INFOMCV Assignment 1
(Deepikaa Balaji, Nitikaa Rajesh, Olga Capmany Dalmas)

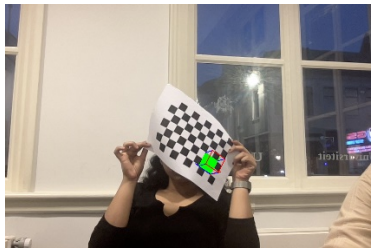
Calibration

(For the three runs, add (1) intrinsic camera matrix, and (2) test image with cube. Approx 1/2 page.)

Run 1:

$$K = \begin{bmatrix} 2733.91 & 0 & 1102.36 \\ 0 & 2714.9 & 1098.72 \\ 0 & 0 & 1 \end{bmatrix}$$

Image resolution = (1680, 2520)

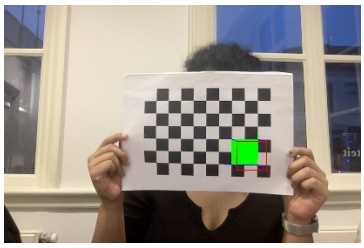


total error: 3.686550135280766

Run 2:

$$K = \begin{bmatrix} 1513.52 & 0 & 1111.66 \\ 0 & 1512.22 & 849.431 \\ 0 & 0 & 1 \end{bmatrix}$$

Image resolution = (1680, 2520)

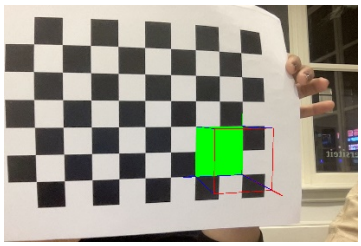


total error: 0.3211004005801239

Run 3:

$$K = \begin{bmatrix} 1492.52 & 0 & 1109.27 \\ 0 & 1494.14 & 857.587 \\ 0 & 0 & 1 \end{bmatrix}$$

Image resolution = (1680, 2520)



total error: 0.44103507988147833

Discussion

(Discuss briefly how the parameters of *K* change between runs. What can you say about the quality of each run. Motivate your answer. Approx. 1/4 page)

Intrinsic Camera Matrix: Run 1 involved manual annotation for 6 images, potentially leading to slight variations in the intrinsic camera matrix compared to fully automatic detection. Runs 2 and 3, with entirely automatic corner detection, likely offer more consistent and reliable intrinsic matrices

Image Resolution: The image resolution remains constant at (1680, 2520) pixels across all runs, indicating consistent camera setup and imaging conditions

Total Error: Run 1 exhibits the highest total error (3.69) due to manual annotation and interpolation, indicating lower calibration accuracy. Runs 2 and 3 show lower total errors (0.32 and 0.44, respectively), suggesting higher calibration accuracy with automatic corner detection

Quality Assessment: Run 1, with manual annotation, shows lower calibration quality compared to Runs 2 and 3, which feature automatic corner detection. Runs 2 and 3 demonstrate higher calibration accuracy, with Run 2 exhibiting the lowest total error

Choice tasks

1. **Produce a 3D plot with the locations of the camera relative to the chessboard when each of the training images was taken**

   To obtain the camera position, the rotation vectors (rvecs), translation vectors (tvecs) and object points (objpoints) obtained during training must be known. For each dataset corresponding to a training image, it converts the rotation vectors (rvecs) into a rotation matrix using the cv2.Rodrigues() function. It then calculates the camera position relative to the chessboard in the camera coordinate system using the formula camera_position = -R.T.dot(tvec.reshape(-1)).

2. **Real-time performance with webcam in online phase**
   We have implemented real-time performance using webcam by capturing frames and processing them on-the-fly. The program prompts the user to choose between webcam usage or providing an image directory. For webcam usage, it continuously captures frames until the user presses the 's' key to save a frame. It then detects chessboard corners, performs calibration, and visualizes the results, providing instant feedback to the user. This feature enhances user experience and facilitates quick calibration without the need for pre-captured images.

3. **Implement a way to enhance the input to reduce the number of input images that are not correctly processed by findChessboardCorners, for example by enhancing edges or getting rid of light reflections**

   We have also enhanced input images in the training code by applying histogram equalization to improve contrast and morphological gradient to enhance edges, reducing the likelihood of missed corner detections or false positives in the function”findChessboardCorners”. This preprocessing step helps mitigate issues caused by uneven lighting or reflections, leading to more accurate corner detection and fewer incorrectly processed images.