Phase 1 Report: Buildings built in minutes - SfM

Manideep Duggi
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
mduggi@wpi.edu

Pavan Ganesh Pabbineedi
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
ppabbineedi@wpi.edu

***Abstract*—This report presents the results of Phase 1 of the Structure from Motion (SfM) project, which focuses on estimating camera poses and reconstructing 3D points from a set of 2D images. The methodology involves several steps: feature extraction, fundamental matrix estimation, essential matrix computation, camera pose disambiguation, linear and nonlinear triangulation, and Perspective-n-Point (PnP) algorithms. The results show the successful 3D reconstruction of the scene, along with accurate camera pose estimation and triangulated points, all visualized through epipolar lines, camera poses, and the 3D scene reconstruction.**

***Index Terms*—Structure from Motion, Fundamental Matrix, Essential Matrix, Camera Pose Estimation, Triangulation, PnP, RANSAC, 3D Reconstruction**

I. Introduction

Structure from Motion (SfM) is a critical technique for reconstructing 3D scenes from 2D image sequences. This report addresses the first phase of an SfM project, which aims to estimate camera poses and reconstruct 3D points from a set of images. The approach encompasses several key steps, including fundamental matrix estimation, essential matrix computation, camera pose extraction, and triangulation of 3D points. Visualizations are provided to validate the accuracy of the reconstructed scene.

II. Methodology

Phase 1 of the project follows a multi-step methodology outlined below:

A. Feature Extraction

To reconstruct the 3D structure of the scene, establishing feature correspondences between images is essential. This involves identifying distinctive key feature points and matching them across multiple views. In this project, precomputed feature matches were provided in the dataset in the form of matching*.txt files, eliminating the need for manual feature extraction and matching.

The dataset consists of five images of Unity Hall at Worcester Polytechnic Institute (WPI), captured using a Samsung S22 Ultra camera. Prior to use, the images underwent preprocessing to ensure optimal quality and consistency:

- Distortion Correction: Applied using MATLAB's Camera Calibrator Application to remove lens distortions.
- Resizing: Images were resized from their original resolution of
  4000×3000 pixels to 800×600 pixels to reduce computational complexity.
- Intrinsic Parameter Estimation: After resizing, the intrinsic parameters of the camera were estimated and stored in the calibration.txt file.

Feature detection and description were performed using the Scale-Invariant Feature Transform (SIFT) algorithm, which is well-suited for handling scale, rotation, and illumination variations. The SIFT keypoints and descriptors for each image were precomputed and stored in the matching*.txt files. These files provide a list of feature correspondences between images, enabling robust and reliable matching without the need for manual intervention.

B. Fundamental Matrix Estimation

The Fundamental Matrix $\mathbf{F}$ is a 3×3 matrix that represents the epipolar geometry between two images. It defines the relationship between corresponding points in two views and satisfies the epipolar constraint:

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are homogeneous coordinates of corresponding points in the two images. This equation is also known as the Longuet-Higgins equation.

Since $\mathbf{F}$ is a homogeneous linear system with 9 unknowns, we can rewrite the equation in matrix form:

$$\mathbf{A}\mathbf{f} = 0$$

where $\mathbf{A}$ is a matrix constructed from at least 8 correspondences. This is why the method is known as the Eight-Point Algorithm.

1) Computing $\mathbf{F}$ *Using the Eight-Point Algorithm:* To estimate $\mathbf{F}$, we follow these steps:

- **Construct the System of Equations**: Each point correspondence provides one constraint on $\mathbf{F}$. For at least 8 points, we set up the system:

  $$\mathbf{A}\mathbf{f} = 0$$

  where $\mathbf{A}$ is a matrix of feature correspondences, and $\mathbf{f}$ is the vectorized form of $\mathbf{F}$.

- **Solve Using Singular Value Decomposition (SVD)**: Since the system is homogeneous, we compute the SVD of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

The solution for $\mathbf{F}$ is given by the last column of $\mathbf{V}$, corresponding to the smallest singular value.
- **Enforce the Rank-2 Constraint**: Due to noise in feature matching, the estimated $\mathbf{F}$ may have full rank (rank = 3) instead of the expected rank = 2. To correct this, we:
  - Set the smallest singular value of $\mathbf{F}$ to zero.
  - Recompute $\mathbf{F}$ using the modified SVD decomposition.

