Course: CS436 – Computer Vision Fundamentals

Group 32

Abstract

This project presents a two-phase implementation of a classical Structure-from-Motion (SfM) pipeline. Phase 1 performs two-view reconstruction using feature matching, Essential Matrix estimation, relative pose recovery, and triangulation. Phase 2 extends this baseline to an incremental multi-view reconstruction, where each additional image is registered to the growing 3D map using 2D–3D correspondences and a reprojection-error–minimizing camera pose estimate. Our implementation emphasizes the mathematical foundations of multiview geometry rather than code-level detail. A basic virtual tour viewer was also implemented. For global consistency and to mitigate drift from the incremental stage, the final point cloud and poses were exported from Agisoft Metashape. GitHub repository: `https://github.com/blndeyes/Group32---CV-project`.

1 Introduction

SfM aims to recover camera poses and a 3D point cloud from multiple images by leveraging epipolar geometry and projective reconstruction. Phase 1 initiates a baseline reconstruction from two images; Phase 2 performs incremental mapping by registering new views to existing 3D structure. The implementation follows the classical pipeline: feature extraction, pose estimation, triangulation, and map expansion. Our focus is on mathematical correctness and geometric consistency rather than user-interface components. For the final virtual tour, Agisoft-derived poses were used to provide a stable point cloud for use in the Three.js visualization.

2 Methodology: Structure from Motion

2.1 Feature Extraction and Correspondence

Across all phases, feature detection is performed using scale- and rotation-invariant descriptors. For two-view reconstruction, image correspondences $\{x_i \leftrightarrow x_i'\}$ are obtained after ratio-test filtering. These pairs serve as inputs to epipolar geometry estimation.

2.2 Essential Matrix and Relative Pose

Given calibrated points $\tilde{x} = K^{-1}x$, the Essential Matrix satisfies

$$\tilde{x}'^{\top} E \tilde{x} = 0, \qquad E = [t]_{\times} R,$$

where R and t represent the relative orientation and translation between two cameras. A robust estimator selects the E that maximizes inlier support under the Sampson distance.

The physically correct $[R|t]$ is chosen via cheirality: reconstructed points must satisfy positive depth in both views.

2.3 Triangulation

For projection matrices $P_1 = K[I|0]$ and $P_2 = K[R|t]$, each 3D point X satisfies

$$x \sim P_1 X, \qquad x' \sim P_2 X.$$

Triangulation is computed by solving a homogeneous linear system of the form $AX = 0$, followed by normalization to recover Cartesian coordinates. Points with negative depth or large reprojection error are discarded.

2.4 Incremental SfM: Camera Registration

For each new image, features are matched to previously reconstructed 3D points. A camera with pose $[R|t]$ is estimated by minimizing the reprojection error

$$\min_{R,t} \sum_i \|x_i - \pi\left(RX_i + t\right)\|^2,$$

where $\pi(\cdot)$ denotes projection via the intrinsic matrix K. A RANSAC variant ensures robustness against mismatches. Once a pose is estimated, new points are triangulated between this camera and others that observe the same features.

3 Virtual Tour Implementation

3.1 Point Cloud Reconstruction

Due to implementation challenges in Phase 2, specifically difficulties achieving stable multi-view triangulation and managing accumulated drift, the final sparse point cloud was generated using Agisoft Metashape. Agisoft performed dense feature matching across all 32 input images followed by bundle adjustment to jointly optimize camera poses and 3D points. The resulting point cloud contained approximately 150,000 points with RGB color, and camera poses were exported as 4×4 transformation matrices $M = [R|t]$ in a Three.js-compatible coordinate system.

3.2 View Graph and Navigation

A view graph $G = (V, E)$ was constructed where vertices represent camera poses and edges connect cameras within distance threshold $d_{\max} = 1.5$ meters. Two cameras i and j were connected if $\|C_i - C_j\| < d_{\max}$, where C_i and C_j are camera centers. This distance-based criterion naturally captured the sequential structure while permitting local shortcuts.

3.3 Camera Transitions

Smooth navigation between cameras was achieved through interpolation. Given start pose $[R_s|t_s]$ and target $[R_e|t_e]$, the camera pose at parameter $\alpha \in [0,1]$ was computed as: **Position:** $C(\alpha) = \text{lerp}(C_s, C_e, \alpha)$ using linear interpolation. **Orientation:** $q(\alpha) = \text{slerp}(q_s, q_e, \alpha)$ using spherical linear interpolation of quaternions derived from R_s and R_e. A cubic ease-in-out function was applied to $\alpha(t)$ for natural acceleration/deceleration, with total transition duration of 1.2 seconds.

