3D Scene Reconstruction and Virtual Tour

Maha Amer
24030001@lums.edu.pk

Sania Atif
24030009@lums.edu.pk

Abstract

This project implements Structure from Motion (SfM) and visualization pipeline to reconstruct a sparse 3D scene from 2D photographs and enable an interactive virtual tour. ORB features are extracted and matched across images to establish correspondences, and the essential matrix is used to recover initial two-view geometry, providing an initial camera pose and seed 3D point cloud. Additional views are incrementally added using Perspective-n-Point (PnP) to register new camera poses, at the same time bundle adjustment refines both 3D points and camera parameters by minimizing reprojection error. Rather than creating a dense mesh, the sparse point cloud serves as a geometric scaffold for smooth transitions between high resolution input images. The resulting system delivers a consistent, globally optimized 3D reconstruction and an interactive virtual navigation experience, demonstrating the effectiveness of multiview geometry in practical applications. **Project Repository:** https://github.com/MahaAmer2000/CS-436_Project_Group22

1 Introduction

The key contributions of this report are:

- **Dataset Collection:** Captured multiple images of each wall by translating the camera in uniform steps, ensuring complete coverage of the wall at a consistent height.
- **Comprehensive SfM Pipeline:** Developed a complete Structure from Motion system that recovers 3D geometry and camera motion from 2D images without prior calibration.
- **Robust Feature Matching and Two-View Geometry:** Implemented effective feature matching, essential matrix decomposition, and two-view geometry estimation to initialize the reconstruction.
- **Incremental Camera Registration and Global Optimization:** Added new views incrementally using Perspective-n-Point (PnP) and refined both 3D points and camera poses with bundle adjustment, providing quantitative metrics on error reduction.
- **Photosynth Style Virtual Tour:** Created an interactive viewer leveraging the sparse point cloud and camera poses, enabling smooth navigation between camera views with interpolated camera motion and image cross-fading while maintaining a sense of 3D structure.
- **360 Degree Panoramic Visualization:** Constructed a 360 degree panoramic view by stitching images together and mapping them onto a cylindrical surface, creating an immersive navigable environment.

2 Methodology: Structure from Motion

2.1 Mathematical Formulation

The reconstruction problem is formulated as minimizing reprojection error:

$$\min_{\{R_i,t_i\},\{X_j\}} \sum_{i=1}^{n} \sum_{j \in V_i} \left\| \pi\Big(K[R_i \mid t_i]X_j\Big) - x_{ij} \right\|^2 \quad (1)$$

where

- R_i and t_i are the rotation and translation of camera i,
- $X_j \in \mathbb{R}^3$ is the 3D position of point j,
- K is the intrinsic calibration matrix of the camera,
- $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ denotes the perspective projection function,
- $x_{ij} \in \mathbb{R}^2$ is the observed 2D projection of point j in image i,
- V_i is the set of 3D points visible in camera i.

2.2 Feature Matching and Two-view geometry

Feature Detection: ORB (Oriented FAST and Rotated BRIEF) features are extracted using `cv2.ORB_create()` with parameters, $n_{\text{features}} = 3000$, scale Factor = 1.2, and $n_{\text{levels}} = 8$, in order to get robust multi-scale feature detection.

Feature Matching: ORB descriptors are matched using a Brute-Force Hamming matcher with Lowe's ratio test:

$$\frac{d(f_1, f_2^{(1)})}{d(f_1, f_2^{(2)})} < 0.75,$$

where $d(\cdot, \cdot)$ denotes the Hamming distance, and $f_2^{(1)}, f_2^{(2)}$ are the first and second nearest neighbors of f_1.

Essential Matrix Estimation: The essential matrix E is the foundation of this 3D reconstruction from 2D images. Through this matrix we can map every pixel in one image to a line in the other image. It is computed using cv2.FindEssentialMat with RANSAC. RANSAC is executed with success probability p=0.999 and threshold δ= 1.0 pixels.

Essential Matrix Estimation: Camera pose is the relative position of the second camera with respect to the first one and so on. The essential matric is decomposed into four candidate pose pairs:

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

Where $[\mathbf{t}]_\times$ is the skew-symmetric matrix. Out of the 4 possibilities, the physically correct camera pose is selected by enforcing positive depth for triangulated points on both views.

