ROB 498/599 3D Robot Perception: HW1 Writeup

Name: Sarah Chan
Uniqid: sjchan
Email: sjchan@umich.edu

Problem 1: Camera Calibration

i. The camera calibration process involves computing a 3x4 projection matrix that maps 3D world coordinates to 2D image coordinates. The function takes as input a set of corresponding 2D-3D points and constructs a linear system of equations based on equation 1. Expanding equation 1 results in equation 2. Then, I rearranged these equations to create a set of equations of the form $Ax = 0$, where each point correspondence contributes two rows to matrix A shown in equation 3. Finally, I solved for M using Singular Value Decomposition (SVD). The projection matrix is the last column of V in the SVD decomposition of A reshaped into a 3x4 matrix. The projection matrix I obtained from my code is shown in equation 4.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -M_1- \\ -M_2- \\ -M_3- \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{1}$$

$$x_i = \frac{M_{11}X_i + M_{12}Y_i + M_{13}Z_i + M_{14}}{M_{31}X_i + M_{32}Y_i + M_{33}Z_i + M_{34}}, y_i = \frac{M_{21}X_i + M_{22}Y_i + M_{23}Z_i + M_{24}}{M_{31}X_i + M_{32}Y_i + M_{33}Z_i + M_{34}} \tag{2}$$

$$A_{2i,2i+2} = \begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -x_iX_i & -x_iY_i & -x_iZ_i & -x_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -y_iX_i & -y_iY_i & -y_iZ_i & -y_i \end{bmatrix} \tag{3}$$

$$M = \begin{bmatrix} 0.4583 & -0.2947 & -0.0140 & 0.0040 \\ -0.0509 & -0.0546 & -0.5411 & -0.0524 \\ 0.1090 & 0.1783 & -0.0443 & 0.5968 \end{bmatrix} \tag{4}$$

ii. The reprojection error measures how accurately the 3D points are projected back onto the 2D image plane using the projection matrix M. It quantifies the discrepancy between the original 2D points and the reprojected 2D points obtained from the estimated 3D reconstruction. In my results, the reprojection error was very small, indicating that the computed projection matrix accurately maps the 3D world points to their corresponding 2D image locations. The error is measured in pixels, as it represents the pixel distance between the true and estimated projections. I would consider a reprojection error of less than 0.01 pixels to be acceptable because it shows that the projection is a pretty accurate representation of the 2D image.

$$error_{reprojection} = \frac{1}{N}\sum_{i=1}^{N} ||proj(MX_i) - p_i||_2 \tag{5}$$

$$errror_{reprojection} = 0.00223 \tag{6}$$

Sparse 3D Reconstruction

Problem 2: Estimation of Fundamental Matrix

i. The fundamental matrix is a 3x3 matrix which describes how points in two images of the same scene are related. In order to obtain the fundamental matrix, I implemented the 8-point algorithm. First, I normalized the correspondences by multipling them by the scaling matrix T. Then, I expanded the epipolar constraint, $x_1Fx_2 = 0$, at each point shown in equation 7. This resulted in the system $Ax = 0$ which I decomposed using SVD. Then I enforced the rank 2 constraint by setting the second element in S from the decomposition to zero, and I re-composed F using U, the new S, and V. Lastly, I de-normalized the the matrix using equation 8. The projection matrix I obtained from my code is shown in equation 9.

$$A_i = \begin{bmatrix} x_{i_2}x_{i_1} & x_{i_2}y_{i_1} & x_{i_2} & y_{i_2}x_{i_1} & y_{i_2}y_{i_1} & y_{i_2} & x_{i_1} & y_{i_1} & 1 \end{bmatrix} \tag{7}$$

$$F_{denorm} = T^T F T \tag{8}$$

$$F = \begin{bmatrix} 3.5644 \times 10^{-9} & -5.9213 \times 10^{-8} & -1.6503 \times 10^{-5} \\ -1.3083 \times 10^{-7} & -1.3110 \times 10^{-9} & 1.1247 \times 10^{-3} \\ 3.0478 \times 10^{-5} & -1.0801 \times 10^{-3} & -4.1658 \times 10^{-3} \end{bmatrix} \tag{9}$$

