Computer Vision - EE 046746
HW4

Tomer Raz Adi Hatav

October 4, 2024

Disclaimer: Chat GPT was used in the process of writing this document.

Contents

Part 1 Sparse Reconstruction

1.1 Eight Point Algorithm

In this section, we implement the Eight Point Algorithm to estimate the fundamental matrix F. The following steps outline the implementation process:

1.1.1 Step 1: Normalize the Points

The point correspondences between the two images are normalized by dividing each coordinate by the maximum of the image's width and height, denoted as $p_{\max}$. This scaling is done using a transformation matrix T, applied to the homogeneous coordinates of the points in both images:

$$T = \begin{bmatrix} \frac{1}{p_{\max}} & 0 & 0 \\ 0 & \frac{1}{p_{\max}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.1.2 Step 2: Construct the Matrix A

Using the normalized coordinates of the points, we construct a matrix A, where each point correspondence (x_i, y_i) from the first image and (x'_i, y'_i) from the second image contributes to a row in A. The elements of each row are given by:

$$A_i = \begin{bmatrix} x'_i x_i & x'_i y_i & x'_i & y'_i x_i & y'_i y_i & y'_i & x_i & y_i & 1 \end{bmatrix}$$

1.1.3 Step 3: Compute the Fundamental Matrix F

The fundamental matrix F is computed by solving the homogeneous system $Af = 0$, where f is the vectorized form of F. This is achieved using SVD on A, where the last column of the right singular vectors V corresponds to the solution for f:

$$F = \text{reshape}(V[-1], (3, 3))$$

1.1.4 Step 4: Enforce the Rank-2 Constraint

The computed fundamental matrix F may not have the required rank of 2. To enforce this, we perform SVD on F to decompose it into $U\Sigma V^T$. We then set the smallest singular value in Σ to zero and recompute F:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad F' = U\Sigma V^T$$

1.1.5 Step 5: Un-normalize the Fundamental Matrix

Finally, we un-normalize the fundamental matrix by applying the inverse of the normalization transformation:

$$F_{\text{unnorm}} = T^T F' T$$

The final F_{unnorm} is the estimated fundamental matrix.

1.1.6 Implementation

The complete implementation of the Eight Point Algorithm is provided in the `eight_point` function, which takes as inputs the points `pts1`, `pts2`, and the scalar value $p_{\max}$, and outputs the fundamental matrix F.

1.1.7 Results

The fundamental matrix computed for the two images of the temple is

$$F = \begin{pmatrix} 3.56440672 \times 10^{-9} & -5.92131870 \times 10^{-8} & -1.65029959 \times 10^{-5} \\ -1.30829066 \times 10^{-7} & -1.31095126 \times 10^{-9} & 1.12471525 \times 10^{-3} \\ 3.04775126 \times 10^{-5} & -1.08013400 \times 10^{-3} & -4.16583180 \times 10^{-3} \end{pmatrix}$$

