RBE549 Project 2 - Structure from Motion

Using 1 Late Day

Colin Balfour
BS/MS in Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
Email: cbalfour@wpi.edu

Simran Chauhan
MS in Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
Email: schauhan@wpi.edu

***Abstract*—This report presents the implementation of a Structure-from-Motion (SfM) pipeline. The SfM pipeline analyzes multiple 2D images to determine camera positions and generates 3D point clouds, effectively reconstructing the scene's 3D structure.**

I. Phase 1: : Structure from Motion

Structure from motion is a computer vision method that reconstructs 3D scenes from 2D images by estimating camera positions and 3D coordinates that align with observed 2D image points. This problem requires simultaneous solving for camera positions and 3D points while managing noise and outliers. In this phase, we build a basic structure from motion pipeline to determine camera positions and 3D points from 2D images. We begin with the mathematical foundations and algorithms for processing image pairs, then expand our approach to handle multiple images. We address the limitations of the basic approach by proposing a more robust, scalable pipeline. The section concludes with results from applying this pipeline to Unity Hall at WPI.

A. Estimating the Fundamental Matrix

The first step in the Structure from Motion (SfM) pipeline is to estimate the fundamental matrix. The fundamental matrix describes the epipolar geometry between two images, relating the points in one image to the corresponding epipolar lines in the other image. This relationship is captured in the following mathematical expression, referred to as the epipolar constraint:

$$\mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i = 0 \tag{1}$$

where $\mathbf{x}_i$ and $\mathbf{x}_i'$ are the homogeneous coordinates of the corresponding points in the two images, and $\mathbf{F}$ is the fundamental matrix. This equation is then expanded to the following form:

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0 \tag{2}$$

which can be transformed into a the following linear equation:

$$\begin{bmatrix} x_i'x_i & x_i'y_i & x_i' & y_i'x_i & y_i'y_i & y_i' & x_i & y_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{3}$$

Note that we need at least 8 point correspondences to solve for Equation 3, as each correspondence only contributes 1 constraint to the system, as the epipolar constraint is a scalar equation.

To solve this system, we use singular value decomposition (SVD) to find the least squares solution to the linear equation. The fundamental matrix is then constructed from the least squares solution. However, a valid fundamental matrix must have rank 2 due to its geometric properties. We enforce this rank-2 constraint by setting the smallest singular value to zero.Finally, we normalize the fundamental matrix to improve numerical stability.

B. Match Outlier Rejection via RANSAC

The eight-point algorithm is sensitive to noise and requires a sufficient number of point correspondences for accurate results.To remove outliers from point correspondences generated by feature descriptors, we implement the RANSAC algorithm to obtain a more accurate fundamental matrix estimate.

This approach systematically evaluates multiple Fundamental matrices and selects the one with the maximum number of inlier points. The pseudo-code below outlines our process for determining the optimal fundamental matrix from corresponding points identified through SIFT matching. Algorithm 1 illustrates our RANSAC pipeline in detail.

Further, we also implemented the Sampson distance metric in our RANSAC algorithm for fundamental matrix estimation. This approach provides a geometrically meaningful error measure that better approximates true reprojection error while remaining computationally efficient. Using this distance measure rather than simple algebraic error, our implementation effectively identifies and removes outlier correspondences between image pairs. The output of the fundamental mattix RANSAC with Sampson distance is shown in Figure 1.

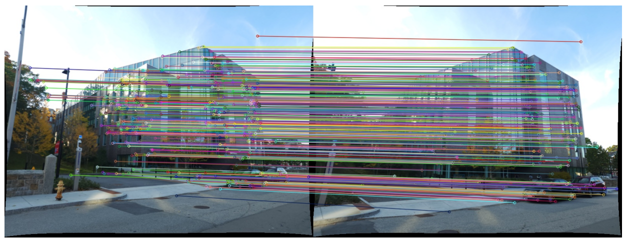
Fig. 1: Matched features for image 1 and image 2 (RANSAC with Sampson distance)

