A Unified Multi-Frame Strategy for Autonomous Vehicle Perception and Localization Using Radar, Camera, LiDAR, and HD Map Fusion

by

Ahmad Reza Alghooneh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Changzhi Li Professor, Dept. of Electrical and Computer Engineering, Texas Tech University
Supervisor(s):	Amir Khajepour Professor, Dept. of MME, University of Waterloo George Shaker Adjunct Associate Professor, Dept. of ECE, University of Waterloo
Internal Member:	HJ Kwon Associate Professor, Dept. of MME, University of Waterloo
Internal-External Member:	Patrick Mitran Professor, Dept. of ECE, University of Waterloo
Internal-External Member:	John Zelek Associate Professor, Dept. of SYDE, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents a novel unified perception-localization module by developing an advanced late fusion system that integrates radar, LiDAR, camera, and High Definition (HD) map information. Emphasizing reliability, robustness to inclement weather, and real-time implementation, this approach particularly focuses on radar data—the most crucial component in the fusion process. Although radar data is rich in features and robust against weather changes, it is often with false alarms and clutters. These issues pose challenges to the fusion process if not addressed. Therefore, the radar point cloud is first refined to remove these false alarms and clutter. Unlike existing approaches in the literature that require massive neural networks, this thesis presents a stochastic alternative that provides the same level of accuracy while maintaining real-time performance and minimal resource usage. By utilizing statistical information from the HD map, the system develops a likelihood of object occurrences in the Frenet coordinate system. This likelihood is then fused with a specific set of radar features to formulate a probabilistic classifier. The classifier focuses on detecting clutter and false alarms, removing them to prevent the radar point cloud from misleading the downstream fusion process. This approach significantly reduces noise and irrelevant data, precision up to 94 percent on our dataset, ensuring that the radar primarily focuses on potential obstacles and critical elements such as cars and pedestrians. The **video** of the results is available here.

In the subsequent phase, the refined radar data is fused with LiDAR and camera data using a novel, robust frustum approach in a late-fusion manner. Unlike previous works that rely solely on perspective equations, our method constructs a cost function and employs the Hungarian matching algorithm to associate camera data with the 3D positions of objects. This late-fusion methodology significantly enhances overall detection accuracy and reliability by effectively integrating data from multiple sensors. The resulting objects are then fed into a novel tracking module that leverages the radar's radial velocity measurements as partial observations while associating objects across frames using the Shortest Simple Path algorithm and radar features. For each individual object, the tracking module builds an Extended Kalman Filter, providing a full-state estimation of their motion states, including position, velocity, and acceleration. The **video** showing the performance of the perception, and the **one** for the tracking.

In the localization phase, the previously processed perception information is utilized to improve positioning accuracy without relying on the Global Positioning System (GPS). This approach ensures a unified module that substantially reduces resource consumption. Moreover, methods relying on GPS and inertial sensors such as the Inertial Measurement Unit (IMU) are susceptible to weather and road conditions, which can challenge the reliability of autonomous driving. In contrast, a robust perception unit can overcome this challenge by using radar, a sensor resilient to inclement weather. In this approach, the fused radar-LiDAR point cloud is used to form a local map across multiple frames, and with the help of the Iterative Closest Point (ICP) algorithm, the vehicle's odometry is estimated. Then, using camera-based lane detection, the lateral distance to the centerline and curb is appended to the odometry measurement. Finally, the vehicle's motion model is rewritten in the Frenet frame, incorporating the measurement models from the radar-LiDAR point cloud and camera lane detection in an Extended Kalman Filter (EKF) formulation. This approach outperforms state-of-the-art positioning solutions in both accuracy and resource consumption. The **video** of the performance with the localization in the loop.

Acknowledgements

I would like to express my deepest gratitude to several individuals and organizations who have contributed significantly to the completion of this thesis.

First and foremost, I am profoundly grateful to my supervisors, Professor Amir Khajepour and Professor George Shaker. Their exceptional guidance, invaluable insights, and unwavering support have been instrumental in shaping the direction and quality of this research. Their encouragement and expertise have profoundly influenced my academic and professional growth. Their constructive criticism, coupled with their patience and encouragement, provided me with the confidence to navigate the complexities of this research. I am deeply thankful for their mentorship and for believing in my capabilities even during challenging times. I would also like to extend my sincere appreciation to the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support. A special thanks go to my friends and colleagues whose support and camaraderie have been invaluable throughout this journey. Minghao Ning, without whom, the development of the perception module was not possible, and the rest of my friends and colleagues Victoria Yang, Pouya Panahandeh, Chen Sun, Ehsan Mohammadbagher, Ruihe Zhang, and Ted Ecclestone – your encouragement, constructive discussions, and unwavering friendship have made this challenging endeavor both rewarding and enjoyable. The countless hours spent discussing ideas, troubleshooting problems, and sharing knowledge have been a source of inspiration and motivation. Additionally, I extend my heartfelt thanks to my family for their unconditional love and support. Their belief in me has been a constant source of strength and motivation. My parents' unwavering support and my siblings' encouragement have provided me with the resilience needed to persevere through my academic life. Finally, to everyone who has contributed to this journey, directly or indirectly, please accept my heartfelt thanks. Your contributions have been essential to the successful completion of this thesis.

Dedication

To my brother, Mansoor! To my mother, Goli! to all my family!

Table of Contents

Examining Committee ii

Author's Declaration iii

Abstract iv

Acknowledgements vi

Dedication vii

List of Figures xii

List of Tables xv

1 Introduction 1

1.1 Motivation 1

1.2 Objectives 4

1.3 Thesis Outline 6

2 Literature Review 8

2.1 Radar Fundamentals 8

2.1.1 Range Measurement for FMCW radar . . . 10
2.1.2 Range Resolution . . . 11
2.1.3 Velocity Measurements . . . 11
2.1.4 Angle Measurement . . . 12
2.2 Multi-Modal Object Detection . . . 13
2.3 Object Tracking and Prediction . . . 17
2.4 Localization . . . 18
2.5 Summary . . . 20

3 Radar-HD Map Perception System 21
3.1 Problem Formulation . . . 21
3.2 Radar HD Map Fusion . . . 24
3.2.1 ROI and Velocity Compensation for the Point Cloud . . . 25
3.2.2 Multi-Frame Two Staged Consistent Clustering . . . 27
3.2.3 Geometrical and Radar Feature Extraction . . . 29
3.2.4 Fusion with HD Map Information . . . 32
3.2.5 Cost-Map Creation . . . 33
3.3 Summary . . . 34

4 Radar-LiDAR-Camera-HD map Perception and Localization System 35
4.1 Perception . . . 35
4.1.1 LiDAR Point Cloud Clustering . . . 39
4.1.2 Radar-LiDAR Fusion System . . . 48
4.1.3 Camera Perception . . . 51
4.1.4 Camera-3D Point Association . . . 53

4.1.5 Radar-LiDAR-Camera Fusion . . . 56
4.2 Object Tracking . . . 56
4.2.1 Individual Object State Observer . . . 56
4.2.2 Scaling The State Observers . . . 60
4.3 Localization . . . 63
4.3.1 Accurate Motion Model . . . 64
4.3.2 Adjusted ICP Approach . . . 65
4.4 Final Fusion . . . 66
4.5 Summary . . . 69

5 Experiments and Results **70**
5.1 Experiments . . . 70
5.2 Dataset . . . 73
5.3 Results and Discussion . . . 73
5.3.1 Improvements of Radar Precision with HD map Fusion . . . 76
5.3.2 Perception Results and Comparative Study . . . 79
5.3.3 Localization Results and Comparative Study . . . 88
5.3.4 Unified Localization and Perception Computational Performance . . . 96
5.4 Applications . . . 98

6 Conclusion **101**
6.1 Conclusion . . . 101
6.2 Future Work . . . 102

References **104**

APPENDICES **116**

A Technical Concerns Regarding Radar **117**

A.1 Automotive Radars 117

A.2 Installation Concerns and Calibration process 118

A.2.1 Operating Principles 118

A.2.2 Applications in ADAS and Autonomous Vehicles 119

A.3 Installation Concerns and Calibration Process 120

A.3.1 Installation Concerns 120

A.3.2 Calibration Process 121

A.4 Radar models used in this study 121

A.5 Common Automotive Radar Sensor Interfaces 122

A.6 CAN interface 123

A.6.1 Overview of CAN Interface 123

A.6.2 Technical Specifications 123

A.6.3 CAN Protocol Layers 124

A.6.4 Applications in Automotive Systems 124

A.6.5 Advantages of CAN Interface 125

A.7 CAN driver 125

A.8 Radar Full-State Observer Real-Time C++ Code 136

List of Figures

1.1 The overview of the unified perception-localization method. 5

3.1 A schematic of the radar point cloud manipulation and cleansing, and the cost-map generation algorithm has been brought here. First, a sequence of the radar point clouds are consistently concatenated on one another, then with the aid of the HD map the point cloud is focused on the road, removed from false alarms and clutters, and added HD map features for the final classifier. Finally the output is used to form the cost-map. 22

3.2 Radar point cloud in the frenet frame, the black dots are the s-points on the HD map, and the blue arrows are for the lateral distance in (s, d) frame. The red arrow represents the magnitude of the s coordinate. The blue dots are the radar point clouds. The coordinate attached to the bus is the radar frame. 23

3.3 Considering the ego vehicle traveling with linear velocity v_e and turning with yaw rate of $\dot{\psi}_e$, the radar relative radial velocity can be compensated by projecting the ego velocity into the object's radial unit vector and then adding up with the relative radial velocity. 26

3.4 Same scenario as the Figure 3.2, however, here the location of the pedestrian with the vehicle are switched to emphasize the unlikeliness of this scenario to happen. 31

4.1 Overview of Early, Mid, and Late fusion. 38

4.2 The adaptive removal method. On the left the road is divided into a grid format where the closer the grids are to the vehicle, the tighter the grids are, and it relaxes as it moves away. On the right, the overall of the plane fitting method is shown. 43

4.3 The figure to emphasize on the necessity of a adaptive clustering approach for range sensors. 46

4.4 The YOLOv8 architecture [28] 52

4.5 The Frustum approach for camera and 3D point association. 54

4.6 The figure to show the velocity in the world coordinate in relation to the compensated doppler velocity. The world coordinate is the East-North frame, and to improve robustness, the observer is designed in this frame. 58

4.7 The final fusion where the perception and localization are unified. 62

4.8 Bicycle model in Frenet frame 64

5.1 Illustration of WATonoBus sensor suite, compute system, control interface, and visualization utilities. 71

5.2 The features used for this study are mostly built with the RCS features of the cluster. To prove that the method stays consistent in performance across different weathers, aside from the metric results, the consistency in the distributions of these features is shown. . 74

5.3 The geometrical features that emphasizes on the point density of the radar point cloud. As shown in the figure, this also shows consistency across the weathers. 75

5.4 The qualitative results of the perception module. In the above, two consecutive frames are shown, and its below, there are the camera frames from front, right, and left side cameras. The purple line is the footprint of each object from tracking, and the green vector is the scale of the velocity. the red points are the objects detected by the perception module. . 80

5.5 Geese detection as an example of irregular small object detection. Though the YOLO was not trained on Geese class, the module detected them, and the bus stopped for them. [5]. 81

5.6 Detection performance in heavy snow conditions. 84
5.7 Caption for LOF 87
5.8 Vehicle Frenet coordinate estimate 90
5.9 Vehicle yaw and yaw rate 91
5.10 Vehicle estimated velocity and arc length rate 92
5.11 Vehicle lateral rates 93
5.12 Vehicle lateral, curvature, and their rate 94
5.13 Application of the unified Perception-Localization Module in the Human Follower Software 96
5.14 The cost-map results from a rainy day on the campus. As it can be seen the vision is completely disturbed, shown on the right, but the radar managed to create a reliable cost-map, shown on the left. The images in the middle are for the radar point cloud put on top of the LiDAR point cloud; the LiDAR points are used here to help visualization of the scene. The black dots are false alarms and clutters. The pink points are the detected objects that here are pedestrians, and metal guard rail by the road, they are better shown by the orange circles. 97
5.15 The schematics for the Red-VS-Blue approach, in which the tracking and prediction are utilized to classify the actors to those who come into interaction with the vehicle and those who doesn't. 98

List of Tables

5.1 Sensor specifications and details. 72

5.2 Random Forest without HD map features evaluation 76

5.3 Random Forest with HD map features evaluation 77

5.4 Random Forest with HD map features and likelihoods evaluation 77

5.5 Detection Evaluation 86

5.6 RMSE and MAE calculated for our localization (EKF) and Kiss-ICP . . . 89

Chapter 1

Introduction

This chapter discusses the motivation behind the research undertaken in this thesis. It starts with a summary of highlighted works from the existing literature, emphasizing the recent advancements in the field of Autonomous perception and localization that frame the context and necessity of this study. Subsequently, a statement of the objectives outlines the specific aims and contributions of the work, providing a clear road map of the intended research outcomes. The chapter concludes by presenting a detailed outline of the thesis, which organizes the structure of the document and guides the reader through the sequential development of the methodologies employed.

1.1 Motivation

Autonomous vehicles (AVs) are at the forefront of advanced technology, promising a future of safe and efficient transportation [88]. These vehicles heavily rely on a robust perception system to accurately interpret and understand their surroundings, which is crucial for making informed decisions while navigating through complex environments. To provide a reliable perception system, AVs are equipped with a variety of sensors, including cameras, radars, LiDARs, and etc [3]. Among these sensors, cameras and radars are widely used due to their maturity, cost-effectiveness, and complementary properties [85, 58, 63, 48, 49].

A majority of work in the literature in the area of perception is done on object detection. The existing literature predominantly concentrates on methodologies employing LiDAR [36, 93, 63] and fusion of LiDAR-camera [58, 13]. LiDAR and camera sensors, which rely on light waves for active or passive sensing, are significantly affected by weather conditions. The short wavelengths of these sensors are susceptible to scattering by snow or rain droplets, introducing substantial noise and severely hindering the detection process. For LiDAR, this results in noisy point clouds that complicate accurate object detection. In the case of cameras, adverse weather conditions can disrupt the visual input, posing challenges for neural networks trained to detect objects under normal conditions. Even with extensive training to handle disturbed vision, camera-based depth detection remains unreliable and imprecise.

In contrast, radar sensors operate using radio waves with much longer wavelengths, which are not as easily affected by precipitation. This characteristic allows radar waves to penetrate snow and rain droplets, making radar a robust perception modality for autonomous vehicles. However, radar sensors come with their own set of challenges, primarily the high incidence of false alarms, clutter, and lower resolution compared to LiDAR and cameras. Although advanced automotive radars with a large number of antennas have been developed to improve resolution, the increased number of detections also leads to a higher rate of false alarms and clutter. Despite these challenges, radar remains an intriguing modality for autonomous vehicle applications due to its robustness in adverse weather conditions. Nonetheless, there is a noticeable scarcity of research specifically focused on object detection using radar technology. One plausible explanation for this research gap is the absence of a reliable dataset incorporating radar data until the publishing of the 2019 NuScene dataset [10]. Within the limited scope of radar-based object detection, the majority of studies have been inclined towards leveraging radar-camera fusion approaches to accomplish the task [50, 27, 48]. However, these methods are compromised when visual conditions are poor, i.e., during rain, snow, or dense fog, where camera visibility is significantly reduced. For a perception system to be deemed robust, it must maintain consistent performance even in such challenging environments. Therefore, some of the literature focused on the radar-only methods [53, 37, 89, 89, 90]. Much of the existing research relies on deep learning techniques that necessitate expensive GPUs, and often require transfer

learning to adapt to changes in the environment or sensors. Methods executable on CPUs tend to exhibit reduced accuracy. Yet, with the advent of open-source platforms that provide widespread access to the HD maps, there remains a gap for a method that leverages this data without depending on deep neural networks. Moreover, these methods lack reliability, as the deep neural network exhibit poor performance in the case of situations that they are not trained on. With all the advancement in the neural networks, the case of unknown objects still pose a serious challenge to these methods.

Localization as another piece in the AV operation, plays an important role in the domain of autonomous vehicle (AV) driving, acting as a crucial counterpart to the perception system. While perception allows AVs to make decisions based on environmental stimuli, it is localization that tracks the consequences of those decisions. In essence, perception helps AVs decide, while localization ensures that those decisions are effectively followed. Localization forms a vital feedback loop for the AV controller, enabling it to continually adjust and refine driving actions based on real-time positional information. Without robust localization capabilities, AVs would lack the necessary feedback mechanism to ensure precise navigation and safe operation in dynamic environments. Localization in AVs faces significant challenges, particularly in harsh environmental conditions and dense urban areas. In these scenarios, GPS signals can be obstructed or weakened by tall buildings, bridges, tunnels, and adverse weather conditions, leading to inaccuracies in determining the vehicle's position. Additionally, relying solely on GPS for localization may not provide the required level of precision for safe navigation, especially in environments where centimeter-level accuracy is crucial. Addressing these challenges is imperative to ensure the reliability and safety of AVs, emphasizing the need for robust localization solutions that can operate across diverse environments. Perception algorithms are integral to localization systems. These algorithms not only identify objects but also extract features crucial for localization, such as lane markings and road boundaries. The lane detection algorithm [4] stands as a method capable of enhancing vehicle localization by supplying center lane information to AVs. Given the diverse sources of information, filter-based algorithms [79, 72], such as the Extended Kalman Filter (EKF), can be employed to fuse various data sources, ensuring accurate knowledge about vehicle states.

These challenges and opportunities underpin the objectives of this study, which are

explained in detail in the next section.

1.2 Objectives

The primary objective of this thesis is to design and implement a robust multi-sensor fusion system for autonomous vehicles unified perception-localization module, enhancing their reliability in diverse environments. The proposed system merges inputs from radar, high-definition (HD) maps, cameras, and LiDAR data. This integration facilitates an understanding of the vehicle's surroundings by leveraging the unique strengths of each sensor type, e.g., while radar provides reliable data through adverse weather conditions, LiDAR offers high-resolution information about the vehicle's surroundings, and cameras contribute rich contextual and textural details, by combining these sensors, the system aims to achieve a high level of accuracy in real-time perception and localization tasks, essential for dynamic and complex driving scenarios.

The second objective is to ensure that the multi-sensor fusion system maintains high performance in the inclement weather. The system is specifically designed to counter issues like reduced visibility, or noise, which are common in conditions such as fog, heavy rain, or snow. Techniques such as adaptive filtering, sensor redundancy, and context-aware processing are utilized to mitigate the effects of environmental factors on sensor performance, ensuring that the system remains robust and reliable regardless of the weather.

The last objective of the thesis is to maintain real-time performance of the unified module. The integration of advanced algorithms is pivotal in processing the diverse and voluminous data from multiple sensors in real-time. This will ensure the immediate integration of the unified localization and perception module into the AV technology while maintaining the standards of industrial developments.

