Point Cloud Aggregation Based on GNSS-INS Data and Moving Object Filtering

Pravin Oli*
Eötvös Loránd University, IFROS, Hungary
Email: pravin.oli.08@gmail.com

***Abstract*—This work presents a pipeline for generating dense, static, and realistically colored 3D point clouds of urban environments using multi-sensor data from a moving vehicle. LiDAR sweeps are transformed into a global frame using GNSS-INS ego poses and aggregated over time. Dynamic objects are removed through a two-stage unsupervised filtering approach combining temporal consistency and spatial clustering. Finally, multi-view camera images are fused to assign RGB values to the static points, producing visually informative 3D maps. The approach demonstrates accurate and efficient reconstruction suitable for visualization, simulation, and autonomous driving applications.**

***Index Terms*—Point Cloud Aggregation, GNSS-INS, LiDAR, Moving Object Filtering, RGB Colorization, Sensor Fusion, nuScenes**

1 Introduction

This assignment focuses on building a static and visually enriched 3D representation of an urban environment using multi-sensor data from a moving vehicle. The work is based on the nuScenes dataset, which provides synchronized and calibrated measurements from a top-mounted LiDAR, a GNSS-INS system with RTK corrections, and multiple RGB cameras.

Three main tasks are addressed. First, frame-by-frame LiDAR point clouds are transformed into a global coordinate system using ego-motion information from the GNSS-INS sensor and aggregated into a single virtual point cloud. Second, dynamic objects are removed from the aggregated data using an unsupervised filtering approach that exploits temporal consistency and spatial density, without relying on manual annotations. Finally, the resulting static point cloud is colorized using camera images by projecting 3D points onto the image planes and assigning RGB values.

The proposed pipeline demonstrates how accurate ego-motion, temporal analysis, and multi-view sensor fusion can be combined to generate a clean and realistic 3D map of an urban scene.

2 Data Description

The nuScenes v1.0-mini dataset is used in this assignment. It is a reduced subset of the full nuScenes dataset and retains the complete multi-sensor setup, calibration accuracy, and data structure, making it well suited for developing and evaluating multi-sensor fusion pipelines. The dataset consists of urban driving sequences collected using a sensor-equipped vehicle, with all sensor measurements time-synchronized and provided with accurate ego poses in a global coordinate frame.

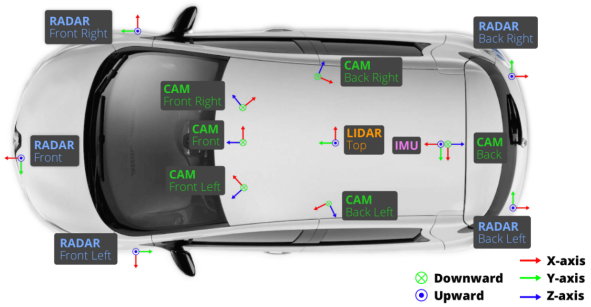

Fig. 1. Vehicle sensor configuration used in the nuScenes dataset, showing the placement of the top-mounted LiDAR, GNSS-INS unit, and surround-view cameras.

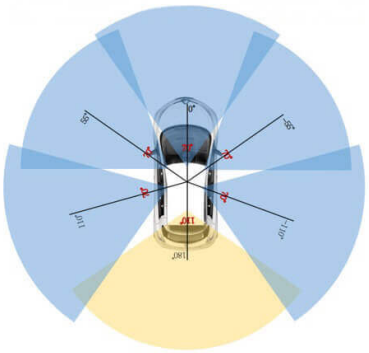

Fig. 2. Surround-view camera configuration in the nuScenes dataset, illustrating the six RGB cameras providing 360° visual coverage.

2.1 Sensor Setup

The vehicle is equipped with the following sensors:

- **Top-mounted LiDAR (Velodyne HDL-32E):** A 32-beam spinning LiDAR providing 360° horizontal coverage and dense 3D point clouds for geometric mapping.
- **GNSS-INS system:** Provides accurate ego-motion estimates using fused GNSS and inertial measurements, supported by RTK corrections for centimeter-level positioning accuracy.
- **Six surround-view RGB cameras:** Front, rear, and side-facing cameras providing full 360° visual coverage of the environment, used for point cloud colorization.

2.2 Dataset Features

Key features of the nuScenes dataset relevant to this assignment include:

- Accurate intrinsic and extrinsic calibration for all sensors, enabling precise transformation between sensor coordinate frames.
- Time-synchronized sensor measurements, allowing reliable fusion of LiDAR, GNSS-INS, and camera data.
- Ego poses provided as translation vectors and rotation quaternions in a global coordinate system.
- Scene-based organization with temporally ordered samples, which is well suited for point cloud aggregation and temporal consistency analysis.

