NYU CS-GY 6643, Computer Vision
Assignment 1

Name: Amitesh Kumar Sah
NYU ID: N19714360

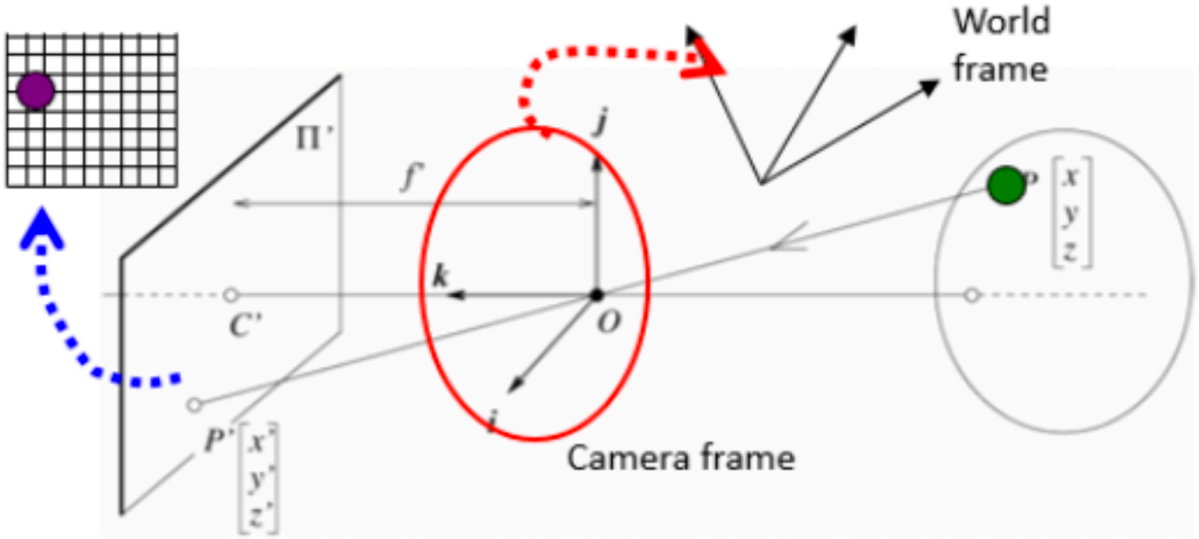
Problem 6: Camera Calibration

Objective:
To calibrate a (digital) camera so as to be able to capture images of objects from known locations and with a known camera model.

Resources Used

1. Mobile Phone (Asus Zenfone 5)
2. Two sheets of checkerboard
3. Matlab Program
4. Glue stick , scale and color pencil

Calibration Setup

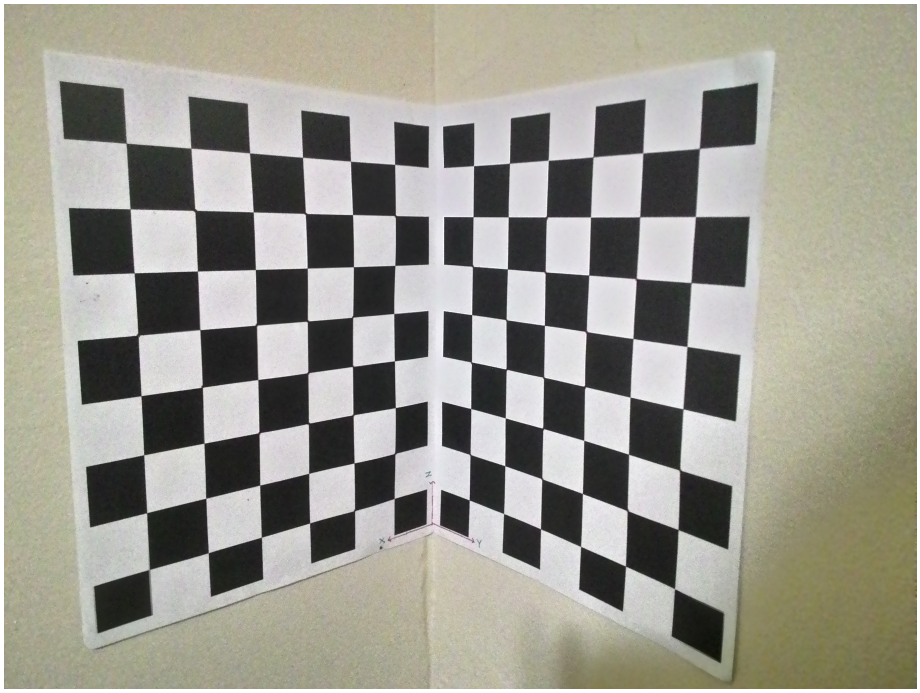


Procedure:

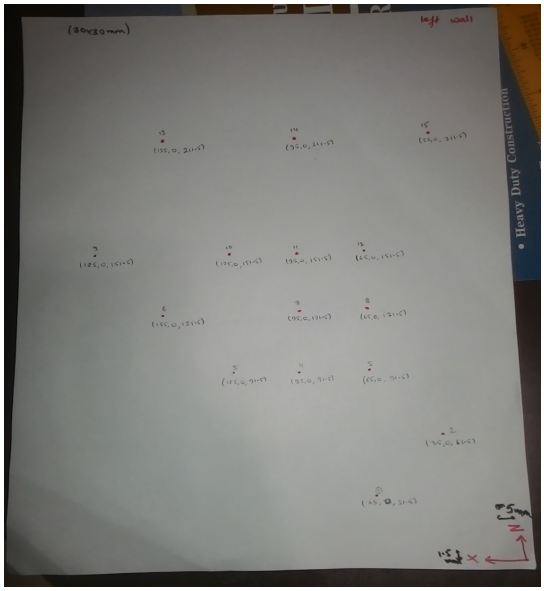
Physical Setup

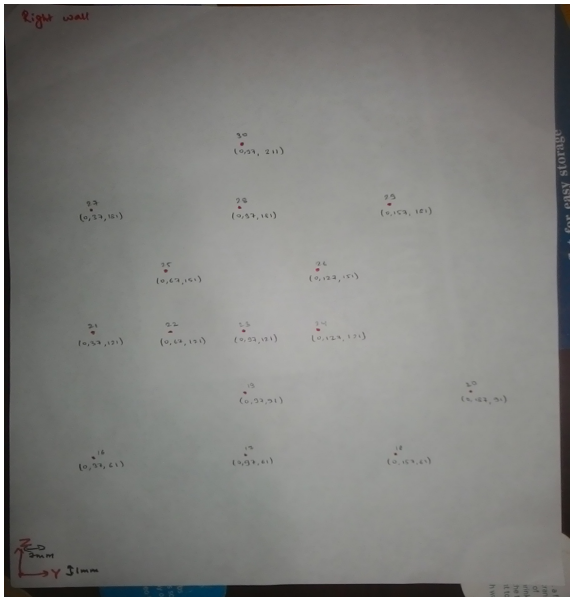
1. Take two page printout of checkerboard .
2. Find the appropriate wall corner where the walls are at 90 degree to each other.
3. Glue the checkerboard onto the wall as shown below in picture .
4. Draw the X,Y,Z axis frame for the world coordinates.

5. Draw 6 or more than 6 points on the checkerboard. Its convenient to measure if the coordinates are at the corner edges of black and white boxes. In my experiment I selected 30 points on the checker board.



6. Now measure the position of the points in Cartesian form from the origin of the XYZ frame.
7. Here I used a different technique to measure the position. I took two blank pages of same size and replicated the coordinate on the blank page and measured the deviation of X, Y , Z from the first row and column of checkerboard. And instead of measuring individual points manually with scale , I measured the size of one box in a checker board. The task was simplified now. So I got the position of each point by just counting the number of boxes, multiplying by 30mm and adding the deviation from X,Y,Z axis respectively . Here is a snapshot.





Data Capturing

8. Select the phone in which you can find the detail information about the sensor used in phone, example, the focal length, the size of the sensor, etc.
9. Position your camera to the level of the setup of checkerboard. Do not zoom in or zoom out. Take the picture so that the whole checkerboard fits the screen approximately as shown above
10. Also note down the approximate distance of the camera from the checkerboard.
11. Load the image file in your computer in the appropriate folder.
12. Write down the world coordinates into a text file with x,y,z coordinates separated by a comma and save it as 'world_coordinates.txt'

Specification of camera used.

