ENPM673: Project 2

Aditi Arun Bhoir

UID: 119197257

Question1 Calibrate the camera (Find the intrinsic matrix K)

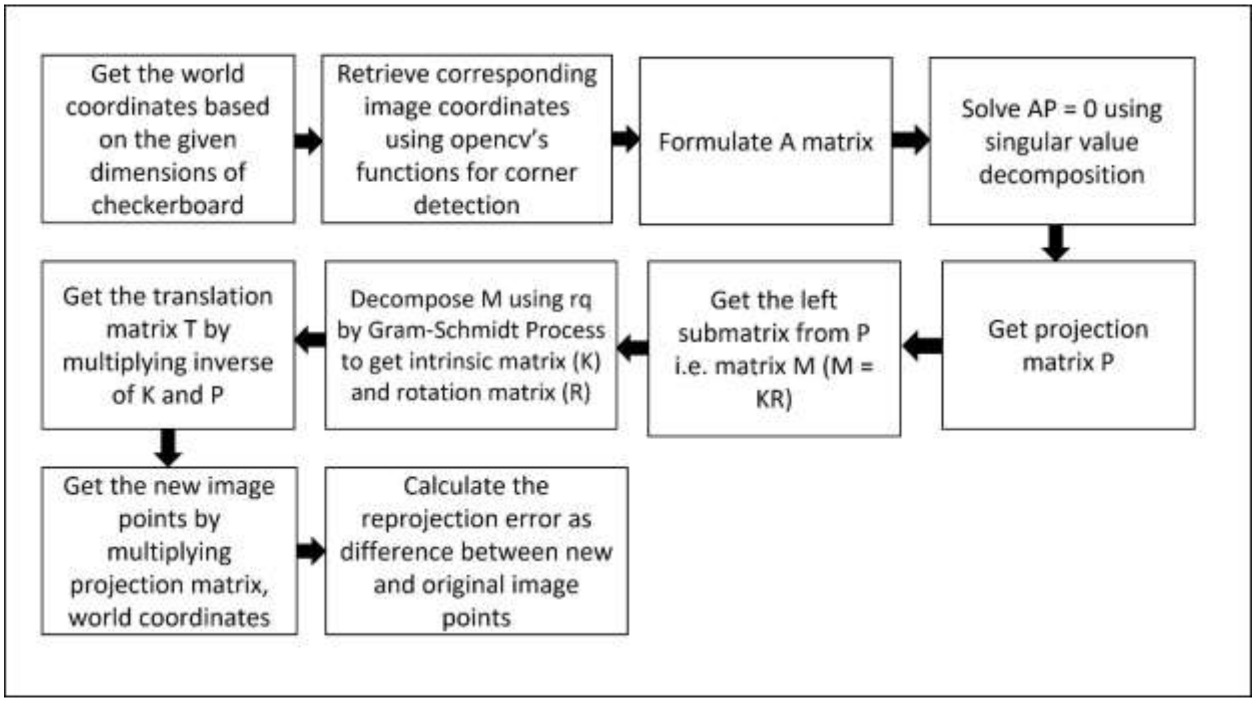
1. What is the minimum number of matching points to solve this mathematically?

Ans: We need minimum 6 number of matching points to solve this mathematically. Projection matrix P has 12 unknowns (11 neglecting the scaling). Each matching point gives two equations (for x coordinates and y coordinates) when we need to solve AP = 0. So, we need at least 6 matching points which gives a system of 12 linearly independent equations which is enough to solve for 12 unknowns.

However, having a minimum number of matching points may result in poor estimation of projection matrix and intrinsic matrix. Depending upon the specific application and quality of the data, a higher number of matching points maybe be required.

2. What is the pipeline or the block diagram that needs to be done in order to calibrate this camera given the image above.