Algorithm 1 Robust Fundamental Matrix Estimation

```
1: Input: Matched point correspondences $\{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^n$
2: Parameters: Number of iterations $N$, inlier threshold $\tau$
3: Output: Fundamental matrix $\mathbf{F}$, set of inliers
4: Extract point correspondences from matches
5: Initialize best inlier count $best\_count \leftarrow 0$
6: Initialize best fundamental matrix $\mathbf{F}_{best} \leftarrow \emptyset$
7: for $i = 1$ to $N$ do
8:     Randomly select 8 point correspondences
9:     Estimate $\mathbf{F}_i$ using the 8-point algorithm
10:    Initialize inlier set $inliers \leftarrow \emptyset$
11:    for each point correspondence $(\mathbf{x}_j, \mathbf{x}'_j)$ do
12:        Compute epipolar constraint error $e_j = |\mathbf{x}'^T_j \mathbf{F}_i \mathbf{x}_j|$
13:        if $e_j < \tau$ then
14:            Add $j$ to $inliers$
15:        end if
16:    end for
17:    if $|inliers| > best\_count$ then
18:        $best\_count \leftarrow |inliers|$
19:        $\mathbf{F}_{best} \leftarrow \mathbf{F}_i$
20:        $best\_inliers \leftarrow inliers$
21:    end if
22: end for
23: Re-estimate $\mathbf{F}$ using all points in $best\_inliers$
24: return $\mathbf{F}$, $best\_inliers$
```

C. *Estimate Essential Matrix from Fundamental Matrix*

In our SfM pipeline, we first compute the essential matrix from the fundamental matrix. The essential matrix encodes the geometric relationship between corresponding points in calibrated stereo views. We obtain it using:

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K} \tag{4}$$

Noise in the fundamental matrix estimation typically results in an essential matrix that deviates from its theoretical properties. We correct this by enforcing its algebraic constraints through SVD:

$$\mathbf{E} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^T \tag{5}$$

D. *Estimate Camera Pose from Essential Matrix*

After estimating the essential matrix, the next step in the visual reconstruction pipeline is to determine the camera pose. The essential matrix $\mathbf{E}$ represents the relative pose between two cameras, and can be factorized into rotation and translation components as:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} \tag{6}$$

where $\mathbf{R}$ is the rotation matrix, $\mathbf{t}$ is the translation vector, and $[\mathbf{t}]_\times$ denotes the skew-symmetric matrix of $\mathbf{t}$.

The decomposition of the essential matrix is not unique and results in four possible camera poses. To resolve this ambiguity, we use the following method:

1) Compute the four possible camera poses using the essential matrix
2) Triangulate 3D points for each of the four pose configurations
3) Select the pose that satisfies the chirality condition (points must be in front of both cameras)

The chirality condition ensures that the reconstructed 3D points lie in front of both cameras. We verify this by examining the depth of each 3D point relative to each camera view. Camera pose configurations where points appear behind either camera are discarded. Mathematically, the chirality condition is expressed as:

$$\mathbf{R}(:,3)^T(\mathbf{X} - \mathbf{C}) > 0 \tag{7}$$

where $\mathbf{R}(:,3)$ represents the third column of the rotation matrix, $\mathbf{X}$ denotes the coordinates of the 3D point and $\mathbf{C}$ is the center location of the camera.

Step 1 uses the singular value decomposition (SVD) of $\mathbf{E}$:

$$\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{8}$$

Define matrix $\mathbf{W}$ as:

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

The four possible camera poses are then computed using the following equations:

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^T \qquad \mathbf{C}_1 = \mathbf{U}(:,3) \tag{10}$$
$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}\mathbf{V}^T \qquad \mathbf{C}_2 = -\mathbf{U}(:,3) \tag{11}$$
$$\mathbf{R}_3 = \mathbf{U}\mathbf{W}^T\mathbf{V}^T \qquad \mathbf{C}_3 = \mathbf{U}(:,3) \tag{12}$$
$$\mathbf{R}_4 = \mathbf{U}\mathbf{W}^T\mathbf{V}^T \qquad \mathbf{C}_4 = -\mathbf{U}(:,3) \tag{13}$$

where $\mathbf{U}(:,3)$ denotes the third column of matrix $\mathbf{U}$, $\mathbf{R}_i$ represents the rotation matrix, and $\mathbf{C}_i$ is the camera center for the i-th pose hypothesis.

