



Robust Automated Driving in Extreme Weather

Project No. 101069576

Deliverable D5.2

SW on Adaptive Sensor Fusion and Perception Solutions – Final report

WP5 – Robust perception system

Authors	Eren Erdal Aksoy (HH), Idriss Gouigah (HH), Adam Ahlman (HH), Adam Taylor (HH), Mahmoud Alshaikh (HH), Joel Ekstrand (HH), Gayan Himesha Pathirage (HH), Abu Mohammed Raisuddin (HH), Thu Ngo (HH), Nour Alhuda Albashir (HH), Lars Pernickel (HH), Danial Hamoud (HH)
Lead participant	HH
Delivery date	28 August 2025
Dissemination level	Public
Type	Document, report

Version 1



Funded by the European Union (grant no. 101069576). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them. UK and Swiss participants in this project are supported by Innovate UK (contract no. 10045139) and the Swiss State Secretariat for Education, Research and Innovation (contract no. 22.00123) respectively.

Revision history

Author(s)	Description	Date
Eren Erdal Aksoy (HH)	First draft, report structure	15/08/2025
Joachim Glaess (KO)	Revision 1 (for internal review)	22/08/2025
Stefan Parzefall (AVL)	Revision 2 (update after review)	22/08/2025
Eren Erdal Aksoy (HH)	Revised based on Revision 2	28/08/2025
Johannes Ripperger (accelCH)	Formatting and formal check	28/08/2025

Contents

List of Figures	4
List of Tables	4
Partner short names	5
Abbreviations	5
Executive summary	6
Objectives	6
Methodology and implementation	6
Outcomes	6
Next steps	6
1 Introduction	7
1.1 Sensor Fusion and Object Detection	7
1.1.1 Related Work	8
1.1.2 Fusion Method	8
1.1.3 Experimental Results	10
1.1.4 LiDAR and RGB-D Fusion for 3D Object Detection on NVIDIA Jetson AGX Orin	11
1.2 Free-Space Detection	12
1.3 Weather-type Detection	13
1.4 Trajectory Prediction	15
2 Functional Implementation	16
3 Relation to other deliverables	16
4 Conclusions and Continuous Improvement	16
5 References	17
6 Appendix	19

List of Figures

Figure 1 ROADVIEW Perception Architecture. See D2.3 for a more comprehensive representation. The red, green, blue, and yellow colours represent the synchronized RGB camera, LiDAR, RADAR, and Thermal cameras, respectively. The dashed lines are for the filtered data streams.	7
Figure 2 Illustration of the characteristics of LiDAR, RADAR, and Camera sensors. Contextually, LiDAR-RADAR-Camera fusion has the highest complementary properties.	8
Figure 3 ROADVIEW Adaptive sensor fusion pipeline for 3D object detection. The pipeline has two fusion methods. The early fused three sensor readings, shown in the red branch, are passed to a joint voxel feature extractor. The mid-level gated fusion is then applied to learn weights for each sensor stream (i.e., red, green, and blue branches) and fuse all multi-modal features adaptively.	9
Figure 4 Early fusion strategy	9
Figure 5 Mid-level gating module.....	9
Figure 6: Sample qualitative results showing detected object bounding boxes on nuScenes.	10
Figure 7 Mobile robot platform with onboard 3D LiDAR and RGB-D sensors	11
Figure 8 Flowchart the hybrid fusion architecture, combining the early, mid, and late fusion strategies.	12
Figure 9 A sample scene with detected semantic classes on nuScenes.	12
Figure 10 The LRC-WeatherNet architecture. Early fused LiDAR-RADAR BEV features are further fused with RGB features via a mid-level gated fusion module. Raw sensor outputs are passed through different backbone encoders	13
Figure 11 Sample qualitative results showing a spring rainy (left) and spring snowy (right) scenes predicted accurately.....	14
Figure 12 ByteTrack's tracking predictions at different time steps: Frame 1, 5, 20, and 70.	15

List of Tables

Table 1 Object detection performance on the nuScenes validation set	10
Table 2 Comparison of the mAPs for IoU thresholds of 25% and 50%, including the FPS of each model.	11
Table 3 Multi-class semantic segmentation results on the ROADVIEW Snowy Scenes test dataset.	12
Table 4 Quantitative performance evaluation across different unimodal and fusion configurations.	14
Table 5 Object tracking results on CADC (higher ↑ is better; lower ↓ is better.	15

Partner short names

HH	Halmstad University
LUA	Lapin Ammattikorkeakoulu
THI	Technische Hochschule Ingolstadt
VTI	Statens Vag- och Transportforskningsinstitut
CE	CEREMA
RISE	RISE Research Institutes OF Sweden
FGI	Maanmittauslaitos – Finnish Geospatial Research Institute
R5	Repli5
KO	Konrad GMBH
FORD	Ford Otomotiv Sanayi A.S
CRF	Canon Research Centre France S.A.S.
accelCH	accelopment Schweiz AG
VTT	Technical Research Centre of Finland
AVL	AVL Software and Functions GmbH
QMUL	Queen Mary University of London

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
AD	Automated Driving
BEV	Bird-Eye-View
GA	Grant Agreement
GNSS	Global Navigation Satellite System
HD	High Definition
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
MS	Milestone
RADAR	Radio Detection And Ranging
WP	Work Package

Executive summary

This document is an official public deliverable of the ROADVIEW project funded under the Horizon Europe Framework Programme within the call HORIZON-CL5-2021-D6-01-01 "More powerful and reliable on-board perception and decision-making technologies addressing complex environmental conditions (CCAM Partnership)". This deliverable reports the final version of D5.1 "*SW on Adaptive Sensor Fusion and Perception Solutions - First report*" as a part of Task 5.1 "*Adaptive Sensor Fusion and Perception Solutions*", developed within WP5 dedicated to "*Robust perception system*".

More specifically, this deliverable details the final report concerning the algorithms and software implementation of the modules listed below:

- **Sensor Fusion:** To fuse Camera, LiDAR, and RADAR data streams.
- **Object Detection:** To detect pedestrians, vehicles, and other agents (e.g., cyclists) in the environment.
- **Free-space Detection:** To detect drivable free areas in the scene.
- **Weather type Detection:** To distinguish between different types of weather (e.g., fog, rain, snow, etc.).
- **Trajectory Prediction:** To track detected objects in the scene.

The deliverable consists of three separate scientific reports focusing on these topics and algorithms listed above. Some of the algorithms covered will be submitted as manuscripts to international peer-reviewed conferences. For a more comprehensive understanding, all manuscripts are also appended to this deliverable. The first two papers are dedicated to our sensor fusion algorithms for object detection and weather type classification. These papers are based on the final version of the ROADVIEW adaptive sensor fusion algorithm, which learns to adaptively weight the importance of each modality by combining early- and mid-level fusion methods. More specifically, the first paper describes how we customized this fusion algorithm for deployment on our mobile robot platform, which is equipped with LiDAR and RGB-D sensors. This customized version uses early, mid, and late fusion mechanisms to achieve close-to-real-time performance on embedded devices, such as the NVIDIA Orin, which is used in different ROADVIEW demonstration vehicles in WP8. The second paper applies our adaptive sensor fusion strategy to a state-of-the-art feature extraction model to classify nine weather conditions using LiDAR, 4D RADAR, and RGB cameras. The final paper focuses on detecting, tracking, and predicting the trajectories of objects detected by our sensor fusion algorithm, particularly in snowy weather conditions. This deliverable, D5.2, forms part of Milestone 12 ("*Perception algorithms and SW ready*"), which is due in Month 36. It is important to note that the dissemination level for D5.2 is public.

Objectives

The main objective of this deliverable is to provide the final update on T5.1 developed in WP5. The algorithms and perception solutions provided are being tested in the ROADVIEW demonstration vehicles in WP8 to provide an overview of the outcome of T5.1.

Methodology and implementation

This deliverable is an extension of deliverable D5.1, presenting an early version of our perception solutions. Note that both D5.1 and D5.2 are based on D2.4, "Requirements of the Physical System Setup," which provides a comprehensive description of perception- and control-related functional components, their requirements, and priority levels during the implementation and integration phases of the ROADVIEW reference architecture. The actual content of this final deliverable is these perception-related functional requirements.

Outcomes

The outcome of this deliverable is the final perception solutions to be used in different ROADVIEW demonstrations.

Next steps

The algorithms developed in this deliverable are being integrated into different demo vehicles for the upcoming demonstrations in WP8.

1 Introduction

The ROADVIEW perception stack involves a set of AI-based perception modules that provide a semantically enriched weather-aware representation of the vehicle surroundings. This perception stack covers the fusion of different sensor modalities (e.g., RGB cameras, 3D LiDARs, and 4D RADARs), object detection, free-space detection, and weather-type detection, among others. The functional requirements of each perception solution, together with their sensor modalities and hardware needs, were already identified in deliverable D2.4 “Requirements of the physical system setup”. Figure 1 shows the final ROADVIEW perception architecture, which has been detailed in D2.3 “ROADVIEW system reference architecture”.

Recalling the ROADVIEW functional perception modules described in D2.4, we, in this deliverable, provide the final implementation details of the following five perception solutions:

- **Sensor Fusion:** To fuse Camera, LiDAR, and RADAR data streams.
- **Object Detection:** To detect pedestrians, vehicles, and other agents (e.g., cyclists) in the environment.
- **Free Space Detection:** To detect free drivable areas in the scene.
- **Weather-type Detection:** To distinguish between different types of weather (e.g., fog, rain, snow, etc.)
- **Trajectory Prediction:** To predict and anticipate the trajectory of detected objects (e.g., vehicles).

Note that although the trajectory prediction module is classified as a planning module in the ROADVIEW reference architecture in D2.4, we provide the implementation details in this deliverable dedicated to the perception solutions. The remaining perception solutions in Figure 1 are detailed in other relevant deliverables, such as Data Filtering in D4.8. Implementation details for Slipperiness Detection and Visibility Detection can be found in D5.5.

In the following subsections, we provide brief overviews of the final implementation of these five perception solutions listed above. For more technical descriptions of some of our methods, we provide the comprehensive scientific manuscripts attached to this deliverable (See the Appendix in 6).

1.1 Sensor Fusion and Object Detection

Robust object detection is a crucial perception component for autonomous vehicles to enrich the semantic understanding and situation awareness of the vehicle surroundings in three-dimensional space. This component helps vehicles have safer navigation by accurately identifying and localizing surrounding vehicles and vulnerable road users, such as pedestrians and cyclists, among others.

Autonomous vehicles are equipped with different sensors, each of which has different characteristics to increase autonomy. Figure 2 provides a map illustrating the unique characteristics of LiDAR, RADAR, and RGB camera sensors. Although RGB cameras are cheap and provide semantically rich scene information, they are, unlike RADAR and LiDAR sensors, not robust to challenges related to adverse weather conditions. On the other hand, RADARs can sense the object's velocity very precisely and return depth information with limited sensor resolution. Despite the cost, LiDARs can provide high-precision 3D point cloud data, leading to accurate depth perception. To cope with the downgraded perceptual capability due to these limitations of individual sensors, the readings coming from each modality need to be fused, leading to more comprehensive and accurate scene perception. In this respect, by combining the advantages of different sensor types, sensor fusion compensates for their individual limitations and integrates different sensors' complementary strengths to gain resilience against diverse environmental conditions

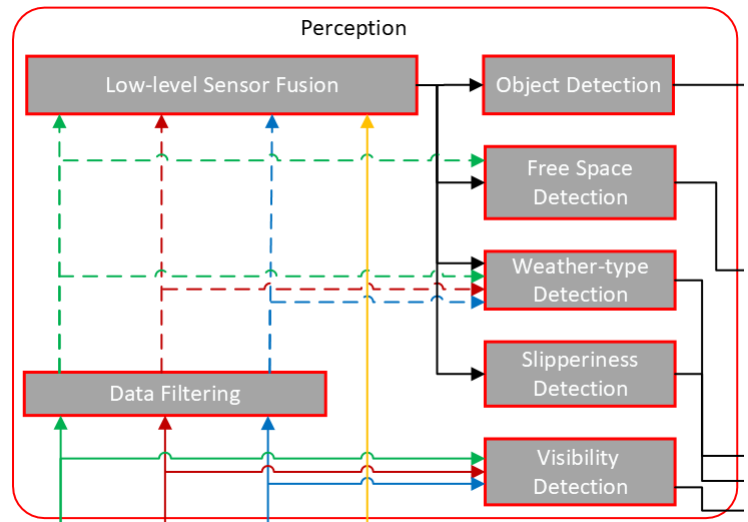


Figure 1 ROADVIEW Perception Architecture. See D2.3 for a more comprehensive representation. The red, green, blue, and yellow colours represent the synchronized RGB camera, LiDAR, RADAR, and Thermal cameras, respectively. The dashed lines are for the filtered data streams.

and improve the overall effectiveness of the downstream tasks, such as 3D object detection. However, introducing more sensors in the fusion pipeline also creates a need for effective strategies to minimize redundancy, handle missing data, and manage computational complexity [1].

1.1.1 Related Work

There exist different types of fusion strategies. Early fusion integrates the raw sensor readings before applying any further downstream processing, such as object detection [2]. Due to relying solely on raw sensor information, early fusion has the advantage of not leaking any sensor readings. Late fusion stands out as an alternative information aggregation method to merge high-level predictions derived individually from each modality. As in the case of [3], a semantic consistency filter is applied as a plug-and-play late fusion strategy for LiDAR and camera data streams.

Contemporary research on multimodal fusion-based object detection tends to unify features coming from different sensors in a global latent space. These mid-level fusion strategies can capture descriptive scene features unique to each modality. Along these lines, the Bird's Eye View (BEV) Fusion model BEVFusion [4] combines image and LiDAR features in the BEV space to jointly solve multiple tasks such as 3D object detection and map segmentation. PointPainting [5] enhances 3D object detection by transferring semantic information from camera images to the 3D LiDAR points, effectively leveraging the high-resolution semantic cues from cameras to improve object localization and classification in sparse LiDAR point clouds. In a similar vein, our previous work Aug-VirConv [6] augments sparse LiDAR point clouds with denser pseudo points by solely relying on the LiDAR-based scene semantics, without requiring cameras, for 3D object detection. On the other hand, CR3DT fuses a camera and RADAR for object detection [7].

Transfusion [8] introduces a transformer-based architecture to fuse camera and LiDAR readings to generate a heatmap indicating the object center to further regress the bounding box information. Decorating fusion [9] constitutes an alternative mid-level fusion strategy aimed at integrating feature-level and point-level representations. This approach enables the system to leverage explicit associations between modalities through calibration matrices, while also learning implicit connections between point clouds and images. Initially, feature extraction is performed using a two-dimensional convolutional neural network, which subsequently enhances the representation of LiDAR points. During the mid-level fusion stage, queries combined with a center heatmap help to adjust the initial positions, bringing them closer to the target locations. Alternatively, deep fusion [10] fuses LiDAR BEV with LiDAR front view and camera images by combining region-wise features pooled from each view.

When it comes to 3D object detection, most architectures either process raw point clouds directly [11] or encode them in the voxel space with regular grids [12] or convert them into the Bird's Eye View (BEV) representation [13] to further apply sparse 3D convolution [14]. VoxelNet [12] converts unordered point cloud data into structured voxels and applies 3D convolutions for feature extraction. CenterFusion [15] identifies objects by designating center points, eliminating the need for fixed-size anchor boxes. Voxel R-CNN [16] further improves point cloud feature representation through voxel pooling. The seminal work PointPillars [17] groups raw 3D points into grid-based pillars to apply 2D convolution to the resulting pseudo-image.

1.1.2 Fusion Method

The ROADVIEW project proposes a new adaptive sensor fusion framework for 3D object detection that applies a mix of early- and mid-level fusion strategies to aggregate information from multiple 2D RGB cameras, 3D LiDARs, and 4D RADAR modalities. The ROADVIEW adaptive sensor fusion heavily relies on adaptive gated networks [18] [19] [20], which learn weights adapted to each modality during fusion to selectively filter and merge information from different modalities. In a nutshell, our applied gating mechanism assigns weights to each input sensor, enabling the model to identify and prioritize the most important sensor features during fusion. The gating mechanism could be considered as a mixture of experts, where networks dedicated to different modalities act as experts.

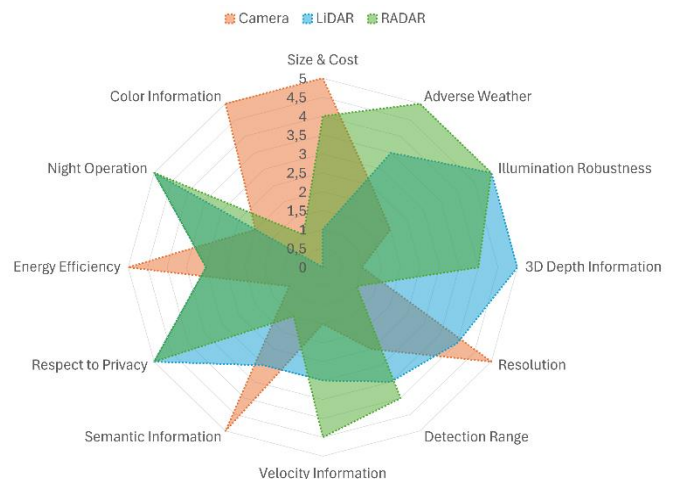


Figure 2 Illustration of the characteristics of LiDAR, RADAR, and Camera sensors. Contextually, LiDAR-RADAR-Camera fusion has the highest complementary properties.

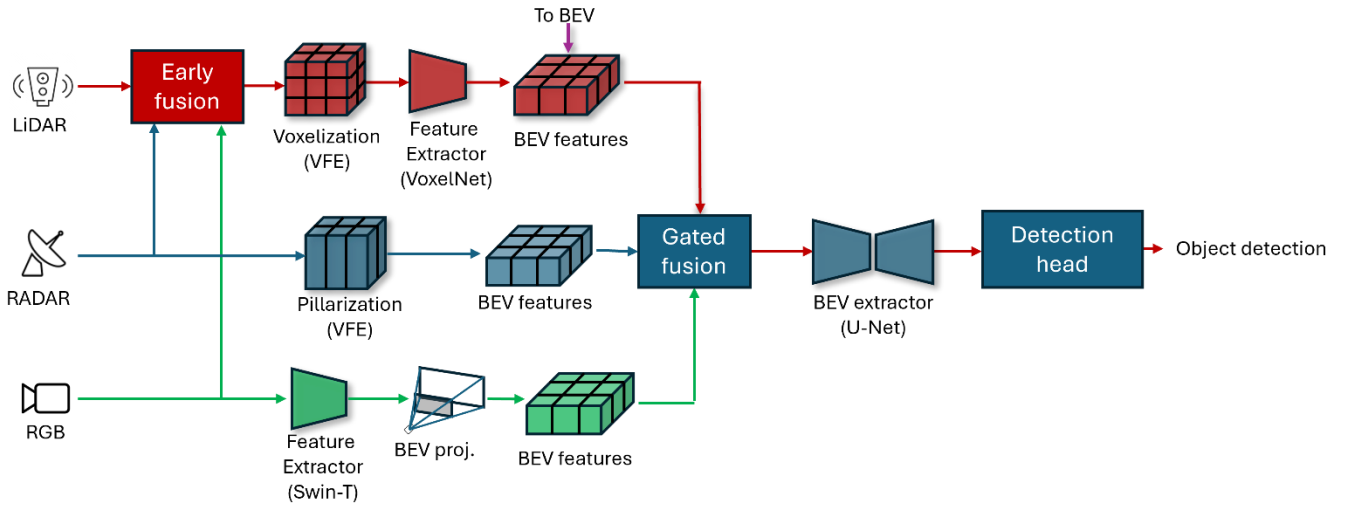


Figure 3 ROADVIEW Adaptive sensor fusion pipeline for 3D object detection. The pipeline has two fusion methods. The early fused three sensor readings, shown in the **red branch**, are passed to a joint voxel feature extractor. The mid-level gated fusion is then applied to learn weights for each sensor stream (i.e., **red**, **green**, and **blue** branches) and fuse all multi-modal features adaptively.

Figure 3 illustrates the ROADVIEW adaptive sensor fusion pipeline for 3D object detection. The pipeline is composed of two crucial fusion steps: an **early fusion** of raw sensor readings is first introduced, followed by the application of the mid-level **gated fusion** of features extracted in each modality branch, individually. The main motivation for applying early fusion is to extract much denser point cloud data where all available feature information (such as accurate 3D position, color, and velocity cues) is fused in the voxelized space and passed to a common feature extractor, which is based on VoxelNet [12] [18]. The RADAR branch is based on the pillar representation, as in PointPillars [17]. The RGB camera branch relies on the Swin Transformer model introduced in [21]. In mid-level gated fusion, each sensor branch acts as an individual expert delivering complementary information in BEV space. Following the gated fusion, the BEV feature extractor is applied, the output of which is directly fed to the object detection head to predict 3D object bounding boxes.

Figure 4 details our early fusion strategy, which fuses raw RGB camera, LiDAR, and RADAR inputs at the earliest stage to extract richer voxel features. In LiDAR readings, we store position and average intensity. In RADAR readings, we keep the position data together with the RADAR Cross Section (RCS) and velocity information since they play crucial roles in distinguishing static objects from dynamic ones. The camera data has only color information. We apply Multi-layer Perceptrons (MLP) to extract camera and RADAR features. A simple concatenation of these camera, LiDAR, and RADAR features forms the final voxel feature, named “Enhanced LiDAR voxel feature” in Figure 4.

