CS117 Final Project Report

Zachary Thomas (zachat1)

Min Seop Lee (leems6)

Project GitHub

Introduction

Gaussian Splatting is a modern view-synthesis and radiance-field technique that reconstructs a scene from a set of posed camera images using a sparse 3D point representation and differentiable rendering. A crucial prerequisite to training such a model is the computation of accurate camera intrinsics, extrinsics, and a reliable sparse 3D point cloud.

In the original pipeline, these elements are supplied by COLMAP, a widely used Structure-from-Motion (SfM) and Multi-View Stereo (MVS) system. Our project focuses on manually implementing the SfM stage, which includes camera calibration, pose estimation, and sparse reconstruction, using only OpenCV and geometry techniques discussed in class lectures.

A successful implementation was defined as one that generated camera poses and sparse reconstructions sufficiently accurate to enable the Gaussian Splatting training pipeline to converge and produce a reconstruction visually similar to the baseline COLMAP-generated model.

Baseline Reproduction Setup

Before attempting custom calibration, we established a functional baseline environment. We cloned the official Gaussian Splatting repository and trained the reference model on the “train” dataset provided in the project distribution.

Because Gaussian Splatting requires a CUDA-compatible NVIDIA GPU, and not all team members had access to such hardware, we utilized a Google Cloud Platform (GCP) instance with a T4 GPU for training. Training the reference model on the 301-image dataset took approximately 2–3 hours.

The resulting reconstruction closely matched the expected output reported in the original paper. Rendering the model in SuperSplat Editor showed strong visual fidelity, and quantitative metrics such as SSIM, PSNR, and LPIPS were consistent with published numbers.

Figure 1: Visualization of baseline Gaussian Splat model in SuperSplat Editor (reproduced pipeline).

SSIM	PSNR	LPIPS
0.8219895362854004	22.016555786132812	0.19625388085842133

SSIM: Structural Similarity Index / PSNR: Peak Signal-to-Noise Ratio / LPIPS: Learned Perceptual Image Patch Similarity

These results confirmed that our environment was correctly configured and that we understood the official pipeline well enough to meaningfully evaluate our custom implementation.

COLMAP Pipeline Reproduction

To gain insight into the preprocessing steps required for Gaussian Splatting, we reproduced the COLMAP pipeline manually using the provided Python scripts. COLMAP follows a four-stage workflow:

1. Feature extraction
2. Feature matching
3. Incremental SfM
4. Sparse point-cloud generation and camera pose optimization

We captured our own 23-second video of a static outdoor scene on a Samsung S25. The video was downsampled to 70 frames at 2 FPS, and COLMAP successfully reconstructed a sparse model from the images. We chose 2 FPS as the sampling rate rather than a higher rate because capturing frames too frequently produces nearly identical images with very low disparity. This reduces the effective signal-to-noise ratio for feature matching and ultimately decreases the number of reliable inlier correspondences.