Ans:



Get the world coordinates based on the given dimensions of checkerboard
Retrieve corresponding image coordinates using opencv's functions for corner detection
Formulate A matrix
Solve AP = 0 using singular value decomposition
Get projection matrix P
Get the left submatrix from P i.e. matrix M (M = KR)
Decompose M using rq by Gram-Schmidt Process to get intrinsic matrix (K) and rotation matrix (R)
Get the translation matrix T by multiplying inverse of K and P
Get the new image points by multiplying projection matrix, world coordinates
Calculate the reprojection error as difference between new and original image points

Computer Vision Pipeline

Explanation:

Define the arrays for the image coordinates (x and y) and corresponding world coordinates (X, Y, and Z) of a set of 3D points. These points are used to calibrate the camera and estimate its intrinsic and extrinsic parameters. Initialize the matrix A, which will store the coefficients of the linear system of equations used in the DLT method. Matrix A has dimensions of 2 * number of points by 12, where each row represents two equations (one for each image coordinate) for a single 3D point. The coefficients of the equations are computed and stored in A using a loop over all the points. Perform Singular Value Decomposition (SVD) on matrix A to obtain the singular vectors and values. The last row of the right singular vector matrix Vt corresponds to the null space of A, which represents the camera projection matrix P after reshaping into a 3x4 matrix. The intrinsic matrix M is extracted from P by taking the first three columns. The intrinsic matrix M is then factorized into an upper-triangular matrix R and an orthonormal matrix Q using the RQ decomposition. The intrinsic matrix R is normalized by dividing by its last element to ensure that its last element is equal to 1. The inverse of the rotation matrix R is computed and used to estimate the translation vector T by multiplying it with the projection matrix P. Compute the reprojection error for each point by projecting the 3D world coordinates back to the image plane using the estimated camera parameters and computing the Euclidean distance between the projected image coordinates and the actual image coordinates.

3. First write down the mathematical formation for your answer including steps that need to be done to find the intrinsic matrix K.

Ans:

i. Import the necessary libraries i.e., numpy and scipy.

ii. Given 2D and 3D matching points we can write as follows.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Where u, v is the image pixel coordinates and X, Y, Z are the world coordinates. And M is the projection matrix which projects world coordinates to image coordinates. Our task is to find a projection matrix.

iii. We can set up a system of linear equations to find the least square solution for the elements of the projection matrix. Therefore, we can write.

$$su = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$sv = m_{21}X + m_{22}Y + m_{23}Z + m_{24}$$

$$s = m_{31}X + m_{32}Y + m_{33}Z + m_{34}$$

We know that,

$$u = \frac{su}{s} \text{ and } v = \frac{sv}{s}$$

Therefore,

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

iv. Projection matrix is only defined up to a scale and therefore an infinite number of solutions are possible for these equations. Therefore, we can set the last element as 1 and then find the other elements.

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + 1}$$

$$(m_{31}X + m_{32}Y + m_{33}Z + 1)u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$(m_{31}X + m_{32}Y + m_{33}Z)u + u = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$

$$u = m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31}Xu - m_{32}Yu - m_{33}Zu$$

$$v = m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31}Xv - m_{32}Yv - m_{33}Zv$$

v. We can now formulate the equation AP = 0 as follows

$$\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -Xu & -Yu & -Zu \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -Xv & -Yv & -Zv \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{pmatrix} = 0$$

vi. We can write this for n number of matching points as follows:

$$A * \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ \cdot \\ \cdot \\ \cdot \\ m_{32} \\ m_{33} \end{pmatrix} = 0$$

vii. We can solve AP = 0 using SVD, the projection matrix P will be given as the last column of the Vt (transpose of unitary matrix U). Reshape the column vector into 3 cross 4 matrix P.

viii. We have to decompose the matrix P into intrinsic matrix K and rotation matrix R. Get the left submatrix M of projection matrix P by taking the first three columns of it.

ix. To decompose the M matrix, we can use QR factorization. Any non-singular square matrix M can be decomposed into the product of an upper triangular matrix K and orthogonal matrix R. Such that
M = KR

x. We can use Gram-Schmidt Process for this which is given as follows:

$$
\begin{aligned}
\mathbf{u}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{a}_2 - \mathrm{proj}_{\mathbf{u}_1}\,\mathbf{a}_2, & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{a}_3 - \mathrm{proj}_{\mathbf{u}_1}\,\mathbf{a}_3 - \mathrm{proj}_{\mathbf{u}_2}\,\mathbf{a}_3, & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{a}_k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}_j}\,\mathbf{a}_k, & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}
\end{aligned}
$$

xi. The rotation matrix can be given as follows.
R = [e1 e2 e3]
K = M R^{-1}

xii. In the code I have directly used the qr function of scipy library to find K and R.

xiii. We must normalize the intrinsic matrix for that we divide all the elements by the third row, third column element.

xiv. After finding K we can find the translation vector as follows:
Translation = K_inv @ projection_matrix

xv. The translation vector will be the last column of this multiplication.