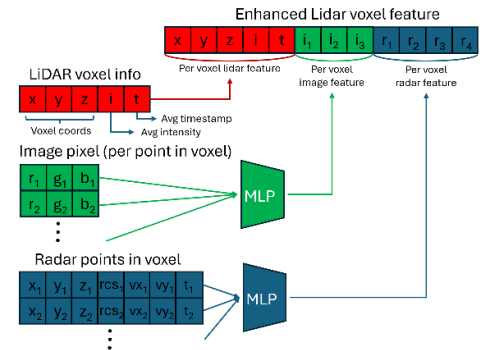


Figure 4 Early fusion strategy

Our mid-level fusion strategy is illustrated in Figure 5. Once each sensor feature is extracted, they are represented in Bird’s Eye View (BEV) by flattening over the z-axis. These BEV features form the core inputs (F_L , F_R , and F_C) for the mid-level fusion, which performs adaptive sensor fusion on these feature maps using a gating mechanism [18] [19] [20]. The core idea here is to adaptively learn individual weights for each sensor modality over the channel dimension. More specifically, all three feature maps (F_L , F_R , and F_C) are first concatenated

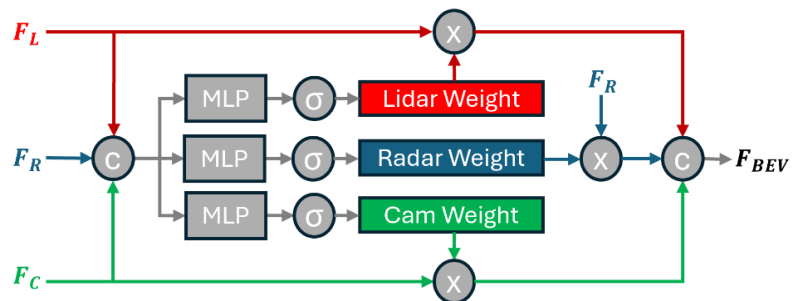


Figure 5 Mid-level gating module

(C) in the channel dimension. Next, these concatenated feature maps are passed through MLP layers and sigmoid functions (σ) to learn individual weights for each modality. We then apply element-wise multiplication ($\times$) between the learned weights and the original input feature maps (F_L, F_R , and F_C). The obtained gated camera, LiDAR, and RADAR features are finally concatenated to form the output feature map (F_{BEV}).

1.1.3 Experimental Results

We evaluate the object detection performance of our fusion algorithm on the nuScenes dataset [22]. As the evaluation metrics, we employ the mean Average Precision (mAP) and nuScenes Detect Score (NDS) [22].

Table 1 presents the object detection performance of our fusion algorithm on the nuScenes validation set. In contrast to LiDAR-only (L) results, we obtain higher NDS and mAP scores when we fuse LiDAR with RADAR (LR) or LiDAR with Camera (LC). These results clearly show that our fusion strategy learns to extract the most descriptive features from each modality. Likewise, the highest scores are obtained when fusing all three modalities (LRC). As a main baseline, we employ the LiRa framework introduced in [18]. In contrast to LiRa, we obtain slightly lower NDS and mAP scores.

Table 1 Object detection performance on the nuScenes validation set

Method	Modality	NDS $\uparrow$	mAP $\uparrow$
Ours	L	59.9	55.5
Ours	LR	61.9	57.2
Ours	LC	60.5	63.3
Ours	LRC	65.0	63.4
LiRa	LRC	68.6	64.7

Figure 6 shows some sample detection results after running our fusion algorithm on nuScenes.

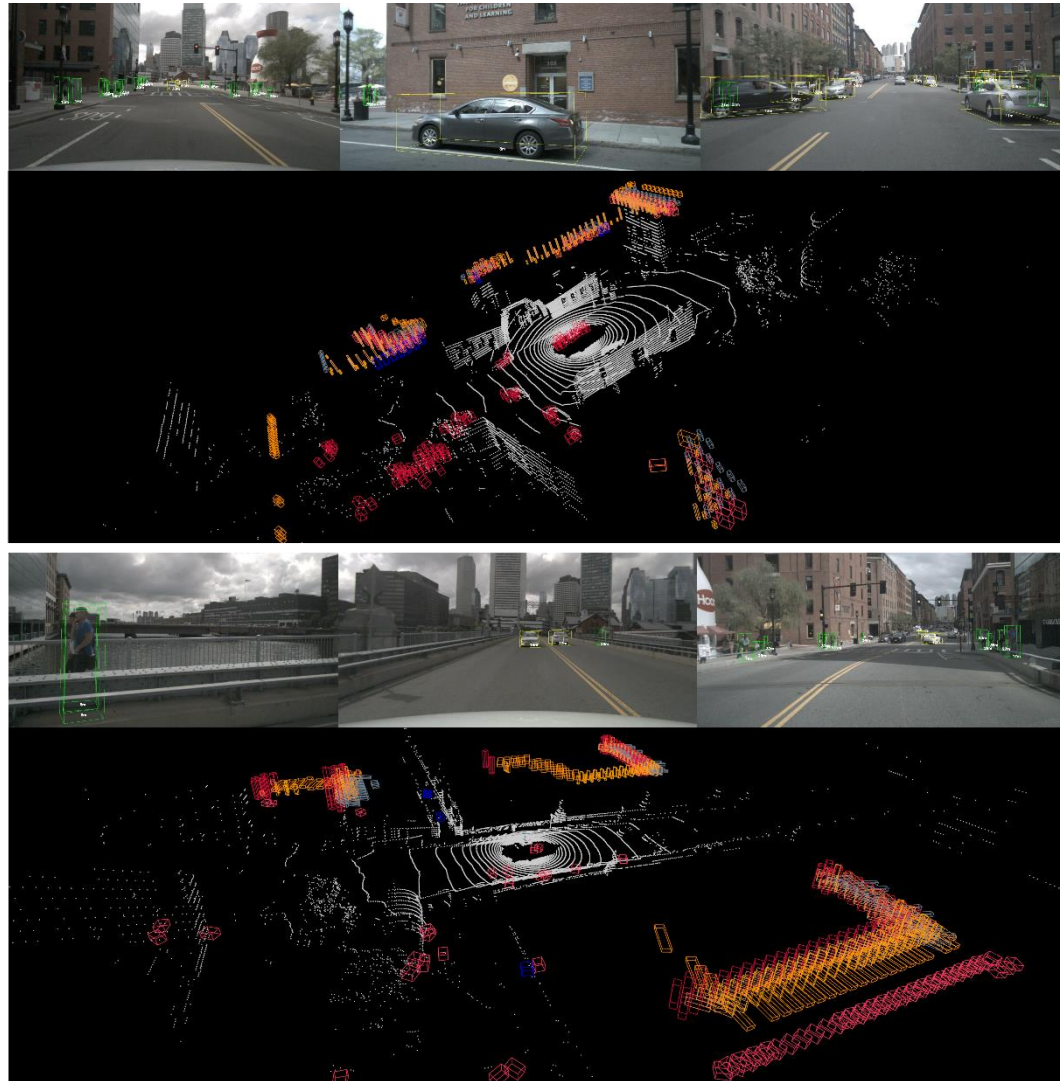


Figure 6: Sample qualitative results showing detected object bounding boxes on nuScenes.

1.1.4 LiDAR and RGB-D Fusion for 3D Object Detection on NVIDIA Jetson AGX Orin

Sensor fusion is an essential component of autonomous robotic systems. It can be used to improve the accuracy of various tasks, such as perception and localization. However, advanced perception models, such as 3D object detection, have high computational demands, making sensor fusion challenging to implement in embedded environments. Since autonomous vehicles and mobile robots operate under strict energy and hardware constraints, the size of perception models is an important factor, further highlighting the need for compact and efficient computational hardware.

In this deliverable, we further explore the applicability of early, mid, and late fusion of 3D LiDAR and RGB-D sensor readings on the NVIDIA Jetson AGX Orin embedded platform in an indoor environment. More specifically, we deploy various versions of our sensor fusion-based object detection models explained in Section 1.1.2 on the NVIDIA Jetson AGX Orin, a compact edge computing platform, using real-world data from a mobile robot operating in indoor environments, as shown in Figure 1Figure 7. Note that the same embedded hardware is to be used in the ROADVIEW demo vehicle platforms in WP8.

RGB-D cameras combine colour and depth data, enabling more enriched scene understanding than traditional 2D images alone. However, they are limited by short depth range and sensitivity to lighting conditions. LiDAR sensors provide high-resolution 3D point clouds, offering reliable spatial data across various lighting conditions. While LiDAR provides high depth accuracy, it lacks semantic information like colour, making it difficult to distinguish between objects with similar geometric shapes.

Sensor fusion methods are commonly categorized into early, mid, and late fusion. Early fusion combines raw data from multiple modalities before feature extraction, mid-level fusion integrates learned features from individual sensors, and late fusion merges decisions or final predictions from separate models. Since each approach has trade-offs in terms of complexity, efficiency, and performance, we propose a new hybrid fusion strategy that combines early, mid, and late fusion methods to leverage the strengths of both techniques, as shown in Figure 8. Note that the early and mid-fusion methods are already described in Section 1.1.2. The additional late fusion method is nothing else than applying a confidence majority voting to the unimodal (i.e., RGB-D-only and LiDAR-only) predictions.

The obtained results (see Table 2) show that our hybrid fusion, which integrates early, mid, and late fusion strategies, attains the highest mAP among all evaluated methods, demonstrating that a combination of low-, mid-, and decision-level fusions is particularly effective when the individual modality predictions are already strong. Considering both accuracy and latency, our hybrid fusion model (last row in Table 2) maintains a relatively high FPS (4.07) compared to pure late fusion or combinations of early and late fusion. A combination of early and mid-fusion methods has a better performance than the Camera-only model but cannot surpass the LiDAR-only model.

Paper 01 [23] (attached in the Appendix in Section 6) provides a comprehensive description of this hybrid sensor fusion framework. Note that this work [23] will be submitted to the IEEE Intelligent Vehicles Symposium (IV) 2026. To encourage further research on the subject, we also release our source code and trained models: https://github.com/Adam20Taylor/AGX_Orin-Fusion.

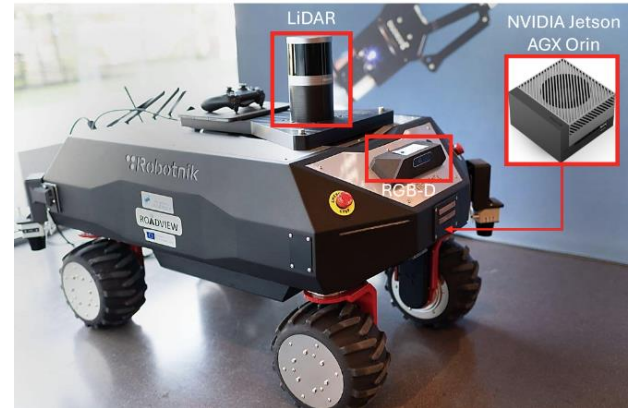


Figure 7 Mobile robot platform with onboard 3D LiDAR and RGB-D sensors

Table 2 Comparison of the mAPs for IoU thresholds of 25% and 50%, including the FPS of each model.

	mAP		FPS
	@0.5	@0.25	
LiDAR-only	0.210	0.359	7.52
Camera-only	0.076	0.193	6.13
Early Fusion	0.192	0.192	7.19
Mid Fusion	0.061	0.121	7.35
Early + Mid Fusion	0.152	0.152	8.40
Concatenation Fusion	0.030	0.061	7.87
Early + Concatenation Fusion	0.091	0.091	8.70
Late Fusion	0.215	0.379	3.06
Early + Late Fusion	0.253	0.397	3.86
Early + Mid + Late Fusion	0.260	0.403	4.07

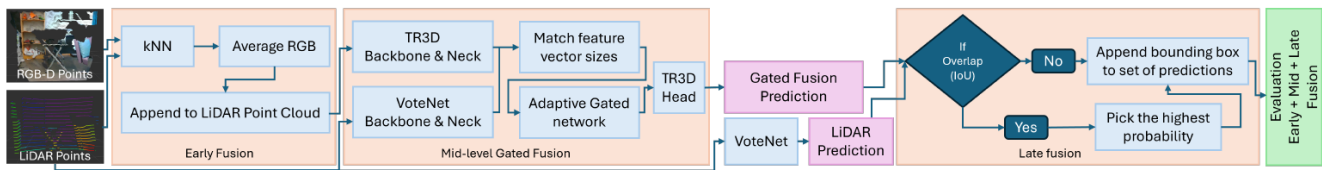


Figure 8 Flowchart the hybrid fusion architecture, combining the early, mid, and late fusion strategies.

1.2 Free-Space Detection

The free-space detection module has already been introduced in the first version of this deliverable, i.e., in D5.1 “SW on Adaptive Sensor Fusion and Perception Solutions - First report”. In addition to this model in D5.1, we also trained customized versions of different well-known 3D semantic segmentation models, such as SalsaNext [24], PTv3 [25], and Cylinder3D [26] using the ROADVIEW Snowy Scenes dataset introduced in [27]. Table 3 shown below reports the accuracy of different semantic segmentation models. Note that the seventh column (named “road”) in this table refers to the detected free space. This table indicates that projecting 3D LiDAR point clouds into a cylindrical coordinate system can lead to higher performance (78.6 mIOU achieved by Cylinder3D). Figure 9 shows a sample scene with the obtained semantic segments on the nuScenes dataset [22]. Note that in this figure, the orange colour represents the free space. These quantitative and qualitative results show that our AI-based model can accurately detect drivable free spaces in the scene.

Table 3 Multi-class semantic segmentation results on the ROADVIEW Snowy Scenes test dataset.

Model	car	bicycle	on-rails	truck	other-veh	person	road	parking	sidewalk	other-gro	building	fence	other-stru	vegetation	terrain	pole	bus	traffic-sig	other-obj	moving-c	moving-p	moving-o	moving-t	moving-o	falling-sp	mIOU
SalsaNext [23]	71.0	69.2	84.4	76.7	72.3	68.4	80.3	69.6	68.0	65.6	69.9	58.6	74.4	76.6	65.8	52.6	60.7	57.2	51.7	80.8	68.7	47.8	65.4	89.5	56.7	68.1
PTv3 [24]	92.9	66.9	66.2	59.7	63.4	82.6	75.9	64.8	61.9	21.0	94.9	83.6	92.4	94.4	72.3	79.2	77.3	81.4	71.2	86.3	85.9	49.1	41.4	96.9	77.6	73.6
Cylinder3D [25]	91.1	83.9	68.9	83.6	62.6	84.6	86.2	83.7	79.2	79.4	88.3	87.0	94.2	92.1	82.6	75.6	43.0	82.4	78.3	89.5	81.3	42.9	45.8	97.0	82.9	78.6

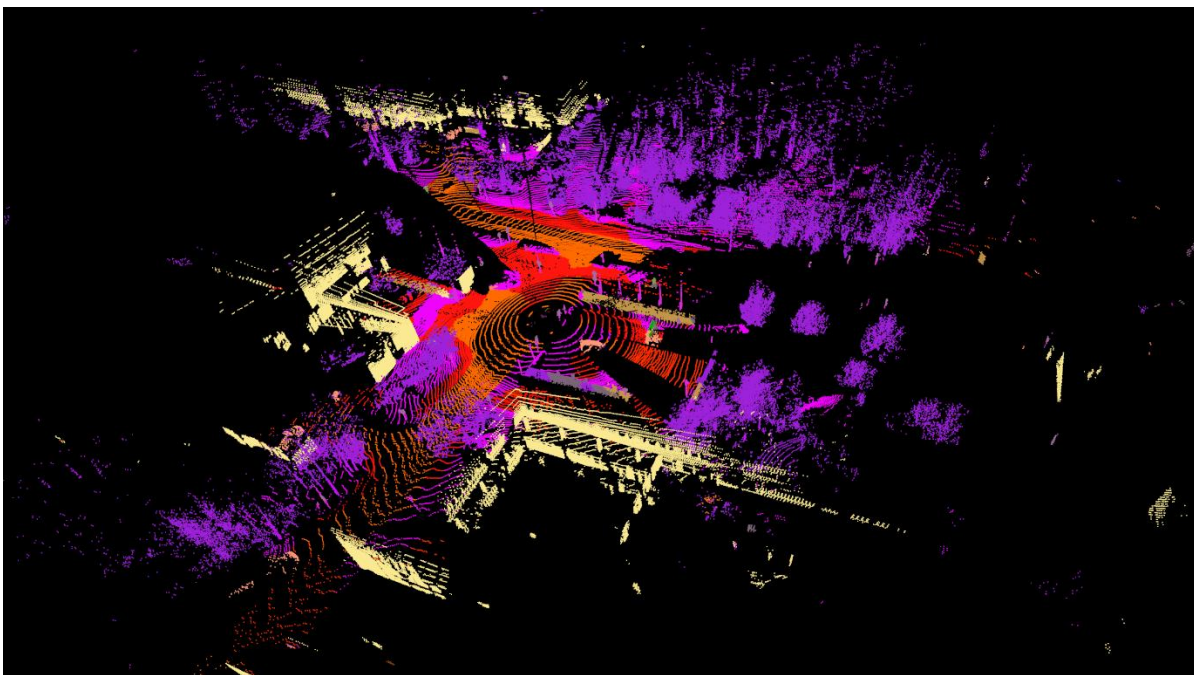


Figure 9 A sample scene with detected semantic classes on nuScenes.

1.3 Weather-type Detection

The ROADVIEW project proposes a novel weather-type detection algorithm that classifies different weather conditions (rain, snow, clean, etc.) by applying a combination of early- and mid-level fusion strategies to 2D and 3D sensor modalities.

As shown in Figure 2, each sensor type offers distinct advantages and limitations. For instance, RADARs maintain robust functionality in challenging weather such as heavy rain or fog; however, occasionally, they misinterpret reflections from nearby raindrops as false signals, introducing inaccuracies at close ranges. LiDARs provide precise 3D perception; however, their effectiveness diminishes significantly in dense fog or precipitation. Cameras are good at capturing rich visual cues with color and contextual details, yet their utility drops in suboptimal weather conditions. Fusing readings from these complementary sensors and exploiting their unique information patterns reduces the shortcomings of each sensor, offering enhanced performance in challenging weather and seasonal conditions.

We address the challenge of robust weather condition classification for autonomous vehicles through a multi-sensor fusion framework. Leveraging the complementary sensing capabilities of LiDAR, RADAR, and camera, we propose a new model, named LRC-WeatherNet shown in Figure 10, integrating readings from all three modalities in real time to improve the accuracy of weather condition recognition. Similar to our previous sensor fusion for object detection work, explained in Section 1.1, this fusion pipeline explores both early fusion, where LiDAR and RADAR features are projected into a unified Bird's Eye View (BEV) representation, and mid-level gated fusion, shown in Figure 5, which combines modality-specific feature maps after backbone processing. Thus, our fusion strategy learns inherent strengths and weaknesses of each modality under different environmental conditions. To assess model performance and sensor contributions, experiments with different backbone architectures (e.g., EfficientNet-B0 [28] and PointPillars [17]) are conducted.

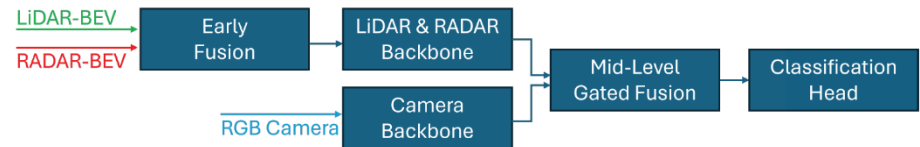


Figure 10 The LRC-WeatherNet architecture. Early fused LiDAR-RADAR BEV features are further fused with RGB features via a mid-level gated fusion module. Raw sensor outputs are passed through different backbone encoders

We evaluate the performance of LRC-WeatherNet on the Michigan State University Four Seasons (MSU-4S) dataset [29], which captures a wide range of seasonal and environmental conditions. The classification task includes nine distinct weather conditions, such as rain, spring snow, snow, fall, sunset, late summer, and early fall scenarios. We report a quantitative comparison of our model performance and computational efficiency using metrics such as accuracy, F1-score, confusion matrices, inference time, and the number of operations (measured in GMACs). The obtained results show our lightweight BEV-based multi-modal fusion strategy significantly improves classification performance over different baselines, particularly under challenging weather conditions.

A summary of the quantitative results on MSU-4S [29] is provided in Table 4. The RADAR-only baseline model exhibited limited classification capability when using RADAR BEV inputs exclusively. The LiDAR-only baseline demonstrated a substantial improvement over the RADAR-only approach, achieving a test accuracy of 56.34% and a macro-averaged F1 score of 0.55. The Camera-only model achieved the strongest results among all unimodal baselines, with a test accuracy of 77.86% and a macro-averaged F1 of 0.76. Combining LiDAR and RADAR BEV features through our early fusion improved performance relative to RADAR-only and LiDAR-only unimodal baselines, reaching a test accuracy of 61.06% and a macro F1 score of 0.59. Our early fusion model also maintains computational efficiency, with 4.67 million parameters and an inference time of just 4.21 ms, while providing improved discriminability in challenging conditions. Finally, in our proposed LRC-WeatherNet, the mid-level gated fusion, combining early fused LiDAR and RADAR features with camera image features, produced the best overall performance, achieving 86.66% test accuracy and a macro-averaged F1 score of 0.85.

Our baseline experiments (LRC-WeatherNet-PP), where a PointPillars-based backbone is used for the early fused LiDAR and RADAR data, achieved a classification accuracy of 87.77% and an F1 score of 0.87, demonstrating effective integration of spatial and visual modalities. However, the LRC-WeatherNet-PP baseline becomes substantially expensive due to extremely high computation time of 64.40 ms in contrast to 7.13 ms for the original LRC-WeatherNet, as shown in Table 4.

Table 4 Quantitative performance evaluation across different unimodal and fusion configurations.

Model	Val. Acc.	Test Acc.	Macro F1	Inference (ms)	Params	GMAC
Camera-only	91.96%	77.86%	0.76	4.86	4.67M	0.41
LiDAR-only	57.46%	56.34%	0.55	3.52	4.67M	0.41
RADAR-only	31.99%	27.32%	0.26	3.79	4.67M	0.41
Early Fusion (LiDAR + RADAR)	62.98%	61.06%	0.59	4.21	4.67M	0.41
LRC-WeatherNet	93.82%	86.66%	0.85	7.13	19.17M	0.84
LiDAR-only-PP	83.11%	81.49%	0.81	8.57	4.86 M	29.57
RADAR-only-PP	43.80%	37.80%	0.37	3.51	4.86 M	01.85
Early Fusion-PP (LiDAR + RADAR)	82.10%	82.30%	0.81	8.58	4.86 M	29.57
LRC-WeatherNet-PP	91.67%	87.77%	0.87	64.40	8.68 M	88.67

Figure 11 depicts two sample images from the MSU-4S dataset [5] for the qualitative evaluation. Here, early fused LiDAR and RADAR point clouds are superimposed on the RGB camera image, and the proposed LRC-WeatherNet correctly predicts the right weather type for each scene.