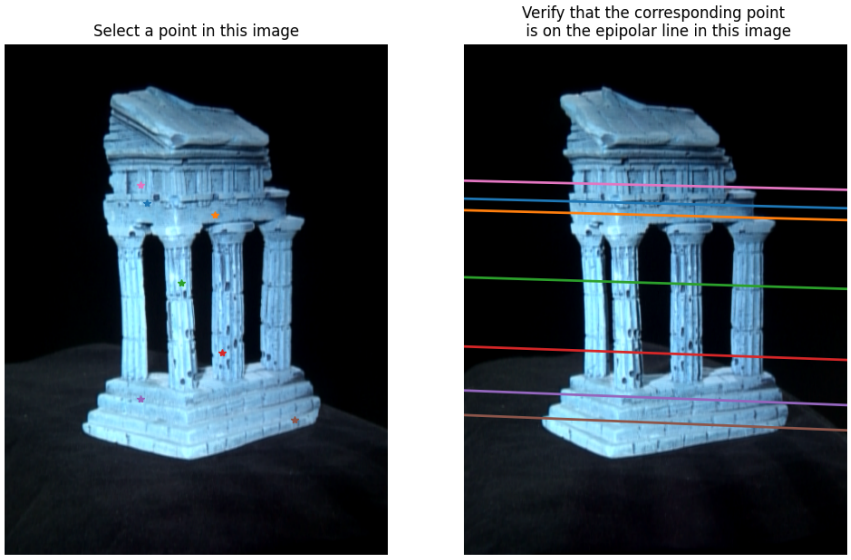

Figure 1: GUI output for epipolar lines

ii. Based on my observations, the selected points in figure 1 were generally accurate. The epipolar lines in the second image appeared to pass directly over the corresponding points, which aligns with expectations. This suggests that the computed fundamental matrix correctly models the geometric relationship between the two images. Despite the overall accuracy, there are potential sources of error that could cause misalignment in the epipolar lines. For example, numerical instability in the SVD step in fundamental matrix computation can cause small errors. Another source of error can be produced in the epipolar line approximation. Since pixel values are discrete, small deviations can occur when computing and visualizing the epipolar lines.

Problem 3: Find Epipolar Correspondences

i. To find correspondences between the two images, I iterated through each 2D point in Image 1, computing its corresponding epipolar line in Image 2 using Equation 10. For each point, I identified the best match along the epipolar line by evaluating the sum of squared differences (SSD) between 7×7 pixel patches centered around the candidate points in Image 2 and the reference point in Image 1. The point with the smallest SSD value was selected as the corresponding 2D point in Image 2. In addition to SSD, I also experimented with the Manhattan distance as a similarity measure. However, this approach resulted in less accurate correspondences, which I verified using the `epipolar_correspondences_GUI_tool`. The SSD method ultimately produced better results, making it more effective for feature matching in this context.

$$line = F \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10}$$

ii. The correspondences on the stairs at the bottom of the image appeared to have the best results, probably because the features were the most unique there. The points on the pillars seemed to produce the worst results, likely due to the similar features on the four pillars, so my distance metric might falsely identify one pillar as another in image 2.

Problem 4: Compute the Essential Matrix

i. To derive the essential matrix, I used equation $F = K_1^{-T} E K_2^{-1}$ to determine that $E = K_2^T F K_1$, where K_1 and K_2 represent the intrinsic matrices of cameras 1 and 2 respectively. The essential matrix is necessary to determine the relative position and orientation between the cameras and the 3D position of the corresponding image points. It can be broken down into the rotation and translation components which are used to find the camera matrix. The essential matrix I obtained from my code is shown in equation 11.

