

UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITY OF MODENA AND REGGIO EMILIA

"Enzo Ferrari" Department of Engineering

Master's Degree in Artificial Intelligence Engineering (LM-32)

Radar-Only 3D Object Detection for Long-Range Automotive Perception: Architectures, Datasets and Embedded Deployment

Supervisor:
Prof. Simone Calderara

Co-supervisor:
Prof. Micaela Verucchi

Candidate:
Matteo Gombia

ACADEMIC YEAR 2025/2026

Abstract

Radar-Only 3D Object Detection for Long-Range Automotive Perception: Architectures, Datasets and Embedded Deployment

Robust environmental perception is a fundamental prerequisite for advanced autonomous driving functions, particularly demanding high-speed tasks such as overtake assist in performance vehicles and autonomous heavy-duty truck platooning. While LiDAR and camera systems are widely used, automotive radar offers unique advantages due to its extensive range, cost-effectiveness, and resilience to adverse weather and lighting conditions. However, performing accurate 3D object detection using radar-only data remains a significant challenge due to the sensor's inherent sparsity, lack of precise geometric resolution, and susceptibility to multi-path noise (ghost points). This thesis comprehensively evaluates state-of-the-art neural network architectures for radar-only 3D object detection, specifically analyzing grid-based methods (PointPillars, PillarNet), graph-based approaches (RadarGNN), and LiDAR-radar cross-modal feature hallucination.

To address the stark domain shift between existing public datasets—which often focus on high-speed highway scenarios—and dense urban environments, a custom 3,429-frame dataset was engineered utilizing a 3+1D radar configuration. The ground truth was generated through a robust, semi-automated labeling pipeline incorporating an ego-motion compensated Kalman Filter, the Hungarian algorithm for data association, and rigorous manual refinement.

Extensive experiments were conducted to baseline the architectures against standard public datasets (TruckScenes, nuScenes, View-of-Delft) before executing a multi-phase fine-tuning process to adapt the models to the heavily static, urban distribution of the custom dataset. Model accuracy was quantitatively evaluated using KITTI Average Precision and nuScenes evaluation metrics. Additionally, a multi-radar Cross-Potential preprocessing algorithm was investigated for spatial noise reduction, ultimately highlighting critical limitations regarding long-range detection preservation.

The experimental evaluation yields a definitive verdict: PillarNet significantly outperforms the other architectures. By leveraging a 2D sparse convolutional encoder and an anchor-free CenterHead, PillarNet achieved a peak Car Mean Average Precision (mAP) of 0.63 on the TruckScenes dataset, demonstrating superior spatial accuracy and overall detection precision. Furthermore, to

bridge the gap between theoretical accuracy and real-world deployment, the PointPillars architecture was subjected to rigorous post-training acceleration using ONNX and NVIDIA TensorRT. This optimization successfully slashed steady-state inference latency to sub-millisecond levels—achieving execution times as low as 0.75 ms-demonstrating real-time feasibility on desktop-class hardware while effectively bounding dynamic peak memory consumption. Ultimately, this work delivers a holistic framework for evaluating radar-based perception, from raw signal challenges and data engineering to hardware-accelerated inference.

Keywords: Automotive Radar, 3D Object Detection, Deep Learning, Domain Adaptation, Inference Optimization

To my family, for their endless love, patience, and support.

Contents

Abstract iii

1 Introduction: The perception challenge of radar-only 3D object detection 1

2 Theoretical Foundations of Automotive Radar and Point Clouds 3

2.1 Introduction to Autonomous Driving . . . 3

2.1.1 Challenges in Environmental Perception . . . 4

2.1.2 3D Object Detection and Sensor Suites . . . 4

2.1.3 Target Use Cases and System Requirements . . . 5

2.2 Sensor - Automotive Radar . . . 6

2.2.1 Physical sensor . . . 7

2.2.2 Features . . . 11

2.3 3D Object Detection on Point Clouds . . . 21

2.3.1 Point-Based Methods . . . 22

2.3.2 Grid-Based Methods: From Voxels to Pillars . . . 22

2.3.3 Graph-Based Methods . . . 23

2.3.4 Other approaches to the problem . . . 23

2.3.5 Summary of Evaluated Architectures and Chapter Conclusion . . . 24

3 Algorithmic Paradigms for Sparse 3D Detection 25

3.1 Point Pillars . . . 25

3.1.1 Architecture . . . 25

3.2 PillarNet Architecture . . . 32

3.2.1 Pillar Generation and Feature Encoding . . . 32

3.2.2 2D Sparse Convolutional Encoder . . . 33

3.2.3 Feature Fusion Neck . . . 33

3.2.4 CenterPoint Detection Head (CenterHead) . . . 33

3.3 LiDAR-Radar Cross-Modal Feature Augmentation . . . 35

3.3.1 Architecture . . . 35
3.3.2 Cross-Modal Loss Function . . . 37
3.4 RadarGNN . . . 37
3.5 Summary and Expected Outcomes . . . 39

4 Datasets: Overcoming the Range Limitations of Public Benchmarks 43
4.1 RadarScenes . . . 45
4.2 View-of-Delft (VoD) . . . 45
4.3 NuScenes . . . 48
4.4 MAN TruckScenes . . . 49
4.4.1 Annotation Sparsity and Filtering . . . 49
4.4.2 Z-Axis Misalignment and Multipath Artifacts . . . 51
4.5 Proprietary Custom Datasets . . . 53
4.5.1 2+1D Radar . . . 56
4.5.2 3+1D Radar . . . 58
4.5.3 SUSTechPoints . . . 66
4.6 Summary and Rationale for Custom Dataset Creation . . . 67

5 Experimental Evaluation: Radar Optimisation, Comparison, and Hardware Acceleration 71
5.1 Hardware and Software Infrastructure . . . 71
5.2 Implementation Frameworks . . . 71
5.2.1 Base Training Hyperparameters . . . 72
5.2.2 Dataset Split . . . 73
5.3 Evaluation Metrics for 3D Object Detection . . . 73
5.3.1 KITTI Evaluation Benchmark . . . 74
5.3.2 nuScenes Evaluation Benchmark . . . 75
5.3.3 Inference Efficiency Metrics . . . 76
5.4 Preprocessing: Cross-Potential Point Cloud Evaluation . . . 77
5.4.1 Theoretical Background: DBSCAN and Cross-Potential . . . 77
5.4.2 Experimental Implementation on the Custom Setup . . . 79

5.4.3 Results and Rejection of the Methodology 79
5.5 Ablation Study for Tuning 81
5.5.1 Batch Size 81
5.5.2 Maximum Voxel Limits 82
5.5.3 Voxel Size 85
5.5.4 Frame Accumulation 88
5.5.5 Doppler-driven Point Compensation 90
5.6 Comparison: PointPillar vs CenterHead 91
5.6.1 Evaluation on MAN TruckScenes 91
5.6.2 Evaluation on the Proprietary Dataset 93
5.6.3 Optimization in ONNX and TRT 95
5.7 Appendix: Experiments on LiDAR-Radar Cross-Modal Feature Augmentation . . . 96
5.8 Appendix: Experiments on RadarGNN 98

6 Conclusion and Future Horizons **101**
6.1 Future Works 102

Acknowledgements **103**

References **104**

List of Figures

2.1 Continental ARS548RDI automotive radar 6
2.2 Frequency shift due to the Doppler effect caused by a moving target [13] 8
2.3 Example of ghost points. Different types of multi-path reflections are highlighted alongside the real object [7] 10
2.4 Ghost points caused by a radar misconfiguration 12
2.5 Difference between LiDAR and radar in spatiotemporal consistency in consecutive scans [23] 17
2.6 Comparison point clouds between LiDAR and Radar 18
2.7 2D Bird-Eye-View of radar point clouds with aggregation 19
2.8 Comparison among three different temporal aggregation methods: (1) no aggregation, (2) standard ego-motion compensation, and (3) Doppler-driven temporal aggregation [24] 20

3.1 Architecture overview of PointPillars [26] 26
3.2 SSD anchors on a 4x4 feature map [31] 29
3.3 PillarNet architecture overview [30] 32
3.4 Cross-modal feature augmentation overview [33] 36
3.5 Overview of RadarGNN [25] 38

4.1 Missing label on 2 consecutive frames 47
4.2 Comparison of the BEV of a TruckScenes frame with the filtered stripe of vehicles highlighted in red and the BEV of the isolated stripe 52
4.3 Side view of the isolated stripe, highlighting the severe vertical misalignment of radar points relative to ground truth bounding boxes 52
4.4 Comparison of the timestamps of the different radar frames using a closest frame association. The center radar is used as reference 54
4.5 Comparison of the timestamps of the different radar frames using a one-to-one frame association. The center radar is used as reference 55

4.6 Comparison of the distributions of the RCS across the dataset 57
4.7 Comparison of the distributions of the RCS between the TruckScenes and custom 3+1D datasets 59
4.8 Example of the manually annotated dataset with radar partially occluded while stationary behind a vehicle 60
4.9 Distribution of car annotations over longitudinal distance (x) 64
4.10 Distribution of car annotations over lateral distance (y) 65
4.11 Distribution of car annotations over absolute Euclidean range 66
4.12 Distribution of car annotations over rotation (yaw angle) 67
4.13 Multi-view of the interpolation of a single car across 10 frames 68
4.14 Distribution of range of car annotations across different datasets 69

5.1 Cross-Potential between the pairs of radars 80
5.2 Comparison of the global point cloud before and after Cross-Potential thresholding 80
5.3 Training progression of the Mean Average Precision (mAP) over 30 epochs for the PointPillars architecture. The lighter line displays the raw mAP measured epoch by epoch, while the darker line represents a smoothed version to clearly highlight the underlying trend 83

1. Introduction: The perception challenge of radar-only 3D object detection

Robust environmental perception remains the cornerstone of autonomous driving, demanding sensory systems that can flawlessly interpret complex surroundings under unforgiving conditions. While LiDAR and optical cameras dominate current perception pipelines, they harbor critical vulnerabilities to extreme weather, airborne particulates, and dynamic lighting changes. Automotive radar emerges as an indispensable, cost-effective alternative, offering unmatched environmental resilience, extended detection ranges, and the unique ability to measure instantaneous radial velocity via the Doppler effect. However, translating raw radar signals into precise 3D object detection introduces severe algorithmic hurdles. Radar point clouds are inherently sparse, suffer from coarse vertical resolution, and are highly susceptible to multipath reflections or "ghost points," making geometric bounding box regression an exceptionally difficult task for traditional algorithms.

This thesis bridges the gap between raw signal limitations and real-world deployment by rigorously evaluating state-of-the-art neural network architectures for radar-only 3D object detection. The research is specifically driven by demanding, high-speed use cases—such as overtake assist in performance vehicles and autonomous heavy-duty truck platooning—which strictly require reliable perception horizons exceeding 150 meters. To identify the most viable solutions, the study benchmarks established grid-based frameworks like PointPillars and the highly efficient, sparse-convolutional PillarNet, alongside specialized paradigms such as the graph-based RadarGNN and LiDAR-radar cross-modal feature hallucination.

A foundational component of this work is overcoming the critical data limitations found in academic literature. Because popular public benchmarks often artificially crop detection ranges or lack full 4D data, the initial evaluations leverage the expansive, highway-biased MAN TruckScenes dataset. To address the stark domain shift between high-speed environments and dense, static urban clutter, a custom 3,429-frame 3+1D radar dataset was systematically engineered. Annotated through a semi-automated pipeline utilizing an ego-motion compensated Kalman Filter and the Hungarian algorithm, this proprietary dataset enables rigorous domain adaptation and fine-tuning experiments.

Ultimately, theoretical accuracy must survive the constraints of embedded vehicle hardware. By subjecting these neural architectures to aggressive post-training acceleration using ONNX and NVIDIA TensorRT, this research delivers a comprehensive analysis of the critical trade-offs between bounding box precision, steady-state inference latency, and dynamic peak memory consumption for automotive radar.

The remainder of this thesis is organized as follows to systematically address these challenges. Chapter 2 establishes the theoretical foundation by detailing the physical principles, inherent limitations, and distinct data characteristics of automotive radar, alongside a comprehensive review of state-of-the-art 3D object detection methodologies applied to point clouds. Chapter 3 provides a technical analysis of the selected neural network architectures, specifically outlining the structural adaptations required to process sparse radar data using grid-based models, cross-modal feature hallucination frameworks, and graph neural networks. Chapter 4 critically evaluates existing public datasets against the strict requirements of high-speed perception and details the collection, filtering, and semi-automated annotation pipeline of the proprietary 3+1D urban dataset. Finally, Chapter 5 presents the comprehensive experimental setup and results, encompassing baseline evaluations, hyperparameter ablation studies, targeted domain adaptation comparisons, and a rigorous analysis of post-training hardware acceleration to benchmark real-time inference viability.

2. Theoretical Foundations of Automotive Radar and Point Clouds

This chapter provides the theoretical background required for the subsequent experimental section, structured to guide the reader logically from the broad context of autonomous driving down to the specific algorithmic solutions evaluated in this work.

First, the chapter introduces the general domain of environmental perception and defines the target use cases, establishing the strict system requirements that motivate the necessity of a radar-centric approach. Following this top-down progression, the second section deeply analyzes the physical characteristics and specific data features of the automotive radar. By understanding the physics of the sensor, it becomes clear why radar point clouds possess unique advantages but also inherent limitations, such as sparsity and ghost points. Finally, having established the complex nature of radar data, the third section reviews the state-of-the-art Deep Learning architectures for 3D object detection on point clouds, detailing how different methodologies attempt to process this specific type of unstructured information. This sequential order ensures that the algorithmic choices and experimental setups discussed later in the thesis are fully contextualized within the sensor's physical constraints and the demanding requirements of the target applications.

2.1 Introduction to Autonomous Driving

Automated Driving Systems promise a safe, comfortable, and highly efficient driving experience by reducing human error on the roads. The Society of Automotive Engineers (SAE) classifies driving automation into five distinct levels, starting from primitive driver assistance systems at Level 1 to fully automated, unsupervised driving across all conditions at Level 5 [1]. As vehicles progress toward these higher SAE levels, the system assumes full responsibility for the dynamic driving task, making sensor and architectural redundancy strictly necessary to guarantee safety in the event of component failures [1]. Consequently, to manage this complexity and achieve such robust automation, modern vehicle architectures divide the massive task of autonomous driving into

specific, interconnected subcategories: perception, localization, mapping, planning, and vehicle control [1].

2.1.1 Challenges in Environmental Perception

Having established the modular architecture of Automated Driving Systems, it is clear that the perception system serves as the foundational layer of the autonomous pipeline; once the vehicle has perceived its environment, the essential stages of planning and control can commence. In recent years, the integration of deep learning has revolutionized this module, mimicking human neural networks to enable rapid advancements in perceiving complex, unstructured environments [2]. However, designing a robust perception system remains one of the most difficult challenges in the field, due to the complex scenario [2]. Autonomous vehicles must navigate highly dynamic traffic while overcoming severe real-world challenges, including dataset uncertainty, unexpected weather conditions, and inherent sensor limitations [2].

2.1.2 3D Object Detection and Sensor Suites

To overcome these severe real-world challenges within the broader scope of environmental perception, 3D object detection acts as an indispensable component. This task intelligently predicts the exact locations, physical sizes, and semantic categories of critical 3D objects—such as cars, pedestrians, and cyclists—in the real-world coordinate system. Unlike traditional 2D detection, the geometric information generated by 3D object detection allows the vehicle to accurately measure distances and avoid collisions [3].

To accomplish this, vehicles rely on a diverse suite of sensory inputs. Cameras are cheap and provide rich semantic information, but they struggle to directly estimate 3D structural depth and are highly vulnerable to extreme weather and lighting conditions [3]. Different camera types can also be used such as thermal cameras, event cameras and omnidirectional cameras; however while they capture different types of information, they share the same limitations of RGB cameras [1]. Conversely, while sensors like LiDAR provide highly accurate depth measurements, they can be negatively affected by adverse environmental factors. To compensate for the respective weaknesses of cameras and LiDAR, automotive radar has emerged as a critical modality [1],[3] . However, a

significant gap remains in the radar data processing pipeline, as current approaches often still rely on classical clustering and tracking algorithms [4].

2.1.3 Target Use Cases and System Requirements

Having identified automotive radar as a crucial sensor for overcoming environmental limitations, it is now necessary to define the specific target use cases that dictate the technological requirements of the perception system and motivate the focus of this thesis. This work specifically focuses on two primary and highly demanding applications.

The first use case is an overtake assist system designed for high-performance sports cars. In this scenario, the system aims to monitor the environment beyond the immediate preceding vehicle, anticipating upcoming traffic dynamics to inform the driver of what to expect before and during a maneuver. This application demands extended detection range to identify distant oncoming traffic, extremely low processing latency to handle high-speed dynamic scenes safely, and precise radial velocity estimation to accurately judge the relative speeds of vehicles involved in the overtake.

The second use case involves heavy-duty trucks, focusing on both autonomous platooning on highways and navigation in harsh, off-road environments such as quarries. Platooning requires ultra-reliable distance keeping and instantaneous relative velocity tracking to maintain tight, safe formations between trucks. Furthermore, operating in a quarry introduces massive amounts of suspended sand and dust, which create severe visibility issues.

Both of these use cases impose strict, unforgiving system requirements: extended range, low latency, accurate radial velocity extraction, and extreme environmental robustness. In such challenging scenarios—where dense sand, dust, or dynamic occlusions instantly compromise or blind LiDARs and optical cameras—a radar-only setup becomes strictly necessary. Radar's unique ability to penetrate airborne particulates and directly measure instantaneous Doppler velocity makes it the only sensor capable of consistently fulfilling these specific operational requirements. Consequently, these target applications and their derived requirements provide the primary motivation for the radar-centric approach investigated throughout this thesis.



Figure 2.1: Continental ARS548RDI automotive radar

2.2 Sensor - Automotive Radar

Given the strict requirements established by these target use cases, particularly the need for extreme environmental robustness, automotive radar emerges as the primary focus of this research. As introduced in the previous section, radar plays a major role in modern vehicle architectures due to its robust measurements, high range, wide Field of View (FoV), low cost, and small form factor [5]. An example of an automotive radar is shown in Figure 2.1.

Due to the propagation properties of electromagnetic waves, radar can operate during the day or at night, as well as in foggy, snowy, or overcast conditions [6]. Unaffected by poor lighting, radar represents a more robust solution compared to other sensors used for road safety [6]. However, these remarkable advantages in environmental robustness are accompanied by inherent sensor limitations. Primarily, radar systems suffer from severe data sparsity, lower spatial resolution compared to LiDAR, and a high susceptibility to multi-path signal reflections (known as ghost points), all of which pose significant challenges for subsequent data processing and will be detailed later in this section [7].

To understand the growing market for this sensor, we can look at data published in a document by the International Telecommunication Union (ITU), which projects that by 2030, automotive radar penetration will reach approximately 65% in Europe and 50% in the United States [8].

2.2.1 Physical sensor

To fully appreciate why radar can operate effectively under the aforementioned adverse conditions, it is essential to examine the underlying physical sensor. RADAR stands for ***RA***dio ***D***etection ***A***nd ***R***anging [6]. It is a sensor technology that uses radio waves to precisely measure the distance to objects within its range, consisting of transmitting radio antennas and receiving ones [6]. The typical frequency range used today is between 24 and 79 GHz, depending on the application and the radar's maximum range [6]. When the transmitted wave encounters an obstacle, part of its energy is reflected by the object's surface (forming an echo) and detected by the receiving antenna [6]. The time between transmission and reception, combined with the known velocity of radio waves, allows the system to precisely determine the distance to the obstacle [6].

Automotive radars can be divided into three main categories: Short-Range Radar (SRR), Medium-Range Radar (MRR), and Long-Range Radar (LRR). SRR has a range of up to 30 meters, MRR up to 90 meters, and LRR can exceed 200 meters. While SRR provides a solution for parking assistance and MRR for peripheral detection, LRR is used to detect distant vehicles relative to the ego vehicle [5],[6].

CFAR

Once the physical sensor has received the reflected radio waves across these various ranges, the raw analog signal must be processed using a post-processing algorithm to extract a usable point cloud. Despite the publication of many studies proposing alternatives, CFAR (Constant False Alarm Rate) remains the standard and most common algorithm [9]. Evaluated alternatives include Convolutional Neural Networks (CNNs) [9], Monte Carlo methods, and modified versions of CFAR that calculate thresholds differently. Each method targets specific algorithmic deficits and presents unique advantages and disadvantages depending on the use case [9],[10].

First, the raw signal is processed by a Fast Fourier Transform (FFT) to convert the time-domain signal (based on the wave's round-trip time) into the frequency domain. Then, the CFAR algorithm used in modern radars sets a dynamic threshold for the Signal-to-Noise Ratio (SNR) to distinguish real targets from background noise. Crucially, as its name implies, this algorithm dynamically adapts the detection threshold to the local interference level to ensure that the probability of a

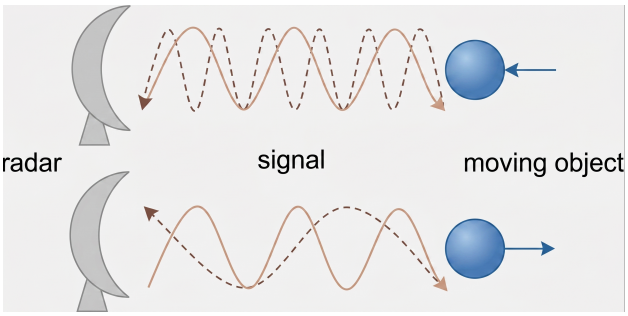


Figure 2.2: Frequency shift due to the Doppler effect caused by a moving target [13]

false alarm—that is, incorrectly declaring the presence of a target when only noise is actually there—remains strictly constant. If the false alarm is the noise overshooting the threshold, on the other hand, the other risk of the algorithm is the detection loss, which occurs when the target is not detected because the threshold is too high. The CFAR algorithm is designed to balance these two risks, ensuring that the system maintains a consistent false alarm rate while maximizing the probability of correctly detecting real targets. For example, this is typically achieved by comparing the signal of a specific cell under test to its surrounding adjacent cells; by estimating the local noise floor from this neighborhood, the algorithm establishes a dynamic threshold capable of filtering out varying background clutter while maintaining the desired false alarm probability. Once the target is isolated, the signal can be converted into multiple points with 2D or 3D coordinates to generate a point cloud, which is commonly used in automotive applications.[10][11]

Doppler effect

While processing the signal through CFAR provides the azimuth, range, and elevation to form 2D or 3D spatial coordinates, the most characteristic capability of radar lies in detecting the radial velocity of a target along the Line-of-Sight (LoS) [12]. To measure this, radar exploits the Doppler effect, a well-known physical phenomenon that causes a frequency shift in the reflected wave depending on whether the target is moving towards or away from the sensor [13]. As shown in Figure 2.2, if the target is approaching, the frequency of the reflected signal increases; if the target is moving away from the ego vehicle, the frequency decreases.

The Doppler frequency (the difference between the emitted and received frequencies) is calculated using the following formula, where Δf is the Doppler frequency, Δv is the relative velocity of

the target compared to the ego vehicle, c is the speed of light, and f_s is the source frequency [13]:

$$\Delta f = 2 \cdot \frac{\Delta v}{c} \cdot f_s \tag{2.1}$$

Therefore, if the source frequency is known and the frequency of the received signal is measured, we can easily calculate the Doppler frequency and determine the difference in speed as follows:

$$\Delta v = \frac{\Delta f \cdot c}{2 \cdot f_s} \tag{2.2}$$

It is important to note that, as the formula shows, the velocity measured by the radar is not the target's absolute velocity, but rather its relative speed compared to the ego vehicle. Furthermore, because it is a radial velocity measured along the LoS, if a target moves along a circular path with the ego vehicle at the center, the radar will return a velocity of zero.

Consequently, radars are often referred to as 2+1D or 3+1D (4D) sensors because they measure azimuth, range, velocity, and—in some models—elevation [14]. However, conventional automotive radars inherently possess a low resolution in elevation due to hardware limitations in their antenna arrays, which often restricts them to returning a mostly two-dimensional point cloud [14],[15]. In fact, this relatively poor angular resolution compared to optical sensors like LiDAR directly results in the spatial uncertainty and inaccuracy visible along the Z-axis in 4D radar data [14].

It should also be noted that CFAR is typically applied across all dimensions of the radar data, including Doppler information. As a result, detections from non-stationary objects are more likely to pass the CFAR threshold and appear as points in the final point cloud [7].

Ghost points

Although the radar's large wavelength enables the reliable spatial and velocity measurements discussed above, this resilience to environmental conditions comes with a notable downside: signal reflection off multiple surfaces [7]. This phenomenon, known as multi-path detection or ghost points, is common around highly reflective surfaces such as metal doors, guardrails, and other vehicles, as shown in Figure 2.3 [7].

These points can lead to false-positive detections, which can be problematic for downstream object detection tasks [7]. On the other hand, multi-path detections can sometimes be useful for identifying objects that would otherwise remain unseen [7]. For example, a vehicle closely following

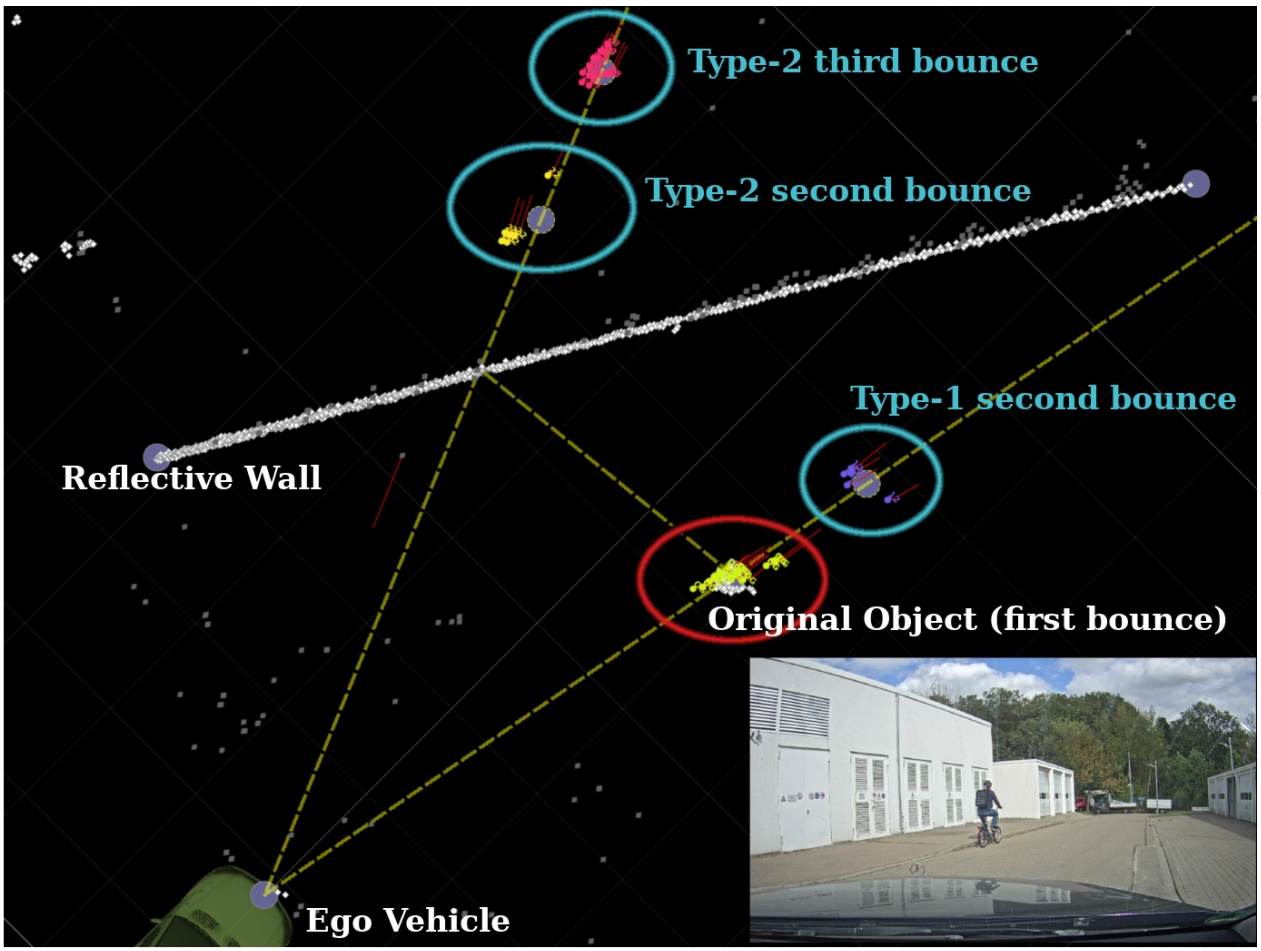


Figure 2.3: Example of ghost points. Different types of multi-path reflections are highlighted alongside the real object [7]

another car might be detected due to multi-path propagation bouncing along the road surface. In such cases, the target's estimated location closely matches its real position due to the shallow angle of these reflections. Furthermore, these detections can be exploited for non-Line-of-Sight tracking, such as detecting a target hidden around a corner [7].