To the best of our knowledge, this is the first work to fuse LiDAR, RADAR, and camera data for real-time weather classification in autonomous driving.

Paper 02 [30], attached in the Appendix in Section 6, is entitled "LRC-WeatherNet: LiDAR, RADAR, and Camera Fusion for Real-time Weather-type Classification in Autonomous Driving" and provides a comprehensive description of our sensor fusion framework for weather-type classification. Note that this work will be submitted for publication to the IEEE Intelligent Vehicles Symposium (IV).

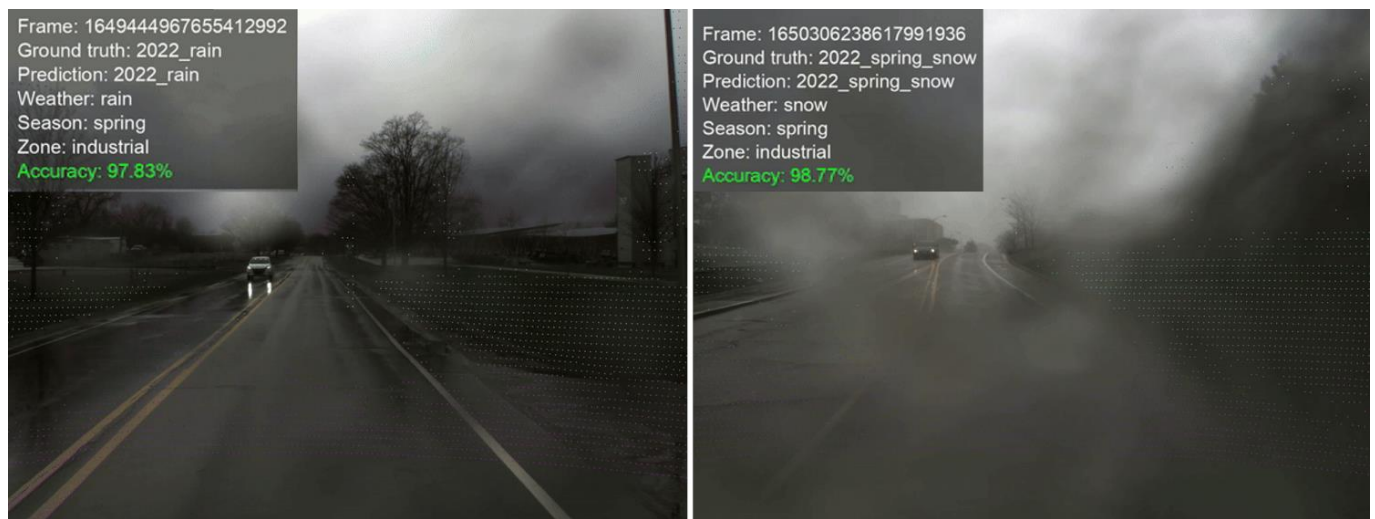


Figure 11 Sample qualitative results showing a spring rainy (left) and spring snowy (right) scenes predicted accurately.

1.4 Trajectory Prediction

In ROADVIEW, the main focus is not on implementing a new trajectory prediction method. Therefore, we have benchmarked different state-of-the-art trajectory prediction models on the Canadian Adverse Driving Conditions (CADC) dataset [31] logged in snowy weather conditions.

More specifically, we benchmark the robustness of 3D object-tracking algorithms under adverse weather conditions, such as snowy scenarios affecting autonomous vehicle perception systems. The principal objective is to evaluate and compare the performance of traditional and advanced AI-based object tracking methods: Kalman Filter, Extended Kalman Filter, Particle Filter, and ByteTrack. Each method was assessed using 3D LiDAR point cloud data obtained from the CADC dataset [31], representing harsh winter conditions, and the nuScenes dataset [22], representing clear, optimal weather conditions. We further anticipate the tracked object trajectories in the upcoming frames by employing an LSTM-based neural network.

The methodology involved processing detected 3D object bounding boxes in sequential LiDAR point clouds and tracking objects through association and state estimation. Standard metrics such as HOTA, IDF1, AMOTA, and AMOTP were used to measure tracking accuracy and consistency across both datasets. HOTA combines detection and association accuracies in a more balanced manner, while IDF1 indicates how long the tracker correctly identifies an object by computing the similarity between prediction and ground truth objects across the entire video, instead of simply counting errors. AMOTA measures the average accuracy of the tracking system, while AMOTP computes the precision of the object locations. Results in Table 5 indicated significant performance degradation for all algorithms under snowy weather conditions compared to clear weather. The Kalman filtering methods suffered from the linear behavior and the noise, while the Particle Filter provided a more robust estimation due to its ability to cope with the uncertainty via its multiple hypotheses. Deep learning-based ByteTrack demonstrated better performance, with higher accuracy and fewer identity switches in challenging scenarios. The study concluded that deep learning-based tracking methods provide a more solid guarantee of point trajectory and, thus, offer enhanced reliability for autonomous vehicles in challenging environments.

Table 5 Object tracking results on CADC (higher ↑ is better; lower ↓ is better).

Method	HOTA↑	IDF1↑	AMOTA↑	AMOTP↓	IDS↓
True	100	100	100	0.0	0
KF	43.2	63.2	55.8	83.7	490
PF	45.8	66.5	56.1	84.3	467
EKF	44.6	63.6	55.9	83.5	484
ByteTrack	53.0	70.0	60.0	86.2	209

Overall, as Table 5 shows, ByteTrack achieves the most robust tracking, excelling in detection and ID stability, while its slightly higher AMOTP reflects a trade-off in spatial precision. Particle Filter (PF) outperforms Kalman Filter (KF) and Extended Kalman Filter (EKF) across most metrics, thanks to its probabilistic tracking approach. KF and EKF perform comparably in snowy conditions but lag learning-based ByteTrack and sampling-based PF in maintaining identities and handling occlusions.

Figure 12 shows ByteTrack's predictions across different frames to illustrate tracking consistency and robustness over time.

Paper 03 [32], attached in the Appendix in Section 6, is entitled "Benchmarking Vehicle Trajectory Tracking and Prediction Methods in Adverse Weather Conditions" and provides a comprehensive description of our object tracking and trajectory prediction models. For more details about the implementation, we refer to the publicly available *source code* <https://github.com/Mahmoudalshaikh00/An-Experimental-Study-on-ObjectTracking.git>.

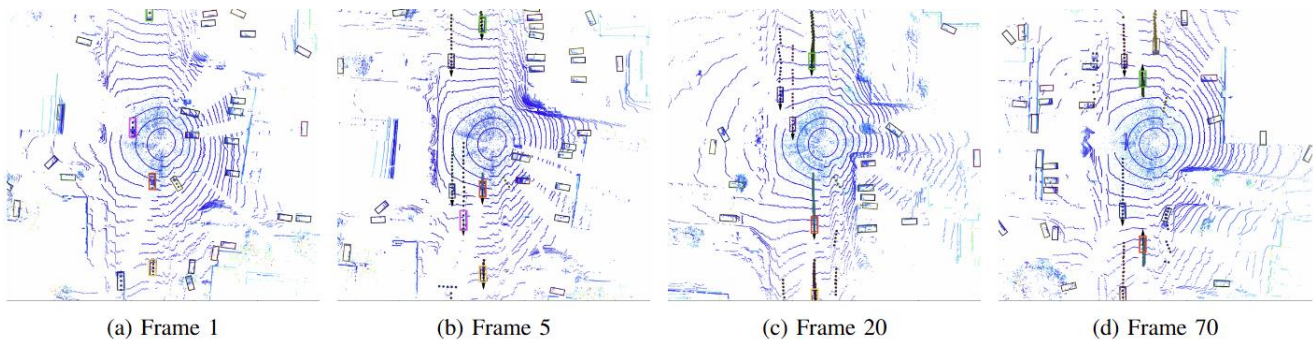


Figure 12 ByteTrack's tracking predictions at different time steps: Frame 1, 5, 20, and 70.

2 Functional Implementation

All perception modules (explained in Section 1) were functionally implemented using the well-known machine learning library PyTorch¹. For the training of these modules, a set of large-scale multimodal datasets (nuScenes [22], MSU-4S [29], and ROADVIEW Snowy Scenes [27]) was employed. Around M30, we started the integration of our core fusion model (see Section 1.1) into the demonstration vehicles in WP8. Therefore, during the functional implementation, we paid particular attention to the optimization of our solutions to reach close to the real-time processing capacity, which is around 10Hz. Note that the sampling rate of mainstream LiDAR scanners is also 10Hz. We, to a large extent, followed the ROS nodes and signals described in the previous deliverable D5.1.

3 Relation to other deliverables

The foundation for this deliverable originates from D2.4 *“Requirements of the Physical System Setup”*, which outlined the functional needs of the ROADVIEW system in detail. Each perception-related requirement specified there corresponds to a distinct functional module developed within the scope of this document. At the same time, the parallel deliverable D5.4 *“Software on Collaborative Perception Solutions – Final Report”* under WP5 presents a range of methods for enhancing perception through the exchange of data gathered from roadside infrastructure sensors. Another closely linked deliverable, D5.7 *“Software on Improved Localization Using High-Density Map Updating – Final Report”* (submitted in month 36 under WP5), builds upon LiDAR-based semantic segmentation approaches also featured in Section 1.2 in this deliverable. Furthermore, the outputs of this deliverable are expected to play a key role in supporting the system integration and demonstration tasks planned within WP8.

4 Conclusions and Continuous Improvement

This deliverable reports the final implementation results of the ROADVIEW perception solutions implemented in Task 5.1 in WP 5. These perception solutions integrate multiple sensor modalities—such as RGB cameras, 3D LiDARs, and RADARs—and incorporate capabilities for sensor fusion, object detection, free-space identification, weather classification, and trajectory prediction. More detailed technical explanations of selected methods can be found in three comprehensive scientific manuscripts included with this deliverable (refer to the Appendix in Section 6). In addition to the benchmarking using publicly available large-scale datasets, to accelerate the integration, some of these modules (e.g., sensor fusion for object detection described in Section 1.1) are further trained using custom ROADVIEW datasets logged by the demo owners (e.g., FORD and THI) in WP8.

¹ <https://pytorch.org/>

5 References

- [1] D. J. Yeong, G. Velasco-Hernandez, J. Barry and J. Walsh, "Sensor and sensor fusion technology in autonomous," *Sensors*, 2021.
- [2] K. Gadzicki, R. Khamsehashari and C. Zetsche, "Early vs late fusion in multimodal convolutional neural networks," in *IEEE 23rd international conference on information fusion (FUSION)*, 2020.
- [3] L. Yu, J. Zhang, Z. Liu, H. Yue and W. Chen, "Rethinking the Late Fusion of LiDAR-Camera Based 3D Object Detection," in *IEEE 19th Conference on Industrial Electronics and Applications (ICIEA)*, 2024 .
- [4] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus and S. Han, "BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [5] S. Vora, A. H. Lang, B. Helou and O. Beijbom, "PointPainting: Sequential Fusion for 3D Object Detection," in *IEEE Computer Vision and Pattern Recognition*, 2020.
- [6] T. Cortinhal, I. Gouigah and E. E. Aksoy, "Semantics-aware LiDAR-Only Pseudo Point Cloud Generation for 3D Object Detection," in *IEEE Intelligent Vehicles Symposium (In press)*, 2024.
- [7] N. Baumann, M. Baumgartner, E. Ghignone, J. Kuhne, T. Fischer, Y.-H. Yang, M. Pollefeys and M. Magno, "CR3DT: Camera-RADAR Fusion for 3D Detection and Tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.
- [8] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu and C.-L. Tai, "TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [9] Z. Yin, H. Sun, N. Liu, H. Zhou and J. Shen, "Decoratingfusion: A lidar-camera fusion network with the combination of point-level and feature-level fusion.," in *International Conference on Artificial Neural Networks*, 2024.
- [10] X. Chen, H. Ma, J. Wan, B. Li and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [11] S. Shi, X. Wang and H. Li, "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [12] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for Point cloud based 3d object detection," in *IEEE conference on computer vision and pattern recognition*, 2018.
- [13] A. Barrera, C. Guindel, J. Beltrán and F. García, "BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.
- [14] Y. Yan, Y. Mao and B. Li, "SECOND: Sparsely Embedded Convolutional Detection," *Sensors* , 2018.
- [15] R. Nabati and H. Qi, "Centerfusion: Center-based radar and camera fusion for 3d object detection," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.
- [16] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang and H. Li, "Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection," in *AAAI Conference on Artificial Intelligence*, 2021.
- [17] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang and O. Beijbom, "PointPillars: Fast Encoders for Object Detection From Point Clouds," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [18] J. Song, L. Zhao and a. S. K.A., "LiRaFusion: Deep Adaptive LiDAR-Radar Fusion for 3D Object Detection," in *IEEE ICRA*, 2024.
- [19] J. Hyeok Yoo, Y. Kim, J. Kim and J. Won Choi, "3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection," in *The European Conference on Computer Vision (ECCV)* , 2020.
- [20] J. Kim, J. Koh, Y. Kim, J. Choi, Y. Hwang and J. Won Choi, "Robust Deep Multi-modal Learning Based on Gated Information Fusion Network," in *Asian Conference on Computer Vision*, 2018.
- [21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [22] H. Caesar, V. Bankiti, A. . H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

- [23] A. Ahlman, A. Taylor, I. Gouigah and E. E. Aksoy, "LiDAR and RGB-D Fusion for 3D Object Detection on NVIDIA Jetson AGX Orin," in *IEEE Intelligent Vehicles Symposium (IV) (To be submitted)*, 2026.
- [24] T. Cortinhal, G. Tzelepis and E. E. Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *International Symposium on Visual Computing*, 2020.
- [25] W. Xiaoyang, J. Li, W. Peng-Shuai, L. Zhijian, L. Xihui, Q. Yu, O. Wanli, H. Tong and Z. Hengshuang, "Point transformer v3: Simpler faster stronger," in *IEEE/CVF CVPR*, 2024.
- [26] H. Zhou, X. Zhu, X. Song, Y. Ma, Z. Wang, H. Li and D. Lin, "Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation," in *arXiv preprint arXiv:2008.01550*, 2020.
- [27] I. Gouigah, A. M. Raisuddin, T.-T. Ngo, P. Litkey, P. Manninen, H. Hyyti, J. Maanpaa, H. Tuomas, V. Sofie, N. Kalle, L. Pernickel and E. E. Aksoy, "Snowy Scenes: A Multimodal Dataset Toward Snow-tonomous Vehicles," *IEEE RA-L (Under Review)*, 2025.
- [28] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *arXiv preprint*, 2020.
- [29] D. Kent, M. Alyaqoub, X. Lu, H. Khatounabadi, K. Sung, C. Scheller, A. Dalat, A. bin Thabit, R. Whitley and H. Radha, "MSU-4S - The Michigan State University Four Seasons Dataset," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024 .
- [30] N. A. Albashir, L. Pernickel, D. Hamoud, I. Gouigah and E. E. Aksoy, "LRC-WeatherNet: LiDAR, RADAR, and Camera Fusion for Real-time Weather-type Classification in Autonomous Driving," in *IEEE Intelligent Vehicles Symposium (IV) (To be submitted)*, 2026.
- [31] M. Pitropov, D. E. Garcia, J. Rebello, M. Smart, C. Wang, K. Czarnecki and S. Waslander, "Canadian adverse driving conditions dataset," *The International Journal of Robotics Research*, 2021.
- [32] M. Alshaikh, J. Ekstrand, G. Himesha Pathirage, A. M. Raisuddin, I. Gouigah and E. E. Aksoy, "Benchmarking Vehicle Trajectory Tracking and Prediction Methods in Adverse Weather Conditions," 2025.
- [33] T. Cortinhal and E. E. Aksoy, "Depth- and semantics-aware multi-modal domain translation: Generating 3D panoramic color images from LiDAR point clouds," *Robotics and Autonomous Systems*, 2024.
- [34] T. Cortinhal, F. Kurnaz and E. E. Aksoy, "Semantics-Aware Multi-Modal Domain Translation: From LiDAR Point Clouds to Panoramic Color Images," in *IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2021.
- [35] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [36] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- [37] M. Alibeigi, W. Ljungbergh, A. Tonderski, G. Hess, A. Lilja, C. Lindstrom, D. Motorniuk, J. Fu, J. Widahl and C. Petersson, "Zenseact Open Dataset: A large-scale and diverse multimodal dataset for autonomous driving," in *IEEE/CVF International Conference on Computer Vision*, 2023.
- [38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, 2021.
- [39] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus and a. H. S., "Bevfusion: Multi-task multi-sensor fusion with unified bird's eye view representation," in *IEEE international conference on robotics and automation (ICRA)*., 2023.
- [40] D. Ye, Z. Zhou, W. Chen, Y. Xie, Y. Wang, P. Wang and a. F. H., "Lidarmultinet: Towards a unified multi-task network for lidar perception," in *AAAI Conference on Artificial Intelligence*, 2023.
- [41] T. Cortinhal, I. Gouigah and E. E. Aksoy, "Semantics-aware LiDAR-Only Pseudo Point Cloud Generation for 3D Object Detection," in *IEEE Intelligent Vehicles Symposium (IV)*, 2024.
- [42] Y. Chen, J. Liu, X. Zhang, X. Qi and J. J., "Voxelnext: Fully sparse voxelnet for 3d object detection and tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [43] Z. Wang, Y.-L. Li, X. Chen, H. Zhao and S. Wang, "Uni3detr: Unified 3d detection transformer," in *Advances in Neural Information Processing Systems*, 2024.
- [44] A. M. Raisuddin, T. Cortinhal, J. Holmblad and E. E. Aksoy, "3D-OutDet: A Fast and Memory Efficient Outlier Detector for 3D LiDAR Point Clouds in Adverse Weather," in *IEEE Intelligent Vehicles Symposium (IV)*, 2024 .

6 Appendix

PAPER 01: LiDAR and RGB-D Fusion for 3D Object Detection on NVIDIA Jetson AGX Orin

Adam Taylor^{1,*}, Adam Ahlman^{1,*}, Idriss Gouigah¹, and Eren Erdal Aksoy¹

Abstract—Sensor fusion is an essential component of autonomous robotic systems. It can be used to improve the accuracy of various tasks, such as perception and localization. However, advanced perception models, such as 3D object detection, have high computational demands, making sensor fusion challenging to implement in embedded environments. Since robots operate under strict energy and hardware constraints, the size of perception models is crucial, underscoring the need for compact, efficient computing hardware.

This study explores the applicability of early, mid, and late fusion of 3D LiDAR and RGB-D sensor readings on the NVIDIA Jetson AGX Orin embedded platform in an indoor environment. The fusion techniques were compared to baselines and evaluated using a custom-annotated dataset consisting of indoor objects, such as chairs, tables, and bookshelves. The baseline models were implemented based on a single modality, such as camera-only or LiDAR-only. Early, mid, and late fusion techniques were evaluated, and various hybrid models were developed to leverage the strengths of each technique. Performance was assessed using standard evaluation metrics, including mean Average Precision (mAP) and inference time. The fusion models outperformed the baselines in terms of accuracy. The combined model achieved the highest overall accuracy, and early fusion had the lowest inference time. The source code, trained models, and the online supplementary information are available at the following URL: https://github.com/Adam20Taylor/AGX_Orin-Fusion.git.

I. INTRODUCTION

The fusion of data from multiple sensor modalities, such as LiDAR and RGB-D cameras, enables better environmental perception in robotics. While recent deep learning models have significantly improved 3D object detection performance, they often rely on computationally expensive architectures that are ill-suited for deployment on embedded, power-constrained platforms like mobile robots and autonomous vehicles. Existing benchmarks tend to focus on outdoor environments, overlooking indoor or short-range scenarios where RGB-D sensors are effective.

This work focuses on comparing and evaluating early, mid, and late fusion models for 3D object detection using 3D LiDAR and RGB-D sensor modalities. We deploy our models on the NVIDIA Jetson AGX Orin, a compact edge computing platform, using real-world data from a mobile robot operating in indoor environments (see Fig. 1). We

*Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them. Project grant no. 101069576.

¹Halmstad University, School of Information Technology, Center for Applied Intelligent Systems Research, Halmstad, Sweden eren.aksoy@hh.se

*These authors contributed equally!

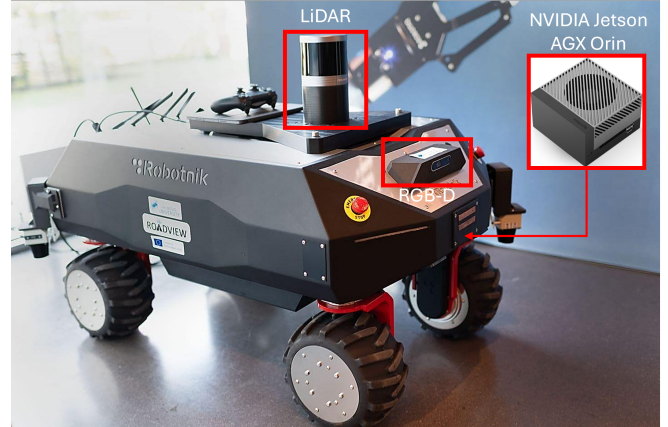


Fig. 1: Our mobile robot platform with onboard 3D LiDAR and RGB-D sensors.

aim to answer whether real-time sensor fusion is feasible using current state-of-the-art embedded hardware and which fusion strategy (early, mid, late, or hybrid) delivers optimal performance in this setting.

RGB-D cameras combine color and depth data, enabling more enriched scene understanding than traditional 2D images alone. However, they are limited by short depth range and sensitivity to lighting conditions. Several deep learning approaches have leveraged this modality for 3D object detection. Frustum PointNets [1] project 2D detections into 3D space to reduce search complexity, while PointFusion [2] fuses RGB and depth features at the point level. TR3D [3], the most relevant to our work, introduces a fully convolutional, multi-modal architecture that achieves strong performance through early fusion of RGB-D features.