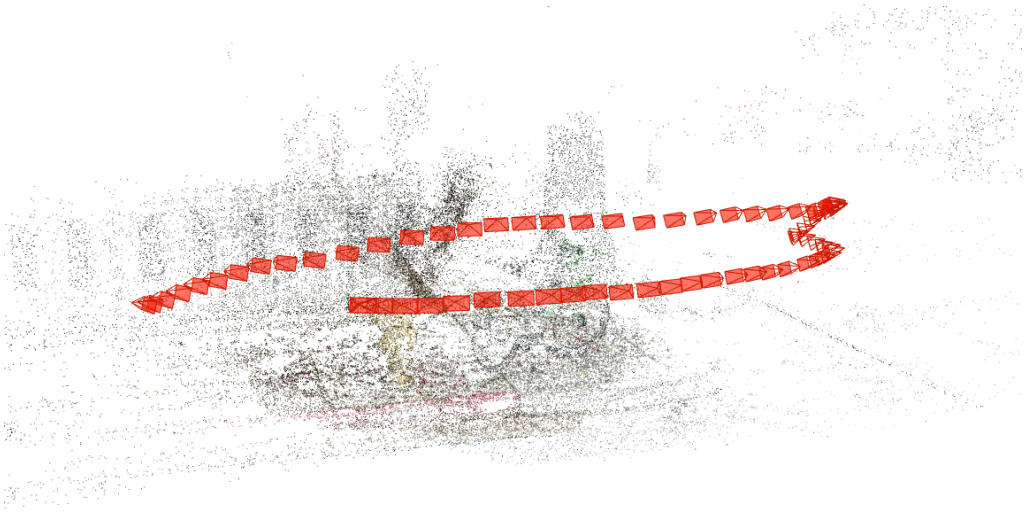

Figure 2: COLMAP-generated camera poses and sparse point cloud from our recorded dataset.

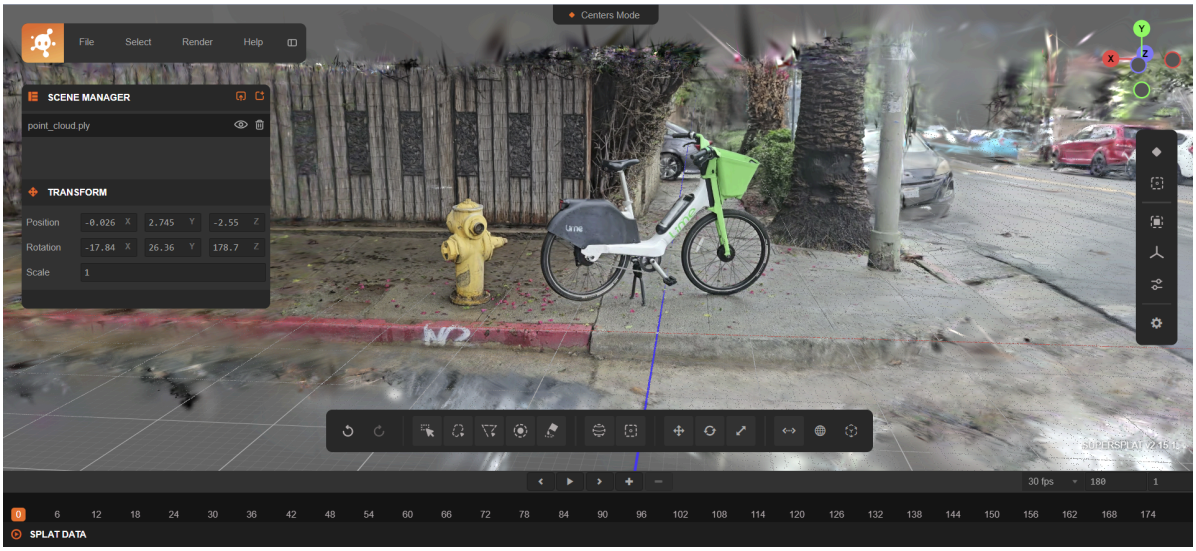

Figure 3: Gaussian Splatting reconstruction (using COLMAP-generated calibration).

We used the COLMAP-derived poses and sparse points as input to Gaussian Splatting training. The resulting reconstruction showed high visual quality:

SSIM	PSNR	LPIPS
0.8723575472831726	28.776405334472656	0.11734525114297867

This validated that our custom dataset was suitable and that COLMAP produced calibration accurate enough for training.

Custom 3D Reconstruction Pipeline

With the COLMAP baseline complete, we built our own calibration and sparse reconstruction system. Our pipeline consists of:

1. Feature extraction
2. Feature matching
3. Structure-from-Motion (SfM)
    - Essential matrix estimation
    - Pose recovery
    - Incremental chaining of extrinsics
    - Point triangulation

These components closely follow the mathematical tools from lectures on epipolar geometry, essential matrix estimation, and triangulation. We adopted a pinhole camera model because it is the only supported camera model for the train stage of the Gaussian Splatting pipeline.