Through achieving these objectives, this study introduces multiple contributions to both perception and localization as follows:

- *A novel way to fuse HD map information*: This study presents a new method to fuse HD map information into a probabilistic classifier for object detection by incorpo-

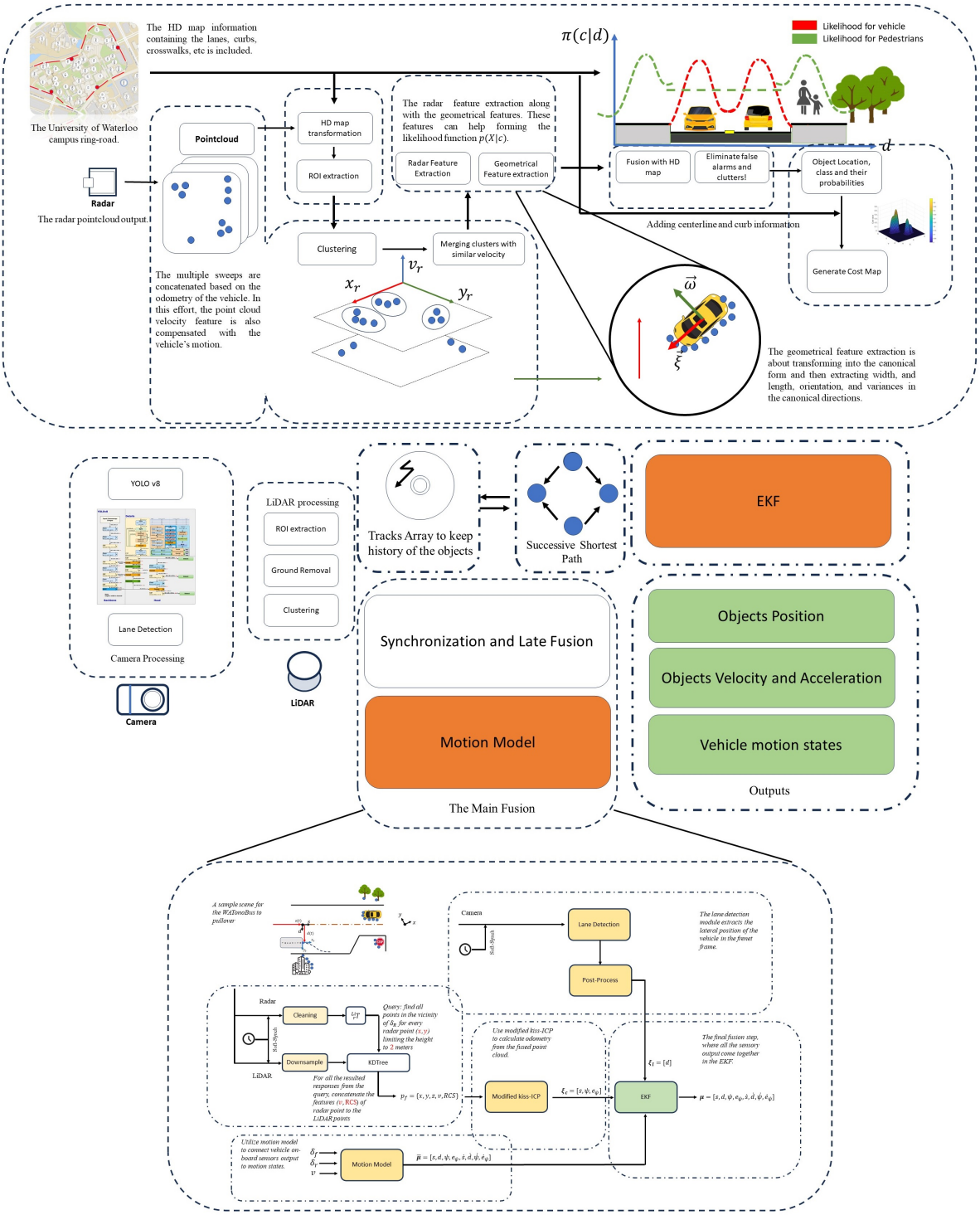

Figure 1.1: The overview of the unified perception-localization method.

rating statistical information about object occurrences in different locations of the map. This creates a likelihood that enhances the classification algorithm's accuracy.

- *A robust perception module able to detect and track irregular objects*: The proposed method overcomes challenges faced by previous perception algorithms when detecting irregular objects of different types. Unlike previous works, this method operates based on clusters, eliminating the need for training on irregular objects. The perception module can detect small objects, such as geese, which pose serious challenges to other methods.

- *A novel robust frustum approach*: Additionally, this study introduces a novel robust frustum approach for associating camera 2D information with the 3D location of objects. The object features in the camera, LiDAR, and radar frames help form a cost function that associates each object with the most likely match using a Hungarian matching algorithm.

- *A multi-sensor fusion for localization*: In the localization, this thesis presents a stochastic fusion method that leverages motion and vision sensors to come up with a robust localization method.

1.3 Thesis Outline

The structure of the thesis is organized as follows: Chapter 2 presents a comprehensive review of existing literature in the fields of perception and localization, highlighting key developments and identifying gaps that the current research aims to fill. In Chapter 3, the methodology and implementation details of the radar-HD map fusion approach are elaborated, discussing its integration with the overall system and its specific contributions to enhancing perception accuracy. Chapter 4 introduces the unified module for perception and localization. This chapter details the integration of multimodal sensor data and describes the innovative algorithms developed to ensure robust and accurate localization and perception in a variety of environmental conditions. In the Chapter 5, the details of the experiments validating the unified perception-localization approach, along with the

application of the unified perception-localization method is presented. Finally, in the last Chapter 5, the conclusion of the work, along with the possible future work is presented. In the Appendix A, the essential technical details of the sensory setup and the implementation code in C++ is presented.

Chapter 2

Literature Review

This chapter goes over a concise introduction of radar fundamentals, as it is the key element of all-weather fusion, and the related works in the literature in the areas of AV perception and localization. First, it discusses the highlights in the literature on multi-modal object detections, then it dives into the literature of object tracking, and finally it discusses the previous works on the localization. In the final section, a summary is presented that concludes the current research gaps in the literature.

2.1 Radar Fundamentals

Radar, which stands for Radio Detection and Ranging Device, operates using radio waves across a frequency range of 20 kHz to 300 GHz. However, for autonomous vehicle (AV) applications, the focus is typically on a narrower band, specifically 20 GHz to 300 GHz. This frequency range corresponds to millimeter-scale wavelengths, which is why AV radars are often referred to as millimeter wave (mmWave) radars. This thesis is exclusively concerned with the fundamentals of mmWave radar systems [32], [68].

Millimeter wave radars use short-wavelength electromagnetic waves. They work by transmitting electromagnetic signals toward objects and capturing the reflected waves, enabling the determination of an object's range, velocity, and angle. One key advantage

of mmWave radars is their compact antenna size, a result of the short wavelengths they use. Additionally, their ability to detect motion at sub-millimeter precision—such as a system operating at 77–81 GHz—makes them particularly suitable for AV applications. However, they also present a challenge: their high operating frequencies require proportionally high sampling rates and advanced processing units, including high-performance analog-to-digital converters (ADCs), microcontrollers (MCUs), and digital signal processors (DSPs). Moreover, mmWave radars have a shorter effective range (0.2–300 meters) compared to MHz-based radars, which can operate over several kilometers. Despite this limitation, their range is well-suited to AV applications and surpasses other modalities like LiDAR and stereo vision in standard AV setups.

Two common types of mmWave radar technologies used in AVs are Pulse Wave Radar and Frequency-Modulated Continuous-Wave (FMCW) Radar.

- Pulse Wave Radar operates by emitting short bursts of electromagnetic waves from its antenna. These bursts occur intermittently, allowing the waves to reach a target, reflect, and return before the next burst is transmitted. By measuring the time delay of the return signal, the distance to the target can be calculated, as the speed of the waves is known. A drawback of pulse radars is their limited bandwidth, which directly impacts resolution. Although higher bandwidths are possible, they come with significant cost increases compared to FMCW radars. Bandwidth, which determines range resolution, is discussed in the next section.

- Frequency-Modulated Continuous-Wave (FMCW) Radar transmits a continuous electromagnetic signal that is linearly modulated over time, sweeping from a lower to a higher frequency to cover a specific bandwidth. The reflected continuous wave is processed by mixing it with the transmitted signal. After low-pass filtering, a low-frequency intermediate signal is produced. This signal can be digitized using cost-effective ADCs operating at low sampling frequencies, eliminating the need for sequential sampling. FMCW radars, therefore, offer high bandwidth and excellent resolution at an affordable cost.

When choosing radar technology for AVs, the decision typically depends on the required accuracy and budget. FMCW radars are often preferred for their high bandwidth

and superior resolution. While pulse radars can achieve similar resolutions, their higher associated costs make them less practical for most applications. For this thesis, an FMCW radar is used, and the following section delves into its mathematical foundations, including range, velocity, and angle measurement, based on technical documentation from Texas Instruments.

2.1.1 Range Measurement for FMCW radar

The fundamental concept in radar systems is the transmission of an electromagnetic signal that reflects off objects in its path. In Frequency-Modulated Continuous-Wave (FMCW) radars, the signal frequency increases linearly over time. This type of signal is also called a *chirp*. An FMCW radar system transmits a chirp signal and captures the reflected signals from objects in its path. These reflected signals, known as RX (received) signals, are combined with the TX (transmitted) signal in a *mixer* to produce an *Intermediate Frequency (IF)* signal. A frequency mixer is an electronic component that combines two signals to create a new one. For example, if x_1 and x_2 are two signals, the mixer output is given by:

$$x_1(t) = \sin(\omega_1 t + \phi_1), \tag{2.1}$$

$$x_2(t) = \sin(\omega_2 t + \phi_2), \tag{2.2}$$

$$x_{\text{out}}(t) = \sin((\omega_2 - \omega_1)t + \phi_2 - \phi_1). \tag{2.3}$$

Here, ω_1 and ω_2 represent the angular frequencies of the two signals, and ϕ_1 and ϕ_2 are their respective phases. On the other hand, the RX signal is a delayed version of the TX signal. This delay causes the *IF* signal of a particular object to have a constant frequency. The time delay τ is given by:

$$\tau = \frac{2d}{c} = \frac{\text{IF}}{S}, \tag{2.4}$$

where d is the distance to the object, c is the speed of light, and S is the slope of the chirp. From this, the distance d can be derived as:

$$d = \frac{c\,\text{IF}}{2S}. \tag{2.5}$$

Thus, if the *IF* is known, the range of the object can be determined.

2.1.2 Range Resolution

Another key concept in radar systems is *range resolution*, which refers to the ability to distinguish between two or more closely spaced objects. When objects move closer, there comes a point at which the radar system can no longer distinguish them as separate objects. According to Fourier transform theory, with an observation window of T, the radar can only distinguish frequencies separated by more than $1/T$ Hz. If ΔF is the frequency difference, the range resolution can be expressed as:

$$\Delta F = \frac{1}{T} = \frac{2S\Delta d}{c}, \tag{2.6}$$

where Δd is the range resolution. This relationship illustrates how the radar's resolution is determined by its bandwidth and signal processing capabilities. Building on the earlier discussion, replacing the bandwidth formula, $BW = ST$, yields the range resolution as follows:

$$\Delta d = \frac{c}{2BW}, \tag{2.7}$$

where BW is the bandwidth of the signal.

2.1.3 Velocity Measurements

In an FMCW radar, velocity measurements are achieved by analyzing the phase difference between two chirps spaced by a time interval T_c. Reflected signals from the same range bin will have phase differences due to the relative velocity of the target. The phase difference $\Delta\phi$ is given by:

$$\Delta\phi = \frac{4\pi v T_c}{\lambda}, \tag{2.8}$$

where v is the relative velocity, λ is the wavelength of the radar signal, T_c is the chirp repetition interval. From this, the velocity v can be expressed as:

$$v = \frac{\lambda\Delta\phi}{4\pi T_c}. \tag{2.9}$$

The maximum velocity measurable by the radar is determined by the phase ambiguity limit, beyond which $\Delta\phi$ exceeds π. This limit is given by:

$$v_{\max} = \frac{\lambda}{4T_c}. \tag{2.10}$$

2.1.4 Angle Measurement

In an FMCW radar with an antenna array, the received signal from a target at the mth antenna element can be represented as:

$$x(t_r, t) = h \exp\left[j\left(2\pi f_b t_r + \frac{2vt_r}{\lambda} + \psi\right)\right] + e(t_r, t), \tag{2.11}$$

where t_r and t refer to different time scales. t_r is the fast-time scale of a single chirp, and t is the slow-time scale across chirps, h is the channel gain, f_b is the beat frequency, v is the velocity of the target, λ is the wavelength of the radar signal, ψ is the phase shift due to the angle of arrival (AoA), $e(t_r, t)$ is additive noise. The signals received by the entire antenna array can be stacked into a column vector. Denoting this vector as $x(t_r, t)$, it can be written as:

$$x(t_r, t) = a(\theta)y(t_r, t) + e(t_r, t), \tag{2.12}$$

where $a(\theta)$ is the steering vector corresponding to the angle of arrival θ, $y(t_r, t)$ is the beat signal, $e(t_r, t)$ is the noise vector. The beat signal $y(t_r, t)$ is expressed as:

$$y(t_r, t) = g \exp\left(j\psi\right), \tag{2.13}$$

where g is the amplitude gain. To estimate the angle of arrival (AoA), a common approach is to calculate the power spectrum of the received signal using the Capon filter:

$$\Phi(\theta) = \frac{1}{a(\theta)^H R^{-1} a(\theta)}, \tag{2.14}$$

where R is the covariance matrix of the signal vector $x(t_r, t)$, H denotes the Hermitian transpose. Note that the covariance matrix R is invertible, as shown. This ensures the validity of the angle estimation process.

2.2 Multi-Modal Object Detection

Object detection is a major area in the literature of the perception, where the existing works predominantly concentrates on methodologies employing LiDAR [36, 93, 63] and fusion of LiDAR-camera [58, 13]. PointPillars, introduced by Lang et al. [36], focuses on encoding point clouds into a format that allows for efficient 2D convolutional neural network (CNN) processing. The method transforms point clouds into a pseudo-image representation, enabling the use of standard 2D CNNs for feature extraction. This transformation reduces computational complexity and improves detection speed, making PointPillars suitable for real-time applications. The authors demonstrated the efficacy of this approach on the KITTI dataset [23], showing competitive performance against other state-of-the-art methods. PivotNet, proposed by Ding et al. [17], offers a novel vectorized pivot learning method for end-to-end high-definition (HD) map construction. This method leverages the spatial relationships within point clouds to enhance the accuracy of HD maps, which are crucial for autonomous driving systems. By learning pivot points and their vectorized representations, PivotNet can efficiently process large-scale point clouds and generate detailed HD maps. The experimental results on the nuScenes dataset [10] highlight the superiority of PivotNet in terms of both accuracy and computational efficiency. Another notable contribution is PointRCNN by Shi et al. [63], which introduces a two-stage framework for 3D object detection from point clouds. The first stage generates 3D object proposals directly from raw point clouds using a region proposal network (RPN). The second stage refines these proposals through a novel point cloud pooling strategy, leading to precise object localization and classification. PointRCNN achieves high accuracy on the KITTI [23] benchmark, demonstrating the effectiveness of end-to-end point cloud processing for object detection. Frustum PointNets, developed by Qi et al. [58], take a different approach by integrating RGB-D data for 3D object detection. The method first extracts 2D region proposals from RGB images and then extends these regions into 3D frustums. Within each frustum, a

PointNet-based network processes the enclosed point cloud to detect and classify objects. This integration of 2D and 3D data helps in leveraging the strengths of both modalities, resulting in robust object detection. The experiments conducted on the SUN RGB-D and KITTI datasets show that Frustum PointNets outperform several existing methods, particularly in handling occlusions and partial observations. Finally, the MLOD method by Deng and Czarnecki [15] focuses on multi-view 3D object detection through robust feature fusion. This approach combines features from multiple views of the same scene to improve detection accuracy. The fusion of different perspectives helps in mitigating issues such as occlusion and varying object appearances. MLOD utilizes a deep learning framework to integrate these features seamlessly, demonstrating enhanced performance on challenging benchmarks like the KITTI dataset [23].

One significant challenge in developing a reliable perception module is maintaining performance during inclement weather. Severe environmental conditions can degrade sensor resolution, diminishing their effectiveness in accurately detecting objects. In such scenarios, the perception system may find it difficult to identify obstacles or other traffic participants, thereby increasing the risk of accidents. Camera sensors, despite being cost-effective, are particularly susceptible to issues like glare and reflections, which can obscure objects in the captured images [43]. Similarly, LiDAR sensors face challenges due to environmental impacts, heavy data processing requirements, and limited detection range [40][82]. To mitigate these individual sensor limitations, sensor fusion techniques are employed to enhance the reliability of perception performance. Specifically, the fusion of point clouds and images is extensively studied [57]. These studies utilize various sensing modalities for detecting cars [54][25], pedestrians [81][15], and cyclists [84]. However, these fusion-based approaches have certain shortcomings, such as dependency on accurate data calibration and association. Additionally, raw data often contains a significant amount of redundant information, posing challenges for real-time processing [57].

Notably, there is a few of research specifically dedicated to object detection via radar technology. One plausible explanation for this research gap is the absence of a reliable dataset incorporating radar data until the advent of the 2019 NuScene dataset [10]. Within the limited scope of radar-based object detection, the majority of studies have been inclined towards leveraging radar-camera fusion approaches to accomplish the task. One

of the pioneering works in this area is by Nobis et al. (2019) [50], who introduced the CameraRadarFusion Net (CRF-Net). This deep learning architecture fuses camera and sparse radar data to enhance 2D object detection. The network uses a training method called BlackIn, derived from BlackOut [27], aiming to teach the network the information value of the sparse radar independent of rich camera representation. Another significant contribution is the CenterFusion approach by Nabati and Qi [48]. Unlike CRF-Net, CenterFusion uses a middle-fusion technique that employs a center point detection network to detect objects in images and associates them with radar detections using a novel depth-based frustum association. The radar-based feature maps are then used to regress to object properties like depth, rotation, and velocity, providing a more comprehensive object detection framework. However, these methods are compromised when visual conditions are poor, i.e., during rain, snow, or dense fog, where camera visibility is significantly reduced. For a perception system to be deemed robust, it must maintain consistent performance even in such challenging environments. Therefore, some of the literature focused on the radar-only methods.

