg');

tic
% The row, column, and channels of the image are obtained
along with the cardinality of the image.
[r, c, ch] = size(im);
Card = r*c;

img = im;
```

```
% This is added in case the image introduced is an RGB
image.
% It functions to convert it to a gray-scale image.
if ch == 3
    img = rgb2gray(img);
end

% Create a Black-white Edge Image
I = edge(img, 'log');

% Find the space matrix, theta, and rho parameters.
[H,T,R] = hough(I);

% Threshold
t = 0.5*max(H(:));

% Find the peaks
P = houghpeaks(H, 20, 'threshold', t);

% Extract the line segments based on Hough Transform
lines = houghlines(I,T,R,P,'FillGap',10,'MinLength',40);

figure %1
subplot(1,2,1)
imshow(im)
title('Original Image')

subplot(1,2,2)
imshow(img), hold on
title('Extracted Lines Using Hough Transform')

for k = 1:length(lines)
    % Obtain the endpoints of each line found earlier.
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',1,'Color','green');

   % Plot beginnings and ends of lines

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

   % Obtain rho and theta for the two intersecting lines
   % First intersection
```

```
    if xy(1) == 863 && xy(3) == 1642 && xy(2) == 1472 &&
xy(4) == 1739
        k1 = k;
    elseif xy(1) == 1357 && xy(3) == 1638 && xy(2) == 1204
&& xy(4) == 2037
        k2 = k;
    end
    if xy(1) == 880 && xy(3) == 1498 && xy(2) == 1380 &&
xy(4) == 1568
        k3 = k;
    elseif xy(1) == 1147 && xy(3) == 1300 && xy(2) == 954 &&
xy(4) == 1714
        k4 = k;
    end
    if xy(1) == 1386 && xy(3) == 1891 && xy(2) == 1845 &&
xy(4) == 1972
        k5 = k;
    elseif xy(1) == 1822 && xy(3) == 1325 && xy(2) == 1626
&& xy(4) == 2008
        k6 = k;
    end
    if xy(1) == 1086 && xy(3) == 760 && xy(2) == 1513 &&
xy(4) == 804
        k7 = k;
    elseif xy(1) == 1116 && xy(3) == 712 && xy(2) == 1146 &&
xy(4) == 846
        k8 = k;
    end
    if xy(1) == 700 && xy(3) == 430 && xy(2) == 803 && xy(4)
== 439
        k9 = k;
    elseif xy(1) == 679 && xy(3) == 329 && xy(2) == 723 &&
xy(4) == 476
        k10 = k;
    end
    if xy(1) == 1279 && xy(3) == 1412 && xy(2) == 1861 &&
xy(4) == 1493
        k11 = k;
    elseif xy(1) == 1822 && xy(3) == 1325 && xy(2) == 1626
&& xy(4) == 2008
        k12 = k;
```

```
    end
    if xy(1) == 1080 && xy(3) == 1837 && xy(2) == 1200 &&
xy(4) == 1858
        k13 = k;
    elseif xy(1) == 1129 && xy(3) == 1739 && xy(2) == 1025
&& xy(4) == 1984
        k14 = k;
    end
    if xy(1) == 1153 && xy(3) == 900 && xy(2) == 1563 &&
xy(4) == 943
        k15 = k;
    elseif xy(1) == 1150 && xy(3) == 863 && xy(2) == 1199 &&
xy(4) == 1072
        k16 = k;
    end
end

% Parameters for the 16 Lines
rho1 = lines(k1).rho; theta1 = lines(k1).theta*pi/180; %
Line 1
rho2 = lines(k2).rho; theta2 = lines(k2).theta*pi/180; %
Line 2
rho3 = lines(k3).rho; theta3 = lines(k3).theta*pi/180; %
Line 3
rho4 = lines(k4).rho; theta4 = lines(k4).theta*pi/180; %
Line 4
rho5 = lines(k5).rho; theta5 = lines(k5).theta*pi/180; %
Line 5
rho6 = lines(k6).rho; theta6 = lines(k6).theta*pi/180; %
Line 6
rho7 = lines(k7).rho; theta7 = lines(k7).theta*pi/180; %
Line 7
rho8 = lines(k8).rho; theta8 = lines(k8).theta*pi/180; %
Line 8
rho9 = lines(k9).rho; theta9 = lines(k9).theta*pi/180; %
Line 9
rho10 = lines(k10).rho; theta10 = lines(k10).theta*pi/180;
% Line 10
rho11 = lines(k11).rho; theta11 = lines(k11).theta*pi/180;
% Line 11
```

```
rho12 = lines(k12).rho; theta12 = lines(k12).theta*pi/180;
% Line 12
rho13 = lines(k13).rho; theta13 = lines(k13).theta*pi/180;
% Line 13
rho14 = lines(k14).rho; theta14 = lines(k14).theta*pi/180;
% Line 14
rho15 = lines(k15).rho; theta15 = lines(k15).theta*pi/180;
% Line 15
rho16 = lines(k16).rho; theta16 = lines(k16).theta*pi/180;
% Line 16