3 Methodology

This section describes the mathematical formulation and processing steps used to aggregate LiDAR point clouds, remove moving objects, and assign color information using camera images. All transformations are performed using the calibrated sensor parameters and ego-motion information provided by the dataset.

3.1 Coordinate Frames and Notation

Three coordinate frames are used throughout this work:

- **LiDAR frame** $\mathcal{F}_L$: The local coordinate system of the LiDAR sensor.
- **Ego frame** $\mathcal{F}_E$: The vehicle-centered coordinate system.
- **World frame** $\mathcal{F}_W$: A fixed global coordinate system provided by the dataset.

A LiDAR point is represented in homogeneous coordinates as

$$\mathbf{p} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T, \tag{1}$$

which allows rotations and translations to be expressed using matrix multiplication.

3.2 GNSS-INS Based Ego-Motion Estimation

The ego pose of the vehicle at each timestamp is provided by the GNSS-INS system as a 3D translation vector and a unit quaternion representing orientation. The quaternion

$$\mathbf{q} = (w, x, y, z) \tag{2}$$

is converted into a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$. Together with the translation vector $\mathbf{t} \in \mathbb{R}^3$, a homogeneous transformation matrix is formed as

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3}$$

This formulation is used to describe both the LiDAR-to-ego transformation and the ego-to-world transformation, enabling the conversion of points between coordinate frames.

For example, consider a LiDAR point $\mathbf{p}_L = [1,\ 0,\ 0,\ 1]^T$ meters and an ego pose with translation $\mathbf{t} = [10,\ 0,\ 0]^T$ meters and no rotation. The transformed point in the world frame becomes $\mathbf{p}_W = [11,\ 0,\ 0,\ 1]^T$.

3.3 LiDAR Point Cloud Aggregation

Each LiDAR scan is first transformed from the LiDAR frame to the ego frame and then into the world frame. This is expressed as

$$\mathbf{p}_W = \mathbf{T}_{W\leftarrow E}\mathbf{T}_{E\leftarrow L}\mathbf{p}_L, \tag{4}$$

where $\mathbf{T}_{E\leftarrow L}$ denotes the calibrated LiDAR-to-ego transformation and $\mathbf{T}_{W\leftarrow E}$ represents the ego pose in the world frame.

The order of multiplication is critical, as the LiDAR points must first be expressed in the ego frame before applying the global ego pose. By applying this transformation to all LiDAR points in each sweep and aggregating the results over time, a dense global point cloud is obtained:

$$\mathcal{P}_{global} = \bigcup_{t=1}^{T} \mathcal{P}_W^{(t)}. \tag{5}$$

3.4 Moving Object Filtering

Aggregating LiDAR scans over time introduces artifacts caused by dynamic objects. To obtain a static representation of the environment, an unsupervised two-stage filtering strategy is applied.

3.4.1 Temporal Consistency Filtering

The global space is discretized into a 3D voxel grid with voxel size s. Each point $\mathbf{p}$ is mapped to a voxel index $\mathbf{v}$ as

$$\mathbf{v} = \left\lfloor \frac{\mathbf{p} - \mathbf{o}}{s} \right\rfloor, \tag{6}$$

where $\mathbf{o}$ denotes the origin of the voxel grid.

For each voxel, the number of LiDAR sweeps in which it appears is counted:

$$C(\mathbf{v}) = \sum_{t=1}^{T} \mathbb{1}(\mathbf{v} \in \mathcal{V}_t), \tag{7}$$

where $\mathbb{1}(\cdot)$ is the indicator function. Voxels observed fewer than N_{min} times are removed:

$$C(\mathbf{v}) \geq N_{\text{min}}. \tag{8}$$

This step removes most points belonging to moving objects, as they do not exhibit temporal consistency.

3.4.2 Spatial Filtering Using DBSCAN

After temporal filtering, spatial clustering is applied using the DBSCAN algorithm. A point $\mathbf{p}$ is considered a core point if its ε-neighborhood contains at least *minPts* points:

$$|\mathcal{N}_\varepsilon(\mathbf{p})| \geq \text{minPts}. \tag{9}$$

Clusters with fewer than N_{cluster} points are removed:

$$|\mathcal{C}_k| \geq N_{\text{cluster}}. \tag{10}$$

This step removes sparse noise and small residual dynamic clusters, resulting in a clean static point cloud.

3.5 Point Cloud Colorization Using Camera Images

To enhance the visual quality of the static point cloud, RGB color values are assigned using synchronized camera images. Each 3D point is transformed from the world frame into the camera frame and projected onto the image plane using the camera intrinsic matrix $\mathbf{K}$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}. \tag{11}$$

Only points satisfying $z > 0$ and lying within the image boundaries are considered visible. Pixel colors are sampled using bilinear interpolation to reduce aliasing effects.

