GEO1016 Assignment 1

Marieke van Arnhem, 4918738 Xiaoluo Gong, 5923476
Zhuoyue Wang, 6093590

May 17, 2024

1 Methodology

For this assignment, an image of black and white squares with known geometries is used to identify the corners and thus the location of the desired points. The corresponding 3D points and image 2D points were chosen manually by using viewing software. See Figure 1 for how the coordinates of one point are obtained. Then, by implementing the camera calibration algorithm, we can find the intrinsic and extrinsic parameters of the camera. There are 11 parameters to solve:

- c_x and c_y are the components of the principal point (intrinsic parameters).
- f_x and f_y correspond to the focal lengths in the x- and y-directions (intrinsic parameters).
- s accounts for the skewness (intrinsic parameter).
- $R = \begin{bmatrix}\mathbf{r}_1^T & \mathbf{r}_2^T & \mathbf{r}_3^T\end{bmatrix}^T$ represents the rotation of the camera (extrinsic parameter).
- $\mathbf{t} = \begin{bmatrix}t_x & t_y & t_z\end{bmatrix}^T$ represents the translation of the camera (extrinsic parameter).

To retrieve these parameters, it is necessary to have at least 6 pairs of points where we know the 3D coordinates ($\mathbf{P}_w = \begin{bmatrix}X & Y & Z\end{bmatrix}^T$) in a world reference system and the pixel coordinates ($\mathbf{p}' = \begin{bmatrix}x & y\end{bmatrix}^T$). The projection matrix P was constructed from the recorded 3D-2D point correspondences.

The unknowns are all included in a matrix M. Which is:

$$M = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} = \begin{bmatrix} \alpha & -\alpha\cot(\theta) & c_x \\ 0 & \beta/\sin(\theta) & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}.$$

Since $p = MP_w$ where $p = \begin{bmatrix}x & y & 1\end{bmatrix}^T$ it is possible to rewrite this to:

$$P\mathbf{m} = \begin{bmatrix} \mathbf{P}_1^T & 0^T & -x_1\mathbf{P}_1^T \\ 0^T & \mathbf{P}_1^T & -y_1\mathbf{P}_1^T \\ & \dots & \\ \mathbf{P}_n^T & 0^T & -x_n\mathbf{P}_n^T \\ 0^T & \mathbf{P}_n^T & -y_n\mathbf{P}_n^T \end{bmatrix} \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \end{bmatrix} = 0.$$

Where m is the vectorized form of the camera matrix M, and it is a row of M, $\mathbf{P}_i = \begin{bmatrix}X_i & Y_i & Z_i & 1\end{bmatrix}^T$ and $0 = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T$. The constraint ($||\mathbf{m}||^2 = 1$) ensures a non-trivial solution, which can be achieved through singular value decomposition with $\mathbf{m}$ being the last column of V from $P = UDV^T$. Resizing $\mathbf{m}$, from a 12x1 matrix, to a 3x4 matrix will give the matrix $\mathcal{M}$. Denote $\mathcal{M} = \begin{bmatrix}A & \mathbf{b}\end{bmatrix} = \begin{bmatrix} \mathbf{a}_2^T & b_1 \\ \mathbf{a}_2^T & b_2 \\ \mathbf{a}_3^T & b_3 \end{bmatrix}$, it is possible to solve the intrinsic and extrinsic parameters by:

$$\begin{array}{lll} \rho = \pm\frac{1}{\|\mathbf{a}_3\|} & \mathbf{r}_1 = \frac{\mathbf{a}_2 \times \mathbf{a}_3}{\|\mathbf{a}_2 \times \mathbf{a}_3\|} & f_x = \alpha \\ \cos\theta = \frac{(\mathbf{a}_1 \times \mathbf{a}_3)\cdot(\mathbf{a}_2 \times \mathbf{a}_3)}{\|\mathbf{a}_1 \times \mathbf{a}_3\|\cdot\|\mathbf{a}_2 \times \mathbf{a}_3\|} & \mathbf{r}_2 = \rho\mathbf{a}_3 & f_y = \beta/\sin(\theta) \\ \alpha = \rho^2\|\mathbf{a}_1 \times \mathbf{a}_3\|\sin\theta & \mathbf{r}_3 = \mathbf{r}_2 \times \mathbf{r}_1 & s = -\alpha\cot(\theta) \\ \beta = \rho^2\|\mathbf{a}_2 \times \mathbf{a}_3\|\sin\theta & \mathbf{t} = \rho K^{-1}\mathbf{b} & c_x = \rho^2(\mathbf{a}_1 \cdot \mathbf{a}_3) \\ & & c_y = \rho^2(\mathbf{a}_2 \cdot \mathbf{a}_3) \end{array} .$$