The detailed implementation of the second step involving triangulation will be discussed in the next section.

E. Linear Triangulation

Given two camera poses $(\mathbf{R}_i, \mathbf{C}_i)$ and the corresponding 2D points in the images $(\mathbf{x}_i, \mathbf{x}_i')$, we can calculate $\mathbf{X}$, the world point of each correspondence. To solve the triangulation problem, we start with the pinhole projection model:

$$\begin{bmatrix}\mathbf{x}\\1\end{bmatrix} = \alpha\mathbf{P}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix} \tag{14}$$

where $\mathbf{P}$ is the camera projection matrix:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{15}$$

This can be rewritten in homogeneous form as:

$$\begin{bmatrix}\mathbf{x}\\1\end{bmatrix} \times \mathbf{P}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix} = \mathbf{0} \tag{16}$$

We stack this equation for each camera pose and its corresponding image point, then solve for $\mathbf{X}$ using singular value decomposition. Figs. 2, 3 and 4 illustrate the result of this process.

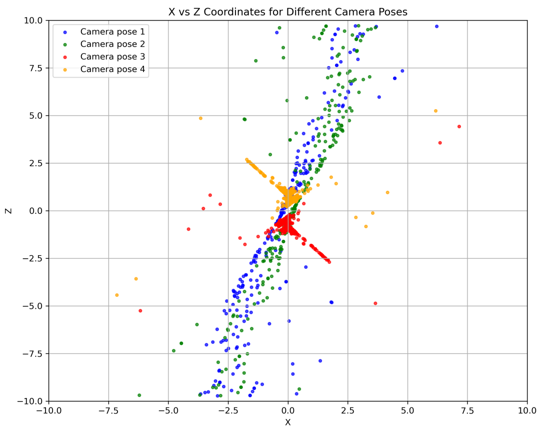

Fig. 2: Initial Linear Triangulation for four poses

F. Non-Linear Triangulation

Given two camera poses and linearly triangulated points $\mathbf{X}$, the locations of the 3D points that minimize the reprojection error can be refined.

Linear triangulation minimizes the algebraic error. However, the reprojection error is a geometrically meaningful error and can be computed by measuring the error between measurement and projected 3D point:

$$\min_{\mathbf{X}} \sum_{j=1,2} \left(u^j - \frac{P_1^{jT}\mathbf{X}}{P_3^{jT}\mathbf{X}}\right)^2 + \left(v^j - \frac{P_2^{jT}\mathbf{X}}{P_3^{jT}\mathbf{X}}\right)^2 \tag{17}$$

