Problem 1

- Minimum number matching points to solve this mathematically is 6 as the number of 11 unknown elements and 2 independent equations.

The relation between the image points and world points is given as follows:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Figure 1: Relation between the image points and world points [1]

Using the given world and image points we calculate the projection error which can be written mathematically as:

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

Figure 2: Calculating Projection Matrix [1]

The P vector is then reshaped in the form as seen in figure 1. The Projection Matrix we get from the given points is as follows:

$$P = \begin{bmatrix} 3.62233658e-02 & -2.21521073e-03 & -8.83242916e-02 & 9.54088881e-01 \\ -2.53833189e-02 & 8.30555705e-02 & -2.80016309e-02 & 2.68827013e-01 \\ -3.49222323e-05 & -3.27184800e-06 & -3.95667607e-05 & 1.26053750e-03 \end{bmatrix}$$

We then take the first 3 columns of the projection matrix to calculate the Intrinsic Matrix and Rotation. We achieve this by doing the RQ Decomposition the upper triangular matrix calculated from this gives us the Intrinsic Matrix which is later normalized. The orthonormal matrix is the Rotation.

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

Figure 3: RQ Decomposition [1]

We get the intrinsic matrix as:

$$K = \begin{bmatrix} 1619.01802 & 1.89270968 & 800.113194 \\ 0 & -1612.02594 & 616.150421 \\ 0 & 0 & 1 \end{bmatrix}$$

and the rotation matrix as

$$R = \begin{bmatrix} 0.74948643 & 0.00587017 & -0.66199368 \\ 0.0453559 & -0.99806642 & 0.04250012 \\ -0.66046418 & -0.06187859 & -0.74830349 \end{bmatrix}$$

Using the Intrinsic Matrix we can calculate the Translation as follows:

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

The Translation is

$$t = \begin{bmatrix} -3.43300100e-05 \\ 3.14798891e-04 \\ 1.26082328e-03 \end{bmatrix}$$

Now that we have all the values for the projection matrix we can calculate the image points for the given world points. We do this by substituting the projection matrix in the equation shown in Figure 1.

The Mean Reprojection error for the points is 0.4703115540184194. We remove the points with high reprojection error to get a better estimate of the projection and intrinsic matrices. The improved Mean Reprojection error is 0.20016079537593187

Problem 2

- Approach: First we read all the images and convert them to gray scale for better reprojection calculation. Then we use the $findChessboardCorners$ function from OpenCV to find all the corners. Using these corners we can calibrate the camera by calling the $calibrateCamera$ function which takes in the real world coordinates which we can measure and the detected coordinates of each image to calculate the intrinsic matrix and the rotation and translation vectors. We then calculate the reprojection error between each image and the real world points and use them to calculate the mean reprojection error.

- The reprojection error for each image is as follows:
  - Reprojection Error of Image 0 = 0.07811860604254106
  - Reprojection Error of Image 1 = 0.09300490101596924
  - Reprojection Error of Image 2 = 0.11927474143340673
  - Reprojection Error of Image 3 = 0.1434668335059415
  - Reprojection Error of Image 4 = 0.06785202612852255
  - Reprojection Error of Image 5 = 0.07945822777751176
  - Reprojection Error of Image 6 = 0.11452972761669802
  - Reprojection Error of Image 7 = 0.06683274545547946
  - Reprojection Error of Image 8 = 0.07528112226845364
  - Reprojection Error of Image 9 = 0.08243326538824611
  - Reprojection Error of Image 10 = 0.11240184156012407
  - Reprojection Error of Image 11 = 0.12624306572562138
  - Reprojection Error of Image 12 = 0.1230613007443954
  - Mean Reprojection Error = 0.09861218497407008

- The Intrinsic Matrix for the camera is as follows:

$$K = \begin{bmatrix} 2040.39 & 0 & 764.58 \\ 0 & 2032.17 & 1359.29 \\ 0 & 0 & 1 \end{bmatrix}$$

- Improvements:
  - The Accuracy of the K matrix can be improved by taking more pictures as input to calibrate the camera.
  - Using better corner detection methods. This will give us more accurate estimated points in the image.

References

1. Nayar, S. K. (2021, May 9). Camera Calibration — Uncalibrated Stereo. Youtube. Retrieved April 17, 2023, from https://www.youtube.com/playlist?list=PL2zRqk16wsdoCCLpoudGo7QQNks1Ppzo

2. 6.3 Orthogonal and orthonormal vectors. https://www.ucl.ac.uk/ ucahmdl/LessonPlans/Lesson10.pdf

3. (n.d.). OpenCV: Camera Calibration.

   https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html