ENPM673 – Perception for Autonomous Robots
Project 3

Question 1:

1. What is the minimum number matching points to solve this mathematically?
2. What is the pipeline or the block diagram that needs to be done in order to calibrate this camera given the image above.
3. First write down the mathematical formation for your answer including steps that need to be done to find the intrinsic matrix K.
4. Find the P matrix.
5. Decompose the P matrix into the Translation, Rotation and Intrinsic matrices using the
Gram–Schmidt process and compute the reprojection error for each point.

Note: You are only allowed to use numpy for this question. No marks will be given if you use any
other library/tool.

Solution:

1. To solve this question we need to compute the Calibration matrix P. To compute the P matrix which has 11 unknowns and for computing these 11 unknowns we need 2 equations. Hence need 6 correspondence points to compute the matrix P. Hence to solve the question mathematically we need **6** matching points.

2. **<u>Pipeline:</u>**

   -To compute the initrinsic matrix K we need to compute P martrix which is the projection matrix.

   - To compute the p matrix we need the A matrix which is:

$$\begin{bmatrix} \mathbf{0}_4^T & -w'_i \mathbf{X}_i^T & v'_i \mathbf{X}_i^T \\ w'_i \mathbf{X}_i^T & \mathbf{0}_4^T & -u'_i \mathbf{X}_i^T \\ -v'_i \mathbf{X}_i^T & u'_i \mathbf{X}_i^T & \mathbf{0}_4^T \end{bmatrix}$$

- In the A matrix the Xi corresponds to the world coordinates represented in the format Xi = [x-world_coordinate, y-world_coordinate, z-world_coordinate,1]
- The ui,vi,wi in the A matrix are computed from the image points x, y.

$$x_i = u / w, \quad y_i = v / w$$

- The formulation for computing the Projection matrix P is

$$\begin{bmatrix} \mathbf{0}_4^{\mathbf{T}} & -w'_{\mathbf{i}}\,\mathbf{X}_{\mathbf{i}}^{\mathbf{T}} & v'_{\mathbf{i}}\,\mathbf{X}_{\mathbf{i}}^{\mathbf{T}} \\ w'_{\mathbf{i}}\,\mathbf{X}_{\mathbf{i}}^{\mathbf{T}} & \mathbf{0}_4^{\mathbf{T}} & -u'_{\mathbf{i}}\,\mathbf{X}_{\mathbf{i}}^{\mathbf{T}} \\ -v'_{\mathbf{i}}\,\mathbf{X}_{\mathbf{i}}^{\mathbf{T}} & u'_{\mathbf{i}}\,\mathbf{X}_{\mathbf{i}}^{\mathbf{T}} & \mathbf{0}_4^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = 0$$

- Here we can compute the p matrix by doing the svd of A matrix.
- SVD function decomposes the p matrix into U,D,V-transpose

$$\mathbf{A} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^{\mathbf{T}}$$

- From the decomposition The P matrix corresponds to the last column of V vector.
- Thus we have computed the Projection matrix P.

```
P matrix
[-3.62171213e-02  2.37539778e-03  8.80579297e-02 -9.54125701e-01
  2.53640422e-02 -8.29092139e-02  2.78474979e-02 -2.68846199e-01
  3.48859064e-05  3.46644406e-06  3.92721669e-05 -1.26060255e-03]
```

- After computing the p matrix we need to reshape the P matrix from 12x1 matrix to 3x4 matrix.
- After reshaping the P matrix we normalise the matrix by dividing all the elements of the P matrix by the last element of that matrix.
- Using the resized P matrix P we are the translation vector C.

$$\mathbf{P}\,\mathbf{C} = 0:$$

- Since C is the null vector of P we can calculate C by performing the SVD on the P matrix.
- SVD of P gives