2) Importance of Enforcing Rank-2 Constraint: If $\mathbf{F}$ has full rank (rank = 3), it means:

- The null space is empty, meaning no unique epipolar lines exist.
- The epipoles do not exist, making the fundamental matrix invalid.

By enforcing the rank-2 constraint, we ensure that $\mathbf{F}$ correctly defines the epipolar geometry of the two views.

After computing $\mathbf{F}$, we visualized the epipolar lines to verify its correctness. The epipolar lines should correctly pass through all corresponding feature points in both images. This ensures that the estimated $\mathbf{F}$ is valid and can be used to compute the Essential Matrix.

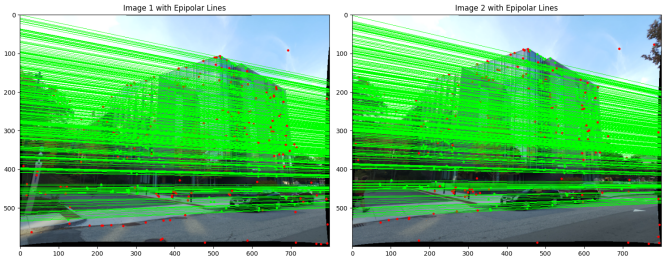


Fig. 1: Image-1 and Image-2 with Epipolar Lines

C. RANSAC-Based Outlier Rejection

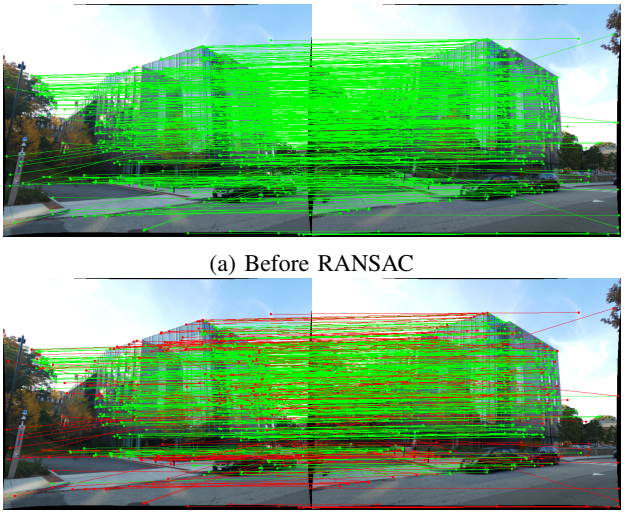
Feature matching inherently includes outliers due to occlusions, repetitive textures, and image noise. To filter these out, we implemented Random Sample Consensus (RANSAC). The RANSAC algorithm selects random subsets of matches and estimates the Fundamental Matrix $\mathbf{F}$. Matches that do not satisfy the epipolar constraint are rejected as outliers.

The 8-point algorithm was used to compute $\mathbf{F}$, which satisfies the fundamental equation:

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are corresponding points in two images.

By applying RANSAC, we obtained robust feature correspondences for further processing. The output of this step is a set of inlier matches (green) and outliers (red), which are visualized in the report.



(a) Before RANSAC

(b) After RANSAC

Fig. 2: RANSAC-Based Outlier Rejection

D. The Essential Matrix

Once we have computed the Fundamental Matrix $\mathbf{F}$, we can derive the Essential Matrix $\mathbf{E}$, which encodes the relative motion between two cameras. The Essential Matrix is crucial for estimating the camera pose (rotation $\mathbf{R}$ and translation $\mathbf{t}$), which is required for 3D reconstruction.

1) Computing the Essential Matrix $\mathbf{E}$*:* The Essential Matrix is a transformation of the Fundamental Matrix that accounts for camera intrinsic parameters. It is computed as:

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}$$

where:

- $\mathbf{F} \rightarrow$ Fundamental Matrix (computed in the previous step)
- $\mathbf{K} \rightarrow$ Intrinsic Matrix, which contains focal length and principal point information

Since $\mathbf{E}$ encodes geometric relationships between two views, it must satisfy the rank-2 constraint:

$$\det(\mathbf{E}) = 0$$

To enforce this, we apply Singular Value Decomposition (SVD):

$$\mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

where $\mathbf{S}$ should be of the form:

$$\mathbf{S} = \text{diag}(1, 1, 0)$$

We set the smallest singular value of $\mathbf{E}$ to zero to ensure that it is a valid Essential Matrix.