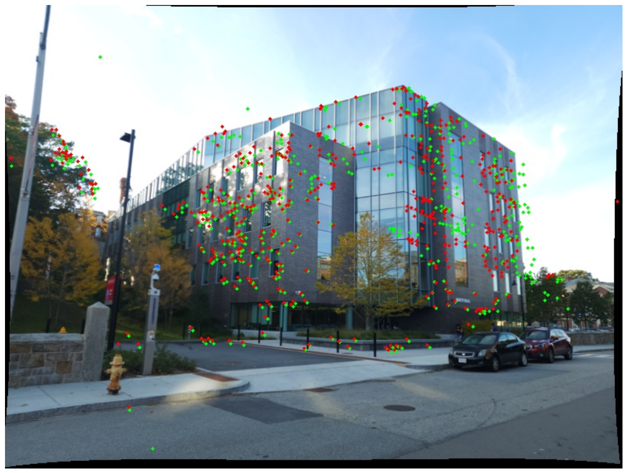

Fig. 3: Linear Triangulation Reprojection Error(1.298) | Img1

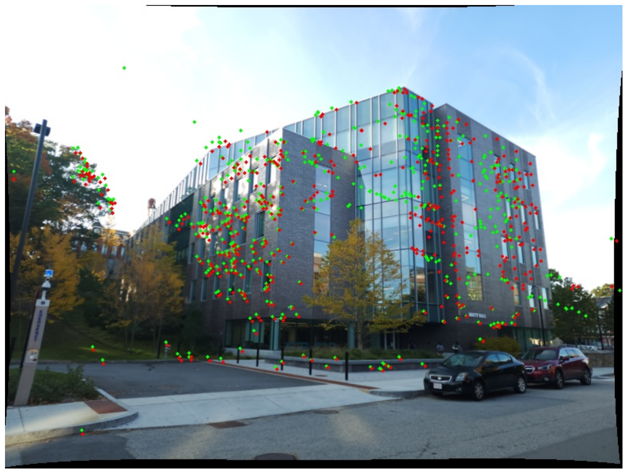

Fig. 4: Linear Triangulation Reprojection Error(2.552) | Img2

Here, j is the index of each camera, $\bar{\mathbf{X}}$ is the homogeneous representation of $\mathbf{X}$. P_i^T is each row of camera projection matrix, P. This minimization is highly nonlinear due to the divisions. The initial guess of the solution, $\mathbf{X}_0$, is estimated via the linear triangulation to minimize the cost function.

To solve this, we implement the `scipy.optimize.least_squares` from the SciPy library. This implementation uses algorithms such as Levenberg-Marquardt which are well-suited for addressing non-linear reprojection error minimization. The results are shown in Figs. 5 and 6.

This method improves the accuracy of the 3D points, as presented in the results section in Table I.

G. Structure from Motion Pipeline

The complete SfM workflow for processing a pair of images can be summarized as follows:

1) Apply RANSAC to identify and eliminate outlier point correspondences while computing the fundamental matrix

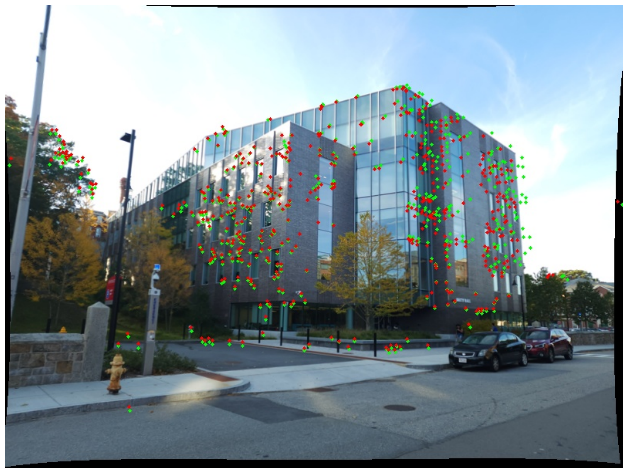

Fig. 5: Non-Linear Triangulation Reprojection Error(0.1080) | Img1

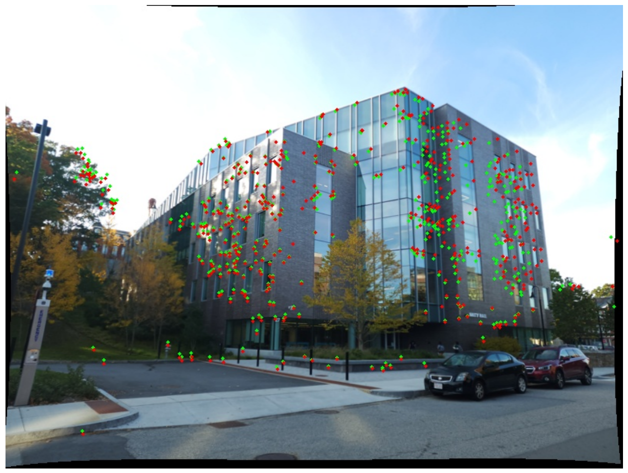

Fig. 6: Non-Linear Triangulation Reprojection Error(0.1018) | Img2

TABLE I: Triangulation reprojection error for each pair of views, before and after nonlinear optimization.