Expanding the radar data dimensions, Palffy et al. (2022) [53] applied PointPillars [37] to 4D radar data for multi-class road user detection. Their research highlighted the significance of radar Doppler and cross-section features. Additionally, they introduced the VoD automotive dataset, a comprehensive resource containing synchronized and calibrated data from various sensors. Wang et al. [89] introduced RODNet, a deep learning model for radar object detection that is cross-supervised by a camera-radar fusion algorithm. The network takes radio frequency (RF) images as input and predicts the likelihood of objects in the radar field of view. Zhang et al. (2021) [89] proposed RADDet, a deep learning model that focuses on Range-Azimuth-Doppler tensors for dynamic road user detection. Their work also introduces a novel radar dataset and an instance-wise auto-annotation method. Another radar-only approach by Zhang et al. (2019) [90] employs a deep convolutional neural network for object detection and 3D estimation using an FMCW radar. The paper also proposes a normalization method essential for radar signal processing, offering a comprehensive solution for radar-only object detection. Lastly, Nobis et al. (2021) [51] developed a low-level sensor fusion network for 3D object detection, fusing lidar, camera, and radar data. Their work shows that radar sensor fusion is especially beneficial

in inclement conditions, such as fog or rain, thereby enhancing the robustness of object detection systems. Much of the existing research relies on deep learning techniques that necessitate expensive GPUs, and often require transfer learning to adapt to changes in the environment or sensors. Methods executable on CPUs tend to exhibit reduced accuracy. Yet, with the advent of open-source platforms that provide widespread access to the HD maps, there remains a gap for a method that leverages this data without depending on deep neural networks.

Few studies have been conducted on utilizing HD maps for object detection. Among them, lanefusion [22] showed that by using lane direction information it can improve the object detection accuracy both in orientation and position. Other studies [20, 86] utilized semantic features of the map, e.g. derivable space, to enhance the overall accuracy of object detection. The dynamic nature of traffic environments and the presence of diverse interacting objects frequently present an Out of Distribution (OOD) challenge to perception systems. In real-world driving scenarios, the system may encounter small animals, pedestrians in unconventional attire, or specialized construction vehicles that were not part of the training dataset [73]. To tackle the OOD challenge, robust learning techniques have been introduced to enhance the resilience of neural networks to distribution shifts [71]. Some studies concentrate on the real-time detection of OOD instances, facilitating a safe transition to manual driving when necessary [21]. Nevertheless, validating the robustness of these systems against the varied distributions encountered in real-world driving remains a formidable challenge. Recently, researchers have investigated methods to circumvent the OOD issue by employing more effective representations of perception results. Techniques such as occupancy grids or drivable space have been explored to reduce the impact of unknown classes on the system's performance. These approaches aim to create a more comprehensive understanding of the environment, thereby improving the overall safety and reliability of autonomous driving systems.

The object detection is a critical component of AV navigation. However, it comes with some challenges that hinders its seamless integration in the downstream of the navigation stack e.g., path planning and controller; One major challenge is the discrete nature of information provided by object detection algorithms, which produce snapshots of the environment. When these snapshots are juxtaposed, they often reveal inconsistencies in

detected objects, leading to a phenomenon known as flickering. Where the path planner module, the downstream of perception, prefers a continous consistent information, creating a gap between perception and the rest of the navigation module.To bridge this gap, the concept of a cost-map emerges as a pivotal solution. The role of cost-map in navigation is crucial, serving as a dynamic representation that guides obstacle avoidance, path planning, and informed decision-making for AV [9, 2, 94, 61]. By assigning varying costs to different spatial regions, the cost-map provides real-time insights into the environment's complexity. This enables AV to navigate safely and efficiently, adapting to changing surroundings while ensuring effective interaction and trajectory generation, making the cost-map an indispensable tool for enhancing navigation capabilities. The presence of inherent uncertainty in sensor measurements [75, 33] necessitates the development of probabilistic cost-map [47, 46]. The probabilistic cost-map offers a valuable approach to retaining a comprehensive representation of the uncertainty inherent in the vehicle's perception of the surrounding environment.

2.3 Object Tracking and Prediction

Most of the literature discussing tracking, attempts at making the prediction and tracking into a single module [64]. They usually show that the velocity estimation that comes from tracking is mostly used for predicting the behaviour of the other objects and therefore improving the safety [42]. Over the past decade, researchers have devoted significant efforts to track and predict the behaviors of other objects and traffic participants. They could be categorized into three major approaches: physics-based prediction, pattern-based prediction, and planning-based prediction [29]. Physics-based prediction entails short-term forecasts of traffic participants' movements, assuming minimal changes in their behavior over a brief period. Predictions are derived from dynamic or kinematic models of the traffic participants' motion. For instance, Brannstrom et al. utilized a linear bicycle model to assess risk by generating potential vehicle paths [6], while Xie et al. employed a vehicle kinematic model to predict candidate paths [83].

These methods usually rely on a typical tracking model that might mislead if the objects

move close to one another or a flickering happen in the object detection pipeline. Moreover, they are able to deal with short-term prediction and consider possible paths for objects. However, in long-term prediction problems, the behavior of an object might change, and using pure physic-based method to predict path is insufficient. Therefore, for long-term prediction, pattern-based prediction is proposed to predict long-term actions of a traffic participant. Mo et al. applied GNN-RNN based Encoder-Decoder network for prediction of vehicles on highway interacting with other vehicles [45]. Messaoud et al. also applied a LSTM-based deep network to train the prediction model [44]. These methods are every effective in common driving scenarios, however, the deep network requires a comprehensive training data, and in real-life application, the corner cases' data is hard to obtain. Which make this kind of method less reliable. Another trendy method overcomes this limitation by using a planning-based prediction, which tries to reason out the most likely goal and corresponding policy of the traffic participant that it is predicting. Shu et al. applied a planning-based behavioral model for the surrounding traffic participants, and takes several features to identify the possible future intention of the surrounding traffic participants [65]. [67] set the utility functions for the surrounding traffic participants, and tracks the intention of the surrounding traffic participants considering uncertainties. The algorithm was tested in an intersection scenario in urban scenarios [66]. Yet, these methods require large resource, and reliablity while also are not as agile for the urban use.

2.4 Localization

The AV localization usually happens with Simultaneous Localization and Mapping (SLAM) which provides a holistic solution for vehicle localization by simultaneously estimating both the position and the map [7]. In the context of autonomous driving, instead of tackling the full SLAM problem—which involves calculating the joint posterior over all poses and the map using comprehensive sensor data—online SLAM is often preferred. Online SLAM is centered on estimating the vehicle's current motion state using the most recent sensor inputs. Various filter-based methods have been explored for this purpose, including the Extended Kalman Filter [72], Information Filter [16], and Particle Filter [79].

SLAM-based localization encounters several challenges, including divergence [62], drift issues [30], and poor performance in adverse weather conditions [92]. To address drift, fault detection techniques can be utilized to take advantage of sensor redundancy and ensure the coherence of multiple sensor sources, identifying drift as a failure [14]. Additionally, fusion strategies are often employed to counteract drift in online SLAM solutions [80]. For instance, [80] describes a method where LiDAR and inertial data, along with limited landmarks, are used to achieve robust localization for mining service vehicles experiencing GNSS dropouts. In [91], multi-modal information such as vision, point clouds, inertial data, and raw GNSS signals are fused based on information coherency to deliver reliable localization results in environments with sparse features. Ultimately, effective drift correction necessitates regular consideration of constraints related to the vehicle's position concerning a known, local, or absolute reference.

Utilizing HD (High Definition) maps [17] for localizing an AV within a known environment is a beneficial strategy to ensure the safe navigation of the AV. HD maps are meticulously detailed representations of the environment, containing precise information about lane markings, road geometries, traffic signs, and other relevant features. By comparing real-time sensor data from the vehicle to the information stored in the HD map, the AV can accurately determine its position and orientation within the environment. In [78, 56, 26], novel semantic localization algorithms that exploit semantic features extracted from images and features in HD map are proposed. However, relying solely on vision makes this approach susceptible to accuracy issues in adverse weather conditions or low-light environments. In [39], a multiscale localization method for AVs based on the semantic keyframes of carpark signs, as a known environment, is utilized, using only a low-cost monocular camera to address the problem due to the unavailability of GNSS signals indoors.

Both online SLAM and the utilization of HD map aim to ensure reliable localization under various conditions such as divergence, drift, and adverse weather conditions. These two localization techniques usually rely on the fusion of different sensing modalities' capabilities. Vehicle on-board sensors employed in localization include GPS, IMUs, cameras, radar, LiDAR, and ultrasonic sensors [35]. Despite the cost-effectiveness of fusion techniques among GPS, IMU, and cameras, particularly prevalent in vision-based SLAM,

visual odometry and lane detection, these approaches often fail to achieve the desired level of accuracy [55] [70]. Radar fusion techniques show promise for short-range SLAM but are hindered by their susceptibility to environmental uncertainties [82]. In contrast, LiDAR fusion techniques have emerged as a standout solution, offering superior accuracy and robustness in SLAM tasks [24] [38]. However, they are susceptible to adverse weather conditions like snow or fog, and their performance relies on consistent GPS inputs. These inputs may be impacted by changes in the landscape or communication disruptions. Considering these challenges, we propose an innovative approach to vehicle localization: the integration of LiDAR and radar data, augmented by odometry calculations. By combining the strengths of LiDAR and radar while compensating for their individual limitations, this integrated approach aims to achieve both high accuracy and robustness in localization tasks. It can deliver reliable performance even in challenging environments, ensuring the safety and efficiency of autonomous vehicles in real-world scenarios.

2.5 Summary

In summary, there appears to be a gap in developing a unified perception-localization module, despite the interconnection and complementary nature of these two systems. In object detection and tracking, few methods effectively address edge cases and reliability; most focus on creating deep neural networks that outperform others on datasets primarily featuring normal weather conditions and regular objects. Additionally, there is a need for perception modules that leverage HD map data to leverage familiarity with the environment while also prioritizing reliability. Regarding tracking, filter-based methods have not yet reached maturity, and the literature predominantly focuses on temporal feature detection networks for handling velocity and predicting the actions of other actors. Current approaches heavily emphasize the advantages of neural networks, utilizing them extensively in various applications. In this study, the approach to perception and localization is revisited while consistently considering the reliability of the developed solutions and leveraging the mathematical models whenever possible.

Chapter 3

Radar-HD Map Perception System

Radar point cloud is occupied with a variety of noises, which would affect the downstream fusion immensely. This chapter discusses the necessary operation with the help of HD map to clean the radar data from noise. To show the effectiveness of the approach, the result of the radar-HD map fusion is used to generate a cost-map. To start the approach, first the problem is formulated, and then the solution is broken into the following sections; point cloud manipulation, velocity compensation, clustering, feature extraction, HD map fusion, and finally cost-map generation.

3.1 Problem Formulation

The objective of this chapter is to clean the radar point cloud using the statistical information of each class in the HD map. This is represented through a probabilistic fusion method that can generalize itself to any other environment, as long as the HD map information is provided. For a general scenario shown in Figure 3.2, each point p_r^i, the blue dots, in the radar point cloud is considered to have the following features,

$$p_r^i = \{x_{ri}, y_{ri}, v_{ri}, \mathrm{RCS}_i\} \tag{3.1}$$

where x_{ri} and y_{ri} are 2D position in the radar frame, and v_{ri} is the relative radial velocity [48], and RCS_i is the radar cross-section, a measure of the reflectivity [32]. These features

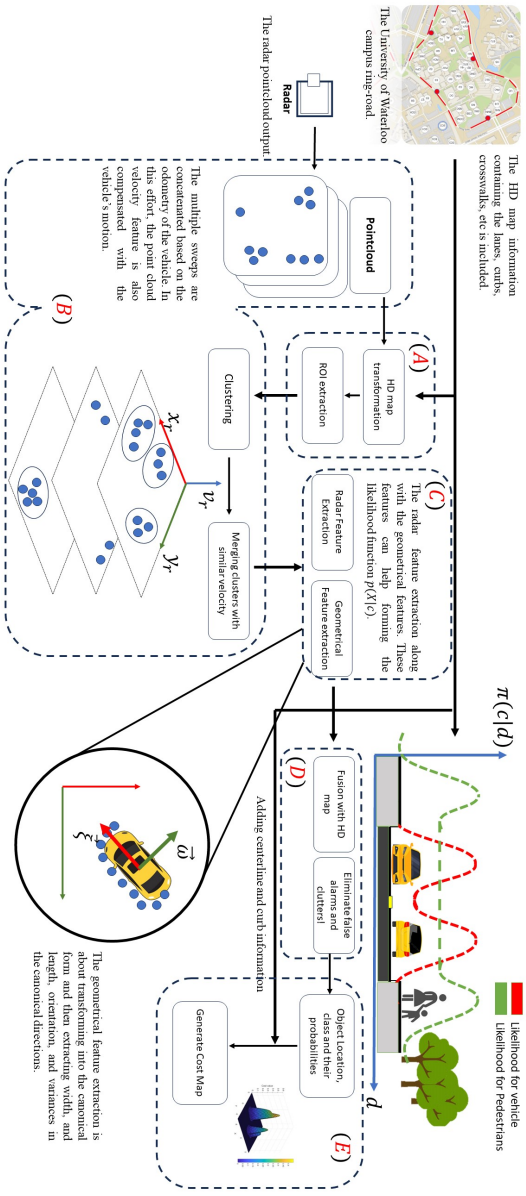

Figure 3.1: A schematic of the radar point cloud manipulation and cleansing, and the cost-map generation algorithm has been brought here. First, a sequence of the radar point clouds are consistently concatenated on one another, then with the aid of the HD map the point cloud is focused on the road, removed from false alarms and clutters, and added HD map features for the final classifier. Finally the output is used to form the cost-map.

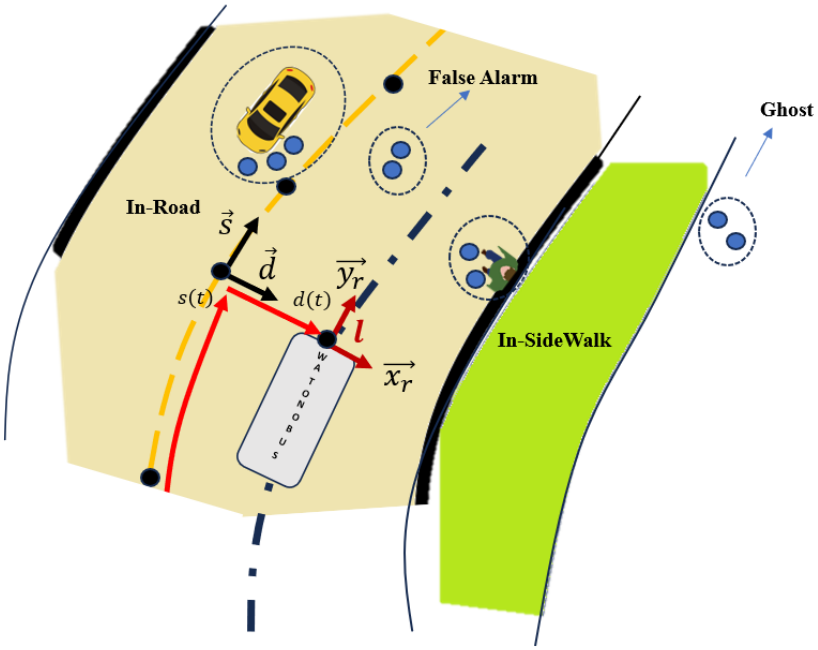

Figure 3.2: Radar point cloud in the frenet frame, the black dots are the s-points on the HD map, and the blue arrows are for the lateral distance in (s, d) frame. The red arrow represents the magnitude of the s coordinate. The blue dots are the radar point clouds. The coordinate attached to the bus is the radar frame.

are common in automotive radars, as they usually provide an array of detections, there are some other radars that provide the radar image, however, focus in this study is the automotive radars with point cloud as their output. The initial aim is to remove the part of the point cloud which is irrelevant, e.g., the false alarms, and clutters, and output the clusters with an associated probability π and class c which is calculated in fusion with the extracted features and the HD map. In this regard, for a sample point l on the HD map the position vector in the frenet frame [52] is expressed as,

$$\vec{X}_l(s(t), d(t)) = \vec{s}(s(t)) + d(t)\vec{d}_r(s(t)), \tag{3.2}$$

where $s(t)$ denotes the covered arc length of the center line, $d(t)$ is the lateral distance, and $\vec{s}$, $\vec{d}$ are the tangential and normal vectors of the centerline. Using Eq. (3.2) extends

the features for each point p_r^i with,

$$\begin{aligned} p_x^i &= \{s_{ri}, d_{ri}, \delta_{Ri}, \delta_{Li}\} \\ \delta_{Ri} &= d_{Ri} - d_{ri} \\ \delta_{Li} &= d_{Li} - d_{ri} \end{aligned} \tag{3.3}$$

where d_{Ri}, and d_{Li} are the d coordinates of the right curb and left curb closest to the point p_r^i at (s_{ri}, d_{ri}) in the HD map respectively. The new radar point cloud representation for each point would be,

$$p^i = \{p_{ri}, p_{xi}\} \tag{3.4}$$

Moreover, at each arc length s, the d_A and d_B which are the lateral positions of the outer left and right sidewalks, along with the odometry of the vehicle,

$$\mu_e = \{s_e, d_e, x_e, y_e, \psi_e, \dot{\psi}_e, v_e\}, \tag{3.5}$$

where x_e, y_e, ψ_e, v_e and $\dot{\psi}_e$ are the 2D position of the ego vehicle in The Universal Transverse Mercator (UTM) frame [69], yaw angle, the linear velocity and yaw rate, respectively. The final aim is to remove the irrelevant detections, and generate a probabilistic cost-map, $U_{s,d}$, with the fusion of HD map semantics. The proposed method is applicable to any radar point cloud and is suitable for all-weather conditions.

3.2 Radar HD Map Fusion

The overview of post-processing for radar detection is shown in Figure 3.1 . Although rich in features, the radar point cloud contains false alarms and clutter. In part (A) of the method, the point cloud outside the vehicle's region of interest (ROI), including the road and sidewalks, is excluded. This step provides a rough elimination of excessive radar information and some false alarms and clutter. In part (B), the ROI point cloud is passed to the DBSCAN clustering algorithm [18], which focuses on spatial features while considering a lower weight for compensated velocities. This clustering algorithm removes some sparse false alarms, but some dense forms of false alarms or clutter may remain in the clusters. To address this issue, in part (C), a series of features are extracted from each cluster to

form likelihood functions for different object classes. In the next step, part (D), HD map information is fused with the cluster features to classify the detected cluster. The HD map information incorporates environmental semantics into the object detection results. Finally, in part (E), a number of clusters are generated, each assigned a probability and class, which are then integrated into the probabilistic cost-map. The next subsections go deeper into the details of each part of the method.