3.4 Implementation Architecture

The virtual tour was implemented in JavaScript using Three.js r128. The application loads the PLY point cloud, renders camera markers color-coded by navigation state (red for current, orange for connected, green for others), and enables user interaction through raycasting for marker clicking and keyboard shortcuts for navigation. Orbit controls allow manual inspection around each viewpoint. Image cross-fading was not implemented; navigation occurs entirely within the 3D point cloud environment, which provides sufficient geometric context for spatial understanding.

4 Results

The two-view baseline consistently produced stable Essential Matrix estimates and physically valid relative poses. Triangulation yielded a sparse yet structurally correct point cloud. In the incremental extension, additional images were successfully registered when they shared sufficient overlap with the initial map. Reconstruction quality degraded in texture-poor areas or when the baseline between views was too narrow, reflecting known limitations of classical SfM. The final virtual tour application successfully integrated 32 camera poses with the point cloud, enabling smooth navigation with real-time rendering at 60 FPS.

5 Discussion and Challenges

This project highlighted the geometric sensitivity of SfM algorithms. The most significant challenges are summarized below.

Feature Distribution and Epipolar Geometry

Stable estimation of E depended heavily on the spatial distribution of correspondences. Matches concentrated on planar regions occasionally made E rank-deficient or ambiguous. Mathematically, this occurs because the set of correspondences does not sufficiently constrain the bilinear form $\tilde{x}'^{\top} E \tilde{x}$. Retaking images with stronger parallax resolved this issue.

Cheirality and Depth Recovery

Early failures in pose selection were traced to small translation magnitudes. When $\|t\|$ was too small, reconstructed depths became numerically unstable: the system $AX = 0$ yielded points with inconsistent sign. Enforcing a cheirality-based pose choice and discarding low-baseline image pairs corrected this.

Incremental Drift and Reprojection Error

In incremental SfM, small pose errors accumulate as new views are added. Mathematically, each camera is placed using

$$\arg\min_{R,t} \|x_i - \pi(RX_i + t)\|^2,$$

making the solution sensitive to noise in both x_i and X_i. Our notebooks included explicit reprojection-error checks, which removed outliers before pose refinement. This pruning noticeably reduced drift, especially in later images.

For this reason, Agisoft Metashape was used for the final reconstruction to obtain a globally optimized point cloud. The manual incremental pipeline demonstrated fundamental SfM concepts but lacked a proper bundle adjustment working algorithm, resulting in drift accumulation. Agisoft's joint optimization over all cameras and points provided the geometric consistency required for smooth virtual tour navigation.

3D–2D Association Errors

A frequent implementation challenge was ensuring consistency between old 3D points and new feature descriptors. Incorrect associations corrupted the reprojection-error cost function and caused divergent pose estimates. Tracking descriptor indices carefully resolved the issue. This step demonstrated the mathematical principle that pose estimation assumes correct correspondence between (X_i, x_i) pairs; any mistake propagates multiplicatively.

Textureless Surfaces

On low-texture walls, correspondences were sparse, causing the PnP optimization to become under-constrained. From a theoretical standpoint, the Jacobian of the reprojection function loses rank when features do not span the image plane sufficiently. Adding images with stronger texture and wider baselines solved the problem.

Absence of Global Optimization

Without bundle adjustment, accumulated drift was expected. Our lightweight pruning strategy acted as a partial regularizer, but a full solution would require jointly minimizing:

$$\sum_{i,j} \|x_{ij} - \pi(R_i X_j + t_i)\|^2$$

over all cameras and points. The project scope excluded this, but our empirical observations aligned with the theoretical necessity of this global optimization.

Coordinate System Conventions

Integrating SfM poses with Three.js required careful attention to coordinate conventions. Agisoft exports rotations as $R_{\text{world}\to\text{camera}}$, while Three.js cameras require proper viewing direction alignment. The correct conversion involves negating the Z-column of the rotation matrix:

$$R_{\text{corrected}} = \begin{bmatrix} R_{00} & R_{01} & -R_{02} \\ R_{10} & R_{11} & -R_{12} \\ R_{20} & R_{21} & -R_{22} \end{bmatrix}$$

This transformation accounts for the difference in camera viewing conventions between Agisoft and Three.js, where cameras look along the negative Z-axis. This ensures that reconstructed camera poses render with the correct orientation in the virtual tour viewer.

6 Conclusion

The project successfully implemented the core mathematical components of a classical SfM pipeline. Two-view geometry was used to initialize the map, and incremental pose estimation was performed through reprojection-error minimization. While the final point cloud reconstruction relied on Agisoft due to accumulated drift, the implementation process provided insights into multiview geometry foundations. The virtual tour successfully demonstrated navigation through reconstructed poses using lerp for position and slerp for orientation. The challenges encountered provided strong intuition for the mathematics underlying SfM systems and their sensitivity to feature quality, baseline geometry, and correspondence accuracy.