ENPM673 Project 3

Ankur Chavan (achavan1@umd.edu)

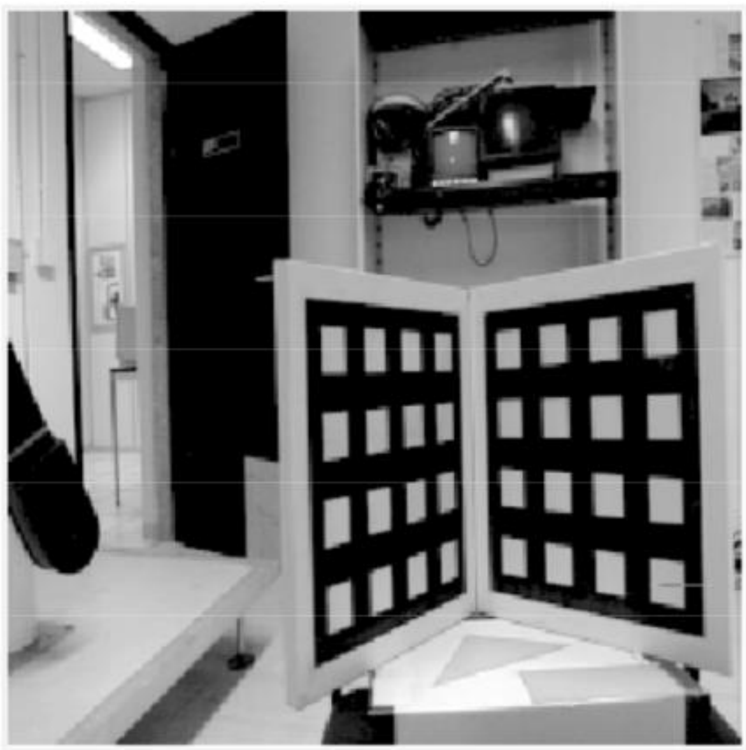
Problem 1:

Calibrate the camera (Find the intrinsic matrix K)

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

To find the intrinsic matrix K and calibrate a camera, we need to take pictures of a calibration pattern with known 3D points and their corresponding 2D image points. The minimum number of matching points required to solve for K depends on the calibration pattern used and the complexity of the camera model.

In general, we need at least 6 matching 3D and 2D points that are not on the same plane to solve for the intrinsic matrix. However, if the camera has lens distortion or a more complex camera model is used, more matching points are needed for accurate calibration. It is recommended to use more than the minimum number of matching points for better calibration accuracy and robustness.



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

Pipeline for Camera Calibration:

STEP 1:
- Capture an image of an object with known geometry, in this case we have the above given image

STEP 2:
- Identify the 3D world points in world frame and corrensponding 2D image points in image plane.

STEP 3:
- For each corresponding point we have the equation "Image Point = Projection Matrix x World Point"

STEP 4:
- Using the equation in STEP 3 and rearranging, we get the measurement matrix to solve for elements of Projection Matrix. Using SVD to solve the equation, we find the Projection Matrix.

STEP 5:
- We know that Projection matrix is product of Intrinsic matrix and Extrinsic matrix. Also, the first 3 left columns of projection matrix is product of Calibration Matrix "K" and Rotation Matrix "R". Since the calibration matrix K is upper triangular matrix and rotational matrix is orthonormal, it is possible to uniquely decouple K and R from their product using QR or RQ factorization.

STEP 6:
- Find the Translation Vector by multiplying the inverse of the Calibration Matrix with the last column of the Projection Matrix.

In this manner by using linear algebra, we can compute the intrinsic and extrinsic parameters of the camera, which results in successful camera calibration.

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

STEP 1: Capture an image of an object with known geometry, in this case we have the above given image

STEP 2: Identify the 3D world points in world frame and corrensponding 2D image points in image plane.