In the literature, various methods have been proposed to address this issue. Some approaches use machine learning to filter out ghost points, while others propose segmentation neural networks to actively detect them [7].

Figure 2.4 shows a frame recorded during a radar misconfiguration, which led to the appearance of multiple ghost points. In front of the ego vehicle, there is a truck with a trailer (highlighted in red) and a guardrail along the street on the right. The color of the points represents their absolute radial velocity. The area where the ghost points appeared is highlighted by two blue dashed lines; in reality, this area is a parking spot with no moving cars. The sensor misconfiguration, combined with highly reflective surfaces like the back of the trailer and the guardrail, generated misleading detections in this frame.

2.2.2 Features

Having established the physical principles and inherent anomalies (such as ghost points) of the sensor, we now turn to the specific types of data provided by radars and how they can be processed to extract additional features for each point in the resulting point cloud.

Coordinates

As a baseline feature derived directly from the physical measurements discussed earlier, we have already seen how the spatial coordinates of the points (2D or 3D) are measured. These components serve as standard input features for neural networks, similar to the processing pipelines commonly used for LiDAR data.

Radar cross section (RCS)

Beyond the basic spatial coordinates, the radar extracts another fundamental piece of information by measuring the returned power. The relationship between the power density of an isotropic antenna

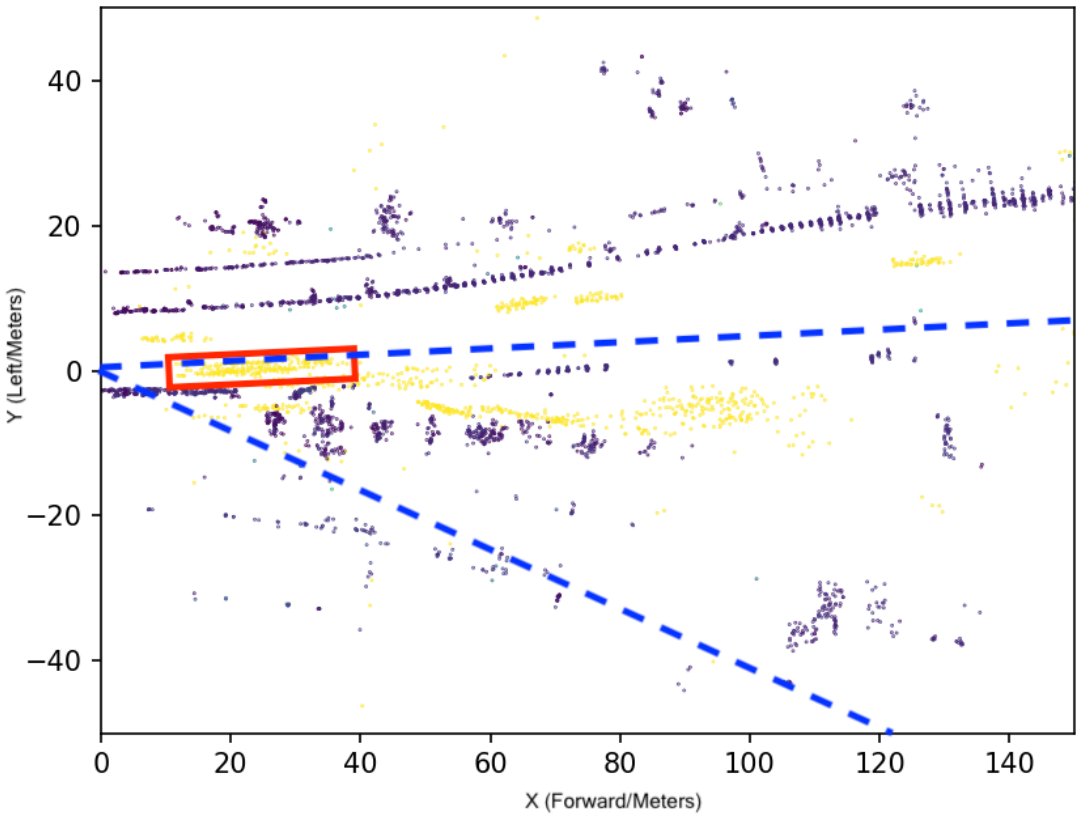


Figure 2.4: Ghost points caused by a radar misconfiguration

(which radiates uniformly in all directions) and the distance to the target can be formulated as follows, where P_t is the transmitted power and R is the range (radius) [12]:

$$\text{Power density from isotropic antenna} = \frac{P_t}{4\pi R^2} \tag{2.3}$$

However, radars use directive antennas that focus power in a specific direction rather than irradiating the entire surrounding area. Therefore, we introduce the gain parameter G, which is the ratio between the power directed specifically at the target and the power that radiates elsewhere [12]. Using the following equation, we can determine the actual power reaching the target:

$$\text{Power density from directive antenna} = \frac{P_t G}{4\pi R^2} \tag{2.4}$$

Every target reflects a different amount of the incident power back toward the radar. This depends heavily on the target's shape and material. The measure of the incident power intercepted by the target and scattered back toward the radar is denoted as the Radar Cross Section (RCS). It is important to note that while RCS describes the target's ability to reflect signals independently of the radar's transmitted power or antenna gain, it is not purely an intrinsic property of the object; rather, it heavily depends on the radar's operating frequency, the polarization of the electromagnetic wave, and the target's aspect angle relative to the sensor [16]. We can determine the power received back based on the RCS, σ [12]:

$$\text{Power density of echo signal at radar} = \frac{P_t G}{4\pi R^2} \frac{\sigma}{4\pi R^2} \tag{2.5}$$

To determine the actual power read by the radar, P_r, we must account for the effective area of the receiving antenna, A_e [12]:

$$P_r = \frac{P_t G}{4\pi R^2} \frac{\sigma}{4\pi R^2} A_e = \frac{P_t G A_e \sigma}{(4\pi)^2 R^4} \tag{2.6}$$

We can then isolate the RCS as a function of the other variables:

$$\sigma = \frac{P_r R^4 (4\pi)^2}{P_t G A_e} \tag{2.7}$$

From this formula, we can see that P_t, G, and A_e are internal radar parameters provided by the manufacturer. The received power P_r and the range R are measured by the sensor. Therefore, the

Class	Average RCS
Static objects	4.81 dBsm
Pedestrian	1.74 dBsm
Bike	3.46 dBsm
Normal vehicles	8.98 dBsm
Big Vehicles	19.97 dBsm

Table 2.1: Mean RCS for each macro-class in the NuScenes dataset [17]

radar can output the RCS, which provides critical information about the object's physical properties. This is a highly valuable feature for neural networks when classifying different objects.

For example, by grouping similar classes from the NuScenes dataset (which will be detailed later) and averaging their RCS values, clear distinctions emerge. As shown in Table 2.1, different object types have noticeably different mean RCS values. The macro-classes are composed of:

1. **Static Objects:** barrier, debris, pushable/pullable, bicycle rack, traffic cone
2. **Pedestrian:** animal, child, adult, construction worker, police officer
3. **Bike:** bicycle, motorcycle, stroller, wheelchair, personal mobility device
4. **Normal vehicles:** car, ambulance, police vehicle
5. **Big Vehicles:** bendy bus, rigid bus, construction vehicle, trailer, truck [17]

Radial velocity

Along with spatial and reflectivity features, and as previously introduced in Section 2.2.1, radar is uniquely able to measure the relative radial speed, which has been proved to be a highly valuable feature, increasing the efficacy of classification and orientation estimation [18]. However, this feature depends on the ego-motion, so it measures only the difference between the target and the ego on the LoS. To remove this dependency and to ensure this feature accurately reflects the absolute state of the observed world, it is possible to calculate the absolute radial speed of the object through the odometry of the car. As many autonomous or semi-autonomous vehicles are equipped with

multiple sensors, it is common today for these systems to inherently estimate the ego-vehicle's odometry. Assuming the vehicle's odometry is known, we can compute the absolute radial velocity to remove the bias introduced by the ego-motion.

First, radar data and odometry are usually not synchronized because the odometry is usually at a higher frequency than the radar's operating frequency. To overcome the possible issue of using data that are misaligned with the radar data, linear interpolation is performed between the discrete odometry samples adjacent to the radar's timestamp. Due to the high-frequency odometry data and the short interval between the data, we can assume a constant acceleration over the interval [19]. Therefore, if the radar timestamp t_{radar}, velocities ($v_{x_{0|1}}$, $v_{y_{0|1}}$, $v_{z_{0|1}}$, $v_{\omega_{0|1}}$) and timestamps ($t_{odom_{-1}}$, t_{odom_0}) are known, then it is possible to calculate through the 1D linear interpolation the ego velocity at the radar timestamp:

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \tag{2.8}$$

which, if we consider y as the velocity and x as the time and isolate the interpolated velocity, becomes:

$$v_{x_{radar}} = v_{x_0} + (t_{radar} - t_{odom_0}) \cdot \frac{v_{x_1} - v_{x_0}}{t_{odom_1} - t_{odom_0}} \tag{2.9}$$

The same concept is applied for all the other velocities on the other axis.

The vehicle's odometry is typically referenced to the location of the estimating sensors or the center of gravity. However, radars are usually located on the front bumpers of a car, so we first need to calculate the odometry of the sensor itself. To do so, we can use the formula of the rigid body, assuming chassis deformations are negligible, as accounting for them would require a complex dynamic model. If we call the 3D position of the sensor B and the point of reference for odometry A, which we can consider as the origin, we can call the displacement from B to A $\overrightarrow{r}_{B/A}$ which can be called $\overrightarrow{r}_B$ as we consider the point A as the origin. We call the angular velocity ω, which is perpendicular to the plane where the rigid body rotates (so it can be seen as the yaw in a vehicle, where roll and pitch are mostly irrelevant for non-racing vehicles) and it has been proven that the angular velocity is independent of the choice of the points on the rigid body. So, we can define the rotational velocity at the sensor position as the cross product of these vectors [20]:

$$\overrightarrow{v}_{B,rot} = \overrightarrow{\omega} \times \overrightarrow{r}_B \tag{2.10}$$

In general, the total motion of a point in a rigid body is characterized by the sum of a translation and a rotational component. If we take into account the translational component, given by the shift in odometry, we obtain the total velocity at the sensor B [20].

$$\overrightarrow{v}_B = \overrightarrow{v}_A + \overrightarrow{\omega} \times \overrightarrow{r}_B \tag{2.11}$$

Note that if we consider $\overrightarrow{v}_A$ and $\overrightarrow{r}_B$ to be oriented as the odometry reference system, so $\overrightarrow{v}_B$ will be oriented in the same reference system. So, if the radar is mounted with a different yaw than the odometry reference, we need to apply a rotation to the vector $\overrightarrow{v}_B$ to bring it into the radar frame and in order to be able to use the ego-velocity for the ego-motion compensation.

Once the velocity vector is obtained in the radar's frame and at the radar's position, for each point we need to project the velocity to the LoS vector $\hat{u}$, which in 2D is:

$$\hat{u} = \begin{bmatrix} cos(\alpha) \\ sin(\alpha) \end{bmatrix} \tag{2.12}$$

The dot product between the velocity vector and the LoS vector gives us the radial velocity of the ego for each point [21].

$$v_{ego,rad} = \overrightarrow{v}_B \cdot \hat{u} \tag{2.13}$$

Finally, to obtain the absolute radial velocity for each point, we need to subtract the ego radial velocity from the measured relative radial velocity of each target.

$$v_{target,absolute} = v_{target,relative} - v_{ego,rad} \tag{2.14}$$

Some datasets, such as NuScenes dataset, process these data and divide them into coordinate components, creating v_comp_x and v_comp_y by multiplying again the absolute radial velocity $v_{target,absolute}$ and the LoS vector $\hat{u}$ [22].

Time for point cloud accumulation

While the aforementioned features (coordinates, RCS, and absolute velocity) provide rich point-level information, the overall point cloud suffers from sparsity and temporal inconsistency. The

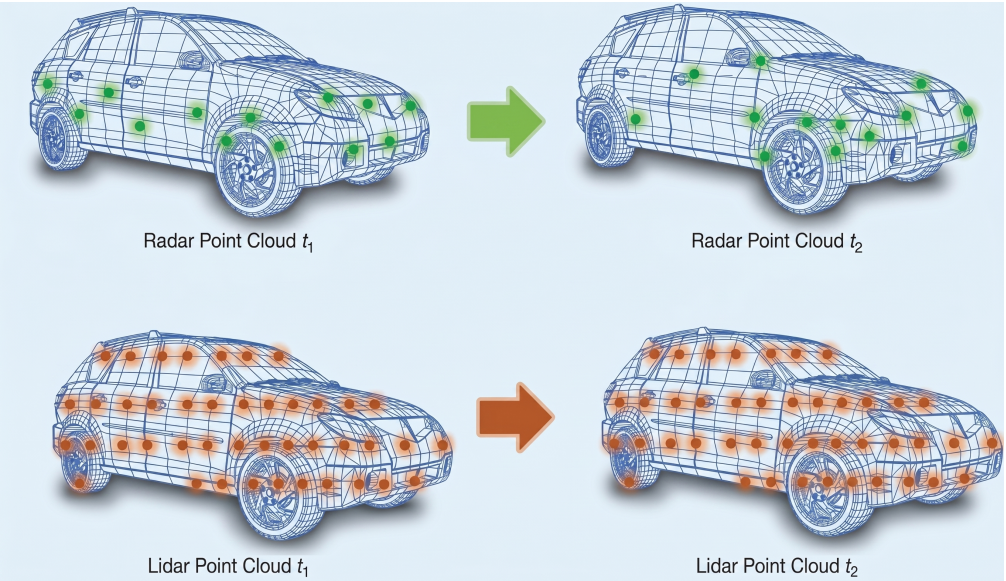
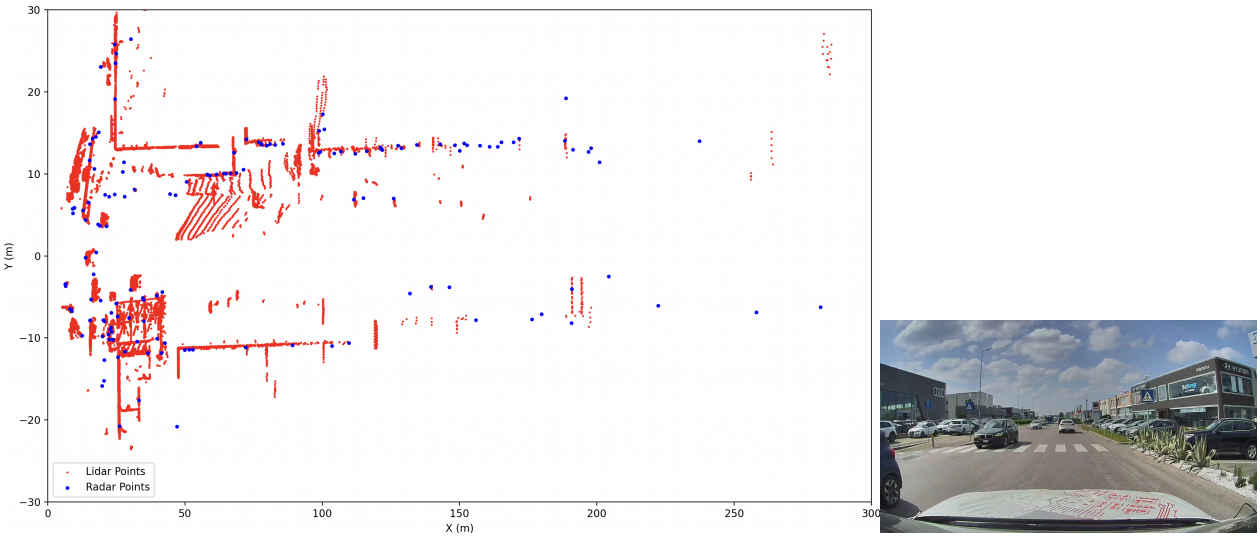


Figure 2.5: Difference between LiDAR and radar in spatiotemporal consistency in consecutive scans [23]

advantages of radar systems have already been discussed and why they are often chosen in the automotive sector. However, compared to LiDAR scans and point clouds, characterized by high angular resolution and spatiotemporal consistency, the radar resolution is significantly lower, resulting in a sparse point cloud. Moreover, the complex target detector decision used by radar can drastically change with a small variation (in position or orientation) of the target, making the point cloud temporally inconsistent and causing the same target to appear completely different between consecutive scans. Radar is also not able to show the precise shape of the target due to both the lower resolution and the CFAR detection algorithm. In Figure 2.5, there is a visualization of the difference described between LiDAR and radar in consecutive scans [23].

In figure 2.6, both radar and LiDAR scans were recorded from the same scene to compare the two point clouds and provided the corresponding camera image as a reference. A filter to only keep the points with a positive z-coordinate has been applied to the LiDAR point cloud to better visualise the obstacles from the BEV. It is visible how just a few radar (blue) points are on the target while many more LiDAR (red) points can show the exact shape of the (close) targets. This comparison utilizes the setup from the second experimental phase of this thesis, with a LiDAR Innovusion Falcon and a radar 3+1D Continental ARS 540. The total LiDAR points of this scan are 81073, while the total radar points are 227. However, from this image we can also notice the long range of



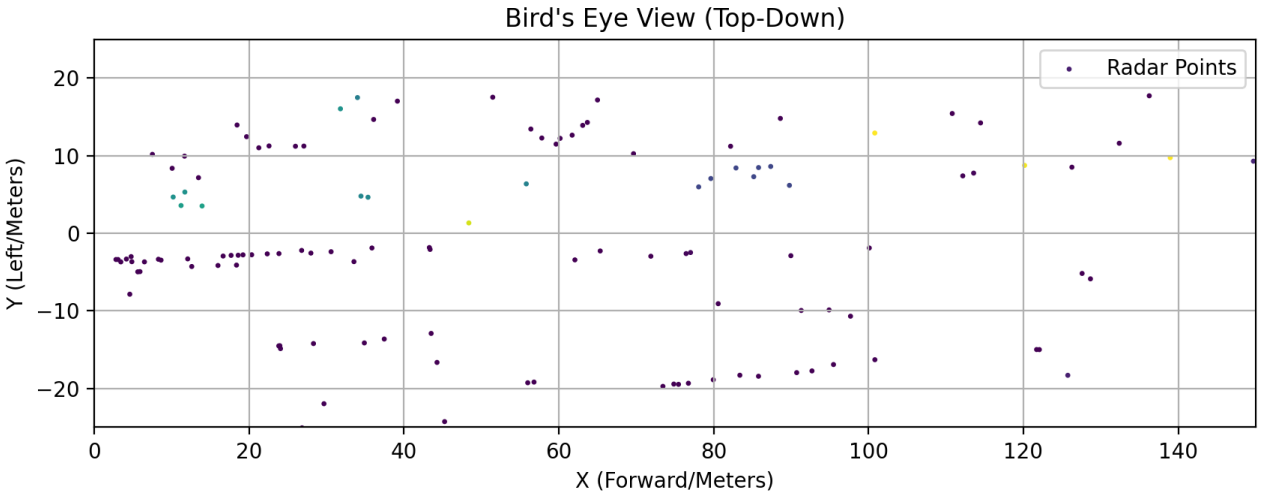
(a) Radar (Blue) and LiDAR (Red) (b) Camera image

Figure 2.6: Comparison point clouds between LiDAR and Radar

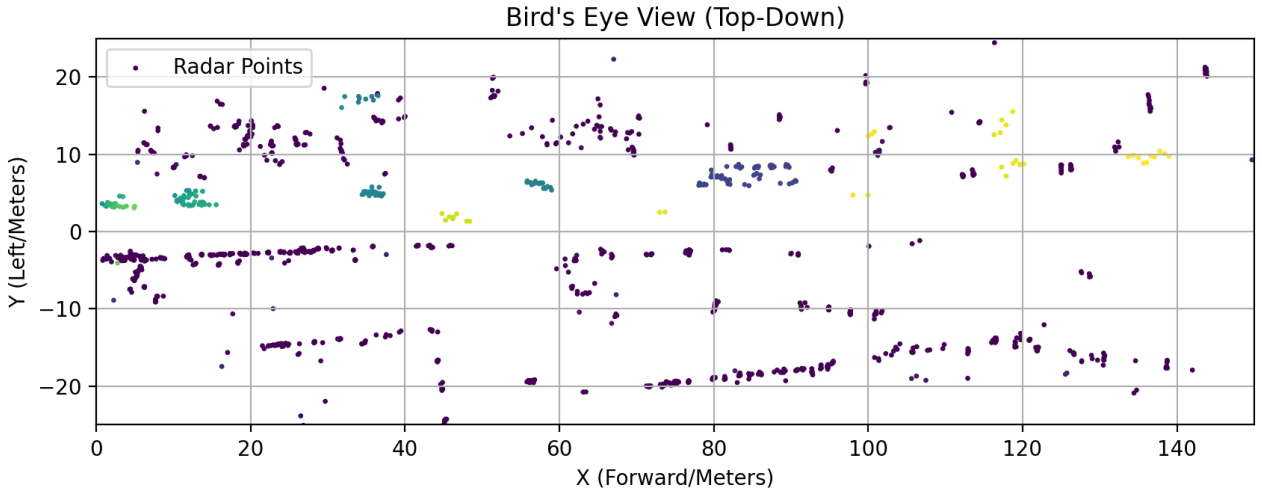
radar detections which are less disturbed by obstacles compared to LiDAR.

In order to compensate for the lack of temporal and spatial coherence, many studies attempted to aggregate multiple consecutive radar scans by appending a feature that represents the relative time of the aggregated scan [18],[22]. This feature can either be the difference between the actual scan timestamp and the aggregated scan timestamp, or, while the frame rate is constant, it could be an integer representing the number of frames between the actual frame and the aggregated frame [18],[22].

In figure 2.7, with the same setup and scene as in 2.6, we have the comparison between a single point cloud and aggregating 10 consecutive scans and compensating the ego motion in the older scans. The colour represents the absolute radial velocity speeds of the targets to better understand the scene. It is visible how the shapes become more understandable and we have more points hitting the different objects. However, we can also notice how moving objects leave a tail of points because the ego-motion compensation left the point where the target was in the specific scan, and it does not take into account the movement of the target itself.



(a) Single radar point cloud



(b) Ten point clouds aggregated

Figure 2.7: 2D Bird-Eye-View of radar point clouds with aggregation

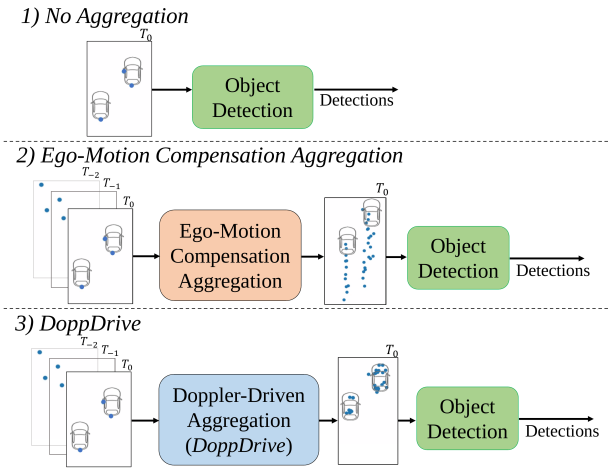


Figure 2.8: Comparison among three different temporal aggregation methods: (1) no aggregation, (2) standard ego-motion compensation, and (3) Doppler-driven temporal aggregation [24]

Doppler-driven Temporal Aggregation

To address this "tail" effect observed during frame accumulation, a further spatial compensation step can be applied utilizing the measured radial velocity [24]. While standard ego-motion compensation successfully aligns the static background across consecutive scans, it inherently fails to account for the displacement of dynamic targets. This causes their historical points to smear, trailing behind the vehicle's true current position. By leveraging the absolute radial velocity ($v_{target,absolute}$) extracted from the Doppler shift, it is possible to project these historical points forward in time [24]. Specifically, the positional offset is estimated by multiplying the target's radial velocity by the time delta between the historical scan and the current frame. Shifting the accumulated points along their Line-of-Sight (LoS) vector by this computed offset significantly reduces the spatial smearing. This Doppler-driven temporal aggregation effectively compresses the scattered tail into a denser, more geometrically accurate point cloud, providing a much stronger structural representation of moving targets for the downstream 3D detection network [24]. As illustrated in Figure 2.8, the input of the network is denser and more coherent, which is expected to improve the detection performance.

This compensation works well for targets moving along the LoS, but it is important to note that it does not fully correct for lateral motion. For example, if a vehicle is moving perpendicular to the radar's LoS, the Doppler compensation will not accurately project its historical points forward in space. However, in practice, most vehicles on the road tend to move predominantly along the LoS

of the radar, especially in highway scenarios. Therefore, this method still provides a significant improvement in point cloud quality for the majority of dynamic targets encountered in automotive applications.

Feature Selection for Network Input

To conclude the analysis of radar data characteristics, it is crucial to clarify how this diverse set of extracted features is actually structured and fed into the 3D object detection networks. While the radar internally processes raw power, relative radial velocity, and intermediate Doppler metrics, not all of these are directly utilized by the neural network.

In the scope of this thesis, the final input vector assigned to each point in the cloud is constructed by selecting only the most informative, ego-independent features. The geometric spatial coordinates (x, y, and z) and the physical Radar Cross Section (RCS) are retained as the foundational spatial and semantic inputs. Instead of using the raw relative radial velocity—which is heavily biased by the vehicle's own movement—the network specifically receives the ego-motion compensated absolute radial velocity components (v_comp_x and v_comp_y). Finally, to provide the necessary temporal context and mitigate the sensor's inherent spatial sparsity, the time offset (or frame index) of the accumulated scans is appended to each point.

Consequently, intermediate relative velocities, uncompensated Doppler shifts, and the raw analog radar signals are entirely discarded prior to the deep learning pipeline. This carefully selected combination of spatial coordinates, RCS, absolute velocity components, and temporal information forms the complete, consolidated feature vector that serves as the direct input for the neural network architectures evaluated in the following sections.

2.3 3D Object Detection on Point Clouds

Having explored the specific features, sparsity, and temporal characteristics of radar point clouds, this section provides an overview of how deep learning architectures utilize this data to perform 3D object detection. While some methods were originally designed for LiDAR sensors and subsequently adapted for radar data, others were explicitly developed to target the unique characteristics of radar. Notably, there is no single architecture that universally outperforms the others across all metrics

and scenarios.

In recent years, driven by improvements in sensor hardware, numerous algorithms have been proposed for automotive object detection. However, the inherent sparsity of radar point clouds makes estimating precise bounding boxes highly challenging—especially for smaller targets like pedestrians—as an object may be represented by only a handful of reflections. Furthermore, even though radar point clouds visually resemble LiDAR point clouds, LiDAR-native networks were not originally designed to process Doppler information, and extracting the most meaningful semantic context from radial velocity remains an open research challenge.[18]

Broadly, modern detection architectures can be categorized based on how they process this unstructured point cloud data.

2.3.1 Point-Based Methods

The most direct way to handle the sparse radar data previously described is through point-based networks, which process the input cloud in its raw, unstructured form without applying any intermediate data transformations (such as voxelization) [25]. They are frequently used to perform tasks like semantic segmentation directly on the point cloud, with pioneering architectures like PointNet relying on shared Multi-Layer Perceptrons (MLPs) to extract features [25]. The primary advantage of this approach is that it strictly maintains the original geometric structure of the data [25]. However, standard point-based methods often struggle to efficiently model the complex, local spatial relationships and neighborhood contexts between individual points [25].

2.3.2 Grid-Based Methods: From Voxels to Pillars

Because standard point-based methods often struggle to efficiently model local spatial relationships, an alternative approach emerged. Historically, processing 3D point clouds involved dividing the continuous 3D space into a dense volumetric grid of individual 3D pixels, known as voxels [26]. While voxel-based networks can capture fine geometric details, they require the application of 3D Convolutional Neural Networks (3D CNNs) [26]. This approach is notoriously computationally expensive and slow, making it difficult to deploy on real-time embedded systems in vehicles [26].

To increase computational efficiency, a more recent approach involves projecting the 3D point

cloud onto a 2D Bird's Eye View (BEV) plane, effectively collapsing the vertical dimension [26]. This results in a grid of pillars, which can be processed using 2D CNNs. This method significantly reduces computational complexity while still capturing essential spatial relationships in the horizontal plane [26].

Despite their computational efficiency, a fundamental drawback of grid-based methods is that the creation of voxels or pillars inherently discretizes the continuous 3D space. This quantization process inevitably results in a loss of fine-grained spatial information [25].

2.3.3 Graph-Based Methods

Seeking to combine the structural preservation of point-based methods with better relational modeling, graph-based architectures have also been introduced. The core idea behind these architectures is to leverage not only the individual features of each point but also the explicit relationships between them to correctly classify targets [25]. By adopting a graph-based approach, the points represent the nodes of the graph, which maintains the original unstructured geometry of the data as seen in point-based approaches, while the relationships between the points are explicitly modeled as edges [25]. One of the pioneering applications of this approach to solve the sparse radar detection task is the RadarGNN architecture [25].