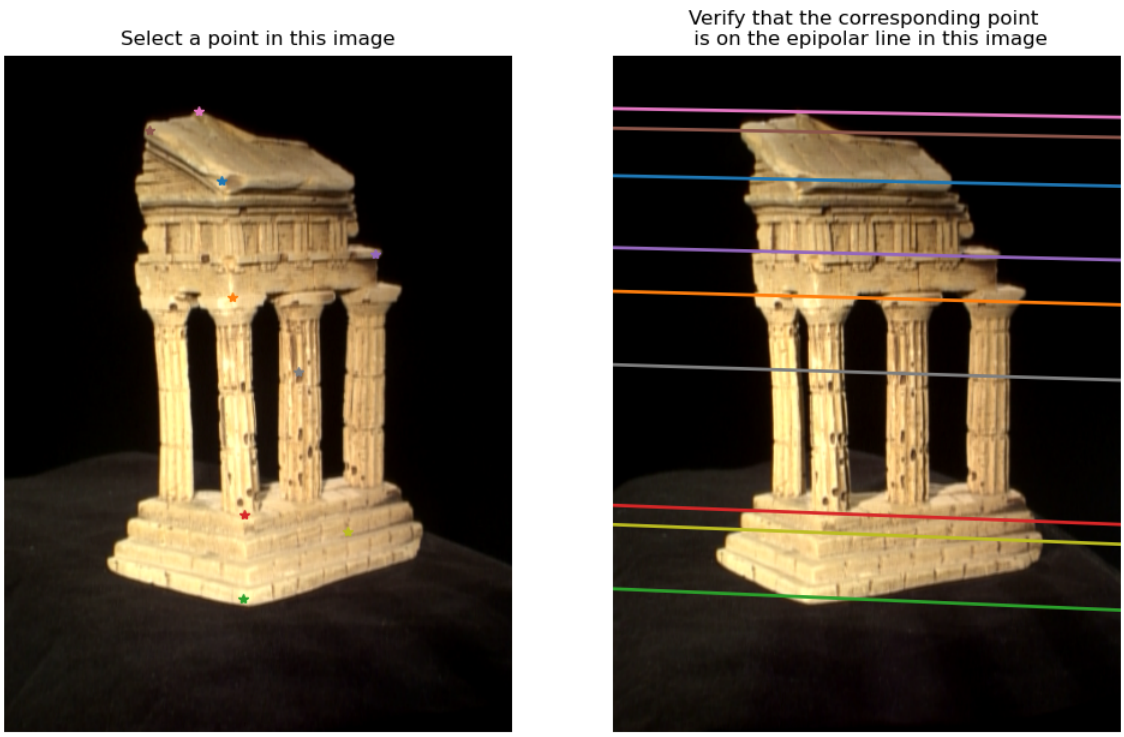

Figure 1: Epipolar lines using the supplied visualization function, given the fundamental matrix we computed. The points in the second image (right) corresponding to those selected in the first image (left) visually appear to be incident with the corresponding epipolar lines. (The next part deals with finding corresponding points.)

1.2 Epipolar Correspondences

In this section, we implement the 'epipolar correspondences' function to find corresponding points in the second image for a given set of points in the first image. The goal is to use the fundamental matrix F to restrict the search for corresponding points to the epipolar lines, simplifying the problem of finding point pairs.

1.2.1 Implementation Details

The process of finding epipolar correspondences involves several key steps:

1. **Epipolar Line Calculation:** For each point $\mathbf{p_1} = (x_1, y_1)$ in the first image, we compute the corresponding epipolar line $\mathbf{l}$ in the second image using the fundamental matrix F. The epipolar line is given by:

$$\mathbf{l} = F \cdot \mathbf{p_1}$$

2. **Window-Based Matching:** To determine the best corresponding point $\mathbf{p_2} = (x_2, y_2)$ along the epipolar line in the second image, we utilize a window-based matching approach. For each candidate point along the epipolar line, we extract a small square window of pixels centered at $\mathbf{p_1}$ in the first image and at the candidate point $\mathbf{p_2}$ in the second image. The size of this window is defined by the parameter `window_size`, and we denote the half-window size as $\texttt{half\_window} = \frac{\texttt{window\_size}}{2}$.

3. **Similarity Metric - Sum of Squared Differences (SSD):** To evaluate the similarity between the windows from the two images, we compute the Sum of Squared Differences (SSD) between the pixel intensities of the two windows. The SSD for a candidate point $\mathbf{p_2}$ is given by:

$$\text{SSD} = \sum_{(i,j)\in\text{window}} \left(I_1(x_1 + i, y_1 + j) - I_2(x_2 + i, y_2 + j)\right)^2$$

   Here, I_1 and I_2 represent the intensity values of the pixels in the first and second images, respectively. The SSD measures the dissimilarity between the two windows, with a lower SSD indicating a higher similarity. The candidate point $\mathbf{p_2}$ that minimizes the SSD is selected as the corresponding point in the second image.