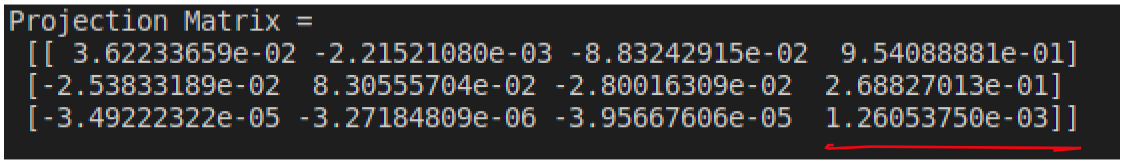
xvi. To find the reprojection error- 3D point in the world coordinates is represented as a homogeneous vector [X, Y, Z, 1], where (X, Y, Z) are the 3D coordinates of the point, and the trailing 1 represents the homogeneous coordinate. The projection of this point onto the 2D image plane can be calculated by multiplying it with the projection matrix P, resulting in a new homogeneous vector [x', y', w'], where (x', y') are the image coordinates of the projected point, and w' is the scaling factor.

xvii. We need to convert this to homogeneous coordinates by dividing all the elements by w'.

xviii. The reprojection error is then calculated as the Euclidean distance between the observed image coordinates (x, y) and the calculated image coordinates (x', y').

xix. The mean of reprojection is calculated by taking mean of the list containing individual errors.

4. Find the P matrix.

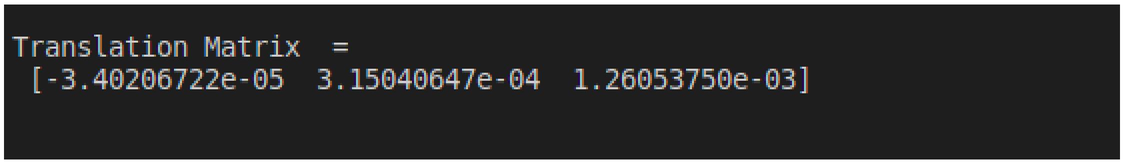


```
Projection Matrix =
 [[ 3.62233659e-02 -2.21521080e-03 -8.83242915e-02  9.54088881e-01]
 [-2.53833189e-02  8.30555704e-02 -2.80016309e-02  2.68827013e-01]
 [-3.49222322e-05 -3.27184809e-06 -3.95667606e-05  1.26053750e-03]]
```

Normalize

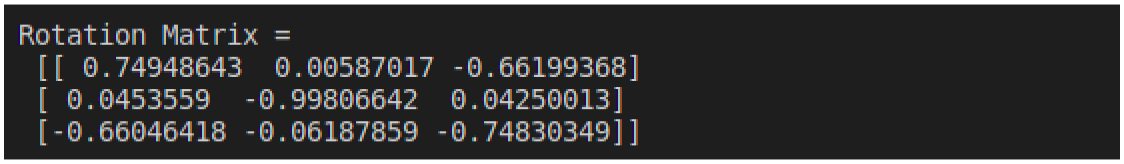
Projection matrix calculated from the code.

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.



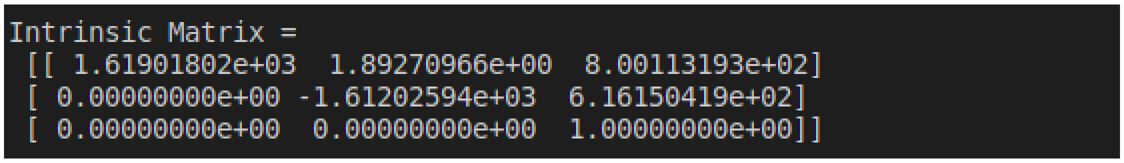
```
Translation Matrix  =
 [-3.40206722e-05  3.15040647e-04  1.26053750e-03]
```