2 Results

Table 1 shows the calibration results of our algorithm on both provided data and our own data. The last two row shows the root mean squarred error (eq. 1) of the calculated pixel coordinates (p_i) and the obtained pixel coordinates ($\hat{p}_i$). The calculated pixel coordinates is calculated with $\mathbf{p}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T = M\mathbf{P}_i = \begin{bmatrix} \frac{\mathbf{P}_i^T \mathbf{m}_1}{\mathbf{P}_i^T \mathbf{m}_3} & \frac{\mathbf{P}_i^T \mathbf{m}_2}{\mathbf{P}_i^T \mathbf{m}_3} \end{bmatrix}^T$. For further explanation, see Subsection 3.1.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - \hat{p}_i)^2} \tag{1}$$

First, we tested our algorithm on our own measured data. The results are shown in Figure 2. As illustrated, the images do not appear identical, as further evident in Figure 4. This discrepancy is likely due to the coplanarity of the points, as shown in Figure 3. Although the data did not initially appear coplanar, from certain angles, this becomes evident. We then recalibrated using new data. Figure 5 shows the result of the second dataset.

	Own data(6 pts)	**Test data (6 pts)**	**Test data (9 pts)**	**Test data (11 pts)**
f_x	935.181	1935.76	1930.34	1931.37
f_y	933.238	1933.53	1939.39	1931.37
s	3.17897	0.948959	0.706441	-0.00382369
c_x	459.587	952.358	937.39	959.979
c_y	391.496	802.277	788.821	798.013
R	0.47809 -0.038489 -0.87747 0.19398 -0.96974 0.14823 -0.85662 -0.24108 -0.45615	0.81536 -0.0040875 -0.57894 0.16864 -0.95494 0.24424 -0.55385 -0.29678 -0.778	-0.9302 0.29638 0.21650 0.025496 0.64061 -0.76744 -0.36614 -0.70836 -0.60346	0.61790 -0.78199 0.081817 -0.4457 -0.26262 0.85580 -0.6477 -0.56526 -0.5108
t	2.82256 1.52644 35.2442	-0.595783 1.19318 30.9886	3.78561 -0.0632815 36.3244	0.345605 -0.560556 32.8091
Proj_RMSE	0.0085	0.03638	0.9319	0.0007
Cali_RMSE	0.0120	0.0514	1.318	0.0009

Table 1: Calibration results on provided data and own data.

3 Discussion

3.1 Verify the intermediate results

We verified the intermediate results, the 3*4 matrix $\mathcal{M}$, by applying $\mathcal{M}$ on the 3D points to project them into 2D coordinates. We first formed a 4*n homogeneous coordinates matrix of the 3D points, in which each column should be $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$. Then, we multiplied the matrix M with the homogeneous coordinates matrix to get a 3*n matrix p, where $p = \begin{bmatrix} sx_{proj} & sy_{proj} & s \end{bmatrix}^T$ and the s here is the scaling factor. To get the 2D x_{proj} and y_{proj}, we divided the scaling factor here to scale them correctly and the value of the projected 2D x_{proj} and y_{proj} should be very close to the input $x_{observed}$ and $y_{observed}$, if the $\mathcal{M}$ matrix is correct.

Since there will be at least 6 pairs of $x_{observed}$ and $y_{observed}$, checking the projected x_{proj} and y_{proj} with the original ones manually is not very effective. Thus, we calculated the RMSE between the input 2D coordinates and the projected 2D coordinates to quick examine the results. If the matrix $\mathcal{M}$ is correctly constructed and the input data is approximately valid (even with some noise), the RMSE should have a relative small value, usually less than 1.

3.2 Determine the sign of ρ

Vector $\mathbf{t}$ aligns the camera coordinate system, with the z-axis component (t_z) pointing towards the object we want to capture in the image. For the camera to face the object correctly, t_z must always be positive. If t_z is negative, it indicates that the camera is facing the opposite direction and moving away from the object. To address this issue, we adjust the sign of ρ to ensure that the z-axis is positive. This adjustment method, referred to as *method2* in our code, guarantees that the camera is oriented correctly towards the object.