STEP 3: For each corresponding point we have the equation “Image Point = Projection Matrix x World Point”

Step 3: For each corresponding point i in scene and image:

$$\underbrace{\begin{bmatrix} u^{(i)} \\ v^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}} \equiv \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{\text{Unknown}} \underbrace{\begin{bmatrix} x_w^{(i)} \\ y_w^{(i)} \\ z_w^{(i)} \\ 1 \end{bmatrix}}_{\text{Known}}$$

STEP 4: Using the equation in STEP 3 and rearranging, we get the below matrix to solve for elements of Projection Matrix

Step 4: Rearranging the terms

$$\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & 0 & 0 & 0 & 0 & -u_1x_w^{(1)} & -u_1y_w^{(1)} & -u_1z_w^{(1)} & -u_1 \\ 0 & 0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & 1 & -v_1x_w^{(1)} & -v_1y_w^{(1)} & -v_1z_w^{(1)} & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & 0 & 0 & 0 & 0 & -u_ix_w^{(i)} & -u_iy_w^{(i)} & -u_iz_w^{(i)} & -u_i \\ 0 & 0 & 0 & 0 & x_w^{(i)} & y_w^{(i)} & z_w^{(i)} & 1 & -v_ix_w^{(i)} & -v_iy_w^{(i)} & -v_iz_w^{(i)} & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & 0 & 0 & 0 & 0 & -u_nx_w^{(n)} & -u_ny_w^{(n)} & -u_nz_w^{(n)} & -u_n \\ 0 & 0 & 0 & 0 & x_w^{(n)} & y_w^{(n)} & z_w^{(n)} & 1 & -v_nx_w^{(n)} & -v_ny_w^{(n)} & -v_nz_w^{(n)} & -v_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

STEP 5: The Projection Matrix can be obtained by finding the null space of the measurement matrix. The null space can be computed using Singular Value Decomposition (SVD).

STEP 6: We know that Projection matrix is product of Intrinsic matrix and Extrinsic matrix.

We know that:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{M_{int}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{ext}}$$

STEP 7: Also, the first 3 left columns of projection matrix is product of Calibration Matrix "K" and Rotation Matrix "R". Since the calibration matrix K is upper triangular matrix and rotational matrix is orthonormal, it is possible to uniquely decouple K and R from their product using QR or RQ factorization.

That is:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = KR$$

That is:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \boxed{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}} = KR$$

1 Gram-Schmidt process

Consider the GramSchmidt procedure, with the vectors to be considered in the process as columns of the matrix A. That is,

$$A = \Big[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n \Big].$$

Then,

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{a}_1, \qquad \mathbf{e}_1 = \frac{\mathbf{u}_1}{||\mathbf{u}_1||}, \\ \mathbf{u}_2 &= \mathbf{a}_2 - (\mathbf{a}_2 \cdot \mathbf{e}_1)\mathbf{e}_1, \qquad \mathbf{e}_2 = \frac{\mathbf{u}_2}{||\mathbf{u}_2||}. \\ \mathbf{u}_{k+1} &= \mathbf{a}_{k+1} - (\mathbf{a}_{k+1} \cdot \mathbf{e}_1)\mathbf{e}_1 - \cdots - (\mathbf{a}_{k+1} \cdot \mathbf{e}_k)\mathbf{e}_k, \qquad \mathbf{e}_{k+1} = \frac{\mathbf{u}_{k+1}}{||\mathbf{u}_{k+1}||}. \end{aligned}$$

Note that $||\cdot||$ is the L_2 norm.

1.1 QR Factorization

The resulting QR factorization is

$$A = \Big[\mathbf{a}_1 \mid \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n \Big] = \Big[\mathbf{e}_1 \mid \mathbf{e}_2 \mid \cdots \mid \mathbf{e}_n \Big] \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{e}_1 & \mathbf{a}_2 \cdot \mathbf{e}_1 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_1 \\ 0 & \mathbf{a}_2 \cdot \mathbf{e}_2 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_n \end{bmatrix} = QR.$$

Note that once we find $\mathbf{e}_1, \ldots, \mathbf{e}_n$, it is not hard to write the QR factorization.

STEP 8: Find the Translation Vector by multiplying the inverse of the Calibration Matrix with the last column of the Projection Matrix.