View 1	View 2	Error (Linear)	Error (Nonlinear)
1	2	1.9250	0.1049
1	3	3.25751	5.88682
1	4	2.65231	0.07601
1	5	4.17573	0.47376
2	3	3.37742	0.05506
2	4	4.10149	0.09746
2	5	2.51806	0.35619
3	4	1.66990	0.00308
3	5	0.07568	0.52498
4	5	1.21428	0.06885

2) Calculate the essential matrix based on the estimated fundamental matrix
3) Extract the four possible camera pose configurations from the essential matrix
4) Perform linear triangulation to obtain 3D points for each potential camera pose
5) Resolve the pose ambiguity by applying the chirality condition
6) Refine the 3D point coordinates through non-linear triangulation to reduce reprojection error

This pipeline produces reliable 3D reconstructions, as demonstrated in Figure 5, which shows sample output from our implementation.

H. Perspective-n-Points (PnP) and PnP RANSAC

To extend our pipeline to multiple images, we must determine the camera pose for each image. This is accomplished through the Perspective-n-Points (PnP) approach, which estimates a camera pose from a set of 3D points and their corresponding 2D projections.

The PnP method uses variable numbers of point correspondences, but we chose to use $n = 6$ to simplify the implementation while maintaining accuracy.

The initial step in the PnP algorithm involves creating a linear formulation to approximate the camera pose parameters. This is accomplished by transforming Equation 10 into the following form:

$$\begin{bmatrix} X, Y, Z, 1, 0, 0, 0, 0, -xX, -xY, -xZ, -x \\ 0, 0, 0, 0, X, Y, Z, 1, -yX, -yY, -yZ, -y \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \mathbf{0} \tag{18}$$

This equation is stacked n times, one for each point in the linear PnP.

The linear system is then solved using SVD, yielding the projection matrix $\mathbf{P}$. We then decompose $\mathbf{P}$ into rotation $\mathbf{R}$ and translation $\mathbf{t}$:

$$\mathbf{R} = \mathbf{K}^{-1}\mathbf{P}(:, 1:3) \tag{19}$$

$$\mathbf{t} = \mathbf{K}^{-1}\mathbf{P}(:, 4) \tag{20}$$

To ensure that the rotation matrix maintains proper orthonormal properties, we apply SVD cleanup:

$$\mathbf{U}, \mathbf{D}, \mathbf{V} = \text{SVD}(\mathbf{R}) \tag{21}$$

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T \tag{22}$$

$$\mathbf{t} = \mathbf{t}/\mathbf{D}_{11} \tag{23}$$

Finally, if $\det(\mathbf{R}) = -1$, we flip the signs of both $\mathbf{R}$ and $\mathbf{t}$.

This procedure provides a decent estimate of camera pose, but is highly susceptible to noise and outliers. To address this vulnerability, we implement the RANSAC algorithm with 6 randomly selected points per iteration. Inliers are identified based on reprojection error. The results of this process are illustrated in Figure 7.

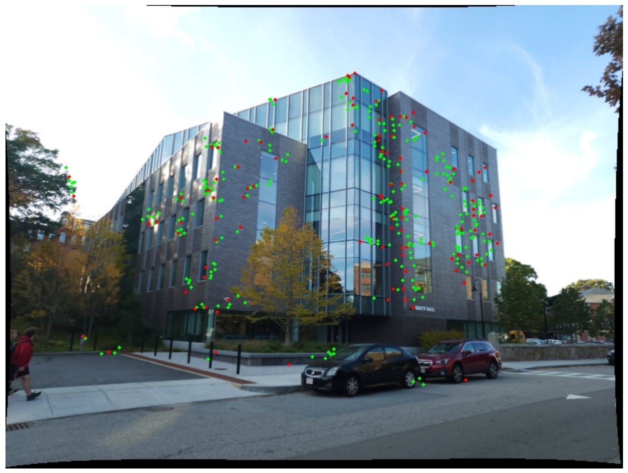

Fig. 7: Linear PnP Reprojection Error(16.3787) | Img4

I. NonLinear PnP

Although the linear PnP RANSAC algorithm provides a reasonable initial estimate, further refinement is necessary for optimal results. We implement the Levenberg-Marquardt algorithm to improve the camera pose estimation. This non-linear optimization technique minimizes the reprojection error defined by the geometric disparity between projected points and their actual image coordinates:

$$\min_{\mathbf{C},\mathbf{R}} \sum_{j=1,J} \left(u^j - \frac{P_1^{jT}\bar{\mathbf{X}}_j}{P_3^{jT}\bar{\mathbf{X}}_j} \right)^2 + \left(v^j - \frac{P_2^{jT}\bar{\mathbf{X}}_j}{P_3^{jT}\bar{\mathbf{X}}_j} \right)^2 \tag{24}$$