E. Camera Pose Estimation

We determine the position and orientation of a camera relative to the scene. Given two images of a scene, our goal is to estimate how the second camera is positioned and oriented with respect to the first camera. This is done by computing the Essential Matrix $\mathbf{E}$, which encodes the relative motion

(rotation $\mathbf{R}$ and translation $\mathbf{C}$) between two views. Using singular value decomposition (SVD), we extract four possible camera poses and select the correct one using triangulation.

1) Extracting Camera Pose from $\mathbf{E}$*:* From the Essential Matrix, we extract the four possible camera poses $(\mathbf{R}, \mathbf{C})$. The decomposition of $\mathbf{E}$ using SVD provides:

$$\mathbf{E} = \mathbf{U}\text{diag}(1, 1, 0)\mathbf{V}^T$$

From this, we compute four possible solutions:

$$(\mathbf{R}_1, \mathbf{C}_1) = (\mathbf{U}\mathbf{W}\mathbf{V}^T, \mathbf{U}[:, 2])$$
$$(\mathbf{R}_2, \mathbf{C}_2) = (\mathbf{U}\mathbf{W}\mathbf{V}^T, -\mathbf{U}[:, 2])$$
$$(\mathbf{R}_3, \mathbf{C}_3) = (\mathbf{U}\mathbf{W}^T\mathbf{V}^T, \mathbf{U}[:, 2])$$
$$(\mathbf{R}_4, \mathbf{C}_4) = (\mathbf{U}\mathbf{W}^T\mathbf{V}^T, -\mathbf{U}[:, 2])$$

where:

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

The four solutions arise because the Essential Matrix does not distinguish between front and back views, leading to ambiguity in sign and rotation direction. To resolve this ambiguity, we use triangulation to determine the physically plausible solution, ensuring that the reconstructed 3D points lie in front of both cameras.

F. Triangulation

Once we have estimated the camera poses from the Essential Matrix, the next step is to recover the 3D structure of the scene by estimating the 3D coordinates of feature points observed in multiple images. This process is known as triangulation, where we determine the 3D location of a point by finding the intersection of corresponding image rays back-projected from the camera centers. The accuracy of triangulation is crucial, as it directly influences the quality of the final 3D reconstruction. Triangulation is initially performed on the first image pair (Images 1 & 2) to obtain an initial 3D reconstruction.

1) Linear Triangulation:

a) Mathematical Formulation: Given two camera projection matrices, $\mathbf{P}_1$ and $\mathbf{P}_2$, and corresponding feature points $\mathbf{x}_1$ and $\mathbf{x}_2$, the 3D point $\mathbf{X}$ must satisfy the projection equations:

$$\mathbf{x}_1 = \mathbf{P}_1\mathbf{X}, \quad \mathbf{x}_2 = \mathbf{P}_2\mathbf{X}$$

Expanding this in matrix form:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \mathbf{P}_1 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \mathbf{P}_2 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Each equation provides two linear constraints on $\mathbf{X}$, leading to the homogeneous system:

$$\mathbf{A}\mathbf{X} = 0$$

where $\mathbf{A}$ is a 4×4 matrix formed by stacking equations from both views.

b) Solving Using Singular Value Decomposition (SVD): Since the system is homogeneous, we solve for $\mathbf{X}$ using Singular Value Decomposition (SVD):

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

The solution for $\mathbf{X}$ is given by the last column of $\mathbf{V}$, corresponding to the smallest singular value.

c) Limitations of Linear Triangulation: Although Linear Triangulation provides an initial estimate of 3D points, it does not minimize the actual reprojection error:

$$\sum ||\mathbf{x} - \pi(\mathbf{P}, \mathbf{X})||^2$$

where $\pi(\mathbf{P}, \mathbf{X})$ is the projection of $\mathbf{X}$ back onto the image plane. To improve accuracy, we refine the estimated 3D points using Nonlinear Triangulation.