Since it is hard to generate the special case where the ρ is negative due to no negative axis provided, we firstly handled the the sign of ρ from the practical aspect and implied it in code as method1.

We defined a separate function (method 1 in our code) to calculate ρ. Then, we used both the positive and negative ρ to perform the test extraction of the $\mathcal{M}$ parameters. We verified $\mathcal{M}$ for different signs of ρ by applying it to the input coordinates. Specifically, we multiplied $\mathcal{M}$ with the 3D coordinates to obtain the transformed 2D coordinates. We then calculated the RMSE between the input 2D coordinates and the transformed 2D coordinates generated by both versions of $\mathcal{M}$, comparing the results. The ρ value with the smaller RMSE was chosen.

3.3 Accuracy of our results

As mentioned in Subsection 3.2, we calculated the RMSE between the calibrated coordinates and the input coordinates. See row Cali_RMSE in Table 1. Also, we visualized the errors by comparing the input and calibrated 2D coordinates, see Figure 7.

We implied four groups of data to test the robustness of our code and better understand the accuracy. According to Table 1, the test data (9 pts) with some noise has the lowest accuracy and the exact test data (11 pts) has the best accuracy. Also, since we did the sampling accurately comparing to the test data (6 pts), our own data also shows higher accuracy than the test one. Thus, doing a precise sampling can improve the accuracy.

Figure 6 indicates that while the images are somewhat different (likely due to translation differences), they are almost similar.

3.4 Feedback and improvements

We got the following feedback: 1. We should add input validation to prevent errors. 2. We should refactor the code to enhance the readability and maintainability. 3. We should figure out how to determine the sign of rho. From the feedback, we were aware of the lack of input validation. To deal with this issue, we implemented checks to validate the consistency of 3D-2D point correspondences. In this way, we could prevent crashes and send meaningful error messages during the process. Furthermore, to improve the readability, we refactored the code into shorter functions and only kept the essential part inside the main function, which greatly improved the readability of our code.

For the implementation of determining the scaling factor rho, we developed another method to adjust rho to ensure that the z-axis is positive, adding up to the existing method of calculating and comparing results with both positive and negative values (the accurate method and result are detailed in Section 3.2).

4 Who did what

Xiaoluo Gong: Wrote the code to check if the input is valid, verify the matrix M and handle the sign of ρ. Wrote section 3.1, 3.2, 3.3 parts in the report and made the Figure 7.

Zhuoyue Wang: Wrote the part of the code for calculating intrinsic and extrinsic parameters with Marieke. Gathered and gave feedback to the other group. Finished report section 3.4.

Marieke van Arnhem: Wrote the part of the code for calculating the matrix P and calculating the intrinsic and extrinsic parameters with Zhuoyue. Wrote Chapter 1 Methodology and Chapter 2 Results. Made the Figures 1, 2, 3, 4, 5, 6 and Table 1.

Appendix

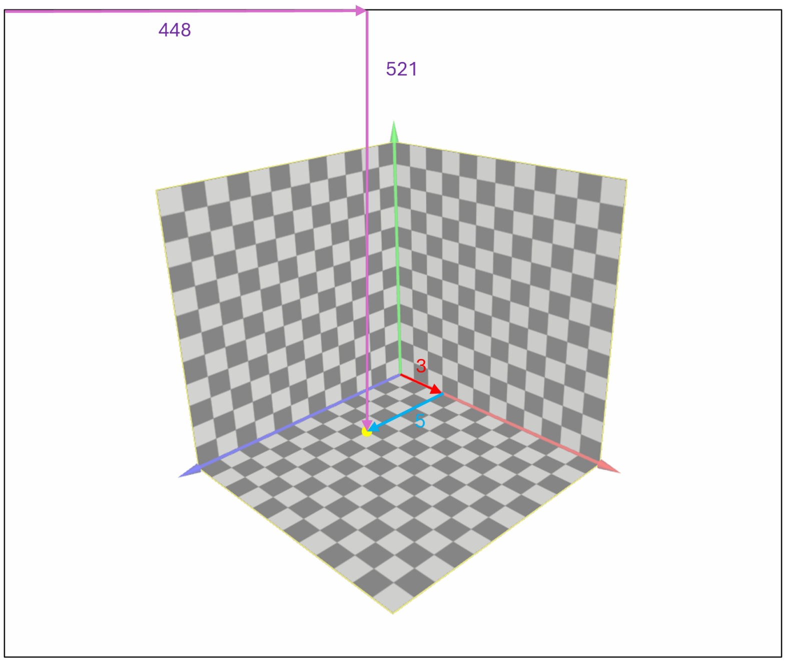

Figure 1: This image shows how the coordinates of a point are obtained. There are three axes: red (x-axis), green (y-axis) and blue (z-axis). One yellow point is shown. This point has the 3D coordinates in a world reference system $\begin{bmatrix} 3 & 0 & 5 \end{bmatrix}^T$ and the pixel coordinates are $\begin{bmatrix} 448 & 521 \end{bmatrix}^T$.