3.2.1 ROI and Velocity Compensation for the Point Cloud

The range of radar can be adjusted, yet automotive radars operating above 60 GHz can detect objects up to 400 meters away. While this capability is advantageous, it can also introduce challenges such as multiple reflections (multi-paths) across the radar's coverage area due to the strong waveform. However, focusing the classifier on the Region of Interest (ROI) using HD map information allows the vehicle to prioritize detection of the road and its crosswalks, thereby mitigating potential interference from non-critical objects. The HD map information helps to have three possible geo-labels for each point in the radar point cloud as p_l^i,

$$p_l^i = \{\text{in-Road},\ \text{in-SideWalks},\ \text{Out}\}. \tag{3.6}$$

From Eq. (3.4), point *ith* in the radar point cloud is defined as,

$$p^i = \{x_i, y_i, v_{ri}, \text{RCS}_i, d_i, \delta_{Ri}, \delta_{Li}\}, \tag{3.7}$$

the v_{ri}, in Eq. (3.7), reported by the radar, is the relative radial velocity with respect to the ego vehicle frame. Considering the ego vehicle motion information, the velocity can be compensated as in Figure 3.3,

$$r_i = \sqrt{x_{ri}^2 + y_{ri}^2} \tag{3.8}$$

$$\theta_i = \arctan 2(x_{ri}, y_{ri}) \tag{3.9}$$

$$v_{ri_{\text{comp}}} = v_{ri} - v_e \cos\theta_i, \tag{3.10}$$

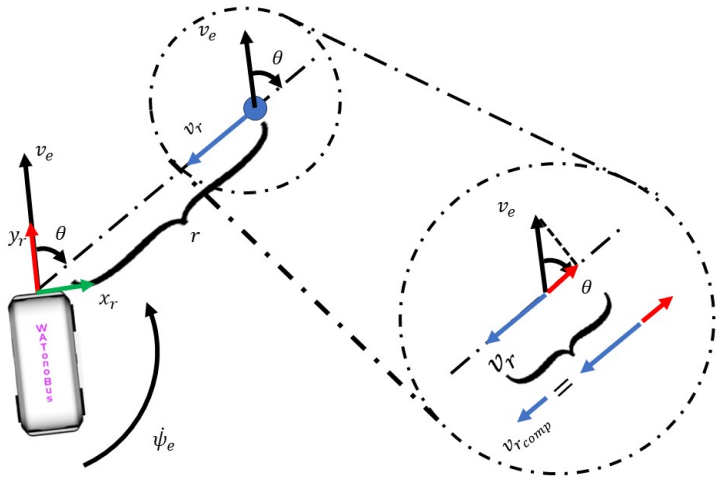

Figure 3.3: Considering the ego vehicle traveling with linear velocity v_e and turning with yaw rate of $\dot{\psi}_e$, the radar relative radial velocity can be compensated by projecting the ego velocity into the object's radial unit vector and then adding up with the relative radial velocity.

where r_i is the range of the point and θ_i is the azimuth angle. Subsequently, point ith is categorized into three labels utilizing d_i coordinate information,

$$p_l^i = \begin{cases} \text{in-Road}, & \delta_L \geq 0 \ \& \ \delta_R \leq 0 \\ \text{in-SideWalks}, & 0 \leq \delta_L < d_L - d_A \text{ or} \\ & d_B - d_R < \delta_R \leq 0 \\ \text{Out}, & \text{otherwise}, \end{cases} \tag{3.11}$$

where d_B is the limit of the sidewalk on the right, and d_A is the limit on the left. For driving, the ROI that is important to the vehicle defined to be,

$$\text{ROI} : \forall p^i \ \textbf{that} \ p_l^i \in \{\text{in-Road}, \text{in-SideWalk}\}. \tag{3.12}$$

Updating Eq. (3.4) the final representation of radar point cloud to be,

$$p^i = \{p_r^i, p_x^i, p_l^i\}. \tag{3.13}$$

This step focuses the algorithm on prioritizing the road, enhancing its ability to distinguish between noise and actual detections. It also enriches the point cloud by incorporating additional features, creating a more elaborate space for the classifier to make precise decisions. Furthermore, empirical observations highlight strong correlations between different classes and their specific spatial locations on the HD map. This correlation aids in refining the algorithm's understanding of the environment, improving its capability to accurately interpret and respond to real-world scenarios, such as identifying road features and differentiating them from false alarms and clutters.

3.2.2 Multi-Frame Two Staged Consistent Clustering

A single frame of radar point cloud often suffers from limitations in point density, which can compromise the system's ability to accurately identify and interpret surrounding objects. To address this issue, radar points are accumulated across multiple frames. It is crucial to note that this frame accumulation is not carried out in a naive manner. One of the primary challenges in accumulating radar points across multiple frames is accounting for the vehicle's motion. If the vehicle is in motion, the radar points from different frames would represent different positions in the world coordinates, and a simple overlay of these points would yield inaccurate and misleading results. To overcome this challenge, the vehicle's motion obtained from Eq. (3.5) is incorporated, a transformation is applied to each set of radar points, mapping them from their original ego vehicle frame to the map coordinate system. This transformation ensures that the radar points from different frames are correctly aligned with respect to the vehicle's position and orientation. As a result, the accumulated point cloud presents a more comprehensive and accurate representation of the surrounding environment. If the point cloud Cartesian positions, $[x, y]$, are placed in the vector P, at the time t, then considering the vehicle odometery and localiztion module, there is a transformation matrix T_t that can transform the point cloud into the map frame. The important fact about the map frame should be its constant position and orientation, facilitating the consistent concatenation of the multi-frame point cloud. The map frame orientation is set on North and East coordinates, and its origin must be off seated at somewhere around the operation area of the AV. In the case of this thesis, the

origin is seated at the center of the E3 intersection of the University of Waterloo campus ringroad.

$$P_t^m = T_t.P_t, \tag{3.14}$$

and for the frame k times ago the point cloud is,

$$P_{t-k}^m = T_{t-k}.P_{t-k}, \tag{3.15}$$

then if the multi-frame considers 1 frame ago, the concatenated point cloud is,

$$P_t^{m'} = [P_t^m, P_{t-1}^m] \tag{3.16}$$

The point cloud created in Eq. 3.16, is a more dense point cloud, however, since the environment is dynamic, it could be misleading, e.g. the length or width of a moving vehicle could show larger than the reality. To address this issue, an optimal number of frames is considered in the multi-frame concatenation process. This depends on the relative velocity between the ego-vehicle and the other actors. A timer is attached to each point of the point cloud depending on the relative radial velocity, such that the oldest frame could at most travel up to 0.75 of vehicle length, 3.5. Then the timer for each point,

$$\delta t_i = \mathbf{Sat}(\frac{3.5}{v_{ri}}), \tag{3.17}$$

that the δt_i is the timer, and is saturated by 6 frames from up, and 2 from below, the v_{ri} is the relative velocity.

The resulting point cloud should undergo a clustering process as the objects are often present in the dense parts of the point cloud. DBSCAN, a Density-Based Spatial Clustering of Applications with Noise [18], is employed to group together points based on their density in the feature space, forming clusters of varying shapes and sizes. Unlike traditional clustering methods, DBSCAN can efficiently handle noisy data and discover clusters without requiring the number of clusters as an input, making it an ideal candidate for radar point cloud clustering. It defines core points as those with a minimum number of neighboring points within a specified radius, while non-core points are considered part of a cluster if they are reachable from a core point.

The DBSCAN algorithm for radar clustering is usually formulated in grid-based approaches and with (x, y) coordinates [31]. This approach only considers spatial features and in case there are multiple objects close to one another, it will cluster them into one object. However, the radar point cloud comes with the Doppler velocity, v_{ri}, which can be used to make the feature space more separable. The sparsity of the radar points around large objects causes to result in multiple clusters for the same object, therefore in our clustering, there are two stages: 1- Initial clustering, and 2- Merging. From Eq. (3.13), the first stage of the clustering takes $(x_i, y_i, \Omega v_{ri})_{UTM}$ as the features, where Ω is the weight factor for velocity feature, valued between 0 and 1, so the choice for ϵ [18] is mainly based on the spatial position. Then, each cluster will be merged with its neighbor of a larger distance δ_ϵ, only if their mean velocity difference is less than a threshold δ_v. After applying the two stages of DBSCAN to the radar point cloud in the ROI of the ego-vehicle the point cloud is represented as a set of clusters C_j,

$$S = \{C_0, C_1, ..., C_n, O\}, \tag{3.18}$$

where O is the outliers defined as,

$$O = \{p^i \in O \,|\, p^i \notin C_0, C_1, ..., C_n\}. \tag{3.19}$$

The Eq. (3.18) needs more post-processing to be included in any downstream operation including fusion or cost-map; in the radar point cloud, the presence of a dense cluster of points does not always correspond to a genuine object, i.e. it can be the result of phenomena like false alarms, or other forms of clutters. Consequently, a more in-depth analysis, extracting additional features from each cluster, is necessary to ascertain its true nature and determine its validity as an object.

3.2.3 Geometrical and Radar Feature Extraction

Although the points on each cluster could be sparse, the geometrical features stay consistent and it can hint about the object type, e.g. the width, and length of the cluster can be a good indicator to at least show the difference between a pedestrian and a car. However, the

geometrical feature can change with orientation. To make a standard representation for each cluster, each cluster is transformed into its canonical form; the canonical coordinate system for a cluster denotes that (1) the origin is located at the center of the cluster $(\bar{x}_i, \bar{y}_i)$ (2) the $\vec{\xi}$ direction is across the length and $\vec{\omega}$ is across the width of the cluster. Then the following features will be extracted,

$$\Phi_{C_j} = \{\omega_j, \xi_j, \alpha_j, \lambda_{xj}^2, \lambda_{yj}^2\}, \tag{3.20}$$

where ω_j and ξ_j stands for width and length of the cluster, α_j is the principal orientation, and λ_{xj}^2 and λ_{yj}^2 are variances along the principal directions. Moreover, for each cluster, the following set of radar features can also be drawn,

$$\begin{aligned} \gamma_{C_j} = \{&\bar{v}_{rj}, \bar{\mathrm{RCS}}_j, \|C_j\|, \\ &\mathrm{RCS}_{\mathrm{minj}}, \mathrm{RCS}_{\mathrm{maxj}}, v_{\mathrm{minj}}, v_{\mathrm{maxj}}, \\ &\eta_1\sqrt{r_j}(\|C_j\|), \eta_2\frac{\mathrm{RCS}_{maxj}}{\omega_j\xi_j}, \\ &\eta_3\frac{\mathrm{RCS}_{\max}}{r_j^2}, \eta_4\frac{\|C_j\|}{r_j^2}\} \end{aligned} \tag{3.21}$$

where r, $\bar{v}_{\mathrm{r}}$, $\|C_j\|$ and $\bar{\mathrm{RCS}}$ are the range of the centroid, average radial velocity, number of points, and average RCS in the cluster, respectively. The η_i acts as a normalizer to scale the feature between 0-1. Making the final features to be considered for classification,

$$\chi_{C_j} = \{\gamma_{C_j}, \Phi_{C_j}\}. \tag{3.22}$$

The $\sqrt{r}(\|C_j\|)$ is an indicator for the density of the cluster regardless of its range; the radar point cloud on a specific object varies in density with range, where the clusters located in the shorter range have higher number of points and those in the farther vice versa. According to our observation for each class, the value for $\sqrt{r}(\|C_j)\|$ stays consistent. The $\mathrm{RCS}_{max}/\omega\xi$ is the density of the RCS inside the cluster which measures the ratio of the cluster's size in the radar's view to the actual measured size. One of the observations that helped us in choosing the features is the behavior of false alarms; they usually show negligible $\bar{v}_r$, and $\bar{\mathrm{RCS}}$, Furthermore, urban and semi-urban environments introduce challenges

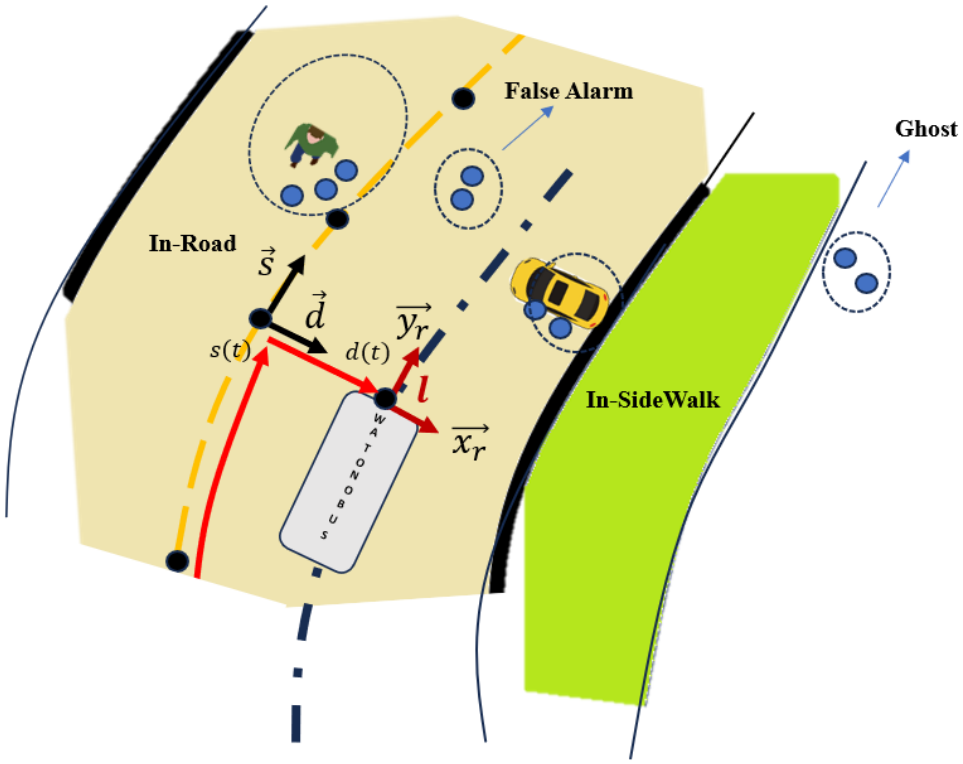

Figure 3.4: Same scenario as the Figure 3.2, however, here the location of the pedestrian with the vehicle are switched to emphasize the unlikeliness of this scenario to happen.

in the form of metal objects, e.g., poles and trash cans. These objects, despite their small size, exhibit a high RCS value. However, they are also characterized by a small cluster size when detected, the $\text{RCS}_{max}/\omega\xi$ feature is large for these objects compared to pedestrians and cars. The clusters so far generated by DBSCAN represent potential objects detected by the radar system, which may include a mix of true objects, noise, ghosts, and clutter that has eluded the radar's Constant False Alarm Rate (CFAR) filtering [60]. To refine this assortment, one can integrate the HD map data into the analysis. The following section dives deeper into this argument.

3.2.4 Fusion with HD Map Information

The features extracted from each cluster play a crucial role in determining the class of the cluster . Given our knowledge of the HD map, such as the distinction between sidewalks and roads, statistics can be leveraged related to the typical presence of classes in these locations. For instance, pedestrians are more commonly found on sidewalks, making cars less likely to be spotted there. Similarly, it's highly improbable to observe a car driving on a curb or encountering other analogous scenarios, an instance is presented in Figure 3.4. It shows the same scenario as in Figure 3.2, though it emphasizes the correlation between location in the HD map and the object class. Using the Bayes theorem, one can find the probability π for class c for cluster C_j located at the d_i in the HD map (s, d) coordinate,

$$\pi(c|d_j, \chi_{C_j}) = \frac{\pi(c, d_j, \chi_{C_j})}{\pi(d_j)\pi(\chi_{C_j})}, \tag{3.23}$$

reordering the equation and considering that χ_{C_j} and d_j are independent variables,

$$\pi(c|d_j, \chi_{C_j}) = \frac{\pi(c|d_j)\pi(c|\chi_{C_j})\pi(d_j)\pi(\chi_{C_j})}{\pi(d_i)\pi(\chi_{C_j})}, \tag{3.24}$$

and then reordering again,

$$\pi(c|d_j, \chi_{C_j}) = \pi(c|\chi_{C_j})\pi(c|d_i) \tag{3.25}$$

where $\pi(c|\chi_{C_j})$ must be a probabilistic classifier, and the $\pi(c|d_j)$ is the likelihood to have an object of type c at the specific lateral lane location d_j. In this work, it is formed as a mixture of Gaussian defined similar to what is shown in Figure 3.1. If $\pi(c|\chi_{C_j})$ represents a generic classifier, an approximation $\pi^*(c|\chi_{C_j})$ is employed instead. However, using this approximation disrupts the scale of the $\pi(c|d_j, \chi_{C_j})$ to exceed from 1. Therefore, a normalization factor κ is introduced to rescale, ensuring they remain between 0 and 1,

$$\pi(c|d_j, \chi_{C_j}) = \kappa\ \pi^*(c|\chi_{C_j})\,\pi(c|d_j). \tag{3.26}$$

Making for each cluster C_j the following information to be used to create cost-map,

$$h_{C_j} = \{\omega_j, \xi_j, \bar{s}_j, \bar{d}_j, C_j, \pi_j\}. \tag{3.27}$$

Algorithm 1 Cost-Map creation

1- Object cost calculation:

for For each object i **do**

  for $s = s_{\min}{:}n_s{:}s_{\max}$ **do**

    for $d = d_{\min}{:}n_d{:}d_{\max}$ **do**

      $\sigma = f(\omega_j, \xi_j, c_j)$

$V^j_{s,d} = \pi_j \cdot \exp(-\frac{(s-\bar{s}_j)^2}{2\sigma^2} - \frac{(d-\bar{d}_j)^2}{2\sigma^2})$

    end for

  end for

end for

2- Cost-Map Update:

for $s = s_{\min}{:}n_s{:}s_{\max}$ **do**

  for $d = d_{\min}{:}n_d{:}d_{\max}$ **do**

    $U_{s,d}(t) = \sum_{j=1}^{m} V^j_{s,d} + U_{s,d}(t-1) - \mathcal{D}$

  end for

end for

m is the total number of detected objects with probabilities and classes.

3.2.5 Cost-Map Creation

The results from radar-HD map fusion can be used for the downstream fusion, however, there are some scenarios that the vision is entirely disturbed and therefore LiDAR and camera sensors are not much of use. To tackle these challenging scenarios, in this thesis, an scheme to create cost-map purely from radar data is explained and tested. As previously discussed, by combining radar detection data with the HD map, information regarding various objects is acquired, Eq. (3.27). Utilizing this information, a cost-map can be constructed based on the positions and detection probabilities of these objects within the drivable area around the ego, i.e., ($s_{\min}$, $s_{\max}$, $d_{\min}$, $d_{\max}$). This process also takes into account the grid resolution employed for cost-map calculations, i.e., n_s and n_d. To compute the cost associated with each individual detected cluster j, grid points are iterated within the specified drivable area. At each grid point (s, d),a cost value $V^j_{s,d}$ is calculated. This cost value is determined by considering the object's probability, π_j, the object class, c_j, object dimension, (ω_j, ξ_j), and applying a Gaussian-like function that accounts for the distance between the grid point and the object's location $(\bar{s}_j, \bar{d}_j)$. The outcome represents

how each object contributes to the cost-map at every grid point. Finally, the cost map $U_{s,d}(t)$ is updated by summing all the $V^j_{s,d}$ values for all objects, the computed cost map from the previous time step $U_{s,d}(t-1)$, and a factor denoted as $\mathcal{D}$. As time advances, $\mathcal{D}$ gradually reduces the likelihood associated with the presence of an object when there is no detection. The comprehensive procedure is outlined in Algorithm 1.