When a point is observed in multiple camera views, its final color is computed by averaging all valid observations:

$$\mathbf{c}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} \mathbf{c}_i^{(k)}, \tag{12}$$

where N_i is the number of valid camera projections. This multi-view fusion improves color consistency and robustness to occlusions.

4 Implementation

The point cloud aggregation pipeline was implemented in Python notebook using the nuScenes SDK, Open3D, and standard scientific libraries (NumPy, SciPy, OpenCV). The workflow follows three main stages: point cloud transformation and aggregation, moving object removal, and RGB colorization.

4.1 Dataset Loading and Preprocessing

The nuScenes v1.0-mini dataset was loaded using the `NuScenes` API. LiDAR points from each sweep were extracted and transformed from the sensor frame to the world coordinate frame using calibrated sensor extrinsics and ego poses from the GNSS-INS system. Helper functions converted quaternions to 4×4 homogeneous transformation matrices, enabling consistent frame-to-frame alignment. Single-frame and multi-frame point clouds were visualized using `Open3D` and `Matplotlib` for verification.

4.2 Point Cloud Aggregation

All LiDAR sweeps within a scene were transformed to the global frame and aggregated into a dense point cloud. This step provides a complete geometric representation of the environment, serving as the input for subsequent filtering stages.

4.3 Moving Object Filtering

A two-stage unsupervised filtering approach was applied to remove dynamic objects:

1) **Temporal Consistency Filtering:** The global space was discretized into a 3D voxel grid, and voxels observed in fewer than a threshold number of sweeps were removed. This effectively eliminates points from transient objects such as vehicles and pedestrians.
2) **Spatial Filtering with DBSCAN:** The remaining points were clustered using the DBSCAN algorithm. Clusters smaller than a minimum size were discarded to remove sparse noise and residual moving objects, resulting in a clean static point cloud.

4.4 RGB Colorization

To produce a realistic colored map, static points were projected into the six synchronized camera images. Each point's RGB value was sampled using bilinear interpolation, and multi-view observations were averaged to improve color consistency and reduce occlusion artifacts. The resulting colored point cloud was visualized in 3D and saved as a PLY file for further use.

4.5 Tools and Libraries

Key libraries and tools used in the implementation include:

- `NuScenes SDK` for dataset access and sensor geometry.
- `Open3D` for point cloud manipulation and 3D visualization.
- `NumPy` and `SciPy` for numerical computations and interpolation.
- `OpenCV` for image loading and processing.
- `Matplotlib` for 2D scatter visualizations during debugging.

5 Results

The proposed pipeline was evaluated from the nuScenes v1.0-mini dataset. The results demonstrate the step-by-step improvement in the point cloud quality, from raw sensor measurements to a fully colored static map.

5.1 Raw LiDAR Points

Fig. 3 shows the raw LiDAR sweep in the sensor frame. The points are dense but oriented relative to the LiDAR, making it difficult to perceive the global structure. After transforming the points into the world frame (Fig. 3), the vehicle trajectory and the urban environment become clearly visible, providing a global reference for aggregation.

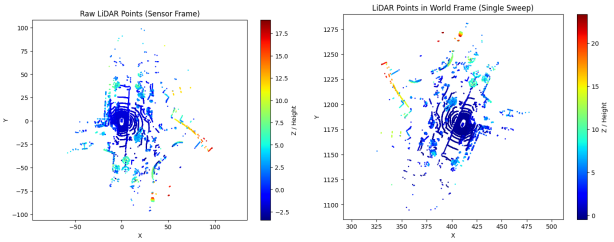

Fig. 3. Left: raw LiDAR points in the sensor frame. Right: points transformed into the world frame.

5.2 Aggregated Global Point Cloud

Aggregating all LiDAR sweeps from the scene results in a dense global point cloud (Fig. 4). This provides a comprehensive geometric map of the environment but contains artifacts due to moving objects.

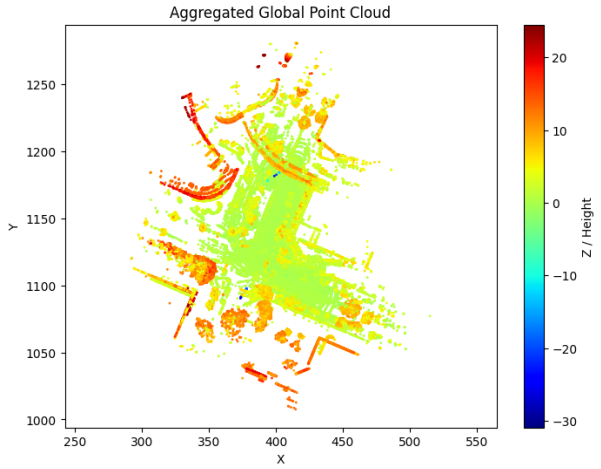

Fig. 4. Aggregated global point cloud before filtering.