Translational Matrix



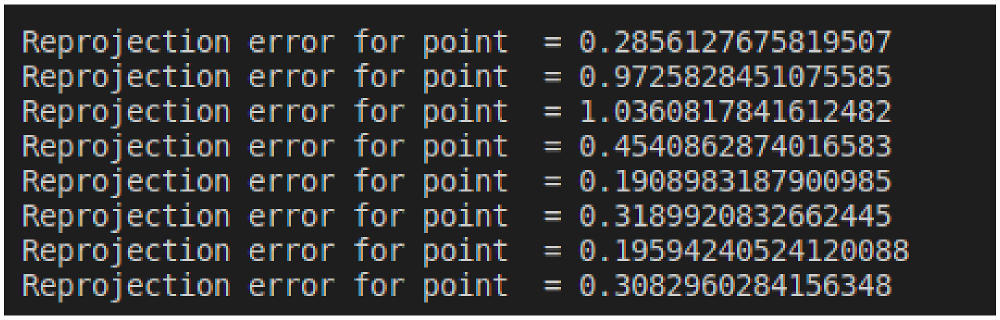
```
Rotation Matrix =
 [[ 0.74948643  0.00587017 -0.66199368]
 [ 0.0453559  -0.99806642  0.04250013]
 [-0.66046418 -0.06187859 -0.74830349]]
```

Rotation Matrix



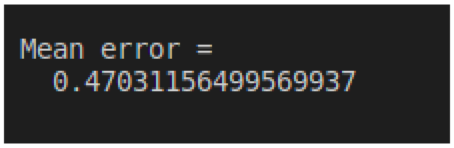
```
Intrinsic Matrix =
 [[ 1.61901802e+03  1.89270966e+00  8.00113193e+02]
 [ 0.00000000e+00 -1.61202594e+03  6.16150419e+02]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

Intrinsic Matrix



```
Reprojection error for point  = 0.2856127675819507
Reprojection error for point  = 0.9725828451075585
Reprojection error for point  = 1.0360817841612482
Reprojection error for point  = 0.4540862874016583
Reprojection error for point  = 0.1908983187900985
Reprojection error for point  = 0.3189920832662445
Reprojection error for point  = 0.19594240524120088
Reprojection error for point  = 0.3082960284156348
```

Reprojection error for individual matching points

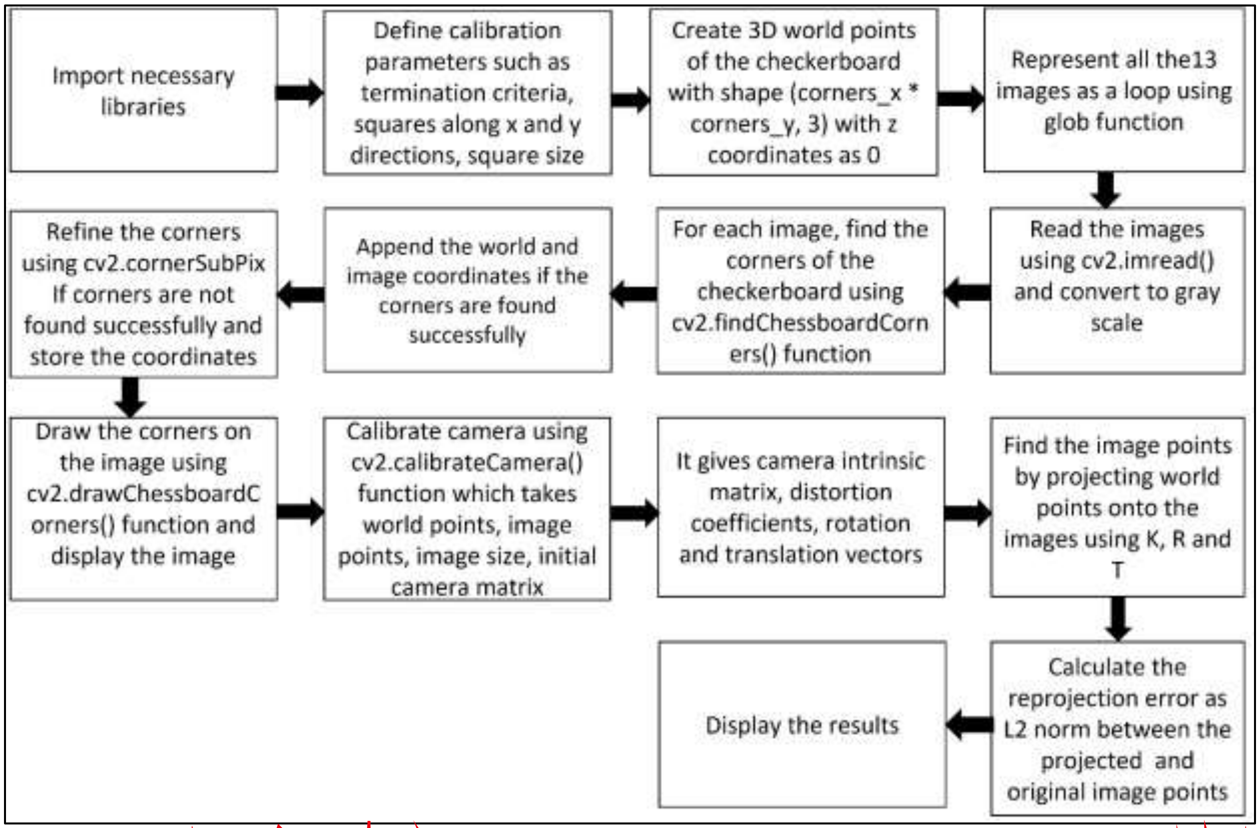


