Aryan Dawer

Department of Computer Science

Dartmouth College

18/02/2024

PSET 3

For this pset, I shared ideas with my fellow classmate Atul Agarwal, however all the work in this document is my own.

Section 2.1

Fundamental Matrix

$$F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0.001 \\ 0 & -0.001 & -0.004 \end{bmatrix} \tag{1-1}$$

Visualization of Epipolar Lines

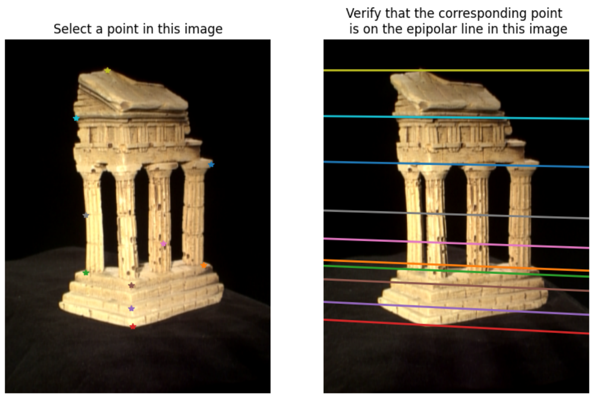
Select a point in this image
Verify that the corresponding point is on the epipolar line in this image

Figure 1: Epipolar lines in two views corresponding to several matched points.

Section 2.2

Methodology

The function first converts the input images to grayscale if they are in color. It then defines a window of size w (in this case $w = 5$), which is used to compare the similarity of points across the two images. For each point in the first image, the corresponding epipolar line in the second image is computed using the fundamental matrix. We then examine points along this line and search for the point that has the most similar intensity pattern to the original point, within a window of size w centered around the point.

Similarity Metric

The similarity between two points is defined by the Euclidean distance between the intensity patterns (windows) surrounding the points. Mathematically, for a point $\mathbf{x}_1$ in the first image with a window W_1 and a candidate point $\mathbf{x}_2$ in the second image with window W_2, the distance d is computed as:

$$d(\mathbf{x}_1, \mathbf{x}_2) = ||W_1 - W_2||^2 \tag{1-2}$$

The point $\mathbf{x}_2$ with the smallest distance d is selected as the corresponding point to $\mathbf{x}_1$.

Visualization of Epipolar Correspondences

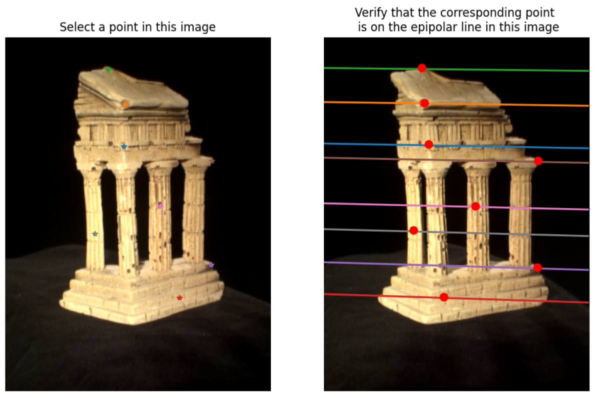
Select a point in this image
Verify that the corresponding point is on the epipolar line in this image

Figure 2: Visualization of the epipolarMatchGUI with the implementation of epipolar correspondences.

Limitations and Failures

The algorithm may fail in cases of repetitive textures or edges where multiple locations along the epipolar line may yield similar intensity patterns, leading to ambiguity in identifying the correct correspondence. This is particularly true in regions with low texture (A good example of this is seen in Figure 2 above in the base of the temple) or high noise, where the window might not capture enough distinctive information to differentiate between candidate matches.

Section 2.3

Essential Matrix

$$E = \begin{bmatrix} 0.006 & -0.130 & -0.034 \\ -0.309 & 0.002 & 1.655 \\ -0.006 & -1.675 & -0.002 \end{bmatrix} \tag{1-3}$$

Section 2.4

Method to choose best extrinsic parameters

Given the essential matrix E, I utilized the function `camera2` from the `helper.py` module, which yields four potential extrinsic matrices. The best configuration is the one that produces 3D points with positive depth values when observed from both camera viewpoints.