4. **Search Along the Epipolar Line:** For each point $\mathbf{p_1}$, the function iteratively checks all possible candidate points along the epipolar line in the second image. It calculates the SSD for each candidate and keeps track of the point that yields the minimum SSD. The final corresponding point is the one with the lowest SSD value.

1.2.2 Challenges and Considerations

The window-based SSD approach is effective for matching distinct features such as corners or edges. However, it struggles in regions with repetitive patterns or low texture, where the SSD values for different points may be similar, leading to potential mismatches. Additionally, the choice of window size is crucial: a small window may be sensitive to noise, while a larger window may include irrelevant information, leading to inaccurate matches. We ended up going with a window size of 5 (two pixels on each side of the center pixel).

1.2.3 Results

Areas that consistently fail to be identified correctly (even under slight perturbations of the selected point) are mainly areas of homogeneous coloring, such as the image background. Some points on the temple's pillars and steps are also misidentified.

Figure 2: Three misidentified points: The top point was mistakenly placed on an adjacent column, the middle point was placed on the brick to the left of the one it is actually on, and the bottom point was also matched to a point to the left of its true placement.

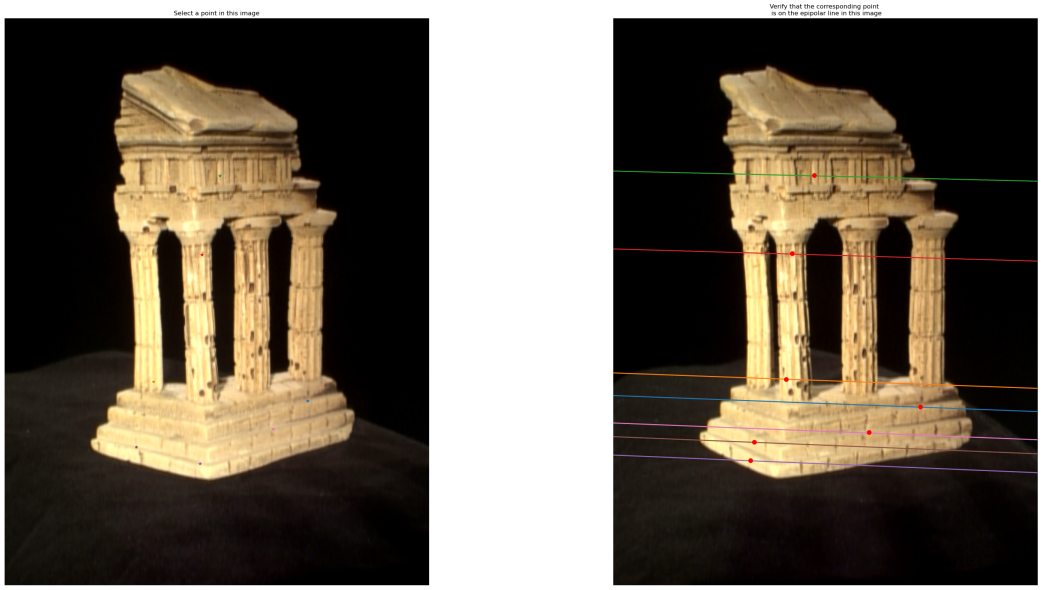

Figure 3: Mixed results. The top two points were matched properly (to good visual degree). The point selected on the leftmost pillar was matched to a point on the pillar to the right. The two highest points selected on the steps are properly matched while the lowermost two are not.

1.3 Essential Matrix

To compute the full camera projection matrices, we first need to compute the Essential matrix E. The Essential matrix is related to the Fundamental matrix F through the intrinsic camera matrices $K1$ and $K2$ of the two images:

$$E = K_2^\top F K_1$$

Here, F is the Fundamental matrix computed between the two images, and K_1 and K_2 are the intrinsic camera matrices. The Essential matrix E encapsulates the relative rotation and translation between the two camera views. In our implementation, the essential matrix is computed using the provided intrinsic matrices for the temple images.