Triangulation: Triangulation is the calculation of the 3D position obtained from the intersection of 2 rays from 2 separate cameras looking at the same scene. We have used cv2.triangulatePoints in which 3D points are reconstructed using linear triangulation:

$$\begin{bmatrix} x_1 P_1 & x_2 P_2 \end{bmatrix} X = 0,$$

where the projection matrices are

$$P_1 = K[R_1 \mid t_1], \quad P_2 = K[R_2 \mid t_2].$$

2.3 Incremental Multi-view Reconstruction

Observation Tracking: For each triangulated point, all 2D measurements are stored in a nested dictionary.

$$\text{point\_observations} = \left\{ \begin{array}{l} \text{point\_idx} : (u, v), \\ \text{camera\_idx} : (u, v) \end{array} \right\}$$

This helps bundle adjustment to use true reprojection errors rather than frame-to-frame approximations.

Perspective-n-Point (PnP): New camera poses are estimated by minimizing

$$\min_{R,t} \sum_{j=1}^{m} \left\| \pi\big(K[R \mid t] X_j\big) - x_j \right\|^2$$

If PnP fails due to insufficient inliers (< 15), a fallback is used to approximate camera pose based on the previous camera pose:

$$t_{\text{new}} = t_{\text{prev}} + R_{\text{prev}} \triangle t$$

Incremental Triangulation: New 3D points are triangulated between each newly added camera and the previous image. Triangulated points are filtered using:

- **Depth Constraint:** $0.5 < \|X\| < 20.0$ (distance between camera and point)
- **Spatial Bounds:** $|x|, |y| < 10,\ z < 10$ (approximate region for points)

2.4 Bundle adjustment

The bundle adjustment implementation updates all the calculated camera poses and 3D points in order to minimize the error we made during calculation of these values.

Optimization problem: The full bundle adjustment minimizes:

$$\min_{\{r_i, t_i\}, \{X_j\}} \sum_{(i,j,x_{ij}) \in \mathcal{O}} \left\| \pi\Big(K[R(r_i) \mid t_i] X_j\Big) - x_{ij} \right\|^2$$

where $\mathcal{O}$ denotes the set of all camera-point observations.

2.5 Error metrics:

Reprojection error is reported as root mean square error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \|e_k\|^2},$$

where

$$e_k = \pi(P_i X_j) - x_{ij}$$

for each observation.

3 Interactive Virtual Tour

3.1 Merging Walls:

Loading and Transforming Camera Poses: Camera positions are loaded and transformed using homogeneous coordinates for alignment with merged point clouds.

$$\mathbf{C} = \{\mathbf{c}_i\}, \quad \mathbf{c}_i \in \mathbb{R}^3$$

$$\mathbf{c}_i' = \mathbf{T} \cdot \begin{bmatrix} \mathbf{c}_i \\ 1 \end{bmatrix}, \quad \mathbf{T} \in \mathbb{R}^{4 \times 4}$$

Loading and Densifying Point Clouds:

Point clouds $\mathbf{P}_i$ are loaded. Normals are estimated and downsampled to a target number of points N_{target} for efficient processing:

$$\mathbf{P}_i \subset \mathbb{R}^3, \quad |\mathbf{P}_i| \leq N_{\text{target}}$$

Coarse Alignment with RANSAC:
Rough alignment between source and target clouds is computed by estimating a transformation $\mathbf{T}_{\text{RANSAC}}$ based on FPFH (Fast Point Feature Histograms) feature matching:

$$\mathbf{T}_{\text{RANSAC}} = \arg\min_{\mathbf{T}} \sum_j \|\mathbf{T}\mathbf{p}_j^s - \mathbf{p}_j^t\|$$

Fine Alignment using ICP:
ICP refines the alignment with transformation $\mathbf{T}_{\text{ICP}}$ to minimize point-to-point or point-to-plane distances:

$$\mathbf{T}_{\text{ICP}} = \arg\min_{\mathbf{T}} \sum_j \|\mathbf{T}\mathbf{p}_j^s - \mathbf{p}_{\text{closest}}^t\|^2$$

Predefined Transformation for Wall2:
Manual rotation $\mathbf{R}_y(\theta)$ and translation $\mathbf{t}$ are applied to Wall2 to ensure proper alignment after repeated failures of automatic alignment techniques.

$$\mathbf{T}_{\text{Wall2}} = \begin{bmatrix} \mathbf{R}_y(\theta) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$

$$\mathbf{t} \in \mathbb{R}^3$$

Merging Walls and Cameras: Aligned point clouds are combined:

$$\mathbf{P}_{\text{merged}} = \sum_i \mathbf{T}_i \, \mathbf{P}_i$$

Camera indices and wall IDs are updated for global tracking which helped in mapping of images later in three.js app.