The procedure for determining the correct extrinsic matrix is as follows:

1. Utilize the `camera2` function to derive the four possible camera projection matrices $P2$ from the essential matrix E.

2. For each $P2$ candidate, triangulate the corresponding points $pts1$ and $pts2$ using a triangulation algorithm. This produces sets of 3D points for each possible configuration.

3. Count the number of 3D points with positive depth values for each set.

4. Select the $P2$ matrix and its associated 3D points set that yield the maximum count of positive depth values.

This process ensures the selection of an extrinsic matrix that results in a physically plausible 3D reconstruction.

Re-projection Error

As mentioned in the assignment pdf, re-projection error should be less that 2 pixels, and my `epipolar_correspondences` returns an error of 0.1625.

Section 2.5

0.1 3D Reconstructions

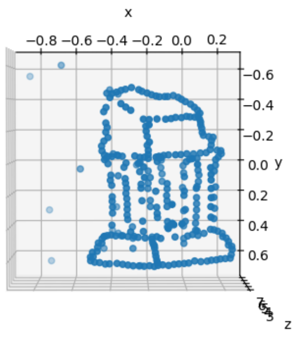

Figure 3: View from Angle 1

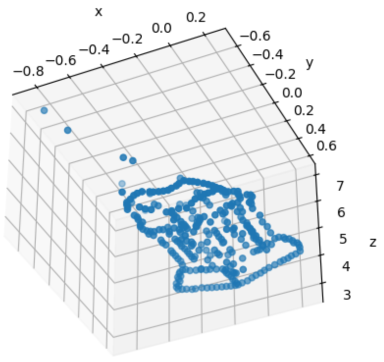

Figure 4: View from Angle 2

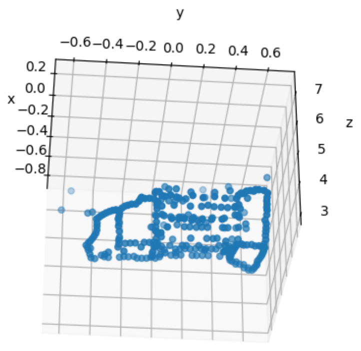

Figure 5: View from Angle 3

Section 3.1

Rectification Result

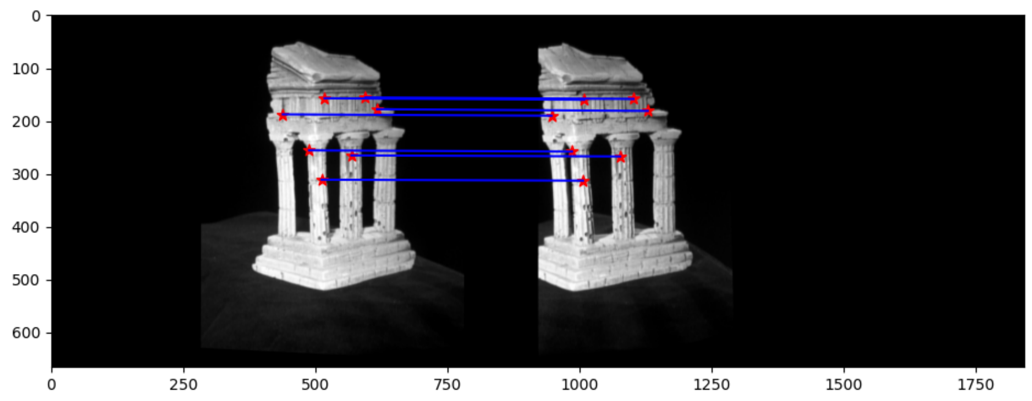

Figure 6: Screenshot of the rectified temple images with horizontal epipolar lines