LiDAR sensors provide high-resolution 3D point clouds, offering reliable spatial data across various lighting conditions. VoxelNet [4] and PointPillars [5] quantize point clouds into voxels or pillars for CNN-based operations, while RangeDet [6] transforms LiDAR data into range images. VoteNet [7], in contrast, directly processes raw point sets, using a Hough voting mechanism to predict object centers. While LiDAR provides high depth accuracy [8], [9], it lacks semantic information like color, making it difficult to distinguish between objects with similar geometric shapes.

Sensor fusion methods are commonly categorized into early, mid, and late fusion. Early fusion combines raw data from multiple modalities before feature extraction [10], mid fusion integrates learned features from individual sensors [11], and late fusion merges decisions or final predictions

from separate models [12]. Each approach has trade-offs in terms of complexity, efficiency, and performance.

PointPainting [13] exemplifies early fusion by projecting semantic segmentation maps from 2D images onto LiDAR points. An early fusion method for multi-view visual tracking is presented by EarlyBird [14], which projects image features from several cameras into a common bird’s-eye-view (BEV) space and fuses them via channel-wise concatenation. Aug-VirConv [15] enriches sparse LiDAR point clouds with early fused additional pseudo points generated from LiDAR scene semantics to enrich 3D object detection. While effective in enriching LiDAR-based detection, such methods can be sensitive to projection errors and may increase input dimensionality, affecting real-time performance on embedded platforms. Mid-level fusion combines different sensor features in a shared latent space, leveraging each modality’s unique descriptors while balancing efficiency and performance. BEVFusion [16] projects sensor features into a shared BEV representation, enabling spatial alignment. Adaptive gated networks for mid level fusion [11], [17], [18] selectively merge features based on learned attention, demonstrating state-of-the-art results across sensor combinations. However, most research has focused on LiDAR-camera or LiDAR-Radar fusion, with limited work exploring RGB-D and LiDAR fusion in real-time, embedded scenarios. Late fusion methods, such as confidence voting or majority voting, merge independent predictions from each sensor stream [19]. These techniques are computationally simple and modular but fail to exploit feature-level complementarities between modalities, potentially limiting accuracy in ambiguous scenarios.

Benchmarking of multimodal deep neural networks on embedded platforms is underrepresented compared to the vast amount of sensor fusion research. In a recent study [20], the authors benchmarked multiple LiDAR Semantic Segmentation models on the NVIDIA RTX 4090, Jetson Xavier, and the Jetson AGX Orin. The study included four types of 3D semantic segmentation models: sparse Conv-based, projection-based, fusion-based, and point-based models. However, the study was conducted in a unimodal setting, relying exclusively on LiDAR data without incorporating information from any other sensors. This study [20] revealed that the projection-based unimodal SalsaNext [21] was able to achieve real-time inference of 20 FPS on the Jetson AGX Orin while still performing above 0.68 validation mIoU.

The work in [22] performed a comprehensive benchmark of accuracy, inference time, resource usage, and energy consumption of multiple 3D object detection models running on all commercially available Jetson devices. This study [22] also investigated the impact of the TensorRT optimization library on the models. The authors found that the TensorRT library increased the FPS of the models by four times and cut the CPU usage by half. With the use of TensorRT, the Orin device achieved an average of 18.4 FPS, further showcasing the embedded platform’s performance.

Our study differs from all these recent works by particularly focusing on the fusion of 3D LiDAR and RGB-D

sensor modalities for the 3D object detection task on the NVIDIA Jetson AGX Orin device. More specifically, we compare the performance of different fusion approaches in indoor scenarios.

II. EXPERIMENTAL SETUP

A. Equipment

The experiments and research were performed using a state-of-the-art embedded platform, NVIDIA Jetson Orin AGX. The platform specs are detailed in Table I, where the power is set to maximum for all experiments. The system runs on NVIDIA Jetpack 5.12 operating system and is based on Arm®Cortex®-A78AE. The sensor data is generated from sensors mounted on a robotic platform Robotnik RB-VOGUE, as shown in Fig. 1. The 3D LiDAR sensor (3D Laser Robosense RS-HELIOS-1615) is mounted on the topside of the robot, and the RGB-D camera (Intel Realsense Camera D435f) sits in the front.

B. Dataset

To facilitate the model evaluation, we collected and annotated a custom dataset. The dataset was generated using Rosbags, which recorded data from ROS topics corresponding to the camera feed, LiDAR, and RGB-D point clouds. Following data collection, the LiDAR point clouds were cropped to retain only the points within the camera’s field of view. A sample of three frames of point clouds and their corresponding camera images are shown in Fig. 2. Some of the frames also include decoy objects that should not be classified by the detection models; one of these frames is showcased in Fig. 3.

Our custom dataset comprises 71 unique indoor scenes densely populated with common indoor household objects such as chairs, bookshelves, and tables. This diversity and object density provide a suitable testbed for evaluating the performance of 3D object detection models with and without sensor fusion. Fig. 4 illustrates a representative LiDAR point cloud frame from the dataset, with ground truth bounding boxes overlaid. For reference, the same scene is shown in Fig. 2, where objects such as a chair, table, and bookshelf are annotated in blue, green, and yellow, respectively.

AI performance	275 TOPS
GPU	2048-core NVIDIA Ampere c GPU with 64 Tensor Cores
GPU Max Frequency	1.3 GHz
CPU	12-core Arm®Cortex®-A78AE v8.2 64-bit CPU
CPU Max Frequency	2.2 GHz
Memory	32GB 256-bit LPDDR5 204.8GB/s
Storage	64GB eMMC 5.1
Power	15W - 60W
Dimensions	110 x 110 x 71.65 mm

TABLE I: Specifications of the NVIDIA Jetson AGX Orin developer kit

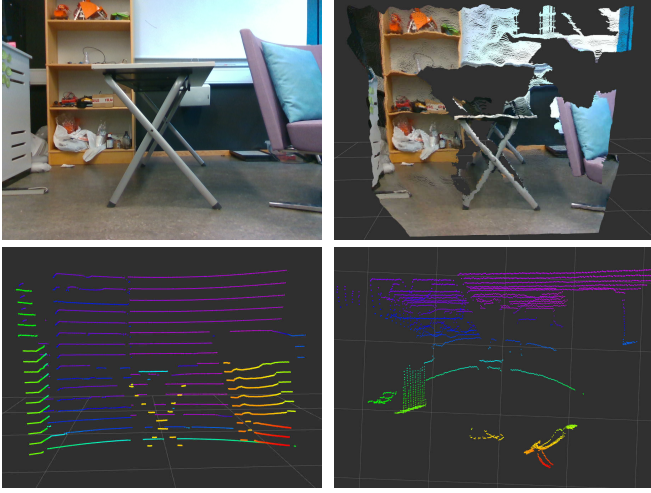


Fig. 2: Sample frame showcasing the point clouds. The top-left image shows the 2D RGB image, the top right shows the RGB-D point cloud, and the bottom images show the LiDAR point cloud in front and top-down views.

III. METHOD

Fig. 5 illustrates our complete architecture used for evaluating different sensor fusion strategies for 3D object detection. RGB-D and LiDAR point clouds are processed either independently or in combination through various fusion stages. TR3D [3] and Votenet [7] serve as detection heads for RGB-D and LiDAR streams, respectively. Light orange boxes represent fusion operations: Early Fusion combines point clouds before detection, while Late Fusion merges predictions post-detection. Our mid-level fusion integrates features individually extracted in each modality branch. Each output is evaluated independently as Camera-only, LiDAR-only, Early Fusion, Mid Fusion, Late Fusion, and hybrid

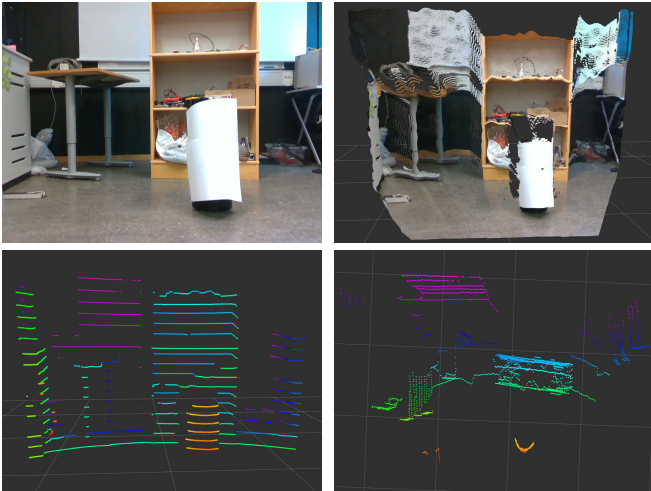


Fig. 3: Sample frame that includes a decoy object, the black cylinder with a white paper attached. The images show the 2D RGB image, the RGB-D point cloud, and the LiDAR point cloud in front and top-down views.

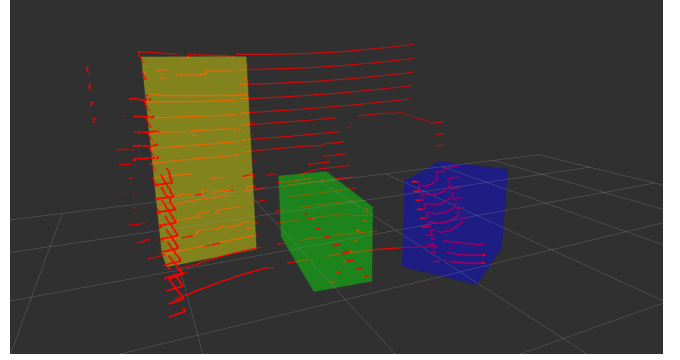


Fig. 4: Bounding boxes and the cropped LiDAR points of a frame in the dataset. The yellow bounding box corresponds to a bookshelf, the green to a table, and the blue to a chair.

fusion (e.g., Early + Late Fusion), as highlighted by the green boxes in Fig. 5. The details of each proposed fusion model and trained detection model are provided below.

1) *Detection heads*: The implementation of the detection heads is based on the MMDetection3D framework [23]. We rely on the object detection models pretrained on multiple indoor 3D object detection datasets, such as ScanNet [24].

The relevant classes that the detection heads can predict are chairs, bookshelves, and tables. The choice of models was based on different factors for each head. An initial study was conducted to evaluate which of the pre-trained models available in MMDetection3D achieved the best performance on our custom dataset, in terms of accuracy and frames per second (FPS). For RGB-D-based 3D object detection, the TR3D model [3] was selected due to its high accuracy and its focus on real-time performance. Additionally, TR3D incorporates semantic information for each point in the point cloud. Votenet [7] was chosen as the object detection model for the LiDAR modality, as it performed the best out of all MMDetection3D’s models that rely solely on spatial information. Additionally, it also uses a point-based anchor-free approach, making it well-suited for handling sparse point clouds.

2) *Early fusion through kNN*: To enrich the LiDAR point cloud with color information from the RGB-D camera, we apply k-nearest neighbors (kNN) to link each LiDAR point with its nearest neighbors in the RGB-D point cloud. In

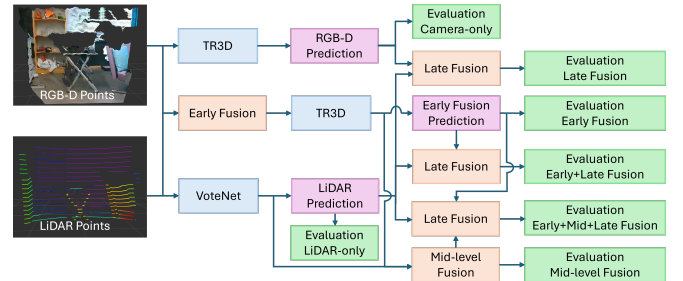


Fig. 5: Overarching architecture of the currently implemented fusion models and detection heads.

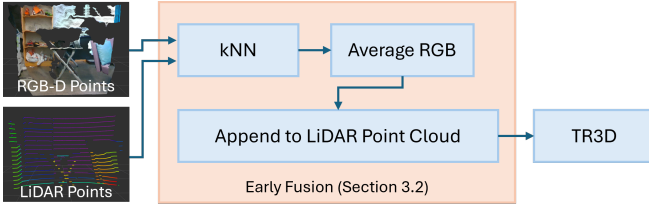


Fig. 6: Flowchart of the early fusion.

other words, for each LiDAR point, the algorithm finds the k closest points in the RGB-D depth map and computes the average RGB value of these neighbors. The averaged color is assigned to the LiDAR point, effectively coloring the point cloud with visual features. This decorated point cloud is then fed to a single detection head, which considers both spatial and color information. To visualize the early fusion algorithm, a flowchart is shown in Fig. 6.

3) *Mid fusion through gated networks*: As illustrated in the mid-level fusion flowchart in Fig. 7, once the RGB-D and LiDAR streams are processed by the corresponding backbones, extracted low-dimensional feature vectors are first matched to have the same size. Since the TR3D feature vectors vary in size, the coordinate vectors, produced alongside the features, were used to adjust the size of the vote features produced by VoteNet. This feature vector size matching was achieved by iteratively mapping each TR3D feature point to its closest vote feature until the sizes were identical. Next, building on the approaches in [11], [17], [18], the matched and concatenated feature vectors are processed through an adaptive gated network. This network dynamically calibrates the contribution of each modality to the final fused representation. Rather than relying on simple feature concatenation, this gated fusion mechanism adaptively learns spatially and channel-wise weights during training. These weights are derived from the joint feature representations produced by the backbone modules, allowing the model to evaluate the relative importance of RGB-D and LiDAR features. By adaptively reweighting modalities, the gated network can emphasize the most informative inputs while reducing the impact of less relevant or noisier signals.

4) *Late fusion through confidence voting*: To perform late fusion, RGB-D and LiDAR-based detection heads are executed independently, and their predictions are subsequently combined. The 3D object detection models from MMDetection3D produce probability scores for each predicted bounding box [23], enabling the use of a confidence voting algorithm. In this approach, all predicted bounding

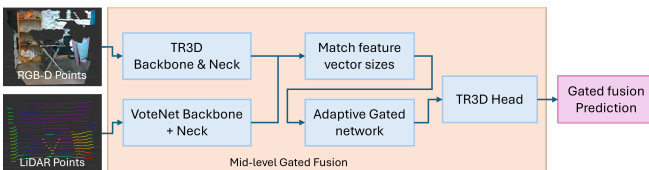


Fig. 7: Flowchart of the mid fusion with the gating module.

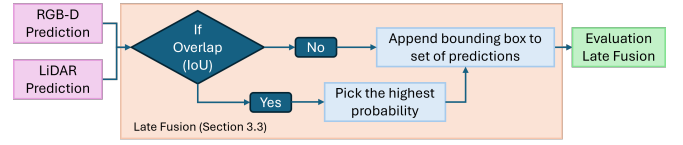


Fig. 8: Flowchart of the late fusion with confidence voting.

boxes are added to the scene. When overlapping boxes from different modalities are detected, the one with the lowest confidence score is removed. Fig. 8 illustrates the confidence voting process for each pair of overlapping bounding boxes.

5) *Hybrid fusion*: We benchmark different combination of early, mid, and late fusion algorithms as previously described. A detailed visualization of the final hybrid fusion architecture, which integrates all three fusion strategies, is provided in Fig. 9. Compared to the camera-only baseline, the early fusion approach generates a sparser yet more accurate point cloud by combining RGB-D and LiDAR data. Therefore, in our hybrid fusion architecture in Fig. 9, the raw LiDAR data, together with the early fused RGB-D and LiDAR data are fed into the mid-level gated fusion module. Finally, to complete the fusion process, a late fusion stage combines the predictions from the mid-level gated fusion with those from the LiDAR-only branch (see Fig. 9).

IV. RESULTS

This section presents and analyzes the performance of various unimodal and multimodal object detection models on the NVIDIA Jetson AGX Orin platform. As quantitatively reported in Table II, we compare LiDAR-only, RGB-D camera-only, early fusion, mid fusion, late fusion, and a set of different hybrid fusion approaches in terms of mean Average Precision (mAP) and inference speed measured in Frames Per Second (FPS).

A. 3D Object Detection Accuracy

As shown in Table II, unlike early and mid fusion, late fusion outperforms all unimodal (single-modality) baselines in detection accuracy. In particular, early fusion yields a higher mAP than the camera-only model while achieving performance on par with the LiDAR-only baseline. Mid fusion on its own does not show a performance gain. As

TABLE II: Comparison of the mAPs for IoU thresholds of 25% and 50%, including the FPS of each model.

	mAP		FPS
	@0.5	@0.25	
LiDAR-only	0.210	0.359	7.52
Camera-only	0.076	0.193	6.13
Early Fusion	0.192	0.192	7.19
Mid Fusion	0.061	0.121	7.35
Early + Mid Fusion	0.152	0.152	8.40
Concatenation Fusion	0.030	0.061	7.87
Early + Concatenation Fusion	0.091	0.091	8.70
Late Fusion	0.215	0.379	3.06
Early + Late Fusion	0.253	0.397	3.86
Early + Mid + Late Fusion	0.260	0.403	4.07

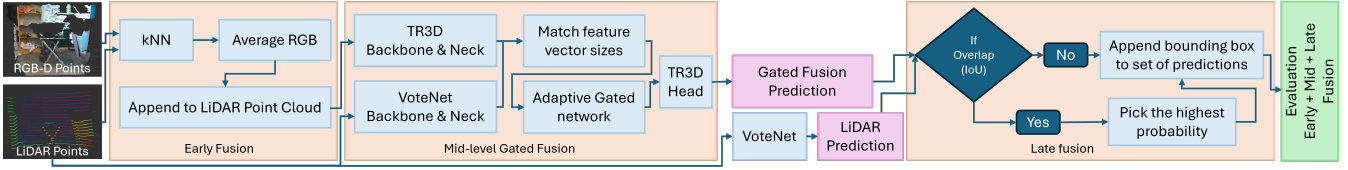


Fig. 9: Flowchart the hybrid fusion architecture, combining the early, mid, and late fusion strategies.

an additional baseline, we evaluated a simple feature concatenation strategy for mid fusion, which similarly failed to achieve improvements over gated network-based mid fusion. Our proposed hybrid model (see Fig. 9) which integrates early, mid, and late fusion strategies, attains the highest mAP among all evaluated methods, demonstrating that a combination of low-, mid-, and decision-level fusions is particularly effective when the individual modality predictions are already strong.

B. Inference Performance

In terms of real-time feasibility, early fusion with simple feature concatenation achieves the highest FPS, albeit with relatively low accuracy. Despite requiring feature-level integration of two modalities, early fusion still surpasses the camera-only model in both mAP and FPS, suggesting that low-level fusion has minimal impact on overall inference speed on the embedded platform. In contrast, late fusion (and any fusion scheme involving it) substantially reduces FPS, primarily due to the need to run independent detection models for each modality before merging results. These findings highlight the trade-off between model complexity and deployability in real-time robotic applications. Considering both accuracy and latency, our hybrid fusion model (last row in Table II) maintains a relatively high FPS (4.07) compared to pure late fusion or combinations of early and late fusion.

C. Class-wise Accuracy

The performance of the fusion models is further showcased in Tables III and IV, where each class is compared based on its Average Precision (AP). The models struggle to detect tables in the majority of scenes, as reflected by the low AP scores. In contrast, the fusion models demonstrate a significant improvement in accuracy for the bookshelf class, while only a small gain is observed for the chair class when

TABLE III: Comparison of the average precision for each class at an IoU threshold of 25%.

Modalities	AP@0.25		
	Chair	Table	Bookshelf
LiDAR-only	0.709	0.035	0.333
Camera-only	0.515	0.019	0.045
Early Fusion	0.455	0	0.121
Mid Fusion	0.364	0	0
Early + Mid Fusion	0.455	0	0
Concatenation Fusion	0.182	0	0
Early + Concatenation Fusion	0.273	0	0
Late Fusion	0.709	0.058	0.37
Early + Late Fusion	0.727	0.05	0.415
Early + Mid + Late Fusion	0.788	0.038	0.382

TABLE IV: Comparison of the average precision for each class at an IoU threshold of 50%.

Modalities	AP@0.5		
	Chair	Table	Bookshelf
LiDAR-only	0.455	0.004	0.170
Camera-only	0.182	0	0.045
Early Fusion	0.455	0	0.121
Mid Fusion	0.182	0	0
Early + Mid Fusion	0.455	0	0
Concatenation Fusion	0.091	0	0
Early + Concatenation Fusion	0.273	0	0
Late Fusion	0.461	0.012	0.173
Early + Late Fusion	0.503	0.008	0.248
Early + Mid + Late Fusion	0.576	0.019	0.185

fusion is used. Additionally, the early fusion results at 50% IoU threshold show a substantial improvement compared to the camera-only model. Additionally, the early fusion has the same AP score for both the 25% and 50% IoU thresholds, which indicates that each of the TR3D's predictions is highly accurate.

D. Qualitative Results

To further showcase the performance of the hybrid fusion model, two test scenes are shown with the predictions and corresponding ground truths. These scenes can be seen in Fig. 10, where the blue bounding boxes represent the chair class and the yellow bounding boxes represent bookshelves. The lighter shades indicate predictions, while the darker shades of both colors represent ground truths.

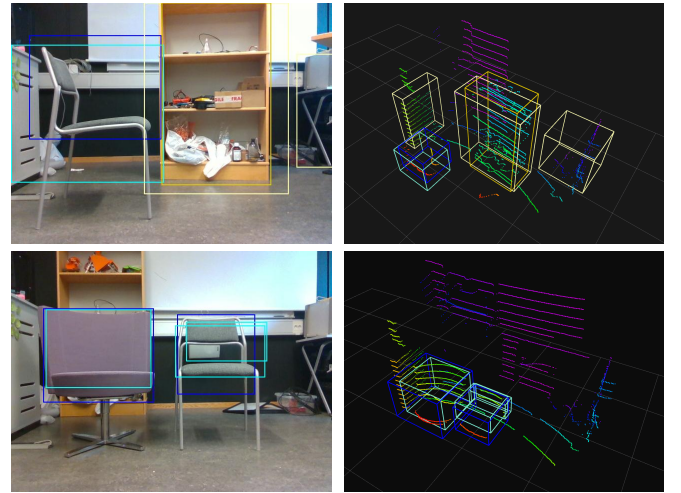


Fig. 10: Qualitative results of two frames showcasing predictions and ground truths.