Asus Zenfone 5 black

Camera:
Primary : 8MP, f/2.0, autofocus, LED Flash
Rear Sensor: Sony IMX219PQ CMOS
¼" BSI Sensor, resolution 8MP
pixel size: 1.12 micrometer
29mm-effective f/2.0 lens
Image size: Diagonal 4.60mm(Type ¼)
Total number of pixel: 3296(H) * 2512(V)
Approx 8.28 MPixels
Number of active pixels: 3280(H) * 2462(V)
Approx 8.08 MPixel
Unit cell size: 1.12 um(H) * 1.12um(V)

World Coordinate(X,Y,Z):
65,0,31.5
35,0,61.5
125,0,91.5
95,0,91.5
65,0,91.5
155,0,121.5
95,0,121.5
65,0,121.5
185,0,151.5
125,0,151.5
95,0,151.5
65,0,151.5
155,0,211.5
95,0,211.5
35,0,211.5
0,37,61
0,97,61
0,157,61
0,97,91
0,187,91
0,37,121
0,67,121
0,97,121
0,127,121
0,67,151
0,127,151
0,37,181
0,97,181
0,157,181
0,97,211

Image Coordinate(x,y,z)

1255.5513,1813.7531
1398.5319,1610.1142
913.2646,1536.4575
1086.5743,1506.1283
1246.8858,1471.4664
709.62566,1371.8133
1090.9071,1319.8204
1246.8858,1293.8239
479.99027,1185.5053
904.59912,1150.8434
1086.5743,1133.5124

1251.2186,1120.5142
687.96195,739.23274
1082.2416,752.23097
1402.8646,756.56372
1671.4947,1614.4469
1922.7938,1692.4363
2230.4186,1813.7531
1927.1265,1519.1265
2416.7265,1666.4398
1675.8274,1280.8257
1797.1442,1302.4894
1931.4593,1332.8186
2074.4398,1371.8133
1797.1442,1129.1796
2083.1053,1168.1743
1675.8274,938.53894
1940.1248,960.20265
2260.7478,981.86637
1940.1248,760.89646

Program in Matlab R2015a version

```
clc; clear all; close all
% Input world coordinates
[PX,PY,PZ]=textread('world coordinate.txt','', 'delimiter', ',');


% Input pixel coordinates by mouse
 Checkerboard=imread('checkerboard.jpg');
  figure,imshow(Checkerboard);
%   [r,c]=size(Checkerboard)
% [px,py] = ginput(30);
%  A1=[px,py];
% dlmwrite('image_coordinate.txt',A1,'precision',8)

% Load the image coordinates from the text file
[px,py]=textread('image_coordinate.txt','', 'delimiter', ',');

% Plotting 3D fiducial points and corresponding image points
figure,
subplot(1,2,1),scatter3(PX,PY,PZ) %origin
title('World Coordinates');
axis equal;   %# Make the axes scales match
hold on;      %# Add to the plot
xlabel('x');ylabel('y');zlabel('z');

subplot(1,2,2),imshow(Checkerboard), hold on, scatter(px,py)
```

```
xlabel('x');ylabel('y');
title('Image Coordinates');
% For each point in image and world, relation is
% (m1-ui*m3)*Pi=0
% (m1-vi*m3)*Pi=0
% Form a matrix of Q*m=0 considering all the point

% matrix M=[a b c d
%           e f g h
%           i j k l]    3*4

% Define the world points in one matrix
l=length(PX);
one=ones(l,1);
Pi=[PX PY PZ one];

% Create a  P matrix
j=1;
zero=zeros(1,4);
for i=1:30
   P(j,:)=[Pi(i,:) zero -px(i).*Pi(i,:)];
   P(j+1,:)=[zero Pi(i,:) -py(i).*Pi(i,:)];
   j=j+2;
end


% Solve 'm' to minimise (Q*m)^2

% Perform SVD of P
[U,S,V]=svd(P);
[min_val,min_index]=min(diag(S(1:12,1:12)));

% m is given by right singular vector of min. singular value
m=V(1:12,min_index);

% Projection matrix reshaping
M=[m(1:4,1)';m(5:8,1)';m(9:12)']

%%
% 2)Estimation of Intrinsic and Extrinsic Parameter

A=M(1:3,1:3);
b=M(1:3,4);

a1=A(1,:);
a2=A(2,:);
a3=A(3,:);
```

```
rho=-1/norm(a3);
r3=rho*a3;
x0=(rho^2)*dot(a1,a3);
y0=(rho^2)*dot(a2,a3);

cos_teta=-(dot(cross(a1,a3),cross(a2,a3)))/((norm(cross(a1,a3)))*(norm(cross(a2,a3))));
sine_teta=sqrt(1-cos_teta^2);
alpha=(rho^2)*norm(cross(a1,a3))*sine_teta;
beta=(rho^2)*norm(cross(a2,a3))*sine_teta;

r1=cross(a2,a3)/norm(cross(a2,a3));
r2=cross(r3,r1);

teta=acosd(cos_teta);
cot_teta=cos_teta/sine_teta;
kappa=[alpha,-alpha*cot_teta,x0;0,beta/sine_teta,y0;0,0,1];
R=[r1;r2;r3]
kappa_inverse=inv(kappa);
t=(rho*kappa_inverse*b)'

% Creating a table for intrinsic paramenter
Intr_par={'alpha';'beta';'teta';'x0';'y0'};
value=[alpha;beta;teta;x0;y0];
Intrinsic_Parameter=table(value,'RowNames',Intr_par)
```