- $U D V^T$
- Here the the value of C vector is equal to the last column of vector V
- The found C matrix is normalized by dividing all its elements with the last element of C and finally removing the last element.
- Thus we have computed the Translation matrix C
-
```
Translation Matrix
[[16.01106826]
 [ 7.43927187]
 [17.21967888]]
```
- $P = K R \left[I_3 \mid -\tilde{C} \right]$
- By using the above equation we can calculate the Intrinsic matrix K and Rotational matrix R
- The KR value corresponds to the non-singular square matrix M.
- $M = K R$
- The M matrix can be computed by adding an 3x3 identity matrix to C and taking the inverse of the resulting matrix and performing matrix multiplication of this matrix to the projection matrix P.
- Thus the M matrix is obtained.
- From the obtained M matrix we can compute the Intrinsic matrix and Rotational matrix by performing RQ factorization on the M matrix.
- The RQ factorization of the M matrix decomposes it into a upper triangle matrix which is the Intrinsic matrix K and an orthogonal matrix which is the rotational matrix R.
- Here we perform the RQ factorization by using the Gram–Schmidt process.
- Thus by the RQ factorization of M matrix we have computed the Rotational matrix R and the Intrinsic matrix K.

```
Intrisic Matrix K
[[ 6.78713819e+01  7.10428037e-02  3.31962888e+01]
 [ 0.00000000e+00 -6.75951837e+01  2.54824021e+01]
 [ 0.00000000e+00  0.00000000e+00  4.17606430e-02]]
Rotational Matrix
[[ 0.74737193  0.00548748 -0.66438324]
 [ 0.04784159 -0.99781461  0.04557607]
 [-0.6626812  -0.06584743 -0.74600116]]
```

- Finally to compute the reprojection errors for each points.
- First we need to add a column of ones of 8x1 to the World Coordinates and find the transpose of that matrix.
- Then we need to take the dot product of P matrix and the changed World matrix.
- Then we are normalizing the obtained matrix by dividing all the elements in the matrix by its last element and removing the last element from the matrix.
- Then the reprojection error of each points are calculated by calculating the normalized of difference between the image points and the obtained matrix.
- Thus the Reprojection error for all the points are obtained.

```
Reprojection Errors for every points
The reprojection error for the point 1 = 0.2933771578291177
The reprojection error for the point 2 = 1.1360585621705546
The reprojection error for the point 3 = 0.9217783054589999
The reprojection error for the point 4 = 0.12486937190560454
The reprojection error for the point 5 = 0.12190867422398974
The reprojection error for the point 6 = 0.5552338467303624
The reprojection error for the point 7 = 0.10225835309858652
The reprojection error for the point 8 = 1.5580221059967057
```

Question 2:

In this problem, you will perform camera calibration using the concepts you have learned in class.
Assuming a pinhole camera model and ignoring radial distortion, we will be relying on a
calibration target (checkerboard in our case) to estimate the camera parameters. The calibration
target used can be found here.
This was printed on an A4 paper and the size of each square is 21.5 mm. Note that the Y axis has an
odd number of squares and X axis has an even number of squares. It is a general practice to neglect
the outer squares (extreme squares on each side and in both directions).
Thirteen images taken from a Google Pixel XL phone with focus locked can be downloaded from
here which you will use to calibrate.
For this question, you are allowed to use any in-built function.

- Find the checkerboard corners using any corner detection method (inbuilt OpenCV functions
such as `findChessboardCorners` are allowed) and display them for each image.
- Compute the Reprojection Error for each image using built-in functions in OpenCV
- Compute the K matrix
- How can we improve the accuracy of the K matrix?

Solution:

Pipeline:

1. The given images are stored in an package.
2. The images are read one by one and for each image the image are gone through preprocessing.
3. The given image are gray scaled to reduce the complexities for computation.

4. Then the corners of the image are computed using the findchessboardCorners function.
5. The accurate images are obtained by performing the threshold on the obtained image points.
6. Then the obtained corners are plotted in every given images using the drawChessboardCorners function
7. From the given object coordinates and obtained image coordinates the intrinsic matrix, distortion matrix,rotational matrix,translational matrix are computed by using the function calibrateCamera().
8. Thus the intrinsic matrix for the images are obtained
9.
```
calibration matrix(K)
[[2.72842760e+03 0.00000000e+00 9.97269362e+02]
 [0.00000000e+00 1.15698974e+03 7.69218139e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```
10. From the image points the error of the image is calculated by performing the norm function cv.norm().
11. After computing the error for each image the mean error for all the images are calculated.