3.3 Summary

In this chapter, a novel probabilistic approach for radar classification and false alarm removal is presented. Initially, excess radar points outside the vehicle's region of interest (ROI) are removed with the help of the HD map. Outliers are then removed using DBSCAN, and clusters are formed from the dense part of the point cloud. Geometrical and novel radar features are extracted from each cluster and appended with HD map features. This approach leverages environmental familiarity to form a likelihood that aids classification. Likelihoods are formed using the HD map's statistical occurrence of each class and are fed into the probabilistic classifier. The probabilities of each class occurrence are then used to create a cost map, which acts as both a smoothing layer over the radar output and a continuous input for downstream path planning modules.

Chapter 4

Radar-LiDAR-Camera-HD map Perception and Localization System

Achieving a robust perception and localization system necessitates the inclusion of radar. In the previous chapter, an algorithm was presented for cleaning radar data, resulting in a cost-map and adjusted point cloud. This chapter concludes with a presentation of the method used to fuse the adjusted radar point cloud with other essential vehicle sensors: LiDAR, camera, and wheel speed encoders. The output of this method includes accurate object detection and tracking, as well as a full-state observer of the vehicle's motion states.

4.1 Perception

The first part of this chapter goes into the intricate workings of the perception module and explores the sophisticated processes involved in fusing data from various sensors. Sensor fusion is a pivotal aspect of modern perception systems, especially in applications such as autonomous vehicles, robotics, and advanced driver-assistance systems (ADAS). Effective sensor fusion enhances the system's ability to perceive and interpret the environment accurately, thereby improving decision-making and operational safety as s.

In the realm of sensor fusion, three primary methodologies are recognized in the liter-

ature: Early Fusion, Middle Fusion, and Late Fusion showing Figure 4.1. Each of these methods has distinct characteristics, advantages, and challenges. However, among them, Late Fusion stands out as particularly reliable, especially when considering the weaknesses inherent in Early and Middle Fusion.

Early Fusion, often referred to as raw data fusion, involves combining sensor data at the earliest possible stage, typically at the raw data level. This method integrates the data directly from different sensors before any significant processing or feature extraction occurs. For instance, in an autonomous vehicle, raw data from cameras, LiDAR, radar, and ultrasonic sensors could be fused together to create a comprehensive and cohesive representation of the environment. While Early Fusion leverages the full richness and resolution of the raw sensor data, it presents several critical challenges. The sheer volume of raw data from multiple sensors can be overwhelming, requiring substantial computational resources for processing and storage. This can lead to delays and inefficiencies in real-time applications. Moreover, the calibration and synchronization of different sensors are critical to ensure that the fused data is coherent and accurately represents the environment. Any discrepancies in timing, orientation, or calibration between sensors can degrade the quality of the fused data, potentially leading to erroneous interpretations and decisions.

Middle Fusion, or feature-level fusion, represents an intermediate approach where sensor data is processed individually to extract relevant features before being fused together. In this method, each sensor undergoes preliminary processing to identify and extract key features, such as edges, corners, or object boundaries, which are then combined to form a unified perception of the environment. This method balances the benefits of both Early and Late Fusion, but it also has notable weaknesses. By processing each sensor's data separately before fusion, Middle Fusion reduces the computational burden compared to Early Fusion. However, it still requires significant computational resources to extract and process features from each sensor's data. Designing effective feature extraction algorithms is a complex task, and determining the optimal way to combine these features to achieve accurate and robust perception can be challenging. Additionally, feature extraction may lead to the loss of important information contained in the raw data, potentially reducing the accuracy and reliability of the fused perception.

Late Fusion, also known as decision-level fusion, involves fusing data at the highest

level of abstraction. In this approach, each sensor's data is processed independently to generate individual interpretations or decisions about the environment. These decisions are then combined to form a final, consolidated perception. The primary advantage of Late Fusion is its simplicity and modularity. Since each sensor operates independently up to the decision-making stage, the system can be more easily scaled and adapted to incorporate new sensors or algorithms. This modularity also makes Late Fusion less sensitive to sensor calibration and synchronization issues, as the fusion occurs after the individual sensor data has been processed. By integrating decisions rather than raw data or features, Late Fusion minimizes the risk of errors arising from discrepancies in sensor data alignment or calibration. Moreover, Late Fusion is more robust to sensor failures and inconsistencies. If one sensor's data is compromised or unavailable, the system can still function effectively based on the decisions derived from other sensors. This redundancy enhances the reliability and resilience of the perception system, making it particularly suitable for safety-critical applications like autonomous driving.

In summary, while Early and Middle Fusion methods have their advantages, they also suffer from significant weaknesses related to computational demands, complexity, and potential data inaccuracies. Late Fusion, with its modularity, simplicity, and robustness, emerges as a more reliable approach for sensor data fusion in perception systems. By focusing on decision-level integration, Late Fusion ensures that the perception system remains accurate, efficient, and resilient, even in the face of challenging conditions and sensor discrepancies.

With regards to the points mentioned above, this thesis adopts a late fusion approach, where it leverages the unique strengths of each sensor before integration, optimizing the overall performance of the system. In the following sections, each of these processing and fusion steps will be detailed comprehensively.

In section 4.1.1, the LiDAR point cloud data undergoes several preprocessing steps to improve quality and relevance. Ground points are eliminated since they generally do not contribute information for detecting objects above the surface. In case of noise introduced by adverse weather conditions, such as rain or fog, it is filtered out through the adaptive DBSCAN method. Points outside the vehicle's Region of Interest (ROI) are discarded; the ROI, as defined in the previous chapter, includes sidewalks and roads where relevant

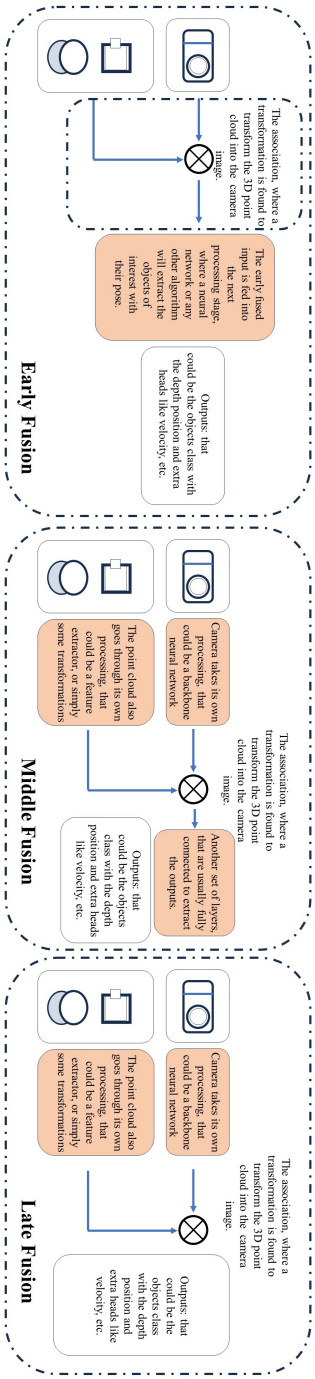

Figure 4.1: Overview of Early, Mid, and Late fusion.

objects are likely to be found. These steps ensure that the LiDAR data used for object detection is both accurate and pertinent to the areas of interest.

In section 4.1.2, the cleaned radar points are then fused with the processed LiDAR clusters using a KDTree algorithm. A K-dimensional tree (KDTree) is a data structure that efficiently organizes points in a multi-dimensional space, allowing for rapid nearest-neighbor searches. This is crucial for associating radar points with corresponding LiDAR clusters based on spatial proximity. The fusion of LiDAR and radar data enhances object detection by combining the spatial resolution of LiDAR—which provides precise spatial location and shape information of objects—with the velocity and RCS data from radar, which relates to the object's size and material properties. In this fusion process, LiDAR data primarily determines the location and origin of detected objects, while radar data supplements with additional attributes. This complementary integration enriches the feature set for each detected object, improving detection accuracy and robustness.

In section 4.1.3, camera images are processed using YOLO version 8, a real-time object detection algorithm known for its speed and accuracy. YOLO v8 generates bounding boxes and class labels for objects detected within the images. These bounding boxes are fused with the clusters derived from LiDAR and radar data to assign semantic class labels to the detected objects. The fusion is through frustum approach detailed in the section 4.1.4. Bounding boxes that do not correspond to any cluster are disregarded to ensure that only relevant detections are considered. This step reduces false positives and enhances the reliability of the detection system.

4.1.1 LiDAR Point Cloud Clustering

In the field of autonomous driving and advanced mapping, LiDAR technology plays a crucial role by providing highly detailed and accurate 3D representations of the environment. However, the enormous volume of data generated by each LiDAR scan, it can range from 20k to 60k points, presents a significant challenge. The manipulation of such large datasets consumes considerable memory and CPU resources. Additionally, many of these points are often irrelevant to the specific application, such as points representing trees and vegetation, which do not contribute valuable information and only add to the computational burden.

This high volume of data can quickly overwhelm the system's processing capabilities, leading to slower processing times and increased latency, which are critical issues in real-time applications like autonomous driving.

To address these challenges, a targeted approach is necessary. By leveraging high-definition (HD) map information, the focus is set on the Region of Interest (ROI) within the LiDAR scan, which primarily includes the road and areas where relevant objects like vehicles and pedestrians are located. HD maps provide detailed information about the environment, including road geometry, lane markings, and fixed infrastructure. This method effectively filters out irrelevant points, significantly reducing the data volume and enhancing processing efficiency. By integrating this information with the LiDAR data, it is possible to isolate the ROI, concentrating computational resources on the most pertinent areas. This not only reduces the amount of data to be processed but also ensures that the system's attention is directed toward the most critical parts of the environment.

Even within the ROI, a large number of points fall onto the ground, complicating object detection. Ground points, while essential for certain applications like terrain mapping, can obscure other objects and add unnecessary complexity to the data. To mitigate this, A precise ground removal technique that adapts dynamically to varying slopes and terrain conditions is implemented. This adaptive ground removal method ensures that only the points representing objects of interest are retained, improving the clarity and usefulness of the data. A static ground removal method might fail in environments with significant elevation changes or uneven terrain, but an adaptive approach dynamically adjusts to the slope and undulations of the ground, effectively filtering out ground points while retaining important data.

The ground removal process begins with an initial estimation of the ground plane using a subset of the LiDAR points. The algorithm then adapts to the detected slope, adjusting the ground plane model to account for variations in terrain. As the vehicle moves, the ground removal algorithm continuously updates its model, ensuring that it remains accurate even in changing conditions. This dynamic filtering process is crucial for maintaining the integrity of the data and ensuring that the system can effectively detect and identify objects in the environment.

For object detection, same as the radar, the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) method, which is known for its ability to handle noise and identify clusters of points representing objects, is implemented. However, traditional DBSCAN has limitations in environments with varying point densities and noise levels, such as those caused by snow and rain. These conditions can significantly impact the accuracy of object detection, as fixed parameters in traditional DBSCAN may not be sufficient to handle the variability in the data.

To overcome these limitations, an adaptive version of DBSCAN that dynamically adjusts its parameters based on the current environmental conditions and point densities is developed. This adaptive DBSCAN maintains high accuracy in object detection even under challenging conditions, ensuring reliable performance. The process begins with noise level estimation, where the algorithm considers factors such as snow or rain to determine the current noise level in the environment. Based on this estimation, the DBSCAN parameters are adjusted dynamically. This includes modifying the minimum number of points required to form a cluster and the distance threshold for point inclusion in a cluster. The adapted parameters are then used to form clusters, effectively identifying objects within the LiDAR data. This adaptive approach allows the system to maintain robustness and accuracy in object detection, even in adverse weather conditions. The flexibility of adaptive DBSCAN is particularly valuable in environments where conditions can change rapidly, ensuring that the system remains effective in detecting and identifying objects. In the following, a more elaborate explanation is presented in the following.

LiDAR Downsampling:LiDAR (Light Detection and Ranging) technology has become a cornerstone in applications such as autonomous driving, advanced mapping, and environmental monitoring due to its ability to produce precise 3D representations of the environment. However, one of the main challenges associated with LiDAR is the enormous volume of data it generates. Each scan can produce thousands of data points, creating significant demands on computational resources and storage. This is where LiDAR points downsampling becomes crucial. Downsampling is the process of reducing the number of points in a LiDAR data while preserving the essential features and structures of the scanned environment. The need for downsampling arises from the limitations in processing power and memory capacity. Processing large volumes of high-resolution data in real-time can

overwhelm the system, leading to inefficiencies and potential delays. By reducing the number of points, downsampling helps manage computational load, ensuring that the system can process data more efficiently and effectively.

The process of downsampling can be effectively achieved through the voxel grid filtering method. In this method, the space is divided into a grid of equally sized cubes, known as voxels. Each voxel represents a volumetric pixel in 3D space, and the goal is to replace all points within each voxel with a single representative point. This technique reduces the number of points while maintaining a uniform distribution of points across the scanned area.

The voxel grid filtering process involves the following steps:

1. *Voxel Grid Creation*: The 3D space of the LiDAR scan is divided into a grid of voxels. Each voxel is a cube with a specified side length l. The size of the voxel determines the level of downsampling; larger voxels result in more aggressive downsampling.

2. *Voxel Indexing*: Each point p_i in the LiDAR data is assigned to a voxel based on its coordinates. The voxel index (i, j, k) for a point $p_i = (x_i, y_i, z_i)$ is computed as:

$$i = \left\lfloor \frac{x_i}{l} \right\rfloor, \quad j = \left\lfloor \frac{y_i}{l} \right\rfloor, \quad k = \left\lfloor \frac{z_i}{l} \right\rfloor \tag{4.1}$$

   where $\lfloor \cdot \rfloor$ denotes the floor function, which rounds down to the nearest integer.

3. *Centroid Calculation*: For each voxel, the representative point is calculated as the centroid of all points assigned to that voxel. If P_v is the set of points within voxel v, the centroid c_v is given by:

$$c_v = \frac{1}{|P_v|} \sum_{p \in P_v} p \tag{4.2}$$

   where $|P_v|$ is the number of points in voxel v, and p represents each point within the voxel.

4. *Point Replacement*: Each point within a voxel is replaced by the centroid c_v of that voxel. This results in a significantly reduced number of points, with one representative point per voxel.

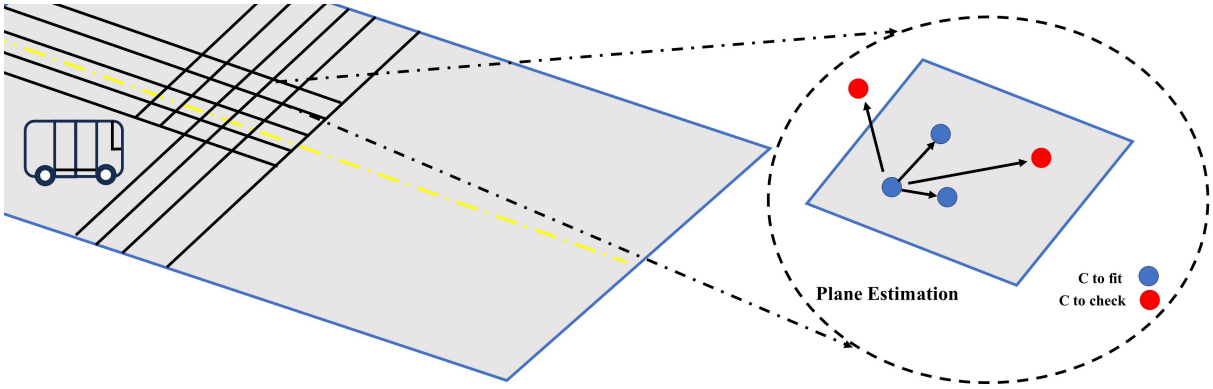

Figure 4.2: The adaptive removal method. On the left the road is divided into a grid format where the closer the grids are to the vehicle, the tighter the grids are, and it relaxes as it moves away. On the right, the overall of the plane fitting method is shown.

The voxel grid filtering method is advantageous because it ensures a uniform reduction in data points across the scanned area, preserving the overall structure and key features of the environment. This method is particularly effective in reducing redundant points in regions where the environment is relatively flat or contains homogeneous features. By adjusting the voxel size l, the level of downsampling can be controlled, allowing for a trade-off between data resolution and processing efficiency.

Eyes on The Road, ROI extraction: The process of focusing on ROI for the LiDAR point cloud is quite similar to the one for the radar in the Chapter 3. Where the limits of the road is a already extracted with the HD map information. The difference however is the point cloud representation Eq. (3.7), that for LiDAR point cloud, there is a shorter set,

$$p^i = \{x_i, y_i, z_i, I_i, \delta_{Ri}, \delta_{Li}\}, \tag{4.3}$$

where x, y, z, and I are longitudinal, lateral, height, and intensity of each point in the point cloud, and δ_R, δ_L are the same as Eq. (3.2). That if it undergoes the same filter as Eq. (3.11), the ROI for the LiDAR points are also extracted. This process, usually optimizes the number of points furthermore after the downsampling.

Adaptive Ground Removal: Initially, the entire point cloud is divided into grids

based on the coarse road boundaries, where each grid undergoes a plane estimation using RANSAC to differentiate obstacle points from the ground. To fit a plane to a set of 3D points using RANSAC, these steps are followed:

1. *Model Definition*: A plane in 3D space can be defined by the equation:

$$\rho_1 x + \rho_2 y + \rho_3 z + \rho_4 = 0, \tag{4.4}$$

   where (ρ_1, ρ_2, ρ_3) is the normal vector to the plane and ρ_4 is the plane's distance from the origin along the normal vector.

2. *Random Sampling*: Randomly select a minimal subset of points from the dataset. For plane estimation, the minimal subset is three points because a plane is uniquely defined by three points.

3. *Plane Calculation*: Using the three randomly selected points, compute the plane parameters. Suppose the three points are $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_2, y_2, z_2)$, and $p_3 = (x_3, y_3, z_3)$. The normal vector can be found using the cross product of the vectors $\overrightarrow{p_1p_2}$ and $\overrightarrow{p_1p_3}$, Figure 4.2:

$$\overrightarrow{p_1p_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1) \tag{4.5}$$

$$\overrightarrow{p_1p_3} = (x_3 - x_1, y_3 - y_1, z_3 - z_1) \tag{4.6}$$

$$(\rho_1, \rho_2, \rho_3) = \overrightarrow{p_1p_2} \times \overrightarrow{p_1p_3} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} \tag{4.7}$$

   Expanding the determinant gives:

$$\begin{aligned} (\rho_1, \rho_2, \rho_3) = ((y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1), \\ (z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_1), \\ (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)) \end{aligned} \tag{4.8}$$

   To find ρ_4, substitute one of the points, say p_1, into the plane equation:

$$\rho_4 = -(\rho_1 x_1 + \rho_2 y_1 + \rho_3 z_1) \tag{4.9}$$

4. *Consensus Set*: Determine how many points from the entire dataset fit the plane model within a specified tolerance. This involves calculating the perpendicular distance from each point (x_i, y_i, z_i) to the plane and counting how many points lie within this distance. The distance ρ_{4i} from a point (x_i, y_i, z_i) to the plane $(\rho_1 x + \rho_2 y + \rho_3 z + \rho_4 = 0)$ is given by:

$$d_i = \frac{|\rho_1 x_i + \rho_2 y_i + \rho_3 z_i + \rho_4|}{\sqrt{\rho_1^2 + \rho_2^2 + \rho_3^2}} \tag{4.10}$$

5. *Best Model Selection*: After all iterations, select the model with the largest consensus set as the best estimate for the plane. The RANSAC algorithm provides a robust way to estimate the parameters of a plane even in the presence of outliers, making it ideal for processing LiDAR data, which often contains significant noise and irrelevant points.