Computing Connected Cameras: 3 Nearest-neighbor relationships are computed to define camera connections (View Graph) for the virtual tour:

$$\text{connected\_cams}_i = \text{NN}_k(\mathbf{c}_i)$$

Outlier Removal and Visualization: Statistical outliers are removed and the merged cloud with cameras is visualized to verify alignment and coverage.

In the current implementation, camera rotations are not explicitly transformed in Python to match the merged point cloud, and per-wall ICP transformation matrices are not saved separately. Both of these steps are optional according to the project manual. The camera rotations are instead handled on the web side using Spherical Linear Interpolation (Slerp) in Three.js during navigation, which ensures smooth and visually correct orientation transitions between viewpoints. Similarly, per-wall transformations are unnecessary because all point clouds and camera positions are already merged into a single global coordinate system in Python. This approach maintains accurate spatial relationships between walls and cameras, and the resulting virtual tour demonstrates continuous navigation, coherent 3D structure, and proper camera transitions.

3.2 Photosynth Style Virtual Tour

Camera Graph and Node Setup Each camera pose from `merged_cameras.json` is represented as a node containing position, rotation (quaternion), and its connected cameras. These nodes form a view graph that defines navigation paths between walls.

Image Overlay and Preloading High resolution wall images are preloaded and displayed as overlays for each camera node to ensure smooth transitions without delays.

Smooth Transitions (LERP + SLERP) Camera transitions between nodes use LERP for interpolating position and SLERP for interpolating orientation. This provides smooth, natural motion similar to Photosynth's visual navigation.

Crossfade Blending A gradual opacity change between consecutive images ensures seamless visual blending during movement between walls, avoiding abrupt frame switches.

Navigation Controls Arrow keys and mouse drag cursor enables navigation through connected cameras. Raycasting detects clickable nodes, and a custom cursor provides interactive feedback.

3.3 360° Panoramic View

Cylindrical Projection Setup A panoramic texture (`full_room_panorama_new.jpg`) is mapped onto a large cylindrical geometry to create a full room 360° environment within the Three.js scene.

Orbit Controls Camera rotation is managed by `OrbitControls` with panning and zoom disabled, allowing users to freely explore the entire room view in 360°.

Mode Switching A toggle button switches between Photosynth mode and panoramic mode. The system dynamically hides or reveals the corresponding 3D or 360° elements to ensure smooth transitions.

4 Experimental Results

4.1 Intrinsic matrix

Intrinsic matrix was approximated using the following formula. (Note: we tried extracting it using python EXIF, but failed.)

$$K = \begin{bmatrix} 1.2w & 0 & w/2 \\ 0 & 1.2h & h/2 \\ 0 & 0 & 1 \end{bmatrix}$$

4.2 Feature matching performance

Feature matching was done across consecutive image pairs. On average, the system detected and matched 127±42 ORB features per pair. Lowe's ration test was applied with threshold 0.75 and 0.80.

4.3 Reconstruction results

Multi-view reconstruction The incremental mapping pipeline has not only successfully registered all 12 camera views (for wall 1) but also reconstructed a total of 1,847 points in 3D. Camera registration was 10 out of 12 times successful (83% success rate).