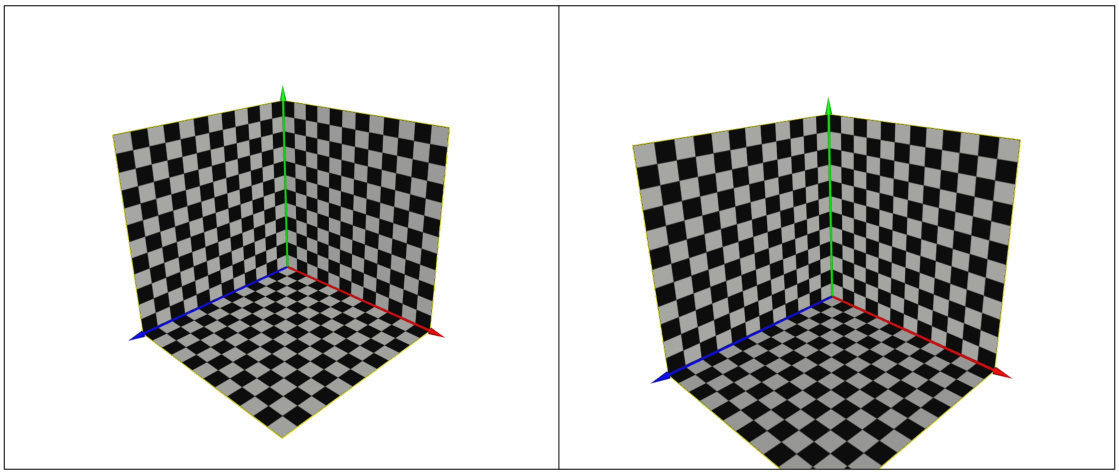

Figure 2: The left image shows the input image with our first dataset as input. The right image shows the registration result after running our code with our own data.

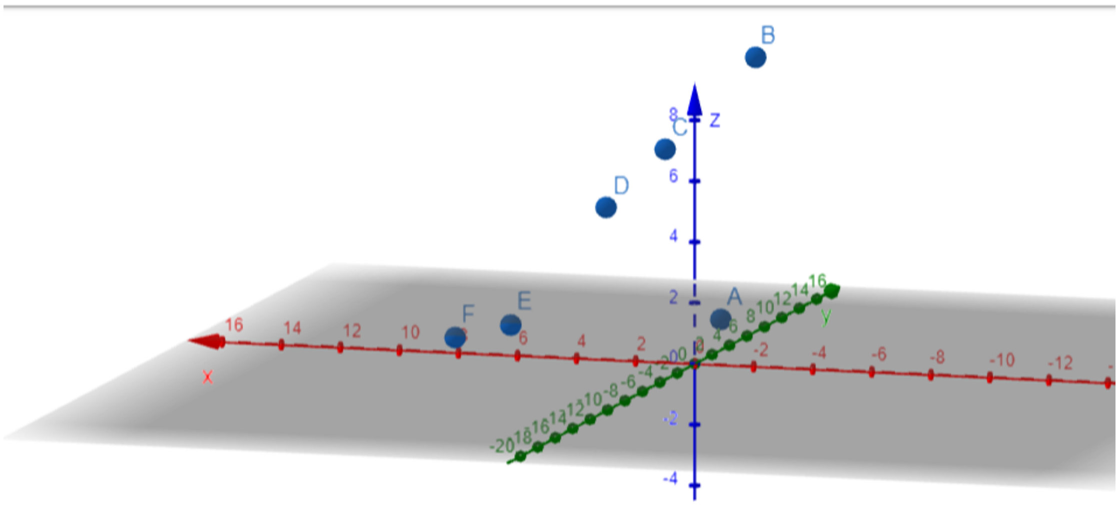

Figure 3: Showing the coplanarity of 5 points of our first dataset. Displayed on the red x-, green y- and blue z-axis.