Step by step explanation of the program

1. Load the text file into your Matlab program and read it and store it in separate variable PX, PY ,PZ. We know the coordinates in text file is separated by a comma , so we use comma as a delimiter.

```
% Input world coordinates
[PX,PY,PZ]=textread('world coordinate.txt','', 'delimiter', ',');
```

2. Corresponding pixel locations are obtained in the image, e.g. by using MATLAB function ginput() to acquire image positions with mouse clicking, and show clicked position with function scatter(). This gives us the list of points in image space associated with the world points pi = (u, v) T . Do not run this part of the program once we have saved the image coordinates in a text file.

```
% Input pixel coordinates by mouse
Checkerboard=imread('checkerboard.jpg');
figure,imshow(Checkerboard);
[r,c]=size(Checkerboard)
[px,py] = ginput(30);
A1=[px,py];
dlmwrite('image_coordinate.txt',A1,'precision',8)
```

3. Once you have saved the image coordinates, put the step 2 in comments, except the line that reads the image and displays it.

```
% Load the image coordinates from the text file
```

```
[px,py]=textread('image_coordinate.txt','', 'delimiter', ',');
```

4. Now we plot the world coordinates and the image coordinate in one figure.

```
% Plotting 3D fiducial points and corresponding image points
figure,
subplot(1,2,1),scatter3(PX,PY,PZ) %origin
title('World Coordinates');
axis equal;  %# Make the axes scales match
hold on;     %# Add to the plot
xlabel('x');ylabel('y');zlabel('z');

subplot(1,2,2),imshow(Checkerboard), hold on, scatter(px,py)
xlabel('x');ylabel('y');
title('Image Coordinates');
```

5. Make the world coordinates Homogeneous by putting one at each coordinate end in the matrix.

```
% Define the world points in one matrix
l=length(PX);
one=ones(l,1);
Pi=[PX PY PZ one];
```

6. We stack all the coordinates from i=1 to n , into one big matrix of the form

$$\begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \cdots & \cdots & \cdots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{pmatrix}$$

```
% Create a  P matrix
j=1;
zero=zeros(1,4);
for i=1:30
   P(j,:)=[Pi(i,:) zero -px(i).*Pi(i,:)];
   P(j+1,:)=[zero Pi(i,:) -py(i).*Pi(i,:)];
   j=j+2;
end
```

7. We perform Least Square Error. As we need to find unit vector m, that minimizes the |Qm|^2. The Eigen vector associated to the eigen value of the matrix Q^T Q, gives us 'm' , because it is the unit vector x that minimizes the x^TQ^TQx .

   Least squares method is used to estimate the calibration matrix. There are 60 homogeneous linear equations in twelve variables, which are the coefficients of the calibration matrix M. Lets denote this system of linear equations as

$$Pm = 0,$$
$$m := [m1\ m2\ m3]\ T \text{ ----------(1)}$$

where, m1, m2, m3 are first, second and third rows of the matrix M respectively. m is a 12 ×1 vector, and P is a 60 × 12 matrix.