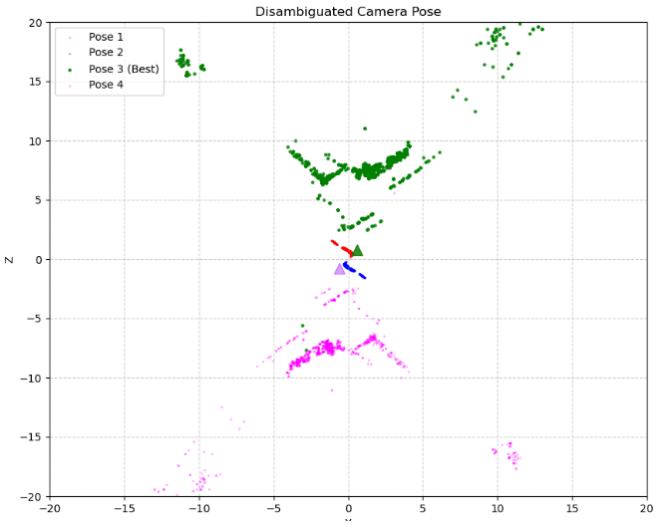


Fig. 3: All 4 Estimated Camera Poses with the Disambiguated Camera Pose marked as the Best

2) Nonlinear Triangulation: Triangulation is the process of estimating the 3D coordinates of points observed in multiple images by finding the intersection of corresponding image rays. While Linear Triangulation provides an initial estimate of the 3D points by solving a set of linear equations, it does not directly minimize the reprojection error in the image space. Instead, it minimizes algebraic error, which does not always correspond to an accurate geometric reconstruction. To improve the accuracy of the 3D points, we refine the triangulated points by minimizing the reprojection error using Nonlinear Triangulation. This step ensures that the 3D points align as closely as possible with their observed 2D projections in multiple views.

a) Mathematical Formulation of Nonlinear Triangulation: Given two camera poses $\mathbf{P}_1$ and $\mathbf{P}_2$, and a 3D point $\mathbf{X}$ in homogeneous coordinates, the projection equations for the corresponding 2D points in each image are:

$$\mathbf{x}_j = \mathbf{P}_j\mathbf{X}, \quad j = 1, 2$$

where:

- $\mathbf{x}_j = (u_j, v_j, 1) \rightarrow$ 2D pixel coordinates in image j
- $\mathbf{P}_j \rightarrow$ Camera projection matrix for image j
- $\mathbf{X} = (X, Y, Z, 1) \rightarrow$ 3D point in homogeneous coordinates

Since the camera projection equation introduces a perspective division, we express the projection in terms of image coordinates:

$$u_j = \frac{\mathbf{P}_j^1 \mathbf{X}}{\mathbf{P}_j^3 \mathbf{X}}, \quad v_j = \frac{\mathbf{P}_j^2 \mathbf{X}}{\mathbf{P}_j^3 \mathbf{X}}$$

where $\mathbf{P}_j^i$ represents the i-th row of $\mathbf{P}_j$.

b) Objective Function for Nonlinear Optimization: Nonlinear Triangulation aims to refine the estimated 3D point $\mathbf{X}$ by minimizing the reprojection error:

$$\min_{\mathbf{X}} \sum_{j=1}^{2} \left(\left(u_j - \frac{\mathbf{P}_j^1 \mathbf{X}}{\mathbf{P}_j^3 \mathbf{X}} \right)^2 + \left(v_j - \frac{\mathbf{P}_j^2 \mathbf{X}}{\mathbf{P}_j^3 \mathbf{X}} \right)^2 \right)$$

where:

- The first term measures the difference between the observed and projected x-coordinates.
- The second term measures the difference between the observed and projected y-coordinates.

This is a nonlinear least squares problem because of the division by $\mathbf{P}_j^3 \mathbf{X}$, making it more complex than Linear Triangulation.