In this formulation, $\bar{\mathbf{X}}$ represents $\mathbf{X}$ in homogeneous coordinates, while P_i^T indicates the i-th row of the projection matrix P, defined as $P = KR[I_{3\times3} \; -\mathbf{C}]$. To maintain rotation matrix orthogonality constraints, we adopt quaternion representation for rotations, expressing $\mathbf{R}$ as $\mathbf{R}(q)$ where q denotes a 4D quaternion vector. This transforms our optimization problem to:

$$\min_{\mathbf{C},q} \sum_{j=1,J} \left(u^j - \frac{P_1^{jT}\bar{\mathbf{X}}_j}{P_3^{jT}\bar{\mathbf{X}}_j} \right)^2 + \left(v^j - \frac{P_2^{jT}\bar{\mathbf{X}}_j}{P_3^{jT}\bar{\mathbf{X}}_j} \right)^2 \tag{25}$$

The optimization faces significant nonlinearity challenges from both the perspective division operations and quaternion parameterization. Effective convergence requires initializing with the linear PnP solution $(\mathbf{C}_0, \mathbf{R}_0)$. We implement this minimization using numerical optimization algorithms from the SciPy package, such as `scipy.optimize.leastsq` or `scipy.optimize.least_squares`.he results of this process are illustrated in Figure 8.

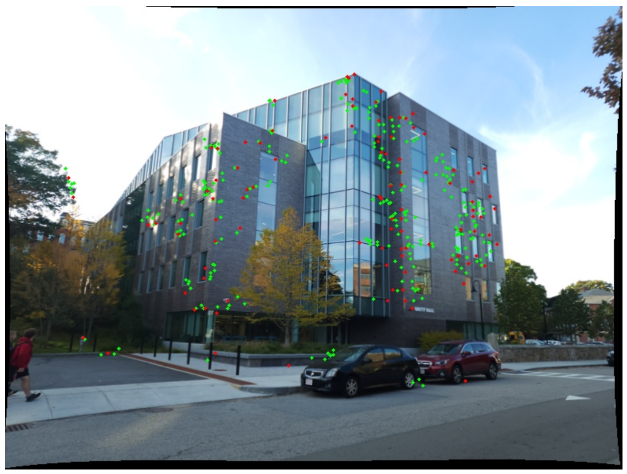

Fig. 8: Non-Linear PnP Reprojection Error(7.11) | Img4

J. Structure from Motion Pipeline for Multiple Image Sets

With the introduction of PnP techniques into our framework, we can extend our SfM pipeline to handle multiple images:

1) Perform the basic SfM pipeline for a single pair of images.
2) For each additional image in the sequence:
    a) Identify all feature points in the current image that correspond to already reconstructed 3D world-points
    b) Compute the camera pose using PnP RANSAC with these 2D-3D correspondences
    c) Enhance pose accuracy through non-linear PnP optimization
    d) Perform triangulation for all feature points that match with the previous images but don't have corresponding 3D coordinates
    e) Incorporate the newly triangulated 3D points into the global reconstruction

The results of applying this pipeline to five different views of Unity Hall are illustrated in Figure 9, showing the reconstruction prior to bundle adjustment.

The PnP reprojection error for each added view is shown in Table II.

