Structure from Motion

Computer Vision (RBE549) Project 2

Tejas Rane
MS Robotics Engineering
Worcester Polytechnic Institute
Email: turane@wpi.edu
Using 1 Late Day

Hrishikesh Pawar
MS Robotics Engineering
Worcester Polytechnic Institute
Email: hpawar@wpi.edu
Using 1 Late Day

***Abstract*—The report outlines the implementation of end-to-end pipeline for Structure from motion (SfM).**

I. Phase 1: Structure from Motion

The main goal of this project is to implement Structure from Motion (SfM) to reconstruct a three-dimensional scene from a series of two-dimensional images captured from different viewpoints, while simultaneously determining the camera poses relative to the scene. Our implementation of the SfM pipeline is as follows:

1) Feature Matching and Outlier rejection using RANSAC.
2) Estimating Fundamental Matrix.
3) Estimating Essential Matrix.
4) Estimate Camera Pose from Essential Matrix.
5) Check for Cheirality Condition using Triangulation
6) Non-Linear Triangulation
7) Perspective-n-Point
8) Bundle Adjustment

Each section provides a detailed explanation of the methodology employed and the resulting output of the images.

A. Feature Matching and Outlier rejection using RANSAC

Since Structure from Motion (SfM) relies heavily on good features and their matching, Scale-Invariant Feature Transform (SIFT) keypoints and descriptors are used. These feature correspondences were provided in the format of a `.txt` file. The data folder contains 4 matching files named `matching*.txt` where $\star$ refers to numbers from 1 to 5. For example, `matching3.txt` contains the matching between the third image and images that come after (such as images 4 and 5), i.e., $I_3 \leftrightarrow I_4, I_3 \leftrightarrow I_5$.

Each matching file, denoted as `matching`i`.txt`, outlines feature correspondences for the *i*th image as follows:

nFeatures: The total number of features in the i^{th} image. Each subsequent row details matches for a specific feature in this image.

Row Format: *[Number of matches] [R] [G] [B] [$u_{current}$] [$v_{current}$] [Image ID] [u_{match}] [v_{match}] ...*

For example, in `matching1.txt` with 3930 features, a row might read: '2 255 255 255 5.08304 116.978 3 49.0748 166.783', indicating two matches for a given feature, each with RGB values, coordinates in the current and matching images.

To ensure the reliability of these correspondences, we implemented a two-tier outlier rejection strategy. The initial layer employs RANSAC with a homography model to filter mismatches. Following this, a second layer of rejection utilizes the fundamental matrix RANSAC, further refining the matches. This dual-layer approach significantly enhances the accuracy of feature matching, laying a solid foundation for subsequent SfM processes.

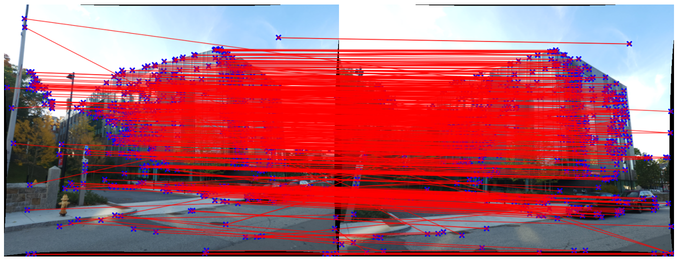
Fig. 1: Feature Matches before RANSAC

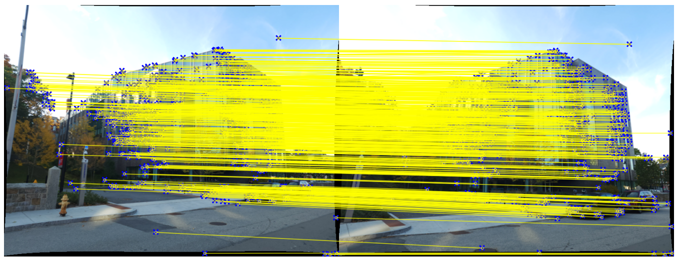
Fig. 2: Feature Matches after Homography RANSAC

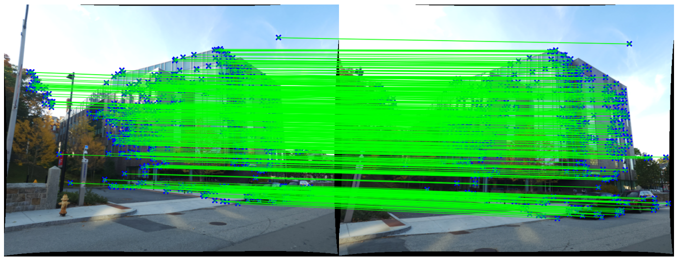
Fig. 3: Feature Matches after 8-point RANSAC

B. Estimating the Fundamental Matrix

The second phase in our computational pipeline focuses on estimating the Fundamental Matrix (F) between pairs of images, guided by the epipolar constraint. This estimation leverages the feature correspondences refined through homography RANSAC.