Feature Extraction

Feature extraction is foundational to the 3D reconstruction process. It identifies distinctive points that can be reliably located across multiple images. We evaluated two OpenCV feature detectors:

- SIFT: robust and scale-invariant
- ORB: fast but less stable under viewpoint change

Experimentation showed that while ORB detected more features, SIFT yielded higher-quality matches and more stable reconstructions. Therefore, the final pipeline used SIFT exclusively.

Figure 4: Comparison of SIFT (left) and ORB (right) keypoints (1000 max features).

Feature matching and point correspondence

We used the FLANN-based matcher to identify feature correspondences across consecutive image frames. To remove ambiguous or low-quality matches, we applied Lowe's ratio test with a threshold of 0.75, commonly recommended for SIFT.

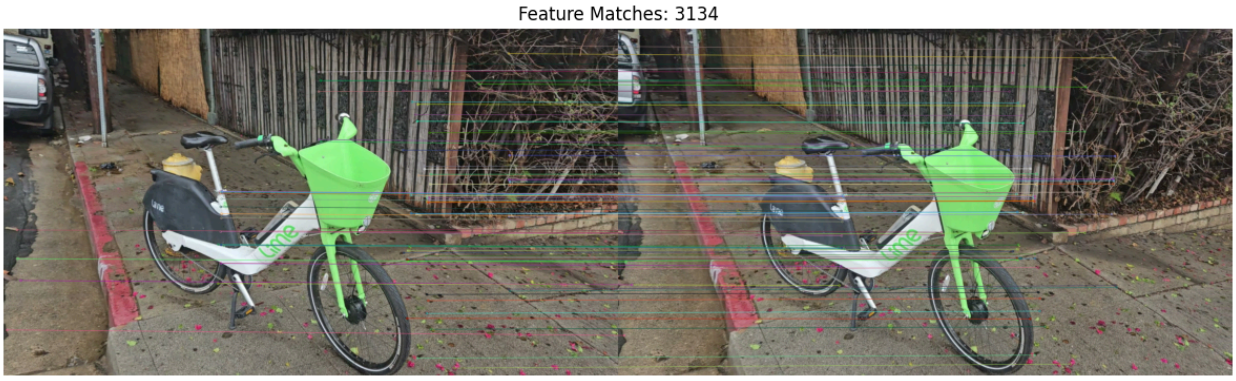

Figure 5: FLANN feature matches between two consecutive frames.

Structure from Motion

For camera motion estimation and 3D reconstruction, we implemented a basic incremental SfM pipeline:

1. Initial pair selection
    - Use the first image pair to estimate the essential matrix E.
    - Recover relative pose (R,t) via cv2.recoverPose.
2. Incremental camera pose chaining
    - For each new frame, compute matches with the previous frame.
    - Estimate the essential matrix and recover the relative pose.
    - Accumulate transforms to obtain global camera poses.
3. Triangulation
    - Use cv2.triangulatePoints to generate new 3D points.
    - Keep only points with positive depth in both cameras.
4. Track management
    - Attempt to merge point tracks across multiple views.

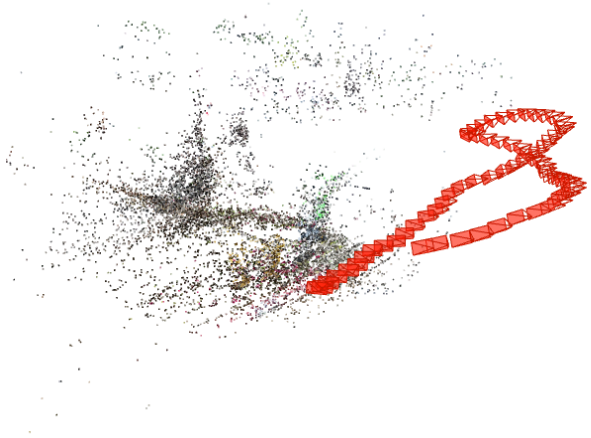

Figure 6: Custom-pipeline sparse reconstruction output

While this implementation captures the structure of incremental SfM, it lacks many refinements found in mature systems such as COLMAP, especially bundle adjustment and multi-view optimization.