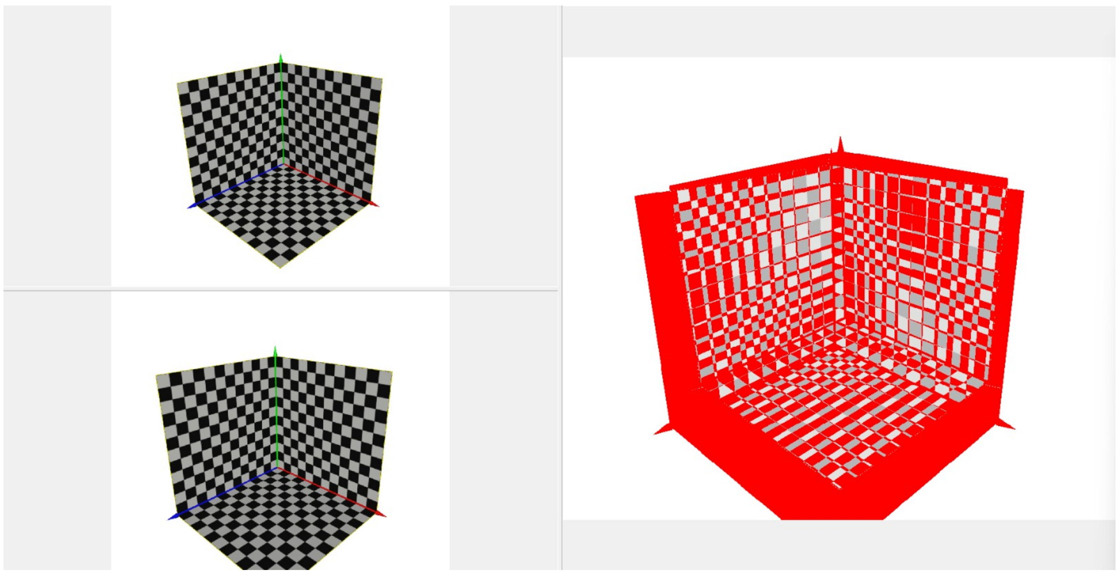

Figure 4: Tool to visualize the difference between the two images with the first dataset as input. ImageDiff is used.

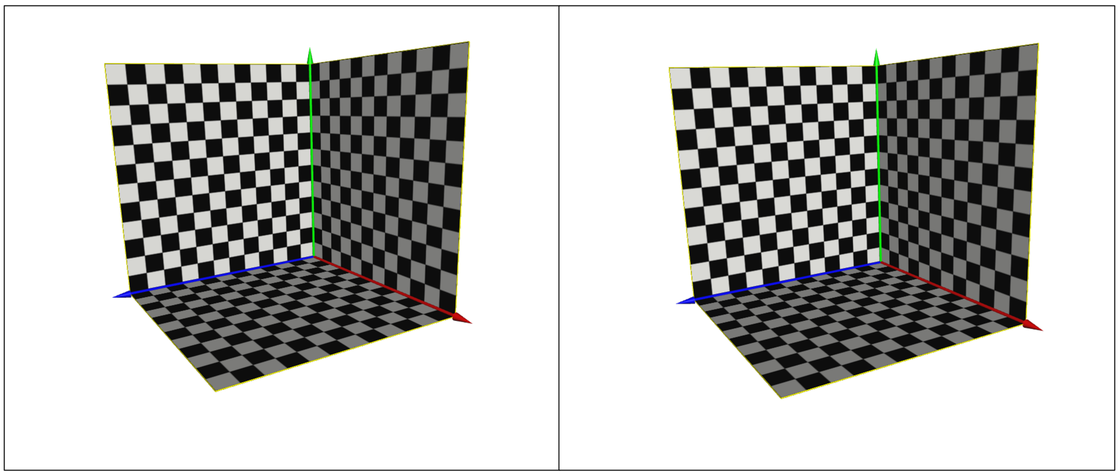

Figure 5: The left image shows the input image with our second dataset as input. The right image shows the registration result after running our code with our own data.