We know that:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \boxed{p_{14}} \\ p_{21} & p_{22} & p_{23} & \boxed{p_{24}} \\ p_{31} & p_{32} & p_{33} & \boxed{p_{34}} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

That is:

$$\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = K\mathbf{t}$$

Therefore:

$$\boxed{\mathbf{t} = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}}$$

Image Sources:

https://www.math.ucla.edu/~yanovsky/Teaching/Math151B/handouts/GramSchmidt.pdf

https://www.youtube.com/watch?v=qByYk6JggQU&list=PL2zRqk16wsdoCCLpou-dGo7QQNks1Ppzo&index=2

4. Find the P matrix: Given below is the output from the code. Run the code to see the matrix.

```
-------------------------------------------------------------------------
Projection Matrix from World Frame to Image Frame:
-------------------------------------------------------------------------
[[ 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]]
-------------------------------------------------------------------------
```

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.

Given below is the Intrinsic Matrix, Rotation Matrix, and Translation Matrix as output from the code.

```
------------------------------------------------------------
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]]
------------------------------------------------------------
Rotation Matrix:
------------------------------------------------------------
[[ 0.74948643  0.00587017 -0.66199368]
 [ 0.0453559  -0.99806642  0.04250013]
 [-0.66046418 -0.06187859 -0.74830349]]
------------------------------------------------------------
Translation Vector:
------------------------------------------------------------
[-3.40206722e-05  3.15040647e-04  1.26053750e-03]
------------------------------------------------------------
```

Reprojection error for individual points and mean reprojection error:

```
------------------------------------------------------------
The reprojection error for individual Points
------------------------------------------------------------
The reprojection error for point 1 is 0.28561276758292053
------------------------------------------------------------
The reprojection error for point 2 is 0.9725828450556381
------------------------------------------------------------
The reprojection error for point 3 is 1.0360817843232593
------------------------------------------------------------
The reprojection error for point 4 is 0.45408628727914896
------------------------------------------------------------
The reprojection error for point 5 is 0.19089831863054724
------------------------------------------------------------
The reprojection error for point 6 is 0.31899208342793967
------------------------------------------------------------
The reprojection error for point 7 is 0.19594240508863206
------------------------------------------------------------
The reprojection error for point 8 is 0.30829602814307683
------------------------------------------------------------
The mean reprojection error is 0.47031156494139537
------------------------------------------------------------
```

The mathematical equation used to calculate the reprojection error for each point can be expressed as:

$$\text{reprojection\_error}_i = \sqrt{(u_i - \grave{u}_i')^2 + (v_i - \hat{v}_i')^2}$$

where,

u_i is the actual x-coordinate of the i-th point in pixels.

v_i is the actual y-coordinate of the i-th point in pixels.

$\grave{u}_i'$ is the estimated x-coordinate of the i-th point in pixels.

$\hat{v}_i'$ is the estimated y-coordinate of the i-th point in pixels.

Problem 2:

Find the checkerboard corners using any corner detection method (inbuilt OpenCV functions such as findChessboardCorners are allowed). Use these corners to estimate the Projection matrix P. Decompose the P matrix into the Translation, Rotation, and Intrinsic matrix using the Gram–Schmidt process and compute the reprojection error for each image.

Solution:

Process or Pipeline:

1. Import necessary libraries and modules. (Numpy, cv2, glob, os, pprint)
2. Define termination criteria for the iterative algorithm.
3. Create an array of object points that correspond to the location of the corners of a chessboard calibration pattern in 3D space.
4. Create empty lists to store the object points and image points.
5. Define the path where the calibration images are stored.
6. Get the file paths for all the images in the specified directory using the os library.
7. Loop through each image in the directory.
8. Read the image using cv2.imread().
9. Convert the image to grayscale using cv2.cvtColor().
10. Find the chessboard corners in the grayscale image using cv2.findChessboardCorners().
11. If the corners are found, add the object points and image points to their respective lists after refining the corners using cv2.cornerSubPix().
12. Draw the corners on the image using cv2.drawChessboardCorners() and
13. Rescale the display window and show the image using cv2.imshow() and cv2.waitKey().
14. Repeat steps 7-13 for all images in the directory.
15. Use the cv2.calibrateCamera() function to obtain the camera matrix, distortion coefficients, and rotation and translation vectors.
16. Loop through each image again and calculate the reprojection error using cv2.projectPoints() to obtain the final image points, and then calculate the Euclidean distance between the image points and the final image points using np.sqrt() and np.sum().
17. Compute the mean reprojection error by dividing the total error by the number of images.
18. Print the mean reprojection error.