Section 3.2

Disparity and Depth Maps

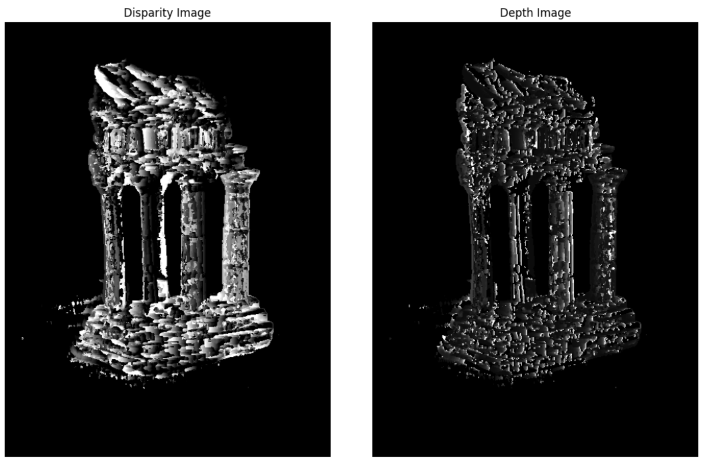

Figure 7: Disparity and Depth Maps

Section 4.1

Pose Test Result

The following table presents the results from executing the `test_pose.py` script. It summarizes the reprojection error and pose error for both clean and noisy 2D points:

	Clean Points	**Noisy Points**
Reprojection Error	1.75×10^{-10}	6.28
Pose Error	2.38×10^{-12}	1.11

Table 1: Pose test results showcasing the reprojection and pose errors for clean and noisy 2D points (in pixels)

Section 4.2

Parameter Test Result

The results from the `test_params.py` script are summarized in the table below. The errors for intrinsic parameters, rotation, and translation are presented for scenarios with clean and noisy 2D points:

	Clean Points	**Noisy Points**
Intrinsic Error	2.00	2.18
Rotation Error	2.83	2.83
Translation Error	3.70	3.67

Table 2: Parameter test results indicating intrinsic, rotation, and translation errors for clean and noisy 2D points (in pixels)

Section 4.3

Project CAD Results

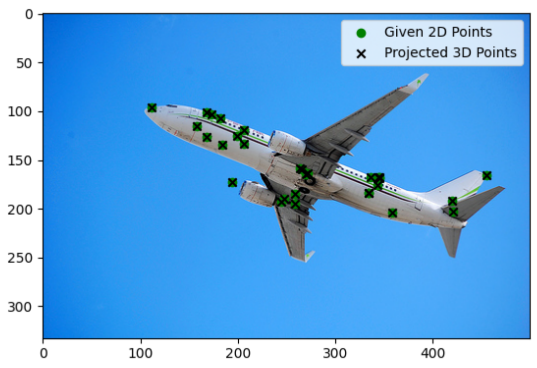

Figure 8: The image annotated with given 2D points and projected 3D points

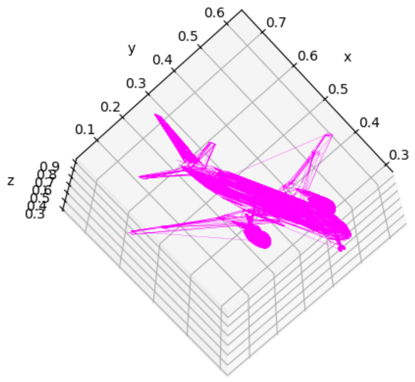

Figure 9: CAD model (rotated)

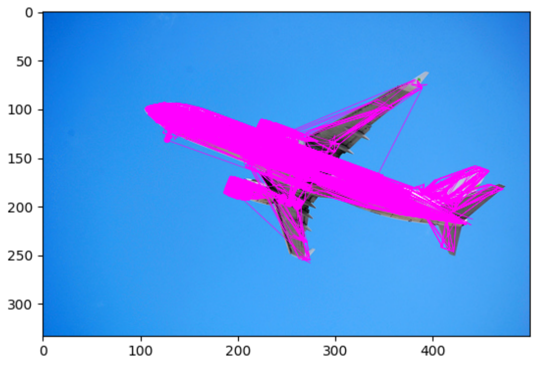

Figure 10: CAD model overlayed on image

Submitted by Aryan Dawer on 18/02/2024.