```
srv@srv-Dell-G16-7620:~/perception673/project3$ /bin/python3 /home/srv/perception673/project3/project3.py
The projection error for image 1 = 0.04940387225589656
The projection error for image 2 = 0.0648664669665458
The projection error for image 3 = 0.06758002565199948
The projection error for image 4 = 0.02908817808108118
The projection error for image 5 = 0.1465210765025145
The projection error for image 6 = 0.14977268030511498
The projection error for image 7 = 0.07604768960459095
The projection error for image 8 = 0.08600951389748925
The projection error for image 9 = 0.12550063616389734
The projection error for image 10 = 0.08542158164673168
The projection error for image 11 = 0.09569387517560372
The projection error for image 12 = 0.05605824850156516
The projection error for image 13 = 0.03895962745665912
total error: 0.08237872863151458
```

Code for computing the error:

```
mean_error = 0
for i in range(len(object_coord)):
    image_points,_ = cv.projectPoints(object_coord[i],rvecs[i],tvecs[i],mtx,dist)
    error = cv.norm(image_coords[i],image_points,cv.NORM_L2)/len(image_points)
    print(f"The projection error for image {i+1} = {error}")
    mean_error += error


print("total error: {}".format(mean_error/len(object_coord)))
```

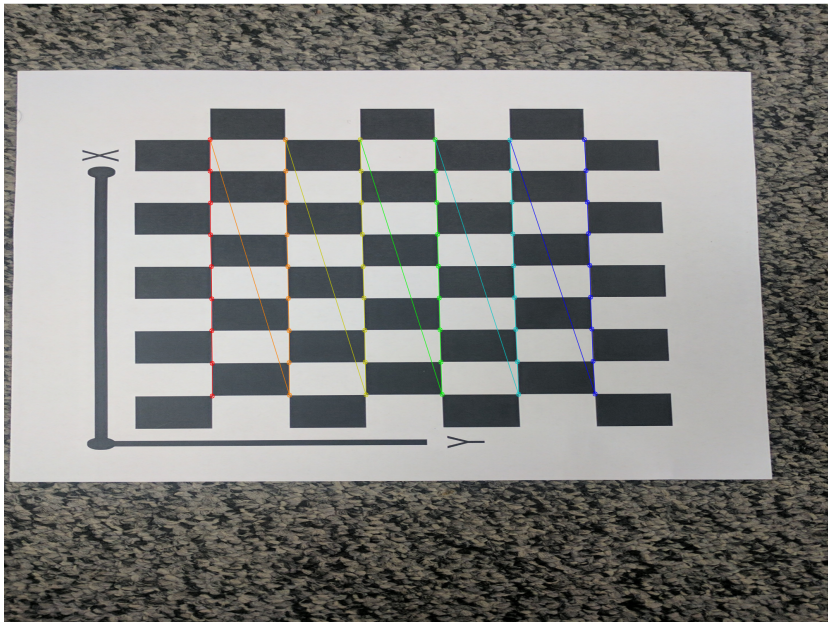
Code for computing the Intrinsic matrix K:

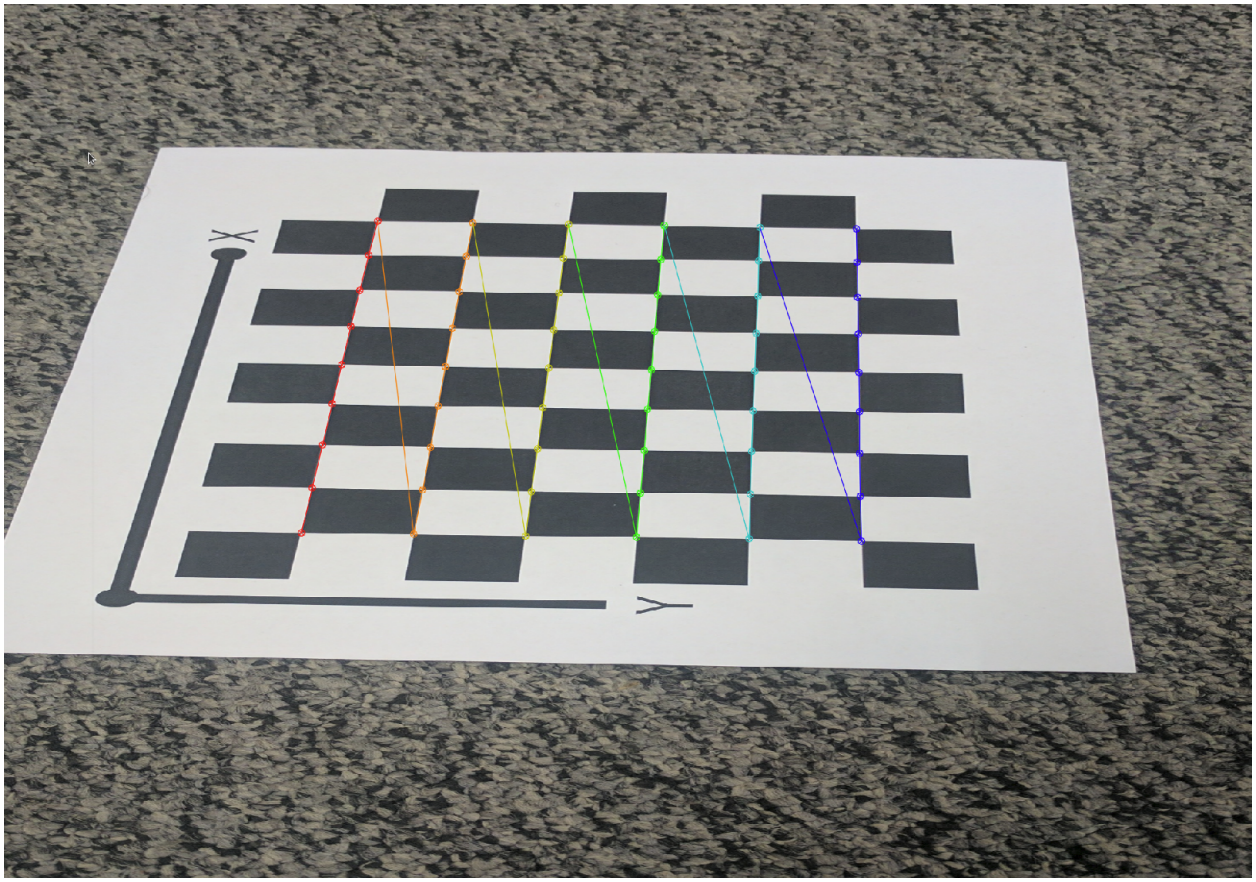
```
import cv2 as cv
import os
import matplotlib.pyplot as plt
import numpy as np

criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)
object_coord = []
image_coords = []
amc= []
arr_map = np.zeros((6*9,3), np.float32)
arr_map[:,:2] = np.mgrid[0:9,0:6].T.reshape(-1,2)

folder_path = "/home/srv/perception673/project3"

for filename in os.listdir(folder_path):
    if filename.endswith(".jpg"):
        image_path = os.path.join(folder_path, filename)
        image = cv.imread(image_path)
        image = cv.resize(image,(2000,1512))
        gray = cv.cvtColor(image,cv.COLOR_BGR2GRAY)
        ret,corner = cv.findChessboardCorners(gray, (9,6),None  )

        if ret:
            object_coord.append(arr_map*21.5)
            acc_corners = cv.cornerSubPix(gray,corner,(11,11),(-1,-1),criteria)
            image_coords.append(acc_corners)
            cv.drawChessboardCorners(image,(9,6),acc_corners,ret)
            cv.imshow("image",image)
            cv.waitKey(250)
ret, mtx, dist, rvecs, tvecs = cv.calibrateCamera(object_coord, image_coords, gray.shape[::-1], None, None)
```

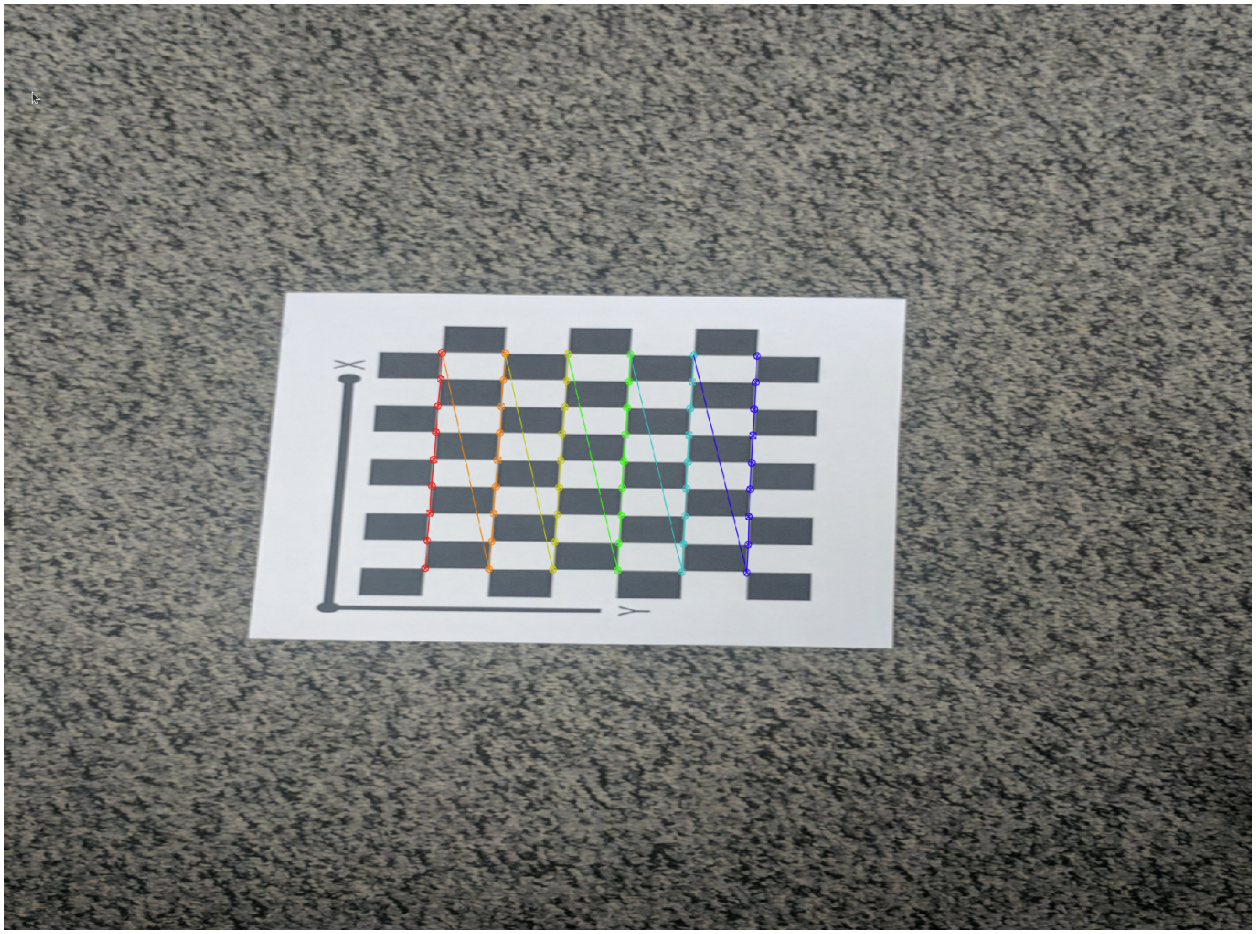
2.4: To improve the accuracy of intrinsic matrix K:

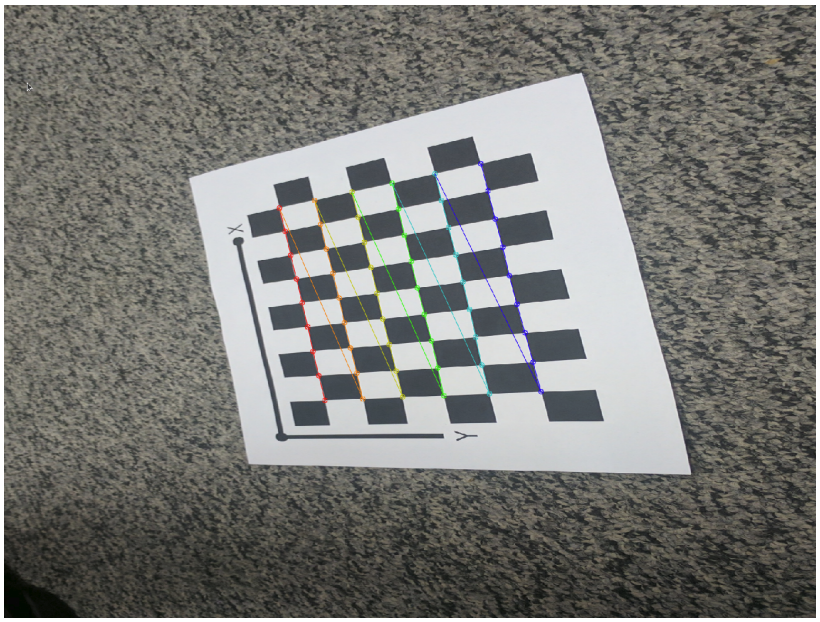
1. The improve the accuracy of intrinsic matrix we can increase the number of images used to calibrate the points of image.
2. It can be improved by taking the images of calibration from different position with different camera angles.
3. Usage of high resolution cameras to capture the images used for calibration.
4. Considering the distortion value of camera lens. By correcting the radial and tangential distortion of the camera, this gives us precise intrinsic matrix values.