V. DISCUSSION & LIMITATIONS

The results showcase that our hybrid sensor fusion framework illustrated in Fig. 9 can effectively increase the performance of 3D object detection with limited real-time capability (4.07 FPS). The improved AP performance can be credited to the accuracy of the LiDAR measurements rather than the sparsity of the point cloud, especially since the RGB-D model was originally designed to operate on dense point clouds. However, the reduced point cloud density does contribute to faster inference times.

Early fusion enriched with a simple feature concatenation achieves the highest FPS with considerably low accuracy. Late fusion showed improved accuracy overall, but at the cost of almost halving the FPS. Lastly, the small fluctuations in accuracy between the LiDAR-only model and the late fusion model are most likely due to the non-deterministic nature of the LiDAR detection model.

Crucially, none of the models showed real-time performance when deployed on the Orin device. However, it is important to note that none of these models utilized optimization libraries such as TensorRT to speed up their inference time. This limitation is significant as these libraries have been shown to have the potential for an increase in FPS by four times on the same device as reported in [22].

REFERENCES

- [1] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.
- [2] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 244–253.
- [3] D. Rukhovich, A. Vorontsova, and A. Konushin, "Tr3d: Towards real-time indoor 3d object detection," in *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2023, pp. 281–285.
- [4] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.
- [5] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.
- [6] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, "Rangedet: In defense of range view for lidar-based 3d object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 2918–2927.
- [7] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.
- [8] N. Li, C. P. Ho, J. Xue, L. W. Lim, G. Chen, Y. H. Fu, and L. Y. T. Lee, "A progress review on solid-state lidar and nanophotonics-based lidar sensors," *Laser & Photonics Reviews*, vol. 16, no. 11, p. 2100511, 2022.
- [9] H. Zhong, H. Wang, Z. Wu, C. Zhang, Y. Zheng, and T. Tang, "A survey of lidar and camera fusion enhancement," *Procedia Computer Science*, vol. 183, pp. 579–588, 2021.
- [10] K. Gadzicki, R. Khamsehashari, and C. Zetsche, "Early vs late fusion in multimodal convolutional neural networks," in *2020 IEEE 23rd international conference on information fusion (FUSION)*. IEEE, 2020, pp. 1–6.
- [11] J. Song, L. Zhao, and K. A. Skinner, "Lirafusion: Deep adaptive lidar-radar fusion for 3d object detection," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 18 250–18 257.
- [12] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, 2021.
- [13] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4604–4612.
- [14] T. Teepe, P. Wolters, J. Gilg, F. Herzog, and G. Rigoll, "Earlybird: Early-fusion for multi-view tracking in the bird's eye view," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. IEEE, 2024, pp. 102–111.
- [15] T. Cortinhal, I. Gouigah, and E. E. Aksoy, "Semantics-aware lidar-only pseudo point cloud generation for 3d object detection," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, 2024, pp. 3220–3226.
- [16] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *2023 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 2774–2781.
- [17] J. H. Yoo, Y. Kim, J. Kim, and J. W. Choi, *3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-view Spatial Feature Fusion for 3D Object Detection*. Springer International Publishing, 2020, p. 720–736.
- [18] J. Kim, J. Koh, Y. Kim, J. Choi, Y. Hwang, and J. W. Choi, "Robust deep multi-modal learning based on gated information fusion network," in *Computer Vision – ACCV 2018*, 2018, p. 90–106.
- [19] T.-Y. Lim, A. Ansari, B. Major, D. Fontijne, M. Hamilton, R. Gowaikar, and S. Subramanian, "Radar and camera early fusion for vehicle detection in advanced driver assistance systems," in *Machine learning for autonomous driving workshop at the 33rd conference on neural information processing systems*, vol. 2, no. 7, 2019.
- [20] S. Abou Haidar, A. Chariot, M. Darouich, C. Joly, and J.-E. Deschaud, "Are we ready for real-time lidar semantic segmentation in autonomous driving?" in *Workshop on Planning, Perception, and Navigation for Intelligent Vehicles, IROS*, 2024.
- [21] T. Cortinhal, G. Tzelepis, and E. Erdal Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *International Symposium on Visual Computing*. Springer, 2020, pp. 207–222.
- [22] C. Choe, M. Choe, and S. Jung, "Run your 3d object detector on nvidia jetson platforms: a benchmark analysis," *Sensors*, vol. 23, no. 8, p. 4005, 2023.
- [23] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection," <https://github.com/open-mmlab/mmdetection3d>, 2020.
- [24] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

PAPER 02: LRC-WeatherNet: LiDAR, RADAR, and Camera Fusion for Real-time Weather-type Classification in Autonomous Driving

Nour Alhuda Albashir^{1,*}, Lars Pernickel^{1,*}, Danial Hamoud^{1,*}, Idriss Gouigah¹, and Eren Erdal Aksoy¹

Abstract—Autonomous vehicles face major perception and navigation challenges in adverse weather such as rain, fog, and snow, which degrade the performance of LiDAR, RADAR, and RGB camera sensors. While each sensor type offers unique strengths—such as RADAR’s robustness in poor visibility and LiDAR’s precision in clear conditions—they also suffer distinct limitations when exposed to environmental obstructions. This study proposes LRC-WeatherNet, a novel multi-sensor fusion framework that integrates LiDAR, RADAR, and camera data for real-time weather condition classification. By employing both early fusion using a unified Bird’s Eye View representation and mid-level gated fusion of modality-specific feature maps, our approach adapts to the varying reliability of each sensor under changing weather. Evaluated on the extensive MSU-4S dataset covering nine weather types, LRC-WeatherNet achieves superior classification performance and computational efficiency, significantly outperforming unimodal baselines in adverse conditions. This work is the first to combine all three modalities for robust, real-time weather classification in autonomous driving.

I. INTRODUCTION

Autonomous vehicles (AVs) encounter substantial challenges in perception and navigation during adverse weather conditions such as rain, fog, and snow. These environmental factors significantly degrade the performance of key sensors, namely LiDAR, RADAR, and RGB camera, thus compromising the accuracy and reliability of downstream perception tasks such as scene classification and object detection.

In recent years, there has been growing interest in employing a diverse set of these sensor modalities for more reliable scene perception. Each sensor type, however, offers distinct advantages and limitations as illustrated in Fig. 1. RADAR technology, for instance, maintains robust functionality in challenging weather such as heavy rain or fog, due to its reliance on long-wavelength electromagnetic signals that more effectively penetrate atmospheric obstructions [1]. While this resilience ensures stable operation, RADARs occasionally misinterpret reflections from nearby raindrops as false signals, introducing inaccuracies at close ranges. LiDARs, on the other hand, provide precise 3D perception under clear skies. Their effectiveness, however, diminishes significantly in dense fog or precipitation due to light scattering, particularly at longer detection ranges. Cameras excel

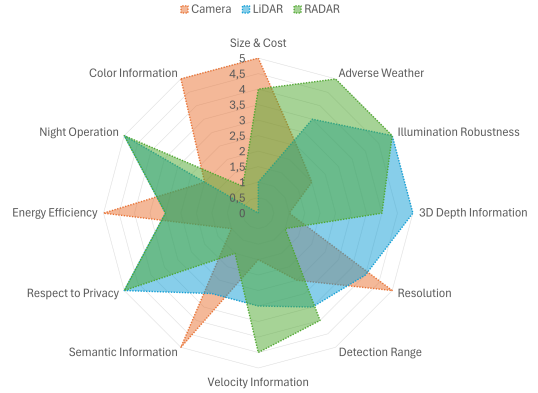


Fig. 1: Illustration of the characteristics of LiDAR, RADAR, and Camera sensors. Contextually, LiDAR-RADAR-Camera fusion has the highest complementary properties.

at capturing rich visual cues, including color and contextual details critical for interpreting scene semantics, yet their utility plummets in suboptimal weather conditions. Fusing readings from these complementary sensors and exploiting their unique information patterns reduces the shortcomings of each sensor, offering enhanced performance in challenging weather and seasonal conditions [2].

In this study, we address the challenge of robust weather condition classification for autonomous vehicles through a multi-sensor fusion framework. Leveraging the complementary sensing capabilities of LiDAR, RADAR, and camera, we propose a new model, named LRC-WeatherNet, integrating readings from all three modalities in real time to improve the accuracy of weather condition recognition. Our fusion pipeline explores both early fusion, where LiDAR and RADAR features are projected into a unified Bird’s Eye View (BEV) representation, and mid-level gated fusion, which combines modality-specific feature maps after backbone processing. Thus, our fusion strategy learns inherent strengths and weaknesses of each modality under different environmental conditions. To assess model performance and sensor contributions, experiments with different backbone architectures (e.g., EfficientNet-B0 [3] and PointPillars [4]) are conducted.

We evaluate the performance of LRC-WeatherNet on the Michigan State University Four Seasons (MSU-4S) dataset [5], which captures a wide range of seasonal and environmental conditions. The classification task includes nine distinct weather conditions, such as *rain*, *spring snow*, *snow*, *fall*, *sunset*, *late summer*, and *early fall* scenarios. We report

*Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them. Project grant no. 101069576.

¹Halmstad University, School of Information Technology, Center for Applied Intelligent Systems Research, Halmstad, Sweden eren.aksoy@hh.se

*These authors contributed equally!

a quantitative comparison of our model performance and computational efficiency using metrics such as accuracy, F1-score, confusion matrices, inference time, and the number of operations (measured in GMACs). The obtained results show our lightweight BEV-based multi-modal fusion strategy significantly improves classification performance over different baselines, particularly under challenging weather conditions.

To the best of our knowledge, this is the first work to fuse LiDAR, RADAR, and camera data for real-time weather classification in autonomous driving.

II. RELATED WORK

A variety of fusion strategies have been developed in the literature. Early fusion, as in [6], integrates the raw sensor readings prior to processing for the downstream perception task. Early Fusion has the key advantage of not leaking any sensor readings because it relies exclusively on raw sensor data. Late fusion, as in [7], stands out as an alternative information aggregation method to merge high-level predictions derived individually from each modality. Contemporary research explores mid-level fusion strategies to capture descriptive scene features unique to each modality. Along these lines, the LiRa fusion in [8] leverages LiDAR’s dense spatial features and further incorporates RADAR’s attributes to improve 3D object feature representation. LiRa begins with the early fusion of LiDAR and RADAR voxel features extracted from intensity, velocity, and RADAR Cross Section (RCS) data, which are jointly encoded using sparse convolutions. Next, in the middle fusion stage, LiRa introduces adaptive gated networks to assign channel-wise weights to the fused feature maps, enabling selective emphasis on sensor contributions based on contextual needs. Along the same lines, the work in [9] introduces a deep gated fusion approach to combine camera and LiDAR data for object detection. These architectures demonstrate the potential of adaptive gated fusion networks to dynamically balance sensor contributions. Our study extends this methodology by incorporating cameras with LiDAR and RADAR for weather classification in the 2D Bird’s Eye View (BEV) space.

DeepFusion in [10] begins by extracting high-level features from each sensor: LiDAR data is voxelized and converted into a BEV, while camera data is processed through CNNs to efficiently extract visual features for 3D object detection. These features are then fused and aligned at an advanced feature level, ensuring their complementarity. A key enhancement of DeepFusion is its learnable alignment layer, which adaptively aligns camera features with LiDAR voxels based on a cross-attention mechanism.

EarlyBird in [6] proposes an early fusion framework for multi-view visual tracking, where image features from multiple cameras are projected into a shared BEV space and fused through a simple channel-wise concatenation prior to downstream processing. The architecture avoids explicit cross-attention or modality-specific alignment mechanisms, demonstrating that early spatial fusion can yield competitive performance while remaining computationally efficient.

Inspired by this paradigm, we extend EarlyBird’s early fusion concept from multi-view visual inputs to multi-modal sensory inputs, more specifically LiDAR and RADAR. Unlike LiRa [8], in our approach, LiDAR and RADAR point clouds are independently projected into a common BEV space, and their modality-specific attributes (i.e., LiDAR intensity, RADAR SNR, and RADAR RCS) are stacked into a unified three-channel BEV representation. This design preserves spatial alignment across modalities while allowing the downstream convolutional backbone (EfficientNet [3]) to learn discriminative features from complementary sensory inputs. Following the work in [8], we further introduce a gating mechanism to adaptively learn the early fused LiDAR and RADAR features with those from the RGB camera sensor. By leveraging this architectural simplicity in the context of multi-modal data, we aim to balance representational richness with inference efficiency for weather-type classification tasks.

In the context of the classification of diverse weather conditions, RECNet was introduced in [11] as a camera-only model for environmental condition classification. Likewise, a lightweight network, RangeWeatherNet, was proposed in [12] to classify both weather types and road conditions by using LiDAR-only data. RangeWeatherNet processes LiDAR point clouds using a range image projection, converting 3D data into a 2D representation that is further processed through a modified DarkNet architecture [13]. Although this network enables efficient feature extraction while maintaining low computational overhead, it struggles with challenging scenarios such as fog and rain, often misclassifying dense fog as snow or light fog as clear weather [12]. This limitation underscores the need for additional sensor data to improve robustness under adverse conditions. Our study addresses these gaps by employing an adaptive data fusion framework that integrates LiDAR, RADAR, and camera modalities, thereby capitalizing on the complementary strengths of each sensor to enhance classification performance through prioritization of the most salient sensory information.

III. METHOD

Fig. 2 illustrates the architectural design of our proposed multimodal fusion framework, LRC-WeatherNet. The aim of our architectural design is to evaluate the effectiveness of the backbone models in modeling multimodal sensor input under varying environmental conditions, while maintaining computational efficiency and scalability.

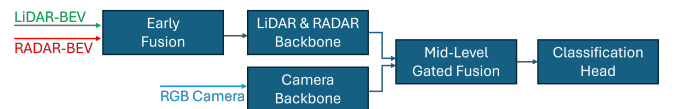


Fig. 2: The LRC-WeatherNet architecture. Early-fused LiDAR-RADAR BEV features are further fused with RGB features via a mid-level gated fusion module. Raw sensor outputs are passed through different backbone encoders.

Unlike previous approaches that rely on 3D point cloud backbones (PointPillars [4], VoxelNeXt [14], LiRa [8]), our framework utilizes EfficientNet-B0t [3] to process 2D representations of multimodal sensor data. By leveraging the high performance and efficiency of EfficientNet, the model is able to extract meaningful spatial and contextual features from Bird’s Eye View (BEV) projections of LiDAR and RADAR data, as well as from raw RGB images.

A. Data Representation

Raw LiDAR and RADAR point clouds are converted into 2D Bird’s Eye View (BEV) representations for robust weather classification. This transformation simplifies the inherently sparse and unstructured 3D data into a dense, top-down grid format, facilitating the extraction of spatially coherent patterns by convolutional backbone networks.

LiDAR and RADAR data are confined to a predefined frontal region aligned with the vehicle’s forward field of view to minimize noise and computational complexity. In the LiDAR frustum filtering, the longitudinal range (x) is set to $[0, 50]$ m (forward direction), while the lateral range (y) is $[-25, 25]$ m (sideways). This $50 \times 50 \text{ m}^2$ frustum approximates the vehicle’s immediate operational zone, excluding extraneous data from rear and peripheral regions. Next, each 3D LiDAR point is mapped to a 2D cell based on a fixed grid resolution defined as $x_{\text{bev}} = \lfloor \frac{x - x_{\text{min}}}{\text{resolution}} \rfloor$, $y_{\text{bev}} = \lfloor \frac{y - y_{\text{min}}}{\text{resolution}} \rfloor$, with coordinates clipped at the boundary limits. A grid resolution of 0.1 m is used to ensure sufficient granularity in BEV to resolve delicate environmental structures while avoiding computational overhead. In LiDAR BEV, the maximum intensity value is taken per cell to emphasize dominant surface returns. Finally, the raw 500×500 BEV grid is downsampled to 224×224 pixels via bilinear interpolation, aligning with the input specifications of the backbone architecture.

RADAR data, stored as 5D vectors ($x, y, z, \text{SNR}, \text{RCS}$), undergo analogous preprocessing with modality-specific adaptations. The region filtering is applied to retain forward-facing returns within the bounds $x = [0, 50]$ m and $y = [-25, 25]$ m. Next, two distinct BEV maps are constructed: SNR and RCS maps. The SNR map encodes signal-to-noise ratios to highlight reliable detections. The RCS map is for storing RADAR cross-section values correlating with material reflectivity and moisture content. After maximum values per cell are retained to emphasize dominant RADAR signatures, both SNR and RCS maps are combined into a $[2, H, W]$ tensor and resized to 224×224 pixels.

B. Early Fusion of LiDAR and RADAR

LiDAR and RADAR BEV representations are fused at the input level using timestamp-based alignment and channel-wise concatenation to exploit complementary sensor signatures for weather classification. This approach ensures temporal and spatial coherence between modalities while retaining raw sensor-specific features critical for detecting weather-induced distortions.

Formally, for a LiDAR tensor $L \in \mathbb{R}^{1 \times H \times W}$ and a RADAR tensor $R \in \mathbb{R}^{2 \times H \times W}$, the fused tensor F is defined

as $F = \text{Concat}(L, R) \in \mathbb{R}^{3 \times H \times W}$, where $H = W = 224$ aligns with the input requirements of the backbone model. This early fusion strategy allowed the backbone network to jointly learn cross-modal feature representations from LiDAR and RADAR within a shared input space.

C. Backbone Models

There exist two backbone models: one is for the early fused LiDAR and RADAR data, while the other is for the raw RGB camera images (see Fig. 2). Both backbones are customized EfficientNet-B0 model [3], initialized with pretrained ImageNet [15] weights without altering its architecture. An early fusion backbone is constructed to evaluate the benefits of combining complementary sensor modalities by integrating LiDAR and RADAR BEV features at the input level. The backbone for the camera image stream is the same EfficientNet-B0 model [3], pretrained on ImageNet [15]. All RGB inputs are resized to 224×224 and normalized using the ImageNet statistics.

D. Mid-Level Gated Fusion

The effective integration of spatial information derived from early fused LiDAR and RADAR sensors with visual information captured from camera images is crucial, yet poses significant challenges due to their unique data characteristics and representations. To address this, our proposed LRC-WeatherNet architecture in Fig. 2 employs a middle fusion strategy at the intermediate feature level after backbone feature extraction but before the final classification.

Inspired by works in [8], [16], [17], the middle fusion mechanism in our approach employs a gated fusion module, which dynamically modulates the contribution of each modality to the final fused representation. Instead of a simple feature concatenation, the gated fusion mechanism learns spatially and channel-wise adaptive weights during training. These weights are computed from the joint feature representations extracted by the backbone modules, thereby enabling the model to assess the relative importance of the early fused LiDAR and RADAR as well as camera features. This adaptive weighting enables the network to prioritize the most informative modality while suppressing the influence of those that provide less relevant or noisier signals.

The detailed architecture of our gated fusion mechanism is illustrated in Fig. 3. More specifically, let the inputs

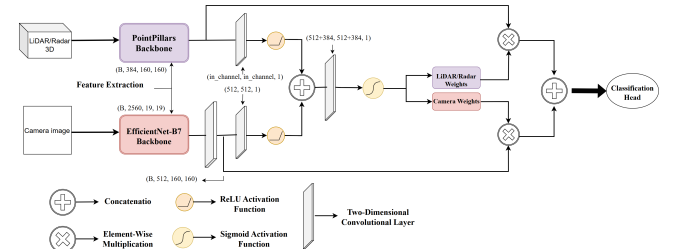


Fig. 3: Illustration of the gated fusion mechanism. Adaptive weights are computed to control modality contributions before generating the fused representation.

be $\mathbf{x}_f \in \mathbb{R}^{3 \times H \times W}$ and $\mathbf{x}_c \in \mathbb{R}^{3 \times H \times W}$ for the fused LiDAR and RADAR BEV data ($\mathbf{x}_f$) and camera streams ($\mathbf{x}_c$), respectively. Each input is passed through a modified EfficientNet-B0 backbone to compute the feature vectors as $\mathbf{f}_f = \text{EffNet}_f(\mathbf{x}_f) \in \mathbb{R}^d$ and $\mathbf{f}_c = \text{EffNet}_c(\mathbf{x}_c) \in \mathbb{R}^d$, where $d = 1280$ is the feature dimension after global average pooling. Next, each modality feature vector is first projected through its own MLP as $\mathbf{f}'_f = \text{ReLU}(W_f \mathbf{f}_f + \mathbf{b}_f)$ and $\mathbf{f}'_c = \text{ReLU}(W_c \mathbf{f}_c + \mathbf{b}_c)$.

These projected features are then concatenated as $\mathbf{f}_{\text{concat}} = [\mathbf{f}'_f, \mathbf{f}'_c] \in \mathbb{R}^{2d}$. This vector is passed through a gating MLP as $\mathbf{g} = \sigma(W_g \mathbf{f}_{\text{concat}} + \mathbf{b}_g) \in \mathbb{R}^{2d}$.

The gating vector is then split back to individual components $\mathbf{g} = [\mathbf{g}_f, \mathbf{g}_c]$, $\mathbf{g}_f, \mathbf{g}_c \in \mathbb{R}^d$ to be gated with the original feature vectors as $\mathbf{f}_f = \mathbf{f}_f \odot \mathbf{g}_f$ and $\mathbf{f}_c = \mathbf{f}_c \odot \mathbf{g}_c$.

Finally, both gated features are concatenated $\mathbf{f}_{\text{fused}} = [\mathbf{f}_f, \mathbf{f}_c] \in \mathbb{R}^{2d}$ to be passed through the classification head.

The gating module employs a sigmoid-activated linear layer to compute modality-specific weighting coefficients. These weights are applied element-wise to the original (non-projected) feature vectors, ensuring preservation of input fidelity. The adaptively weighted features are concatenated and propagated through a unified two-layer classifier to generate predictions across nine weather condition classes.

E. The Classification Head

To ensure a fair comparison with baseline unimodal and multimodal experiments, the original classification head of both backbones was replaced with a custom classifier comprising:

- a 512-unit dense layer with Batch Normalization and ReLU activation,
- a dropout layer (dropout rate = 0.3), and
- a 9-unit output layer for softmax-based multi-class classification.