The epipolar constraint underpins the geometric relationship between corresponding points across two views given by:

$$x_2^T F x_1 = 0$$

where $x_1 = [u_1, v_1, 1]^T$ and $x_2 = [u_2, v_2, 1]^T$ represent the homogeneous coordinates of matching points in the first and second image, respectively.

To compute F, we employed the 8-point algorithm, supplemented with a Singular Value Decomposition (SVD) cleanup step to enforce a rank-2 constraint on F. RANSAC method is then applied to this preliminary estimation to robustly identify inliers and reject outliers. The threshold for distinguishing inliers from outliers is set by the condition:

$$|x_2^T F x_1| < \epsilon$$

where ϵ is a predefined threshold, x_2 are the coordinates of a feature in the second image, and x_1 are the coordinates of the corresponding feature in the first image. Once inliers are identified (from Fig. 3), a final estimation of the Fundamental Matrix is computed using all inliers and the 8-point algorithm. This final step, again incorporating an SVD cleanup, ensures the F matrix accurately reflects the underlying epipolar geometry between the two images. Fig 4 shows the respective epipolar lines derived from the computed Fundamental matrix.

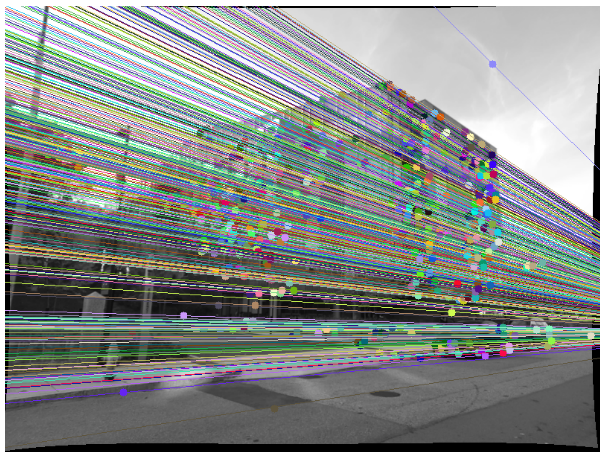

Fig. 4: Epipolar Lines

C. Estimating Essential Matrix

Following the estimation of the Fundamental Matrix (F) between a pair of images, we proceed to calculate the Essential Matrix (E) using the camera intrinsic parameters (K) obtained from camera calibration. The calculation of E is straightforward, defined by the equation:

$$E = K^T F K$$

However, to account for potential noise in the calculated E, we apply a refinement step using Singular Value Decomposition (SVD) and rank reduction. Specifically, we decompose E into UDV^T, and then adjust D to enforce the rank-2 constraint essential for the Essential Matrix. This adjustment is done by replacing D with a diagonal matrix:

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

D. Estimate Camera Pose from Essential Matrix.

Following the calculation of the Essential Matrix $E = UDV^T$, we define W as:

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

Utilizing W, we derive four possible camera configurations:

- $C_1 = U(:,3)$ and $R_1 = UWV^T$,
- $C_2 = -U(:,3)$ and $R_2 = UWV^T$,
- $C_3 = U(:,3)$ and $R_3 = UW^TV^T$,
- $C_4 = -U(:,3)$ and $R_4 = UW^TV^T$.

These configurations as seen in Fig. 5 represent the potential orientations and positions of the second camera relative to the first, calculated from the Essential Matrix decomposition.

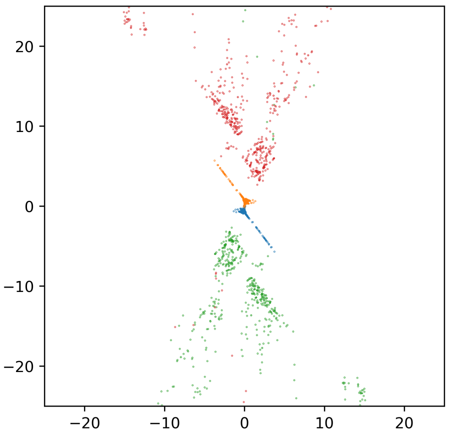

Fig. 5: Possible camera pose configurations

E. Check for Cheirality Condition using Triangulation

After computing four potential camera poses using the essential matrix, we focus on triangulating 3D points from two camera poses and correspondences through linear least squares.