Non-ground points are extracted with Eq. (4.10), and those close to road boundaries are marked as candidate points for curb detection that can be followed in other studies. For the region closest to the vehicle, it uses the height of the LiDAR sensor as the initial plane model, then iteratively improves the plane fitting. This estimation serves as the initial value for ground estimation in other regions. The output for the adaptive ground removal is a list of estimated ground plane models $\mathbb{F} = \{z = f_1(x, y), \cdots, z = f_M(x, y)\}$, where M denotes the number of grids. This adaptive approach allows for dynamic adjustment of fitting thresholds, which are stricter closer to the vehicle to ensure the detection of small objects, thus enhancing safety.

Adaptive Clustering: Falling rain and snow can introduce significant amounts of noisy reflections to LiDAR sensors. This noise can greatly affect the quality of the point cloud data, leading to false detections and potentially triggering emergency stops in autonomous vehicles. Traditional de-noising techniques often struggle with the variability and intensity of noise introduced by adverse weather conditions. Failing to properly de-noise the point cloud data may result in false positive detections, where non-existent obstacles are identified, or false negatives, where actual obstacles are missed. One common approach to mitigating noise in LiDAR data is through clustering methods, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [19]. DBSCAN works by grouping

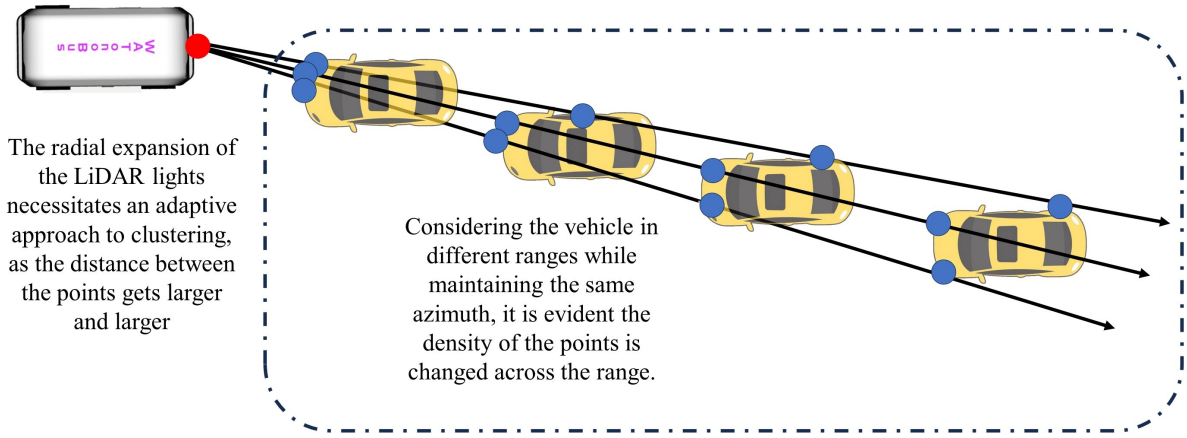

Figure 4.3: The figure to emphasize on the necessity of a adaptive clustering approach for range sensors.

points that are closely packed together, marking them as part of the same cluster, while points that lie alone in low-density regions are treated as noise. The effectiveness of DBSCAN heavily relies on the parameters ϵ, which defines the maximum distance between points in the same cluster, and minPts, which is the minimum number of points required to form a core region. Although clustering methods like DBSCAN can help to some extent by setting a smaller clustering distance threshold ϵ or a larger minimum number of points required to form a core region minPts, these adjustments are not without drawbacks. Reducing ϵ or increasing minPts can inadvertently remove points that belong to actual objects. This can lead to over-segmentation, where points from the same object are divided into multiple clusters, complicating object detection and classification.

To address these issues, an adaptive DBSCAN clustering method has been proposed that leverages the LiDAR scanning pattern to dynamically adjust ϵ and minPts. This approach allows for more accurate clustering that adapts to the specific characteristics of the scanning environment, enhancing the robustness of the de-noising process.

Consider a planar object with height h and width ω facing the LiDAR at a distance Λ. The LiDAR has a horizontal scanning resolution $\Delta\varphi$ and a vertical resolution $\Delta\alpha$. Typically, the point cloud is first downsampled using a voxel grid filter with a voxel size

ΔD to reduce the processing time. The number of scan lines N_Λ and the number of points per scan line N_{pl} falling on the planar object can be calculated as follows:

$$N_\Lambda = \left\lfloor \frac{h}{\max(\Lambda\Delta\alpha, \Delta D)} \right\rfloor \tag{4.11}$$

$$N_{pl} = \left\lfloor \frac{w}{\max(\Lambda\Delta\varphi, \Delta D)} \right\rfloor \tag{4.12}$$

In our Adaptive-DBSCAN method, ϵ is adjusted to maintain a constant number of points per scan line, N_{pl}, ensuring that the clustering adapts to the object size and distance. The number of points per line N_{pl} is determined by the smallest object width $\omega_{\min}$ in the environment and the voxel size ΔD^*, as the final resolution of the point cloud:

$$N_{pl} = \left\lfloor \frac{\omega_{\min}}{\Delta D^*} \right\rfloor \tag{4.13}$$

Thus, the clustering distance threshold $\epsilon(s)$ is given by:

$$\epsilon(\Lambda) = \max(\omega_{\min}, N_{pl}.\Lambda.\Delta\varphi) \tag{4.14}$$

The number of scanning lines $N_\Lambda(\Lambda)$ is calculated based on the minimum height $h_{\min}$ of the objects in the environment:

$$N_\Lambda(\Lambda) = \max(1, \left\lfloor \frac{h_{\min}}{\max(\Lambda\Delta\alpha, \Delta D^*)} \right\rfloor) \tag{4.15}$$

With these adjustments, minPts is set to the product of the number of scan lines and the number of points per line:

$$\text{minPts} = N_\Lambda \cdot N_{pl} \tag{4.16}$$

This adaptive approach allows the clustering algorithm to more accurately reflect the physical characteristics of the environment and the objects within it, improving the robustness and accuracy of object detection. By dynamically adjusting the clustering parameters based on the scanning pattern, Adaptive-DBSCAN can effectively handle the

challenges posed by adverse weather conditions, reducing false detections and improving overall system performance. The proposed Adaptive-DBSCAN method provides a significant improvement over traditional fixed-parameter clustering methods by leveraging the inherent properties of the LiDAR scanning process. This results in more accurate and reliable de-noising of point cloud data, which is crucial for the safe operation of autonomous vehicles in varying environmental conditions.

4.1.2 Radar-LiDAR Fusion System

Chapter 3 went into a novel approach to enhance radar accuracy by reducing false alarms and clutter. The chapter demonstrated how integrating HD map information with radar point cloud data ensures that most points in the radar point cloud correspond to actual objects. Following involves dereferencing the point cloud data from the classes and integrating it with the LiDAR, resulting in a more precise and reliable fusion system.

Camera fusion presents unique challenges because camera detections are 2D projections of the 3D world, often leading to ambiguous detections due to perspective distortions and occlusions. Conversely, LiDAR provides a dense and accurate 3D representation of the environment, capturing details such as the width and height of objects, which greatly facilitates the fusion process.

Given the complementary nature of radar and LiDAR, starting with radar-LiDAR fusion is wiser. Both sensors operate in the 3D coordinate system, making the fusion process more straightforward and robust. The LiDAR's dense point cloud enhances the radar's object detection capabilities by providing detailed spatial information, which helps in accurately distinguishing between objects and background clutter. This fusion leverages the strengths of both sensors, combining radar's ability to detect objects in adverse weather conditions with LiDAR's precise spatial measurements, resulting in a comprehensive and reliable perception system.

Ultimately, the improved radar-HD map perception system, when integrated with LiDAR data, forms a solid foundation for a multi-sensor fusion approach. This approach not only increases the accuracy and reliability of object detection but also paves the way for

incorporating additional sensor data, such as from cameras, in a more effective manner. By tackling the radar-LiDAR fusion first, the system gains a robust 3D understanding of the environment, which is crucial for the subsequent integration of camera data, despite its inherent challenges. This step-by-step fusion strategy ensures a scalable and adaptable perception system capable of operating in complex real-world scenarios.

Radar point clouds are valued for their inclusion of robust features, such as instantaneous relative radial velocity and the Radar Cross Section (RCS), yet they fall short in density when compared to LiDAR point clouds and are also prone to inaccuracies due to clutter and false alarms. Conversely, LiDAR point clouds boast a high density of data points, providing a rich, detailed representation of the environment, but they lack the dynamic features inherent to radar data. By fusing these two data sources, one can harness the strengths of each—utilizing the dynamic, feature-rich information from the radar and the dense, high-resolution spatial information from the LiDAR—to create a more accurate and comprehensive representation. This fusion aims to mitigate the limitations of each sensing modality, enabling a more robust perception system.

The first step of the fusion is radar-LiDAR synchronization, as they should both describe the same scene to be fused. The second step is to transform all the radar point clouds into the LiDAR point clouds so that the search algorithm can be deployed. A KD-Tree (k-dimensional tree) is a space-partitioning data structure used for organizing points in a k-dimensional space. It is particularly useful for applications involving multidimensional search keys, such as range searches and nearest neighbor searches. The construction and utilization of a KD-Tree involve several mathematical concepts and operations. Here, the fundamental steps and equations involved in the KD-Tree algorithm are outlined.

KD-Tree Construction: The KD-Tree is constructed by recursively partitioning the k-dimensional space. Given a set of points $P = \{p_1, p_2, \ldots, p_n\}$ where each point p_i is a k-dimensional vector $p_i = (x_1, x_2, \ldots, x_k)$:

1. *Selecting the Splitting Dimension*: At each level of the tree, a dimension D is chosen to split the set of points. The splitting dimension D is typically chosen in a round-robin fashion or by choosing the dimension with the greatest variance.

2. *Finding the Median*: For the selected dimension D, the points are sorted by their D-th coordinate and find the median. The median point becomes the root of the current subtree. The median, m_D, is chosen to ensure a balanced tree, which helps in optimizing search operations.

3. *Partitioning the Points*: The the points are partitioned into two subsets: those with coordinates less than the median and those with coordinates greater than or equal to the median. Formally, if $p_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$:

$$P_{\text{left}} = \{p_i \in P \mid x_{iD} < m_D\} \tag{4.17}$$

$$P_{\text{right}} = \{p_i \in P \mid x_{iD} \geq m_D\} \tag{4.18}$$

4. *Recursion*: The process is repeated recursively for the left and right subsets, alternating the splitting dimension at each level.

Once the tree is constructed on the LiDAR clusters, the query is formed with the radar points. The search can be efficient operations such as nearest neighbor range search.

1. *Nearest Neighbor Search*: Given a query point q, the goal is to find the point $p^* \in P$ that is closest to q in Euclidean distance. The distance between two points $p = (x_1, x_2, \ldots, x_k)$ and $q = (y_1, y_2, \ldots, y_k)$ is given by:

$$R(p, q) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{4.19}$$

   The search algorithm traverses the tree, starting from the root, and follows the path determined by comparing the query point with the median points. It uses a recursive backtracking approach to explore other branches that might contain closer points.

2. *Range Search*: Given a range defined by $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_k, b_k]$, the goal is to find all points p that lie within the specified range. This involves checking each point against the range criteria for all dimensions:

$$\{p_i \in P \mid a_j \leq x_{ij} \leq b_j \text{ for all } j = 1, 2, \ldots, k\} \tag{4.20}$$

The range search algorithm traverses the KD-Tree, pruning branches that do not intersect with the given range, thereby reducing the number of points that need to be checked.

The LiDAR point cloud with features $p_l = \{x_l, y_l, z_l\}$, and for each radar point with $p_r = \{x_r, y_r, v_r, \text{RCS}\}$, the query, $q(x_r, y_r)$, for nearest neighbours in the radius $R \leq \delta_R$. The final fused point cloud will be an efficient rich feature representation of the environment as,

$$p_f = \{x_f, y_f, z_f, v_f, \text{RCS}\}. \tag{4.21}$$

4.1.3 Camera Perception

Cameras excel in capturing color and texture details but lack depth information. Therefore, images are used for 2D detection, i.e., detecting features on the image plane. In recent years, camera-based 2D detection has been extensively studied in the deep learning field. Compared to 3D detection, it benefits from more public datasets, as it is much easier to label 2D features on the image plane than 3D coordinates, which may require stereo vision or other high-density range sensors to provide depth information. Additionally, single-camera-based 3D detection does not generalize as well as 2D detection, as it is difficult to estimate depth information for cameras with different configurations. YOLOv8 [28] is the latest in the YOLO (You Only Look Once) series of real-time object detectors. It employs state-of-the-art backbone and neck architectures, achieving high accuracy with high inference speed. For this work, we created a customized dataset for outdoor autonomous driving applications by selecting relevant objects from public datasets including COCO (Common Objects in Context) [41], Argoverse [11], and nuScenes [10]. YOLOv8 was then trained on 90% of the dataset and evaluated on the remaining 10%, achieving an mAP50 of 0.708. Finally, the trained model was converted to a TensorRT model to maximize inference speed.

TensorRT is a high-performance deep learning inference library developed by NVIDIA, designed to optimize neural network models for efficient deployment on NVIDIA GPUs. By converting trained models into a highly optimized runtime format, TensorRT reduces

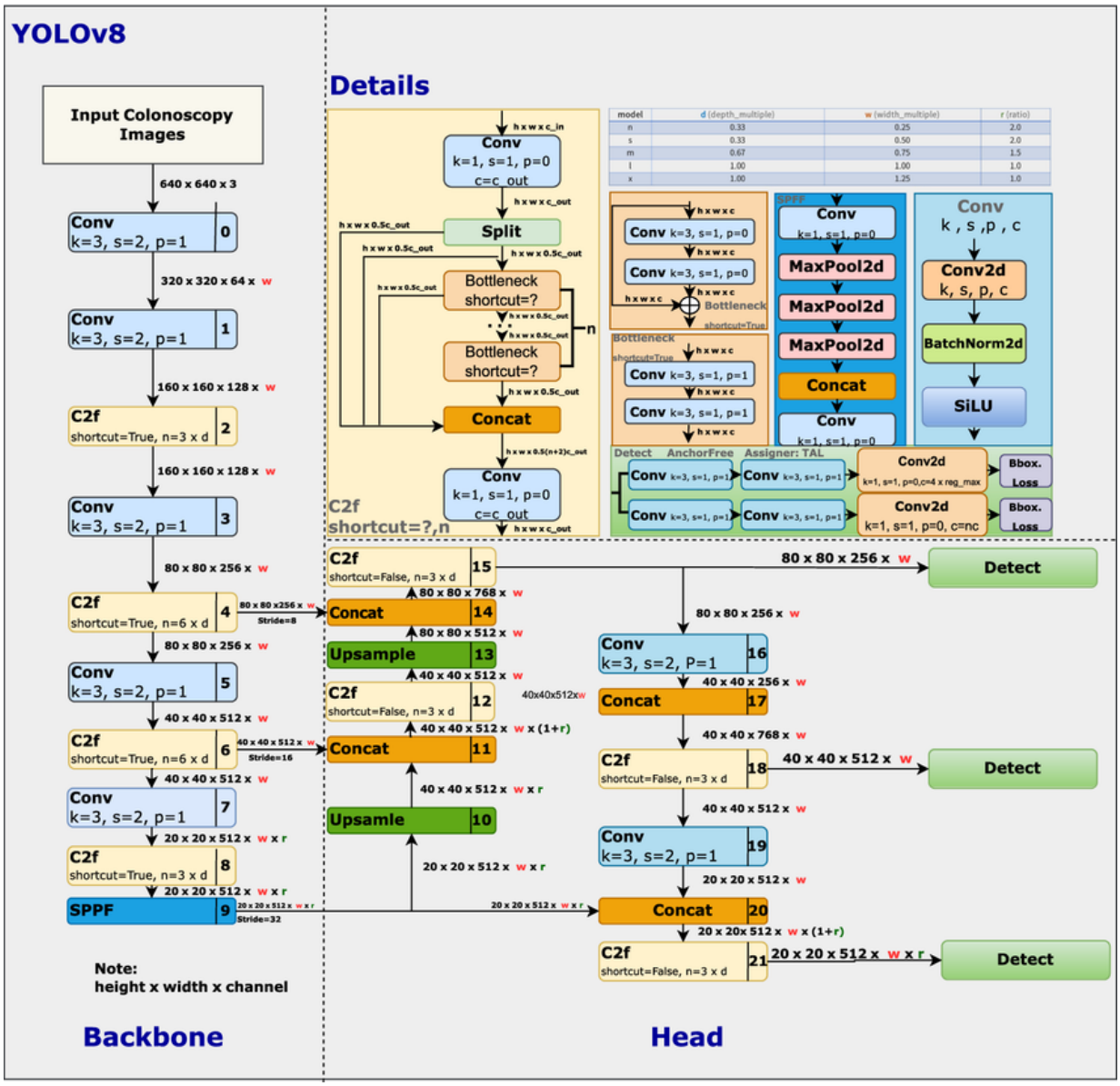

Figure 4.4: The YOLOv8 architecture [28]

latency and increases throughput during inference. The optimization process involves techniques such as layer fusion, precision calibration, kernel auto-tuning, and dynamic tensor memory management. These optimizations enable TensorRT to maximize computational efficiency, leading to significantly faster inference times compared to standard runtime environments. By leveraging the full capabilities of NVIDIA GPUs, TensorRT minimizes data transfer overheads, reduces memory footprint, and executes optimized computational kernels. This makes it an ideal solution for real-time applications requiring high-speed and low-latency predictions.

4.1.4 Camera-3D Point Association

This part aims to fuse the object clustering results from the LiDAR and the 2D object detection results from the cameras to obtain clusters with semantic information. To achieve this, an association cost is calculated for each cluster and each 2D bounding box, and the Hungarian algorithm [34] is used to find the best associations. The key to improving the radar-LiDAR-camera fusion performance is the way to calculate the association cost. The 2D association cost based on the intersection over union (IOU) of the camera bounding box and the projected cluster points bounding box is a common choice due to its simplicity. However, it fails in handling distant objects where the bounding box is small and the intersection region could be zero due to imperfect calibration or object motion. We fix this issue by introducing the 3D information into the association cost.