2.3.4 Other approaches to the problem

Having reviewed point-based, grid-based, and graph-based approaches that operate on standard point clouds, we finally mention other alternative designs used for radar object detection for the sake of completeness.

One of the most common approaches is to use a multi-modal input, where radar is not the only input to the network. These networks usually outperform radar-only networks due to additional information from multiple sensors, such as cameras, which can compensate for the inherent limitations of radar systems [27]. These methods are excluded from this study, which strictly evaluates radar-only architectures.

Another approach involves bypassing the CFAR point cloud entirely and directly using the raw 4D radar data, specifically, the power measurements over the different dimensions [28],[29]. By

bypassing the CFAR, the signal is filtered so that the information is denser to give the network the possibility to obtain more information about the structural context of the target [28],[29]. However, not only the datasets available with the raw data are extremely limited, but many radars do not expose the raw measurements in output, making this approach hard to generalize and use at the moment.

2.3.5 Summary of Evaluated Architectures and Chapter Conclusion

To conclude this theoretical background, it is crucial to clearly delimit the scope of the algorithms that will be practically evaluated in the experimental chapters of this thesis. Based on the stringent requirements of the target use cases (overtake assist and heavy-duty platooning) and the practical constraints of current sensor hardware, this work strictly focuses on radar-only, point-cloud-based detection architectures.

Specifically, the subsequent experiments will exclusively evaluate networks that process filtered CFAR point clouds, utilizing the carefully selected feature vector defined in Section 2.2 (spatial coordinates, RCS, absolute radial velocity, and temporal aggregation). This includes the investigation of grid-based methods—such as PointPillars and PillarNet—which leverage BEV projections for computational efficiency, as well as methodologies designed to preserve the sparse geometry of the data.

Conversely, as highlighted in the preceding subsections, multi-modal sensor fusion pipelines (e.g., combining radar with cameras or LiDAR) and architectures requiring raw, pre-CFAR 4D radar tensors are explicitly excluded from this study. By isolating the radar modality and relying on standardized CFAR outputs, the following chapters aim to rigorously and fairly assess the true independent capabilities, limitations, and potential algorithmic improvements of purely radar-driven 3D object detection in challenging environments.

3. Algorithmic Paradigms for Sparse 3D Detection

In this chapter, the technical implementation of the architectures utilized during the experimental phase is detailed, with a specific focus on the modifications required to process radar data. The selection of these networks follows a progressive methodology: starting from established LiDAR-based architectures that can be adapted for radar, before exploring frameworks designed exclusively for radar sensors. Specifically, PointPillars was selected because it remains a widely adopted, foundational baseline in point cloud detection. Building on this foundation, PillarNet was included as it optimizes the pillar-based representation and currently outperforms state-of-the-art models[30]. Finally, specialized approaches—such as cross-modal feature augmentation and RadarGNN—are explored to directly address the inherent sparsity and unique physical properties of radar data.

3.1 Point Pillars

As the foundational baseline for this study, PointPillars provides a robust grid-based framework. To overcome the severe computational bottleneck of 3D voxel networks, the PointPillars architecture was introduced. PointPillars revolutionized grid-based detection by proposing a novel encoder that learns features on vertical columns (pillars) of the point cloud [26]. Unlike traditional voxel networks that discretize the z-axis, PointPillars sets the height of the voxels equal to the total range along the z-axis, effectively eliminating the vertical dimension. This paradigm shift allows the network to collapse the 3D point cloud into a 2D pseudo-image [26]. Consequently, highly optimized 2D Convolutional Neural Networks can be applied, resulting in a faster and drastically more efficient execution compared to traditional 3D voxel networks, making it an order of magnitude faster than previous methods [26].

3.1.1 Architecture

As shown in figure 3.1, the main components of the network are a Pillar Feature Network (PFN), a 2D CNN Backbone, and SSD Detection Head [26].

Architecture	Paradigm	Encoder	Detection Head	Dataset	Reason
PointPillars	Grid-based (Pillars)	Pillar Feature Network + 2D CNN Backbone	Anchor-based (SSD)	Various	Foundational baseline for point cloud detection.
PillarNet	Sparse Grid-based	2D Sparse CNN + Neck	Anchor-free (CenterHead)	Various	Efficiently processing sparse data.
Cross-Modal	Feature hallucination	Instance Aggregation + Alignment Projection	Anchor-based (SSD)	VoD	Simulates LiDAR spatial features.
RadarGNN	Graph Neural Networks	Graph Constructor + Message Passing Layer	Box / Seg. Head	RadarScenes NuScenes	Designed exclusively for radar data.

Table 3.1: Summary of selected network architectures, their structural paradigms, and the reason for their inclusion

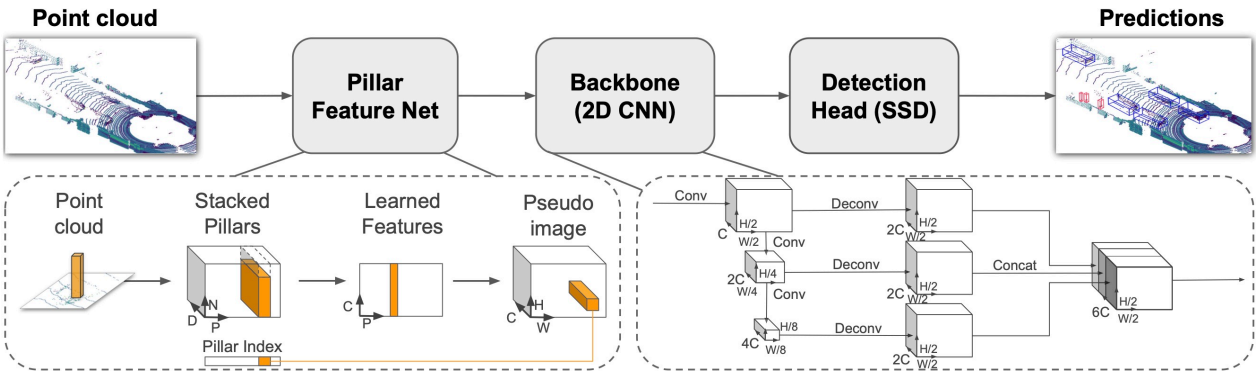


Figure 3.1: Architecture overview of PointPillars [26]

Pillar Feature Network

The PFN is used to convert the input point cloud to the 2D pseudo-image to apply the 2D CNN [26]. The LiDAR is composed of points that are characterized by x, y, z and the reflectance r, later it will be discussed the difference in radar implementation. The point cloud is then divided on the xy plane in pillars of fixed dimensions, hyperparameters of the architecture. So we obtain $B = \frac{Range_x}{pillar_size_x} \cdot \frac{Range_y}{pillar_size_y}$ pillars containing points [26]. Then the features of the points are augmented and the features x_c, y_c, z_c, x_p and y_p are added to the 4 features of the initial points, where c represents the distance from the arithmetic mean of all the points of the single pillar and p is the offset from the center of the pillar [26]. The result is a collection of all the pillars with every point of the final dimension $D = 9$. However, LiDAR creates a sparse output, with a sparsity of around 97% using the LiDAR HDL-64E Velodyne [26]. To account for sparsity and to have a fixed input for the 2D CNN, both the number of pillars P and the number of points in each pillar N are limited [26]. Only non-empty pillars are considered; if non-empty pillars or points for each pillar are not enough, zero padding is applied [26]. Conversely, if the number of points exceeds this threshold, they are randomly sampled [26]. Now, we have a tensor of size (D, P, N) that is passed to a linear layer, a Batch-Norm and ReLU to generate a tensor of size (C, P, N), where C represents the number of output channels extracted by the linear layer [26]. A max operation is applied on the channels to finally obtain a tensor of size (C,P) [26]. Once encoded, the features are scattered back to the original pillar locations to create a pseudo-image of size (C,H,W) where H and W indicate the height and width of the canvas [26].

Backbone - 2D CNN

As shown in 3.1, the backbone is designed to extract features at varying spatial resolutions and is systematically divided into two main sub-networks:

- Top-down network: This sub-network is tasked with multi-scale feature extraction. It is made up of a series of sequential convolutional blocks. Each block typically applies 2D convolutions, followed by Batch Normalization and a ReLU activation function [26]. Crucially, the initial convolution of each block utilizes a stride greater than one to downsample the spatial dimensions of the input. This progressive downsampling allows the network to

increase its receptive field, capturing broad, high-level semantic information at the cost of spatial resolution [26].

- Upsampling and concatenation network: Accurate object detection requires a balance between the high-level semantic context (found in deeper, low-resolution layers) and fine-grained spatial details (found in earlier, high-resolution layers) [26]. To achieve this, the second sub-network takes the intermediate output feature maps from each block of the top-down network [26]. It applies transposed 2D convolutions to upsample all these feature maps to a single, unified spatial resolution. Once harmonized in size, these maps are concatenated along the channel axis [26]. This concatenated tensor provides a rich, multi-scale feature representation ready for the final detection stage [26].

Detection Head (Anchor-based)

The final component of the PointPillars architecture is the detection head, which translates the enriched multi-scale features from the backbone into concrete 3D object predictions. Inspired by the Single Shot MultiBox Detector (SSD) framework, PointPillars employs an anchor-based detection strategy [26]. The detection process relies on several key mechanisms:

- Anchor Definition: The model superimposes a grid of prior boxes, commonly known as anchors, on the spatial feature map. Each anchor is a predefined 3D bounding box shaped with fixed dimensions (width, length, and height) suited for specific target classes, such as cars or pedestrians. Furthermore, to account for vehicle heading, these anchors are instantiated at predefined rotation angles, most commonly at 0 and 90 degrees [31],[26]. In figure 3.2, a dotted line illustrates how multiple boxes of predefined dimensions are spread along the feature map.

- Matching Strategy: During training, the network must assign which anchor is responsible for predicting a specific ground-truth object. PointPillars simplifies this by performing the matching process strictly in the 2D Bird's Eye View (BEV) plane. It calculates the 2D Intersection over Union (IoU) between the anchors and the ground-truth boxes. The vertical components (elevation and bounding box height) are intentionally excluded from this matching

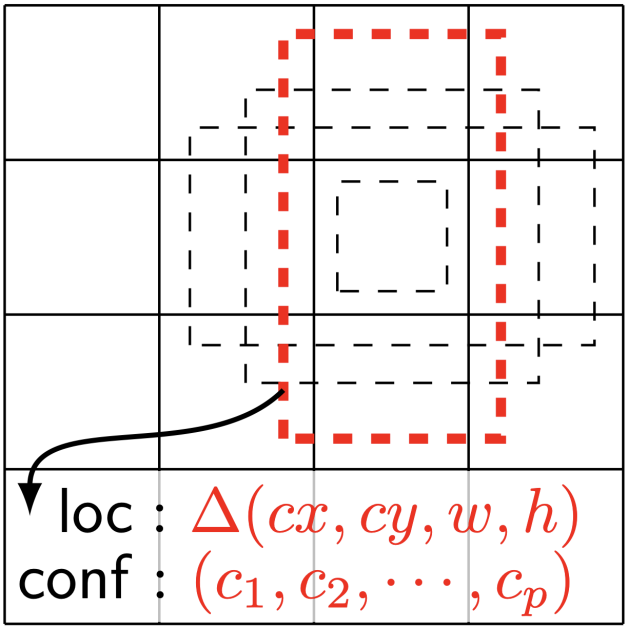


Figure 3.2: SSD anchors on a 4x4 feature map [31]

criterion. Instead, once an anchor is matched to an object based on the 2D IoU threshold, the z-coordinate and height become additional parameters for the network to regress [31],[26].

- Network Outputs: For every predefined anchor in the spatial grid, the detection head computes two distinct types of predictions. The first is a classification score that estimates the probability that the anchor contains an object of a particular class rather than background noise. The second is a bounding box regression that produces a vector of continuous offsets. These offsets instruct the model on how to adjust the anchor's default center coordinates (x,y,z), spatial dimensions, and yaw angle to closely wrap the detected target [31],[26].

Loss Function

To optimize the network during the training phase, PointPillars employs a multi-task loss function introduced by the SECOND architecture. The total loss is a weighted sum of three distinct components: a localization loss for the bounding box regression, a classification loss for object categorization, and a direction loss to resolve heading ambiguity [26].

Each 3D bounding box, whether a ground truth label or a predefined anchor, is parameterized by seven physical dimensions: spatial coordinates, physical sizes, and the yaw angle, denoted

as $(x, y, z, w, l, h, \theta)$. To compute the localization error, the network does not predict absolute coordinates, but rather the continuous residuals (offsets) between the ground truth boxes (gt) and the matched anchor boxes (a) [26]. The regression residuals are defined as follows:

$$\begin{gathered} \Delta x = \frac{x^{gt} - x^a}{d^a}, \quad \Delta y = \frac{y^{gt} - y^a}{d^a}, \quad \Delta z = \frac{z^{gt} - z^a}{h^a} \\ \Delta w = \log\left(\frac{w^{gt}}{w^a}\right), \quad \Delta l = \log\left(\frac{l^{gt}}{l^a}\right), \quad \Delta h = \log\left(\frac{h^{gt}}{h^a}\right) \\ \Delta\theta = \sin(\theta^{gt} - \theta^a) \end{gathered} \tag{3.1}$$

where $d^a = \sqrt{(w^a)^2 + (l^a)^2}$ represents the diagonal of the anchor box base. The localization loss (L_{loc}) is then calculated using the Smooth L1 Loss function over these residuals, which is less sensitive to extreme outliers than the standard L2 loss [26]:

$$L_{loc} = \sum_{b \in (x,y,z,w,l,h,\theta)} \text{SmoothL1}(\Delta b) \tag{3.2}$$

For object classification, the network faces an extreme class imbalance problem. Because the spatial grid generates thousands of prior anchors, the vast majority of them correspond to empty background space rather than actual objects. To prevent the loss from being overwhelmed by easy-to-classify background anchors, PointPillars utilizes the Focal Loss (L_{cls}) [26]. By dynamically scaling the standard cross-entropy loss based on the prediction confidence (p^a), Focal Loss forces the network to focus its learning capacity on hard, misclassified examples [26]:

$$L_{cls} = -\alpha_a (1 - p^a)^\gamma \log(p^a) \tag{3.3}$$

where α_a and γ are the class weighting and focusing parameters, respectively.

Finally, because the regression residual for the yaw angle relies on a sine function ($\Delta\theta = \sin(\theta^{gt} - \theta^a)$), it inherently cannot distinguish between an object facing forwards (θ) or completely backwards ($\theta + \pi$) [26]. To resolve this orientation ambiguity, a discrete direction classifier is added to the network. This uses a standard Softmax cross-entropy loss (L_{dir}) to classify the heading into one of two opposite bins [26].

The overall objective function optimized during training is the weighted sum of these three components, normalized by the total number of positive anchors (N_{pos}):

Method	Features	Entire annotated area					In Driving Corridor				
		Car	Pedestrian	Cyclist	mAP	mAOS	Car	Pedestrian	Cyclist	mAP	mAOS
PP-radar (no elevation)	x, y, RCS, v_r	32.4	28.8	34.6	31.9	25.1	68.2	44.2	63.6	58.6	50.1
PP-radar (no Doppler)	$x, y, z, RCS,$	35.6	21.3	30.4	29.1	22.1	67.3	31.0	58.7	52.3	41.0
PP-radar (no RCS)	x, y, z, v_r	33.9	33.1	42.7	36.6	30.3	66.8	45.3	**67.2**	59.8	55.6
PP-radar	x, y, z, RCS, v_r	**35.9**	**34.9**	**43.1**	**38.0**	**30.5**	**74.1**	**47.8**	67.1	**63.0**	**56.8**
PP-radar (3 scans)	x, y, z, RCS, v_r, t	44.4	40.4	**54.2**	46.3	39.1	78.4	56.9	**76.6**	70.6	67.1
PP-radar (5 scans)	x, y, z, RCS, v_r, t	**44.8**	**42.1**	54.0	**47.0**	**39.6**	**78.8**	**59.2**	76.1	**71.4**	**68.2**
PP-LiDAR (LiDAR)	$x, y, z,$ *intensity*	75.6	55.1	55.4	62.1	49.4	90.8	71.4	82.5	81.6	70.3

Table 3.2: Results from VoD experiments with different features.[18]

$$L_{total} = \frac{1}{N_{pos}} \left(\beta_{loc} L_{loc} + \beta_{cls} L_{cls} + \beta_{dir} L_{dir}\right) \tag{3.4}$$

where β_{loc}, β_{cls}, and β_{dir} are scalar weights that balance the contribution of the respective loss terms.

Radar Adaptation

As already mentioned, the designed network was created for the LiDAR sensor. To convert the architecture to work optimally with the radar, we need to modify a few parts to include all the information that we can extract from the radar. For the adaptation of the network, the work presented by [18] has been followed, which adopted the solution to simply add the new features to the normal LiDAR input and modify the Voxel Feature Encoder to account for a different number of features as input. Unlike the processing of LiDAR features, no augmentation is performed on the new features [18]. A study has been conducted on how the removal of the different features affected the precision of the network [18]. Additionally, the accumulation of radar point clouds was evaluated. The results are presented in table 3.2, the study was performed with a ZF FRGen21 3+1D radar for the publication of VoD dataset [18]. The table shows how missing information due to the removal of any feature results in a worse performance, confirming how all the information captured by the radar represents important data to include. Moreover, the accumulation of multiple scans, with the additional information of the time of the scan, improves even more the performance in the case of this dataset.

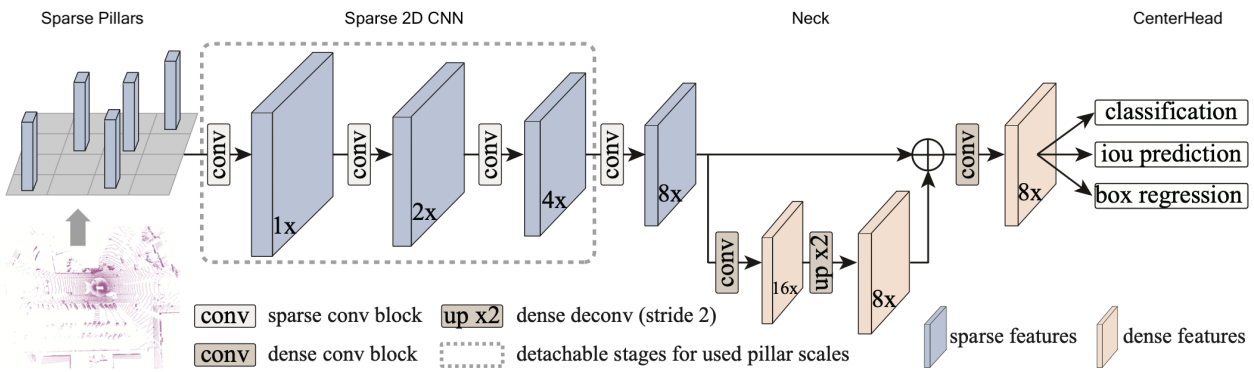


Figure 3.3: PillarNet architecture overview [30]

Beyond adding input channels, adapting the network's structural hyperparameters, such as increasing the spatial dimensions of the pillars (width and length) and adjusting the maximum number of pillars and points per pillar—is crucial to compensate for the extreme sparsity of the radar point cloud. The specific tuning of these parameters, alongside their impact on detection accuracy, will be thoroughly detailed in the experimental evaluation.

3.2 PillarNet Architecture

While PointPillars provides a robust grid-based foundation, its reliance on dense 2D convolutions introduces computational inefficiencies when applied to the extreme sparsity of radar point clouds. To process sparse 3D point clouds efficiently while maintaining high detection accuracy, this work utilizes the PillarNet architecture. While building upon the foundational pillar-based representation of PointPillars, PillarNet addresses the computational bottlenecks of early grid-based methods by replacing dense 2D convolutions with a highly efficient 2D sparse convolutional encoder [30]. Furthermore, it integrates a flexible, anchor-free CenterHead to eliminate the need for rigid bounding box anchors. The architecture operates through four main sequential stages as shown in figure 3.3 [30].

3.2.1 Pillar Generation and Feature Encoding

The initial stage discretizes the continuous point cloud into an evenly spaced grid in the x-y plane, creating vertical columns (pillars) without discretizing the z-axis. The features of the points within

each active pillar are processed by a simplified PointNet module to extract a single representative feature vector per pillar [30]. Unlike standard PointPillars, which immediately scatters these features into a dense pseudo-image, PillarNet strictly retains the data in a sparse coordinate format to leverage specialized spatial operations.

3.2.2 2D Sparse Convolutional Encoder

The core architectural distinction of PillarNet is its spatial backbone, which relies entirely on 2D Sparse Convolutional Neural Networks rather than standard dense convolutions. Because radar and LiDAR point clouds are inherently sparse, projecting them onto a Bird's-Eye-View (BEV) grid leaves the vast majority of pixels empty [30]. By employing 2D sparse convolutions, the network selectively performs computations only on active spatial locations. This encoder uses a series of downsampling sparse blocks (typically organized into five distinct stages) to efficiently extract deep geometric features and progressively expand the receptive field, capturing complex local relationships without the severe computational overhead of dense matrices [30].

3.2.3 Feature Fusion Neck

To transition from the sparse domain to the dense representations required by the detection head, PillarNet utilizes a dedicated neck module, structured similarly to a Spatial Feature Pyramid Network (SpatialFPN) [30]. Rather than aggregating all downsampling scales, the neck gathers specific high-level semantic and low-level spatial features generated by the sparse encoder [30]. It applies dense upsampling operations and concatenates these targeted feature maps into a single, rich, high-resolution dense 2D BEV feature map that encapsulates both fine-grained local details and broad contextual geometries [30].

3.2.4 CenterPoint Detection Head (CenterHead)

Unlike the traditional SSD-based approach that relies on numerous predefined anchor boxes, the PillarNet architecture utilizes an elegant, anchor-free detection head, commonly referred to as CenterHead. This module significantly simplifies the 3D object detection pipeline by representing and detecting vehicles and pedestrians as single points in space—specifically, their geometric centers

[32].

Heatmap-Based Center Prediction

The core of the CenterHead is a keypoint detector that processes the Bird's-Eye-View (BEV) feature map extracted by the 2D CNN backbone. For a dataset comprising C distinct object classes, the head predicts a multi-channel spatial heatmap $\hat{Y} \in [0, 1]^{C \times H \times W}$. Each pixel in this output heatmap represents the probabilistic confidence of being the center of an object of a specific class [32].

During the training phase, the ground-truth 3D bounding box centers are projected onto the 2D BEV grid. To prevent the network from heavily penalizing predictions that are very close (but not exactly aligned) to the true center, a 2D Gaussian kernel is applied around each ground-truth center point, creating a continuous, smoothed probability distribution [32]. The network is subsequently optimized to predict this distribution using a penalty-reduced Focal Loss. This specific loss function effectively mitigates the massive class imbalance between the few positive object centers and the overwhelming majority of empty background pixels [32].

Attribute Regression

Once an object's center is localized on the BEV grid, the CenterHead employs a series of parallel, lightweight convolutional layers to directly regress the remaining 3D physical properties of the target strictly from that localized center feature. For each predicted center, the regression sub-heads output a dense set of parameters:

- **Sub-grid Offset:** To compensate for the quantization error introduced by discretizing the continuous 3D point cloud into a finite BEV grid, a continuous offset $(\Delta x, \Delta y)$ is predicted to recover the exact coordinates of the target [32].
- **Elevation:** The absolute vertical coordinate (z) of the object's center [32].
- **3D Dimensions:** The physical bounding box size, denoted by its length, width, and height (l, w, h) [32].
- **Orientation:** The yaw angle (α) of the bounding box. To circumvent the mathematical discontinuities of direct angle regression (e.g., at 0 and 2π), the orientation is typically encoded and regressed as a continuous vector $[\sin(\alpha), \cos(\alpha)]$ [32].

- **Velocity:** The estimated 2D velocity vector (v_x, v_y), which is a particularly critical feature when integrating Doppler measurements from 4D radar sensors [32].

Because these geometric and kinematic attributes are extracted directly from the peak center point features, they are optimized using a standard L1 Loss. Crucially, this loss is calculated only at the positive ground-truth center locations. This paradigm eliminates the computationally expensive Intersection over Union (IoU) matching phase and the complex anchor-tuning heuristics required by traditional bounding box detectors. Compared to the anchor-based detection head, which relies on boxes aligned with the axis of the ego to predict objects, centerhead aims to improve predictions in the situations where objects are not aligned with the ego, such as in turns [32].

3.3 LiDAR-Radar Cross-Modal Feature Augmentation

While architectures like PointPillars and PillarNet focus on optimizing the internal network structure to more efficiently process radar data, they still fundamentally rely on the inherently sparse radar point cloud. To address the physical limitations of the sensor itself, rather than solely refining the computational pipeline, a different learning paradigm can be adopted. As seen previously, due to the sparse output, radar lacks spatial and geometric information, while it contains other valuable information like RCS and radial velocity. LiDAR, on the other hand, represents a good solution for detecting shapes. This framework aims to introduce cross-modal hallucination, in which the network learns how to simulate the other sensor features only from the main sensor [33]. In fact, this network uses both radar and LiDAR in the learning process, while during inference only the main sensor is used. Radar and LiDAR are interchangeable, they can both be an auxiliary or primary sensor in different trainings [33]. However, for the objective of this thesis, only the case of radar as a primary modality is described, as it evaluates radar object detection models.

3.3.1 Architecture

In figure 3.4, we can see how during inference, only the main modality is used. The main body of the network is identical to the Point Pillars design, with the same encoder, backbone and detection head. However, this network tries to learn to simulate the features of the other modality [33].

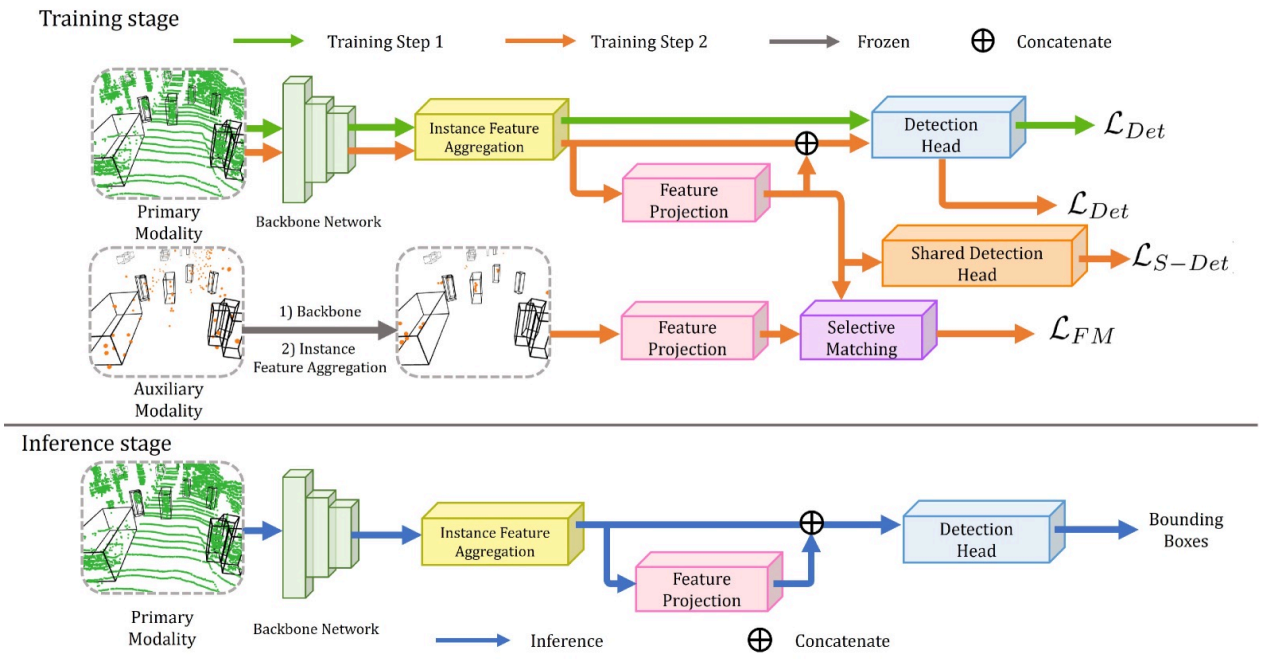


Figure 3.4: Cross-modal feature augmentation overview [33]

- **Instance Feature Aggregation:** This module is useful for performing a spatial alignment between radar and LiDAR. Due to the extreme sparsity of the radar data, there is an inherent geometrical discrepancy between the sensors, which could cause the cross-modal matching to fail. To mitigate this, the module predicts an offset to shift the points toward the centroid of the target, generating "centered" points [33].

- **Alignment-aware Feature Projection:** The distinct physical properties of the sensors result in modality-specific features, making direct matching impossible. To resolve this, the framework projects the aggregated features from both modalities into a shared, common latent space with an empirically set dimension [33].

- **Selective Matching:** Operating within this shared latent space, the framework pairs points between the two modalities using a cross-modal Nearest Neighbor (1-NN) search, successfully finding the correct geometric matches within a specified radius [33].

