

## **Assignment 2: Incremental SfM Report**

Kim Chae-eun (2023100064)

Korea University, Department of Computer Science and Engineering

Introduction to Computer Vision

Moon Gyeongsik

October 30, 2025

## **1. Introduction**

In this assignment, I implemented an incremental Structure-from-Motion (SfM) pipeline from scratch. The dataset consists of a video (29.98fps, 1920×1080) captured while circling around a Hyundai NEXO vehicle in a semi-circular trajectory in front of the Jeong Un-oh IT Hall at Korea University. To ensure sufficient parallax and feature overlap, 52 images were extracted at 10-frame intervals.

Figure 1. Example Frames



## **2. Implementation**

### **2.0. Camera Intrinsics Initialization**

As the assignment requirements prohibited the use of EXIF metadata, the intrinsic parameter matrix  $K$  was initialized based on image dimensions. The focal length  $f$  was set to  $\max(\text{width}, \text{height})$ , and the principal point was set to the image center ( $\text{width}/2, \text{height}/2$ ). Although this is a rough approximation, the four parameters of  $K$  ( $f_x, f_y, c_x, c_y$ ) are continuously optimized together with 3D points and camera poses in subsequent Bundle Adjustment to obtain accurate intrinsic parameters.

### **2.1 Feature Detection & Matching**

SIFT (Scale-Invariant Feature Transform) algorithm was used to extract keypoints from each frame. Feature matching between frames was performed using BFMatcher with Lowe's ratio test (threshold=0.75) to filter out ambiguous matches. RANSAC-based Essential matrix estimation was additionally applied to remove geometrically inconsistent outliers, obtaining approximately 800 inlier matches on average between consecutive frames.

### **2.2 Two-View Initialization**

The relative pose ( $R, t$ ) was recovered from the Essential matrix of the first two frames using `cv2.recoverPose()`. After fixing the first camera at the origin  $[I|0]$ , 1,403 initial 3D points were generated using `cv2.triangulatePoints()` based on the second camera's pose. Triangulated points were considered valid only if they were located in front of both cameras ( $\text{depth} > 0$ ) and had reprojection error less than 10.0 pixels. Bundle Adjustment was performed on the initial two cameras to refine the initial structure.

### **2.3 Incremental Registration**

The process of sequentially registering new frames was repeated starting from frame 2. First, SIFT matching was performed with the previous frame to find 2D-2D correspondences, which were then converted to 2D-3D correspondences using the `point2d_to_3d` dictionary. This dictionary has (`frame_idx, keypoint_idx`) as key and `point3D_id` as value, linking keypoints in the previous frame that already have 3D points with their matched keypoints in the current frame. When more than 20 2D-3D correspondences were secured, `cv2.solvePnP` (iterations=1000, reprojectionError=8.0, confidence=0.99) was used to estimate the camera pose of the current frame. Registration was considered complete only when at least 15 inliers remained after PnP success, and the 2D points corresponding to inliers were added to `point2d_to_3d` for use in subsequent frames. New 3D points were

generated by selecting only 2D-2D correspondences where neither side has a 3D point yet among matches between the current and previous frames for triangulation. This means keypoints in the previous frame can also acquire 3D points for the first time at this stage. Triangulated points were added only when they were located in front of both cameras and had reprojection error less than 10.0. Bundle Adjustment was performed after every frame registration. Optimized variables include intrinsic parameters  $K$  ( $f_x$ ,  $f_y$ ,  $c_x$ ,  $c_y$ ), extrinsic parameters of all registered cameras ( $R$  represented as 3 axis-angle values + 3 translation values), and all 3D point coordinates. Reprojection error was used as the cost function with Huber loss ( $\delta=1.0$ ) applied for robustness against outliers. Exploiting the structure where each observation depends only on one camera and one 3D point, a sparse Jacobian matrix (99.9% zeros) was constructed using `scipy.sparse.coo_matrix` and passed to `least_squares()` via the `jac_sparsity` argument, improving the computation speed of the Trust Region Reflective method by approximately 10 $\times$ . Through this process, all 52 frames were successfully registered, and a total of 20,751 3D points were reconstructed.