Cheirality Condition: To identify the correct camera pose, we apply the cheirality condition, ensuring that reconstructed 3D points lie in front of both cameras. A point X is considered in front if $r_3(X - C) > 0$, where r_3 is the third row of the rotation matrix, representing the camera's z-axis. This step is crucial due to correspondence noise, and the optimal pose is determined by maximizing the number of points satisfying this condition.

F. Non-Linear Triangulation

With the camera poses and linearly triangulated points X, we refine their positions to minimize reprojection error, a more geometrically meaningful metric than the algebraic error minimized in linear triangulation. The minimization problem is formulated as:

$$\min_{x} \sum_{j=1,2} \left(\left(u_j - \frac{P_j^{T_1} \tilde{X}}{P_j^{T_3} \tilde{X}} \right)^2 + \left(v_j - \frac{P_j^{T_2} \tilde{X}}{P_j^{T_3} \tilde{X}} \right)^2 \right)$$

where j indexes the cameras, $\tilde{X}$ is in homogeneous coordinates, and P_i^T are rows of the camera projection matrix P. This nonlinear optimization starts with an initial guess from linear triangulation and was solved using `scipy.optimize.least_squares` in the Scipy library.

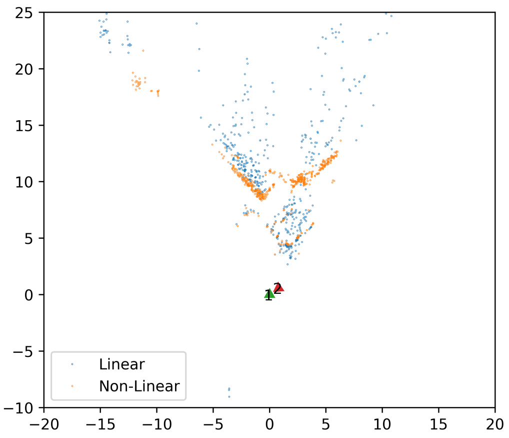

Fig. 6: Linear and Non-linear triangulations

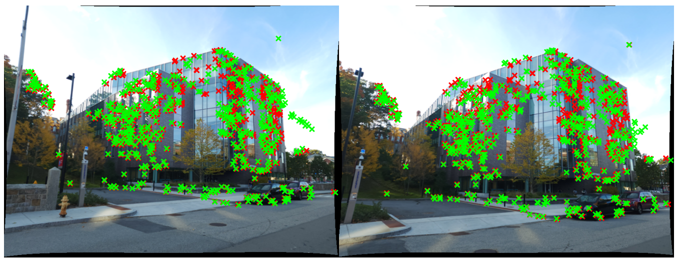

Fig. 7: Linear triangulation reprojection error

G. Perspective-n-Points (PnP)

With optimized world coordinates from two camera frames established, we extended our pose estimation to the other three frames using all available image data. The camera poses were computed through a series of steps:

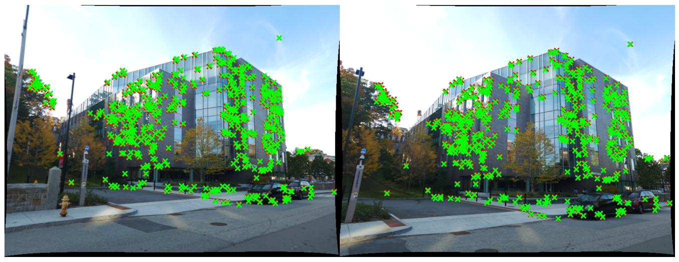

Fig. 8: Non-linear triangulation reprojection error

1) **Linear PnP:** A linear least squares problem was formulated with a minimum of 6 point correspondences (totaling 12 correspondences) to determine the new camera pose parameters R and T. Singular Value Decomposition (SVD) facilitated the extraction of these values from the last row of V^T obtained from the decomposition.
2) **PnP RANSAC:** Given the vulnerability of PnP to outliers, a robust camera pose estimation was implemented. Points yielding a reprojection error below a defined threshold ϵ were classified as inliers, contributing to a more stable pose estimation. Reprojection error was calculated using,

$$\min_{C,R} \sum_{i=1,J} \left(\left(u_j - \frac{P_j^{T_1} \tilde{X}_j}{P_j^{T_3} \tilde{X}_j} \right)^2 + \left(v_j - \frac{P_j^{T_2} \tilde{X}_j}{P_j^{T_3} \tilde{X}_j} \right)^2 \right)$$

3) **Nonlinear PnP:** We refined the camera pose by minimizing geometric loss through a nonlinear least squares method, analogous to the process used in triangulation.