3.3.2 Cross-Modal Loss Function

To enable the primary sensor (radar) to learn and hallucinate the features of the auxiliary sensor (LiDAR), the training process utilizes a customized joint loss function. The total loss (L_{total}) combines the standard bounding box detection loss with a specific Feature Matching loss (L_{FM}) [33]:

$$L_{total} = L_{det} + \lambda L_{FM} \tag{3.5}$$

where L_{det} represents the standard detection objectives (such as the Focal Loss for classification and Smooth L1 Loss for box regression, identical to the base PointPillars architecture) computed on the primary modality. The parameter λ is a scalar weight that balances the two objectives [33].

The core of the cross-modal learning lies in L_{FM}. Once the selective matching module identifies the valid pairs of points between the radar and LiDAR point clouds, the network calculates the feature distance between these paired points in the projected latent space [33]. The Feature Matching loss is typically formulated using a distance metric, such as the L1 norm, to penalize the difference between the primary radar features (F_{rad}) and the auxiliary LiDAR features (F_{lid}), where N_{pairs} represents the total number of valid point pairs successfully matched, and i denotes the index of each specific pair [33]:

$$L_{FM} = \frac{1}{N_{pairs}} \sum_{i=1}^{N_{pairs}} ||F_{rad}^{(i)} - F_{lid}^{(i)}||_1 \tag{3.6}$$

By actively minimizing this distance during training, the network forces the radar encoder to extract and generate rich, LiDAR-like geometric representations. Crucially, because L_{FM} is only applied during the training phase, the auxiliary LiDAR sensor can be entirely removed during inference, allowing the primary radar sensor to benefit from the augmented spatial awareness without incurring any additional computational or hardware costs at runtime [33].

3.4 RadarGNN

The architectures discussed so far all share a common foundational step: they discretize the 3D point cloud into 2D grid-based representations (pillars) to apply dense or sparse convolutional operations.

While effective, this grid projection can lead to quantization errors and a loss of the native, fine-grained geometric relationships between individual radar reflections. Shifting away from grid-based processing entirely, RadarGNN offers an alternative approach by directly processing the raw points as interconnected nodes [25]. It is crucial to state immediately that, due to its entirely different data representation and processing pipeline, this architecture is not strictly comparable on a one-to-one basis with the previously described grid-based models. RadarGNN is a design of a neural network that uses Graph Neural Networks (GNN) specifically targeting radar data [25].

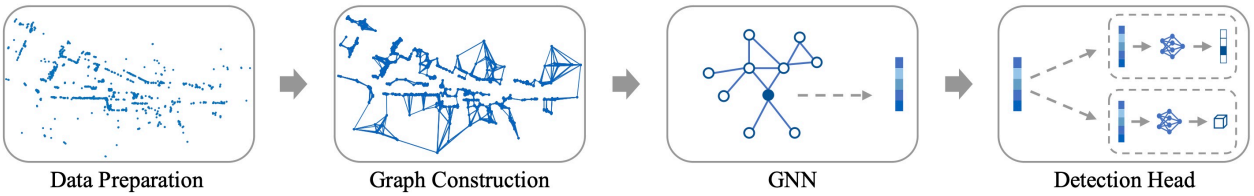


Figure 3.5: Overview of RadarGNN [25]

The overall architecture of the RadarGNN model, shown in Figure 3.5, consists of three major components: a Graph Constructor, the core GNN, and a detection head.

During the initial phase, the Graph Constructor processes the raw point cloud to build the graph representation. Crucially, the model is designed to consider both point features (nodes) and point-pair features, which are embedded directly into the edges of the graph [25]. To process this rich relational data, the network utilizes a generalized message passing neural network layer. This allows the GNN to consider edge features within its update function, enabling individual nodes (radar points) to iteratively aggregate contextual information from their neighboring nodes [25].

Mathematically, at a given layer l, the hidden state representation of a node i, denoted as $h_i^{(l)}$, is updated using the features of its neighboring nodes $j \in \mathcal{N}(i)$ and the connecting edge features e_{ij}. The message passing operation can be formulated as [25]:

$$h_i^{(l+1)} = U^{(l)}\left(h_i^{(l)}, \mathrm{AGG}_{j\in\mathcal{N}(i)} M^{(l)}\left(h_i^{(l)}, h_j^{(l)}, e_{ij}\right)\right) \tag{3.7}$$

In this formulation, $M^{(l)}$ is a learnable message function (implemented as a Multi-Layer Perceptron) that computes the "message" sent from node j to node i based on their current states and the edge feature e_{ij} between them. The AGG operator represents a permutation-invariant aggregation function, specifically keeping the maximum of the embedded features, which pools all incoming

messages from the neighborhood $\mathcal{N}(i)$. Finally, $U^{(l)}$ is the update function that combines the node's previous state $h_i^{(l)}$ with the aggregated messages to produce the refined feature representation $h_i^{(l+1)}$ for the next layer [25].

A major focus of the RadarGNN framework is handling the limited data availability and varying perspectives inherent to automotive radar. Achieving an invariant object detection model is of great interest because the model should be robust to unseen scenarios, where objects can occur in different locations or sensor mounting positions. RadarGNN addresses this by proposing a general approach for achieving transformation invariance [25]. In particular, transformation invariance is achieved by an invariant data representation rather than an invariant model architecture. By transforming the input features into an invariant state before passing them through the network, the model effectively counteracts the limited data availability and generalizes better to new driving scenes. Evaluated on the RadarScenes dataset, this approach outperforms all previous methods for bounding box prediction as well as semantic segmentation [25].

3.5 Summary and Expected Outcomes

To conclude this overview of the selected network architectures, it is crucial to outline the theoretical expectations for each model and define the key performance indicators that will be measured in the subsequent experimental phase.

Following the designed progression, PointPillars is expected to establish a solid, reliable baseline. Given its mature grid-based nature, it should offer fast inference times; however, it is anticipated to struggle with the extreme sparsity of radar data, potentially producing lower localization accuracy compared to its LiDAR counterparts. PillarNet, on the other hand, is expected to significantly outperform PointPillars. By leveraging its 2D sparse convolutional encoder and the anchor-free CenterHead, it should mitigate the computational waste of empty radar pillars and improve predictions for objects not perfectly aligned with the ego-vehicle's axes, such as turning vehicles.

Moving beyond structural optimizations, the Cross-Modal Feature Augmentation framework is expected to yield the highest spatial accuracy among the grid-based networks. By successfully hallucinating LiDAR-like geometric features from pure radar inputs, it should dramatically improve the bounding box regression and the detection of geometrically ambiguous targets, such as

pedestrians or cyclists, without increasing inference latency. Finally, RadarGNN is expected to demonstrate superior robustness and transformation invariance. By processing the raw point cloud as a graph rather than forcing it into a quantized grid, it is anticipated to generalize better across unseen sensor mounting positions and diverse driving scenarios, even though its fundamentally different architecture prevents a direct one-to-one benchmark with the pillar-based models.

To empirically validate these expectations, the subsequent experimental phase will measure several critical metrics. The primary evaluation will rely on standard 3D object detection metrics, specifically the Mean Average Precision (mAP) for 3D bounding boxes. Beyond global accuracy, the evaluation will measure class-specific performance, analyzing the discrepancy between highly reflective, massive targets (e.g., cars) and Vulnerable Road Users (VRUs) with low Radar Cross Sections.

Furthermore, the experiments will measure range-dependent accuracy, evaluating how detection degrades at extended distances, which is particularly critical for the long-haul trucking scenarios provided by the TruckScenes dataset. Finally, the models' capacity for domain adaptation will be measured. By evaluating pre-trained models on our custom urban datasets, we will quantify the performance drop caused by domain shifts (e.g., transitioning from high-speed highway traffic to densely packed, stationary urban vehicles) and measure the recovery of accuracy following targeted fine-tuning.

A fundamental aspect that must be considered alongside detection accuracy is the expected computational cost of each architecture. Autonomous driving applications impose strict real-time execution requirements, meaning that an algorithm must not only be accurate but also fast enough to process continuous sensor streams without introducing dangerous latency. In this context, PointPillars was inherently designed for speed and is expected to easily satisfy real-time constraints. PillarNet, through its sparse convolutional backbone, aims to maintain this high efficiency by bypassing empty space computations. The Cross-Modal Feature Augmentation framework is particularly advantageous regarding this metric: while the training phase is computationally demanding due to the dual-sensor setup, the inference phase relies solely on the primary radar sensor, effectively adding zero computational overhead to the base network at runtime. Conversely, RadarGNN, due to the complexity of dynamic graph construction and message passing operations, is anticipated to incur a higher computational cost. The critical trade-off between bounding box precision and infer-

ence latency, and whether these architectures successfully meet the strict real-time requirements of an in-vehicle deployment, will be a central focus of the analysis presented in Chapter 5.

4. Datasets: Overcoming the Range Limitations of Public Benchmarks

In the literature, the selection of public datasets for evaluating 3D object detection architectures is often driven by popularity rather than specific application requirements. This chapter deliberately breaks from that trend by evaluating the datasets strictly against the perceptual demands of the autonomous driving tasks targeted in this thesis: specifically, overtake assist and highway platooning.

These high-speed applications inherently dictate a long planning horizon, requiring reliable object detection at distances of 150 to 200 meters or more. Consequently, a core premise of this evaluation is that the most commonly cited datasets in academic papers are not necessarily the most representative or suitable for real-world deployment.

To properly assess a dataset's suitability, a fundamental distinction must be made between the hardware range of the sensor and the annotation range of the ground truth labels. A radar sensor might physically detect targets at 250 meters, but if the dataset's authors only labeled bounding boxes up to 50 meters, the neural network cannot be trained or evaluated for long-range tasks. The following sections critically analyze the major public datasets through this specific, requirement-driven lens.

During the preliminary research phase, a wide array of public autonomous driving datasets containing radar data was systematically evaluated. However, the majority of these datasets were ultimately discarded due to fundamental incompatibilities with our specific objectives. Many feature outdated or mismatched radar hardware that does not align with the modern 2+1D or high-resolution 3+1D sensors utilized in this study. Furthermore, several available datasets suffer from severely limited ground truth annotation ranges or possess an insufficient number of annotated samples to train complex neural architectures effectively. Consequently, the final selection presented in this chapter includes only the most representative datasets that explicitly match our hardware configuration and provide the necessary data volume and scope to rigorously evaluate our targeted perception tasks.

Dataset	Radar	Features	Radars & FoV	Frames	Scenario	Dynamic vs. Static	Range	Task Suitability
RadarScenes	2+1D (ARS408)	x, y, RCS, v_rad_comp, t	4 (Front/Sides)	72,868	Mixed	Dynamic	100 m	**Unsuitable**
VoD	3+1D (ZF)	x, y, z, RCS, v_rad_comp, t	1 (Frontal)	8,693	Dense Urban	Static (7.2% mov.)	50 m	**Unsuitable**
nuScenes	2+1D (ARS408)	x, y, RCS, vx_comp, vy_comp, t	5 (360°)	40,000	Mixed Urban/Driving	Static (23.5% mov.)	**~141 m**	**Acceptable**
TruckScenes	3+1D (ARS540)	x, y, z, RCS, vx_comp, vy_comp, t	6 (360°)	30,000	Highway (>70%)	Dynamic (~87% mov.)	**> 230 m**	**Highly Suitable**
Custom 3+1D	3+1D (ARS540)	x, y, z, RCS, vx_comp, vy_comp, t	3 (Front/Sides)	3,429	Urban (Modena)	Static (27% mov.)	**300+ m**	**Suitable (Domain Adapt.)**

Table 4.1: Comparative evaluation of public and custom radar datasets based on the strict requirements for long-range autonomous driving functions

4.1 RadarScenes

The RadarScenes dataset is a large-scale, real-world dataset specifically designed for automotive instance segmentation and object detection. The sensor array utilizes four 77 GHz automotive radar sensors operating in near-to-middle-range mode [34]. These sensors provide a maximum detection range of 100 m and a 60° Field of View (FoV) per sensor [34]. Two of the sensors are oriented at ±85° and the other two at ±25° relative to the driving direction; their point clouds are temporally aligned and merged to capture a comprehensive view of the vehicle's surroundings [34]. The dataset contains over 100 km of driving data across 158 distinct sequences, providing highly dense point-wise annotations and track IDs for dynamic road users [34], resulting in 72868 annotated frames [35]. Rather than relying solely on bounding boxes, the dataset provides ego-motion compensated Doppler velocities and radar cross-section (RCS) values for each individual point [34]. Even if the exact model of the radars is not specified, the characteristics given are comparable and aligned with the characteristics of the radar Continental ARS 408-21. This dataset has been chosen to conduct experiments on the RadarGNN architecture, as it is currently the most complete accessible dataset with point labels.

Consequently, while RadarScenes allows for the precise measurement of point-level algorithms and semantic segmentation tasks, it does not permit the evaluation of standard 3D bounding box regression or volumetric detection metrics by default. Moreover, its maximum annotation range of 100 meters is insufficient for the long-range perception requirements of the target applications of this thesis. Nevertheless, despite its unsuitability for volumetric regression, this dataset is retained in this study exclusively to train and evaluate the RadarGNN architecture, allowing us to validate our implementation against the original authors' findings and to benchmark point-wise semantic predictions against traditional 3D bounding box approaches.

4.2 View-of-Delft (VoD)

The View-of-Delft (VoD) dataset serves as the primary example of the discrepancy between academic popularity and practical applicability for high-speed tasks. It is a multimodal dataset focused on complex urban traffic environments, comprising 8,693 synchronized frames and providing over

123,000 3D bounding box annotations [18]. For radar perception, it utilizes a single ZF FR-Gen21 3+1D radar, which operates at a frequency of approximately 13 Hz, yielding 3+1D data that inherently includes range, azimuth, elevation, and Doppler velocity [18].

This dataset has been selected for analysis because it is currently one of the most heavily utilized datasets in literature to evaluate 3+1D radar networks. Its popularity stems from its dense urban context and an excellent coverage of Vulnerable Road Users. As visible in Table 4.2, the total number of bounding boxes for pedestrians is almost equal to the car detections [18].

However, VoD exposes a critical flaw regarding the distinction between sensor capabilities and annotation limits. While the radar hardware itself is physically capable of long-range detection, the ground truth 3D bounding box annotations are strictly artificially cropped. Even though [36] mentions a range of 26 meters for VoD annotations, our exploratory data analysis reveals they extend to a maximum of 50 meters. Because applications like overtake assist and platooning strictly require perception horizons of 150-200+ meters, VoD is structurally out of specification for these tasks by design. Its popularity in literature does not translate to representativeness for real-world driving scenarios, making it fundamentally unsuitable for training long-range models.

	car	**pedestrian**	**cyclist**
# objects	26949 (21.9%)	26587 (21.6%)	10800 (8.8%)
# unique obj	423 (22.6%)	380 (20.3%)	183(9.8%)
% moving	7.2%	73.2%	96.1%

Table 4.2: Subset of class distribution of the VoD dataset from the original source [18]

Furthermore, during the experiments, it was noticed that some labels within this already limited range are entirely missing, causing potential degradation in network performance. An example of this is shown in figure 4.1, where two consecutive frames, with an accumulation of 5 frames, are shown from the Bird Eye View (BEV), and it is highlighted with a dashed red line the bounding box of the car missing, same vehicle present in the consecutive frame.

Consequently, researchers who evaluate their novel network architectures exclusively on the VoD dataset often draw optimistic conclusions that simply do not transfer to real-world, high-speed autonomous driving scenarios. Optimizing an architecture to achieve high precision within a dense, 50-meter urban bubble completely ignores the radar's physical capacity and the vehicle's kinematic

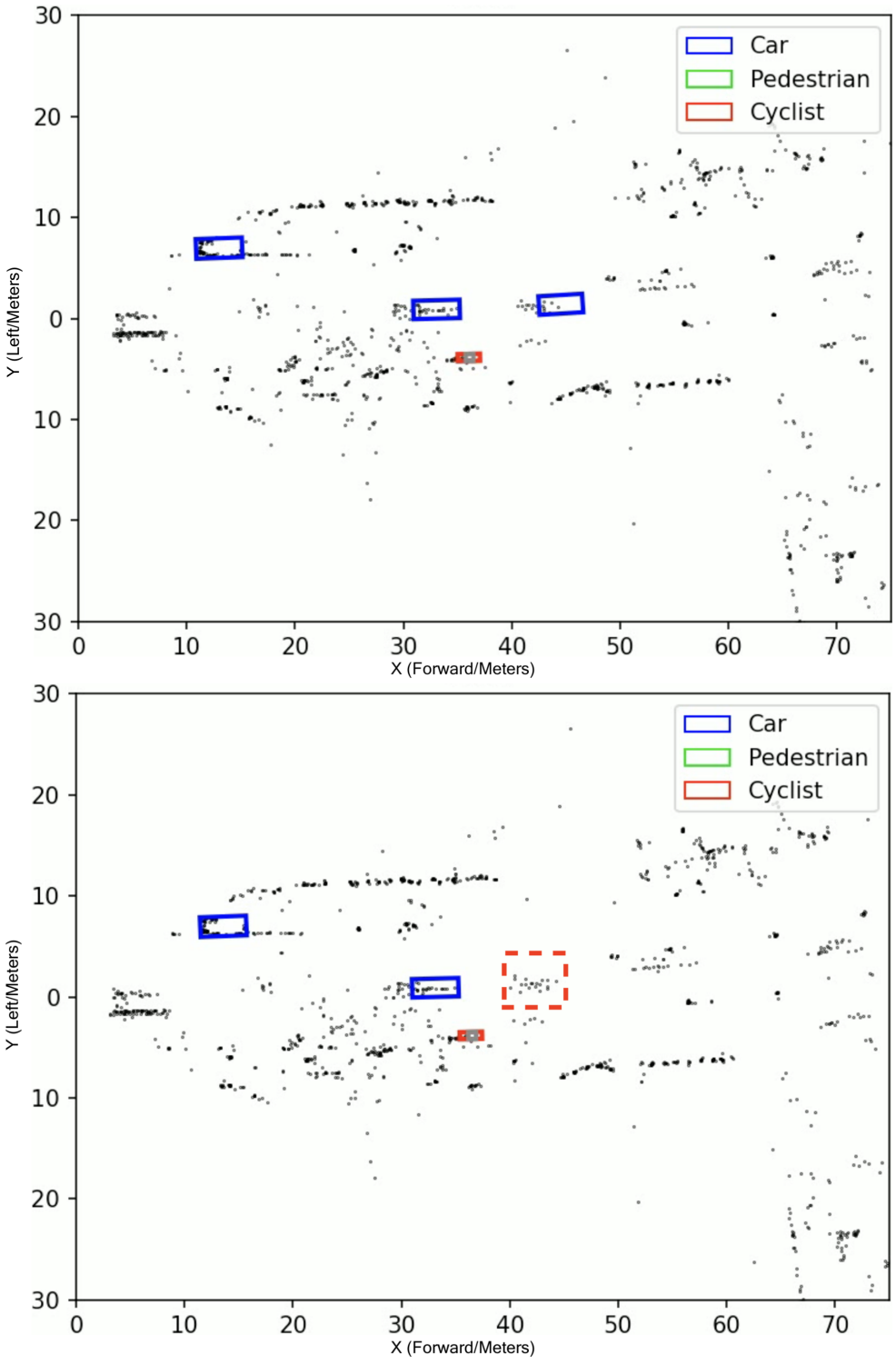


Figure 4.1: Missing label on 2 consecutive frames

requirements.

In summary, VoD allows for the measurement of VRU detection and multi-class classification within a dense, short-range urban bubble, but it completely fails to provide a platform for evaluating the high-speed, long-range detection capabilities required by our target applications. Despite this critical limitation for our target applications, VoD is intentionally retained in the initial experimental phase strictly to conduct foundational studies on the PointPillars architecture, ensuring that our baseline implementations and ablation studies remain directly comparable with the established academic literature.

4.3 NuScenes

The nuScenes dataset is a pioneering, large-scale dataset for autonomous driving that provides complete 360° sensor coverage. The dataset consists of 1,000 driving scenes, each lasting 20 seconds, resulting in 40,000 annotated keyframes sampled at 2 Hz [22]. Its radar network consists of five long-range Continental ARS 408-21 radar sensors [22]. These sensors operate at 77 GHz with a 13 Hz capture frequency, offering a hardware detection range of up to 250 m, while the dataset's ground truth annotations extend up to a maximum range of roughly 141 m[36] from the ego-vehicle. They independently measure distance and radial velocity within a single cycle, achieving a velocity accuracy of ±0.1 km/h [22]. While the sensor suite is highly robust, the provided radar data lacks vertical elevation, functioning primarily as a sparse 2D representation [22]. Unlike VoD, nuScenes provides a ground truth annotation range that approaches our requirements. Extending up to roughly 141 m, it represents a complete dataset for 2+1D radars that bridges the gap between urban datasets and the long-range requirements of the tasks evaluated, despite lacking elevation data.

Ultimately, nuScenes permits a robust baseline measurement of bounding box accuracy and velocity estimation at medium-to-long ranges for 2+1D radar data. However, the hardware's inherent lack of elevation information limits its applicability for 3D detection tasks.

4.4 MAN TruckScenes

The MAN TruckScenes dataset addresses the specific perceptual requirements of autonomous long-haul trucking. It provides an unprecedented radar configuration composed of six Continental ARS540CES 4D radar sensors. It is important to note, as specified in the dataset's official documentation, that the sensor variant and specific setup utilized do not correspond to a standard, approved series version by Continental; therefore, its performance may be limited or differ compared to commercial series applications [36]. Operating at 76 GHz and 20 Hz, these sensors collectively deliver a full 360° field of view [36]. The dataset contains 747 diverse scenes, providing 30,000 annotated samples across different environmental conditions. TruckScenes stands as the largest dataset providing 4D radar data with annotated 3D bounding boxes. Most importantly for this study, TruckScenes successfully resolves the annotation range bottleneck. To accommodate the necessity for long planning horizons, the dataset's 3D bounding box annotations explicitly emphasize long-range detection, extending up to more than 230 meters [36]. This makes TruckScenes the only dataset in this evaluation that fully meets and exceeds the 150-200+ meter requirement for overtake assist and platooning, structurally justifying its selection as the primary training domain for high-speed applications. However, more than 70% of scenes are on highways [36]. Note that the selection of these datasets reflects the state-of-the-art available during the active experimental phase of this work; very recent releases, such as the TruckDrive dataset[37], were published subsequently and are therefore left for future evaluation.

4.4.1 Annotation Sparsity and Filtering

While the TruckScenes dataset provides a massive volume of global annotations, not all labeled bounding boxes are practically usable for training a radar-based object detector. For this study, which focuses specifically on the frontal perception field, a rigorous filtering process was applied to the ground truth data. Table 4.3 illustrates the drastic reduction in available annotations when filtering the dataset to include only bounding boxes that contain at least one valid radar reflection, and subsequently, only those located within the frontal region of interest.

This data strongly highlights the severe sparsity issue inherent to automotive radar. When evaluating the entire 360-degree dataset, over half of the labeled objects contain absolutely zero

Class	Total Annotations	≥ 1 Radar Point	≥ 1 Point & Frontal Area
Car (Total)	322,191	144,116	79,846
Car (Moving)	281,764	123,203	71,579
Car (Stopped)	40,427	20,913	8,267
Pedestrian	10,457	3,141	1,959
Bicycle	1,151	434	268
Total	**333,799**	**147,691**	**82,073**

Table 4.3: Degradation of usable annotations in the TruckScenes dataset after applying visibility and spatial filters

radar points, meaning the sensor completely failed to capture a reflection from those targets due to occlusion, long range, or low RCS. Furthermore, when restricting the field of view to the frontal area relevant to this study, the number of effectively usable annotations drops to less than 25% of the original dataset volume. For Vulnerable Road Users (VRUs) like pedestrians, the drop is even more critical, plummeting from 10,457 global labels to just 1,959 usable frontal samples.

This dramatic reduction in viable training data provides clear insight into why neural networks operating on radar point clouds struggle to achieve the performance metrics seen in LiDAR-based architectures. A bounding box devoid of interior points offers zero geometric or semantic information for a network to learn from. Consequently, while a dataset may boast hundreds of thousands of annotations on paper, the actual volume of information-rich data available to train a robust radar detector is significantly smaller, further emphasizing the need for highly optimized, data-efficient neural architectures.

Therefore, TruckScenes allows for the critical measurement of true 3+1D long-range detection and the evaluation of models under high-speed highway conditions, but its environmental bias means it does not permit a balanced measurement of urban VRU detection or dense static clutter resolution.

4.4.2 Z-Axis Misalignment and Multipath Artifacts

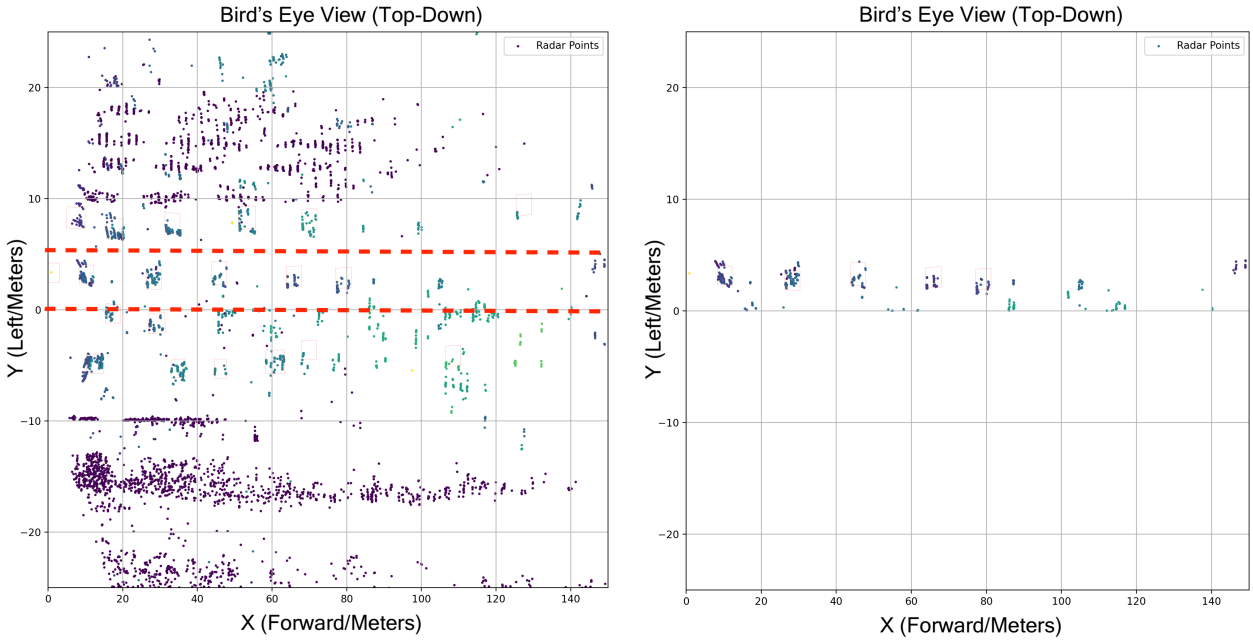
To practically illustrate the previously discussed phenomena of multipath reflections and the inherent limitations in radar elevation resolution, a visual analysis was conducted on the MAN TruckScenes dataset.

Initially, a dense, highly cluttered highway scene was examined from the Bird's Eye View (BEV), which includes heavy traffic and continuous environmental structures like guardrails. To isolate the behavior of radar reflections on distinct targets without the interference of adjacent lanes and background clutter, a specific longitudinal "stripe" containing a single line of vehicles was filtered from the overarching point cloud as visualised in Figure 4.2a.

When observing this isolated stripe in the BEV projection, the spatial alignment between the radar points and the ground truth bounding boxes appears highly accurate in the 2D x-y plane. The radar reflections cluster cohesively within the lateral and longitudinal boundaries of the targeted cars as displayed in Figure 4.2b. We also observe how, in this isolated stripe, the bounding boxes have no points on their sides, which is fundamental for the following analysis, as it allows to evaluate the z-axis misalignment without the interference of points from other vehicles.

However, projecting this exact same filtered subset into a lateral side view (x-z plane) reveals severe geometric distortions as shown in Figure 4.3. While the red ground truth bounding boxes are correctly positioned at the physical road elevation (roughly between $z = 0$ and $z = 1.5$ meters), the associated radar points exhibit massive vertical scattering. A significant portion of the reflections register well below ground level, reaching depths of $z = -2$ meters, while other points scatter high above the actual roofs of the vehicles.