Results and Evaluation

Camera Trajectory Alignment

We compared our recovered camera trajectory against the COLMAP derived path by aligning them using the Kabsch–Umeyama method and calculating the mean and max position error. The overall shapes were similar, though our trajectory exhibited noticeable drift.

Mean Position Error	0.6443
Max Position Error	2.1763

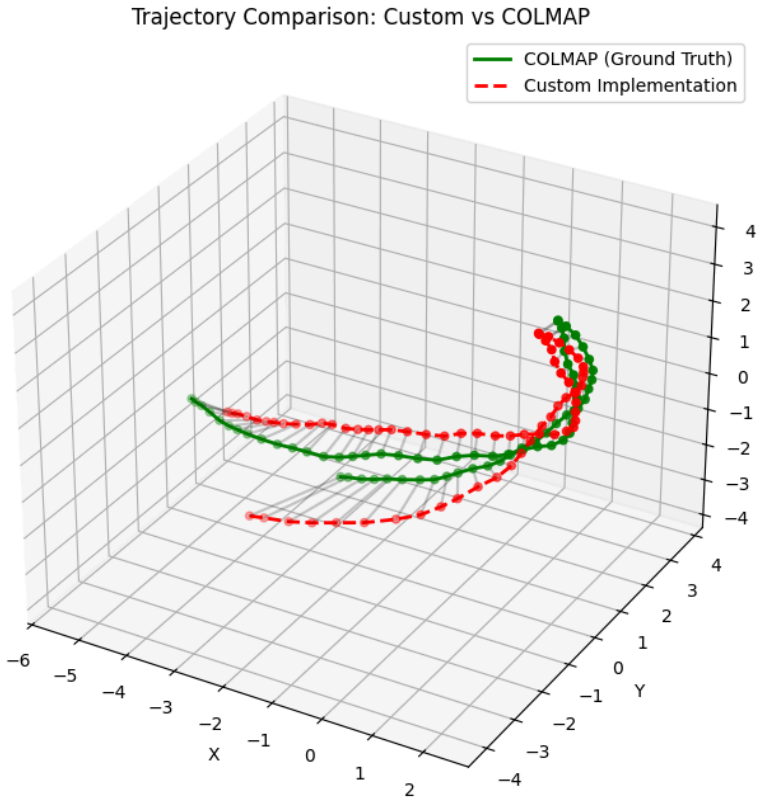

Figure 7: Alignment of custom trajectory (red) to COLMAP trajectory (green).

Reprojection Error

To evaluate internal consistency, we projected each reconstructed 3D point back onto the image plane and measured the pixel error relative to observed keypoints. Results showed a low overall reprojection error with a mean error of 2.1977 pixels.

Mean Reprojection Error	2.1977 pixels
Median Reprojection Error	1.6608 pixels

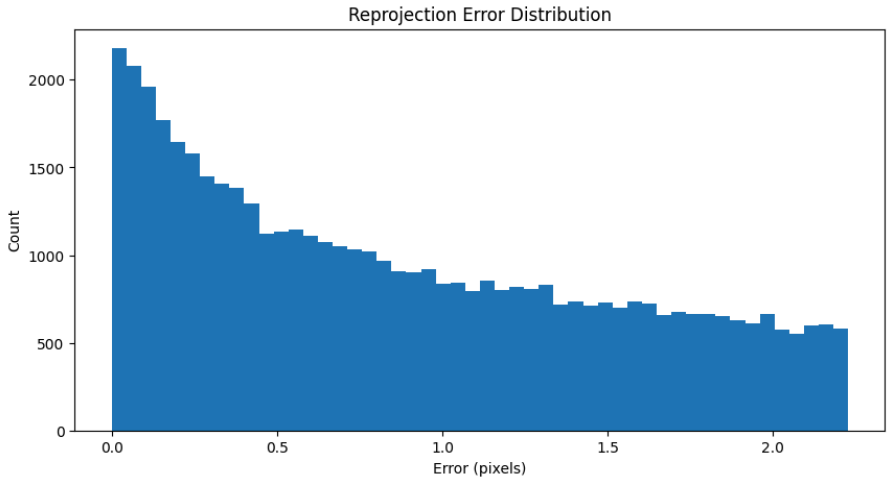

Figure 8: Histogram of reprojection errors.