More specifically, the final fused feature vector ($\mathbf{f}_{\text{fused}}$) is passed through a two-layer MLP as $\hat{\mathbf{y}} = W_2 \cdot \text{ReLU}(\text{BN}(W_1 \cdot \mathbf{f}_{\text{fused}} + \mathbf{b}_1)) + \mathbf{b}_2$. Here, $\hat{\mathbf{y}} \in \mathbb{R}^9$ denotes the predicted class logits. BN represents Batch Normalization. Dropout is applied before the final layer.

IV. EXPERIMENTS

A. Training Protocol

Dataset splits were stratified into training, validation, and test sets across nine weather classes.

The model was trained using the AdamW optimizer with a fixed learning rate of 3×10^{-4} and weight decay of 1×10^{-4} . Learning rate scheduling was applied to reduce the learning rate when validation loss plateaued.

To mitigate class imbalance, class-weighted cross-entropy loss was employed [18]. Class weights were computed inversely proportional to the class frequency in the training set and clamped to avoid extreme values. Gradient clipping (max norm = 5.0) was applied to stabilize training.

B. Baselines

In addition to the recent weather-type classification models such as RECNet [11] and RangeWeatherNet [12], we trained various unimodal and multimodal baselines.

Unimodal: We trained LiDAR-only, RADAR-only, and Camera-only models using the same EfficientNet-B0 backbone for a fair and consistent comparison. The only architectural difference lies in the input configuration: the first convolutional layer of EfficientNet-B0, originally defined for 3-channel RGB inputs, was modified to accept single-channel input, corresponding to the LiDAR intensity-only BEV representation in the LiDAR-only modal. Specifically, the original 3-channel kernel weights were averaged across the input dimension to initialize a new `Conv2d` layer with a single input channel. This preserved compatibility with the pretrained ImageNet weights while allowing the model to accept LiDAR tensors of shape $[1 \times 224 \times 224]$. The input data was normalized using dataset-specific statistics: mean = 0.0471, std = 0.1659. In the RADAR-only model, the first convolutional layer of EfficientNet-B0 was modified to accept 2-channel RADAR BEV tensors, replacing the default 3-channel RGB input. The pretrained weights for this layer were adapted by averaging the original weights across the RGB channels. The input data was normalized using precomputed dataset-specific statistics: mean = [0.0072, 0.0040], std = [0.2507, 0.2326]. All other model components, optimizer settings, loss function, and evaluation protocol remained unchanged, ensuring that performance differences arise solely from the input modality rather than architecture or training procedure variations.

During training, extensive data augmentation was applied to the camera stream to improve generalization. The augmentation includes horizontal and vertical flips, random rotations ($\pm 45^\circ$), color jittering, affine translations, and random resized cropping. This augmentation was applied both in Camera-only and in the multimodal fusion pipelines to ensure fairness in input variability.

Given the sparse and noisy nature of RADAR data, a moderate augmentation pipeline, including random affine rotations ($\pm 5^\circ$) and Gaussian noise injection (2% noise intensity), was applied during training. These augmentations aimed to simulate variability in RADAR returns caused by sensor drift and minor environmental changes, thereby enhancing the model's generalization ability.

LRC-WeatherNet-PP: We further evaluated the performance of our proposed LRC-WeatherNet using a PointPillars-based backbone for the early fused LiDAR and RADAR readings, while preserving the camera branch.

This architecture, named LRC-WeatherNet-PP, processes raw 3D point clouds into pillar-encoded pseudo-images, enabling the model to retain multi-scale geometric detail through spatially-aware feature maps at different resolutions. PointPillars [4] represents point cloud data using vertical columns, referred to as *pillars*, on the x-y plane. Unlike voxel-based models that discretize all three spatial dimensions, PointPillars preserves the z-dimension (height) as a

TABLE I: Quantitative performance evaluation across different unimodal and fusion configurations.

Model	Val. Acc.	Test Acc.	Macro F1	Inference (ms)	Params	GMAC
Camera-only	91.96%	77.86%	0.76	4.86	4.67M	0.41
LiDAR-only	57.46%	56.34%	0.55	3.52	4.67M	0.41
RADAR-only	31.99%	27.32%	0.26	3.79	4.67M	0.41
Early Fusion (LiDAR + RADAR)	62.98%	61.06%	0.59	4.21	4.67M	0.41
LRC-WeatherNet	93.82%	86.66%	0.85	7.13	19.17M	0.84
LiDAR-only-PP	83.11%	81.49%	0.81	8.57	4.86 M	29.57
RADAR-only-PP	43.80%	37.80%	0.37	3.51	4.86 M	01.85
Early Fusion-PP (LiDAR + RADAR)	82.10%	82.30%	0.81	8.58	4.86 M	29.57
LRC-WeatherNet-PP	91.67%	87.77%	0.87	64.40	8.68 M	88.67

continuous feature within each point. Each point is described by nine input features: the four original attributes (x , y , z , intensity), three offsets from the mean point within the pillar (Δx , Δy , Δz), and two offsets from the pillar center in the x - y plane. These features are passed through a learnable encoding layer that transforms raw point features into richer point-wise representations, which are then aggregated and processed using efficient 2D convolutions. This design allows the model to effectively capture 3D structure while maintaining computational efficiency during inference. Unlike PointPillars, the BEV transformation completely removes the z -dimension, resulting in purely 2D inputs.

C. Dataset

We used the Michigan State University Four Seasons (MSU-4S) dataset introduced in [5], including 100K data samples from different challenging weather scenarios, such as rain and snow. The dataset was recorded over different time periods in four seasons of the year across five distinct geographic regions: thoroughfare (light blue), forest (green), industrial (red), central hub (dark blue), and neighborhood (yellow). Each region reflects a unique environmental context. Most of the data was collected during the evening hours, but some sessions were recorded at other times of day.

The dataset comprises two LiDAR sensors, six RADAR sensors (Continental ARS430), three cameras, and supplementary odometry sensors. For simplicity, we adopted a more constrained evaluation setup, utilizing only the front-view streams from the RADAR and LiDAR (Ouster OS-1, 64-line) sensors together with a single front-facing camera. The dataset was divided into nine classes, each representing a specific combination of season and weather condition. These classes capture both inter-seasonal differences and intra-seasonal variations.

Sensor data lacking corresponding measurements from other sensors at the same timestamp were excluded to ensure consistency and reliability in weather classification. This filtering step guarantees that only fully synchronized data samples are used in multi-sensor fusion experiments.

D. Quantitative Results

A summary of the quantitative results on MSU-4S [5] is provided in Table I. Metrics reported include test accuracy,

macro-averaged F1 score, inference time, number of parameters, and computational cost (GMAC).

The RADAR-only baseline model exhibited limited classification capability when using RADAR BEV inputs exclusively. It achieved a test accuracy of 27.32% and a macro-averaged F1 score of 0.26. This low performance reflects the limited discriminative power of RADAR SNR and RCS maps, which encode low-dimensional motion and reflectivity patterns but lack fine spatial structure or texture. Furthermore, RADAR suffers from low vertical resolution and multipath interference, particularly in urban environments, where reflections from the ground or vehicles may mimic precipitation effects.

The LiDAR-only baseline demonstrated a substantial improvement over the RADAR-only approach, achieving a test accuracy of 56.34% and a macro-averaged F1 score of 0.55. This limited performance stems from scene-level clutter or reduced LiDAR returns due to reflection loss, especially in snowy conditions. Snow essentially introduces scattering artifacts that reduce point cloud density, leading to mispredictions into rain or spring snow categories.

The Camera-only model achieved the strongest results among all unimodal baselines, with a test accuracy of 77.86% and a macro-averaged F1 of 0.76. RGB excelled in scenarios where chromatic and semantic richness were present, e.g., vibrant vegetation. The model leveraged textural differences and lighting variations to distinguish between otherwise geometrically similar scenes. While this baseline excelled under well-lit, high-contrast conditions, its performance degraded in settings affected by glare or low ambient light.

In summary, RADAR is robust to weather but lacks spatial expressiveness. LiDAR provides rich geometric data but struggles under occlusion, snow, or in visually ambiguous urban scenes. The camera sensor achieved the highest performance thanks to rich semantics, but is vulnerable to lighting instability. These results justify the pursuit of multimodal fusion strategies, where LiDAR provides structure, camera provides semantics, and RADAR adds robustness.

Combining LiDAR and RADAR BEV features through our early fusion improved performance relative to RADAR-only and LiDAR-only unimodal baselines, reaching a test accuracy of 61.06% and a macro F1 score of 0.59. Our early fusion model also maintains computational efficiency, with 4.67 million parameters and an inference time of

just 4.21 ms, while providing improved discriminability in challenging conditions.

Finally, in our proposed LRC-WeatherNet, the mid-level gated fusion, combining early-fused LiDAR and RADAR features with camera image features, produced the best overall performance, achieving 86.66% test accuracy and a macro-averaged F1 score of 0.85.

Our baseline experiments (LRC-WeatherNet-PP), where a PointPillars-based backbone is used for the early fused LiDAR and RADAR data, achieved a classification accuracy of 87.77% and an F1 score of 0.87, demonstrating effective integration of spatial and visual modalities. However, the LRC-WeatherNet-PP baseline becomes substantially expensive due to extremely high computation time (64.40 ms) in contrast to 7.13 ms for the original LRC-WeatherNet, as shown in Table I.

Fig. 4 illustrates the prediction distribution of the proposed LRC-WeatherNet fusion model across all weather classes in a confusion matrix. As shown in this confusion matrix, our model LRC-WeatherNet exhibits high diagonal dominance and low off-diagonal noise. Class-wise predictions are notably sharp, reflecting stronger model confidence and fewer ambiguous misclassifications. The model’s confusion matrix indicates clearer separation between visually or seasonally adjacent categories, with low cross-class confusion. These gains are largely attributed to the gated fusion strategy, which enables the network to control the contribution of each modality dynamically. The gated mechanism learns context-dependent weights, emphasizing features that are more informative for a given scene while attenuating noise from less relevant channels. This allows the network to preserve LiDAR’s structural precision, RADAR’s weather robustness, and RGB camera’s semantic richness in a coherent, task-aware representation.

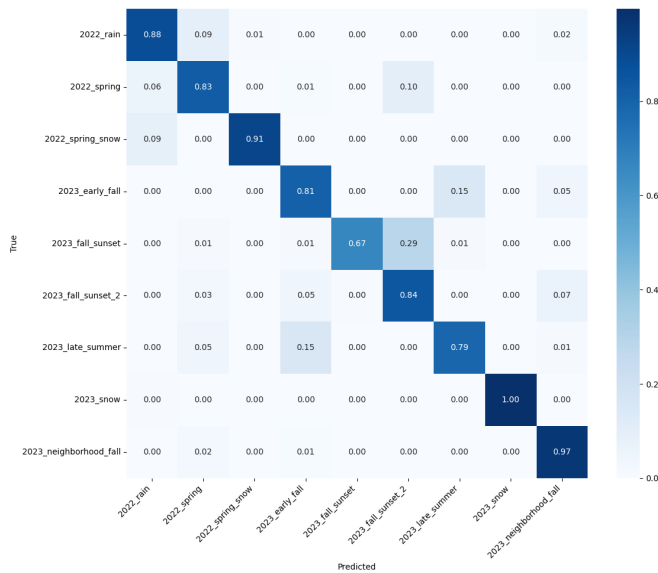


Fig. 4: Confusion matrix of the proposed LRC-WeatherNet model predictions on MSU-4S.

E. Qualitative Results

Fig. 5 depicts two sample images from the MSU-4S dataset [5] for the qualitative evaluation. Here, early fused LiDAR and RADAR point clouds are superimposed on the RGB camera image, and the proposed LRC-WeatherNet correctly predicts the right weather type for each scene.

To elucidate the internal dynamics of LRC-WeatherNet, we analyzed the modality-specific contributions facilitated particularly by the mid-level gated fusion mechanism. Fig. 6 illustrates how the model adeptly balances spatial information from the early fused LiDAR and RADAR data with semantic content from RGB camera imagery, adapting to the contextual demands of each scene. The top-left image in Fig. 6 presents the RGB camera image of a snow-covered scene characterized by flat lighting and low-contrast textures. In this specific instance of a snowy scene, the model accurately predicted the right class with the highest probability. The gated fusion mechanism assigned weights of 0.64 to RGB and 0.35 to the early fused LiDAR and RADAR, indicating a higher reliance on visual cues in this snowy environment. The corresponding RGB weight map (Fig. 6 top-right) reveals heightened activation in the lower regions of the image, particularly where snow accumulation and surface reflections are prominent. This suggests that the model leverages visual patterns such as brightness and texture variations to identify snow-covered areas. Conversely, the early fused LiDAR and RADAR weight map (Fig. 6 top-middle) exhibits strong activation along dense point cloud returns, especially near the foreground and building contours. This reflects the structural information captured by the fused spatial modalities, which complements the visual data, particularly in areas where visual cues may be less distinct.

The bottom-left image in Fig. 6 shows a clear daytime scene at an intersection, with the model predicting the right class with the highest probability. The gated fusion weights were 0.43 for RGB and 0.56 for the early fused LiDAR and RADAR data, indicating a stronger reliance on the LiDAR and RADAR features. This clear daytime scene image displays moderate visual cues such as green trees and a clear sky, but lacks prominent seasonal indicators. In contrast, the early fused LiDAR and RADAR contribution map in Fig. 6 bottom-middle reveals dense and layered return patterns, characteristic of tree canopies and structural road elements.

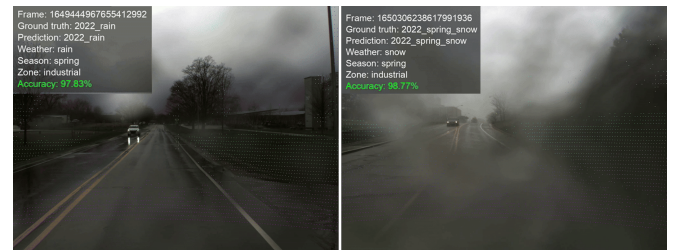


Fig. 5: A sample qualitative result showing a spring rainy (left) and spring snowy (right) scenes predicted accurately.

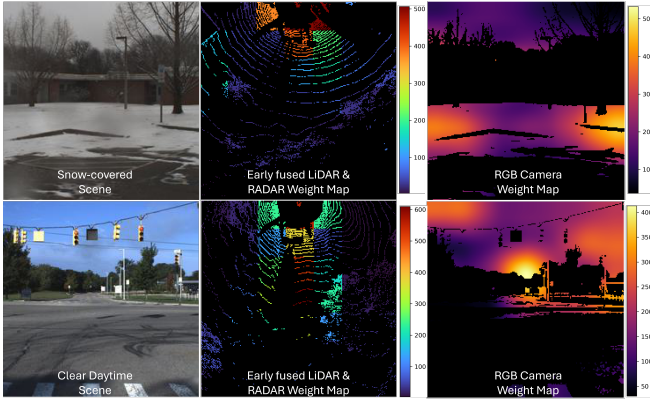


Fig. 6: Visualization of weight maps learned in the gated fusion mechanism for two sample scenes: snowy (top row) and clear daytime (bottom row) scenes.

These geometric and reflectivity-based patterns offer highly distinctive signatures that help the model recognize early fall scenes. This illustrates how, in environments with limited semantic variability in RGB images, the model shifts attention toward spatial and reflective structure captured in 3D BEV data. The LiDAR and RADAR stream provided more discriminative information in this case, allowing the model to confidently identify the correct seasonal class. Taken together, these examples highlight the adaptive nature of the gated fusion mechanism: it does not apply fixed weights to each modality, but instead responds to scene-specific characteristics by emphasizing the sensor stream offering the most informative features. This dynamic allocation of attention enables the model to operate effectively across diverse visual and structural conditions and is a key factor behind its high classification accuracy and minimized inter-class confusion.

These learned weight map examples in Fig. 6 highlight the adaptive behavior of the proposed LRC-WeatherNet model: in visually ambiguous or low-contrast environments, LiDAR and RADAR provide robust spatial and reflective information, whereas in well-lit scenes with clear visual cues, RGB becomes more dominant.

V. DISCUSSION & LIMITATIONS

This work introduced LRC-WeatherNet to address the challenge of real-time weather type classification using large-scale multimodal sensor data comprising LiDAR, RADAR, and camera inputs. Through extensive experiments on the MSU-4S dataset [5], we evaluated the performance of different unimodal and multimodal baseline approaches. The results demonstrated that our middle fusion approach in LRC-WeatherNet, combining early-fused LiDAR and RADAR data streams with the camera input, consistently achieved the highest classification accuracy with real-time performance. Further analysis of LRC-WeatherNet revealed its ability to dynamically adjust attention across modalities based on scene context. Camera input was dominant in visually clear conditions, whereas fused LiDAR and RADAR data became

more influential in ambiguous scenarios.

Our LRC-WeatherNet framework applies a uniform architecture across modalities, using separate but structurally identical EfficientNet-B0 branches for RGB and fused LiDAR-RADAR inputs. While this leads to a higher combined parameter count of 19.17 million, the compute demands are far lower at 1.26 GMACs, with a much faster inference time of 8.27 milliseconds.

From a performance standpoint, LRC-WeatherNet-PP achieves slightly higher test accuracy (87.77% vs. 86.41%) and macro-averaged F1 score (0.870 vs. 0.8452), likely due to its superior ability to model fine-grained 3D structure from LiDAR and RADAR data. However, LRC-WeatherNet delivers strong generalization with over nine times faster inference and significantly lower computational cost.

Although the LRC-WeatherNet-PP baseline achieves higher overall accuracy across most classes, it exhibits increased confusion among fall-related classes and signs of overfitting in those categories. In contrast, our LRC-WeatherNet model demonstrates more balanced class-wise performance. The confusion matrix in Fig. 4 indicates improved discriminability of LRC-WeatherNet among visually and structurally similar classes. While its overall accuracy is slightly lower than that of the PointPillars-based fusion model, LRC-WeatherNet offers enhanced generalization and robustness in distinguishing fall-related seasonal categories.

Overall, this work confirms the value of multimodal fusion for robust and accurate weather classification. The findings emphasize the importance of selecting appropriate fusion strategies and model architectures, and they underscore the benefit of context-sensitive sensor integration in enhancing performance under diverse environmental conditions.

Limitation: The selection of the dataset was primarily influenced by the lack of publicly available options that provide synchronized LiDAR, RADAR, and RGB camera data. Specifically, the MSU-4S dataset [5] includes only one class representing rain and two classes corresponding to snowy conditions. The remaining categories depict seasonal environments, such as spring, summer, and fall, but without active precipitation. These seasonal scenes are defined more by ambient features like foliage color and lighting rather than by dynamic weather phenomena such as fog, heavy rainfall, hail, or snowstorms. As a result, the model has limited exposure to the types of weather conditions that most impact sensor reliability and visibility. In addition, the dataset lacks rare yet critical edge cases such as dense fog, sleet, hail, or nighttime snow, which are essential for training and evaluating perception systems intended for real-world deployment.

REFERENCES

- [1] F. Sezgin, D. Vriesman, D. Steinhauser, R. Lugner, and T. Brandmeier, "Safe autonomous driving in adverse weather: Sensor evaluation and performance monitoring," in *arXiv preprint arXiv:2305.01336*, 2023, accessed: 2025-05-20. [Online]. Available: <https://arxiv.org/abs/2305.01336>

- [2] Y. Li, J. Deng, Y. Zhang, J. Ji, H. Li, and Y. Zhang, "Ezfusion: A close look at the integration of lidar, millimeter-wave radar, and camera for accurate 3d object detection and tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 182–11 189, 2022. [Online]. Available: <https://doi.org/10.1109/LRA.2022.3193465>
- [3] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," arXiv preprint arXiv:1905.11946, 2020, accessed: May 2025. [Online]. Available: <https://arxiv.org/abs/1905.11946>
- [4] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.
- [5] D. Kent *et al.*, "MsU-4s - the michigan state university four seasons dataset," in *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 22 658–22 667. [Online]. Available: <https://doi.org/10.1109/CVPR52733.2024.02138>
- [6] T. Teepe, P. Wolters, J. Gilg, F. Herzog, and G. Rigoll, "Earlybird: Early-fusion for multi-view tracking in the bird's eye view," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. IEEE, 2024, pp. 102–111.
- [7] A. El-Dalhmeh, M. El-Dalhmeh, and J. Li, "Enhanced vehicle detection through multi-sensor fusion utilizing yolo-nas and faster r-cnn," in *Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–6.
- [8] J. Song, L. Zhao, and K. A. Skinner, "Lirafusion: Deep adaptive lidar-radar fusion for 3d object detection," *IEEE international conference on robotics and automation (ICRA)*, 2024.
- [9] J. Kim, J. Choi, Y. Kim, J. Koh, C. C. Chung, and J. W. Choi, "Robust camera lidar sensor fusion via deep gated information fusion network," in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1620–1625.
- [10] Y. Li, A. W. Yu, T. Meng, B. Caine, J. Ngiam, D. Peng, and M. Tan, "Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 182–17 191.
- [11] M. Introvigne, A. Ramazzina, S. Walz, D. Scheuble, and M. Bijelic, "Real-time environment condition classification for autonomous vehicles," *arXiv preprint*, 2024, [Online]. [Online]. Available: <https://arxiv.org/abs/2405.19305>
- [12] G. Sebastian, T. Vatter, L. Lukic, C. Bürgy, and T. Schumann, "Rangeweathernet for lidar-only weather and road condition classification," in *Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 777–784.
- [13] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.
- [14] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Voxelnext: Fully sparse voxelnet for 3d object detection and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 674–21 683.
- [15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [16] J. H. Yoo, Y. Kim, J. Kim, and J. W. Choi, *3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-view Spatial Feature Fusion for 3D Object Detection*. Springer International Publishing, 2020, p. 720–736.
- [17] J. Kim, J. Koh, Y. Kim, J. Choi, Y. Hwang, and J. W. Choi, "Robust deep multi-modal learning based on gated information fusion network," in *Computer Vision – ACCV 2018*, 2018, p. 90–106.
- [18] T. Cortinhal, G. Tzelepis, and E. Erdal Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *International Symposium on Visual Computing*. Springer, 2020, pp. 207–222.