The problem of least square estimation of P is defined as

$$\min||Pm||^2 \text{ , subject to } |m|^2 = 1.\text{---------(2)}$$

As it turns out, the solution of above problem is given by the eigenvector of matrix P TP having the least eigenvalue. The eigenvectors of matrix P TP can also be computed by performing the singular value decomposition (SVD) of P. The 12 right singular vectors of P are also the eigenvectors of P TP. The SVD method is used here to get the eigenvector corresponding to the least eigenvalue. This eigenvector is the solution to the above problem. Reorganizing the 12 ×1 vector m in a matrix of 4×3 gives us the matrix M. The first three elements of the third row of this matrix denote one of the three rotation vectors. As we know that the norm of rotation vectors is unity, this matrix is scaled by the norm of the third rotation vector to get the calibration matrix in canonical form. We take this matrix as the final calibration matrix M.

Here we used Singular Vector Decomposition technique (SVD) as we can use different size of matrix P and P is factored as U*S*V, where U and V are the orthogonal matrices containing the singular vector , S is a matrix of form [D 0;0 0] , where D is a diagonal matrix containing the singular vector of P. So, we choose the minimum value from the diagonal element of S and the location of the minimum value.

```
% Perform SVD of P
[U,S,V]=svd(P);
[min_val,min_index]=min(diag(S(1:12,1:12)));

% m is given by right singular vector of min. singular value
m=V(1:12,min_index);

% Projection matrix reshaping
M=[m(1:4,1)';m(5:8,1)';m(9:12)']
```

8. We calculate the intrinsic and extrinsic parameters by using the formula as mentioned in the book.

```
A=M(1:3,1:3);
b=M(1:3,4);

a1=A(1,:);
a2=A(2,:);
a3=A(3,:);

rho=-1/norm(a3);
r3=rho*a3;
x0=(rho^2)*dot(a1,a3);
```

```
y0=(rho^2)*dot(a2,a3);

cos_teta=-(dot(cross(a1,a3),cross(a2,a3)))/((norm(cross(a1,a3)))*(norm(cross(a2,a3))));
sine_teta=sqrt(1-cos_teta^2);
alpha=(rho^2)*norm(cross(a1,a3))*sine_teta;
beta=(rho^2)*norm(cross(a2,a3))*sine_teta;

r1=cross(a2,a3)/norm(cross(a2,a3));
r2=cross(r3,r1);

teta=acosd(cos_teta);
cot_teta=cos_teta/sine_teta;
kappa=[alpha,-alpha*cot_teta,x0;0,beta/sine_teta,y0;0,0,1];
R=[r1;r2;r3]
kappa_inverse=inv(kappa);
t=(rho*kappa_inverse*b)'
```

9. We display the intrinsic parameter result using the function table

```
% Creating a table for intrinsic paramenter
Intr_par={'alpha';'beta';'teta';'x0';'y0'};
value=[alpha;beta;teta;x0;y0];
Intrinsic_Parameter=table(value,'RowNames',Intr_par)
```

Calculation of Intrinsic and Extrinsic Parameters

Generally we used the matrix and vector formula to compute the individual intrinsic and extrinsic parameter. We used the facts such as 1) the inverse of a rotation matrix is equal to its transpose , 2) the determinant of rotation matrix is one , 3) Column of rotation matrix forms a right handed orthonormal coordinate system of R^3. 4) Rows of a rotation matrix has unit length and perpendicular to each other . 5) dot product of two orthonormal vector is zero.

I used the formula mentioned in the textbook and slides taught in the class.

