Multi-Modal Sensor Fusion: Radar-LiDAR Integration for Object Detection and Tracking

AV Perception Portfolio Team

January 31, 2026

Abstract

This report presents an advanced multi-modal sensor fusion framework that integrates radar and LiDAR data for robust object detection and tracking in autonomous vehicle applications. We extend the baseline perception pipeline with advanced Digital Signal Processing (DSP) techniques, including Doppler FFT and Constant False Alarm Rate (CFAR) detection for radar, alongside statistical outlier removal and RANSAC-based ground plane removal for LiDAR. Furthermore, we implement Bayesian estimation methods, specifically a Particle Filter for non-linear multi-object tracking and an Occupancy Grid map for environmental representation. The system is validated in the CARLA simulator, demonstrating significantly improved tracking stability and geometric awareness in complex urban scenarios.

1 Introduction

Autonomous vehicles rely on heterogeneous sensor suites to perceive their environment. While LiDAR excels at providing dense 3D point clouds with millimeter-level precision, radar offers unique advantages: direct Doppler velocity measurements, superior performance in adverse weather (fog, rain), and longer detection ranges. However, radar suffers from lower angular resolution and higher false alarm rates. This motivates a *sensor fusion* approach that combines radar's kinematic information with LiDAR's geometric accuracy.

The extended Week 2 project addresses the following technically advanced challenges:

1. **Advanced Radar DSP**: Computation of Range-Doppler maps and application of CFAR detectors to manage false alarm rates.
2. **LiDAR Point Cloud Refinement**: Statistical noise reduction and geometric ground plane removal using RANSAC.
3. **Non-Linear Tracking**: Implementation of a Particle Filter to handle multi-modal state distributions and non-linear movement.
4. **Bayesian Environmental Mapping**: Development of a 2D Occupancy Grid map updated via Bayesian recursive inference.
5. **Multi-Sensor Fusion Strategy**: Optimized weighting between radar kinematics and filtered LiDAR geometry.

2 Methodology

2.1 Sensor Data Preprocessing

2.1.1 Radar Point Cloud Generation

CARLA's radar sensor outputs detections in polar coordinates relative to the sensor frame. Each detection d_i contains:

$$d_i = (r_i, \theta_i, v_{rel,i}) \tag{1}$$

where r_i is range [m], θ_i is azimuth [rad], and $v_{rel,i}$ is radial velocity [m/s].

We transform these to Cartesian BEV coordinates (x, y, v_x, v_y) via:

$$\begin{aligned} x_i &= r_i \cos(\theta_i) \\ y_i &= r_i \sin(\theta_i) \\ v_{x,i} &= v_{rel,i} \cos(\theta_i) \\ v_{y,i} &= v_{rel,i} \sin(\theta_i) \end{aligned} \tag{2}$$

This produces a radar point cloud $\mathcal{R} = \{(x_i, y_i, v_{x,i}, v_{y,i})\}_{i=1}^{N_r}$ in the vehicle's coordinate frame.

2.1.2 Advanced LiDAR DSP

To ensure high-quality spatial data, we implement a multi-stage preprocessing pipeline:

1. **Statistical Outlier Removal (SOR)**: Filters sensor noise by analyzing the local neighborhood of each point. For each point p_j, we compute the mean distance $\bar{d}_j$ to its $k = 20$ nearest neighbors. Points are retained if they satisfy:

$$\bar{d}_j \leq \mu + \alpha\sigma \tag{3}$$

where μ and σ are the global mean and standard deviation of neighbor distances, and $\alpha = 2.0$ is the multiplier.

2. **Voxel Grid Downsampling**: Reduces computational load by discretizing the 3D space into voxels of size $0.2m$. All points within a voxel are replaced by their centroid:

$$P_{voxel} = \frac{1}{N_{points}} \sum_{i=1}^{N_{points}} p_i \tag{4}$$

3. **RANSAC Ground Removal**: Iteratively fits a plane model $ax + by + cz + d = 0$ to the point cloud. Points within a threshold of $0.3m$ are classified as ground (inliers) and removed, isolating above-ground obstacles.

2.1.3 Advanced Radar DSP

Raw radar returns are processed to isolate targets from background clutter:

1. **Hamming Windowing**: Applied to time-domain samples to reduce spectral leakage from sidelobes:

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \tag{5}$$

2. **Doppler FFT**: Extraction of radial velocity v_{rel} from complex Intermediate Frequency (IF) signals using Fast Fourier Transform.