PAPER 03: Benchmarking Vehicle Trajectory Tracking and Prediction Methods in Adverse Weather Conditions

Mahmoud Alshaikh¹, Joel Ekstrand¹, Gayan Himesha Pathirage¹,
Abu Mohammed Raisuddin¹, Idriss Gouigah¹, and Eren Erdal Aksoy¹

Abstract—In this work, we benchmark the robustness of 3D object-tracking algorithms under adverse weather conditions, such as snowy scenarios affecting autonomous vehicle perception systems. The principal objective is to evaluate and compare the performance of traditional and advanced AI-based object tracking methods: Kalman Filter, Extended Kalman Filter, Particle Filter, and ByteTrack. Each method was assessed using 3D LiDAR point cloud data obtained from the Canadian Adverse Driving Conditions (CADC) dataset [1], representing harsh winter conditions, and the nuScenes dataset [2], representing clear, optimal weather conditions. We further anticipate the tracked object trajectories in the upcoming frames by employing an LSTM-based neural network.

The methodology involved processing detected 3D object bounding boxes in sequential LiDAR point clouds and tracking objects through association and state estimation. Standard metrics such as HOTA, IDF1, AMOTA, and AMOTP were used to measure tracking accuracy and consistency across both datasets. Results indicated significant performance degradation for all algorithms under snowy weather conditions compared to clear weather. The Kalman filtering methods suffered from the linear behavior and the noise, while the Particle Filter provided a more robust estimation due to its ability to cope with the uncertainty via its multiple hypotheses. Deep learning-based ByteTrack demonstrated better performance, with higher accuracy and fewer identity switches in challenging scenarios. The study concluded that deep learning-based tracking methods provide a more solid guarantee on point trajectory and, thus, offer enhanced reliability for autonomous vehicles in challenging environments. The source code, trained models, and the online supplementary information are available at the following URL: <https://github.com/Mahmoudalshaikh00/An-Experimental-Study-on-ObjectTracking.git>.

I. INTRODUCTION

Autonomous vehicle technology has shown notable advances with the potential to improve traffic safety, optimize transportation efficiency, and reduce environmental impact. The perception system is a core component of autonomous vehicles and allows them to comprehend and interact with the environment. One of the advanced perception sensors is Light Detection and Ranging (LiDAR), which has been identified as an important ingredient for capturing accurate 3D representations of the environment. Yet, adverse weather like snow, rain, and fog brings great challenges to the

LiDAR-based perception systems, especially for the 3D object tracking.

Autonomous vehicles rely on multiple object tracking (MOT) in 3D space to detect and follow objects like vehicles and pedestrians [3], [4]. Fusion of different sensors such as LiDAR, cameras, and RADAR helps track motion despite noise and occlusions [3], [4]. Systems like VoxelNet [5] and PointPillars [6] process 3D point clouds, and combining 2D and 3D views improves accuracy [4]. MOT algorithms detect objects and link them across frames.

This study provides a comprehensive benchmarking of different object (e.g., vehicle) tracking methods, particularly in adverse weather conditions such as snowy conditions. Our benchmarking involves not only various traditional tracking approaches, such as Kalman Filter (KF), Extended KF, and Particle Filter, but also advanced deep learning based methods, such as ByteTrack.

Kalman Filter (KF) and Extended Kalman Filter (EKF) estimate vehicle motion by combining models with sensor data in real time. Studies show these filters improve localization and tracking by combining multiple sensors, adding map data [7], splitting motion models [8], or combining with neural networks [9]. Adaptive KF combines GPS and inertial sensors to reduce drift and handle GPS dropouts [10]. EKF is also used in off-road tasks like land navigation [11]. They help manage uncertainty, work with many sensors, and support real-time decisions. EKF handles nonlinear systems more accurately.

Particle Filters (PFs) are Bayesian estimators that track a system's state using many random samples (particles), each representing a possible state with a weight [12]. Unlike Kalman Filters, PFs work well with non-linear motion and non-Gaussian noise [13]. Each particle follows the motion model to predict a new state. Sensor data is then used to weight the particles based on how likely each state is. Resampling focuses on high-probability particles, improving estimation over time [12]. PFs handle multiple possible states at once, making them useful in uncertain environments (e.g., when a vehicle's location is ambiguous) [14]. They also handle outliers and support complex sensor fusion better than linear filters [12].

ByteTrack is a multi-object tracking method that uses low-certainty detections to restore occluded or missing tracks [15], [16]. It combines a Kalman filter, the Hungarian algorithm for data association, and tracks unmatched detections [16]. ByteTrack's approach increases recall without significant precision loss, making it valuable in autonomous

*Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them. Project grant no. 101069576.

¹Halmstad University, School of Information Technology, Center for Applied Intelligent Systems Research, Halmstad, Sweden
eren.aksoy@hh.se

driving [17], surveillance [16], [18], medical imaging [19], and robotics [20]. It handles crowded scenes and occlusions more reliably, which is critical for real-world applications [21].

We further explore the potential of a Long Short-Term Memory (LSTM) based recurrent neural network to predict the future trajectories of the tracked objects. LSTMs are good at learning long term dependencies in sequential data, which makes them applicable for motion pattern prediction.

II. RELATED WORK

Note that although there exist more advanced 3D MOT methods in the literature, we, in this study, focus on KF, EKF, PF, and ByteTrack. To name a few, the work in [22] proposes a 3D tracking method using an Adaptive Cubature Kalman Filter (ACKF) with a constant turn rate and velocity (CTRV) motion model. The ACKF adapts to sensor bias and uncertainty, improving tracking robustness. This work also introduces a graph-based data association model to better match predictions with detections. Tested on the KITTI dataset [23], this method outperforms baseline trackers in accuracy and speed, showing how KF variants and improved association strategies enhance tracking in real-world driving. The recent work in [24] enhances DeepSORT by adding a quantum computing module for better tracking under occlusion. The quantum approach solves data association by evaluating many match options in parallel, reducing identity switches when objects overlap. This method improves accuracy in dense scenes and shows how deep learning-based trackers can be improved with emerging computing methods. It highlights a path to handle persistent occlusions in autonomous driving. MUTR3D [25] is an end-to-end framework for tracking 3D objects using multiple cameras. Unlike traditional methods that depend on matching object appearances or positions, MUTR3D introduces *3D track queries* to model objects across different cameras and time frames. It links 3D object tracks to 2D images through camera transformations and refines them using image features. The system employs a set-to-set loss function, eliminating the need for post-processing steps like non-maximum suppression. Tested on the nuScenes dataset [2], MUTR3D outperforms previous methods by 5.3 percent in AMOTA, demonstrating its effectiveness in multi-camera 3D tracking scenarios [25].

Vehicle trajectory prediction also benefits from LSTMs. The work in [26] showed LSTMs accurately forecast future positions, while Hou et al. [27] improved accuracy further by sharing information between nearby vehicles. Egovehicle path prediction uses similar ideas; Fan et al. [28] integrated driving style data in an LSTM to forecast lane changes. LSTM models consistently outperform traditional methods by better handling temporal dependencies. Their predictions help autonomous vehicles plan safer maneuvers and navigate complex environments more reliably.

III. METHOD

The main input for the trajectory tracking and prediction methods is 3D object bounding boxes detected by a 3D object

detection method. The data used for tracking consists of a list of 100 frames, where each frame contains a set of detected bounding boxes. Each bounding box is represented as a 7-dimensional array in the format $[x, y, z, dx, dy, dz, \text{yaw}]$, where (x, y, z) represent the 3D position of the box center, (dx, dy, dz) define the box dimensions along each axis, and yaw specifies the orientation around the vertical (z) axis. This consistent encoding provides a unified input format for all benchmarked tracking algorithms: Kalman Filter (KF), Extended Kalman Filter (EKF), Particle Filter (PF), and ByteTrack. The goal of each algorithm is to associate the same object across frames with a consistent identity, to assign and maintain the same ID for each detected object throughout the sequence. Using the same detection input across all methods ensures that performance differences can be attributed solely to the tracking logic, not the object detection step. This standardized input is crucial for fair and interpretable evaluation of multi-object tracking accuracy and identity preservation.

A. Kalman Filter (KF) 3D Object Tracking

This method uses a Kalman Filter (KF) to track 3D bounding boxes over time. It estimates object position and velocity in 3D space from frame-by-frame sensor data, such as bounding boxes generated from LiDAR. The tracking pipeline includes: (1) loading bounding boxes and configuration, (2) initializing KF state, (3) associating detections with tracks using the Hungarian algorithm, (4) creating updated 3D boxes, and (5) outputting annotated results.

KF uses a six-dimensional state vector:

$$\mathbf{x} = [x \ y \ z \ v_x \ v_y \ v_z]^T,$$

with a constant velocity model to predict new states. The transition matrix F includes the time step Δt , and the measurement matrix H maps position from the state. Noise matrices Q and R reflect uncertainty in motion and sensor data. Data association uses Euclidean distance between predicted and detected box centers. Matches below a distance threshold are accepted; others are ignored. KF then updates matched tracks and predicts the next state for all trackers. Additional features include 2D velocity estimation from center displacement and optional point cloud integration for checking if objects contain sufficient LiDAR points. Bounding boxes are annotated with metadata and saved in a JSON-friendly format. In section VI, we provide the pseudo code and link to the source code.

This KF-based approach offers a fast, interpretable solution for 3D object tracking and can be extended by adjusting the motion model or integrating richer sensor data.

B. Extended Kalman Filter (EKF) for 3D Object Tracking

The Extended Kalman Filter (EKF) builds on KF method but supports non-linear system dynamics. While the general pipeline data input, matching, and tracking remain the same, EKF replaces matrix operations with functions and Jacobians to support more complex models. More specifically, EKF defines:

- **State Transition Function:** $\mathbf{f}(\mathbf{x}, \Delta t)$, which updates position using velocity.
- **Measurement Function:** $\mathbf{h}(\mathbf{x})$, which extracts position from state.

Since $\mathbf{f}$ and $\mathbf{h}$ can be non-linear, EKF computes their Jacobians:

$$F = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}, \quad H = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}.$$

In the update (correction) step, given a measurement $\mathbf{z}$, EKF computes the predicted measurement $\hat{\mathbf{z}} = \mathbf{h}(\mathbf{x})$ and the Jacobian H . The measurement residual $\mathbf{y}$ is

$$\mathbf{y} = \mathbf{z} - \hat{\mathbf{z}}.$$

then compute the measurement residual covariance,

$$S = H \mathbf{P} H^T + \mathbf{R},$$

and the Kalman gain,

$$K = \mathbf{P} H^T S^{-1}.$$

Finally, update the state and covariance:

$$\mathbf{x} \leftarrow \mathbf{x} + K \mathbf{y}, \quad \mathbf{P} \leftarrow (\mathbf{I} - K H) \mathbf{P}.$$

Because the Jacobian H is reevaluated at each update to account for potential non-linearity, EKF can correct the state even if $\mathbf{h}$ is significantly more complicated than a simple linear projection.

In the prediction step, EKF computes the following:

$$\mathbf{x} \leftarrow \mathbf{f}(\mathbf{x}, \Delta t), \quad \mathbf{P} \leftarrow F \mathbf{P} F^T + Q.$$

Although the provided functions are linear in this case, the EKF framework allows for future extensions such as adding yaw, acceleration, or non-Cartesian sensor models. This flexibility makes EKF more suitable for real-world scenarios with non-linear motion or sensor readings.

In section VI, we provide the pseudo code and link to the source code.

C. Particle Filter (PF) for 3D Object Tracking

We also explore the potential of a dense Particle Filter (PF) to track 3D object bounding boxes in LiDAR point cloud data. Each object is represented by $N = 1,000,000$ particles, with 6D states for position (x, y, z) and velocity (v_x, v_y, v_z) . Particles start from a Gaussian distribution around the detected bounding box, with zero initial velocities and equal weights.

In the prediction and update state, particles are predicted forward using a constant velocity model in each frame:

$$x_i^{(t)} = F \cdot x_i^{(t-1)} + \mathcal{N}(0, Q),$$

where Q adds noise to model uncertainty. After a new observation, weights are updated based on the Euclidean distance to the detected center, normalized to sum to 1. If weight diversity drops too low, resampling replicates high-weight particles to avoid degeneracy.

During the state estimation and track management, the estimated object position is defined as the weighted mean of the particles:

$$\hat{\mathbf{x}} = \sum_{i=1}^N w_i \cdot x_i.$$

During tracking, PF manages active, lost, and new tracks. Active tracks match to detections by nearest-neighbor association. Lost tracks are restored if a new observation matches within a threshold.

D. ByteTrack for 3D Object Tracking

We also explore ByteTrack [15], originally designed for 2D tracking, to track 3D bounding boxes by projecting them onto a 2D plane. The process follows a similar structure to Kalman Filter methods: receiving frames, associating detections, and updating trackers. The main difference is that ByteTrack uses a refined data association strategy developed for YOLOX object detection model [29]. In ByteTrack, we have the following steps.

1) *3D to 2D Projection:* Each 3D box $[x, y, z, \text{length}, \text{width}, \text{height}, \text{yaw}]$ is projected onto 2D as (x_1, y_1, x_2, y_2) by ignoring height and yaw. Dummy confidence scores are also introduced.

2) *ByteTrack Association:* ByteTrack performs multi-step association: first matching high-confidence detections, then refining with lower-confidence ones. This produces stable track IDs even if some detections are missed. It replaces the Hungarian algorithm used directly in Kalman Filter methods.

3) *Speed-Aware Expansion:* To help with fast-moving objects, boxes are expanded or shifted based on a simple velocity estimate:

$$(v_x, v_y) = (c_x - c_{x,\text{prev}}, c_y - c_{y,\text{prev}}),$$

where (c_x, c_y) is the box center. The box is then adjusted by a small fraction of this velocity, improving association in dynamic scenes.

4) *Reconstruction in 3D:* After ByteTrack finishes, 2D boxes with track IDs are matched back to their original 3D boxes using index alignment. This ensures each 3D bounding box is assigned a persistent ID across frames.

While ByteTrack excels at 2D tracking, projecting 3D boxes into 2D simplifies some spatial details. Still, this method is practical and efficient for 3D object tracking.

E. LSTM for 3D Object Trajectory Prediction

This section presents an LSTM-based recurrent model for predicting 3D bounding box trajectories. The model uses past positions and velocities to predict future object centers. Two region-specific models are trained: one for right-lane motion ($y \in [-3, 7.5]$) and one for left-lane motion ($y \in (7.5, 20]$) to account for different traffic behaviors.

1) *Data Preparation and Normalization:* Bounding box data is loaded from JSON files. Trajectories are filtered by spatial bounds and split into:

- Past sequence: $(T_p = 5, 4)$ with positions and computed velocities.

- Future sequence: $(T_f = 3, 2)$ as displacement from the last past position.

Inputs and outputs are normalized using dataset statistics.

2) *Model Architecture*: The network uses a sequence-to-sequence LSTM structure:

- Five-layer LSTM encoder with dropout.
- A compressed vector is repeated to initialize the decoder.
- Five-layer LSTM decoder predicts future displacements.
- A dense layer maps to 2D displacements.

The model is trained with the Adam optimizer and the Mean Absolute Error (MAE) loss.

3) *Region-Specific Modeling*: Separate models are trained for the right and left lane regions to improve accuracy by handling differences in driving patterns.

4) *Training and Evaluation*: Models are trained for 100 epochs with a batch size of 256. Metrics include ADE, FDE, MAE, and RMSE to evaluate prediction accuracy.

5) *Stepwise Prediction*: During inference, the model predicts one step at a time, updating the input sequence with predicted positions and recalculated velocities.

6) *Model Saving and Visualization*: Models and normalization data are saved for future use. Training loss and prediction performance are visualized as shown in Figures 1 and 2. Fig. 1 depicts that during training, the MAE loss drops in both left and right regions. Likewise, Fig. 2 shows

that predictions are accurately following the expected ground truth values in both regions.

IV. EXPERIMENTAL RESULTS

A. Evaluation Metrics

For the evaluation, we employ different higher-order metrics [30] such as higher order tracking accuracy (HOTA), identity switches (IDs), identity F1 Score (IDF1), average multi-object tracking accuracy (AMOTA), and average multi-object tracking precision (AMOTP). The IDS score indicates the accuracy of a track by describing how many times the identity of the tracked object changes within the track. HOTA combines detection and association accuracies in a more balanced manner, while IDF1 indicates how long the tracker correctly identifies an object by computing the similarity between prediction and ground truth objects across the entire video, instead of simply counting errors. AMOTA measures the average accuracy of the tracking system, while AMOTP computes the precision of the object locations.

B. Tracking Results on the CADC Dataset

In this section, we provide benchmarking results obtained on the Canadian Adverse Driving Conditions (CADC) Dataset [1]. CADC is a multi-modal dataset captured in snowy weather in Waterloo, Canada. It has 7K scans from a 32-channel Velodyne LiDAR sensor. In total, there are 75 sequences recorded on three different dates.

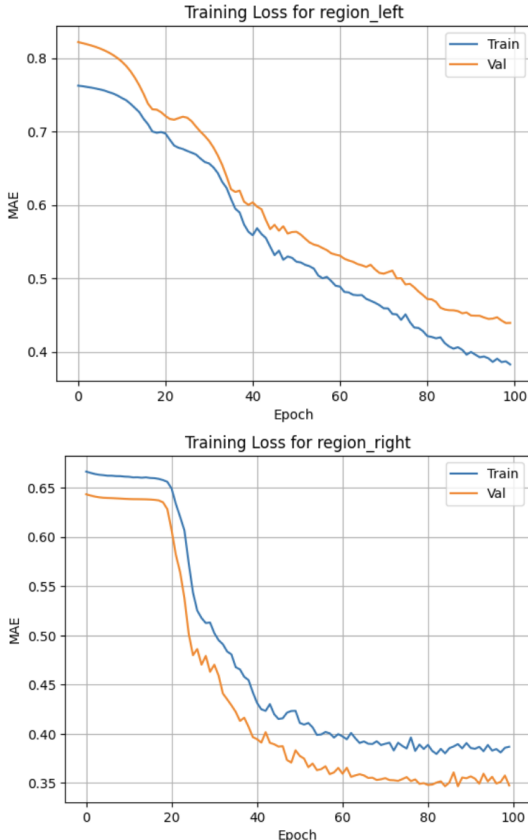


Fig. 1: Training loss curves (MAE) for left and right regions.

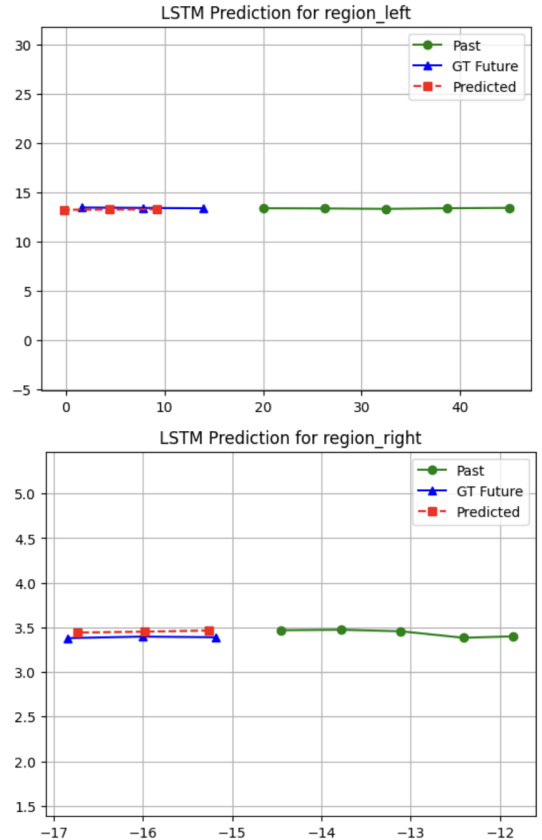


Fig. 2: Predictions against ground truth for both regions.

TABLE I: Tracking results on CADC (higher $\uparrow$ is better; lower $\downarrow$ is better).

Method	HOTA $\uparrow$	IDF1 $\uparrow$	AMOTA $\uparrow$	AMOTP $\downarrow$	IDS $\downarrow$
True	100	100	100	0.0	0
KF	43.2	63.2	55.8	83.7	490
PF	45.8	66.5	56.1	84.3	467
EKF	44.6	63.6	55.9	83.5	484
ByteTrack	53.0	70.0	60.0	86.2	209

Table I shows the tracking performances of Kalman Filter (KF), Extended Kalman Filter (EKF), Particle Filter (PF), and ByteTrack. Note that the “True” row in Table I shows ideal scores for reference. As the HOTA score, ByteTrack leads (53.0), while PF (45.8) outperforms KF (43.2) and EKF (44.6). ByteTrack again scores highest (70.0) IDF1, indicating best identity consistency. PF (66.5) improves over KF (63.2) and EKF (63.6). ByteTrack has the best AMOTA (60.0). KF and EKF have similar AMOTA scores (around 55.8–55.9) and slightly better AMOTP (83.5–83.7). PF’s AMOTA (56.1) and AMOTP (84.3) are in between. ByteTrack minimizes ID switches (209), with PF (467) also showing improvements over KF (490) and EKF (484).

Overall, as Table I shows, ByteTrack achieves the most robust tracking, excelling in detection and ID stability, while its slightly higher AMOTP reflects a trade-off in spatial precision. PF outperforms KF and EKF across most metrics, thanks to its probabilistic tracking approach. KF and EKF perform comparably in snowy conditions but lag behind learning-based ByteTrack and sampling-based PF in maintaining identities and handling occlusions.

The evidence for stating that Kalman filter-based approaches are least effective in snowy conditions comes directly from the metrics in Table I. KF and EKF have the lowest HOTA and IDF1 scores, showing they struggle more with tracking accuracy and identity consistency in snowy scenes.

C. Tracking Results on the nuScenes Dataset

We also present benchmarking results obtained from the nuScenes dataset [2]. nuScenes is a large-scale autonomous driving dataset collected in Boston and Singapore under mostly clean weather conditions during the day and night. The dataset contains 1,000 sequences.

Table II compares four tracking methods Kalman Filter (KF), Extended Kalman Filter (EKF), Particle Filter (PF), and ByteTrack. ByteTrack consistently achieved the best performance in HOTA, IDF1, AMOTA, and IDS, confirming its ability to accurately track and maintain object identities.

