Project 1: Report

**introduction and a brief description of the theory and the algorithms**

Camera calibration is a fundamental task in computer vision and photogrammetry, essential for applications such as 3D reconstruction, augmented reality, and object tracking. By accurately estimating intrinsic and extrinsic camera parameters, we can correct distortion, determine the relationship between image pixels and real-world coordinates, and enable precise geometric transformations. Camera calibration involves determining intrinsic parameters, which characterize the internal geometry of the camera (e.g., focal length, optical center), and extrinsic parameters, which describe the camera's position and orientation in the world coordinate system.

The calibration process typically relies on the pinhole camera model, which assumes that light rays passing through the camera aperture project onto an image plane. The intrinsic parameters include the focal length (fx, fy), optical center (ux, vy), and skew (s), represented by the intrinsic matrix K. The extrinsic parameters consist of the rotation matrix R and translation vector t, which transform world coordinates to camera coordinates. To estimate these parameters, we need correspondences between 3D points in the world coordinate system and their 2D projections on the image plane. By solving a system of equations derived from these correspondences, we can recover the camera parameters.

Data Collection

I used the OpenCV, a python tool to pick the images points. I had to convert the points into x and y coordinates.

Matrices setup

Each point picked takes up two rows in the matrix. Each point are set up in these format

𝑥𝑤,𝑖 𝑦𝑤,𝑖 𝑧𝑤,𝑖 1 0 0 0 0 −𝑥𝑖𝑥𝑤,𝑖 −𝑥𝑖𝑦𝑤,𝑖 −𝑥𝑖𝑧𝑤,𝑖 −𝑥𝑖  
0 0 0 0 𝑥𝑤,𝑖 𝑦𝑤,𝑖 𝑧𝑤,𝑖 1 −𝑦𝑖𝑥𝑤,𝑖 −𝑦𝑖𝑦𝑤,𝑖 −𝑦𝑖𝑧𝑤,𝑖 −𝑦𝑖

6 points were picked resulting in a 12 by 12 matrix. I then proceeded to solve for the eigenvector corresponding to the smallest eigenvalue which gives me a 12 by 1 vector that is reshaped to a 3 by 4 matrix.

Estimate projection matrix.

We know that R3 multiplied by R3 transpose is 1, so we calculate the Euclidean norm of r31, r32, r33 and divide the entire projection by it. This normalizes the projection matrix. The result is the projection matrix.

Recovering the Intrinsic and Extrinsic Parameters

The matrix B is defined as the first 3 by 3 of the resulting projection matrix. The matrix A is B multiplied with B transpose. A is normalized by dividing the entire matrix by the last element. Center of projection u and v is A [1,3] and A [2,3]. Alpha and beta represent the scaling along the x and y axis of the image associated with the focal length. They are calculated with the formula given in the lecture slides. The rotation matrix is then calculated as the inverse of K multiplied by B. The small b vector is the last column vector in the projection matrix. This is used to calculate the translation vector which is the inverse of K multiplied by small b vector.

Error analysis

Trial one

**World Coordinates:** [(12,0,2), (12,0,0), (8,0,0), (4,0,2), (0,4,2), (0,4,0)]

Picked points: [(633, 389), (631, 326), (742, 325), (841, 396), (1037, 396), (1037, 329)]

**The list of x, y coordinates of the image points picked:**

[(633, 389), (631, 326), (742, 325), (841, 396), (1037, 396), (1037, 329)]

**Reprojected points:** [(1.4741397526705076, 4.746093446380766), (1.7334489693529014, 4.300611116734661), (2.0636256338583077, 4.46471300468693), (3.077971968322926, 7.839816418154053), (6.945908191923785, 15.982192687952802), (4681.581349071853, 16620.433970378588)]

**Error analysis on first trial**

I use the Euclidean distance to calculate how far the points are from the reprojected points.

**The Euclidean distance from image**

point one to the reprojection is 739.2401348.

Point two: 706.7297143.

Point three: 806.3800024.

Point four: 923.4617768.

Point five: 1097.91847.

Point Six: 16694.12454

The error rate is very high, this could mean there are a lot of distortions. Maybe skew should have been included in the camera parameters.

A screenshot of a computer program

Description automatically generated

**Second trial**

**Trial Two**

**world coordinates** = [(18,0,0), (18,0,2), (0,16,0), (0,16,14), (0,4,14), (4,0,6)]

The list of x, y coordinates of the image points picked.

[(453, 314), (457, 765), (1407, 320), (1396, 849), (1132, 814), (850, 529)]

**The list of x, y coordinates of the reprojections points**

[(0.523319374173232, 0.22852337365898617), (0.5642357303381001, 0.20232523985290082), (0.6799828751917826, 0.27068197450056525), (0.3573008914365025, 0.37385868843990866), (0.2700948734504428, 0.3975724894577794), (-0.26728279873145855, 0.6947905417029481)]

**Distance of points to reprojection**

Point one: 550.6248142.

Point Two: 890.6453223

Point three: 1094.080625.

Point Four: 1633.396728

Point five: 1393.829792.

Point six: 1001.029893.

A screenshot of a computer program

Description automatically generated

Conclusion

I noticed the second trial was a little better. The overall summation of the six points to their reprojected points is lesser than that of the first trial. I believe this has to do with picking the points that are further apart. The first points were not as far apart as the second points.

I suspect the problem with the reprojection inaccuracy seems to come from the following:

Inaccurate calibration data: if the intrinsic and extrinsic parameters used for calibration are not precisely determined, the projection points may not match the scene geometry.

Lens Distortion: Inherent lens distortions can affect the accuracy of the reprojection process, especially in wide-angle or fisheye lenses.

Imperfect Camera model: The camera model used for reprojection may not fully capture all the nuances of real camera, leading to inaccuracies.

In conclusion, the scene may also be complex.