K. Bundle Adjustment

The final step in our SfM pipeline is bundle adjustment. This process jointly refines all camera poses and 3D point positions. It minimizes the total reprojection error across all image points

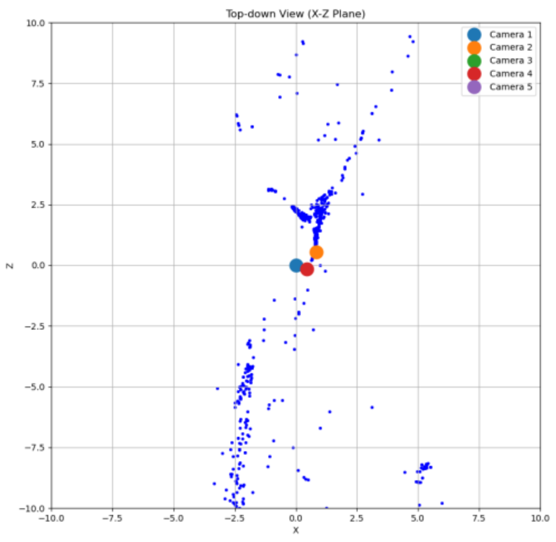

Fig. 9: Output from 5 different pose of Unity Hall, without BA (Bundle Adjustment)

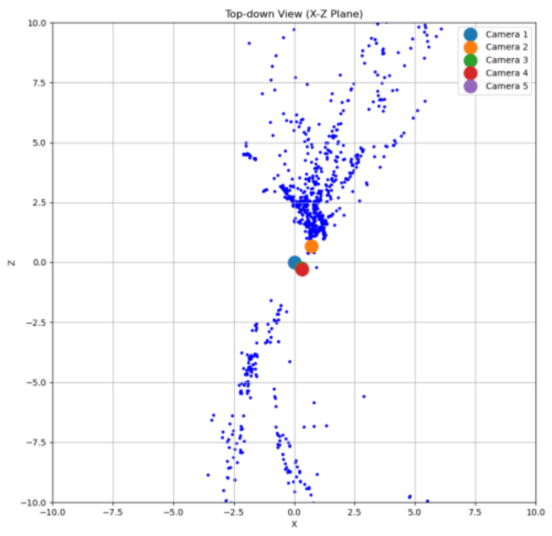

Fig. 10: Output after BA (bundle adjustment) for 5 images of Unity Hall

TABLE II: PnP reprojection error for each added view, before and after nonlinear optimization.

New View	Error (Linear)	Error (Nonlinear)
3	15.8674	6.8934
4	16.3787	7.110252
5	13.3285	7.067279

TABLE III: Bundle adjustment reprojection error for all views.

Views	Error
1, 2, 3	0.89645
1, 2, 3, 4	0.65381
1, 2, 3, 4, 5	0.87643

that have a known 3D world point. The optimization problem can be expressed as:

$$\min_{\{C_i\},\{R_i\},\{X_j\}} \sum_{i=1}^{m} \sum_{j \in V_i} d(\mathbf{x}_{ij}, P_i \mathbf{X}_j)^2 \tag{26}$$

where C_i and R_i are the camera center and rotation quaternion for the i-th camera. X_j is the j-th world point. P_i is the projection matrix for the i-th camera. V_i represents the set of 3D points whose world point is visible in the i-th image.

This is a large optimization problem. It involves many parameters. The solution requires computing a sparse structure for the Jacobian matrix. We derive this from V. The Trust Region Reflective algorithm solves the resulting problem. The results of this process is shown in Fig 10.

This completes the final stage of our full pipeline:

1) Perform the SfM pipeline for a single pair of images on the first two images
2) For each subsequent image:
    a) Determine all image points which have a known 3D world point
    b) Estimate the camera pose using PnP RANSAC
    c) Refine the camera pose with non-linear PnP
    d) Triangulate all matched points without existing 3D coordinates
    e) Add the new 3D points to the known world points
3) Perform bundle adjustment for all images/cameras

Table III shows the bundle adjustment reprojection error for each set.

II. Conclusion

In this project, we presented a complete Structure from Motion pipeline for 3D reconstruction from multiple uncalibrated images. Our approach integrates established techniques with several refinements to enhance accuracy and computational efficiency.

Our experimental results on multiple datasets, including the Unity Hall sequence, validate the effectiveness of our pipeline. The comparison between linear and non-linear methods highlights the importance of geometric error minimization over algebraic approaches.