TABLE II: Tracking results on nuScenes (higher $\uparrow$ better; lower $\downarrow$ better).

Method	HOTA $\uparrow$	IDF1 $\uparrow$	AMOTA $\uparrow$	AMOTP $\downarrow$	IDS $\downarrow$
True	100	100	100	0.0	0
KF	49.90	56.88	54.63	53.33	63
EKF	55.50	56.90	54.63	52.94	67
PF	56.35	57.00	54.56	52.92	65
ByteTrack	63.01	58.40	66.01	52.95	59

PF showed slight improvements over KF and EKF in HOTA and IDF1, due to its probabilistic motion modeling. AMOTP scores were nearly the same for all methods, indicating similar precision in 3D position estimates.

D. LSTM Model Results

To assess the predictive performance of the LSTM-based trajectory forecasting model, four standard error metrics were used: Average Displacement Error (ADE), Final Displacement Error (FDE), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). The model was divided into two specialized variants: one for the right region ($y \in [-5, 7.5]$) and one for the left region ($y \in (7.5, 20]$), each trained separately to reflect differences in object motion patterns and speed.

The results in Table III indicate a clear performance difference between the two LSTM variants. The model trained on the right region outperforms the left-region model across all metrics. Specifically:

- **ADE:** The average displacement error is significantly lower for the right model (0.2818 vs. 0.7232), meaning the predicted trajectories stay closer to the ground truth.
- **FDE:** The final displacement error shows similar trends; the right model ends closer to the actual endpoint.
- **MAE and RMSE:** Both absolute and squared error measures are much lower in the right-region model, confirming its stability and lower variance.

These discrepancies reflect the difference in motion patterns across spatial regions. Vehicles in the left region tend to move faster and more variably, which makes accurate prediction more difficult for the LSTM-based trajectory prediction model. In contrast, the right region often includes slower or stationary traffic, yielding more predictable trajectories.

TABLE III: Prediction errors for region-specific LSTM models. $\downarrow$ means lower is better.

LSTM model	ADE $\downarrow$	FDE $\downarrow$	MAE $\downarrow$	RMSE $\downarrow$
LSTM model Right	0.2818	0.3968	0.1712	0.2603
LSTM model Left	0.7232	0.9702	0.4000	1.5710

For each unique object (identified via UUID), a time-ordered buffer of center positions is maintained. At every frame, up to 7 past positions are stored to provide temporal context. Only the most recent 5 frames are used as model input. Each frame contains the position coordinates and the corresponding velocity vector, where velocity is computed as

$$v_t = p_t - p_{t-1}$$

This yields an input tensor of shape (5, 4), representing 5 consecutive time steps, each with two position and two velocity components. The LSTM model is trained to predict the object’s future positions over the next 3 frames, providing short-term trajectory forecasting, see Fig. 2.

E. Qualitative Results

This section shows a visual comparison of tracking methods using a representative frame from each approach, shown in Figure 3. Specifically, see the detailed views for the ground-truth (Figure 3a), Kalman Filter (Figure 3b), Extended Kalman Filter (Figure 3c), Particle Filter (Figure 3d), ByteTrack (Figure 3e), and LSTM predictions (Figure 3f) algorithms.

Figure 4 shows ByteTrack’s predictions across different frames to illustrate tracking consistency and robustness.

Specifically, see the detailed views for Frame 1 (Figure 4a), Frame 5 (Figure 4b), Frame 20 (Figure 4c), and Frame 70 (Figure 4d).

V. DISCUSSION & LIMITATIONS

This study evaluated the performance of four different object-tracking algorithms: Kalman Filter (KF), Extended Kalman Filter (EKF), Particle Filter (PF), and ByteTrack in snowy weather conditions using LiDAR data. Each algorithm demonstrated specific strengths and limitations influenced by the complexity of environmental conditions, particularly snowy weather such as snow in CADC. The results indicate that snowy weather conditions significantly degrade the performance of tracking algorithms. For instance, metrics such as HOTA, IDf1, and AMOTA revealed reduced accuracy and consistency across all methods when tested with the CADC dataset. Where it was established that snowflakes and other environmental factors create noise and reduce sensor accuracy, presenting substantial challenges for LiDAR-based perception systems.

KF and EKF showed relatively stable performance, benefiting from their ability to handle linear and slightly non-linear motion models efficiently. However, their reliance on Gaussian noise assumptions and linearization limited their effectiveness in the complex and highly variable snow scenarios observed in the CADC dataset. While Kalman-based methods are computationally lightweight, they struggle under significant nonlinearities and erratic environmental noise.

The Particle Filter offered improved performance in harsh conditions by utilizing a set of hypotheses (particles) to represent state distributions. This capability allows PF to handle multi-modal uncertainties effectively, making it suitable for unpredictable movements and sporadic occlusions frequently encountered in snowy weather. This is consistent with previous research aspects mentioned in the line above, which emphasized the merits of PFs in the contexts of high uncertainty and non-linear dynamics.

The deep learning-based method ByteTrack reached the best overall scores on the CADC dataset and showed the strongest general applicability to low-confidence detection and occlusion. However, ByteTrack’s tracking is inherently designed for 2D box-level tracking by projecting 3D boxes into 2D space and associating them based on overlap. This flattening step, while effective for short-term tracking, does not fully resolve 3D tracking challenges in overlapping and complex 3D scenes.

A comparison between the nuScenes dataset (clean weather) and the CADC dataset (snowy weather) revealed stark contrasts. For instance, all algorithms performed substantially better on the nuScenes dataset, showing higher scores across HOTA, IDf1, and AMOTA metrics. This comparison clearly demonstrates that environmental conditions drastically influence tracking reliability. The performance gap underscores the necessity for tracking algorithms to be specifically optimized or trained on datasets reflecting snowy weather conditions if they are to maintain acceptable accuracy and reliability in real-world autonomous driving applications.

In summary, the study clearly indicates that deep learning-based methods like ByteTrack currently offer the best performance under challenging weather conditions, followed closely by particle filtering methods. Kalman-based filters, while computationally efficient, require enhancements to handle the complexity inherent in snowy weather scenarios effectively. Our LSTM-based model can predict the upcoming trajectories of the tracked objects as shown in Fig. 3f.

Failure Case Analysis: KF and EKF fail when the boxes are too close to each other and overlap within the rectangular threshold. The tracker cannot distinguish the objects because they are treated as one merged detection. The Particle Filter fails in the same way as the Kalman Filter. Since it also uses a rectangular threshold for measurement updates, it cannot separate close objects when their detections overlap. ByteTrack fails when boxes are close and overlap within the same expansion ratio. It can’t resolve the identities because it links tracks based on overlap, and overlapping boxes confuse the association step.

Run-time: KF and EKF are fast, usually taking 1 to 2 seconds. ByteTrack takes 2 to 4 seconds because it has more complex data association and extra tracking logic. Particle Filter with 1 million particles is very slow. It takes 15 to 20 minutes because it must evaluate each particle at each step. All tests were performed on a system equipped with an Apple M2 processor (8-core CPU, 10-core GPU) and 8 GB of unified memory.

REFERENCES

- [1] M. Pitropov, D. E. Garcia, J. Rebello, M. Smart, C. Wang, K. Czarnecki, and S. Waslander, “Canadian adverse driving conditions dataset,” *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 681–690, 2021.
- [2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nusenes: A multimodal dataset for autonomous driving,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.
- [3] X. Weng, J. Wang, D. Held, and K. Kitani, “3d multi-object tracking: A baseline and new evaluation metrics,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 359–10 366.
- [4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6526–6534.
- [5] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

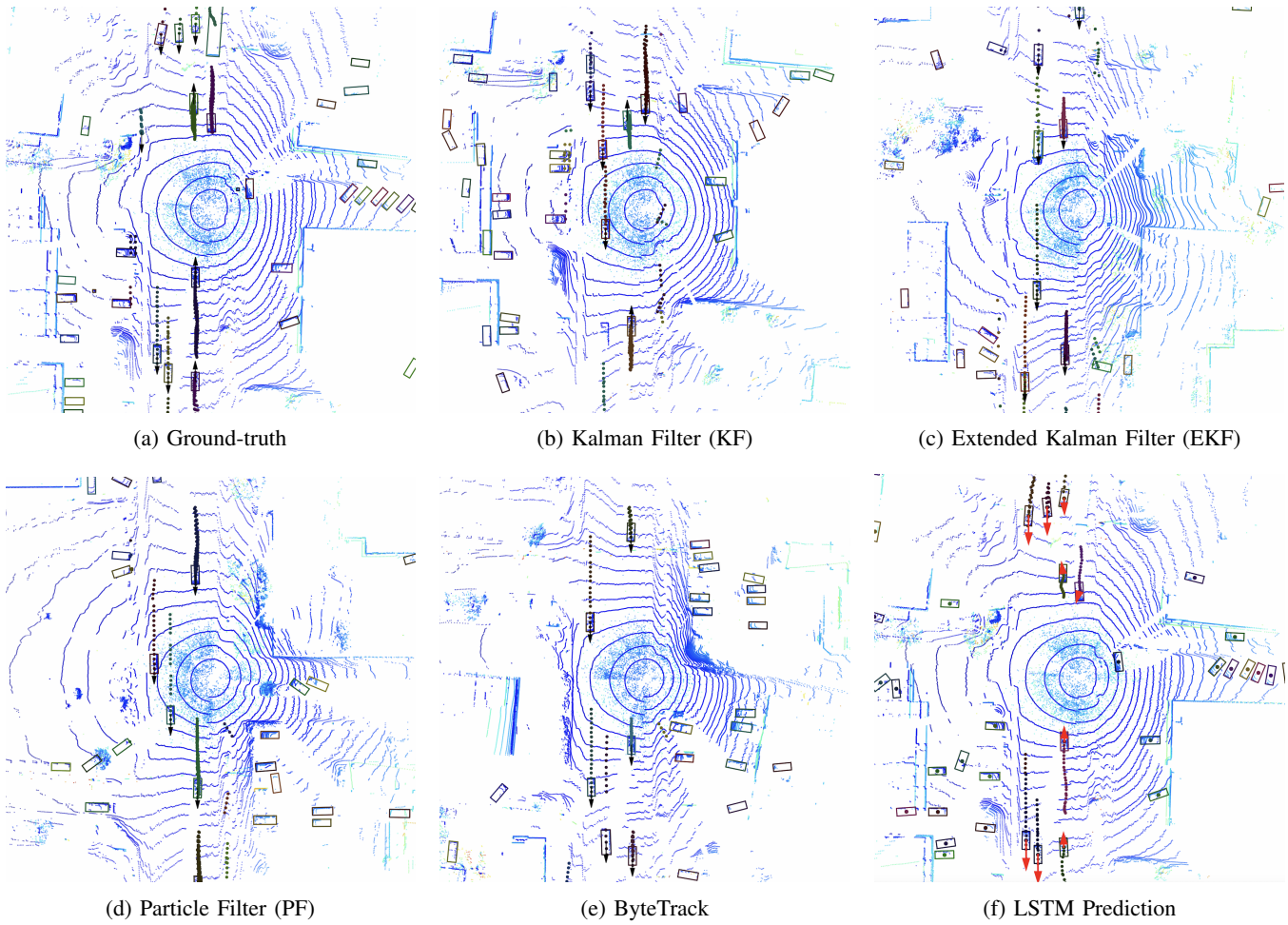


Fig. 3: Comparison of tracking methods on a representative frame. The figures show how each method tracks objects (trajectories and bounding boxes).

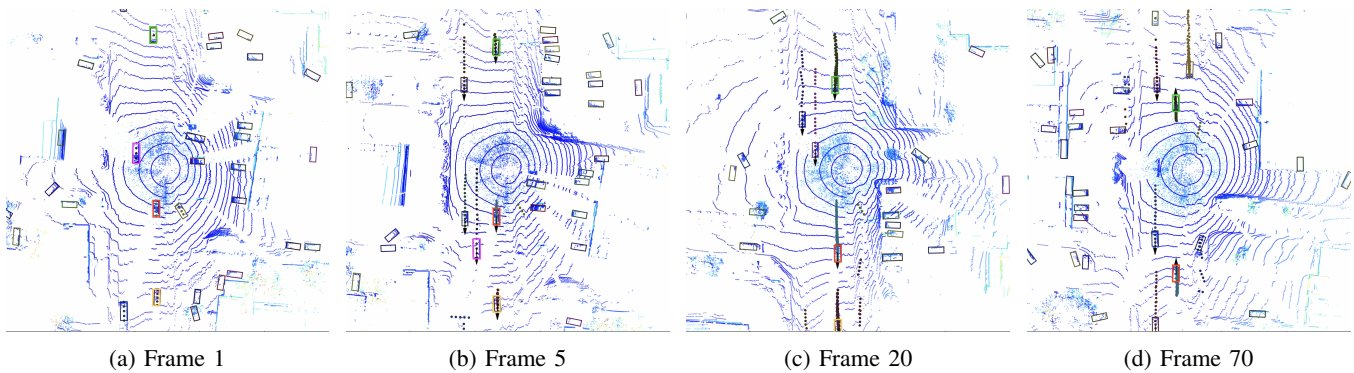


Fig. 4: ByteTrack's tracking predictions at different time steps: Frame 1, 5, 20, and 70.

- [6] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 689–12 697.
- [7] C. Wang, H. Huang, Y. Ji, B. Wang, and M. Yang, "Vehicle localization at an intersection using a traffic light map," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1432–1441, 2019.
- [8] G. Guo, J. Liu, and X. Sun, "A model decomposition kalman filter for enhanced localization of land vehicles," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 10 013–10 023, 2023.
- [9] A. Bertipaglia, M. Alirezaci, R. Happee, and B. Shyrokau, "An unscented kalman filter-informed neural network for vehicle sideslip angle estimation," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 9, pp. 12 731–12 746, 2024.
- [10] Y. Liu, X. Fan, C. Lv, J. Wu, L. Li, and D. Ding, "An innovative information fusion method with adaptive kalman filter for integrated ins/gps navigation of autonomous vehicles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 605–616, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0883327017304211>
- [11] Y. Jing, Q. Li, W. Ye, and G. Liu, "Development of a gnss/ins-based automatic navigation land levelling system," *Computers and Electronics in Agriculture*, vol. 213, p. 108187, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0168169923005756>
- [12] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.
- [13] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1, pp. 99–141, 2001. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370201000698>
- [14] G. G. Rigatos, "Extended kalman and particle filtering for sensor fusion in motion control of mobile robots," *Mathematics and Computers in Simulation*, vol. 81, no. 3, pp. 590–607, 2010, the Sixth IMACS Seminar on Monte Carlo Methods Applied Scientific Computing VII. Forward Numerical Grid Generation, Approximation and Simulation. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0378475410001515>
- [15] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," 2022. [Online]. Available: <https://arxiv.org/abs/2110.06864>
- [16] J. Liu, Y. Xie, Y. Zhang, and H. Li, "Vehicle flow detection and tracking based on an improved yolov8n and bytetrack framework," *World Electric Vehicle Journal*, vol. 16, no. 1, 2025. [Online]. Available: <https://www.mdpi.com/2032-6653/16/1/13>
- [17] Y. Gladiensyah Bihanda, C. Fatichah, and A. Yuniarti, "Multi-vehicle tracking and counting framework in average daily traffic survey using rt-detr and bytetrack," *IEEE Access*, vol. 12, pp. 121 723–121 737, 2024.
- [18] H. Zhang, R. Fang, S. Li, Q. Miao, X. Fan, J. Hu, and S. Chan, "Multi-camera multi-vehicle tracking guided by highway overlapping fovs," *Mathematics*, vol. 12, no. 10, 2024. [Online]. Available: <https://www.mdpi.com/2227-7390/12/10/1467>
- [19] S. Gao, Y. Li, and H. Luo, "Detecting thyroid nodules along with surrounding tissues and tracking nodules using motion prior in ultrasound videos," *Computerized Medical Imaging and Graphics*, vol. 117, p. 102439, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0895611124001162>
- [20] Y. Chang, J. Hu, and S. Xu, "Ote-slam: An object tracking enhanced visual slam system for dynamic environments," *Sensors*, vol. 23, no. 18, 2023. [Online]. Available: <https://www.mdpi.com/1424-8220/23/18/7921>
- [21] D. Gragnaniello, A. Greco, A. Saggese, M. Vento, and A. Vicinanza, "Benchmarking 2d multi-object detection and tracking algorithms in autonomous vehicle driving scenarios," *Sensors*, vol. 23, no. 8, 2023. [Online]. Available: <https://www.mdpi.com/1424-8220/23/8/4024>
- [22] G. Guo and S. Zhao, "3d multi-object tracking with adaptive cubature kalman filter for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 512–519, 2023.
- [23] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.
- [24] F. Ngeni, J. Mwakalonge, and S. Siuhi, "Solving traffic data occlusion problems in computer vision algorithms using deepsort and quantum computing," *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 11, no. 1, pp. 1–15, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2095756424000072>
- [25] T. Zhang, X. Chen, Y. Wang, Y. Wang, and H. Zhao, "Mutr3d: A multi-camera tracking framework via 3d-to-2d queries," 2022. [Online]. Available: <https://arxiv.org/abs/2205.00613>
- [26] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," 2017.
- [27] L. Hou, L. Xin, S. E. Li, B. Cheng, and W. Wang, "Interactive trajectory prediction of surrounding road users for autonomous driving using structural-lstm network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4615–4625, 2020.
- [28] Y. Fan, W. Zhang, W. Zhang, D. Zhang, and L. He, "A double-layer lstm model based on driving style and adaptive grid for intention-trajectory prediction," *Sensors*, vol. 25, no. 7, 2025. [Online]. Available: <https://www.mdpi.com/1424-8220/25/7/2059>
- [29] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," 2021. [Online]. Available: <https://arxiv.org/abs/2107.08430>
- [30] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," *International Journal of Comput Vision*, 2021.

VI. APPENDIX: IMPLEMENTATION

We here provide the pseudo code for KF, EKF, PF, and ByteTrack. All the source code, trained models, and the online supplementary information are available at the following URL: <https://github.com/Mahmoudalshaikh00/An-Experimental-Study-on-ObjectTracking.git>.

Algorithm 1 Kalman Filter (KF) Tracker

```

1: Initialize Kalman Filter with state  $[x, y, z, v_x, v_y, v_z]$ 
2: Define transition matrix  $F$ , measurement matrix  $H$ ,
   noise covariances  $R, Q$ 
3: Set initial position from 3D bounding box
4: for each new frame do
5:   if no tracks then
6:     Create new trackers for all detections
7:   else
8:     for each track and detection do
9:       Compute 2D distance  $(x, y)$ 
10:    end for
11:    Match tracks to detections using Hungarian Algorithm
12:    for each matched pair do
13:      Update Kalman Filter with new detection
14:      Predict next state
15:    end for
16:    for each unmatched track do
17:      Decrease lifespan counter; remove if expired
18:    end for
19:    for each unmatched detection do
20:      Create new trackers
21:    end for
22:  end if
23:  Optionally count 3D points in bounding box using
   Open3D
24:  Optionally visualize boxes
25: end for
26: Export final tracking results to JSON

```

Algorithm 2 Extended Kalman Filter (EKF) Tracker

```

1: Initialize EKF with 6D state  $[x, y, z, v_x, v_y, v_z]$ 
2: Define state transition model  $F$  (Jacobian), measurement
   function  $h(x)$  and Jacobian  $H$ 
3: Define covariances:  $P$  (state),  $Q$  (process),  $R$  (measure-
   ment)
4: for each new frame do
5:   if no active tracks then
6:     Create new tracker for each detection
7:   else
8:     Compute 2D cost matrix (Euclidean distance)
9:     Match using Hungarian Algorithm
10:    for each matched pair do
11:      Update EKF with detection using nonlinear
      update
12:      Predict next state with custom  $f(x, \Delta t)$  and
      Jacobian  $F$ 
13:      Update tracker state, velocity, position, meta-
      data
14:    end for
15:    for each unmatched track do
16:      Decrement countdown; remove if expired
17:    end for
18:    for each unmatched detection do
19:      Initialize new EKF tracker with unique UUID
20:    end for
21:  end if
22:  Optionally count LiDAR points using Open3D
23:  Optionally build 3D box geometry
24: end for
25: Export results to JSON

```

Algorithm 3 ByteTrack Tracker with Speed-Aware Expansion

```
1: Initialize ByteTrack with default parameters (thresholds,
   buffer size)
2: for each frame do
3:   Convert 3D boxes to 2D boxes by ignoring  $z$  and
   yaw
4:   if previous frame exists then
5:     for each new 2D box do
6:       Match to closest previous center to estimate
       motion
7:       Predict next position with velocity estimate
8:       Expand box slightly based on speed and ratio
9:     end for
10:  else
11:    Expand each 2D box using fixed ratio
12:  end if
13:  Run ByteTrack on adjusted 2D detections
14:  for each ByteTrack output do
15:    Match to nearest 3D detection using IoU
16:    if IoU > threshold then
17:      Assign original 3D box and track ID
18:    end if
19:  end for
20:  Save new 2D centers for next frame motion estimation
21:  Store matched 3D boxes with IDs
22: end for
23: Export final results to JSON
```

Algorithm 4 Particle Filter (PF) Tracker

```
1: Initialize large number of 3D particles per object (position,
   velocity, noise)
2: for each frame do
3:   Predict next position for particles using motion
   model and noise
4:   for each matched detection do
5:     Compute distance between particles and detection
6:     Update weights based on Gaussian likelihood
7:     Normalize weights; resample if needed
8:   end for
9:   Estimate object state as weighted average of particles
10:  Match detections to active tracks by 2D distance
11:  for each matched pair do
12:    Update PF, predict next state, update metadata
13:  end for
14:  for each unmatched detection do
15:    if within spatial threshold of lost track then
16:      Re-link
17:    else
18:      Create new PF tracker
19:    end if
20:  end for
21:  for each unmatched track do
22:    Decrease countdown; move to lost_tracks if expired
23:  end for
24:  Remove long-lost tracks after threshold
25: end for
26: Export all frames to JSON with predicted cuboids and
   UUIDs
```