A depth and its uncertainty estimation method using the semantic 2D bounding box is proposed. Considering a common camera placement scheme, where there is a pitch angle θ between the camera coordinate plane X_cOZ_c and the world coordinate plane X_wOZ_w parallel to the ground surface. The projection of a point $P_w = (x_w, y_w, z_w)$ in the world

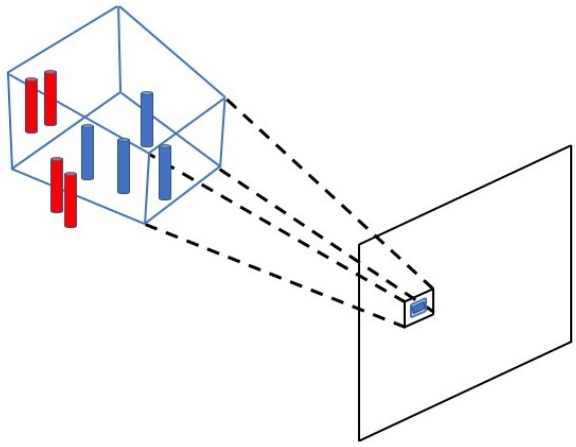

Figure 4.5: The Frustum approach for camera and 3D point association.

coordinate to the pixel coordinate $p_p = (x_p, y_p)$ can be expressed as

$$\begin{aligned} s \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} &= K R_\theta \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \\ &= \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \end{aligned} \tag{4.22}$$

Based on this projection matrix, for an object having a fixed distance z_w, we have:

$$s = y_w \sin\theta + z_w \cos\theta \tag{4.23a}$$

$$s\Delta x_p = f_x \Delta x_w \tag{4.23b}$$

$$s\Delta y_p = ((c_y - y_p)\sin\theta + f_y \cos\theta)\Delta y_w \tag{4.23c}$$

Replace the Δx_w and Δy_w to W_w and H_w denoting the width and height of the object in the world coordinate, and replace the Δx_p and Δy_p to W_p and H_p denoting the width and height of the 2D image bounding box of the object. The y_w and y_p will be the y-coordinate

of the center of the object and bounding box, respectively. For an object standing on the ground, $y_w = H_w^{cam} - H_w/2$, where H_w^{cam} denotes the height of the camera relative to the ground plane. So to calculate the final distance z_w and its uncertainty σ_z^2, we need to calculate the depth estimation based on width and the height prior. Estimation based on width:

$$z_w^{\omega} = \frac{f_x W_w}{W_p \cos\theta} - y_w \tan\theta$$

$$\sigma_z^{\omega^2} = \frac{W_p^4 \sigma_{y_w}^2 \sin^2\theta + W_p^2 f_x^2 \sigma_{W_w}^2 + W_w^2 f_x^2 \sigma_{W_p}^2}{W_p^4 \cos^2\theta}$$

$$\sigma_y^{\omega^2} = \sigma_{H_w^{cam}}^2 + \sigma_{H_w}^2/4$$

and the estimation based on height:

$$z_w^h = \frac{((c_y - y_p)\sin\theta + f_y \cos\theta) H_w}{H_p \cos\theta} - y_w \tan\theta$$

$$\begin{aligned}\sigma_z^{\omega^2} = \frac{1}{H_p^4 \cos^2\theta} \Big[& H_p^4 \sigma_{H_w^{cam}}^2 \sin^2\theta \\ & + \frac{H_p^2 \sigma_{H_w}^2 \left(H_p \sin\theta + 2 f_y \cos\theta + 2\left(c_y - y_p\right)\sin\theta\right)^2}{4} \\ & + H_w^2 \sigma_{H_p}^2 \left(f_y \cos\theta + (c_y - y_p)\sin\theta^2\right]\end{aligned}$$

Finally, the measurements are averaged based on the variances:

$$z_w = \frac{z_w^h \sigma_z^{\omega^2} + z_w^{\omega} \sigma_z^{h^2}}{\sigma_z^{\omega^2} + \sigma_z^{h^2}} \tag{4.26a}$$

$$\sigma_z^2 = \frac{\sigma_z^{h^2} \sigma_z^{\omega^2}}{\sigma_z^{\omega^2} + \sigma_z^{h^2}} \tag{4.26b}$$

This estimated depth information will be used as part of the association cost defined as,

$$\pi_{i,j} = \delta(1 - \mathrm{IOU}_{i,j}) + (1 - \delta)\Delta(\rho_w^{cam}, \rho_w^{Li}) \tag{4.27}$$

where, $\mathrm{IOU}_{i,j}$ is calculated between the LiDAR projected bounding box, and the camera's, δ is the importance weight, $\Delta()$ takes the mahalanobis distance between the object position estimate from camera, ρ_w^{cam}, and from LiDAR, ρ_w^{Li}, and $\pi_{i,j}$ forms the Hungarian matrix Π_{Li}^{cam}, and can be solved with Hungarian matching algorithm in [34].

4.1.5 Radar-LiDAR-Camera Fusion

The radar-LiDAR fusion process provides clusters with RCS values, 3D positions, and their radial compensated radar velocity, resulting in closed contours that delineate the 3D occupied regions in the space ahead of the vehicle. These clusters, which offer detailed spatial information, are then integrated with 2D detections from the camera in the association step. During this association, clusters that successfully match with camera detections are assigned a class, while those that do not remain classified as unknown. The final outcome is a set of contours, each either labeled with a specific class or marked as unknown. The subsequent step involves tracking these contours across frames, allowing for continuous monitoring of objects and their movements relative to the vehicle.

4.2 Object Tracking

Motion features play a pivotal role in autonomous vehicle (AV) software, influencing critical decisions such as object filtering. The sensors integrated into vehicles, such as LiDAR and cameras, primarily provide positional or semantic data. However, when it comes to measuring motion, sensors like radar offer partial observations, such as doppler velocities, which indicate the radial velocity of objects. This partial view can introduce ambiguities, particularly in scenarios involving lateral movements. To resolve these ambiguities for individual actors, integrating Extended Kalman Filter (EKF) with position measurements can yield a comprehensive understanding of object motion states. Once motion states are accurately determined for each actor, this approach can be extended to encompass the entire environment and object list. This scalable algorithm ensures robust decision-making and efficient object tracking within AV systems.

4.2.1 Individual Object State Observer

For a general type of object we can define the constant acceleration motion model as,

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} \tag{4.28}$$

Algorithm 2 Calculation of score matrix for tracking object association

Input: agents, tasks, can_match, score_matrix

Output: Updated score_matrix with computed scores

```
function CALCULATE_SCORE_MATRIX(agents, tasks, can_match, score_matrix)
    rows ← size of tasks
    cols ← size of agents
    Initialize score_matrix as a zero matrix with dimensions rows × cols
    task_idx ← 0
    for each task in tasks do
        for agent_idx from 0 to size of agents do
            score ← 0.0
            if can_match(agent_idx, task_idx) then
                distance ← hypot(task.x -agents[agent_idx].x, task.y - agents[agent_idx].y)
                if distance ≥ max_dist then
                    score ← 0.0
                else
                    score ← (max_dist - distance) / max_dist
                end if
            end if
            score_matrix[task_idx][agent_idx] ← score
        end for
        task_idx ← task_idx + 1
    end for
end function
```

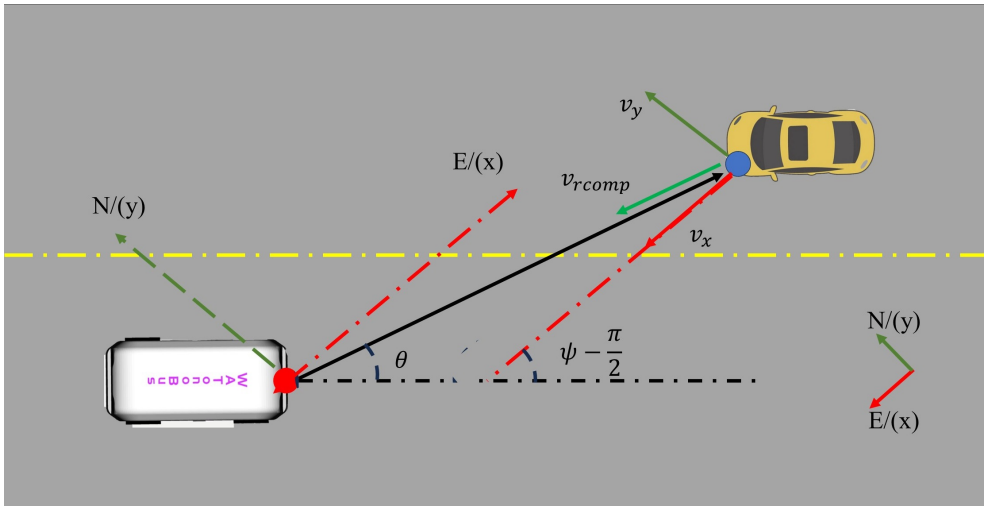

Figure 4.6: The figure to show the velocity in the world coordinate in relation to the compensated doppler velocity. The world coordinate is the East-North frame, and to improve robustness, the observer is designed in this frame.

Algorithm 3 Kalman Filter [74]

Initialize state estimate $\bar{\mu}_0$ and covariance P_0
Initialize state estimates $\bar{\mu}_t$ and covariances P_t for $t = 1, \ldots, n$
for $k = 1$ **to** n **do**
    Predict:
    Predicted state estimate: $\bar{\mu}_t = f(\mu_{t-1})$
    Predicted covariance estimate: $\bar{P}_t = \mathbf{F}_{t-1} P_{t-1} \mathbf{F}_{t-1}^T + Q$
    Update:
    Kalman gain: $K_t = P_t H^T (H P_t H^T + R)^{-1}$
    Updated state estimate: $\mu_t = \bar{\mu}_t + K_t(z_t - H\bar{\mu}_t)$
    Updated covariance estimate: $P_t = (I - K_t H)\bar{P}_t$
end for

where,

$$\mathbf{x} = [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}]^T \tag{4.29}$$

The state transition matrix $\mathbf{A}$ for this model, assuming a sampling time Δt, is given by:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.30}$$

The Δt is measured real-time, from synchronized fusion frame results. However, software-wise the synchronization is strictly controlled over 10 Hz, meaning that the $\Delta t = 0.1$. The effectiveness of sensor measurements directly influences the efficacy of data fusion processes. This integration facilitates a quicker and more comprehensive observation of the vehicle's state by combining positional information with velocity insights provided by radar. On the other hand, if radar fusion is unsuccessful or incomplete, the observer is left with only positional data from LiDAR. In such cases, the availability of velocity and acceleration observations is delayed, potentially limiting the real-time responsiveness and completeness of the system's perception capabilities. Thus, the measurement matrix if radar-LiDAR fusion is a success is as following,

$$\mathbf{H_1} = \begin{bmatrix} x \\ y \\ \dot{x}\cos(\alpha) + \dot{y}\sin(\alpha) \end{bmatrix} \tag{4.31}$$

where α is the angle between the radial coordinate of the object velocity and the east coordinate showing in the Figure 4.6, since the ego-vehicle heading angle ψ compliment is the external angle to the triangle,

$$\psi - \frac{\pi}{2} = \alpha + \theta \tag{4.32}$$

where re-ordering the equation yields,

$$\alpha = (\psi - \theta) - \frac{\pi}{2}. \tag{4.33}$$

To avoid complicating the nonlinearities in the equation, α is put in the equation straight from the radar range and azimuth measurement, as the x and y are already in the filter. In the case of unsuccessful fusion with the radar,

$$\mathbf{H_2} = \begin{bmatrix} x \\ y \end{bmatrix}, \tag{4.34}$$

where the R matrix also updates based on the H matrix. In the case of not receiving the measurement for that specific object, the prediction part of the EKF is used. This is implemented through a object oriented program that tracks multiple objects where each has the EKF as an attribute, and hence making the scaling possible.

4.2.2 Scaling The State Observers

The Eq. (4.28) is used in Algorithm. (3) to provide full observation on the object motion states. However, in the case of having too many objects, the algorithm should decide on associating new measurement correctly to the old ones. This can be achieved by forming a score function that maximise when the two consecutive measurements are from one object, and minimizes when it is not. However, for the cases that objects move close to one another, the algorithm can leverage the semantic features to reject unplausible associations entirely. For instance, if the object was a pedestrian, then the new measurement should also demonstrate the pedestrian semantics, such as RCS, if applicable, and the class type. This is implemented through *can_match* function 4, that returns true if the semantics are consistent and false if not.

The score function mainly focuses on the distance between the predicted position of the object, and the measurement received.

$$s_{i,j} = \frac{d_{max} - d_{i,j}}{d_{max}} \tag{4.35}$$

where $s_{i,j}$ is forced between the last measurement of the velocity times 0.1, and 1 meter.

The association algorithm is based on Successive Shortest Path Problem known as SSPA [1], which takes a score matrix between the old measurements and the new ones,

Algorithm 4 *can_match* function

Input: class_i, class_j, rcs_i, rcs_j, δ
Output: Boolean value (`true` or `false`)
if $\text{class}_i == \text{class}_j$ **then**
    return `true`
else if $(\text{class}_i \neq \text{class}_j)$ **and** $(\text{class}_i == \text{unknown}$ **or** $\text{class}_j == \text{unknown})$ **then**
    return `true`
else if $\dfrac{|\text{rcs}_i - \text{rcs}_j|}{\max\left(|\text{rcs}_i|,\ |\text{rcs}_j|\right)} < \delta$ **then**
    return `true`
else
    return `false`
end if

put as task and agent, then it outputs the plausible pairs. The algorithm to calculate the weight matrix is brought in Algorithm (2).The Successive Shortest Path (SSP) algorithm is an effective method for solving the association problem when given a score matrix between objects. In this context, there are two sets of objects, A and B, and a score matrix S where $S(i, j)$ represents the score or cost of associating object $i \in A$ with object $j \in B$. The goal is to find the optimal pairing of objects from set A to set B that minimizes the total cost or maximizes the total score. The problem can be modeled as a minimum-cost flow problem in a bipartite graph. The cost function to minimize is given by:

$$\text{Minimize} \sum_{i \in A} \sum_{j \in B} S(i, j) x_{ij}, \tag{4.36}$$

where $x_{ij} = 1$ if object i is assigned to object j, and 0 otherwise. The constraints are:

$$\sum_{j \in B} x_{ij} = 1 \quad \forall i \in A \quad \text{(Each object in } A \text{ is assigned to exactly one object in } B) \tag{4.37}$$

$$\sum_{i \in A} x_{ij} = 1 \quad \forall j \in B \quad \text{(Each object in } B \text{ is assigned to exactly one object in } A) \tag{4.38}$$

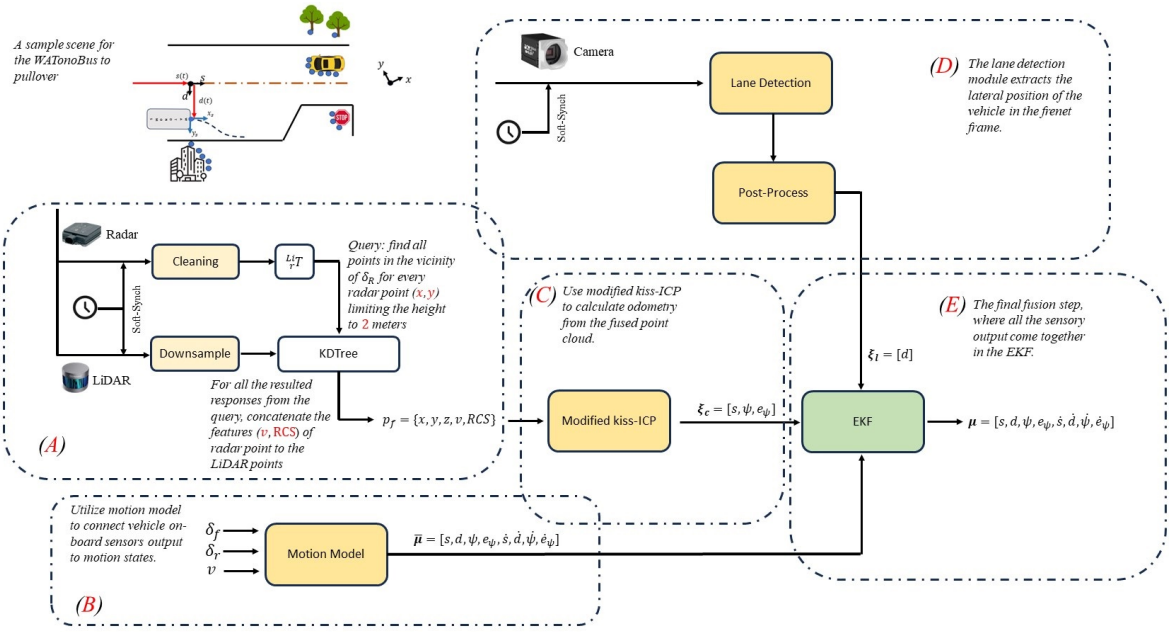

Figure 4.7: The final fusion where the perception and localization are unified.

$$x_{ij} \in \{0, 1\} \tag{4.39}$$

The SSP algorithm constructs a flow network with a source and a sink, and iteratively finds the shortest augmenting paths from the source to the sink in the residual network. By augmenting flow along these paths and updating the residual capacities, the algorithm efficiently finds the optimal assignment. This process continues until no more augmenting paths exist, ensuring that the total score is maximized or the total cost is minimized, thereby providing an optimal solution to the association problem.

4.3 Localization

For the localization process, this approach offers a significant advantage by utilizing the already processed data from the perception module, thereby unifying the perception and localization functions. In contrast, other methods in the literature often require an entirely separate processing step, which demands substantial memory and CPU resources. By leveraging the fused point cloud and camera lane detection data, subject of other studies [94], which are already handled by the perception module, the current method efficiently integrates localization as a byproduct. This integration not only reduces computational overhead but also enhances the overall system performance by minimizing redundant processing tasks. The overall structure of the localization method is outlined in this section. As illustrated in Figure 4.7, the approach includes an optimal fusion of vehicle perception, on-board sensors, and the HD map to achieve localization in the Frenet frame. In part (A), the synchronized point cloud from radar and LiDAR is captured and fused to generate a more accurate and comprehensive representation of the environment. This fused point cloud enhances the scene's detail and reliability, which is crucial for precise localization. In part (B), the on-board sensors are employed to predict the motion of the autonomous vehicle (AV) using the vehicle motion model. This step also constitutes the prediction phase of the Extended Kalman Filter (EKF), providing essential predictive insights based on the vehicle's dynamics. In part (C), the fused point cloud is processed by the modified KISS-ICP (Iterative Closest Point) algorithm [76] to provide odometry information. This algorithm refines the vehicle's position and orientation by matching the current point cloud with the reference map. In part (D), lane detection outputs from the camera are utilized to measure the vehicle's lateral position. This information is critical for maintaining lane-level accuracy and ensuring the vehicle stays within its designated path. Finally, in part (E), all sources of information—fused point cloud data, vehicle motion predictions, and lane detection outputs—are integrated using the EKF. This comprehensive fusion within the EKF framework ensures robust and reliable localization by combining the strengths of each data source, resulting in a unified and accurate localization solution.