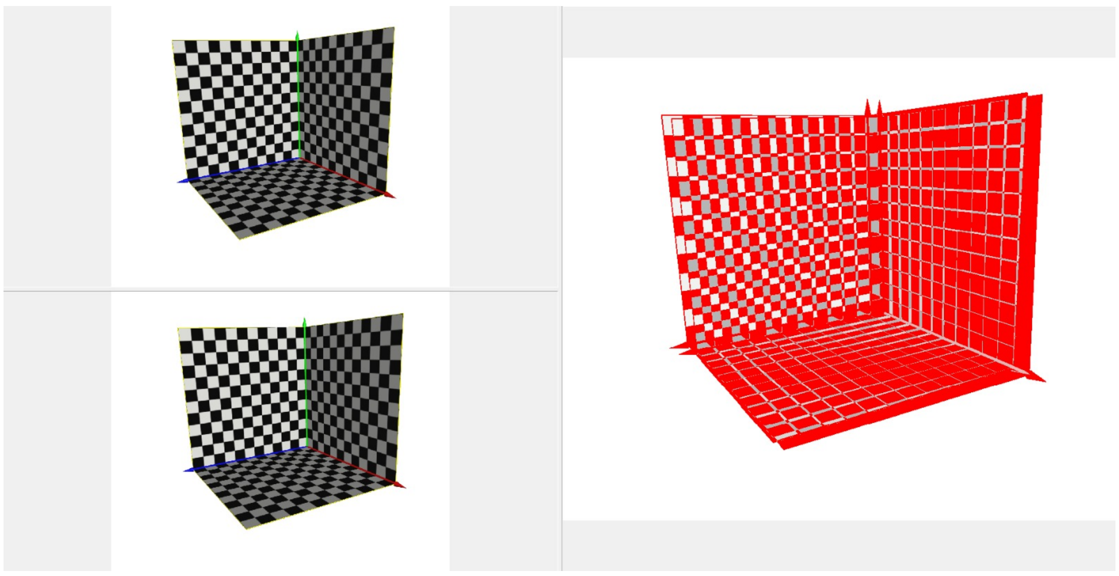

Figure 6: Tool to visualize the difference between the two images with the second dataset as input. ImageDiff is used.

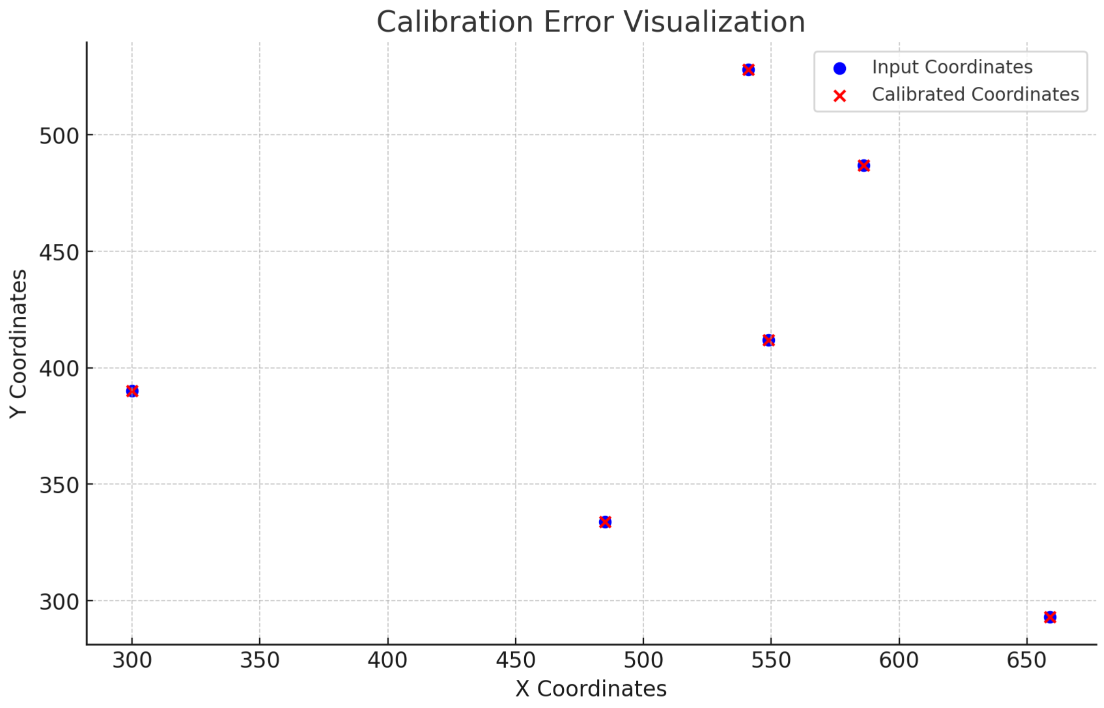

Figure 7: Comparision between input 2D coordinates and calibrated coordinates of our own data.