```
Mean error =
  0.47031156499569937
```

Mean reprojection error.

Question 2.

1. Computer vision pipeline

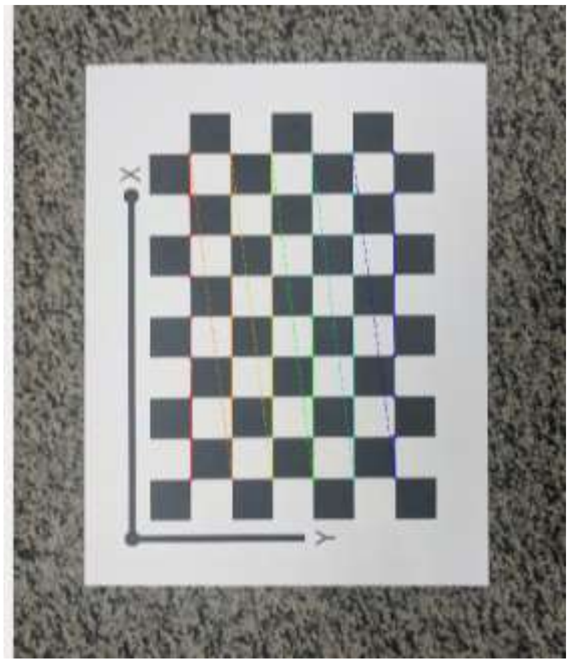

Import necessary libraries
Define calibration parameters such as termination criteria, squares along x and y directions, square size
Create 3D world points of the checkerboard with shape (corners_x * corners_y, 3) with z coordinates as 0
Represent all the13 images as a loop using glob function
Read the images using cv2.imread() and convert to gray scale
For each image, find the corners of the checkerboard using cv2.findChessboardCorners() function
Append the world and image coordinates if the corners are found successfully
Refine the corners using cv2.cornerSubPix If corners are not found successfully and store the coordinates
Draw the corners on the image using cv2.drawChessboardCorners() function and display the image
Calibrate camera using cv2.calibrateCamera() function which takes world points, image points, image size, initial camera matrix
It gives camera intrinsic matrix, distortion coefficients, rotation and translation vectors
Find the image points by projecting world points onto the images using K, R and T
Calculate the reprojection error as L2 norm between the projected and original image points
Display the results

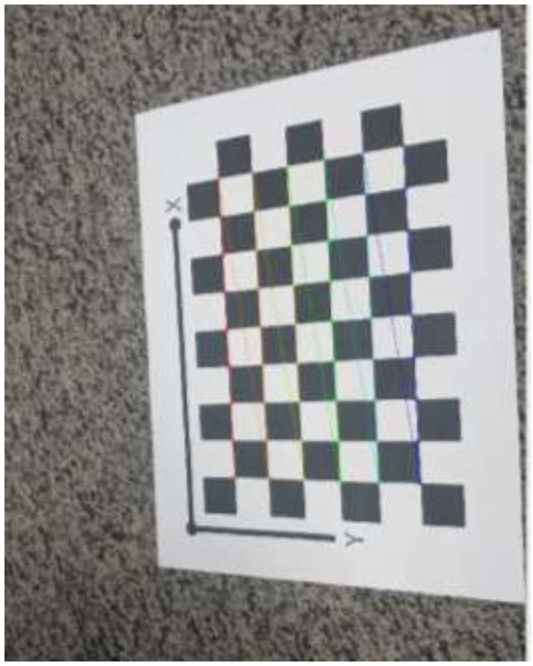
Instead of taking a screen shot, it would be great if you can include directly saved image for better resolution.

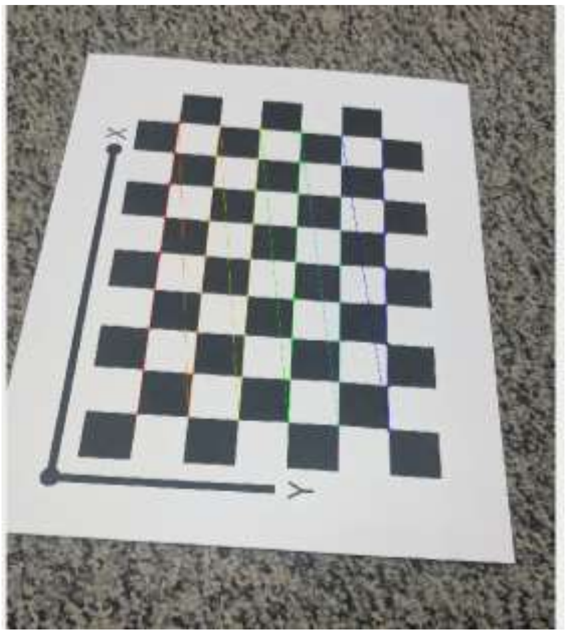
Explanation

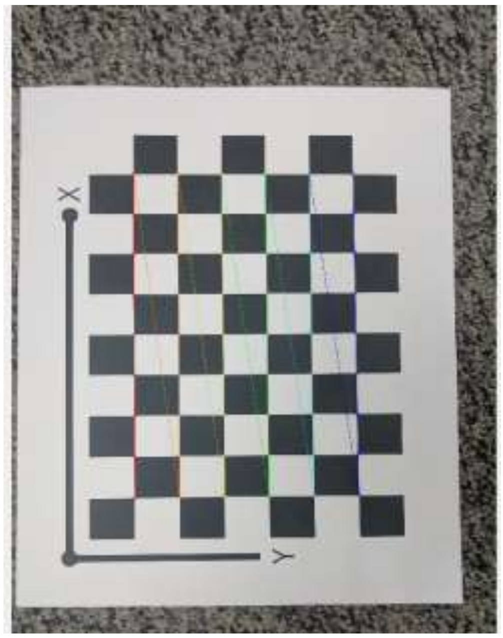
Import the necessary libraries. Define the calibration parameters such as termination criteria for corner refinement (criteria), the number of corners along x and y directions on the checkerboard pattern (corners_x and corners_y), and the size of each square on the checkerboard pattern in millimeters (square_size). Generate the 3D world coordinates of the checkerboard corners using np.mgrid function, which creates a grid of points, and then multiplying the grid with the square size to obtain the world coordinates of the corners. Loos through a list of image file paths obtained using glob.glob function. For each image, read the image, convert it to grayscale, and use the cv.findChessboardCorners function to find the corners of the checkerboard pattern in the image. If corners are found, append the world coordinates of the checkerboard corners to checker_board_world_points list, refine the corner positions using cv.cornerSubPix function, draw the corners on the image using cv.drawChessboardCorners function and display the image. After obtaining the checkerboard points from all the images, use the cv.calibrateCamera function to estimate the camera's intrinsic and extrinsic parameters. It takes the lists of world coordinates and image coordinates of the checkerboard corners, the size of the grayscale image, and optional initial values for camera matrix and distortion coefficients as inputs, and returns the estimated camera matrix, distortion coefficients, rotation vectors, and translation vectors. Calculate the reprojection error for each image, which is the difference between the observed image coordinates and the projected image coordinates of the checkerboard corners using the estimated camera parameters. Use the cv.projectPoints function to project the world coordinates of the checkerboard corners onto the image plane, and then calculate the L2 norm of the difference between the observed and projected image coordinates.

Images with detected corners

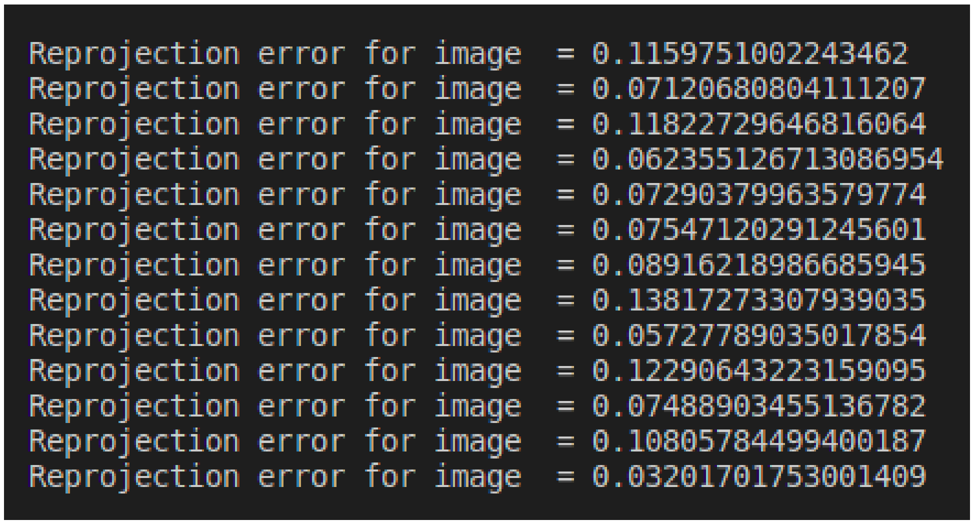






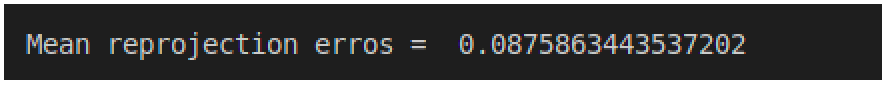


2. Compute the Reprojection Error for each image using built-in functions in OpenCV.