### 3. Result

All 52 frames were successfully registered, resulting in a final reconstruction of 20,751 3D points. An average of 399.1 points per camera were observed, achieving a dense reconstruction. The camera trajectory exhibited a semi-circular pattern rotating around the vehicle, which is consistent with the capture method.

In the initial visualization, the overall structure appeared very small due to some extreme outliers. To address this, percentile-based filtering was applied to remove the most extreme 5% of points along each axis. This eliminated 5,081 outliers (24.5%) from the original 20,751 points, resulting in 15,670 reliable points. The filtered visualization clearly revealed the shape of the car, and the estimated camera trajectory matched the actual semi-circular recording path around the vehicle, confirming the accuracy of my pose estimation. The relationship between the camera trajectory and 3D point cloud could be intuitively confirmed.

Figure 2. Comparison before and after filtering

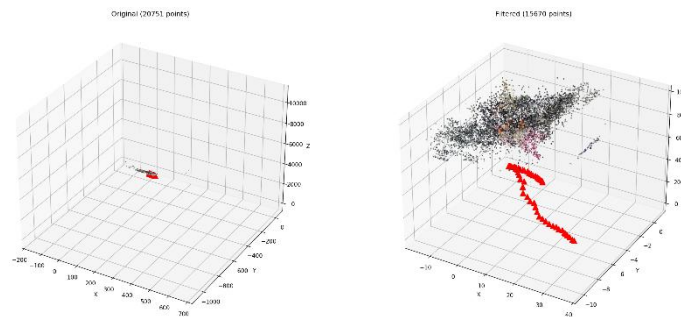


Figure 3. After Outlier Filtering (open3D screenshot)



## 4. Comparison with COLMAP

I ran COLMAP on the same 52 frames for comparison. Initially, point cloud-based Procrustes alignment was attempted, but resulted in a large rotation error of  $152.44^\circ$  due to the vehicle's structural symmetry and lack of correspondences. To address this, I applied frame-based alignment by matching cameras with the same frame name 1:1.

With frame-based alignment, both methods showed similar semi-circular camera trajectories (rotation angle:  $48.58^\circ$ ), and 3D points overlapped well in the aligned overlay. The median point correspondence distance was 0.4607, representing approximately 2.3% relative error, confirming that both methods reconstructed similar 3D structures.

The main difference lies in the number and density of points. My implementation generated 20,751 points (399.1 pts/camera), while COLMAP produced 16,638 points (320.0 pts/camera), approximately 25% fewer. This is because I aggressively triangulated new 2D-2D matches at every frame, while COLMAP uses a more conservative strategy. In terms of scale, COLMAP was normalized approximately  $5\times$  smaller (scale factor: 0.2104), which stems from the inherent scale ambiguity in SfM.

My implementation's limitation is the generation of some outliers due to aggressive triangulation (24.5% filtering required), while COLMAP's limitation is lower point density due to its conservative strategy. The two approaches represent a trade-off between density and stability.