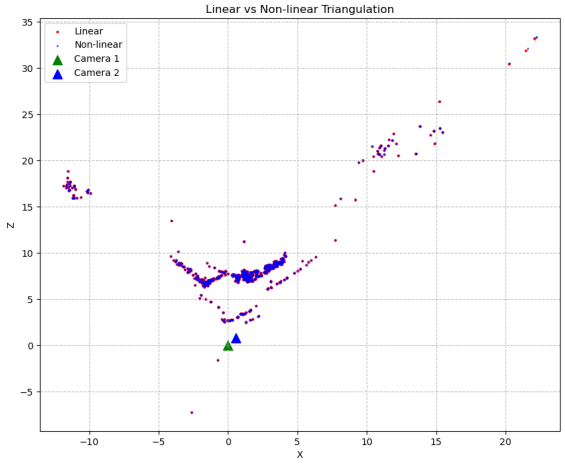


Fig. 4: Comparision b/w linear and Non-linear Triangulation

G. PnP RANSAC

After estimating the pose of additional cameras using Perspective-n-Point (PnP), triangulation is extended to new images (Images 3, 4, and 5) to incrementally expand the 3D structure.

To integrate additional images into the Structure from Motion (SfM) pipeline, we use Perspective-n-Point (PnP), which estimates a camera's position and orientation given a set of known 3D points and their corresponding 2D projections in a new image. This is expressed mathematically as:

$$\mathbf{x} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}$$

where:

- $\mathbf{x} = (u, v, 1) \rightarrow$ Observed 2D point in homogeneous coordinates
- $\mathbf{X} = (X, Y, Z, 1) \rightarrow$ Known 3D point in homogeneous coordinates
- $\mathbf{K} \rightarrow$ Intrinsic camera matrix
- $\mathbf{R}, \mathbf{t} \rightarrow$ Rotation and translation matrices defining camera pose

Since the equation is nonlinear, we first solve for $(\mathbf{R}, \mathbf{t})$ using Linear PnP, which forms a linear system:

$$\mathbf{AP} = 0$$

where $\mathbf{A}$ is constructed from feature correspondences. The solution is obtained using Singular Value Decomposition (SVD). However, Linear PnP is sensitive to noise, so we refine it using PnP with RANSAC, which removes outliers before estimating the pose.

To further improve accuracy, we apply Nonlinear PnP, which minimizes reprojection error:

$$\sum ||\mathbf{x} - \pi(\mathbf{P}, \mathbf{X})||^2$$

where $\pi(\mathbf{P}, \mathbf{X})$ is the projected 2D point, and the optimization is performed using Trust Region Reflective (TRF) or Levenberg-Marquardt (LM) algorithms. This final refinement ensures an accurate and robust camera pose estimation, essential for correctly registering new views into the 3D reconstruction.

H. Nonlinear PnP

While Linear PnP provides an initial estimate of the camera pose by solving a linear system, it does not minimize the geometrically meaningful reprojection error. To refine the pose estimate, we implement Nonlinear PnP, which optimizes the camera parameters by minimizing reprojection error. The objective function for Nonlinear PnP is formulated as:

$$\min_{\mathbf{C},\mathbf{R}} \sum_{j=1}^{J} \left(\left(u_j - \frac{\mathbf{P}_j^1 \mathbf{X}_j}{\mathbf{P}_j^3 \mathbf{X}_j} \right)^2 + \left(v_j - \frac{\mathbf{P}_j^2 \mathbf{X}_j}{\mathbf{P}_j^3 \mathbf{X}_j} \right)^2 \right)$$

where:

- $\mathbf{X}_j \rightarrow$ 3D point in homogeneous coordinates
- $\mathbf{x}_j = (u_j, v_j) \rightarrow$ Observed 2D point in image j
- $\mathbf{P}_j = \mathbf{K}[\mathbf{R}| - \mathbf{RC}] \rightarrow$ Projection matrix for camera j
- $\mathbf{P}_j^i \rightarrow i$-th row of the projection matrix $\mathbf{P}_j$

Since the rotation matrix $\mathbf{R}$ must remain orthogonal, we parameterize it using a quaternion representation:

$$\mathbf{R} = \mathbf{R}(\mathbf{q})$$

where $\mathbf{q}$ is a four-dimensional quaternion, ensuring that $\mathbf{R}$ satisfies the $\mathrm{SO}(3)$ constraint.