The nonlinear PnP approach yielded relative camera poses for each frame, which, when plotted alongside the triangulated points, provided a comprehensive visualization as seen in Fig. 10.

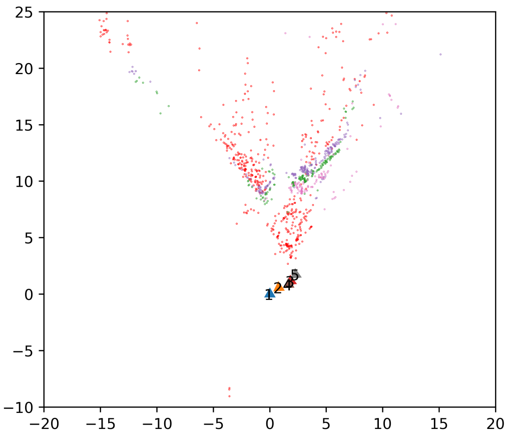

Fig. 9: Linear PnP with camera poses

H. Bundle Adjustment

Once we have the camera poses and the world coordinates, we proceed to refine them using Bundle Adjustment,

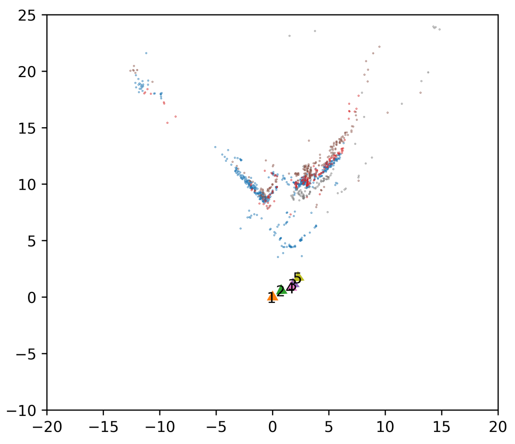

Fig. 10: Non-linear PnP with camera poses

TABLE I: Reprojection error (pixel)

Method	**Error**
Linear Triangulation	164.0128
Nonlinear Triangulation	11.4748
Linear PnP	21.6213
Nonlinear PnP	10.9486

optimizing the camera and point positions concurrently. We utilize Scipy's least square optimizer for Bundle Adjustment, similar to earlier sections. The visibility matrix informs which Jacobians are necessary, optimizing computational efficiency. The core objective is to minimize reprojection error, the discrepancy between observed image points and their corresponding projected 3D points. This is achieved by iterative adjustments to both camera poses and 3D points, with the process continuing until the error reaches a satisfactory minimum. Bundle Adjustment not only diminishes reprojection error but also enhances the accuracy and consistency of the 3D model. It can further assist in outlier detection and correction from the initial reconstructions.

We perform Bundle Adjustment following the Large Scale BA in SciPy. This article uses the Rodrigues' Rotation Formula to convert the rotation matrix to three element rotation vector. We observe that the results after bundle adjustment are almost similar to the ones before.

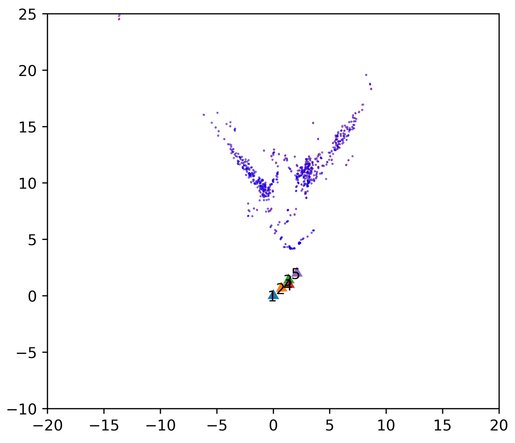

Fig. 11: Before (red) and After (blue) Bundle Adjustment, with camera poses