Write $M = (A, b)$, therefore

$$\rho(A \quad b) = K(R \quad t) \Longleftrightarrow \rho\begin{pmatrix} a_1^T \\ a_2^T \\ a_3^T \end{pmatrix} = \begin{pmatrix} \alpha r_1^T - \alpha \cot\theta r_2^T + u_0 r_3^T \\ \frac{\beta}{\sin\theta} r_2^T + v_0 r_3^T \\ r_3^T \end{pmatrix}$$

Using the fact that the rows of a rotation matrix have unit length and are perpendicular to each other yields

$$\begin{cases} \rho = \varepsilon/|a_3|, \\ r_3 = \rho a_3, \\ u_0 = \rho^2(a_1 \cdot a_3), \\ v_0 = \rho^2(a_2 \cdot a_3), \end{cases} \quad \text{where} \quad \varepsilon = \mp 1.$$

Since θ is always in the neighborhood of $\pi/2$ with a positive sine, we have

$$\begin{cases} \rho^2(a_1 \times a_3) = -\alpha r_2 - \alpha \cot\theta r_1, \\ \rho^2(a_2 \times a_3) = \frac{\beta}{\sin\theta} r_1, \end{cases} \quad \text{and} \quad \begin{cases} \rho^2|a_1 \times a_3| = \frac{|\alpha|}{\sin\theta}, \\ \rho^2|a_2 \times a_3| = \frac{|\beta|}{\sin\theta}. \end{cases}$$

Thus,

$$\begin{cases} \cos\theta = -\frac{(a_1 \times a_3) \cdot (a_2 \times a_3)}{|a_1 \times a_3||a_2 \times a_3|}, \\ \alpha = \rho^2|a_1 \times a_3| \sin\theta, \\ \beta = \rho^2|a_2 \times a_3| \sin\theta, \end{cases} \quad \text{and} \quad \begin{cases} r_1 = \frac{\rho^2 \sin\theta}{\beta}(a_2 \times a_3) = \frac{1}{|a_2 \times a_3|}(a_2 \times a_3), \\ r_2 = r_3 \times r_1. \end{cases}$$

The translation parameters can now be recovered by writing $Kt = \rho b$, and hence $t = \rho K^{-1} b$. In practical situations, the sign of t_z is often known in advance (this corresponds to knowing whether the origin of the world coordinate system is in front or behind the camera), which allows the choice of a unique solution for the calibration parameters.

Result

The calibration matrix obtained is
M =

0.0026	-0.0004	0.0001	-0.6348
0.0005	0.0005	0.0022	-0.7726
0.0000	0.0000	0.0000	-0.0004

Extrinsic Parameter
Rotation matrix

R =

-0.7122	0.7018	0.0119
0.0627	0.0805	-0.9948
-0.6991	-0.7078	-0.1014

Translation Matrix
t =

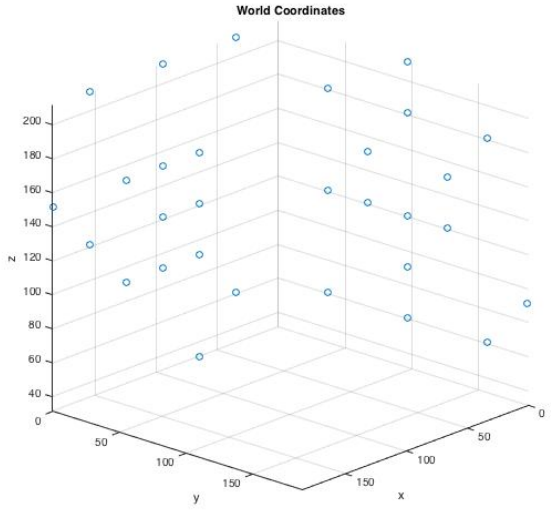
-44.6361 155.4035 470.5426

Intrinsic_Parameter

	value
alpha	2424.3
beta	2459.6
teta	89.771
x0	1775.7
y0	1065.1

Discussion of each result

Verification of world coordinates and corresponding image coordinates. Here we can see that our coordinates from the world are correct and its mapping to the image coordinate is right .



Calibration Matrix (M)- We obtained this matrix using the SVD least square error technique.

Intrinsic Parameter

Image Center (X0 and Y0)
Image resolution is 3264*2448 so our image center theoretically should be at X0=1632 and Y0=1224. From the result obtained, we got the image center at X0= 1776 and Y0= 1066. So the image center does not coincide with the theoretical value. This shows that there is a shift of image center by 144 pixel toward right and by 158 pixel towards upward.