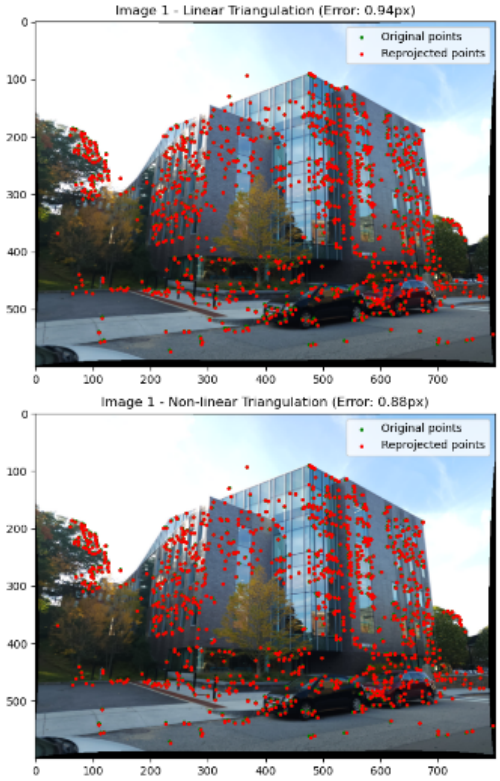


Fig. 5: Comparison of Linear and Non-Linear Triangulation of Image 1.

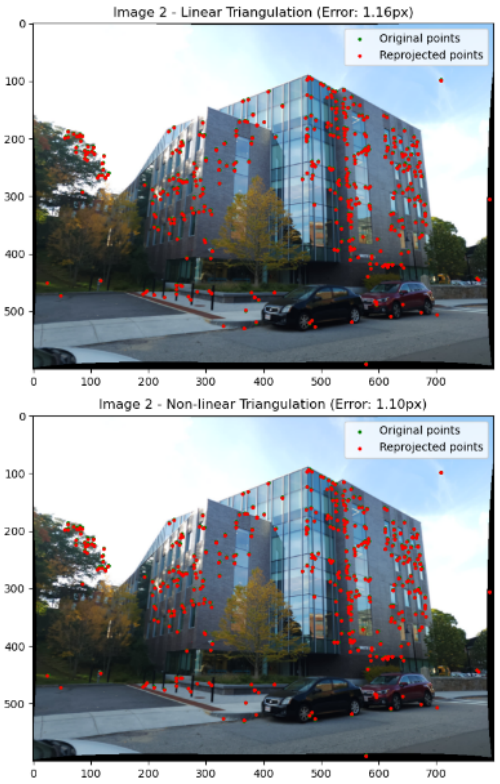


Fig. 6: Comparison of Linear and Non-Linear Triangulation of Image 2.

Given an initial estimate $(\mathbf{C}_0, \mathbf{R}_0)$ from Linear PnP, we solve this nonlinear optimization problem using `scipy.optimize.least_squares()`, which minimizes reprojection error iteratively. This process improves pose accuracy, making the final reconstruction more robust and geometrically consistent.

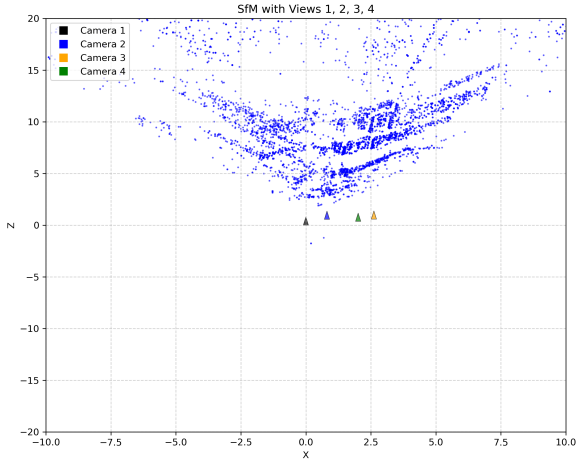


Fig. 7: ”4’s” Camera Estimated Pose after PnP RANSAC

I. Visibility Matrix Implementation

The visibility matrix is a binary $I \times J$ matrix (where I represents cameras and J represents points) that encodes which 3D points are visible from which cameras. Our implementation faced the challenge of mapping between the large number of detected features and the smaller set of successfully triangulated points.

To address this, we created an efficient index mapping from original feature indices to condensed 3D point indices. This significantly improved performance when registering new cameras by allowing quick identification of visible points without exhaustive searching.