3. **Adaptive LMS Filter**: A Least Mean Squares filter for clutter rejection. It estimates and cancels background noise using the update rule:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e_n \mathbf{x}_n \tag{6}$$

where e_n is the estimation error and μ is the learning rate.

4. **Cell-Averaging CFAR Detector**: Maintains a constant false alarm rate P_{fa} by adapting the detection threshold T to the local noise power P_{noise}:

$$T = \gamma \cdot P_{noise}, \quad \gamma = N_{train} \ln(1/P_{fa}) \tag{7}$$

where N_{train} training cells surround the test cell, separated by guard cells.

This yields a radar point cloud $\mathcal{R} = \{(x_i, y_i, v_{x,i}, v_{y,i})\}_{i=1}^{N_r}$.

2.2 Object Detection via DBSCAN Clustering

Radar detections from the same physical object appear as spatial clusters in BEV. We apply DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to group radar points:

- **Parameters**: $\epsilon = 1.5$ m (neighborhood radius), minPts = 2 (minimum cluster size).
- **Output**: Cluster labels $C = \{c_1, c_2, \ldots, c_K\}$ where $c_k \in \{0, 1, \ldots, K-1, -1\}$ (-1 denotes noise).

For each cluster k, we compute the centroid:

$$\mathbf{m}_k = \frac{1}{|C_k|} \sum_{i \in C_k} (x_i, y_i, v_{x,i}, v_{y,i}) \tag{8}$$

where C_k is the set of points assigned to cluster k.

2.3 Multi-Object Tracking with Kalman Filtering

2.3.1 State Representation

Each tracked object maintains a 4D state vector:

$$\mathbf{x}_t = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T \tag{9}$$

representing position and velocity in BEV coordinates.

2.3.2 Motion Model

We employ a *constant velocity* kinematic model:

$$\mathbf{x}_{t+1} = \mathbf{F}_t \mathbf{x}_t + \mathbf{w}_t \tag{10}$$

where the state transition matrix is:

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

and $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q})$ is process noise with $\mathbf{Q} = 0.1 \cdot \mathbf{I}_4$.

2.3.3 Measurement Model

Observations consist of 2D positions from radar cluster centroids:

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t \tag{12}$$

where:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{v}_t \sim \mathcal{N}(0, \mathbf{R}) \tag{13}$$

with measurement noise covariance $\mathbf{R} = \text{diag}(1.0, 1.0)$ m^2.

2.4 Multi-Object Tracking: Kalman vs. Particle Filter

While Kalman filters provide computationally efficient linear estimation, we also implement a **Particle Filter** for non-linear state estimation.

- **Prediction (Proposal)**: Each particle j is propagated forward using the motion model with added process noise:

$$\mathbf{x}_{t+1}^{(j)} = f(\mathbf{x}_t^{(j)}) + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q}) \tag{14}$$

- **Update (Weighting)**: Weights are updated based on the Gaussian likelihood of observing the measurement $\mathbf{z}_t$:

$$w_t^{(j)} \propto w_{t-1}^{(j)} \cdot \exp\left(-\frac{1}{2}(\mathbf{z}_t - h(\mathbf{x}_t^{(j)}))^T \mathbf{R}^{-1} (\mathbf{z}_t - h(\mathbf{x}_t^{(j)})) \right) \tag{15}$$

- **Resampling**: Systematic resampling is triggered when the effective number of particles N_{eff} falls below a threshold, mitigating particle degeneracy.

2.5 Bayesian Occupancy Grid Mapping

A discretized 2D grid represents the environment's occupancy state. For each cell O_k, we update its occupancy probability $p(O_k|z_{1:t})$ via the recursive Bayesian update:

$$p(O_k|z_t) = \frac{p(z_t|O_k)p(O_k)}{p(z_t|O_k)p(O_k) + p(z_t|\neg O_k)p(\neg O_k)} \tag{16}$$

By assuming a binary occupancy model (occupied/free), the grid maintains a spatial belief that is robust to transient sensor noise and occlusions.