5.3 Moving Object Filtering

Dynamic objects were progressively removed using temporal consistency and spatial clustering. Temporal filtering (Fig. 5) eliminated points appearing in only a few sweeps, and DBSCAN clustering (Fig. 5) further removed sparse noise and residual moving objects, resulting in a clean static map.

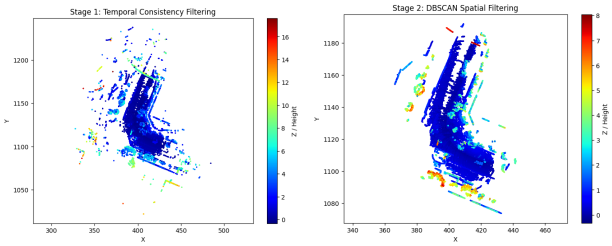

Fig. 5. Left: point cloud after temporal consistency filtering. Right: final static point cloud after DBSCAN spatial filtering.

5.4 RGB Colorization

The static point cloud was colorized using synchronized camera images (Fig. 6). Multi-view color fusion produces realistic RGB values, highlighting buildings, roads, and vegetation, and providing an intuitive, visually informative 3D representation.

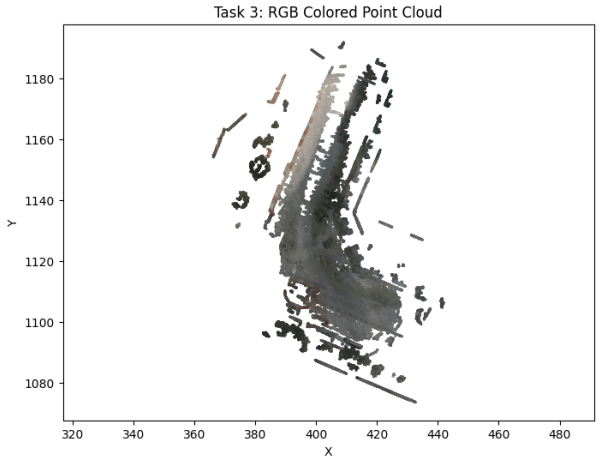

Fig. 6. Realistic RGB-colored static point cloud after multi-view camera fusion.

5.5 Output File

The final colored static point cloud was saved in PLY format as `static_colored_realistic.ply`, which can be used for further visualization, simulation, or 3D mapping applications.

6 Discussion

- **Point Cloud Aggregation:** The LiDAR sweeps were transformed to the world frame using calibrated sensor extrinsics and GNSS-INS ego poses. This ensures accurate global alignment without requiring expensive ICP-based registration, which is unnecessary due to the high-quality localization in nuScenes. Aggregating multiple sweeps increased density and coverage, providing a complete geometric map of the scene.
- **Moving Object Filtering:** A two-stage unsupervised approach combining temporal consistency and DBSCAN clustering was chosen. Temporal filtering effectively removes transient objects that appear in few sweeps, while DBSCAN eliminates sparse noise and small residual clusters. Methods like RANSAC or plane fitting were avoided because the urban scenes contain diverse structures (roads, buildings, vehicles), making model-based fitting unreliable and prone to removing valid static points.
- **RGB Colorization:** Multi-view camera projection with bilinear interpolation was used to assign colors to static points. Averaging over multiple camera views improves consistency and handles occlusions. This approach is straightforward, efficient, and fully leverages the synchronized camera data, producing visually realistic maps suitable for visualization or simulation tasks.

7 Conclusion

This work presented a pipeline for generating dense, static, and realistically colored 3D point clouds from multi-sensor urban driving data. LiDAR sweeps were accurately trans-

formed and aggregated into a global frame using GNSS-INS ego poses, while a two-stage unsupervised filtering method removed dynamic objects and noise. Multi-view camera images were then fused to assign RGB values, producing visually informative 3D maps.

The results demonstrate that combining precise ego-motion, temporal analysis, and sensor fusion can efficiently create high-quality static maps suitable for visualization, simulation, or further perception tasks in autonomous driving scenarios. Future improvements could include better handling of occlusions and more adaptive filtering to enhance detail in complex urban environments.

8 Files Submitted

The following files are included as part of this project submission:

8.1 Report

- Point Cloud Aggregation.pdf (assignment report)

8.2 Zips file containing scripts, output (Images, Videos, Ply file) (.zip)

9 GitHub Repository

The full implementation, source code, and results are available at:

GitHub: Rudip1/nusc_pc_aggregation