% Define the equations of each pair of lines
x1 = 0:1:2505;
y1 = (rho1 - x1*cos(theta1))/sin(theta1); y2 = (rho2 -
x1*cos(theta2))/sin(theta2);
y3 = (rho3 - x1*cos(theta3))/sin(theta3); y4 = (rho4 -
x1*cos(theta4))/sin(theta4);
y5 = (rho5 - x1*cos(theta5))/sin(theta5); y6 = (rho6 -
x1*cos(theta6))/sin(theta6);
y7 = (rho7 - x1*cos(theta7))/sin(theta7); y8 = (rho8 -
x1*cos(theta8))/sin(theta8);
y9 = (rho9 - x1*cos(theta9))/sin(theta9); y10 = (rho10 -
x1*cos(theta10))/sin(theta10);
y11 = (rho11 - x1*cos(theta11))/sin(theta11); y12 = (rho12
- x1*cos(theta12))/sin(theta12);
y13 = (rho13 - x1*cos(theta13))/sin(theta13); y14 = (rho14
- x1*cos(theta14))/sin(theta14);
y15 = (rho15 - x1*cos(theta15))/sin(theta15); y16 = (rho16
- x1*cos(theta16))/sin(theta16);

figure %2
imshow(img), hold on
title('Harris Corners and Sub-pixel Accurate Corner')
plot(x1, y1,'LineWidth',1,'Color','magenta');
plot(x1, y2,'LineWidth',1,'Color','magenta');
plot(x1, y3,'LineWidth',1,'Color','magenta');
plot(x1, y4,'LineWidth',1,'Color','magenta');
plot(x1, y5,'LineWidth',1,'Color','magenta');
plot(x1, y6,'LineWidth',1,'Color','magenta');
plot(x1, y7,'LineWidth',1,'Color','magenta');
plot(x1, y8,'LineWidth',1,'Color','magenta');
```

```
plot(x1, y9,'LineWidth',1,'Color','magenta');
plot(x1, y10,'LineWidth',1,'Color','magenta');
plot(x1, y11,'LineWidth',1,'Color','magenta');
plot(x1, y12,'LineWidth',1,'Color','magenta');
plot(x1, y13,'LineWidth',1,'Color','magenta');
plot(x1, y14,'LineWidth',1,'Color','magenta');
plot(x1, y15,'LineWidth',1,'Color','magenta');
plot(x1, y16,'LineWidth',1,'Color','magenta');
% Find the intersection point of the two lines
intersection1 = [cos(theta1) sin(theta1); cos(theta2)
sin(theta2)] \ [rho1; rho2];
intersection2 = [cos(theta3) sin(theta3); cos(theta4)
sin(theta4)] \ [rho3; rho4];
intersection3 = [cos(theta5) sin(theta5); cos(theta6)
sin(theta6)] \ [rho5; rho6];
intersection4 = [cos(theta7) sin(theta7); cos(theta8)
sin(theta8)] \ [rho7; rho8];
intersection5 = [cos(theta9) sin(theta9); cos(theta10)
sin(theta10)] \ [rho9; rho10];
intersection6 = [cos(theta11) sin(theta11); cos(theta12)
sin(theta12)] \ [rho11; rho12];
intersection7 = [cos(theta13) sin(theta13); cos(theta14)
sin(theta14)] \ [rho13; rho14];
intersection8 = [cos(theta15) sin(theta15); cos(theta16)
sin(theta16)] \ [rho15; rho16];

% Plot the intersection point
plot(intersection1(1), intersection1(2), 'mo',
'LineWidth',2)
plot(intersection2(1), intersection2(2), 'mo',
'LineWidth',2)
plot(intersection3(1), intersection3(2), 'mo',
'LineWidth',2)
plot(intersection4(1), intersection4(2), 'mo',
'LineWidth',2)
plot(intersection5(1), intersection5(2), 'mo',
'LineWidth',2)
plot(intersection6(1), intersection6(2), 'mo',
'LineWidth',2)
plot(intersection7(1), intersection7(2), 'mo',
'LineWidth',2)
```

```
plot(intersection8(1), intersection8(2), 'mo',
'LineWidth',2)

% H = detectHarrisFeatures(I);
% plot(H.selectStrongest(100));

% Find the Harris corners
C_H = corner(I, 'Harris');
plot(C_H(:,1),C_H(:,2),'bo');

% Find the distance between the two points
% The real corner from inspection lies around (1085.5,
1837.5), while the
% Harris corner lies on (1086, 1838)
len = norm(intersection7 - [1085.5; 1837.5]);
len2 = norm([1086; 1838] - [1085.5; 1837.5]);

fprintf('The distance between the Hough corner and the real
corner is %.4f.\n', len)
fprintf('The distance between the Harris corner and the
real corner is %.4f.\n', len2)

toc
```