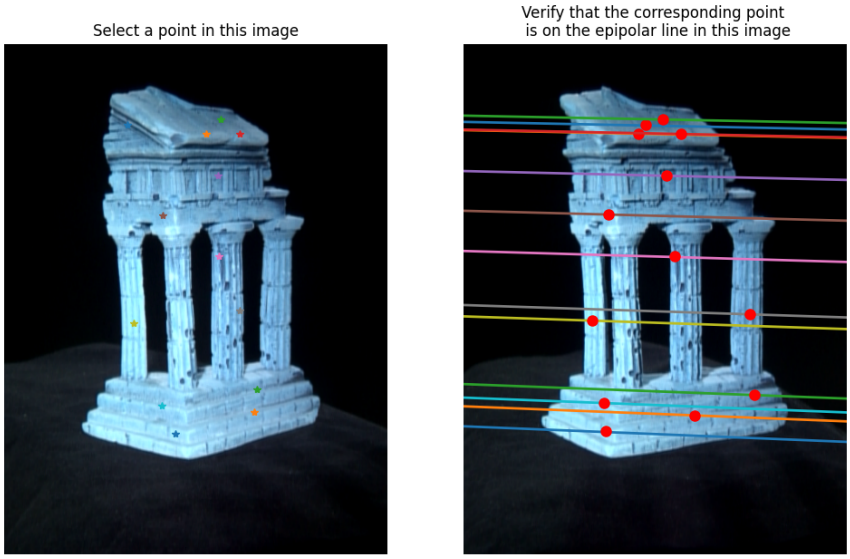

Figure 2: GUI output for epipolar correspondences

$$E = \begin{bmatrix} 8.2395 \times 10^{-3} & -1.3737 \times 10^{-1} & -4.5678 \times 10^{-2} \\ -3.0352 \times 10^{-1} & -3.0524 \times 10^{-3} & 1.6554 \\ -1.1292 \times 10^{-3} & -1.6760 & -2.8730 \times 10^{-3} \end{bmatrix} \tag{11}$$

Problem 5: Triangulation

i. I used the essential matrix to obtain the rotation (R) and translation (t) components of the extrinsic matrix. After using SVD to decompose the extrinsic matrix, the candidates for R were UWV^T or UW^TV^T using W from equation 12 and the candidates for t were the positive and negative versions of the last column of U. I did not automate the process to determine the correct extrinsic matrix. Instead, I ran my program for each combination of $[R|t]$ where $R = R_1, R_2$ and $t = \pm t$. I chose the correct R and t based on the final reconstruction. When I set $R = R1$, there were very few points visible in the reconstruction. When I set $t = -t$, the reconstruction appeared to be reflected across the y-axis. Thus, I determined that the extrinsic matrix is $[R_2|t]$. The extrinsic matrix that I obtained from my code is shown in 13.

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$Ext = \begin{bmatrix} 9.6519 \times 10^{-1} & -2.8456 \times 10^{-2} & 2.6000 \times 10^{-1} & 9.9628 \times 10^{-1} \\ 2.7350 \times 10^{-2} & 9.9959 \times 10^{-1} & 7.8719 \times 10^{-3} & 2.7350 \times 10^{-2} \\ -2.6012 \times 10^{-1} & -4.8702 \times 10^{-4} & 9.6558 \times 10^{-1} & -8.1711 \times 10^{-2} \end{bmatrix} \tag{13}$$

ii. For triangulation, I set up my SVD problem by expanding the projection in equation 14, separating the camera matrix C, which is a camera's intrinsic matrix multiplied by its extrinsic matrix, into its rows C_1, C_2, and C_3. X represents the 3D world coordinate. Then, I rearranged the expanded equations, setting them equal to zero in 15. This allowed me to set up the $Ax = 0$ problem, where A is shown in equation 16 and includes the points on image 1 projected from camera 1 and the points on image 2 projection from camera 2. I set up the $Ax = 0$ problem for each pair of corresponding points to use SVD to estimate the 3D world coordinate of that correspondence. I calculated the reprojection error by projecting the 3D world points from triangulation into 2D image points using the camera matrix from camera 2. Then, I compared these 2D image points to the input points from image 2 and summed those errors using equation 5. This error is represented in pixels and shows how closely points from triangulation project onto the 2D image points. The reprojection error I calculated from my triangulation implementation is shown in 17.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -C_1- \\ -C_2- \\ -C_3- \end{bmatrix} X, X = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{14}$$