This visualization perfectly encapsulates the theoretical hardware limitations detailed earlier. The severe vertical misalignment is the direct physical manifestation of the radar's coarse elevation angular resolution relative to LiDAR, compounded by aggressive multipath propagation. The underground "ghost" points are classic signatures of the radar signal bouncing off the asphalt before returning to the receiver, which artificially inflates the time-of-flight and alters the perceived angle of arrival. Consequently, while the 2D BEV footprint remains reliable for lateral and longitudinal localization, this extreme variance in the z-axis presents a major challenge for neural networks attempting to regress accurate 3D bounding box heights and elevations purely from radar point



(a) BEV of a TruckScenes frame with the filtered stripe of vehicles highlighted in red

(b) BEV of the isolated stripe

Figure 4.2: Comparison of the BEV of a TruckScenes frame with the filtered stripe of vehicles highlighted in red and the BEV of the isolated stripe

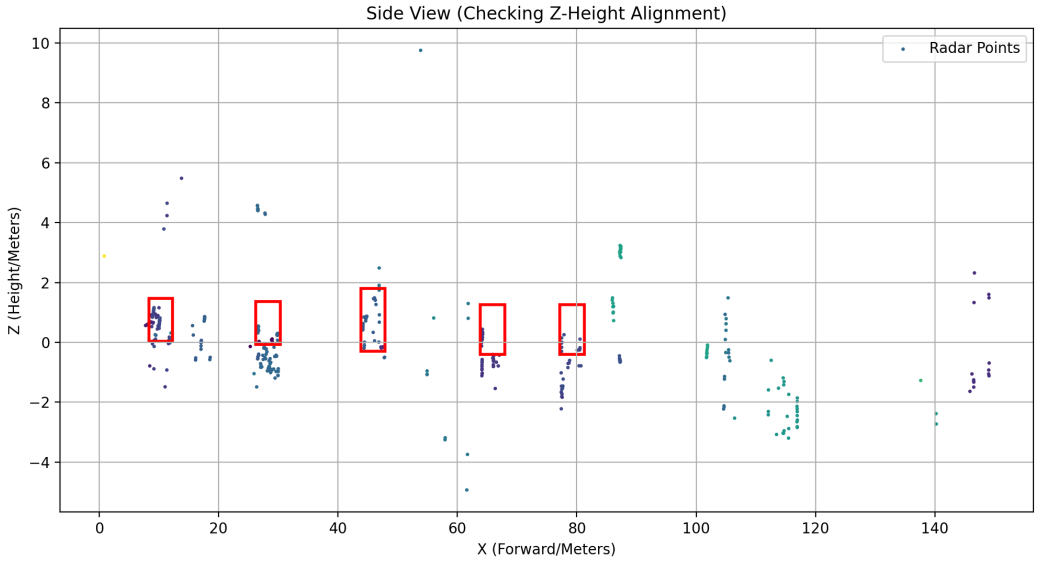


Figure 4.3: Side view of the isolated stripe, highlighting the severe vertical misalignment of radar points relative to ground truth bounding boxes

clouds.

Academic Publishing Bias vs. Real-World Applicability

Given that datasets like nuScenes and TruckScenes provide realistic annotation ranges that actually meet the requirements of advanced autonomous functions, their relatively lower adoption rate in recent architectural literature compared to VoD warrants explanation. This discrepancy is largely driven by an inherent academic publishing bias. Because nuScenes and TruckScenes feature significantly longer ranges, higher vehicle speeds, and far more complex environmental clutter, they represent fundamentally more difficult detection tasks.

Consequently, neural networks evaluated on these extensive datasets inevitably yield much lower Mean Average Precision (mAP) scores. In a research landscape heavily motivated by establishing new "State-of-the-Art" metrics, datasets that artificially restrict the difficulty of the task—such as VoD, with its 50-meter annotation crop—are often preferred. High mAP scores on constrained datasets make architectural contributions appear highly successful and thus more appealing for publication. However, this thesis argues that chasing high metrics on out-of-specification datasets is counterproductive for industrial deployment, which is why our evaluation prioritizes the harder, lower-scoring, but vastly more representative data provided by TruckScenes and our custom long-range logs.

4.5 Proprietary Custom Datasets

As established in the preceding evaluations, while public benchmarks provide essential foundations for pre-training, none perfectly encapsulate the specific hardware configurations and operational domains targeted in this study. To actively overcome these limitations and evaluate the architectures in a targeted, real-world environment, a proprietary data collection and annotation pipeline was developed.

The setup included three radars mounted to cover the front-left, front-right and center. While the frontal radar was mounted at the center of the vehicle parallel to the ego direction, the external radars were shifted by 0.5 meters and rotated at an an angle between 17.5° and 19° outward. Each dataset contained a single scene (or sequence) that included both urban and extra-urban scenarios

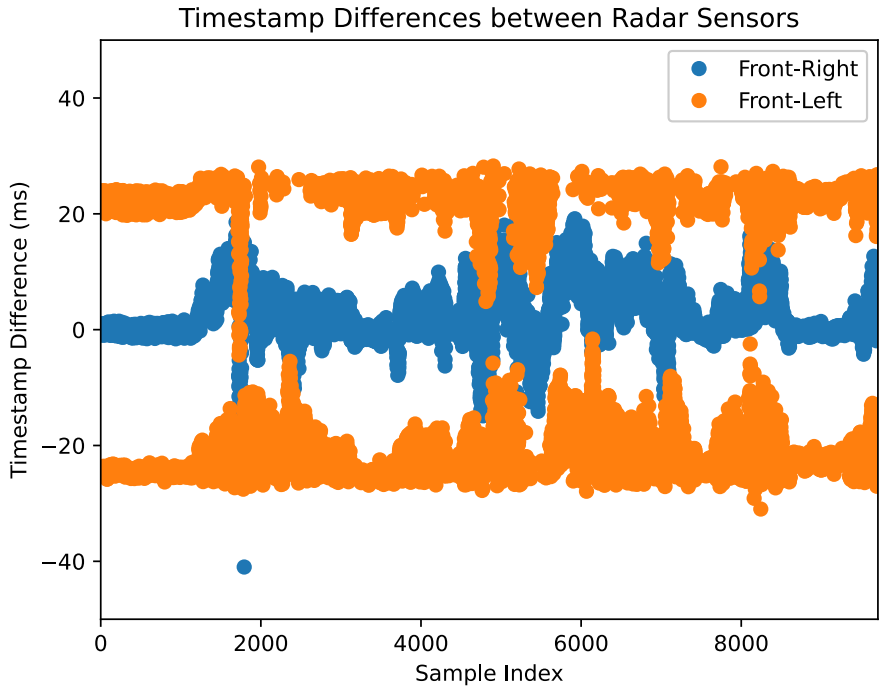


Figure 4.4: Comparison of the timestamps of the different radar frames using a closest frame association. The center radar is used as reference

recorded in Modena in Italy. It included both moving and stopped vehicles, with a small presence of pedestrians and cyclists. Both datasets include odometry of the ego vehicle, lidar and camera sensors as well. The sensors mounted in the car are not synchronized; to address this issue, the data was synchronized in post-processing using timestamps. To associate the different radar frames with the same timestamp, the closest frame was initially selected as shown in Figure 4.4.

However, this naive approach led to frames that were completely removed and some others that were duplicated especially when the difference in timestamps was around $0.5 \cdot \frac{1}{frequency}$, where the frequency in this case is 20Hz. To overcome this issue, a limitation of a maximum one association for each frame was established. As shown in Figure 4.5, this improved approach allowed to keep all the frames and avoid duplicates, while keeping a consistent timestamp difference between the frames. This difference was subsequently compensated using the ego-vehicle odometry.

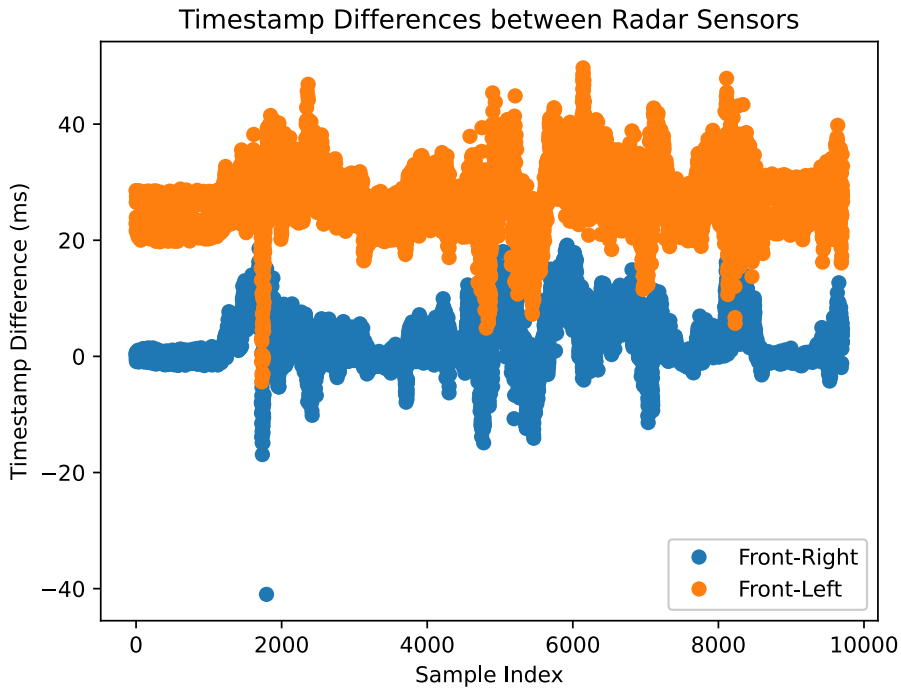


Figure 4.5: Comparison of the timestamps of the different radar frames using a one-to-one frame association. The center radar is used as reference

4.5.1 2+1D Radar

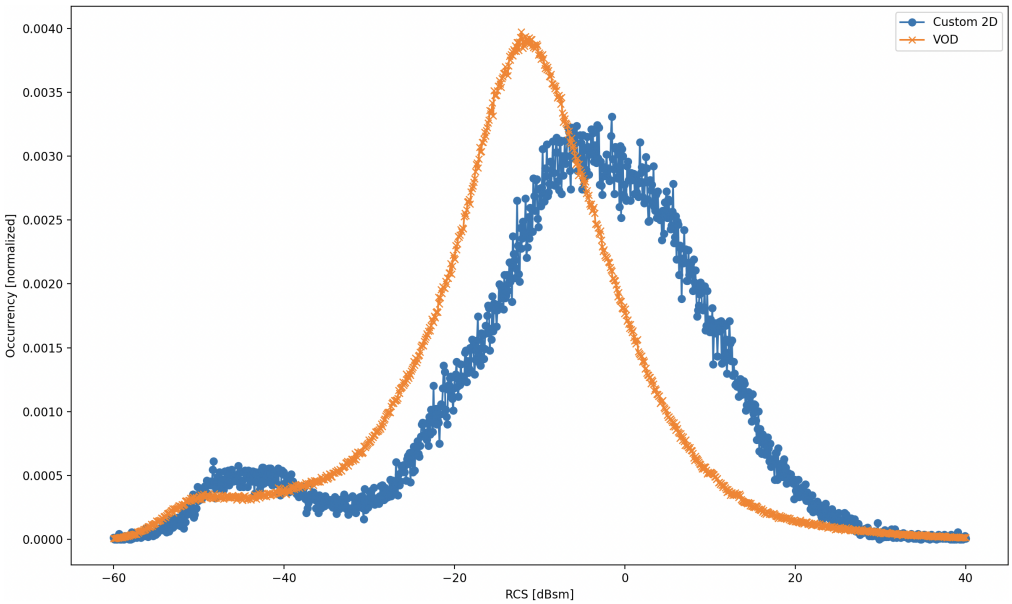
The first dataset was captured using Continental ARS 408-21 radars, same sensors of NuScenes dataset. The data is unlabeled and it was used for visually evaluating how the different solutions were generalizing on our setup.

As a result, this unannotated 2+1D dataset allows for the qualitative measurement of sensor signal behavior (such as CFAR thresholding differences) and the visual evaluation of zero-shot domain generalization, but it lacks the ground truth necessary to permit any quantitative benchmarking of network precision.

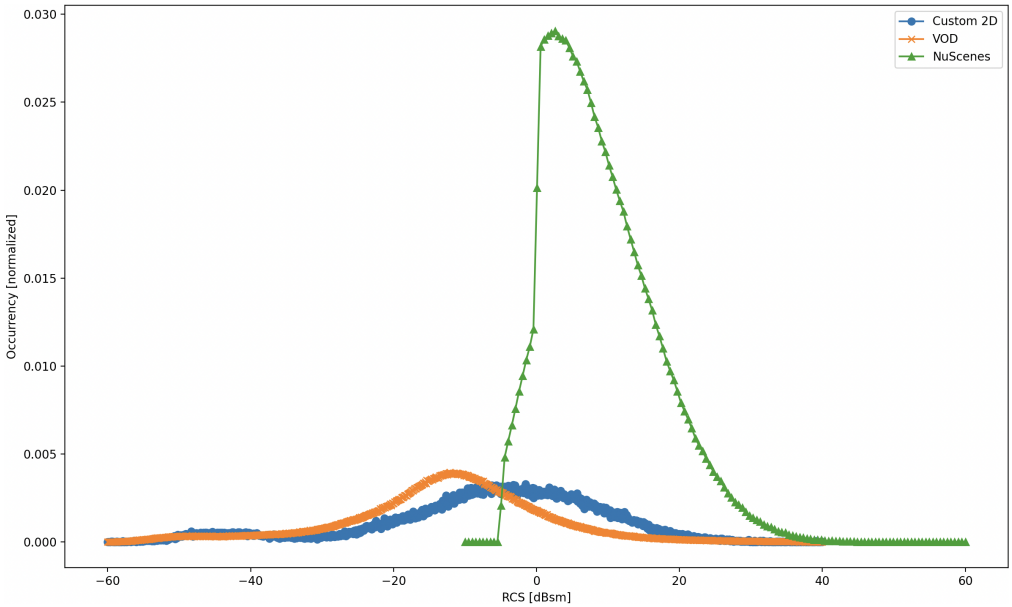
RCS Distribution

During the exploratory data analysis phase, a notable discrepancy was observed regarding the Radar Cross Section (RCS) distributions across the evaluated datasets. Although the custom 2D dataset relies on a Continental ARS 408 radar—the exact same hardware utilized in the nuScenes dataset—the statistical distribution of their respective RCS values diverges significantly. As illustrated by the figures in 4.6, the nuScenes RCS distribution exhibits a steep, right-skewed "half-Gaussian" shape, abruptly truncating values below 0 dBsm. Conversely, the custom 2D dataset demonstrates a full Gaussian-like distribution that captures a wide spectrum of negative RCS values. Interestingly, this bell-shaped distribution closely mirrors the one found in the View-of-Delft (VoD) dataset, despite VoD employing an entirely different sensor hardware (ZF FRGen21).

This phenomenon can be attributed to differences in the low-level signal processing pipelines rather than the physical antennas. Automotive radars rely heavily on proprietary Constant False Alarm Rate (CFAR) thresholds before broadcasting data to the vehicle's network. It is highly probable that the nuScenes data collection pipeline applies a strict heuristic filter or a specific firmware configuration that aggressively clips or discards targets with low RCS values, prioritizing a sparse but highly confident point cloud. In contrast, the custom setup (much like the VoD configuration) appears to operate with a lower detection threshold or an alternative normalization scheme. By allowing the sensor to output weaker reflections from less reflective targets or background elements, the system preserves the lower-energy data. This retention of negative RCS values forms the distinct left tail of the Gaussian curve, ultimately providing a denser, more unfiltered representation of the



(a) Comparison of the RCS distributions of the VoD and custom datasets



(b) Comparison of the RCS distributions of the three datasets

Figure 4.6: Comparison of the distributions of the RCS across the dataset

radar environment.

4.5.2 3+1D Radar

Another dataset was used with the Continental ARS540CES radars, the same sensors as the TruckScenes dataset. This data consisted of 9706 unlabeled keyframes for each radar. However, we chose to label a subset of this data to quantitatively evaluate the results.

RCS distribution

A parallel analysis conducted on the 3+1D custom dataset reveals a notably different pattern compared to the 2+1D sensor logs. When examining the Radar Cross Section (RCS) of the custom 3+1D radar, as shown in the figure 4.7, its statistical distribution aligns closely with the profile observed in the TruckScenes dataset. As illustrated in the corresponding data, both distributions exhibit a distinct right-skewed, "half-Gaussian" shape. However, unlike the aggressive 0 dBsm cutoff observed in the nuScenes data, both the custom 3+1D radar and the TruckScenes sensors (Continental ARS540CES) demonstrate a sharp truncation much lower on the spectrum, at approximately -20 dBsm. This structural similarity suggests that the internal Constant False Alarm Rate (CFAR) thresholds and digital signal processing pipelines of this newer generation of high-resolution radars are configured similarly, allowing for a greater retention of low-energy reflections while still maintaining a hard cutoff for extreme noise.

Furthermore, it is important to contextualize these distributions within the specific driving environments captured by each dataset. While the custom 3+1D logs are heavily focused on urban environments, the TruckScenes dataset is characterized by a high percentage of highway driving scenarios. Highway environments inherently lack low-RCS targets, such as pedestrians and cyclists, and are instead dominated by massive, highly reflective metallic structures, including commercial trucks, trailers, and continuous metallic guardrails. This environmental bias naturally increases the average RCS of the captured points, contributing to a denser and slightly extended right tail in the TruckScenes distribution. Consequently, the similarities in the lower bounds confirm the hardware and firmware alignments between the sensors, while the subtle variances in the higher RCS spectrum accurately reflect the distinct operational domains of the datasets.

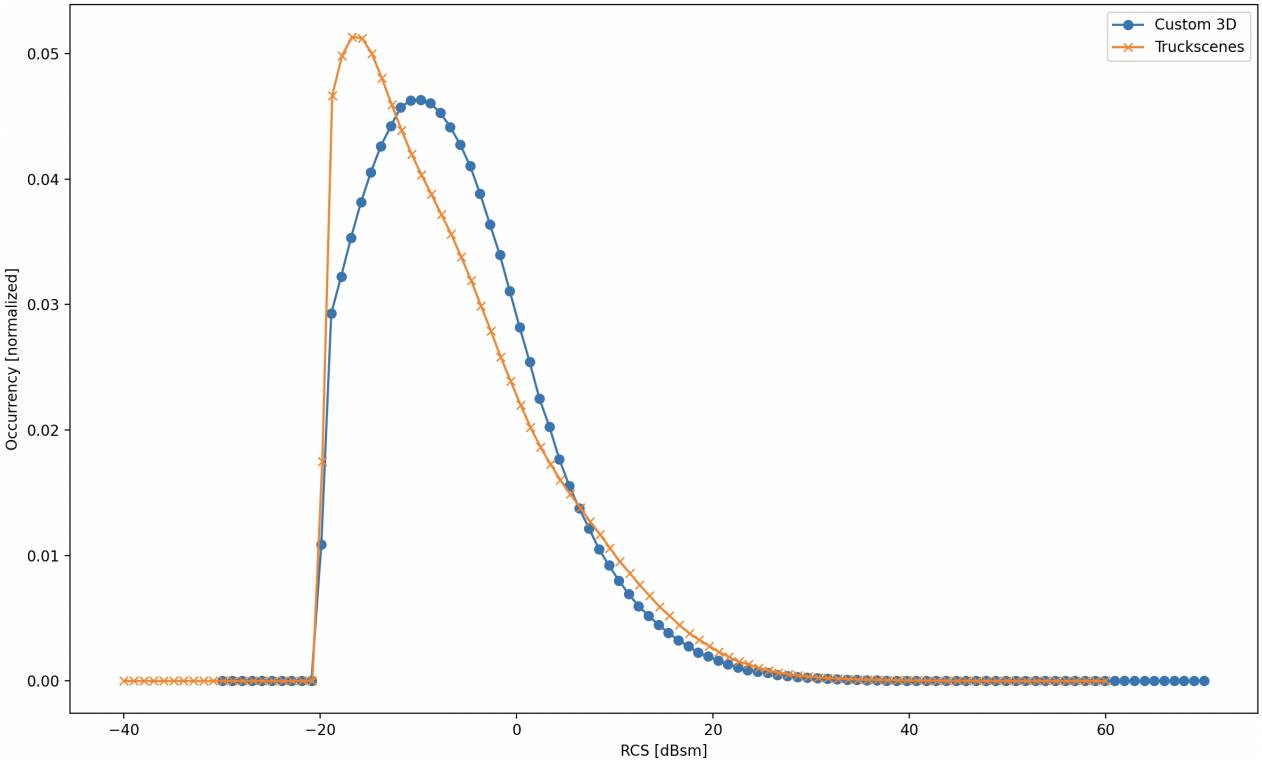


Figure 4.7: Comparison of the distributions of the RCS between the TruckScenes and custom 3+1D datasets



Figure 4.8: Example of the manually annotated dataset with radar partially occluded while stationary behind a vehicle

Semi-Automated Labeling and Tracking Pipeline

To efficiently annotate the data collected from the custom sensor setup, a semi-automated pre-annotation and tracking pipeline was implemented. Initially, inference was performed on the raw logs using the best-performing model weights previously trained on the TruckScenes dataset. The resulting bounding box predictions were then processed through a custom tracking module. This module assigns consistent instance IDs across consecutive scans, allowing for the temporal interpolation of missed detections and the stabilization of bounding box geometries. The core of this pipeline relies on an ego-motion compensated Kalman Filter for state estimation and the Hungarian algorithm for data association.

State Estimation with the Kalman Filter The Kalman Filter operates under the assumption of a linear, constant-velocity physical model. For each tracked object at a time step k, the internal state vector is defined as $x_k = [x, y, v_x, v_y]^T$, representing the 2D spatial coordinates and the absolute target velocities on the ground plane.

During the prediction step, the filter projects the state forward in time based on the elapsed time Δt. The state transition is defined as follows, where $x_{k|k-1}$ denotes the predicted state at time step k given the estimation from the previous time step $k-1$:

$$x_{k|k-1} = Fx_{k-1|k-1} + Bu_k \tag{4.1}$$

where F is the state transition matrix that applies the kinematic translation. To actively compensate for the ego-vehicle's movement between frames, $u_k = [-v_{ego_x}, -v_{ego_y}]^T$ acts as a control input vector applied through the control matrix B. Furthermore, to account for the ego-vehicle's turning dynamics, the state vector and its covariance matrix P are multiplied by a rotation matrix R_{rot} derived from the vehicle's yaw rate prior to the prediction step. This ensures the target's trajectory remains correctly aligned within the ego-centric coordinate frame.

During the update step, the filter incorporates new sensor measurements. The measurement vector z_k consists of the 2D center of the detected bounding box and its estimated velocity. To extract this velocity, a tight 0.5-meter spatial mask is applied around the detection center to isolate the radar points belonging strictly to the target, from which the median absolute radial velocity components are calculated.

The measurement noise covariance matrix R is specifically tuned for radar characteristics: it heavily trusts the spatial position provided by the bounding box, while allowing higher variance for the velocity due to the inherent noise of radar point clouds. In scenarios where a bounding box is detected but contains zero valid radar points, an "infinite noise" heuristic is dynamically applied to the velocity components of R. This forces the filter to rely entirely on its internal kinematic prediction rather than incorporating a faulty zero-velocity measurement from an empty box.

Data Association via the Hungarian Algorithm To continuously link new detections to existing tracks across sequential frames, the pipeline employs the Hungarian algorithm to solve the bipartite matching problem. First, a cost matrix C is computed between all predicted track positions (x_t, y_t) and all new detection positions (x_d, y_d) using the 2D Euclidean distance:

$$C_{t,d} = \sqrt{(x_t - x_d)^2 + (y_t - y_d)^2} \tag{4.2}$$

The Hungarian algorithm processes this matrix to find the optimal global assignment that minimizes the total sum of the distances. To prevent erroneous associations between distant objects,

a spatial gating threshold of 3.0 meters is applied; any assigned match exceeding this threshold is rejected and treated as unmatched.

Track Lifecycle Management Unmatched detections automatically initialize new tracks with an initial velocity derived from their enclosed radar points. Conversely, unmatched tracks are not immediately discarded. To counteract the severe sparsity and flickering typical of automotive radar, tracks are allowed to "coast" purely on their kinematic predictions for up to 15 consecutive frames (provided they do not exit the defined spatial boundaries). This prolonged memory allows the pipeline to seamlessly interpolate bounding boxes through temporary occlusions and signal dropouts, generating a dense, continuous ground truth sequence.

Following an initial phase of manual verification and correction using the SUSTechPoints platform, a preliminary dataset comprising 529 annotated frames was established. This initial subset was strategically utilized to fine-tune the base TruckScenes model, significantly improving its domain adaptation to the specific characteristics of the custom 3+1D radar data. Leveraging the enhanced bounding box accuracy of this intermediate model, the semi-automated tracking and annotation cycle was re-executed on a larger batch of unseen data. After passing these new predictions through the Kalman filter and applying a final manual refinement, the dataset was expanded by an additional 2,900 frames. This iterative methodology ultimately yielded a robust final custom dataset of 3,429 frames, complete with high-quality ground truth bounding boxes fully labeled with persistent track IDs. The labels are provided in the frontal part of the vehicle, covered by at least one radar, and the bounding boxes are provided in the ego-vehicle coordinate system, an example of the dataset is shown in figure 4.8.

Annotations

The composition of the fully annotated dataset heavily reflects its specific urban operational domain in Modena. The recorded sequence captures a mix of dynamic scenarios: for approximately 1,000 samples, the ego-vehicle is stationary along the street while surrounding traffic continues to flow, whereas the remainder of the dataset captures standard, active driving dynamics moving with the traffic.

Consequently, the class distribution exhibits a pronounced imbalance, which is typical for urban

environments but particularly accentuated in this custom log. Passenger vehicles overwhelmingly dominate the dataset, accounting for 55,186 total annotations distributed across 408 unique tracked instances. It is important to note that a substantial fraction of these bounding boxes belongs to static, parked cars lining the urban streets (specifically comprising 73% of the total annotations), which introduces a heavy static-object bias into the training and evaluation phases. The static state classification was systematically determined by applying a velocity threshold of 0.5m/s to the median absolute radial velocity of the points enclosed within each bounding box.

This massive prevalence of static vehicles introduces a significant perception challenge. As detailed in the theoretical background, standard radar CFAR algorithms often suppress stationary reflections to aggressively filter out environmental clutter. Consequently, non-moving targets are inherently harder to detect and geometrically reconstruct than moving ones due to the absence of a prominent Doppler signature.

Furthermore, this heavy static-car bias represents a stark domain shift compared to the TruckScenes dataset, which is inversely characterized by high-speed highway scenarios dominated by moving vehicles. Highlighting this opposing distribution is crucial to the experimental design of this work: this custom dataset is subsequently utilized to fine-tune the base model weights initially trained on TruckScenes. This fine-tuning phase effectively forces the neural network to undergo a drastic domain adaptation, shifting its learned feature extraction from detecting dynamic targets in open environments to detecting dense, stationary passenger vehicles in highly cluttered urban streets.

Conversely, Vulnerable Road Users and large commercial vehicles are significantly underrepresented. The dataset contains only 839 pedestrian annotations (representing just 4 unique individuals) and completely lacks cyclist instances. Similarly, heavy vehicles are scarce, with 563 truck annotations (originating from 7 unique IDs) and 136 bus annotations tracking a single unique instance. A comprehensive statistical summary of the dataset's ground truth is provided in Table 4.4.

Given that cars constitute over 97% of the total labeled instances, deeply evaluating the spatial and orientational properties of this specific class is crucial for understanding the network's performance boundaries. The distribution of these annotations provides valuable insights into the operational range of the custom 3+1D radar setup and the geometry of the urban environment recorded.

The longitudinal distribution (x), illustrated in Figure 4.9, highlights the depth perception of the dataset. There is a massive concentration of annotations in the near field (between 10 and 50 meters),

Class	Total Annotations	Unique Instances (IDs)
Car	55186	408
Pedestrian	839	4
Cyclist	0	0
Truck	563	7
Bus	136	1
$\sum$	**56724**	**420**

Table 4.4: Class distribution and unique object counts for the annotated custom 3+1D radar dataset

which naturally corresponds to immediate moving traffic and nearby parked vehicles. Notably, the histogram also reveals distinct secondary peaks at significantly longer distances (between 130 and 180 meters). This multimodal tendency illustrates the ARS540CES radar's capability to detect targets and maintain track IDs over long, straight urban corridors, successfully capturing distant vehicles even in cluttered scenarios.

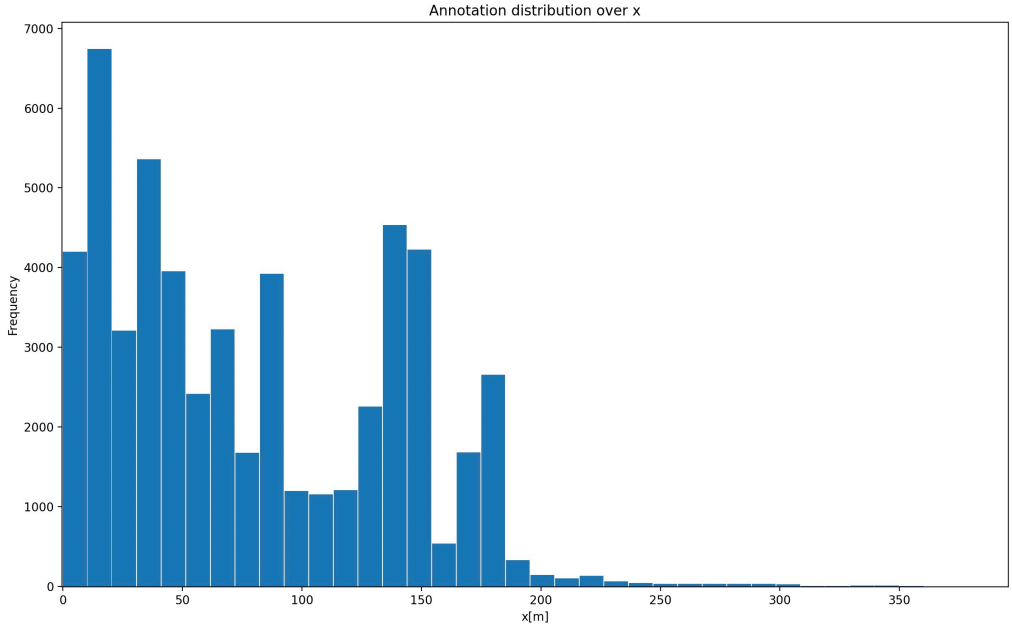


Figure 4.9: Distribution of car annotations over longitudinal distance (x)

The lateral distribution (y), presented in Figure 4.10, provides insight into the road width and the lateral positioning of targets relative to the ego-vehicle. The annotations are predominantly clustered

between -20 and +30 meters. The slight asymmetry and the high volume of targets immediately around the 0-meter axis (the ego-vehicle's lane) reflect the multi-lane nature of the recorded streets. The spread across this axis accurately captures a mix of oncoming traffic in adjacent lanes and continuous rows of vehicles parked parallel to the road edges.