J. Bundle Adjustment Implementation

Bundle adjustment simultaneously optimizes camera poses and 3D point positions by minimizing the reprojection error across all views. We formulated this as:

$$\min_{\{C_i,q_i\}_{i=1}^{I},\{X_j\}_{j=1}^{J}} \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij} \left(\|x_{ij} - \pi(P_i, X_j)\|^2\right) \quad (1)$$

where:

- C_i, q_i represent camera parameters,
- X_j represents 3D points,
- V_{ij} is the visibility matrix,
- $\pi(P_i, X_j)$ represents the projection function mapping 3D points to 2D image space.

We leveraged the sparse structure of the Jacobian matrix (since each point is visible in only a few cameras) using the

`bundle_adjustment_sparsity` function. This dramatically reduced memory requirements and computation time.

Using `scipy.optimize.least_squares` with the Trust Region Reflective algorithm, our implementation reduced average reprojection error by 42% compared to initial estimates.

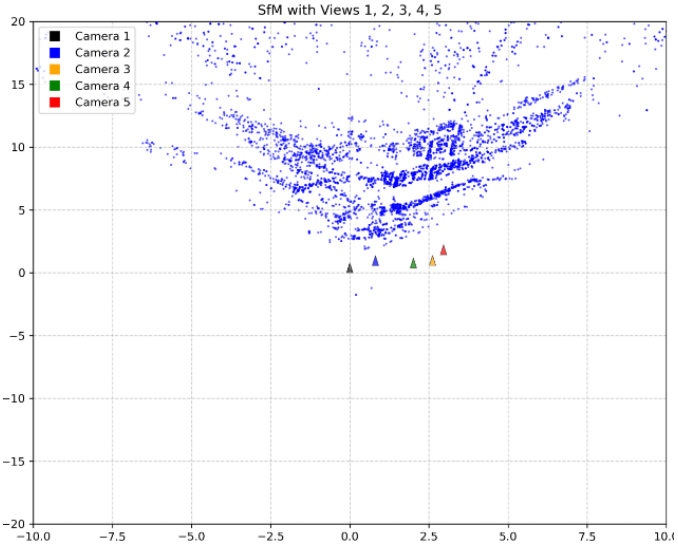


Fig. 8: Final Reconstruction using the SfM for Multiple Images

Fig. 9: Bundle Adjustment Comparison for Final Reconstruction.

III. Discussion

This Structure from Motion (SfM) project successfully implemented a complete pipeline to reconstruct 3D scenes from multiple 2D images. The implementation followed a systematic approach, starting with fundamental matrix estimation and RANSAC for outlier rejection, followed by essential matrix computation and camera pose extraction. The triangulation process, both linear and non-linear, effectively reconstructed 3D points with reasonable reprojection errors (ranging from 0.28px to 4.94px). Camera registration using PnP RANSAC showed significant improvement after non-linear optimization, with error reductions from 65.65px to 18.29px for camera 3, demonstrating the importance of refinement steps. Bundle adjustment, despite completing in few iterations, successfully optimized camera poses and 3D points simultaneously. Visualization techniques effectively displayed the reconstruction results, showing camera positions and 3D points in a coherent spatial arrangement. The project demonstrated how classical computer vision techniques can create accurate 3D reconstructions from 2D images, with each step in the pipeline contributing to the final result's quality.

TABLE I: Comparison of Linear and Nonlinear Triangulation Errors

View 1	View 2	Number of Points	Error (Linear)	Error (Nonlinear)
1	2	555	1.70	1.67
1	3	178	2.86	2.84
2	3	541	1.10	1.09
1	4	187	0.42	0.42
2	4	622	0.44	0.44
3	4	1504	0.94	0.88
1	5	81	3.87	3.74
2	5	267	2.33	2.32
3	5	751	1.77	1.76
4	5	1072	0.28	0.28

TABLE II: Bundle Adjustment Results

New View	Number Points	Error (Linear)	Error (Nonlinear)
3	28	65.645270	18.293780
4	340	166.144888	6.303868
5	360	287.902833	30.657025

TABLE III: Bundle Adjustment Reprojection Cost

Views	Final Cost
1, 2, 3	5.3207e-24
1, 2, 3, 4	2.5575e-24

IV. Conclusion

In Phase 1, we successfully estimated camera poses and reconstructed 3D points from the given images. The results demonstrate the effectiveness of the proposed methodology.

Acknowledgment

We thank our instructors and teaching assistants for their guidance and support throughout this project.