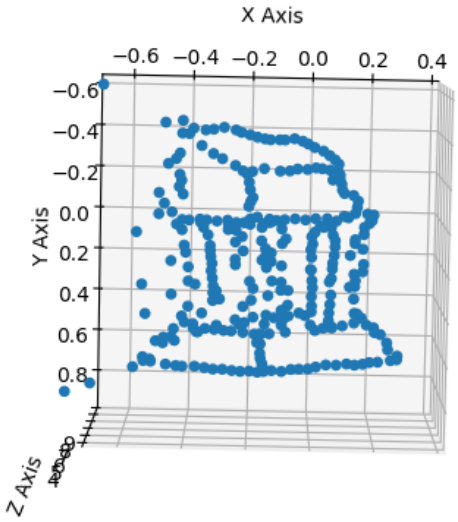

Figure 3: Sparse Reconstruction Visualization

$$x(C_3 - C_1)X = 0, y(C_3 - C_1)X = 0 \tag{15}$$

$$A = \begin{bmatrix} x_1(C_3)_1 - (C_1)_1 \\ y_1(C_3)_1 - (C_2)_1 \\ x_2(C_3)_2 - (C_1)_2 \\ y_2(C_3)_2 - (C_2)_2 \end{bmatrix} \tag{16}$$

$$errror_{reprojection3d} = 0.475670 \tag{17}$$

The reprojection error obtained from OpenCV's triangulation was very similar to the error computed using my implementation. However, OpenCV's error was slightly larger, differing only at the seventh decimal place. This error could be caused by differences in numerical stability, where OpenCV might use different floating-point precision causing the reprojection error to be different. Since the reprojection errors from my solution and OpenCV's solution were very similar, I think OpenCV uses a similar approach to solve the triangulation problem. This error also represents the error between the original 2D points and their reprojected counterparts after triangulation, in pixels.

$$errror_{reprojectionOCV} = 0.475670 \tag{18}$$

Problem 6: Connecting the dots and visualizing the point cloud

i. In the main function of my program, I first computed the fundamental matrix using corresponding points from both images. This matrix was then used to determine epipolar correspondences, ensuring that matching points adhered to epipolar constraints. Next, I used the fundamental matrix along with the intrinsic matrices from both cameras to compute the essential matrix. The essential matrix was used for triangulation, where I estimated the 3D world positions of the corresponding points. Finally, I visualized the reconstructed 3D point cloud using a scatter plot, providing an intuitive representation of the reconstructed scene.

ii. My `visualize(point_cloud)` function uses Matplotlib to create a scatter plot of the 3D points obtained from triangulation. I verified its output by rotating the plot and viewing it from all angles to check that it has the same features as the temple in the 2D images.

iii. Overall, my 3D reconstruction appears mostly correct. However, there are some areas where the results seem slightly inaccurate. For example, while the pillars generally appear straight, there are regions where they seem misaligned. One possible cause of these misalignments is incorrectly identifying 2D correspondences between the two images. Since the pillars share similar visual features, it becomes more challenging to accurately match points across the two views. Any mismatches in these correspondences can introduce errors into the triangulation process, leading to inaccuracies in the estimated 3D world points

Optional Problem 7: Pain points and Flash cards

i. My biggest difficulty with this homework is working with the timing of the lectures. The first two weeks of lectures gave me enough knowledge to solve problem 1 (camera calibration). That left only 1.5 weeks to finish the rest of the homework (sparse reconstruction) which I found much harder than the camera calibration section. As the deadline approached, I started to get worried that I wouldn't be able to solve the the rest of the homework, so I had to seek outside resources to get started. I would have preferred for the lectures to be more evenly distributed to match the pacing of the homework. Other than that, I really enjoyed this assignment and learning about projective geometry!

References and Credits:
Parts of this homework are from David Fouhey's EECS 442 and CMU 16-385, and from Jason Corso's ROB 498/599 classes.