5. Use precise dimensions of the checker board so as to obtain an precise intrinsic matrix.

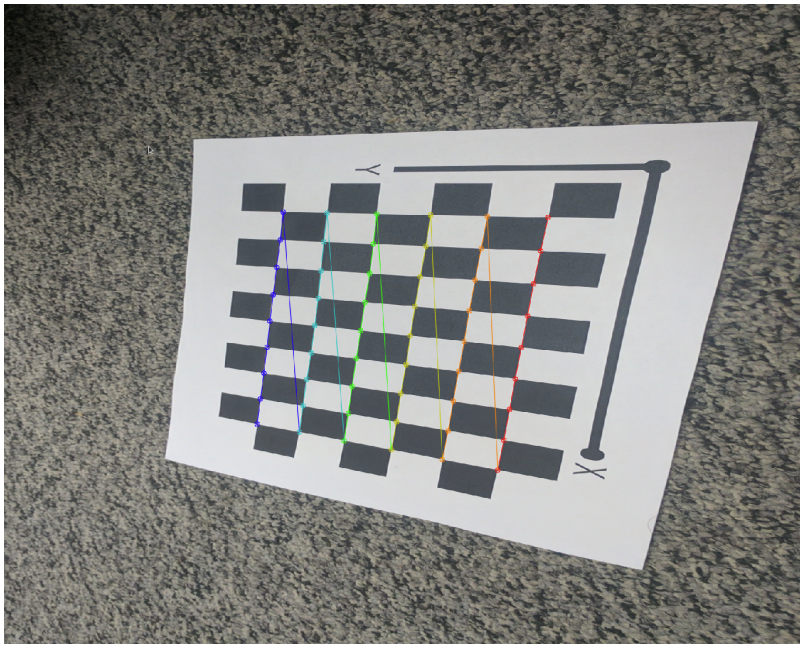
Obtained corners plotted in the given image:

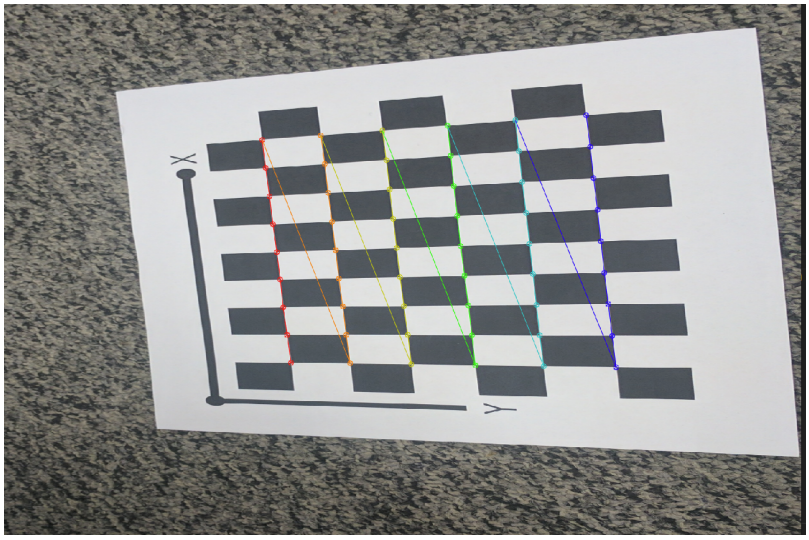


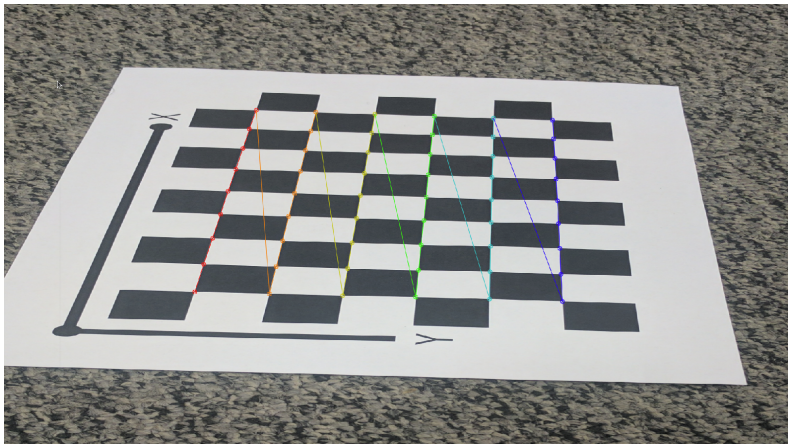

X
Y

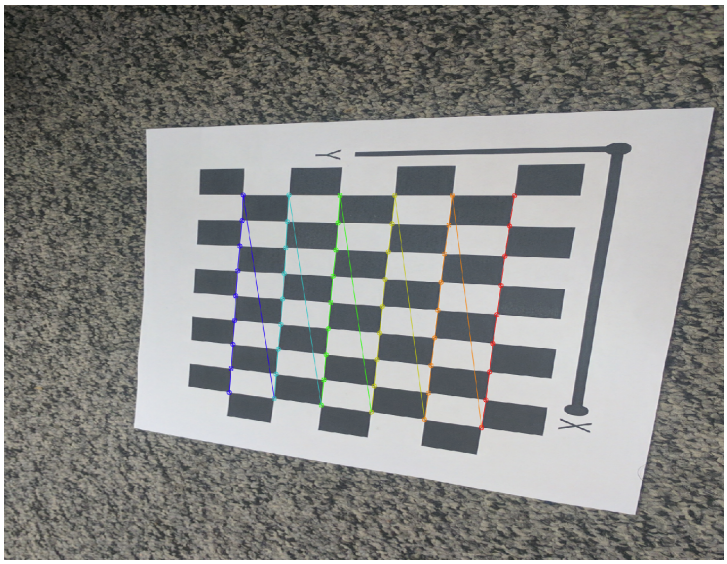

X
Y

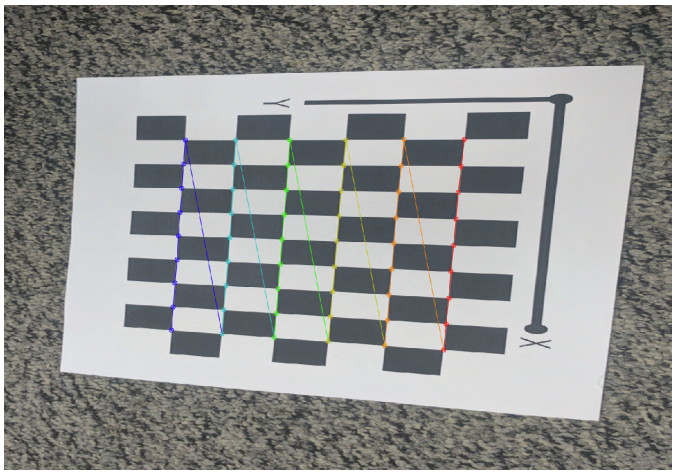

X
Y

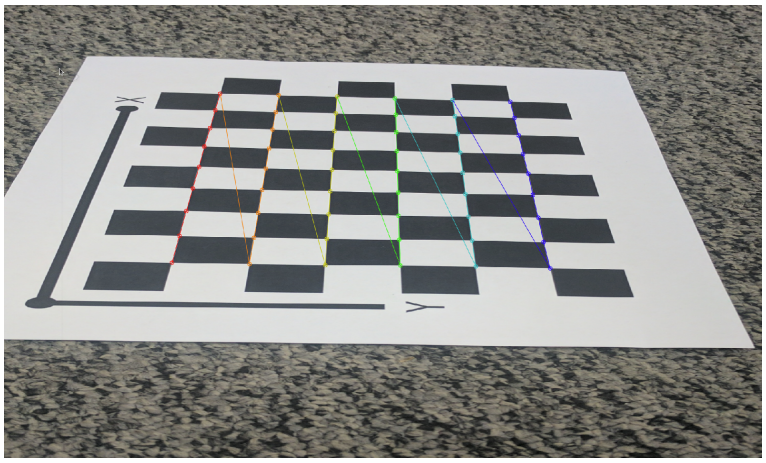

Y
X

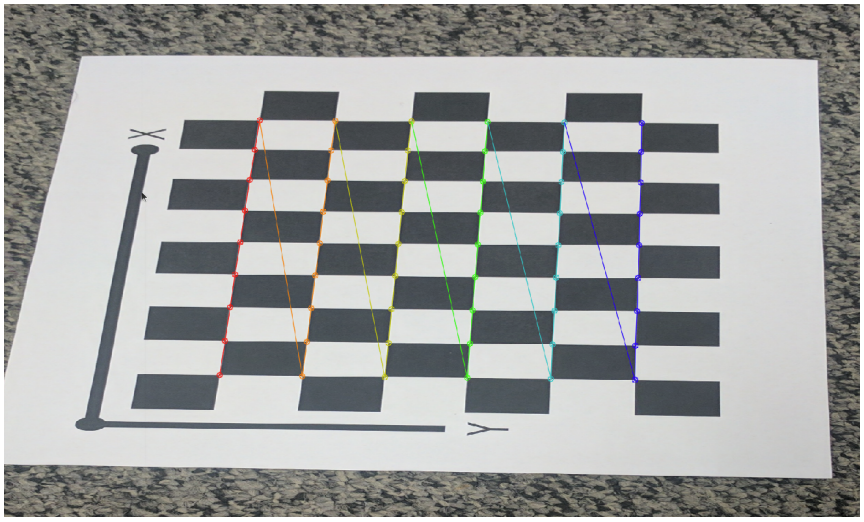

X
Y

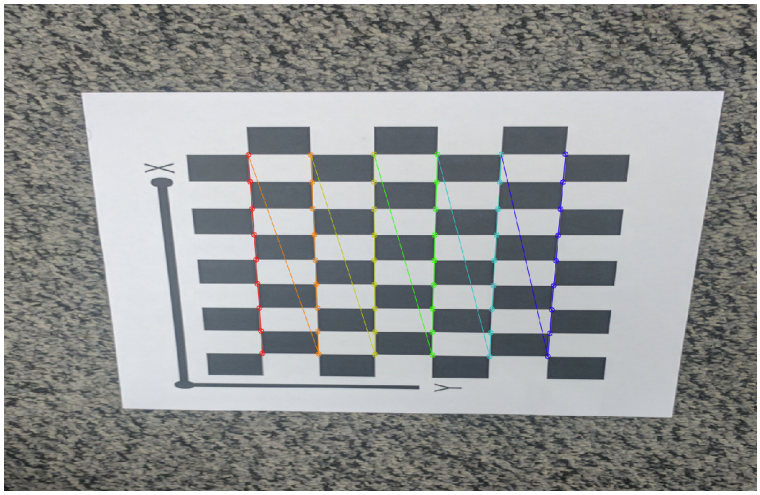

X
Y

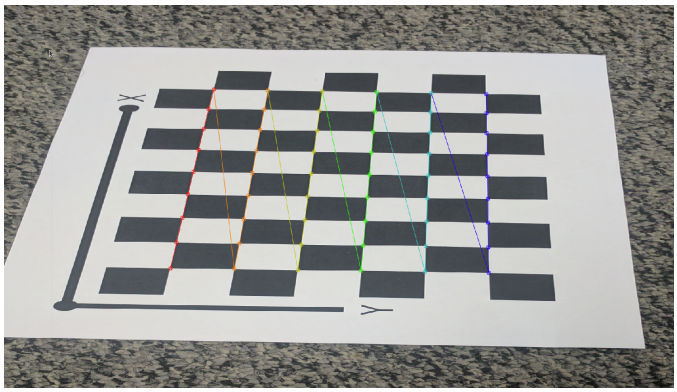

Y
X


Y
X


X
Y


X
Y


X
Y


X
Y