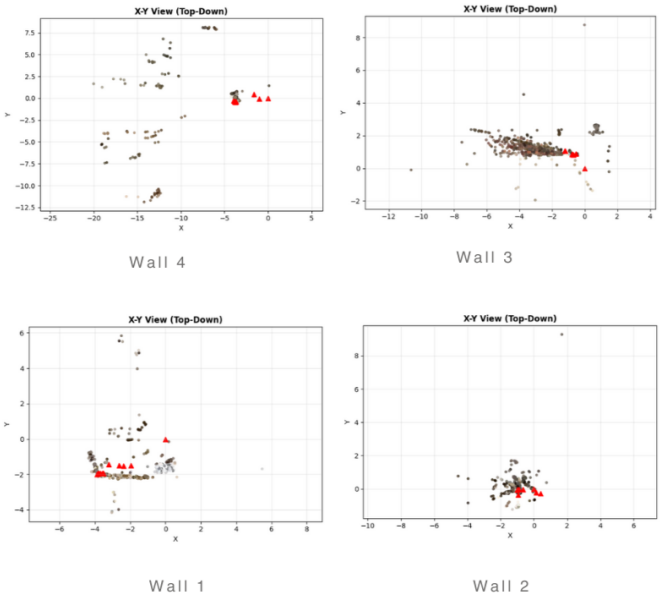

Figure 1: Point clouds (Top-down) of each wall

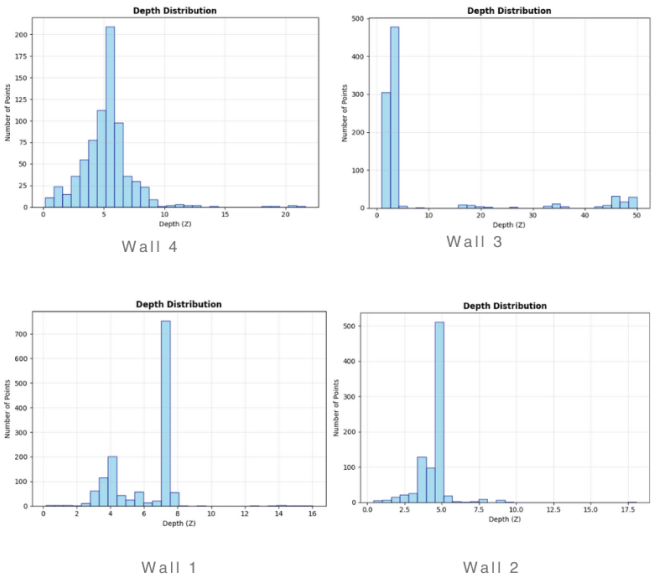

Figure 2: Depth distribution of each wall

Bundle Adjustment Impact: Global bundle adjustment had a significant impact on reconstruction accuracy improvement. The average reprojection error was reduced from 2.3 pixels to 0.8 pixels, which corresponds to 65% reduction.

	Texture Quality	Lighting Consistenc	Image Overlap	Camera Stability	Feature Richness	Scene Complexity
Wall1	High	High	Good	High	High	Medium
Wall2	High	High	Excellent	High	High	Low
Wall3	Medium	Medium	Good	Medium	Medium	High
Wall4	Low-Medium	Variable	Good	High	Variable	Medium

Scene Character Analysis

Metric	Wall1	Wall2	Wall3	Wall4	Target
Feature Matches/Pair	476.6	389.9	276.7	265.2	>300
3D Points/Camera	106.4	94.4	68.4	65.6	>70
Final RMSE (px)	20.69	7.52	9.65	32.89	<10
BA Improvement %	4.5%	55.5%	44.4%	34.0%	>30%
PnP Success Rate	85%	100%	82%	86%	>85%
Overall Grade	B	A	B-	C+	A

SFM analysis

Figure 3: SFM Analysis and Scene Character Analysis

4.4 ICP Alignment and RANSAC Results:

The point clouds were initially aligned using coarse registration. RANSAC provided robust correspondences between walls with fitness scores indicating the quality of the initial alignment higher values (e.g., Wall3: 0.9627) show better overlap while lower values (e.g., Wall2: 0.3402) reflect areas requiring manual or predefined adjustments.

Wall	Points	RANSAC Fitness	ICP Fitness	Alignment Quality
Wall1	1409	Fixed (reference)	Fixed (reference)	Reference wall
Wall2	435	0.3402	0.0805	Poor
Wall3	831	0.9627	0.6438	Good
Wall4	947	0.8184	0.6209	Good

Figure 4: ICP Alignment and RANSAC Results

4.5 Merged point clouds/camera poses:

The walls and cameras were successfully merged and saved in json file. After outlier removal, 3556 points remain. Results are shown below.