For the temple image pair, after computing the Fundamental Matrix and using the provided intrinsic matrices, the resulting Essential Matrix E is:

$$E = \begin{pmatrix} 4.60261645 \times 10^{-2} & -8.37143325 \times 10^{-1} & -2.62254028 \times 10^{-1} \\ -1.73129035 \times 10^{0} & -1.81926284 \times 10^{-2} & 9.65216091 \times 10^{-1} \\ -8.88283922 \times 10^{-3} & -9.77247828 \times 10^{0} & -1.64220428 \times 10^{-2} \end{pmatrix}$$

1.4 Triangulation

In this section we used the camera projection matrices $M1$ and $M2$, and the corresponding points `pts1` and `pts2` in the two images.

1.4.1 Triangulation Process

Given the projection matrices $M1$ and $M2$, and the 2D point correspondences (x_1, y_1) and (x_2, y_2) from the two images, we construct a system of linear equations based on the projection equation:

$$\mathbf{A} = \begin{pmatrix} x_1 M1_{3,:} - M1_{1,:} \\ y_1 M1_{3,:} - M1_{2,:} \\ x_2 M2_{3,:} - M2_{1,:} \\ y_2 M2_{3,:} - M2_{2,:} \end{pmatrix} \mathbf{X} = 0$$

The matrix $\mathbf{X}$ represents the 3D coordinates in homogeneous form. To solve for $\mathbf{X}$, we use Singular Value Decomposition (SVD). This process is repeated for all point correspondences, yielding a set of 3D points.

1.4.2 Determining the Correct $M2$

The essential matrix E provides four possible candidates for the extrinsic camera matrix $M2$. To determine the correct $M2$, we triangulate the points using each candidate and select the one that produces the most 3D points with positive depth values (i.e., points that are in front of both cameras). The correct $M2$ is the one that yields a consistent and plausible 3D reconstruction.

1.4.3 Re-projection Error

The quality of the triangulation can be evaluated by calculating the re-projection error. This error is computed by projecting the 3D points back into the image planes using the camera projection matrices and measuring the Euclidean distance between the projected points and the original 2D points:

$$\text{Re-projection Error} = \frac{1}{N} \sum_{i=1}^{N} \|\texttt{pts1}_i - \texttt{projected\_pts1}_i\|_2$$

For the given data, the re-projection error was calculated to be Re-projection error = 0.4690 pixels. In addition we calculate the re-projection error for each image separately and the scores was really close (0.469 and 0.468) which indicate that the 3D reconstruction did not tend to favor one image on another. In general, a low re-projection error, such as this, indicates that the 3D points have been accurately reconstructed.

1.5 Putting It All Together

In this section, we combine all the components to perform a full 3D reconstruction of the scene. The process involves the following steps:

1. **Compute the Fundamental Matrix:** Using the ‘eight point‘ algorithm on the provided point correspondences.

2. **Compute the Epipolar Correspondences:** Use the 'epipolar correspondences' function to find the corresponding points in the second image.

3. **Compute the Essential Matrix:** Compute E using the previously calculated Fundamental matrix F and the intrinsic camera matrices.

4. **Compute Camera Matrices:** Calculate $M1$ and the four possible $M2$ matrices using 'camera2'.

5. **Triangulate Points:** Determine the correct $M2$ and use the 'triangulate' function to find the 3D points.

6. **Visualize the 3D Reconstruction:** Plot the 3D points using Matplotlib from different angles to ensure a consistent and accurate reconstruction.

After determining the correct camera matrix $M2$ from the four candidates generated by the essential matrix E, we extracted the extrinsic parameters, specifically the rotation matrix $R2$ and the translation vector $t2$.