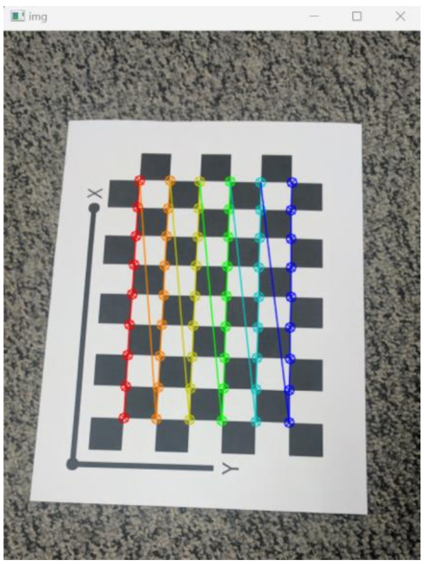
Termination criteria explanation:

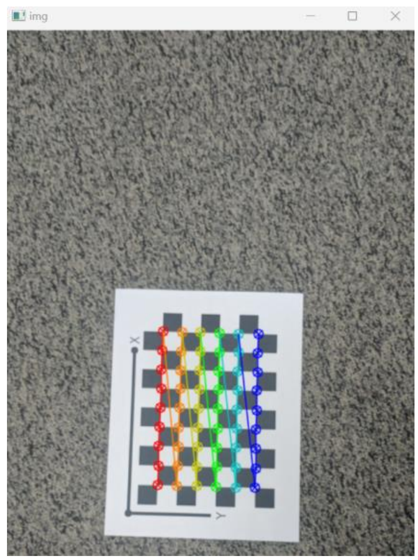
cv.TERM_CRITERIA_EPS is a termination criterion based on the required accuracy (epsilon) achieved during the optimization process. It indicates that the iterative optimization algorithm should stop when the change in the estimated parameters falls below a certain threshold.

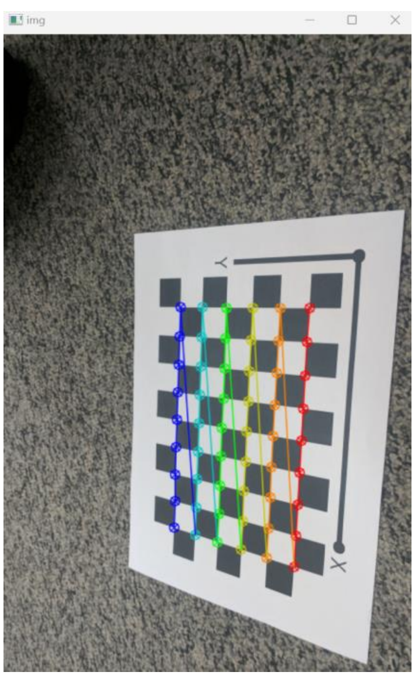
cv.TERM_CRITERIA_MAX_ITER is a termination criterion based on the maximum number of iterations (max_iter) allowed during the optimization process. It indicates that the iterative optimization algorithm should stop after a specified number of iterations have been performed, regardless of the achieved accuracy.