Figure 4. Alignment based on the reconstructed point clouds

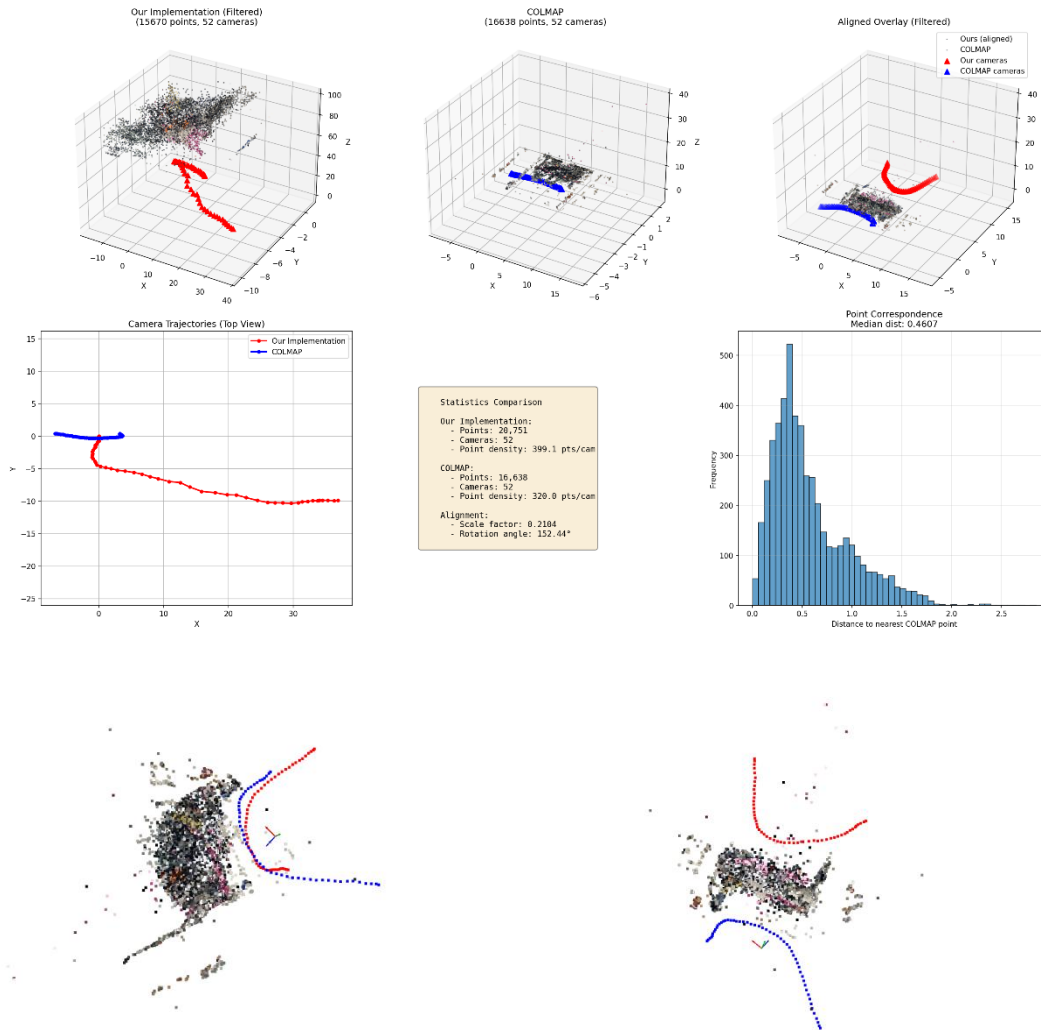
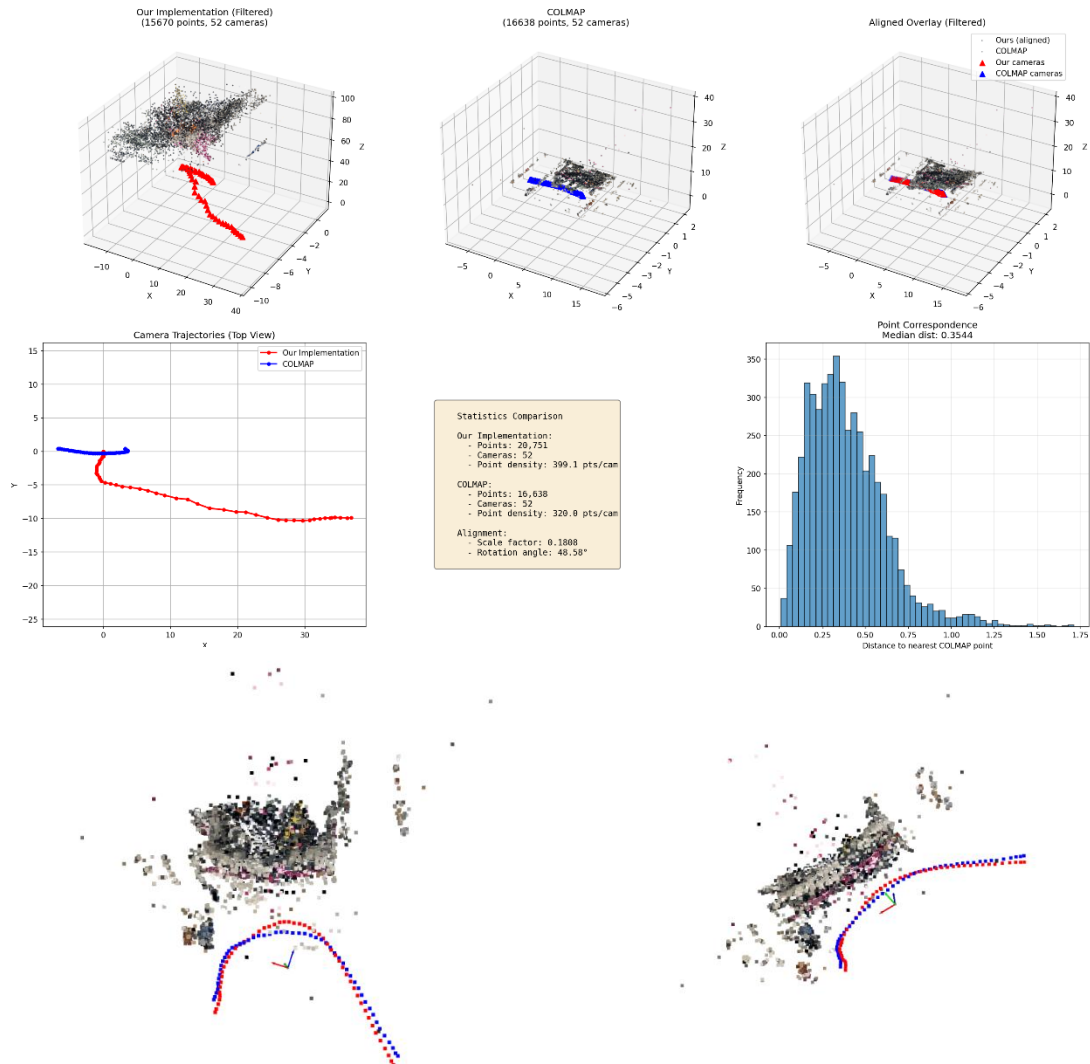