The computed extrinsic parameters for the second camera ($M2$) are:

$$R2 = \begin{pmatrix} 9.65189813 \times 10^{-1} & -2.84559523 \times 10^{-2} & 2.59997852 \times 10^{-1} \\ 2.73497924 \times 10^{-2} & 9.99594929 \times 10^{-1} & 7.87192866 \times 10^{-3} \\ -2.60116537 \times 10^{-1} & -4.87018080 \times 10^{-4} & 9.65577107 \times 10^{-1} \end{pmatrix}$$

$$t2 = \begin{pmatrix} -1.00000000 \\ -0.02745167 \\ 0.08201567 \end{pmatrix}$$

These extrinsic parameters describe the relative orientation and position of the second camera with respect to the first camera. The rotation matrix $R2$ represents the rotation needed to align the second camera's coordinate system with that of the first camera. **TO DO** . The translation vector $t2$ specifies the translation from the first camera's position to the second camera's position in the 3D space.

Significance of the Extrinsic Parameters

The correct extrinsic parameters $R2$ and $t2$ are essential for accurately reconstructing the 3D structure of the scene. By using these parameters, we can ensure that the 3D points are correctly positioned relative to both cameras, leading to a consistent and accurate 3D model.

The extrinsic parameters $R1$ and $t1$ for the first camera are assumed to be:

$$R1 = \mathbf{I} \quad \text{(Identity Matrix)}$$

$$t1 = \mathbf{0} \quad \text{(Zero Vector)}$$

This implies that the first camera is at the origin of the coordinate system with no rotation applied.

Together, these parameters form the foundation of our 3D reconstruction pipeline, enabling us to generate accurate 3D models from 2D image correspondences.

1.5.1 Reconstruction Results

The three figures below are different views of the 3D reconstruction of the temple, chosen to exemplify both the fidelity in shape and depth to what one may intuitively surmise from the pictures.

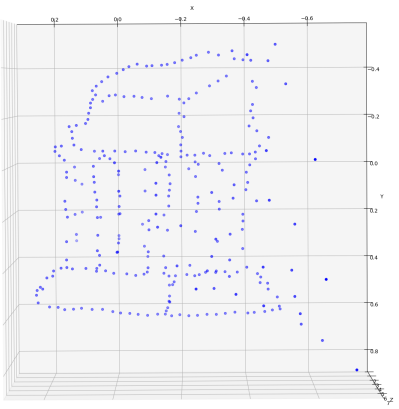

Figure 4: Front view. The shape of the reconstruction seems to be true to the shape of the temple from the images. Depth of points is not easy to infer.

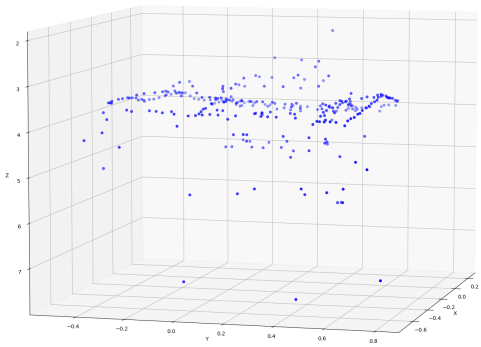

Figure 5: View from the left: depth is now better deducible. The steps and roof are the best represented features depth-wise in the reconstruction. Columns are less comprehensible.

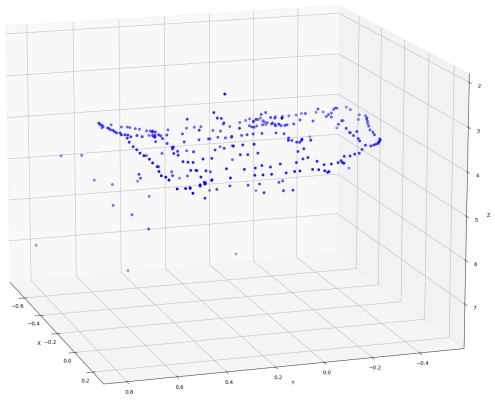

Figure 6: View from the right.