2.6 Radar-LiDAR Fusion and Bayesian Weighted Average

After the state update, we refine each track's position using a Bayesian Weighted Average of radar and LiDAR centroids:

$$\mathbf{p}_{fused} = \frac{\omega_{radar}\mathbf{p}_{radar} + \omega_{lidar}\mathbf{p}_{lidar}}{\omega_{radar} + \omega_{lidar}} \tag{17}$$

where ω_i represents the confidence (inverse variance) of each sensor. In practice, $\omega_{lidar} = 0.7$ and $\omega_{radar} = 0.3$ provide a stable fusion that leverages LiDAR's spatial precision and radar's velocity consistency.

3 Implementation Details

Modular Architecture:

1. `radar_dsp.py`: Signal processing (Doppler FFT, CA-CFAR, Adaptive LMS, Hamming Window).
2. `lidar_dsp.py`: Spatial processing (SOR Filter, RANSAC Ground Removal, Voxel Downsampling).
3. `bayesian_fusion.py`: Particle Filter tracking, Bayesian Occupancy Grid, and Weighted Centroid Fusion.
4. `radar_lidar_fusion_advanced.py`: Advanced integrated perception pipeline.

The system utilizes **CARLA 0.9.15**, **scikit-learn** for clustering, and **NumPy/SciPy** for signal processing.

4 Results and Discussion

4.1 Experimental Setup

The system was evaluated in CARLA's Town03 environment with the following configuration:

- **Duration**: 120 seconds
- **Vehicle Speed**: 50 km/h (autopilot enabled)
- **Radar FOV**: 35° horizontal, 20° vertical, 100 m range
- **LiDAR**: 32 channels, 100 m range, 100k points/sec
- **Frame Rate**: 20 Hz (CARLA tick rate)

4.2 Qualitative Analysis

Figure 1 shows a representative BEV snapshot at frame 150. Key observations:

- **Radar Coverage**: Sparse but velocity-aware detections (colored by speed).
- **LiDAR Density**: Dense cyan point cloud providing geometric context.
- **Fused Tracks**: Green markers indicate stable multi-object tracks with consistent IDs.