```
Reprojection error for image  = 0.1159751002243462
Reprojection error for image  = 0.07120680804111207
Reprojection error for image  = 0.11822729646816064
Reprojection error for image  = 0.062355126713086954
Reprojection error for image  = 0.07290379963579774
Reprojection error for image  = 0.07547120291245601
Reprojection error for image  = 0.08916218986685945
Reprojection error for image  = 0.13817273307939035
Reprojection error for image  = 0.05727789035017854
Reprojection error for image  = 0.12290643223159095
Reprojection error for image  = 0.07488903455136782
Reprojection error for image  = 0.10805784499400187
Reprojection error for image  = 0.03201701753001409
```

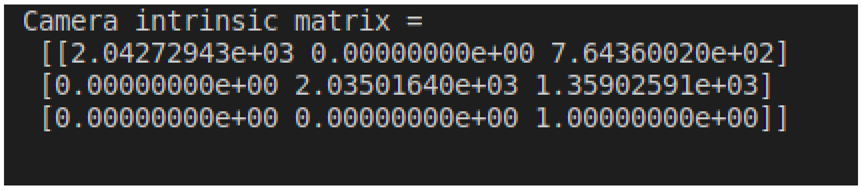
Reprojection error for individual images



```
Mean reprojection erros =  0.0875863443537202
```

Mean reprojection error.

3. Compute the K matrix.



```
Camera intrinsic matrix =
 [[2.04272943e+03 0.00000000e+00 7.64360020e+02]
 [0.00000000e+00 2.03501640e+03 1.35902591e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

Intrinsic Matrix

4. How can we improve the accuracy of K matrix?

Ans

i. The quality of the intrinsic matrix K can be increased by using high quality calibration patterns. It depends on how accurate the world coordinates of the checkboard corners are. Therefore, a high calibration pattern with precise dimensions is needed. We should ensure that the corners and square size of the target checkerboard have been detected precisely.
ii. The accuracy of the K matrix also depends on the diversity of the calibration images and no of calibration images. It is important to capture the images of the checkerboard from different angles, poses, focal lengths and distortions. This will help in getting a more accurate K matrix.
iii. Using a larger number of sample images will improve the accuracy of intrinsic matrix. It will provide better coverage of the parameter space and will reduce errors.

iv. We can improve the accuracy of corner detection by trying different corner detection algorithms. Estimating the correct corner images coordinates will help to get the accurate K matrix.

v. We can also consider the distortion to the images by determining the distortion coefficients along with the intrinsic parameters.

vi. We can try reducing the reprojection error by performing error analysis to know how well the calculated parameters fit the calibration images.