Figure 5. Alignment based on the frame id



## 5. Conclusion

Several challenges arose during implementation. First, the inherent scale ambiguity in SfM made direct comparison with COLMAP difficult, which I addressed through frame-based alignment. Second, outliers from aggressive triangulation were resolved using percentile-based filtering (5%). Third, frame interval selection was critical—30-frame intervals achieved only 13/18 (72%) registration, while 10-frame intervals achieved 52/52 (100%), confirming the importance of sufficient feature overlap. Fourth, Bundle Adjustment's computational cost was significant, but utilizing sparse Jacobian matrices achieved approximately 10× speedup.

As EXIF metadata usage was prohibited per assignment requirements, I initialized intrinsic parameters  $K$  based on image dimensions. However, after completing the assignment, I discovered a method to recover  $K$  using orthogonal constraints from two images[1]. Applying this algorithm to the first frame yielded more realistic values:  $f_x=1082.97$ ,  $f_y=1082.97$ ,  $c_x=747.41$ ,  $c_y=300.76$ . Such methods could provide better initial  $K$  without metadata. Additionally, loop closure detection for drift reduction, adaptive keyframe selection, multi-view triangulation (3+ views), and GPU acceleration could further improve performance.

Intrinsic Matrix:  

$$\begin{bmatrix} 1.08296669e+03 & 0.00000000e+00 & 7.47411023e+02 \\ 0.00000000e+00 & 1.08296669e+03 & 3.90075572e+02 \\ 0.00000000e+00 & 0.00000000e+00 & 1.00000000e+00 \end{bmatrix}$$

[1] [https://github.com/towardsautonomy/cam\\_intrinsic\\_calibration\\_single\\_image](https://github.com/towardsautonomy/cam_intrinsic_calibration_single_image)