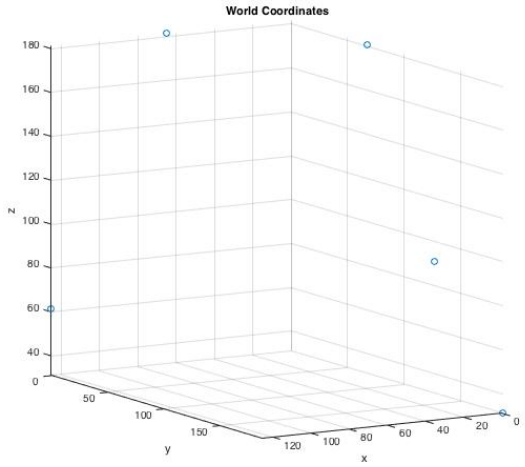
Skew of Camera Coordinate System
Theoretically, camera coordinate system should not be skewed and the angle between the two image axes is 90 degree. However, we obtained the angle to be 89.771 degree and here there is slight skew of 0.32 degree, which is negligible. This occurs due to manufacturing error.

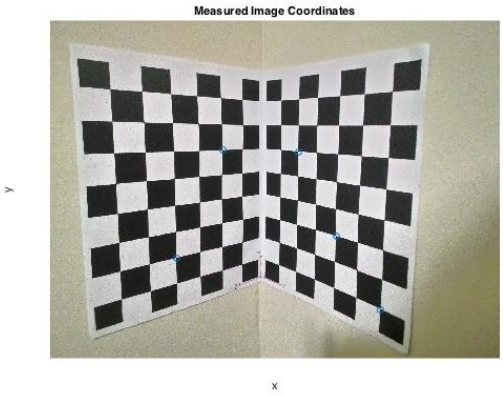
Magnification (alpha and beta)
'k' and 'l' are the pixel density . 'k' is the number of pixel /mm along x-axis and 'l' is the number of pixel/mm along y axis. f is the distance of the sensor image plane from the lens.
Alpha=K*f
Beta=l*f
My mobile focal length is 29mm. Result obtained from the calibration shows
alpha 2424.3
beta 2459.6

Extrinsic Parameter
Rotation Matrix
The magnitude of each row of rotation matrix is unity and also the dot product of r1 ,r2,r3 tends to zero.
Translation Matrix.
The distance of the world frame from the lens is quite accurate. The Z was when measured was around 50 cm and the result we obtained from the matrix calibration is 470mm which is quite comparable.

Reconstruct the image coordinates p from the world coordinates P using estimate of calibration matrix

I took 5 points in the world and measured the corresponding point in the image





World coordinates:

W =
[X Y Z]

125.0000	0	61.5000
65.0000	0	181.5000
0	127.0000	91.0000
0	187.0000	31.0000
0	67.0000	181.0000

