CAMERA CALIBRATION REPORT

Done by : Sai Nihar Tangadipally

Roll no : 21EE01012

Problem Statement : Implementation of camera calibration module: In this assignment, you are going to write a software module (C/Python) using OpenCV platform to extract the 5 intrinsic and 6 extrinsic parameters of the calibration matrix C. You can submit the assignment either in a group of 2 or individual. The submission file (a single zip) includes (i) a PDF file describing the steps including how the intrinsic and extrinsic parameters are estimated from a set of correspondence points (ii) the source code (iii) the readme file to execute the code (iv) the sample images used by your software.

Camera calibration involves estimating two sets of parameters:

Intrinsic Parameters (5): These define the internal characteristics of the camera, including:

- Focal length (fx, fy)
- Optical Point (cx,cy)
- Skew coefficient.

Extrinsic Parameters (6): These describe the camera's position and orientation in the world, consisting of:

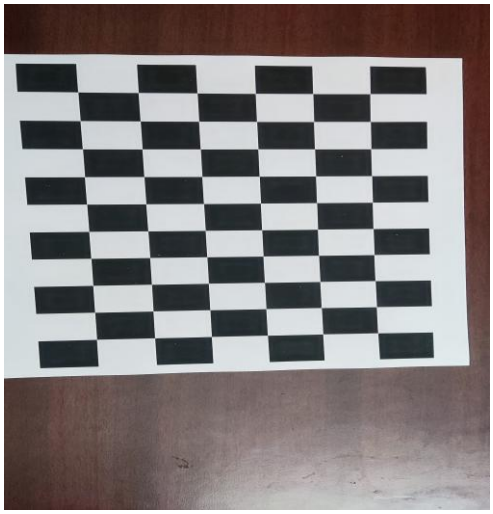
- Rotation vectors
- Translation vectors.

IMPLEMENTATION STEPS:

- I have chosen python language to perform this calibration.
- I took a 11x7 squared checkerboard printed on a A4 size sheet and captured 10 pictures of that sheet at different angles which I have included in the images folder.
- I have imported necessary libraries.
  - **cv2** (OpenCV): For image processing and camera calibration.
  - **numpy**: For matrix operations and numerical computations.
  - **glob**: To find all image files in a folder.
  - **os**: To handle file paths.
- I went on to specify inner corners of the checkerboard (10 columns and 6 rows). Also defined the termination criteria for corner refinement using **cv2.cornerSubPix().**
- Then I prepared 3D object points. Here, for simplicity I imagined the checkerboard to be on the x-y plane so that my z-coordinate is zero for each corner point.
- I initialized the storages for object and image points followed by loading each input image from the images folder (used **cv2.imread()** function to do so).
- By converting each image to **grayscale** using **cv2.cvtColor()** , I ensured that corner detection accuracy is improved.

- I used **cv2.findChessboardCorners()** to detect the **checkerboard pattern** i.e., to find the corners.
- Leveraged **cv2.cornerSubPix()** to refine the **corner positions** to sub-pixel accuracy if corners are found in that particular image.
- Finally performed calibration using **cv2.calibrateCamera(),** which returns:
    - mtx: **Camera matrix** (intrinsic parameters).
    - dist: **Distortion coefficients** (radial & tangential distortion).
    - rvecs: **Rotation vectors** (extrinsic parameters).
    - tvecs: **Translation vectors** (extrinsic parameters).
    - ret: **Reprojection error** (how well the calibration fits the points).
- Then I have printed all the 5 intrinsic parameters which are same for each image as these are the internal characteristics of the camera.
- At the end I ran a loop through which I have printed each of the 6 extrinsic parameter for every input image.

Sample image :



Results :

== Camera Calibration Results ==

Intrinsic Parameters:

1. Focal Length (fx): 1439.1611236060803
2. Focal Length (fy): 1440.2975873800679
3. Principal Point (cx): 780.3524237852556
4. Principal Point (cy): 1688.9335563931452
5. Skew Coefficient: 0.0

Extrinsic Parameters (Rotation & Translation Vectors):

Image 1:

1. Rotation X (r1): 0.05184214456673953
2. Rotation Y (r2): -0.040464196511214204
3. Rotation Z (r3): -1.4006446678234032
4. Translation X (t1): -2.8059622161322677
5. Translation Y (t2): 4.575838556047263
6. Translation Z (t3): 7.830715312770404

Image 2:

1. Rotation X (r1): 0.051150610376787746
2. Rotation Y (r2): -0.06880952078601474
3. Rotation Z (r3): -1.1564232323371422
4. Translation X (t1): -3.589717969670935
5. Translation Y (t2): 4.349910759894034
6. Translation Z (t3): 8.850437224219045

Image 3:

1. Rotation X (r1): 0.05427271491198009
2. Rotation Y (r2): -0.06832049157926581
3. Rotation Z (r3): -1.1531508726396553
4. Translation X (t1): -3.4420753841897267
5. Translation Y (t2): 4.408099332206761

6. Translation Z (t3): 9.065627913576904

Image 4:

1. Rotation X (r1): -0.028568075863365704
2. Rotation Y (r2): -0.0674051323714803
3. Rotation Z (r3): 1.340952871460649
4. Translation X (t1): 2.098661795008253
5. Translation Y (t2): -3.344996488531411
6. Translation Z (t3): 9.18818135487621

Image 5:

1. Rotation X (r1): -0.03407456210335152
2. Rotation Y (r2): -0.049269188774847185
3. Rotation Z (r3): 1.5459366041748785
4. Translation X (t1): 2.707704050973575
5. Translation Y (t2): -4.579296582723349
6. Translation Z (t3): 6.886991736985498

Image 6:

1. Rotation X (r1): 0.3402854578649618
2. Rotation Y (r2): 0.23802981666443448
3. Rotation Z (r3): 1.5522315142022283
4. Translation X (t1): 2.0743193349381466
5. Translation Y (t2): -6.661957818864485
6. Translation Z (t3): 9.639514867119003

Image 7:

1. Rotation X (r1): -0.03959010099397494
2. Rotation Y (r2): -0.13829499401166206
3. Rotation Z (r3): -1.3485927150276076
4. Translation X (t1): -3.1249567778766627
5. Translation Y (t2): 0.9857510784303897
6. Translation Z (t3): 9.932887752758305

Image 8:

1. Rotation X (r1): -0.02898583464740855

2. Rotation Y (r2): 0.02208087016439394
3. Rotation Z (r3): 1.5628595930137765
4. Translation X (t1): 3.2069410236659146
5. Translation Y (t2): -4.92593381454482
6. Translation Z (t3): 7.31489691218186

Image 9:

1. Rotation X (r1): -0.0474586148843221
2. Rotation Y (r2): 0.015269807884656083
3. Rotation Z (r3): 1.544144623174474
4. Translation X (t1): 2.85096566712322
5. Translation Y (t2): -0.28923406214967434
6. Translation Z (t3): 7.650625990849947

Image 10:

1. Rotation X (r1): -0.2584380498543125
2. Rotation Y (r2): -0.2981182962078888
3. Rotation Z (r3): -1.5348723083801008
4. Translation X (t1): -2.5493559423967405
5. Translation Y (t2): 0.8685761262771704
6. Translation Z (t3): 7.442281463111041