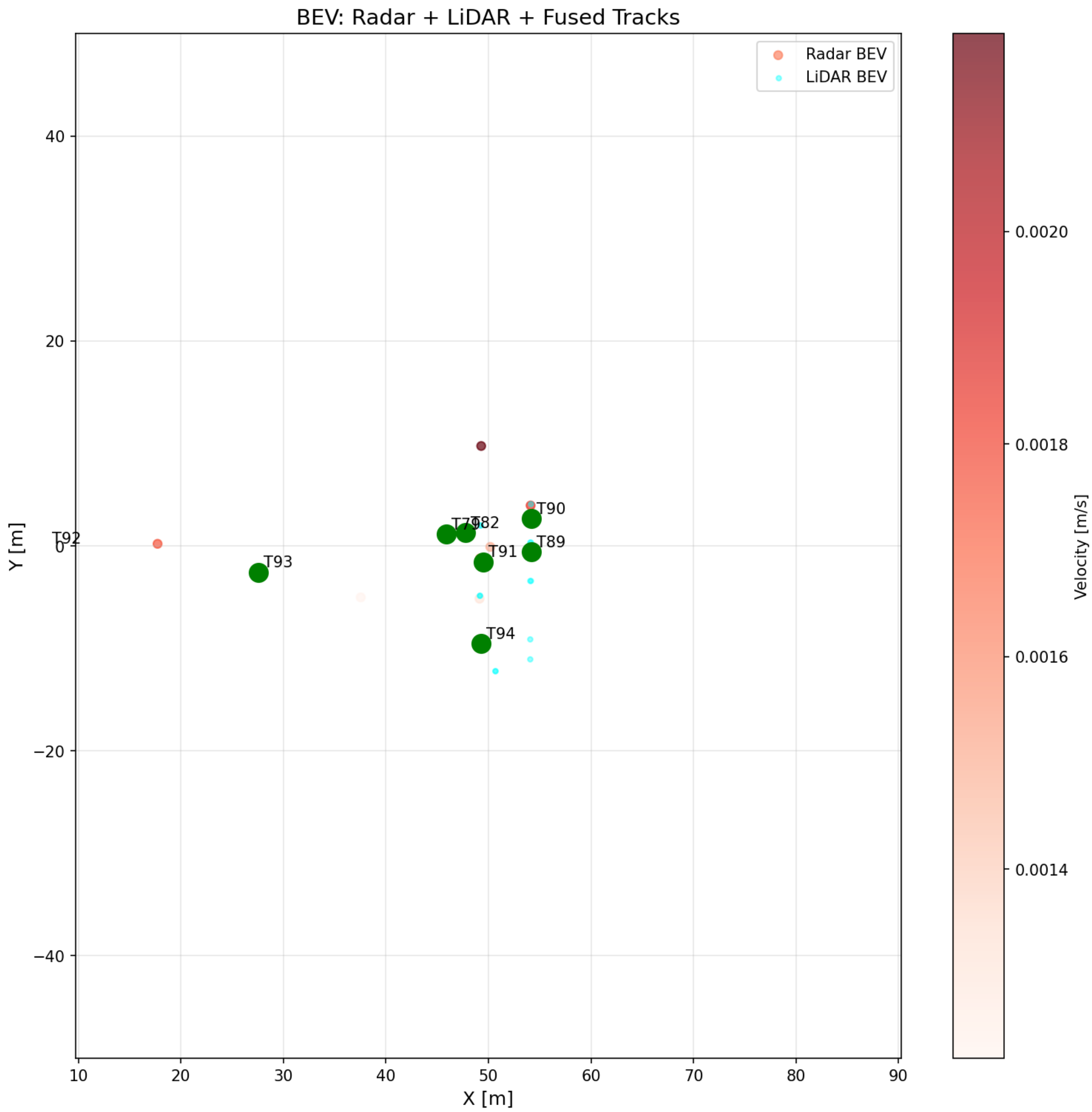
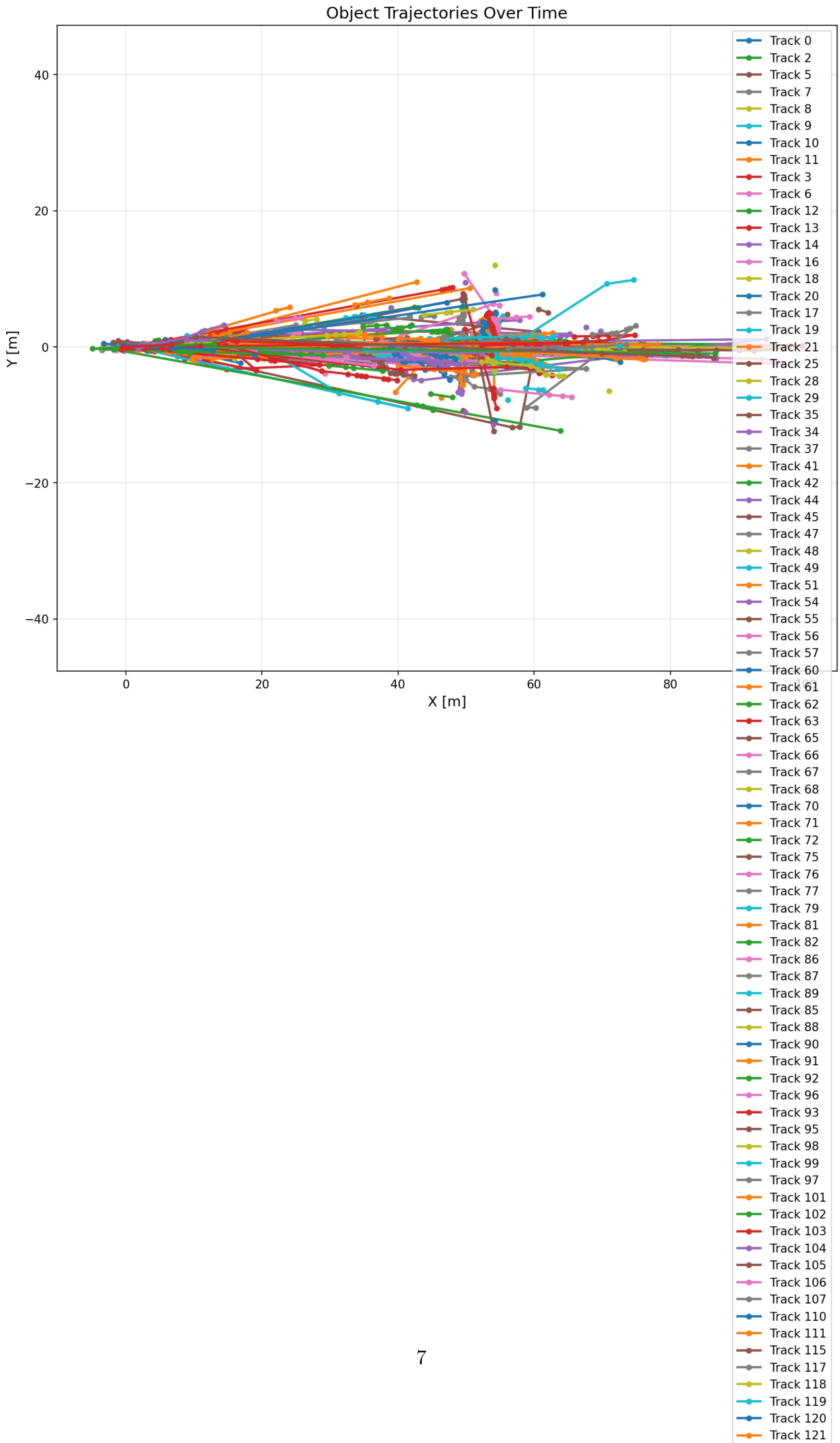