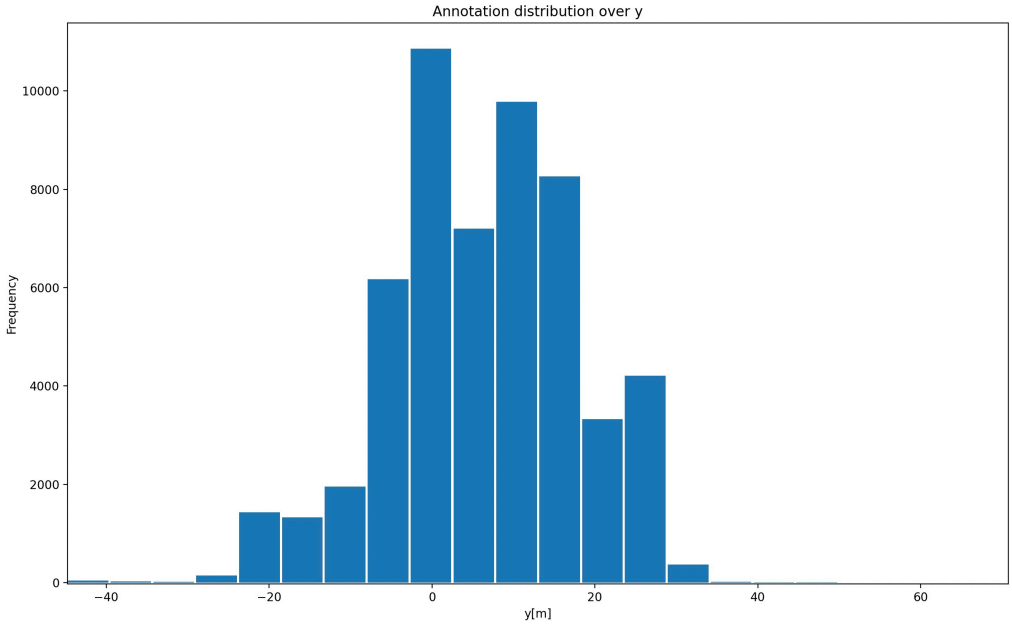


Figure 4.10: Distribution of car annotations over lateral distance (y)

Figure 4.11 illustrates the absolute Euclidean range of the car annotations from the sensor. Because the longitudinal distance (x) is inherently much larger than the lateral distance (y) for front-facing sensors, this absolute range distribution closely mirrors the longitudinal profile. It confirms that the dataset contains a robust volume of annotations spanning from the immediate vicinity up to 200 meters, with a long tail extending up to 300 meters, effectively validating the long-range detection characteristics of the 4D radar hardware.

Finally, the distribution of the yaw angles (rotation), shown in Figure 4.12, strongly corroborates the specific urban layout and the static nature of the parked cars. The histogram displays massive, highly localized peaks at specific angles rather than a uniform spread. In the context of urban driving, this sharp angular quantization indicates that the vast majority of labeled vehicles are strictly aligned parallel to the ego-vehicle's longitudinal axis. This is perfectly consistent with a scenario dominated by cars moving in parallel lanes (same or opposite directions) and roadside parking. The relative absence of annotations at intermediate angles confirms that cross-traffic at

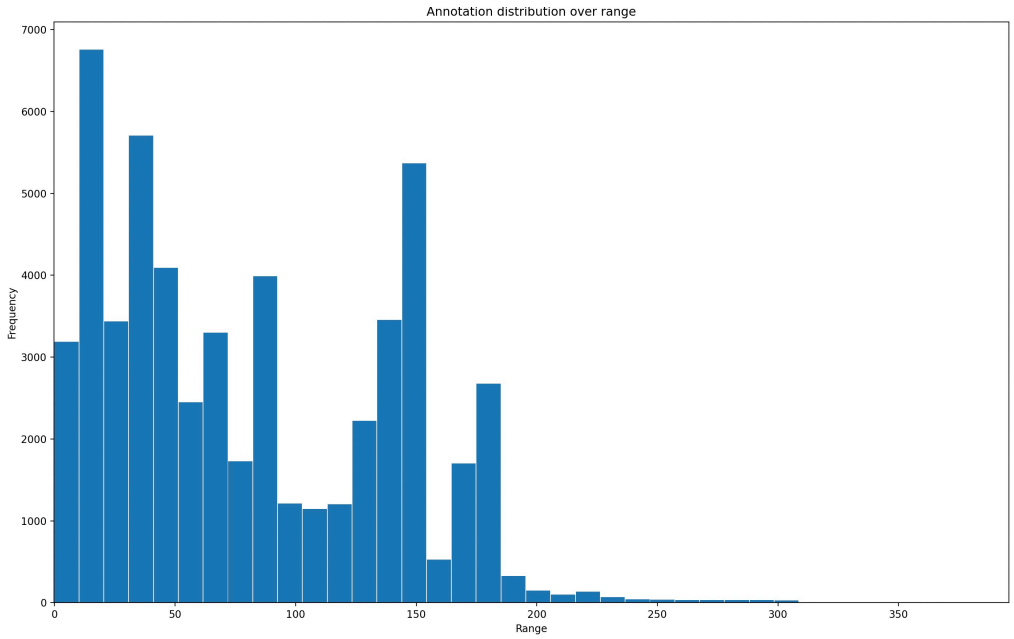


Figure 4.11: Distribution of car annotations over absolute Euclidean range

intersections or turning vehicles represent only a small minority of the recorded data.

Ultimately, this fully annotated custom 3+1D dataset allows for the rigorous measurement of domain adaptation on dense, static urban clutter but its extreme class imbalance and lack of high-speed targets do not permit a comprehensive evaluation of dynamic highway scenarios. However, it excels in providing a highly representative urban dataset for evaluating the detection of stationary vehicles, which is a critical challenge for automotive radar perception.

4.5.3 SUSTechPoints

To facilitate the manual verification and refinement phase of the dataset creation, the open-source platform SUSTechPoints (Portable 3D Point Cloud Interactive Annotation Platform System) was employed. The platform simultaneously displays the point cloud in a 3D perspective view alongside three distinct orthographic projections (top, front, and side views) [38], which was crucial for precisely localizing bounding box errors and manually adjusting their dimensions and yaw angles. While SUSTechPoints natively offers intelligent interactive operations such as box auto-fitting and smart initialization [38], these algorithmic features proved almost unusable in practice; the inherent sparsity of the radar point clouds lacked the geometric density required for automated fitting tools to

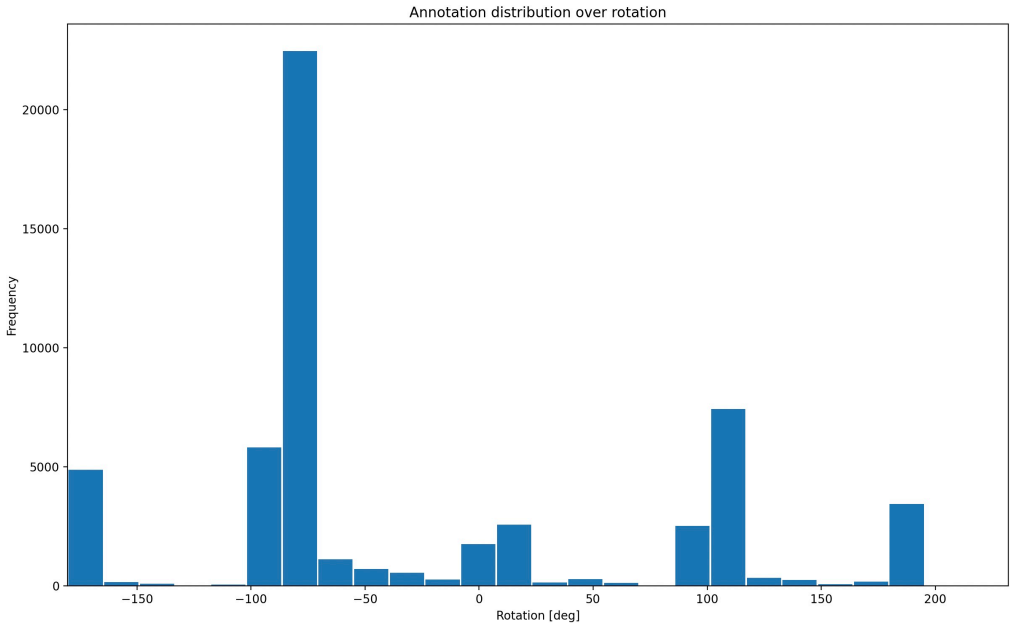
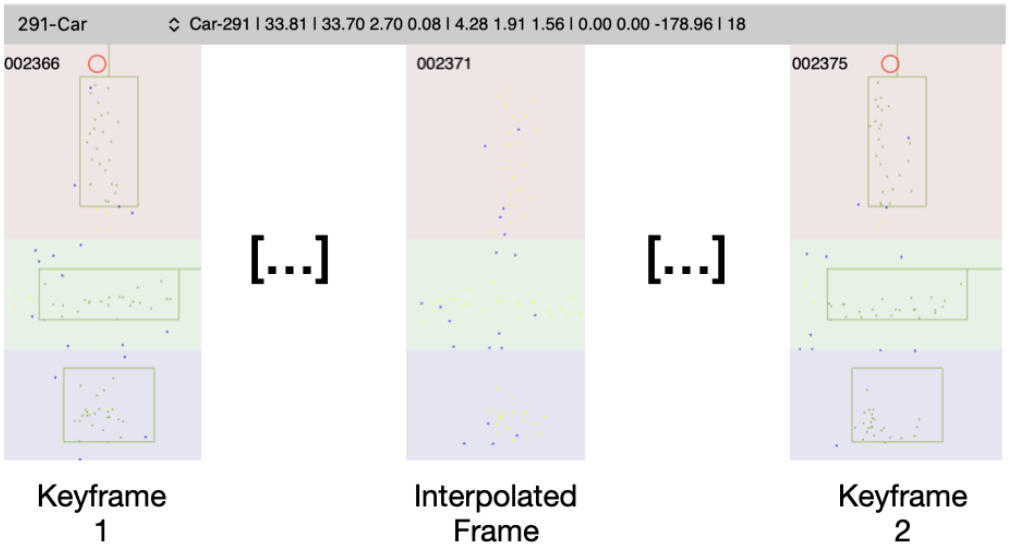


Figure 4.12: Distribution of car annotations over rotation (yaw angle)

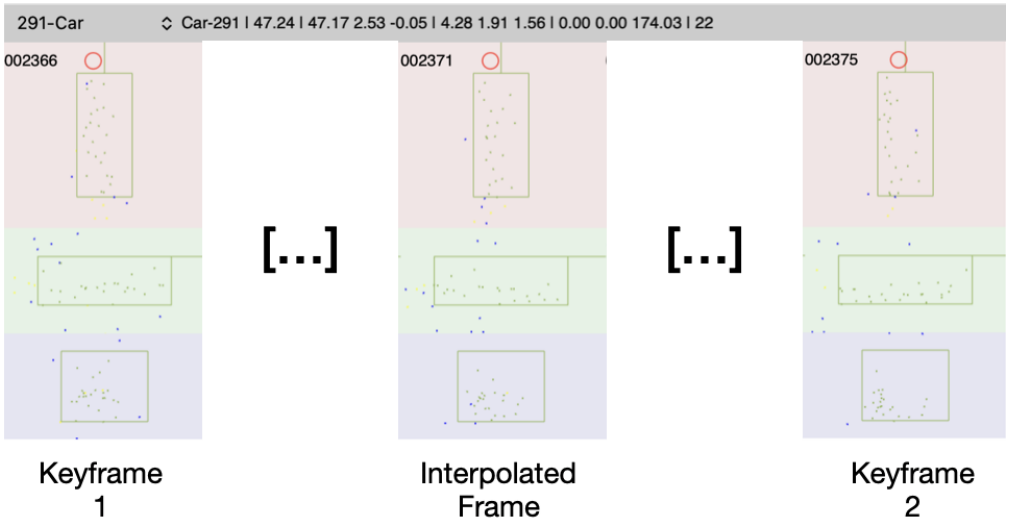
function reliably. To overcome the difficulty of identifying exact object shapes and boundaries solely from sparse radar reflections, the corresponding LiDAR point clouds were loaded into the platform as a visual reference. Because the LiDAR and radar sensors were not hardware-synchronized, the LiDAR point clouds first required ego-motion compensation to spatially align them with the radar timestamps. Consequently, the most valuable and heavily utilized feature of the platform was its support for sequential annotation interpolation. By leveraging the consistent instance IDs assigned by the Kalman Filter tracking module, the tool's interpolation function allowed for rapid cross-frame error checking, the seamless estimation of missed detections, and the efficient finalization of the ground truth track IDs across the 3,429 annotated frames. An example of the interpolation tool is shown in figure 4.13 with all the orthogonal views of the single box.

4.6 Summary and Rationale for Custom Dataset Creation

To conclude this chapter, it is essential to synthesize why the creation and manual annotation of a proprietary dataset was a requirement for this thesis, rather than relying exclusively on public benchmarks. As evaluated in the preceding sections, public datasets provide essential foundations for pre-training, but none perfectly encapsulate both the specific hardware configurations and the



(a) Before interpolation



(b) Interpolated output

Figure 4.13: Multi-view of the interpolation of a single car across 10 frames

operational domains targeted in our study. View-of-Delft (VoD) suffers from severely restricted annotation ranges, rendering it unsuitable for high-speed planning tasks. nuScenes provides excellent scale but physically lacks the elevation data required to thoroughly test 3+1D architectures. Conversely, while MAN TruckScenes perfectly satisfies the long-range hardware requirements and provides true 3+1D point clouds, it is heavily biased towards high-speed, dynamic highway environments.

To bridge this specific gap, the custom 3+1D dataset was engineered. Capturing data in the dense urban environment of Modena using our specific Continental ARS540CES sensor suite allowed us to construct a highly targeted perceptual challenge. As highlighted by the dataset's internal statistics, the resulting logs are overwhelmingly dominated by static targets—specifically parked vehicles lining multi-lane streets. These static objects generate weak, clutter-heavy radar reflections lacking prominent Doppler signatures, representing the polar opposite of the dynamic, highly reflective scenarios found in TruckScenes. Therefore, the primary motivation for creating this dataset was to explicitly measure and resolve domain adaptation.

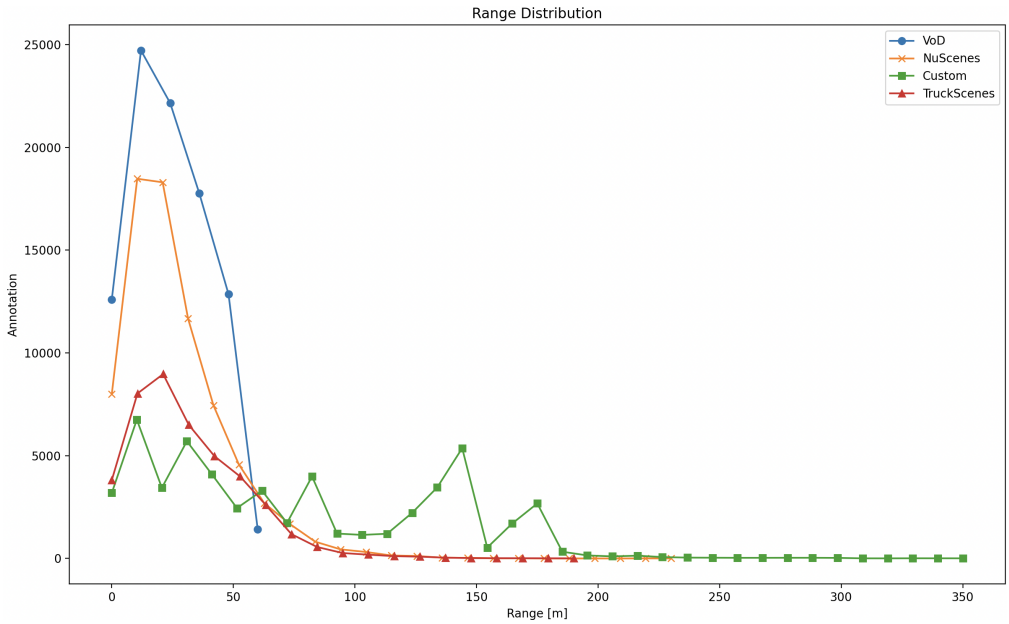


Figure 4.14: Distribution of range of car annotations across different datasets

The profound discrepancies in operational range across the datasets are empirically visualized in Figure 4.14. By restricting the NuScenes and TruckScenes annotations strictly to the frontal area to ensure a fair spatial comparison with the frontal-only VoD and custom setups, the fundamental

limitations of academic benchmarks for high-speed tasks become immediately apparent.

The distribution curves clearly validate the unsuitability of the VoD dataset for long-range planning. Despite containing a massive volume of annotations, these labels are entirely compressed within the immediate near-field, peaking intensely before abruptly plummeting to zero at approximately 60 meters. Training a network on this data enforces a strict perceptual boundary, preventing the model from learning geometric or semantic radar features beyond this artificial crop.

Conversely, NuScenes and TruckScenes exhibit a natural, uncropped decay. Their distributions taper smoothly, extending valuable training samples out to 100-150 meters. This correctly reflects the physical attenuation of the sensor and natural environmental occlusion rather than an arbitrary labeling limit. However, within this specific frontal slice, their absolute sample volumes at extreme ranges remain relatively low.

This comparative analysis directly justifies the creation of the custom dataset. As illustrated by its highly distinct distribution, the custom dataset not only captures near-field urban traffic but maintains a robust density of annotations well into the far-field. The prominent secondary peaks between 130 and 180 meters represent continuous lines of parked vehicles along straight urban corridors in Modena. Consequently, this custom dataset uniquely bridges a critical gap: it exposes the network to the heavy, static clutter typical of an urban environment, while simultaneously sustaining the 150-200 meter annotation ranges strictly required for highway-speed perception tasks.

5. Experimental Evaluation: Radar Optimisation, Comparison, and Hardware Acceleration

To transition from the theoretical design of the neural architectures to their empirical evaluation, this chapter details the physical environment, software frameworks, and quantitative metrics utilized during the testing phase. The subsequent sections present the baseline training results, hyperparameter ablation studies, and the final domain adaptation experiments on the custom radar datasets.

5.1 Hardware and Software Infrastructure

All experiments, including data preprocessing, model training, and inference, were conducted on a high-performance local workstation. The hardware configuration was selected to handle the massive input/output operations required by point cloud datasets and the heavy parallel computations required by 3D object detection networks.

The system is powered by an Intel® Core™ i9-14900KF processor operating at a base frequency of 5.3 GHz (with a maximum boost clock of 6.0 GHz) and is equipped with 128 GB of system memory (RAM). Graphics Processing Unit (GPU) acceleration, which is critical for deep learning tasks, was provided by an NVIDIA GeForce RTX 4090 GPU equipped with 24 GB of dedicated VRAM.

The workstation operates on the Ubuntu 24.04.1 LTS (x86_64) environment. To leverage the hardware acceleration, NVIDIA Driver version 580.173.02 and CUDA Toolkit version 13.0 were installed. The deep learning software stack relies heavily on PyTorch (via `nvidia-tao-pytorch` version 6.26.3) and includes advanced inference optimization libraries such as ONNX (1.21.0), ONNX Runtime GPU (1.26.0), and TensorRT (10.16.1.11) tailored for CUDA 13.

5.2 Implementation Frameworks

Evaluating fundamentally different neural network architectures—ranging from grid-based CNNs to Graph Neural Networks—often introduces severe software dependency conflicts. To ensure strict

isolation, stability, and reproducibility of the experiments, each main architecture was executed within its own dedicated Docker container.

- **OpenPCDet Base Models:** The standard PointPillars architecture and the CenterPoint (CenterHead) implementation were developed utilizing OpenPCDet, a widely adopted, open-source PyTorch-based codebase specifically engineered for 3D object detection from point clouds.
- **Cross-Modal Augmentation:** The LiDAR-Radar Feature Augmentation network was implemented using the official See_beyond_seeing repository published by its authors. Because this framework is a highly modified extension of OpenPCDet, it was deployed using the specific Docker environment provided in the repository to satisfy its custom CUDA operations.
- **RadarGNN:** The Graph Neural Network architecture was implemented using the official repository maintained by the TUM Institute of Automotive Technology (TUMFTM). It was executed within its standalone Docker container to isolate its specific graph-processing dependencies from the standard grid-based frameworks.

5.2.1 Base Training Hyperparameters

To establish a consistent baseline for evaluating the different architectures, a standardized training protocol was adopted across the OpenPCDet-based models prior to dataset-specific fine-tuning:

- **Point Cloud Range:** The spatial boundaries for the network's detection volume were defined from $[0, -25.6, -10.0]$ to $[153.6, 25.6, 10.0]$ meters. Compared to the default OpenPCDet configuration, this range was extended to accommodate the long-range detection requirements of the custom radar dataset along x, at the same time the z dimension was increased to keep into account the wider spread of radar points.
- **Temporal Accumulation:** To increase point density and overcome sensor sparsity, 10 consecutive frames were accumulated.
- **Ground Truth (GT) Filtering:** A strict custom filter was introduced requiring a minimum of 1 valid radar point inside the bounding box for a ground truth label to be retained.

- **Class Mapping:** Evaluations were restricted to three primary categories: *car* (which aggregates *moving_cars* and *stopped_cars* depending on the specific experiment), *pedestrian*, and *bicycle*. All other classes were entirely excluded from the evaluation.

The network weights were optimized using the Adam optimizer, paired with a One-Cycle learning rate scheduler. This scheduling policy is highly effective for 3D object detection tasks, as it dynamically modulates the learning rate to prevent premature convergence and stabilize the training process over sparse point cloud representations [39]. The baseline training runs were consistently executed for a total of 30 epochs.

The batch size, however, was not maintained as a static constant. Because the number of samples per batch directly influences the stability of the gradient updates and the resulting Average Precision (AP) of the detector, it was treated as a tunable hyperparameter. The specific batch sizes evaluated, alongside their corresponding impact on the network's performance, are thoroughly detailed in the experimental ablation studies in the following sections.

5.2.2 Dataset Split

For the public datasets, the original training and validation splits were maintained. For the custom radar dataset, a 80/20 split was applied to create a training set and a validation set, respectively. This split was applied independently for the two main sequence, the first one containing 529 frames and the second one containing the remaining 2900 frames. The split was performed by splitting the 2 sequences into 2 subsequences each to ensure that the training and validation sets were temporally disjoint, preventing any potential data leakage.

5.3 Evaluation Metrics for 3D Object Detection

To quantitatively assess the performance of the proposed 3D object detection architectures, standard evaluation metrics established by the community are employed. The evaluation of an object detector fundamentally relies on its ability to correctly classify an object and accurately regress its physical boundaries or location. To quantify this, we first define the concepts of True Positives (TP), False Positives (FP), and False Negatives (FN), which allow for the calculation of Precision and Recall.

Precision measures the accuracy of the positive predictions, representing the fraction of detections that are actually correct:

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

Recall measures the completeness of the model, representing the fraction of actual ground-truth objects that were successfully detected:

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

Depending on the benchmark being used, the criterion that determines whether a prediction is classified as a True Positive varies significantly. In this work, metrics from both the KITTI and nuScenes benchmarks are utilized to provide a comprehensive evaluation.

5.3.1 KITTI Evaluation Benchmark

The KITTI dataset establishes a strict geometric evaluation protocol based on the Intersection over Union (IoU). For a prediction to be considered a True Positive, its overlap with a ground-truth bounding box must exceed a predefined IoU threshold, specifically 0.7 and 0.5 for cars, and 0.5 and 0.25 for pedestrians and cyclists [40].

Traditionally, KITTI evaluated the Average Precision (AP) using an 11-point interpolation method inspired by the PASCAL VOC challenge. This method calculates the average of the maximum precision values at 11 equally spaced recall levels:

$$AP|_{11} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} \max_{\tilde{r} \geq r} P(\tilde{r}) \tag{5.3}$$

where $P(\tilde{r})$ is the measured precision at recall $\tilde{r}$ [40],[41].

However, to overcome the statistical bias and the coarse discretization introduced by the 11-point interpolation, the KITTI benchmark was updated to use a 40-point interpolation metric ($AP|_{R40}$). This updated metric provides a fairer and more accurate representation of the area under the Precision-Recall curve by sampling the precision at 40 recall positions, omitting the 0 recall point [41]:

$$AP|_{R40} = \frac{1}{40} \sum_{i=1}^{40} \max_{\tilde{r} \geq r_i} P(\tilde{r}) \tag{5.4}$$

with $r_i = \frac{i}{40}$. In this study, the $AP|_{R40}$ is reported as the primary metric when evaluating against KITTI standards.

5.3.2 nuScenes Evaluation Benchmark

Unlike KITTI, the nuScenes benchmark abandons the IoU-based matching criterion. In autonomous driving, accurately predicting the exact 3D shape and overlap of an object is highly challenging, particularly for sparse sensors like radar. Instead, nuScenes defines a True Positive based on the 2D center distance on the ground plane between the predicted box and the ground-truth box [22].

The nuScenes Mean Average Precision (mAP) is calculated by averaging the AP over a set of distinct center distance thresholds $\mathbb{D} = \{0.5, 1, 2, 4\}$ meters, rather than IoU thresholds [41],[22].

Furthermore, to provide a granular analysis of the model's performance on True Positives, the nuScenes benchmark defines a set of True Positive (TP) metrics. These metrics quantify specific error types for matched detections (where the distance between the prediction and the ground truth is strictly less than 2 meters):

- **Average Translation Error (ATE):** The Euclidean center distance in 2D (in meters) [22].

$$ATE = \sqrt{(x_{pred} - x_{gt})^2 + (y_{pred} - y_{gt})^2} \tag{5.5}$$

- **Average Scale Error (ASE):** Calculated as $1 - IoU$ after aligning the orientation and translation of the predicted and ground-truth boxes [22].

$$ASE = 1 - IoU_{aligned} \tag{5.6}$$

- **Average Orientation Error (AOE):** The smallest yaw angle difference between the prediction and ground truth (in radians) [22].

- **Average Velocity Error (AVE):** The absolute difference in the 2D velocity vectors (in m/s).

$$AVE = \sqrt{(v_{x,pred} - v_{x,gt})^2 + (v_{y,pred} - v_{y,gt})^2} \tag{5.7}$$

- **Average Attribute Error (AAE):** Defined as 1 minus the attribute classification accuracy [22].

Finally, to rank models comprehensively, nuScenes aggregates the mAP and the mean values of the True Positive errors (mATE, mASE, mAOE, mAVE, mAAE) into a single scalar value known

as the nuScenes Detection Score (NDS). The NDS represents a weighted sum of the mAP and the complement of the True Positive errors [41],[22]:

$$NDS = \frac{1}{10}\left[5 \cdot mAP + \sum_{mTP \in \mathbb{TP}} (1 - \min(1, mTP))\right] \tag{5.8}$$

where $\mathbb{TP}$ is the set of the five mean True Positive error metrics. This formula ensures that excessive errors are saturated at zero contribution, providing a holistic measure of the detector's spatial, semantic, and kinematic accuracy.

For this these experiments, the nuScenes evaluation metrics, especially the mAP, is used as the primary benchmark for assessing the performance of the proposed 3D object detection architectures. This choice is motivated by the fact that nuScenes' center-distance-based evaluation is more suitable for sparse radar data, where precise bounding box overlap is less critical than accurate localization and classification.

5.3.3 Inference Efficiency Metrics

Beyond detection accuracy, deploying 3D object detection models on autonomous vehicles requires strict adherence to computational constraints. To comprehensively evaluate the impact of post-training acceleration—specifically the graph optimizations and hardware-specific kernel selections performed by ONNX and TensorRT—we explicitly measure both inference latency and memory consumption.