Corresponding measured image coordinate

[x y]
1.0e+03 *

0.9215	1.7255
1.2515	0.9395
2.0780	1.5640
2.3980	2.0960
1.8012	0.9542

Calculated image coordinate using estimate of calibration matrix
cal_image =

1.0e+03 *

0.9181 1.7311
1.2471 0.9376
2.0713 1.5624
2.4010 2.0901
1.8034 0.9481

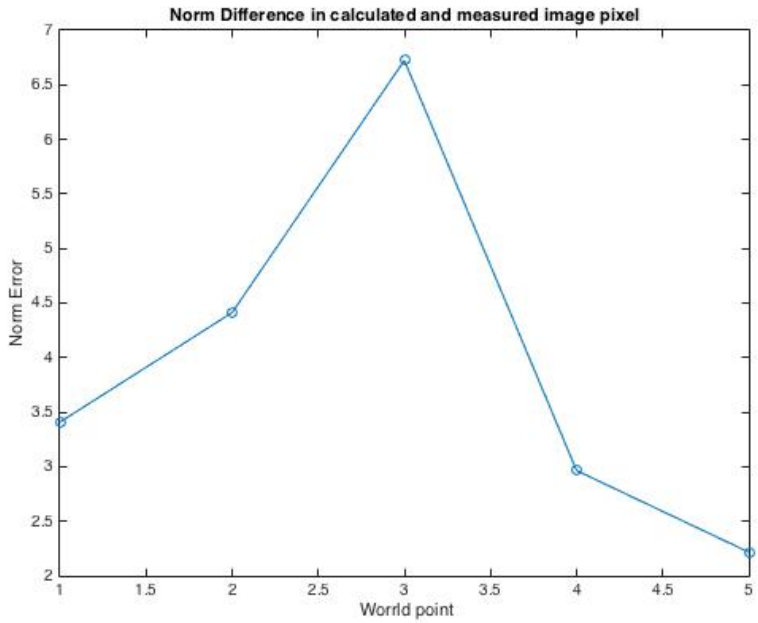
Error in magnitude

3.4089
4.4129
6.7238
2.9640
2.2164

Table showing the world coordinate , measured image coordinate , calculated image coordinate, and error in pixel value.

World Coordinate	Measured Image Coord.(pixel)	Calculated Image Coord.(pixel)	Norm Error	Error in pixel value
(125,0,61.5)	(921.5, 1725.5)	(918.1, 1731.1)	3.4089	-3.4089 5.5873
(65,0,181.5)	(1251.5, 939.5)	(1247.1, 937.6)	4.4129	-4.4129 -1.8727
(0,127,91)	(2078, 1564)	(2071.3, 1562.4)	6.7238	-6.7238 -1.5885
(0,187,31)	(2398, 2096)	(2401, 2090.1)	2.9640	2.9640 -5.9225
(0,67,181)	(1801.2, 954.2)	(1803.4, 948.1)	2.2164	2.2164 -6.1150

Plot showing the error in magnitude



Result:
We obtained the image coordinate from Matlab program by using the calibration matrix and the image coordinate obtained is approximately same as the measured image coordinate. The error is calculated by taking the difference of calculated image and measured image and then we take resultant magnitude to look at the error from the measured image. Norm Error is too small. Hence, our calibration matrix gives the perfect reconstruction of the image coordinate from the world coordinate.

Matlab program for Reconstruction of image coordinated

```
% Reconstruct the image coordinates p from the world coordinates P using estimate of calibration
matrix
W =[125,0,61.5;65,0,181.5;0,127,91;0,187,31;0,67,181]

% Input corresponding image coordinate by mouse
%  Checkerboard=imread('checkerboard.jpg');
 figure,imshow(Checkerboard);
% %   [r,c]=size(Checkerboard)
% [ipx,ipy] = ginput(5);
% A2=[ipx,ipy];
% dlmwrite('measured_image_coordinate.txt',A2,'precision',8)
% Plotting 3D fiducial points and corresponding image points


% Load the image coordinates from the text file
[ipx,ipy]=textread('measured_image_coordinate.txt','', 'delimiter', ',')

figure,
subplot(1,2,1),scatter3(W(:,1),W(:,2),W(:,3)) %origin
title('World Coordinates');
axis equal;   %# Make the axes scales match
hold on;     %# Add to the plot
xlabel('x');ylabel('y');zlabel('z');

subplot(1,2,2),imshow(Checkerboard), hold on, scatter(ipx,ipy)
xlabel('x');ylabel('y');
title('Measured Image Coordinates');

ones_i=ones(1,5);
W(:,4)=ones_i';

for i=1:5
cal_image_co(:,i)=M*(W(i,:))';
cal_ix(i,:)=cal_image_co(1,i)/cal_image_co(3,i);
cal_iy(i,:)=cal_image_co(2,i)/cal_image_co(3,i);

end
```

```
cal_image=[cal_ix,cal_iy]
error=[cal_ix-ipx,cal_iy-ipy]

for i=1:5
mag_error(i,:)=norm(error(i,1),error(i,2));
end

figure
plot(mag_error,'-o');
title('Norm Difference in calculated and measured image pixel');
xlabel(' Worrld point');
ylabel('Norm Error');
```

REFERENCE

1. **Course Textbook**
   Computer Vision: A Modern Approach (2nd Edition) 2nd Edition by David A. Forsyth (Author), Jean Ponce (Author)

2. **Slides taught by Professor Guido Gerig in the lecture.**
3. **CVonline: Monocular Camera calibration**
   http://homepages.inf.ed.ac.uk/cgi/rbf/CVONLINE/entries.pl?TAG250
4. Camera calibration toolbox for matlab
   http://www.vision.caltech.edu/bouguetj/calib_doc/