Figure 9: Observed vs. reprojected keypoints (example image).

These values are reasonable for a simple SfM pipeline, but the absence of bundle adjustment means errors accumulate over time.

Gaussian Splatting Training Results

Using our custom calibration outputs as input to Gaussian Splatting training produced mixed results. While the overall scene layout was recognizable, the reconstruction quality was significantly degraded:

- noticeable ghosting
- inconsistent geometry
- view-dependent distortions
- poor PSNR/SSIM relative to baseline

Figure 10: Ground truth image (left) vs. render from Gaussian Splatting using our reconstruction (right).

This indicates that our poses and/or sparse points lacked sufficient accuracy for the high-precision requirements of Gaussian Splatting training.

Discussion and Challenges

Complexity of Incremental SfM

Many challenges stemmed from implementing a complete SfM pipeline from scratch. Our initial SfM implementation incorrectly retained all triangulated points from two-view matching, producing many outliers. Later attempts to filter points based on multi-view consistency caused excessively sparse reconstructions and increased reprojection error.

The core issue was a misunderstanding of how incremental SfM must repeatedly refine 3D points when new observations are added. Something our implementation lacked in earlier versions.

Absence of Bundle Adjustment

The most significant limitation was the lack of bundle adjustment, a nonlinear optimization that jointly refines all camera poses, all 3D point positions, and optionally camera intrinsics. COLMAP relies heavily on Bundle Adjustment, and without it, our reconstruction could not achieve COLMAP-level accuracy, especially over long image sequences.

COLMAP File Format Integration

Gaussian Splatting requires its input in COLMAP's database and model formats. Re-creating these files (cameras.txt, images.txt, points3D.txt) required careful handling of coordinate conventions and intrinsic/extrinsic parameter encoding.

Conclusion and Future Directions

We successfully implemented a full custom SfM pipeline and integrated it into the Gaussian Splatting preprocessing stage. While our system produces reasonable camera paths and sparse reconstructions, its accuracy is insufficient for high-quality Gaussian Splatting training.

Next steps to improve results:

1. Implement full bundle adjustment using SciPy least-squares.
2. Adopt multi-view point-track merging to ensure consistent correspondences.

3. Reject outliers using RANSAC more aggressively and maintain track quality over long sequences.
4. Incorporate intrinsic refinement rather than assuming known values.

Given these improvements, we believe our pipeline could approach COLMAP-level accuracy and enable high-quality Gaussian Splatting reconstruction from raw consumer video.

References

[1] F. Poux, “Master the 3D reconstruction process: A step-by-step guide,” Towards Data Science. Accessed: Dec. 06, 2025. [Online]. Available: https://towardsdatascience.com/master-the-3d-reconstruction-process-step-by-step-guide/

[2] “COLMAP — COLMAP 3.14.0.dev0,” 5b9a079a (2025-11-14) documentation. Accessed: Dec. 06, 2025. [Online]. Available: https://colmap.github.io/

[3] “- YouTube.” Accessed: Dec. 06, 2025. [Online]. Available: https://www.youtube.com/watch?v=4AvTMVD9ig0&list=PLgnQpQtFTOGSO8HC48K9sPuNliY1qxzV9&index=36

[4] T. Siipola, “Aligning point patterns with Kabsch–Umeyama algorithm.” Accessed: Dec. 06, 2025. [Online]. Available: https://zpl.fi/aligning-point-patterns-with-kabsch-umeyama-algorithm/

[5]“3D Gaussian Splatting for Real-Time Radiance Field Rendering.” Accessed: Dec. 06, 2025. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/