Inference Latency and Warmup Iterations

To obtain an accurate representation of the model's operational speed, latency is measured exclusively during steady-state inference. In modern deep learning frameworks and optimized execution engines like TensorRT, the initial forward passes inevitably incur significant overhead due to lazy memory allocation, caching, and computational graph initializations. Consequently, a predetermined number of "warmup" iterations are executed and discarded prior to recording. The reported inference time is the average latency of the subsequent iterations, ensuring the measurements reflect the true deployment throughput.

Memory Consumption

Although all experimental training and inference profiling were conducted on a high-performance desktop workstation equipped with an RTX 4090 GPU, evaluating the dynamic memory footprint remains a critical metric. The ultimate objective of this research is to identify highly efficient architectures suitable for future deployment on embedded automotive edge devices. Consequently, tracking peak memory allocation ensures that the selected models will not eventually exceed the strict, limited VRAM constraints inherent to those target embedded vehicle systems. Rather than relying on static parameter counts, the memory usage is evaluated dynamically during active inference.

To precisely capture the peak allocation required by the model during a forward pass, consumption was profiled using the `memory_usage` function from the Python `memory_profiler` library. For a set of n valid inference iterations, the maximum memory allocated during each individual iteration i, denoted as $\max(\text{mem}_i)$, is recorded. The final reported metric is the average of these peak consumptions, formulated as:

$$\text{Average Peak Memory} = \frac{1}{n} \sum_{i=1}^{n} \max(\text{mem}_i)$$

This metric guarantees that transient memory spikes—which could otherwise cause out-of-memory failures on edge devices—are accurately represented and averaged over the entire evaluation sequence.

5.4 Preprocessing: Cross-Potential Point Cloud Evaluation

Before establishing the final training pipeline, an initial preprocessing phase was explored to address the inherent noise and multi-path reflections (ghost points) typical of radar sensors. To achieve this, we evaluated the "Cross-Potential" methodology introduced by the Pointillism framework [42].

5.4.1 Theoretical Background: DBSCAN and Cross-Potential

The foundational principle of the Cross-Potential algorithm relies on the spatial diversity provided by multiple, physically separated radar sensors. Because multi-path ghost points are highly sensitive

to the angle of incidence, a ghost point perceived by one radar is highly unlikely to appear at the same spatial coordinates in an adjacent radar [42]. To systematically exploit this, the framework processes the point clouds through a rigorous two-step spatial validation pipeline:

1. Independent Clustering via DBSCAN

First, to isolate cohesive physical objects from random background clutter, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is applied independently to the point cloud of each radar. DBSCAN groups points that are closely packed together based on a defined distance threshold (ϵ) and a minimum density of points ($MinPts$). This step yields a set of clusters for each individual radar [42].

2. Cross-Potential Formulation

Once the independent clusters are formed, the framework establishes correspondences between them across multiple radars by calculating a cross-potential score. This score acts as a confidence metric indicating whether two clusters from different radars belong to the same physical object, and it is inversely proportional to the distance between their centroids [42].

Mathematically, in an N-radar system, the cross-potential $P(c_i^j|\Gamma_k)$ for the i-th cluster in radar j (where c_i^j represents its centroid) with respect to the k-th radar's point cloud Γ_k ($k \neq j$) is defined as:

$$P(c_i^j|\Gamma_k) = \frac{1}{1 + \left[\frac{r_i^{jk}}{2}\right]^2}$$

where r_i^{jk} is the distance between the centroid c_i^j and the respective nearest cluster centroid found in the k-th radar's point cloud Γ_k [42].

The specific formulation of this potential function is motivated by standard vehicle dimensions. Because a typical car has a width of approximately 2 meters, the function is designed to assign a high potential score ($P > 0.5$) whenever the centroid distance $r_i^{jk} < 2$m [42]. During this process, every individual radar point inherits the exact same cross-potential score as its respective cluster centroid. Finally, all points falling below a defined potential threshold are filtered out to create the purified Cross Potential Point Clouds.

5.4.2 Experimental Implementation on the Custom Setup

The original Pointillism architecture was designed and validated using two strictly parallel radars[42]. However, the custom 2D dataset utilized in this study featured a three-radar configuration: one central forward-facing sensor, and two lateral sensors (left and right) rotated outwards at angles between 17° and 19° to maximize the peripheral Field of View (FoV).

To adapt the Cross-Potential algorithm to this specific geometry, the potential fields were computed in pairs. For any given frame, we calculated:

- P_{LC}: The cross-potential between the Left and Central radars (visualized in Figure 5.1a).
- P_{CR}: The cross-potential between the Central and Right radars (visualized in Figure 5.1b).

To consolidate these distinct fields into a unified metric, the final potential for each point was defined as the maximum value between its two respective pairings:

$$P_{final} = \max(P_{LC}, P_{CR})$$

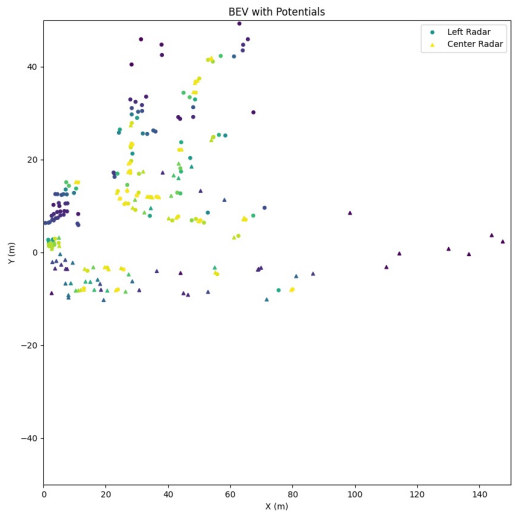
A hard acceptance threshold was subsequently applied, discarding any radar point where $P_{final} < 0.5$.

5.4.3 Results and Rejection of the Methodology

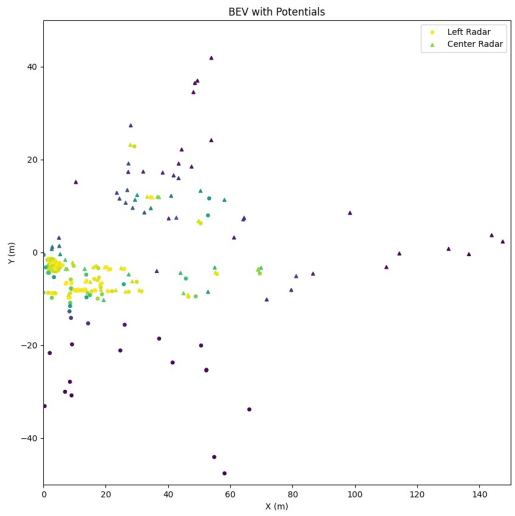
Figure 5.2 illustrates the Bird's-Eye View (BEV) of the multi-radar point cloud before and after the application of the 0.5 Cross-Potential threshold.

The initial results appeared highly promising in the near-field; the algorithm successfully identified and suppressed a significant volume of scattered ghost points directly around the ego-vehicle where the sensors' FoVs heavily overlapped. However, a critical failure mode emerged as the distance from the ego-vehicle increased. Because the outer radars were rotated by up to 19°, their respective FoVs naturally diverged from the central radar's FoV at longer ranges.

Without spatial overlap, valid target points detected by a single sensor had no corresponding points in the adjacent sensors to generate a high cross-potential score. Consequently, the filtering algorithm became overly aggressive, systematically erasing legitimate targets at the following approximate boundaries:

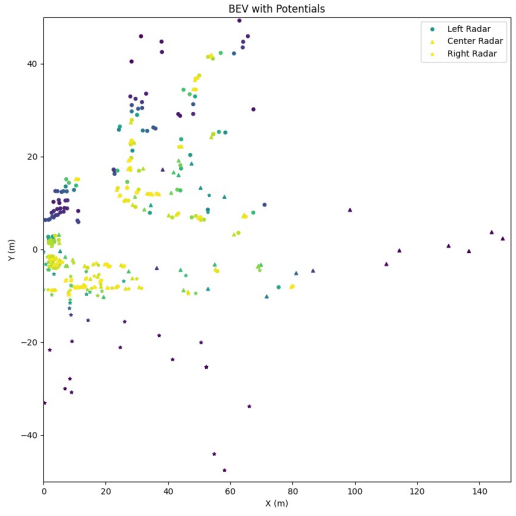


(a) Cross-Potential evaluation between the Left and Central radars

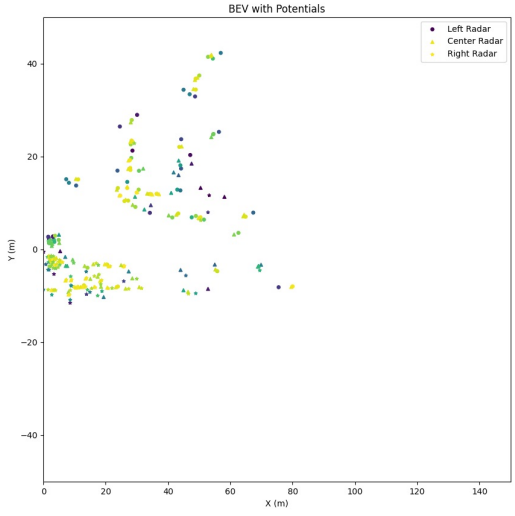


(b) Cross-Potential evaluation between the Central and Right radars

Figure 5.1: Cross-Potential between the pairs of radars



(a) Original multi-radar point cloud (unfiltered)



(b) Point cloud after Cross-Potential filtering

Figure 5.2: Comparison of the global point cloud before and after Cross-Potential thresholding

- **Frontal region (Center):** Valid points were filtered out beyond 70-80 meters.
- **Left region (lx):** Valid points were filtered out beyond 40 meters.
- **Right region (rx):** Valid points were filtered out beyond 10 meters.

Given that achieving accurate, long-range object detection is a primary objective of this thesis, heavily truncating the effective range of the sensor suite is unacceptable. Therefore, despite its efficacy in near-field noise reduction, the Cross-Potential preprocessing step was officially discarded, and all subsequent neural network training was conducted utilizing the raw, unfiltered radar point clouds to preserve maximum range fidelity.

5.5 Ablation Study for Tuning

This section details the empirical hyperparameter optimization required to handle the specific sparsity of the radar data, focusing heavily on batch processing and spatial discretization limits. For these experiments, TruckScenes was selected as the primary benchmark dataset as we have established that it is the publicly available dataset that meets the better the requirements for long-range autonomous driving functions.

5.5.1 Batch Size

To determine the optimal training configuration, an ablation study was conducted focusing on the batch size parameter. The experiments were executed utilizing the PillarNet architecture to evaluate how the number of samples per forward pass affects the network's learning stability and ultimate precision. The performance metrics, including recall at various Intersection over Union (IoU) thresholds and class-specific Mean Average Precision (mAP), were systematically evaluated for batch sizes of 4, 8, and 16.

As detailed in Table 5.1, varying the batch size yields a noticeable impact on both global recall and class-specific accuracy. While a batch size of 4 achieves the highest recall at the strictest 0.7 threshold (0.24), a batch size of 8 demonstrates the strongest overall performance for the primary vehicle classes. Specifically, it peaks at a 0.7028 mAP for moving cars and a 0.1952 mAP for stopped cars.

Metric	Batch Size 4	Batch Size 8	Batch Size 16
Recall @ 0.3	0.66	0.67	0.65
Recall @ 0.5	0.52	0.51	0.47
Recall @ 0.7	0.24	0.21	0.18
Car Moving (mAP)	0.67	0.70	0.67
Car Stopped (mAP)	0.19	0.20	0.16
Bicycle (mAP)	0.13	0.14	0.15
Pedestrian (mAP)	0.06	0.07	0.09

Table 5.1: Comparison of PillarNet performance metrics across different batch sizes with nuscenes evaluation

Conversely, increasing the batch size to 16 degrades the recall metrics across all evaluated thresholds, dropping to 0.18 at the 0.7 threshold. Although this larger batch size shows minor mAP improvements for minority classes such as pedestrians (0.09) and bicycles (0.15), it compromises the detection accuracy of the predominant car classes. Based on these empirical results, a batch size of 8 offers the most effective balance between memory utilization, recall stability, and target accuracy; consequently, it was selected and maintained for all subsequent training iterations.

5.5.2 Maximum Voxel Limits

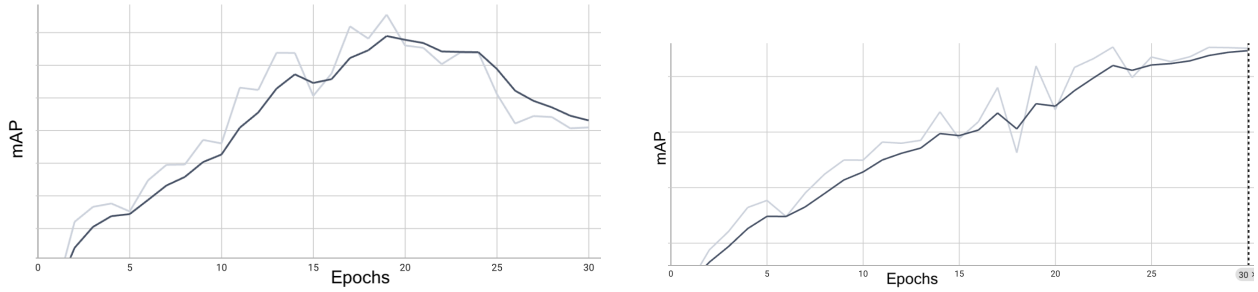
To mitigate the sparse nature of radar point clouds while maintaining computational efficiency, an ablation study was conducted on the maximum number of allowed voxels. While the default configuration for standard point-based pillar networks is set to 30,000 voxels, this limit causes significant resource overhead when processing sparse data. Consequently, evaluations were performed by scaling the maximum voxel limits down to 5,000, 3,000, and 800. Tables 5.2 and 5.3 illustrate the resulting impact on performance for both the anchor-based PointPillars architecture and the CenterHead-based PillarNet framework.

As shown in Table 5.2, utilizing the default 30k voxel limit for PointPillars results in a car mAP of 0.54. Reducing the limit down to 5,000 voxels unexpectedly improves the model's performance,

Metric	**30k**	**5k**	**3k**	**800**
		Max Voxels		
Recall @ 0.3	0.68	0.70	0.68	0.65
Recall @ 0.5	0.52	0.54	0.52	0.45
Recall @ 0.7	0.21	0.23	0.23	0.15
Car (mAP)	0.54	0.58	0.57	0.50
Bicycle (mAP)	0.06	0.04	0.02	0.09
Pedestrian (mAP)	0.01	0.02	0.01	0.01

Table 5.2: Performance evaluation of PointPillars (Anchor-based) across varying maximum voxel limits with nuscenes evaluation

raising the car mAP to 0.58 and increasing the recall at the 0.5 threshold to 0.54. This occurs because radar point clouds are inherently sparse; capping the maximum voxel count acts as an implicit regularization step that filters out noisy, low-density background clutter. However, compressing the limit too severely down to 800 voxels strips away vital geometric details, degrading the car mAP to 0.50.



(a) mAP Training Curve with 30k Max Voxels

(b) mAP Training Curve with 5k Max Voxels

Figure 5.3: Training progression of the Mean Average Precision (mAP) over 30 epochs for the PointPillars architecture. The lighter line displays the raw mAP measured epoch by epoch, while the darker line represents a smoothed version to clearly highlight the underlying trend

The impact of restricting the maximum number of voxels extends beyond peak accuracy; it significantly influences the stability of the training process itself. As illustrated in Figure 5.3,

comparing the training curves of PointPillars configured with the default 30,000 voxels (Figure 5.3a) versus the optimized 5,000 voxels (Figure 5.3b) reveals a distinct divergence in convergence behavior.

When utilizing the 30k voxel limit, the network's performance peaks around epoch 19 but subsequently experiences a continuous degradation toward the end of the 30-epoch training cycle. This late-stage drop indicates that allowing the network to process an excessive amount of sparse, empty space introduces learning instability, likely causing the model to overfit on background noise and clutter.

Conversely, reducing the limit to 5k voxels effectively acts as a spatial regularizer. By discarding redundant spatial information and forcing the network to focus strictly on the densest features, the 5k configuration maintains a steady, upward trajectory. The training curve continuously improves and stabilizes near the end of the cycle, entirely avoiding the performance collapse observed in the 30k setup.

The recall performance also follows the same pattern as the one shown for the mAP for each configuration studied here. Moreover, the training with 3k and 800 voxels, while not shown in the figure, exhibited similar behaviour to the 5k configuration, proving how the reduction of the maximum voxel limit can enhance training stability in sparse radar point cloud scenarios, even if they reached a lower peak as previously observed.

Metric	**Max Voxels: 30k**	**Max Voxels: 5k**
Recall @ 0.3	0.67	0.68
Recall @ 0.5	0.50	0.51
Recall @ 0.7	0.21	0.21
Car (mAP)	0.60	0.61
Bicycle (mAP)	0.11	0.10
Pedestrian (mAP)	0.05	0.08

Table 5.3: Performance evaluation of PillarNet (CenterHead-based) across varying maximum voxel limits with nuscenes evaluation

Conversely, the PillarNet framework evaluated with the CenterHead module (Table 5.3) exhibits

almost unnoticeable improvements when reducing the voxel limit from 30k to 5k, with the car mAP marginally increasing from 0.60 to 0.61. This behavior directly aligns with theoretical expectations regarding its architecture. Because PillarNet employs a sparse 2D CNN encoder, it inherently bypasses empty space computations. Consequently, as long as the maximum voxel limit remains greater than the total number of non-empty voxels present in the sparse radar point cloud, altering this artificial ceiling has a negligible impact on the network's feature representation. This stands in stark contrast to the pronounced improvements observed in PointPillars, further validating the efficiency of sparse convolutions when processing automotive radar data.

Point per Pillar Limit

A similar study has been conducted to evaluate the impact of the maximum number of points per pillar. The default configuration for PointPillars is set to 20, while for PillarNet it is set to 10. While the sparse input of the radar does not generally saturate this maximum threshold, tests were performed to determine if lowering this limit would influence the network's learning behavior. The tests were conducted down to 3 points per pillar, and the results were consistent across both architectures.

Ultimately, the empirical differences in detection accuracy proved to be almost unnoticeable across both architectures. This lack of impact is directly attributable to the physical characteristics of automotive radar point clouds. Because radar reflections rarely cluster densely enough within a single discretized pillar to exceed even lowered thresholds, the remaining point capacity is inherently filled with zero-padding. The results indicate that the respective point-based encoders are highly robust to this zero-padding; they remain completely unaffected by the empty points, efficiently ignoring them during the feature aggregation and extraction phases without incurring any degradation in precision.

5.5.3 Voxel Size

The spatial discretization of the point cloud, dictated by the voxel dimensions, is a critical hyperparameter for grid-based architectures.

PointPillar

By default, PointPillars is configured with a high-resolution voxel size of $[0.16, 0.16, z]$ meters, which is optimized for dense LiDAR scans. To evaluate its effectiveness on sparse radar data, this default configuration was benchmarked against the optimized $[0.32, 0.32, z]$ meters resolution (previously established during the batch size experiments).

Metric	$[0.16, 0.16, z]$	$[0.32, 0.32, z]$
	Voxel Size	
Recall @ 0.3	0.61	0.70
Recall @ 0.5	0.50	0.54
Recall @ 0.7	0.25	0.23
Car (mAP)	0.50	0.58
Bicycle (mAP)	0.08	0.04
Pedestrian (mAP)	0.09	0.02
Average PyTorch Inference Time (ms)	44.1	34.0

Table 5.4: Performance comparison between the default high-resolution voxel size and the optimized larger voxel size on PointPillars with nuscenes evaluation

As detailed in Table 5.4, transitioning to the larger 0.32m grid yields a noticeable increment in both the general recall (at the 0.3 and 0.5 thresholds) and the primary Car mAP, which improves significantly from 0.50 to 0.58. This behavior is directly attributable to the extreme inherent sparsity of automotive radar data. A wider spatial grid effectively allows the network to aggregate sparse, scattered reflections over a larger physical area, generating a more cohesive and robust feature representation for massive objects like vehicles.

Conversely, this coarser resolution severely degrades the detection capabilities for Vulnerable Road Users (VRUs). For the 0.32m configuration, the mAP drops by half for bicycles (0.04) and plummets for pedestrians (0.02). This performance penalty is highly predictable given the small physical dimensions of their bounding boxes. When a 0.32m grid is applied to an already sparse pedestrian signature, the target is compressed into a remarkably small amount of voxels. This

excessive quantization collapses the fine-grained spatial resolution required to correctly classify and localize compact targets, proving that the finer 0.16m voxel size is vastly superior for maintaining the geometry of pedestrians and bicycles.

Moreover, the larger voxel size also yields a significant reduction in inference latency, decreasing the average forward pass time from 44.1 ms to 34.0 ms. This improvement is primarily due to the reduced number of voxels that need to be processed by the network, which directly translates into fewer computations and faster execution times.

PillarNet

In contrast to the behavior observed in the PointPillars architecture, evaluating the voxel size on PillarNet yields fundamentally opposite results. As detailed in Table 5.5, decreasing the spatial dimensions of the voxels systematically improves the detection precision across all object scales.

Metric	$[0.08, 0.08, z]$	$[0.16, 0.16, z]$	$[0.32, 0.32, z]$
		Voxel Size	
Recall @ 0.3	0.69	0.68	0.63
Recall @ 0.5	0.55	0.51	0.47
Recall @ 0.7	0.28	0.21	0.18
Car (mAP)	0.63	0.61	0.55
Bicycle (mAP)	0.09	0.10	0.00
Pedestrian (mAP)	0.11	0.08	0.01
Average PyTorch Inference Time (ms)	76.9	78.3	75.4

Table 5.5: Performance comparison among different voxel size on PillarNet with nuscenes evaluation

When scaling down the voxel grid to a fine-grained $[0.08, 0.08, z]$ resolution, the PillarNet model reaches a peak Car mAP of 0.63, a significant increase from the 0.55 achieved with the coarse 0.32m grid. Similarly, the performance on Vulnerable Road Users (VRUs) improves drastically, with pedestrian detection reaching a 0.11 mAP compared to a near-zero outcome at the 0.32m

resolution.

This diverging trend can be explained by PillarNet's reliance on a sparse 2D convolutional encoder. Unlike PointPillars, which collapses into a dense pseudo-image and struggles to aggregate overly scattered points without losing context, PillarNet leverages specialized sparse convolutions that naturally preserve high-resolution geometric structures. This allows the network to benefit from the fine spatial granularity of smaller voxels without being overwhelmed by empty background space.

Interestingly, this precision enhancement does not compromise steady-state inference capabilities. As shown in the table, decreasing the voxel size does not meaningfully slow down the inference time, which fluctuates only marginally between 75.4 ms and 78.3 ms. Furthermore, the peak memory consumption during inference remains entirely stable at 1,960 MB across all three configurations. However, the finer spatial resolution does exponentially increase the number of active non-empty spatial locations processed during backpropagation, leading to a noticeable slowdown in the training process. Due to these prolonged training times, the 0.16m configuration was maintained for the subsequent experiments in this study. Nonetheless, further lowering the voxel size to 0.08m is a highly logical and effective choice for maximizing network precision, provided there are no strict time constraints imposed on the training phase.

5.5.4 Frame Accumulation

Mitigating the inherent sparsity of automotive radar is critical for achieving robust 3D object detection. As already established, a standard technique to increase point density is the temporal accumulation of consecutive radar sweeps, utilizing the vehicle's odometry to align historical points into the current coordinate frame. The impact of this frame aggregation was evaluated on the PointPillars architecture across two distinct environments: the dense urban View-of-Delft (VoD) dataset (Table 5.6) and the MAN TruckScenes dataset (Table 5.7).

The empirical data across both datasets demonstrates that accumulating frames systematically improves detection accuracy. On the VoD dataset, shifting from a single sweep to a 5-frame accumulation provides a uniform boost across all metrics. Global recall thresholds improve noticeably, and the primary Car 3D AP jumps from 0.31 to 0.36. More importantly, this densification heavily benefits Vulnerable Road Users (VRUs). Because pedestrians and bicycles generate extremely few reflections in a single sweep, accumulating 5 frames injects critical geometric context, raising the

Frame	0.3	0.5	0.7	Car 3D	Pedestrian 3D	Bike 3D
		Recall		AP_R40@0.5	AP_R40@0.25	AP_R40@0.25
1-frame	0.35	0.25	0.08	0.31	0.19	0.53
5-frame	0.40	0.29	0.11	0.36	0.25	0.58

Table 5.6: Comparison of the performance metrics for the PointPillars architecture when using single-frame versus five-frame accumulated radar point clouds on VoD

Pedestrian 3D AP from 0.19 to 0.25 and the Bike 3D AP from 0.53 to 0.58.

Frame	0.3	0.5	0.7	Car	Bicycle	Pedestrian
		Recall			mAP	
1-frame	0.70	0.55	0.25	0.52	0.00	0.00
5-frame	0.70	0.55	0.25	0.56	0.05	0.02
10-frame	0.70	0.54	0.23	0.58	0.04	0.02

Table 5.7: Comparison of the performance metrics for the PointPillars architecture when using single-frame versus five-frame accumulated radar point clouds on the TruckScenes dataset with NuScenes evaluation

This behavior is strongly corroborated by the NuScenes-based evaluation on the TruckScenes dataset. Using a single frame, the PointPillars architecture completely fails to detect bicycles and pedestrians (0.00 mAP). Injecting 5 frames rescues these minority classes, allowing the network to establish baseline detections (0.05 for bicycles, 0.02 for pedestrians), while simultaneously pushing the Car mAP from 0.52 to 0.56.

Further increasing the accumulation to 10 frames on TruckScenes pushes the Car mAP to a peak of 0.58. However, this extended memory introduces a slight trade-off in strict geometric localization, as evidenced by the drop in Recall @ 0.7 (from 0.25 to 0.23). While historical points enrich the spatial structure of static or slow-moving targets, the inherent latency of older frames can cause slight temporal "smearing" or trailing artifacts on high-speed dynamic vehicles, slightly

degrading the precision of the tightest bounding box overlaps.

5.5.5 Doppler-driven Point Compensation

A study has been conducted to evaluate the impact of compensating the spatial positions of radar points using their radial velocity as presented in 2.2.2. The goal is to determine whether this kinematic adjustment enhances the network's ability to accurately detect and localize moving objects.

PointPillars					
Model	**0.3**	**0.5**	**0.7**	**Car**	**AOE**
		Recall		mAP	(Orientation)
Not Compensated	0.70	0.54	0.23	0.58	0.09
Compensated	0.69	0.54	0.24	0.60	0.09
PillarNet					
Model	**0.3**	**0.5**	**0.7**	**Car_moving**	**AOE**
		Recall		mAP	(Orientation)
Not Compensated	0.67	0.52	0.24	0.67	0.04
Compensated	0.67	0.50	0.20	0.68	0.03

Table 5.8: Comparison of the performance metrics for PointPillars and PillarNet with and without compensating the point positions with radial velocity with NuScenes evaluation

As detailed in Table 5.8, actively compensating the spatial positions of the radar points using their radial velocity has a limited impact on the network's detection capabilities.

For the PointPillars architecture, applying this kinematic compensation proves beneficial for global accuracy. It successfully increases the overall Car mAP by two percentage points, raising it from 0.58 to 0.60. Crucially, this improvement in precision is achieved without sacrificing bounding box stability; the recall values remain effectively unchanged, and the Average Orientation Error (AOE) holds steady at 0.09.

When evaluating the PillarNet architecture, the compensation strategy similarly provides a minor

enhancement to the moving car precision, pushing the mAP from 0.67 to 0.68. It is important to note that the experiments on PillarNet have been conducted on moving cars only which are the only class (among the evaluated classes) that is affected by the compensation, and so the improvements expected were supposed to be higher, while in fact they are minimal. However, unlike PointPillars, PillarNet experiences a noticeable trade-off in its localization strictness. The recall at the stringent 0.7 threshold drops from 0.24 to 0.20, indicating that while the network is slightly more confident in its class predictions, it struggles more to perfectly align the predicted box with the ground truth geometries when the points are shifted.

Furthermore, while the theoretical AOE metric on the TruckScenes dataset appears relatively stable across these tests, empirical evaluations on custom data sets revealed a critical behavioral shift. Specifically, compensating the points with radial velocity tends to introduce a slight degradative effect on the predicted orientation of moving vehicles in real-world scenarios. Therefore, while integrating radial velocity compensation offers a consistent 1 to 2 percentage point boost in global mAP, its implementation requires careful consideration regarding its subtle impacts on high-threshold spatial recall and orientation stability.

5.6 Comparison: PointPillar vs CenterHead

5.6.1 Evaluation on MAN TruckScenes

mAP Comparison

The core of this evaluation compares the baseline anchor-based PointPillars model against the sparse, CenterHead-based PillarNet architecture. Both models were trained and evaluated on the high-speed, dynamic environments provided by the MAN TruckScenes dataset. The quantitative results of this comparison are detailed in Table 5.9.