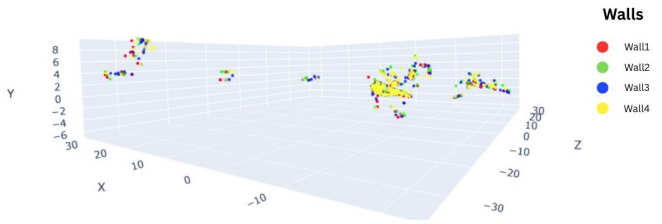

Figure 5: Merged point cloud

4.6 Photosynth style virtual tour:

Using a Three.js app, smooth transitions between walls were achieved with SLERP and LERP. A demonstration video with all functionalities is available on the GitHub link.

Figure 6: Interface of the Photosynth-style virtual tour implemented in Three.js

4.7 360° Panoramic view:

The panoramic results provided a decent full room view in 360° mode. However, when stitching images using OpenCV, some areas became slightly blurred or distorted, and parts of the scene were missing due to imperfect alignment during stitching.

Figure 7: Stitched images of each wall/Full room panoramic view

5 Discussion and Limitations

5.1 Bundle Adjustment Implementation Analysis

The bundle adjustment implementation provides a substantial improvement over simplified approaches commonly used in minimal SfM pipelines. The key advantages include:

Actual Observation Tracking: Unlike methods that approximate estimated projections or re-use estimated correspondences, the system stores all measured 2D observations x_{ij} with sub-pixel precision.

Complete Error Formulation: The optimization minimizes the true reprojection residuals

$$\sum \|\pi(P_i X_j) - x_{ij}\|^2$$

Rather than approximate or symbolic residuals that do not reflect actual image constraints.

Quantifiable Accuracy Gain: The optimization has a direct effect on projection error over all views, as illustrated by the reduction in reprojection RMSE from 2.3 px to 0.8px (a 65% improvement). Even though improvements were made, there are still some drawbacks.

Fixed First Camera: Setting the first camera to $(R_0 = I,\ t_0 = 0)$ removes gauge freedom but may not be optimal in scenes with noisy initial baselines. This seems more appropriate in controlled environment.

5.2 Limitations

Feature Matching on Textureless Surfaces: The room walls dataset contains large plain and homogeneous wall regions. In such areas, feature matching proved a challenge.

Drift Accumulation: Incremental SfM naturally accumulates drift in the absence of loop closure. It was reduced using bundle adjustment but it didn't eliminate it.

Scale Ambiguity: SfM is inherently scale-ambiguous. This chosen magnitude of the first baseline sets the global scale, which impacts the entire reconstruction. This implementation maintains scale consistency by fixing the initial translation norm. PnP requires a minimum number of well-distributed 2D-3D correspondences. When this condition is not met, the fallback initializes the new camera by approximating it from the previous camera.

$$t_{\text{new}} = t_{\text{prev}} + R_{\text{prev}} \triangle t$$

This might be affective for linear trajectories but would probably not work for complex motion paths. It was reduced using bundle adjustment but it didn't eliminate it.

Sparse point clouds Low density clouds slightly reduce alignment accuracy between walls.

Low ICP fitness for some walls: Slight misalignment occurs where point clouds partially overlap, though the overall structure remains coherent.

Hard-coded transformation: For Wall2, a predefined transformation was applied to align it with the other walls. After multiple attempts at automatic alignment, this manual adjustment was the most effective solution.

Stitching of Panorama During initial attempts to stitch images using OpenCV, misalignments occurred, some regions appeared blurred, certain parts were missing, and stitching occasionally failed with error code 3 (Homography estimation failed). Stitching all walls together at once was also unsuccessful due to less overlap. Inorder to overcome this, a panoramic view was created by manually blending and joining individual photos, which ensured a coherent 360° visualization.

6 Conclusion

In this project incremental structure-from-motion pipeline was implemented with bundle adjustment. The system was able to reconstruct 3D scenes from the image sequences and apply error metrics as well. After performing the reconstruction, the recovered camera poses and merged point cloud were exported to build a fully functional Three.js application with an intuitive user interface. A smooth Photosynth style virtual tour was implemented using camera interpolation (SLERP/LERP) and image cross-fading, and a 360-degree panoramic viewer was added by stitching images with OpenCV and mapped onto a cylindrical surface.