Figure 1: Advanced BEV at Frame 500: Statistical LiDAR filtering and RANSAC ground removal provide a significantly cleaner spatial representation compared to raw projections.

Figure 2 illustrates tracked object trajectories over the entire simulation. The smooth, continuous paths demonstrate successful track maintenance despite sensor noise and intermittent detections.



Object Trajectories Over Time
Y [m]
X [m]
40
20
0
−20
−40
0
20
40
60
80
Track 0
Track 2
Track 5
Track 7
Track 8
Track 9
Track 10
Track 11
Track 3
Track 6
Track 12
Track 13
Track 14
Track 16
Track 18
Track 20
Track 17
Track 19
Track 21
Track 25
Track 28
Track 29
Track 35
Track 34
Track 37
Track 41
Track 42
Track 44
Track 45
Track 47
Track 48
Track 49
Track 51
Track 54
Track 55
Track 56
Track 57
Track 60
Track 61
Track 62
Track 63
Track 65
Track 66
Track 67
Track 68
Track 70
Track 71
Track 72
Track 75
Track 76
Track 77
Track 79
Track 81
Track 82
Track 86
Track 87
Track 89
Track 85
Track 88
Track 90
Track 91
Track 92
Track 96
Track 93
Track 95
Track 98
Track 99
Track 97
Track 101
Track 102
Track 103
Track 104
Track 105
Track 106
Track 107
Track 110
Track 111
Track 115
Track 117
Track 118
Track 119
Track 120
Track 121

4.3 Quantitative Metrics

Metric	Value
Total Frames Processed	5615
Average Radar Points/Frame	8
Average LiDAR Points/Frame	180
Total Unique Tracks	263
Tracking Method	Particle Filter

Table 1: System Performance Statistics

4.4 Critical Analysis & Future Work

4.4.1 Limitations

1. **Naive Data Association**: Nearest-neighbor matching fails in dense traffic scenarios with crossing trajectories. A Hungarian algorithm or Joint Probabilistic Data Association (JPDA) would improve robustness.

2. **Fixed Fusion Weights**: The 70/30 LiDAR-radar weighting is heuristic. Adaptive fusion based on measurement uncertainty (e.g., Kalman innovation covariance) would be more principled.

3. **No Occlusion Handling**: When LiDAR support is absent, the system falls back to radar-only tracking without explicitly modeling occlusion events.

4. **Constant Velocity Assumption**: The motion model ignores acceleration. Incorporating IMU data or using a Constant Acceleration model would improve prediction during maneuvers.

4.4.2 Proposed Enhancements

- **Track-to-Track Fusion**: Instead of sensor-level fusion, maintain separate radar and LiDAR tracks and fuse at the track level using a federated Kalman filter.

- **Deep Learning Integration**: Replace DBSCAN with a learned object detector (e.g., PointPillars for LiDAR, radar CNN) for improved detection recall.

- **Temporal Consistency**: Add track smoothing (e.g., Rauch-Tung-Striebel smoother) for offline trajectory refinement.

5 Conclusion

This work demonstrates a functional radar-LiDAR fusion pipeline for autonomous vehicle perception. By combining DBSCAN clustering, Extended Kalman filtering, and weighted sensor fusion, the system achieves robust multi-object tracking in simulation. The modular design facilitates future extensions, including advanced data association, adaptive fusion strategies, and integration with higher-level planning modules. The Week 2 project establishes a foundation for more sophisticated perception architectures in subsequent portfolio milestones.