As illustrated by the metrics, PillarNet clearly outperforms PointPillars in overall detection precision. While PointPillars maintains a marginal advantage in the loosest recall threshold (0.70 at Recall @ 0.3), PillarNet demonstrates greater spatial accuracy, achieving a higher recall at the strict 0.7 threshold (0.28 compared to 0.23). Furthermore, the CenterHead architecture achieves a noticeably higher global accuracy, pushing the primary Car mAP to 0.63 compared to PointPillars'

Model	0.3	0.5	0.7	Car	Bicycle	Pedestrian
	Recall			mAP		
PointPillars	0.70	0.54	0.23	0.58	0.04	0.02
PillarNet	0.69	0.55	0.28	0.63	0.09	0.11

Table 5.9: Performance comparison between the anchor-based PointPillars and the CenterHead-based PillarNet architectures on the MAN TruckScenes dataset

0.58.

PillarNet also exhibits superior capability in detecting Vulnerable Road Users (VRUs), scoring higher mAPs for both bicycles (0.09) and pedestrians (0.11). However, despite these improvements, it must be noted that radar point clouds generally struggle to reliably detect pedestrians and cyclists due to their small cross-sections, proving vastly more effective for large vehicles.

TP Errors Comparison

To further understand the geometric precision of the predictions, the True Positive (TP) errors for the primary Car class were analyzed. The evaluation focused on the Average Translation Error (ATE), Average Scale Error (ASE), and Average Orientation Error (AOE), as detailed in Table 5.10. It has been chosen to focus on the Car class because it is the most prevalent and critical target in the dataset, and it provides a more statistically significant basis for comparison. While for the other classes, the classification errors are more critical than the geometric errors, for the Car class, the geometric errors are more pronounced and provide a clearer insight into the model's spatial accuracy.

This error analysis perfectly corroborates the previously discussed mAP findings. PillarNet significantly outperforms PointPillars in spatial localization, reducing the ATE from 0.36 meters to a highly precise 0.26 meters. This substantial improvement in calculating the 2D center distance is the primary driver behind the CenterHead's superior overall accuracy. Meanwhile, both architectures exhibit identical performance regarding bounding box scaling, maintaining an ASE of 0.18.

Conversely, the anchor-based PointPillars model retains a distinct advantage in predicting the vehicle's heading, achieving a lower AOE of 0.09 compared to PillarNet's 0.11. Because predefined

Model	ATE (Translation)	ASE (Scale)	AOE (Orientation)
		Car Class	
PointPillars	0.36	0.18	0.09
PillarNet	0.26	0.18	0.11

Table 5.10: Comparison of the TP errors of class Car between the anchor-based PointPillars and the CenterHead-based PillarNet architectures on the MAN TruckScenes dataset

anchors are explicitly aligned with specific yaw angles during grid initialization, the network regresses orientation more stably than the anchor-free CenterHead approach.

Nevertheless, despite this slight degradation in orientation accuracy, PillarNet remains the definitive winner of this comparison. Its massive advantage in translation precision and higher global recall vastly outweigh the minor orientation penalty, establishing it as the superior architecture for processing these sparse radar environments.

5.6.2 Evaluation on the Proprietary Dataset

The final phase of the experimental evaluation measures the domain adaptation capabilities of the networks. Models initially trained on the TruckScenes dataset were evaluated on the custom 3+1D proprietary dataset, both before and after applying a targeted fine-tuning process. The primary evaluation utilizes the nuScenes distance-based metrics (Table 5.11), while the KITTI R40 IoU-based metrics (Table 5.12) serve as a secondary, stricter geometric baseline.

Evaluating the pre-trained TruckScenes models directly on the custom dataset demonstrates a severe domain shift. Both PointPillars and PillarNet exhibit near-zero zero-shot generalization, with recall at the 0.7 threshold plummeting to essentially zero. This highlights the fundamental differences between the dynamic highway environments of TruckScenes and the heavy static clutter of the custom urban logs.

However, fine-tuning the models on the custom dataset yields drastic improvements. PillarNet maintains its structural superiority in the primary nuScenes evaluation, achieving a higher global recall (0.3384 at 0.3) and outperforming PointPillars in the Car mAP (0.1908 vs. 0.1376). Interest-

Model	0.3	0.5	0.7	Car	Pedestrian
		Recall		mAP	
PointPillars	0.0653	0.0077	0.0005	0.0582	0.00
PillarNet	0.0920	0.0124	0.0008	0.1388	0.0010
PointPillars (Fine-tuned)	0.2461	0.1179	0.0189	0.1376	0.3363
PillarNet (Fine-tuned)	0.3384	0.1403	0.0267	0.1908	0.3430

Table 5.11: Performance comparison between the anchor-based PointPillars and the CenterHead-based PillarNet architectures on the 3+1D proprietary dataset with nuscenes evaluation

ingly, during this fine-tuning phase, the learning dynamics diverged heavily between classes. The network reached its peak mAP for cars after just a few epochs, whereas pedestrian accuracy required approximately 20 epochs to stabilize. This discrepancy is a direct consequence of the dataset's extreme class imbalance; passenger vehicles overwhelmingly dominate the dataset, accounting for tens of thousands of annotations, whereas pedestrians are significantly underrepresented. The abundance of static, parked cars allowed the network to quickly adapt its vehicle features, while the sparse pedestrian samples required prolonged exposure to converge.

Remarkably, the fine-tuned models achieved unexpectedly high accuracy for pedestrians under the nuScenes metric (0.3363 for PointPillars and 0.3430 for PillarNet). This is a massive anomaly compared to their poor pedestrian detection performance previously observed on the TruckScenes dataset, suggesting that the specific urban radar signatures in the proprietary dataset—combined with the fine-tuning process—provided richer geometric or contextual cues for this class. However, this value is not statistically meaningful: the proprietary dataset contains only four unique pedestrian instances, so the per-class pedestrian metric is reported for completeness only and no conclusion is drawn from it.

Finally, the secondary KITTI evaluation (Table 5.12) highlights the persistent difficulty of precise 3D bounding box regression. Because the KITTI metric enforces a strict Intersection over Union (IoU) overlap rather than a simple center-distance match, the mAP scores are drastically lower. While fine-tuning did rescue the Car mAP from 0.0001 to 0.0388 (PointPillars), the overall low values underscore that while the networks can successfully localize the center of targets in dense

Model	Car@0.5	Pedestrian@0.25
	mAP_R40	
PointPillars	0.0001	0.00
PillarNet	0.0002	0.00
PointPillars (Fine-tuned)	0.0388	0.00
PillarNet (Fine-tuned)	0.0243	0.0007

Table 5.12: Performance comparison between the anchor-based PointPillars and the CenterHead-based PillarNet architectures on the 3+1D proprietary dataset with Kitti R_40 evaluation

urban clutter, predicting their exact bounding box dimensions from sparse radar reflections remains highly challenging.

5.6.3 Optimization in ONNX and TRT

To bridge the gap between theoretical accuracy and real-world deployment on autonomous vehicles, the PointPillars architecture was subjected to rigorous post-training acceleration. Table 5.13 details the compute latency and dynamic memory footprint across the native PyTorch environment, ONNX Runtime (CPU and GPU), and NVIDIA TensorRT. It is important to note that the PyTorch latency measurements encompass both the model and the detection head, whereas the ONNX and TensorRT timings isolate the core model execution. Additionally, the initial point cloud voxelization remained implemented in PyTorch across all tests; this preprocessing step is explicitly excluded from the table's compute times and accounted for a constant overhead of approximately 0.8 ms per scan.

The native PyTorch implementation establishes a strong baseline, executing in an average of 4.0 ms with a peak memory consumption of 1739 MB. Exporting the model to ONNX and executing it strictly on the CPU predictably collapses performance, generating an unacceptable average latency of 73.7 ms and massive spikes up to 502.7 ms.

Surprisingly, while deploying ONNX on the GPU reduces the average compute time to 3.5 ms, it suffers from severe instability, producing maximum latency spikes of 70.1 ms. This bottleneck occurs because certain ONNX operations lack native GPU support for the specific input data types used by the network; consequently, the runtime is forced to perform expensive fallback computations

Aggregation	PyTorch	ONNX (CPU)	ONNX (GPU)	TensorRT
	Compute Time (ms)			
Average	4.0	73.7	3.5	0.75
Max	4.9	502.7	70.1	0.93
Min	3.1	51.3	3.0	0.74
	Memory (MB)			
Average of Maximum	1739	1683	1706	1476

Table 5.13: Performance comparison of optimized inference using ONNX and TensorRT against the original PyTorch implementation on Desktop system

on the CPU, heavily degrading overall GPU performance.

TensorRT emerges as the definitive solution for real-time edge deployment. By aggressively optimizing the computational graph and fusing kernels specifically for the target CUDA architecture, TensorRT slashes the average model inference time to a sub-millisecond 0.75 ms. Furthermore, it demonstrates exceptional stability with a maximum recorded latency of just 0.93 ms, while simultaneously reducing the peak memory footprint to 1476 MB. When combining this 0.75 ms TensorRT execution with the 0.8 ms PyTorch voxelization overhead, the entire perception pipeline comfortably operates well within the strict latency constraints required for high-speed autonomous driving.

5.7 Appendix: Experiments on LiDAR-Radar Cross-Modal Feature Augmentation

An additional experimental phase investigated the integration of LiDAR-radar cross-modal feature augmentation to determine if hallucinating LiDAR-like features could overcome the inherent sparsity of radar data. This approach was benchmarked against the standard PointPillars baseline across both 1-frame and 5-frame accumulation setups. The comparative results are detailed in Table 5.14.

The empirical evaluation reveals that the cross-modal feature augmentation provided almost

Model	**Recall** @ 0.3	**Recall** @ 0.5	**Recall** @ 0.7	**Car 3D** AP@0.5	**Pedestrian 3D** AP@0.25	**Bike 3D** AP@0.25	**Time** (ms)	**Memory Peak** (MB)
Baseline PP 1-frame	0.35	0.25	0.08	0.31	0.19	0.53	0.5	-
Baseline PP 5-frame	0.40	0.29	0.11	0.36	0.25	0.58	0.6	-
1-frame Aug	0.36	0.23	0.07	0.25	0.32	0.59	2.97	10545
5-frame Aug	0.38	0.27	0.10	0.35	0.29	0.59	5.69	10895

Table 5.14: Comparison of the performance between PointPillars and the network with feature augmentation from LiDAR, evaluated on VoD with 1 and 5 frames

no competitive advantage over the standard PointPillars baseline, with one specific exception: the detection of Vulnerable Road Users (VRUs). The augmentation technique yielded a noticeable improvement in pedestrian detection, particularly in the single-frame configuration, where the Pedestrian 3D AP increased from 0.19 to 0.32. Bike detection also saw marginal improvements.

However, these localized gains in VRU detection were overshadowed by a severe degradation in broader network performance. The feature augmentation method resulted in lower global recall across the stricter IoU thresholds and caused a significant drop in primary vehicle detection (e.g., the 1-frame Car 3D AP fell from 0.31 to 0.25).

Beyond the regression in global accuracy, the cross-modal feature augmentation introduces severe computational bottlenecks that render it unsuitable for edge deployment. As detailed in Table 5.14, standard PointPillars achieves highly efficient inference times of 0.5 ms and 0.6 ms. In contrast, the augmentation module increases latency drastically to 2.97 ms and 5.69 ms. Furthermore, the memory consumption of the augmented network is entirely out of scale, peaking at nearly 11 GB. To contextualize this massive overhead, the baseline PointPillars architecture evaluated on the TruckScenes dataset, which covers a significantly larger detection range and processes a substantially higher volume of voxels, peaks at a mere 1739 MB of memory. This extreme memory inflation, coupled with increased latency and reduced global recall, firmly justifies the decision to discard the cross-modal hallucination approach.

5.8 Appendix: Experiments on RadarGNN

While the core chapters of this thesis focus primarily on grid-based approaches (PointPillars and PillarNet), additional experiments were conducted to evaluate graph-based architectures, specifically RadarGNN. This architecture, as previously described, theoretically aligns well with the sparse nature of automotive radar.

To evaluate RadarGNN on the MAN TruckScenes dataset, a specific data preprocessing step was required to generate the necessary ground truth. Because RadarGNN operates on node (point) classifications rather than bounding box regression, radar points were assigned a specific class label strictly if they fell within the geometric boundaries of a ground-truth 3D bounding box.

However, this bounding-box-to-point mapping introduces significant label noise. As observed in subsection 4.4.2, raw radar points are rarely perfectly aligned with the exact physical boundaries of the target and fully contained in the bounding box due to the sensor's lower elevation accuracy and high susceptibility to multi-path reflections. Consequently, physically related reflections originating from the same object can easily fall just outside the bounding box, resulting in conflicting or missing labels for points that belong to the same target cluster. This inherent misalignment significantly contributes to lowering the network's overall learning stability and detection performance.

Class mAP_0.3	**RadarScenes** (original architecture)	**MAN TruckScenes** (original architecture)	**MAN TruckScenes** (architecture modified)
Car	0.72	0.60	0.62
Pedestrian	0.34	0.05	0.06
Truck	0.70	0.57	0.56

Table 5.15: Performance comparison of the RadarGNN architecture using 0.3 IoU threshold on the RadarScenes dataset and the MAN TruckScenes, default used by the authors [25]

The Table 5.15 shows the performance of the RadarGNN architecture on the MAN TruckScenes dataset, compared to the results obtained on the RadarScenes dataset. While results on different datasets are not directly comparable, the significant drop in mAP scores for the MAN TruckScenes dataset, especially for the pedestrian class, highlights the challenges of transferring graph-based

architectures to datasets with different sensor configurations and label characteristics.

mAP Class	**MAN TruckScenes** (original architecture)	**MAN TruckScenes** (architecture modified)
Car	0.51	0.53
Pedestrian	0.05	0.05
Truck	0.26	0.24

Table 5.16: Performance of the RadarGNN architecture using nuscenes evaluation metrics on the MAN TruckScenes

As evidenced by the results in Table 5.16, the mAP scores achieved by RadarGNN on the TruckScenes dataset remain noticeably lower than the performance metrics extracted from the grid-based methods evaluated in the main experimental phase.

Ultimately, while the RadarGNN design actively attempts to address the unique topological characteristics of radar data, its strict requirement for point-level semantic labels remains a critical limitation. In the current autonomous driving research landscape, the vast majority of multimodal datasets provide unified 3D bounding boxes rather than dense, point-wise segmentations. This lack of natively compatible ground truth restricts the broad applicability and scalability of graph-based models compared to standard bounding box detectors.

6. Conclusion and Future Horizons

The robust perception of complex environments remains a defining challenge in the progression of autonomous driving. While LiDAR and camera systems currently dominate the sensory suite, their susceptibility to adverse weather and lighting conditions necessitates a highly reliable alternative. This thesis investigated the capabilities of automotive radar as a standalone modality for 3D object detection, explicitly targeting high-speed applications such as overtake assist and heavy-duty truck platooning that demand extended perception horizons.

Through a comprehensive evaluation of state-of-the-art neural network architectures, this research bridged the gap between the theoretical promise of radar and the practical challenges of processing its sparse, multipath-heavy point clouds. Grid-based architectures, specifically PointPillars and PillarNet, were rigorously benchmarked. PillarNet, leveraging a 2D sparse convolutional encoder and an anchor-free CenterHead, emerged as the superior architecture. It effectively mitigated the computational waste of empty radar pillars while achieving significantly higher spatial precision and global recall compared to standard anchor-based methods. Conversely, specialized approaches such as LiDAR-radar cross-modal feature hallucination and graph-based networks (RadarGNN) were found to introduce severe memory bottlenecks or struggle with bounding-box-to-point mapping noise, rendering them less suitable for the targeted real-time edge applications.

A critical contribution of this work was addressing the severe domain bias present in public datasets. Recognizing that popular benchmarks often lack the long-range annotations required for high-speed planning, initial models were trained on the expansive MAN TruckScenes dataset. To tackle the subsequent domain shift from dynamic highway traffic to dense, static urban environments, a custom 3,429-frame 3+1D radar dataset was engineered. Utilizing an ego-motion compensated Kalman Filter and the Hungarian algorithm, the semi-automated labeling pipeline yielded a robust ground truth that enabled successful domain adaptation. Fine-tuning on this proprietary dataset drastically improved the networks' ability to resolve dense static clutter and accurately detect stationary vehicles.

Finally, this thesis evaluated the feasibility of transitioning high theoretical accuracy to future embedded deployment. By subjecting the optimized models to aggressive post-training accelera-

tion using ONNX and NVIDIA TensorRT on a desktop setup, steady-state inference latency was slashed to sub-millisecond levels (e.g., 0.75 ms for PointPillars), with peak dynamic memory consumption strictly bounded. These desktop-based optimizations serve as a strong proof of concept, demonstrating that the selected radar perception pipelines are well-equipped to eventually satisfy the stringent real-time constraints of edge computing in autonomous vehicles.

6.1 Future Works

Future work will focus on expanding the proprietary dataset to encompass a wider variety of scenarios, environments, Vulnerable Road Users and additional categories. Additionally, as interest in long-range radar perception grows, newly published large-scale open datasets such as TruckDrive—which features promising range claims but was released too late for inclusion in this study—will be thoroughly evaluated. Following the promising desktop-based optimizations, subsequent phases must validate the inference and latency performance directly on physical embedded hardware setups. This should be coupled with the ongoing evaluation of next-generation radar sensors to leverage their continuously improving precision and hardware specifications. On the algorithmic front, the tracking pipeline will be enhanced to better manage target non-linearities and the inherent noise of Doppler velocity measurements. Finally, rather than relying on the simple spatial aggregation of consecutive point clouds, future research should implement neural architectures natively designed to process the temporal dimension, explicitly exploiting the unique kinematic signatures and information measured within radar point clouds.

Acknowledgements

La prima persona che ci tengo a ringraziare è colei che ha creduto in me dal giorno 0... anzi, dal giorno -1. Senza il tuo istinto, che ti ha spinto ad andare oltre il parere dei medici, per me non solo non ci sarebbe stato un futuro, ma nemmeno un inizio. Ed eccoci qua: ne abbiamo passate tante insieme, ma con la tua presenza nei momenti belli e il tuo enorme supporto in quelli più difficili, guarda che traguardo ho raggiunto. Ti dico e ti dirò sempre grazie, Mamma.

A mio papà: non posso negare quanto tu mi manchi, e la tua assenza è stata un vero e proprio terremoto per me. So però che hai sempre fatto, e sempre farai, il tifo per me, ovunque tu sia.

A mia sorella: siamo come Tom e Jerry, ci piace stuzzicarci ma in fondo ci vogliamo bene. So che ci sarai sempre per me e io ci sarò sempre per te.

A Giandomenico: sei entrato in punta di piedi nella mia vita, ma con il tuo esempio mi hai spinto a essere un uomo migliore (e un po' più alcolista).

Ai miei nonni, che ci sono sempre stati e che mi hanno fatto sentire il loro affetto ogni giorno. Grazie nonna Silvana e grazie nonno Giorgio: anche se oggi non puoi essere qui, per me è come se lo fossi.

Un ringraziamento importante va a tutti i miei amici: da chi c'è sempre stato a chi è appena arrivato, da chi è dall'altra parte del mondo ma è pur sempre vicino, a chi ha condiviso con me solo pochi mesi che sono sembrati una vita intera. Una menzione speciale va ai miei compagni, nonché colleghi di università, che non solo sono stati un supporto fondamentale per me in questo percorso, ma anche persone speciali con cui ho condiviso tanto.

A tutto il team di Formula Student: a chi ha contribuito ogni giorno da quando sono entrato a farne parte e a chi è arrivato dopo. Sono stati anni pieni di difficoltà, ma averle affrontate con persone fantastiche ha reso certi momenti indimenticabili. E grazie Diletta, bella e dannata: chiunque ti abbia conosciuta ti odia e ti ama.

Chiuso un capitolo, se ne apre un altro, lungo, lunghissimo. Grazie a chi ne farà parte.

Bibliography

[1] Ekim Yurtsever et al. “A Survey of Autonomous Driving: Common Practices and Emerging Technologies”. In: *IEEE Access* 8 (2020), pp. 58443–58469. DOI: 10.1109/ACCESS.2020.2983149.

[2] Sampo Kuutti et al. “A Survey of Deep Learning Applications to Autonomous Vehicle Control”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.2 (2021), pp. 712–733. DOI: 10.1109/TITS.2019.2962338.

[3] Jiageng Mao et al. *3D Object Detection for Autonomous Driving: A Review and New Outlooks*. June 2022. DOI: 10.48550/arXiv.2206.09474.

[4] Tim Brühl et al. “Adaptive Radar Clustering and Tracking based on Point Criticality Assessment”. In: *2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2025, pp. 490–496. DOI: 10.1109/SMC58881.2025.11343545.

[5] Igal Bilik et al. “Automotive multi-mode cascaded radar data processing embedded system”. In: *2018 IEEE Radar Conference (RadarConf18)*. 2018, pp. 0372–0376. DOI: 10.1109/RADAR.2018.8378587.

[6] Roger Narayan. *Encyclopedia of Sensors and Biosensors*. Elsevier, 2023.

[7] Florian Kraus et al. “The Radar Ghost Dataset – An Evaluation of Ghost Objects in Automotive Radar Data”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2021, 8570–8577. DOI: 10.1109/iros51168.2021.9636338. URL: http://dx.doi.org/10.1109/IROS51168.2021.9636338.

[8] Sergios Theodoridis Rama Chellappa. *Academic Press Library in Signal Processing, Volume 7 Array, Radar and Communications Engineering*. Elsevier, 2018.

[9] Ignacio Roldan et al. “See Further Than CFAR: a Data-Driven Radar Detector Trained by Lidar”. In: *2024 IEEE Radar Conference (RadarConf24)*. IEEE, May 2024, 1–6. DOI: 10.1109/radarconf2458775.2024.10548426. URL: http://dx.doi.org/10.1109/RadarConf2458775.2024.10548426.

[10] Jozef Perďoch et al. “CFAR Algorithm for Improving Detections on Radar Raw Data Matrices”. In: *2022 New Trends in Signal Processing (NTSP)*. 2022, pp. 1–6. DOI: 10.23919/NTSP54843.2022.9920396.

[11] Zeyu Han et al. *4D Millimeter-Wave Radar in Autonomous Driving: A Survey*. 2024. arXiv: 2306.04242 [eess.SP]. URL: https://arxiv.org/abs/2306.04242.

[12] Merrill I. Skolnik. *Introduction to Radar Systems – 2nd edition*. McGraw-Hill Education, 1980.

[13] Ahmad Taher Azar Anis Koubaa. *Unmanned Aerial Systems*. Elsevier, 2021.

[14] Yi Zhou et al. “Towards Deep Radar Perception for Autonomous Driving: Datasets, Methods, and Challenges”. In: *Sensors* 22.11 (2022). ISSN: 1424-8220. DOI: 10.3390/s22114208. URL: https://www.mdpi.com/1424-8220/22/11/4208.

[15] Shunqiao Sun. “MIMO radar technology”. In: *Modern Radar for Automotive Applications*. 2022. Chap. Chapter 3, pp. 57–94. DOI: 10.1049/SBRA553E_ch3. eprint: https://digital-library.theiet.org/doi/pdf/10.1049/SBRA553E_ch3. URL: https://digital-library.theiet.org/doi/abs/10.1049/SBRA553E_ch3.

[16] Kerem Sen, Sinan Aksimsek, and Ali Kara. “Dynamic RCS Modeling and Aspect Angle Analysis for Highly Maneuverable UAVs”. In: *Aerospace* 11.9 (2024). ISSN: 2226-4310. DOI: 10.3390/aerospace11090775. URL: https://www.mdpi.com/2226-4310/11/9/775.

[17] Stefan Muckenhuber, Eniz Museljic, and Georg Stettinger. “Performance evaluation of a state-of-the-art automotive radar and corresponding modeling approaches based on a large labeled dataset”. In: *Journal of Intelligent Transportation Systems* 26 (Aug. 2021), pp. 655–674. DOI: 10.1080/15472450.2021.1959328.

[18] Andras Palffy et al. “Multi-Class Road User Detection With 3+1D Radar in the View-of-Delft Dataset”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4961–4968. DOI: 10.1109/LRA.2022.3147324.

[19] Keenan Burnett, Angela P. Schoellig, and Timothy D. Barfoot. *Continuous-Time Radar-Inertial and Lidar-Inertial Odometry using a Gaussian Process Motion Prior*. 2024. arXiv: 2402.06174 [cs.RO]. URL: https://arxiv.org/abs/2402.06174.

[20] Peter G. Steeneken. *Introductory Dynamics*. TU Delft Open Publishing, 2025, pp. 135–138.

[21] Dominik Kellner et al. “Instantaneous ego-motion estimation using Doppler radar”. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. 2013, pp. 869–874. DOI: 10.1109/ITSC.2013.6728341.

[22] Holger Caesar et al. *nuScenes: A multimodal dataset for autonomous driving*. 2020. arXiv: 1903.11027 [cs.LG]. URL: https://arxiv.org/abs/1903.11027.

[23] Igal Bilik. “Comparative Analysis of Radar and Lidar Technologies for Automotive Applications”. In: *IEEE Intelligent Transportation Systems Magazine* 15.1 (2023), pp. 244–269. DOI: 10.1109/MITS.2022.3162886.

[24] Yuval Haitman and Oded Bialer. *DoppDrive: Doppler-Driven Temporal Aggregation for Improved Radar Object Detection*. 2025. arXiv: 2508.12330 [cs.CV]. URL: https://arxiv.org/abs/2508.12330.

[25] Felix Fent, Philipp Bauerschmidt, and Markus Lienkamp. “RadarGNN: Transformation Invariant Graph Neural Network for Radar-based Perception”. In: *arXiv preprint arXiv:2304.06547* (2023).

[26] Alex H. Lang et al. *PointPillars: Fast Encoders for Object Detection from Point Clouds*. 2019. arXiv: 1812.05784 [cs.LG]. URL: https://arxiv.org/abs/1812.05784.

[27] Dong-Hee Paek and Seung-Hyun Kong. *Availability-aware Sensor Fusion via Unified Canonical Space*. 2025. arXiv: 2503.07029 [cs.CV]. URL: https://arxiv.org/abs/2503.07029.

[28] D.-H. Paek, S.-H. Kong, and K. T. Wijaya. “K-radar: 4D radar object detection for autonomous driving in various weather conditions”. In: *Proc. 36th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track*. Vol. 35. 2022, pp. 3819–3829. URL: https://openreview.net/forum?id=W_bsDmzwaZ7.

[29] Seung-Hyun Kong, Dong-Hee Paek, and Sangjae Cho. *RTNH+: Enhanced 4D Radar Object Detection Network using Combined CFAR-based Two-level Preprocessing and Vertical Encoding*. 2023. arXiv: 2310.17659 [eess.SP]. URL: https://arxiv.org/abs/2310.17659.

[30] Guangsheng Shi, Ruifeng Li, and Chao Ma. *PillarNet: Real-Time and High-Performance Pillar-based 3D Object Detection*. 2022. arXiv: 2205.07403 [cs.CV]. URL: https://arxiv.org/abs/2205.07403.

[31] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, 21–37. ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[32] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. *Center-based 3D Object Detection and Tracking*. 2021. arXiv: 2006.11275 [cs.CV]. URL: https://arxiv.org/abs/2006.11275.

[33] Jianning Deng et al. *Robust 3D Object Detection from LiDAR-Radar Point Clouds via Cross-Modal Feature Augmentation*. 2024. arXiv: 2309.17336 [cs.CV]. URL: https://arxiv.org/abs/2309.17336.

[34] Ole Schumann et al. *RadarScenes: A Real-World Radar Point Cloud Data Set for Automotive Applications*. 2024. arXiv: 2104.02493 [cs.LG]. URL: https://arxiv.org/abs/2104.02493.

[35] Aditya Iyer, Jack Roberts, and Nora Ayanian. *ModTrack: Sensor-Agnostic Multi-View Tracking via Identity-Informed PHD Filtering with Covariance Propagation*. 2026. arXiv: 2603.15812 [cs.CV]. URL: https://arxiv.org/abs/2603.15812.

[36] Felix Fent et al. *MAN TruckScenes: A multimodal dataset for autonomous trucking in diverse conditions*. 2024. arXiv: 2407.07462 [cs.CV]. URL: https://arxiv.org/abs/2407.07462.

[37] Filippo Ghilotti et al. "TruckDrive: Long-Range Autonomous Highway Driving Dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2026.

[38] E Li et al. "SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1108–1115. DOI: 10.1109/IV47402.2020.9304562.

[39] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[40] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) challenge". In: *International Journal of Computer Vision* 88 (June 2010), pp. 303–338. DOI: 10.1007/s11263-009-0275-4.

[41] Andrea Simonelli et al. "Disentangling Monocular 3D Object Detection: From Single to Multi-Class Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.3 (2022), pp. 1219–1231. DOI: 10.1109/TPAMI.2020.3025077.

[42] Kshitiz Bansal et al. "Pointillism: accurate 3D bounding box estimation with multi-radars". In: *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. SenSys '20. ACM, Nov. 2020, 340–353. DOI: 10.1145/3384419.3430783. URL: http://dx.doi.org/10.1145/3384419.3430783.

[43] Bo Yang and Hua Zhang. "A CFAR Algorithm Based on Monte Carlo Method for Millimeter-Wave Radar Road Traffic Target Detection". In: *Remote Sensing* 14 (Apr. 2022), p. 1779. DOI: 10.3390/rs14081779.