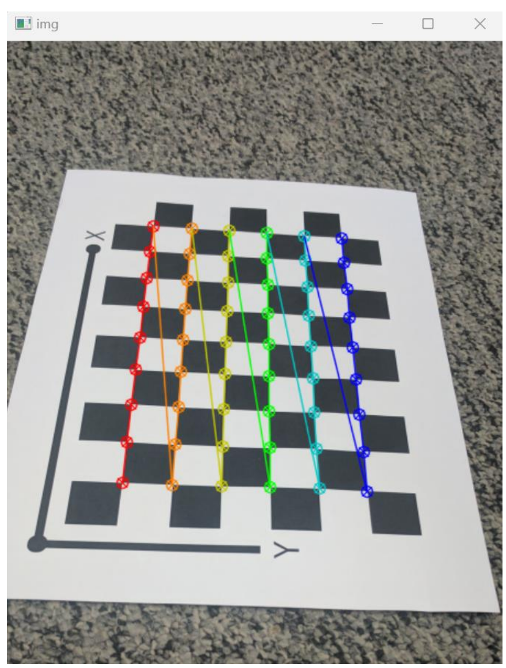
So, **term_criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)** means that the iterative optimization algorithm should stop either when the required accuracy (epsilon) falls below 0.001 or after 30 iterations, whichever comes first.

Results: Detected checkerboard corners for 4 images (You can find this for all images in code output)









Reprojection Error for each image:

```
------------------------------------------------------------
Reprojection error for individual Images:
------------------------------------------------------------
Reprojection error for image  1 is  0.07330641481611463
------------------------------------------------------------
Reprojection error for image  2 is  0.08908931414286296
------------------------------------------------------------
Reprojection error for image  3 is  0.11592670723243996
------------------------------------------------------------
Reprojection error for image  4 is  0.13740256980613427
------------------------------------------------------------
Reprojection error for image  5 is  0.0626768050370393
------------------------------------------------------------
Reprojection error for image  6 is  0.07569479942321777
------------------------------------------------------------
Reprojection error for image  7 is  0.03103626436657376
------------------------------------------------------------
Reprojection error for image  8 is  0.058159280706335
------------------------------------------------------------
Reprojection error for image  9 is  0.0726931095123291
------------------------------------------------------------
Reprojection error for image  10 is  0.07489343925758644
------------------------------------------------------------
Reprojection error for image  11 is  0.10864734649658203
------------------------------------------------------------
Reprojection error for image  12 is  0.12325005178098325
------------------------------------------------------------
Reprojection error for image  13 is  0.11889099191736292
------------------------------------------------------------
Mean Reprojection error:  0.0878205457304278
------------------------------------------------------------
```

K Matrix:

```
--------------------------------------------------
K Matrix:
--------------------------------------------------
[[2.04284559e+03 0.00000000e+00 7.64107788e+02]
 [0.00000000e+00 2.03517002e+03 1.35884993e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
--------------------------------------------------
```

- **How can we improve the accuracy of the K matrix?**

The intrinsic camera parameters, which are represented by the K matrix, are typically determined through a calibration process. In the provided code, the K matrix is obtained using the cv.calibrateCamera() function, which estimates the camera matrix, distortion coefficients, and rotation and translation vectors based on the input object points and their corresponding image points.

To improve the accuracy of the K matrix estimation, you can take the following steps:

1. Increase the number of images used for calibration: Using more images can help improve the accuracy of the calibration process by providing more data points for estimating the camera parameters.
2. Capture a variety of images: The calibration should be performed using images captured from different positions, orientations, and distances to ensure that the calibration is robust and can handle various scenarios.
3. Increase the number of chessboard corners: The accuracy of the calibration process can also be improved by increasing the number of corners on the calibration pattern, as this will provide more constraints on the camera parameters.
4. Improve corner detection: Ensuring that the corners are detected accurately is crucial to achieving accurate calibration. Using a sub-pixel corner detection algorithm can help improve accuracy.
5. Improve the quality of the calibration images: The quality of the images used for calibration can also impact the accuracy of the K matrix estimation. Capturing high-quality images with good lighting, sharp focus, and minimal noise can help improve the accuracy of the calibration process.
6. Optimize the termination criteria: The termination criteria used in the code determine the maximum number of iterations and the minimum value for convergence. Tweaking these parameters can help improve the accuracy of the calibration process.
7. Consider lens distortion: Most lenses have some form of distortion, which can affect the calibration results. Accounting for lens distortion can help achieve better calibration results.