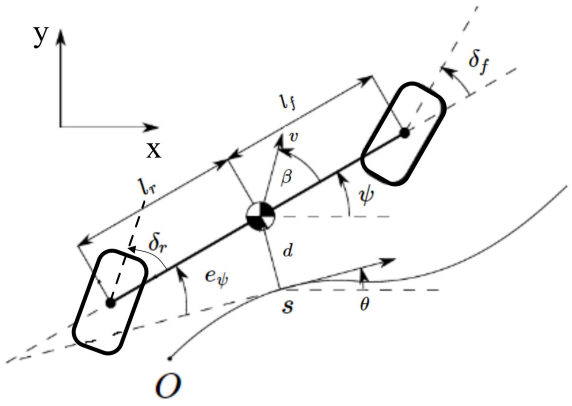

Figure 4.8: Bicycle model in Frenet frame

4.3.1 Accurate Motion Model

In this part, the motion model required to connect on-board sensor output to vehicle states is developed. Various parameters are incorporated into the motion model for the WATonoBus to accurately describe the vehicle's navigation based on its inputs and outputs, leveraging information from wheel encoders and steering sensors. The velocity of the center of mass (v) represents the vehicle's speed along its direction of travel, as measured by the wheel encoders. The front and rear steering angles (δ_f and δ_r, respectively) are inputs obtained from steering sensors, dictating the vehicle's turning behavior, as shown in Figure 4.8. The motion model for the WATonoBus is written as [59],

$$\beta = \tan^{-1}\left(\frac{l_r \tan(\delta_f) + l_f \tan(\delta_r)}{l_f + l_r}\right) \tag{4.40}$$

$$\dot{\psi} = \frac{(\tan(\delta_f) - \tan(\delta_r))\, v \cos(\beta)}{l_f + l_r} \tag{4.41}$$

$$\dot{x} = v \cos(\psi + \beta) \tag{4.42}$$

$$\dot{y} = v \sin(\psi + \beta) \tag{4.43}$$

where x, y, β, ψ, and $\dot{\psi}$ are the 2D position of the vehicle, vehicle side slip angle, vehicle yaw, and vehicle yaw rate, respectively. The distances between the front and rear axles to

the centroid of the vehicle are l_f and l_r, respectively. Next, the kinematic bicycle model is transferred from the Cartesian coordinate system to a moving Frenet reference frame [8]. The Frenet coordinate system defines the position using the signed curvilinear abscissa, s, which is the length of the path from the origin of the curve to the point on the curve closest to the center of mass of the vehicle, and the signed distance from the curve, d, which is the lateral distance from the closest point on the curve to the vehicle. The Frenet system is rotated about the tangent angle θ compared to the Cartesian coordinate system. Using the above definition, the kinematic bicycle model is written in the Frenet frame as follows:

$$\dot{s} = v\frac{\cos(e_\psi + \beta)}{1 - d\kappa(s)} \tag{4.44}$$

$$\dot{d} = v\sin(e_\psi + \beta) \tag{4.45}$$

$$\dot{e_\psi} = \dot{\psi} - \kappa(s)\dot{s} \tag{4.46}$$

where e_ψ is the orientation error between the vehicle yaw, ψ, and the tangent angle $\theta(s)$ of the Frenet frame compared to the Cartesian coordinate system.

4.3.2 Adjusted ICP Approach

The kiss-ICP [76] algorithm represents a significant advancement in the adjustment of the ICP. By streamlining the ICP process, kiss-ICP not only simplifies the algorithm's implementation but also enhances its performance in terms of both efficiency and accuracy. The method itself is an adjustment to the ICP method and is summarized in four steps: motion prediction and de-skewing the point cloud, point cloud sub-sampling, generating the local map and correspondence estimation, and finally aligning the new point cloud frame with the generated local map through robust optimization. These steps have been modified in this work. In the case of implementation on a vehicle, three challenges were observed:

- The algorithm's lateral estimation in the Frenet coordinate, d, is not as accurate as its yaw, ψ, and s estimation. This inaccuracy could be attributed to the constant

velocity motion model considered in the original kiss-ICP, which fails to account for e_ψ and its effect on $\dot{d}$ in Eqs. (4.45) and (4.46). This issue has been addressed and improved in this work through fusion with camera data.

- The method's robustness can be enhanced by focusing on the static parts of the local map. Achieving this using LiDAR features alone is challenging; however, employing radar-LiDAR fusion allows the use of velocity features to create a more robust local map.

- For outdoor LiDARs, the substantial size of the point clouds results in increased CPU resource consumption. This issue can be mitigated by directly inputting the fused point cloud into the algorithm, thereby modifying the second step in the original kiss-ICP implementation. This adjustment serves as an improved alternative for smart subsampling, making the representation focus on areas of the map with significant features rather than uniformly decreasing the density.

These modifications and enhancements to the kiss-ICP algorithm significantly improve its application in vehicle localization, addressing key challenges and optimizing performance.

4.4 Final Fusion

The final fusion is implemented using EKF [74]. The profound benefits of frenet frame in control and tracking of the AVs, pushes the development of the localization algorithm in the frenet frame as well. The modified version of the kiss-ICP provides an observation on vehicle states in frenet frame as $\zeta_c = [s, \psi, e_\psi]$. The d measurement from the modified kiss-ICP is far less accurate than the direct measurement from camera, $\zeta_l = [d]$, thus the final measurement states are $\zeta = [\zeta_c, \zeta_l, \delta_f, \delta_r, v]$. Based on the motion model which is discretized using Euler method, the frenet related states are expressed in $\nu = [s, d, \psi, e_\psi]$ and the input from the vehicle are $u = [\delta_f, \delta_r, v]$, that makes the final prediction states to be $\bar{\mu} = [\nu, \dot{\nu}, u]$ which is the joint representation of both states and input, making the

reordered equations as,

$$\dot{\nu} = \mathbf{g}(\nu, \dot{\nu}, u) \tag{4.47}$$

then the jacobian F would be,

$$\mathbf{F} = \begin{bmatrix} \mathbf{1} & \mathbf{dt} & 0 \\ \partial\mathbf{g}/\partial\nu & \partial\mathbf{g}/\partial\dot{\nu} & \partial\mathbf{g}/\partial u \\ 0 & 0 & \mathbf{1} \end{bmatrix}, \tag{4.48}$$

$$\frac{\partial \mathbf{g}}{\partial \nu_t} = \begin{bmatrix} (\partial\kappa/\partial s)\gamma^2 v d c_\alpha & \kappa(s)\gamma^2 v d c_\alpha & 0 & -\gamma v s_\alpha \\ 0 & 0 & 0 & v c_\alpha \\ 0 & 0 & 0 & 0 \\ (-\partial\kappa/\partial s)\dot{s} & 0 & 0 & 0 \end{bmatrix}, \tag{4.49}$$

$$\frac{\partial \mathbf{g}}{\partial \dot{\nu}_t} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\kappa(s) & 0 & 1 & 0 \end{bmatrix}, \tag{4.50}$$

$$\frac{\partial \mathbf{g}}{\partial u_t} = \begin{bmatrix} (-\partial\beta/\partial\delta_f).v s_\alpha \gamma & (-\partial\beta/\partial\delta_r).v s_\alpha \gamma & c_\alpha \gamma \\ (-\partial\beta/\partial\delta_f).v c_\alpha & (-\partial\beta/\partial\delta_r).v c_\alpha & s_\alpha \\ -\partial\dot{\psi}/\partial\delta_f & -\partial\dot{\psi}/\partial\delta_r & -\partial\dot{\psi}/\partial v \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.51}$$

in which,

$$\gamma = \frac{1}{1 - \kappa(s)d}, \tag{4.52}$$

$$\alpha = e_\psi + \beta, \tag{4.53}$$

the partial derivatives in Eq. (4.51) are,

$$\begin{aligned} \frac{\partial \dot{\psi}}{\partial \delta_f} &= (1 + \tan^2(\delta_f)) \frac{v \cos(\beta)}{l_f + l_r} \\ &\quad - \frac{\tan(\delta_f) - \tan(\delta_r)}{l_f + l_r} \cdot \frac{\partial \beta}{\partial \delta_f} v \sin(\beta), \end{aligned} \tag{4.54}$$

$$\frac{\partial \dot{\psi}}{\partial \delta_r} = -(1+\tan^2(\delta_r))\frac{v\cos(\beta)}{l_f+l_r} - \frac{\tan(\delta_f)-\tan(\delta_r)}{l_f+l_r}\cdot\frac{\partial \beta}{\partial \delta_r} v\sin(\beta), \tag{4.55}$$

$$\frac{\partial \dot{\psi}}{\partial v} = \frac{(\tan(\delta_f)-\tan(\delta_r))\cos(\beta)}{l_f+l_r}, \tag{4.56}$$

$$\frac{\partial \dot{\psi}}{\partial v} = \frac{(\tan(\delta_f)-\tan(\delta_r))}{l_f+l_r}.\cos(\beta), \tag{4.57}$$

$$\frac{\partial \beta}{\partial \delta_{f,r}} = \frac{l_{r,f}}{1+\mathrm{X}^2}.\frac{(1+\tan^2(\delta_{f,r}))}{l_r+l_f}, \tag{4.58}$$

$$\mathrm{X} = \frac{l_f\tan(\delta_r)+l_r\tan(\delta_f)}{l_r+l_f}. \tag{4.59}$$

The state ν is observable as s and ψ come from modified Kiss-ICP, d comes from lane detection, and e_ψ comes as,

$$e_\psi = \psi - \theta(s), \tag{4.60}$$

where the ψ and s are from the previous estimate from EKF, and $\theta(s)$ comes from frenet frame. The input state u is also measured by the sensors placed on the vehicle, $\delta_{f,r}$ comes from the steering sensor, and v comes from the wheel encoders. Making the H matrix as,

$$H = \begin{bmatrix} \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} \end{bmatrix}. \tag{4.61}$$

where plugging the formed matrices in the Algorithm 3, make the observation of the motion states possible. Therefore, localizing the vehicle in the frenet coordinate.

4.5 Summary

In this chapter, it was demonstrated how to effectively process LiDAR data by using the HD map for ROI extraction of LiDAR points, resulting in a more efficient representation of the environment. The subsequent adaptive ground removal technique was shown to effectively eliminate ground points, and adaptive clustering was introduced to overcome noise from snow and rain in the LiDAR point cloud. The fusion of radar points with the LiDAR point cloud was then described, producing a point cloud with RCS and velocity attributes. This fused point cloud was further combined with camera detections from YOLO v8 through a cost-based novel frustum approach. The final objects were tracked across frames using a novel association algorithm that employs radar and semantic features. The perception results were utilized in the localization module as well, leveraging the motion model and wheel encoders. The EKF was implemented in the Frenet coordinate, leading to a robust unified perception-localization module. In the next chapter, the results of this approach will be presented.

Chapter 5

Experiments and Results

In this chapter, the experimental platform, WATonoBus [5], features, and the experiments are presented. Following this, the results of the unified perception-localization module are detailed and discussed. The results begin by showing the effects of HD map fusion on both the radar point cloud and object detection accuracy, then expands to the perception reliability and effectiveness. Some of the edge cases for the perception module are presented and studied, and subsequently, the accuracy of the localization component of the algorithm is demonstrated through a detailed comparison between the estimated location in frenet coordinate, and the ground truth measured by GPS.

5.1 Experiments

For the empirical validation of the proposed methodology, a series of data collection sessions were conducted using the WATonoBus, as depicted in Figure 5.1. This autonomous shuttle, built at the Mechatronics Vehicle Systems (MVS) laboratory at the University of Waterloo, served as the testbed for various experiments. The sessions took place during late winter and early spring, capturing a wide range of environmental scenarios including rain, snowfall, and clear skies. This comprehensive dataset allows for a robust evaluation of this study across different weather conditions. Unlike methodologies that require extensive datasets, the perception-localization module is efficient with its data requirements.

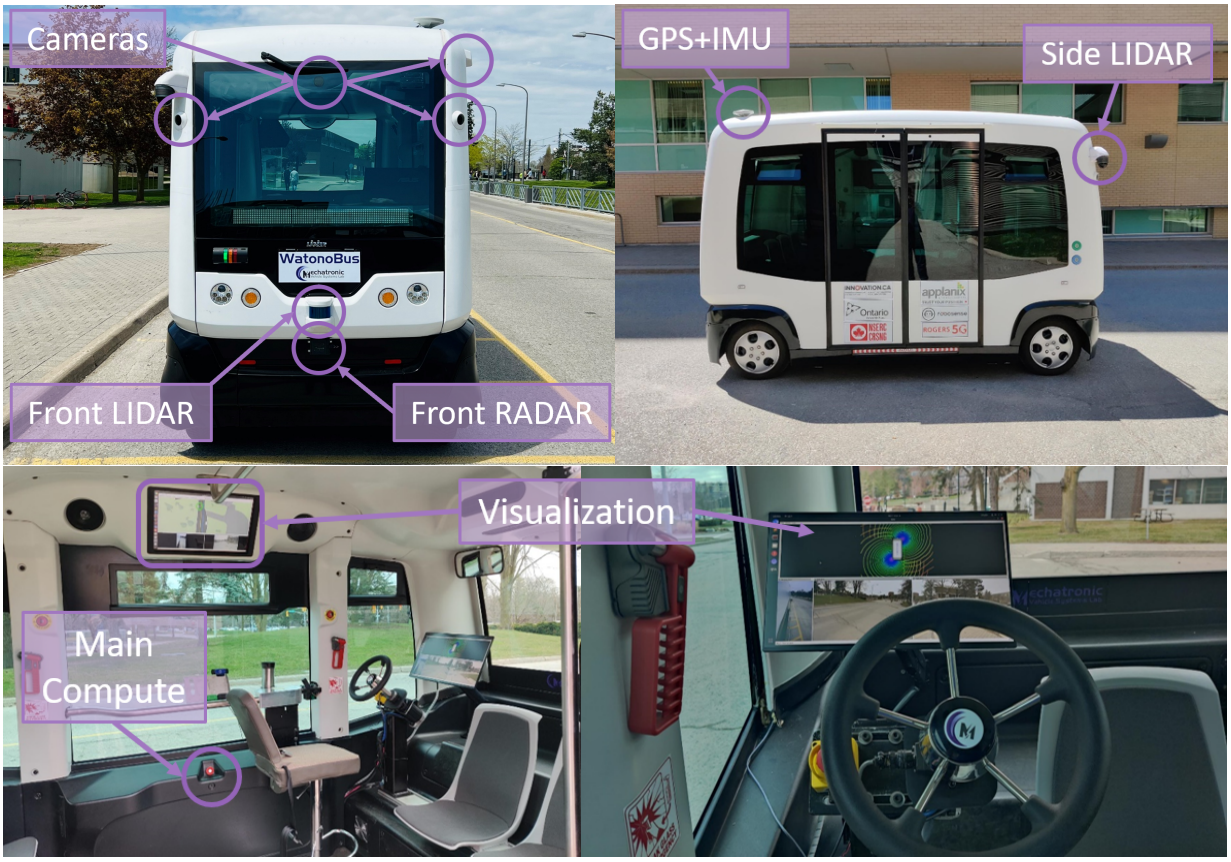

Figure 5.1: Illustration of WATonoBus sensor suite, compute system, control interface, and visualization utilities.

The data collection sessions gathered a total of three hours of multi-modal sensor data, which, despite being relatively limited in duration, is sufficiently diverse to demonstrate the robustness of our algorithms.

The sensor suite used for data collection includes 1 LiDAR sensor, 2 radar units, and 3 cameras, as detailed in Table 5.1. These sensors were carefully positioned to capture the frontal view of the vehicle traveling at speeds up to 20 kph. While the WATonoBus is equipped with a more elaborate array of sensors, including 6 cameras capturing RGB images at 10 FPS with a resolution of 1920x1200 pixels using JPEG compression [77], the focus for this data collection was on the front view. The two radars installed on the

Table 5.1: Sensor specifications and details.

Sensor	**Details**
3x Camera	RGB, 10Hz capture frequency, 1/1.8" CMOS sensor, 1920 × 1200 resolution, auto exposure, JPEG compressed.
1x Lidar	Spinning, 32 beams, 10Hz capture frequency, 360° horizontal FOV, -30° to 10° vertical FOV, $\leq 70m$ range, $\pm 2cm$ accuracy, up to $60K$ points per second.
2x Radar	$\leq 250m$ range, 77GHz, FMCW, 13Hz capture frequency, ±0.1km\ h vel. accuracy.
GPS & IMU	GPS, IMU, AHRS. 0.2° heading, 0.1° roll/pitch, 20mm RTK positioning, 50Hz update rate.

vehicle operate at a frequency of 77 GHz, providing a range of 250 meters and a capturing frequency of 13 Hz. The LiDAR sensor, with its 32 beams and a capture frequency of 10 Hz, generates 60 thousands points per scan. Since the LiDAR is mechanical and the vehicle is moving, the scans are compensated based on the vehicle velocity.

The vehicle's processing unit, a Jetson AGX ORIN with 64 GB of memory, runs a specialized version of the Linux kernel provided by NVidia to meet real-time operating system standards. A middleware, Robot Operating System (ROS), is employed to facilitate soft synchronization and real-time standardized communication between the various processing nodes. The accuracy of the ground truth for validating the localization is ensured by a GPS-IMU unit, providing updates at a rate of 50 Hz with an accuracy of 20 millimeters in position, 0.2 degrees in heading, and 0.1 degrees in pitch and roll.

For operational safety and monitoring, the WATonoBus is equipped with two indoor monitors, one for the passengers, one for the driver, and a steering wheel for the safety operator. Additionally, an emergency stop mechanism is in place to lock the wheels in the event of a critical safety situation. All vehicle operations, from door control and blinkers to autonomous driving, are managed by the autonomous software developed by the Mechatronics Vehicle Systems laboratory. The environment of the experiment is the

ring-road of the University of Waterloo campus. This comprehensive setup ensures that the WATonoBus can safely navigate various environments while providing reliable data for validating the perception and localization algorithms.

5.2 Dataset

The dataset contains 2 loops of normal sunny weather, 2 loops of heavy snow, 1 loop of light snow, and 1 loop of heavy raining. Using the customized annotator developed by the MVS, the ground truth of the dataset contains the object classes, positions, and 3D bounding boxes, and the ground truth drivable space. The annotation process considers various classes, notably: Vehicles (inclusive of cars, trucks, and buses), Pedestrians, and Irrelevant detections, e.g., metal poles, light posts, trash cans, false alarms, ground points, curb points, etc.

For the sake of radar, the annotator also categorizes the irrelevant detections into two classes, first, the metal objects, e.g., poles, and trash cans, and second, false alarms; this is to facilitate the data-driven classification, as the radar features for these objects are quite diverse, e.g., the σ_{RCS} for the metal objects is 20 dbm^2 and for the false alarms is -5.5 dbm^2. In this study, these two categories are denoted as Noise I, and Noise II, respectively.

5.3 Results and Discussion

In this section, both the qualitative and quantitative results of the unified perception-localization is presented and discussed. The results are broken down into three stages; the radar cleansing results through its fusing with HD map, the radar-LiDAR-camera fusion with HD map as object detection module, the tracking results, the localization results, and finally the whole module running runtime, memory and CPU-GPU usage.

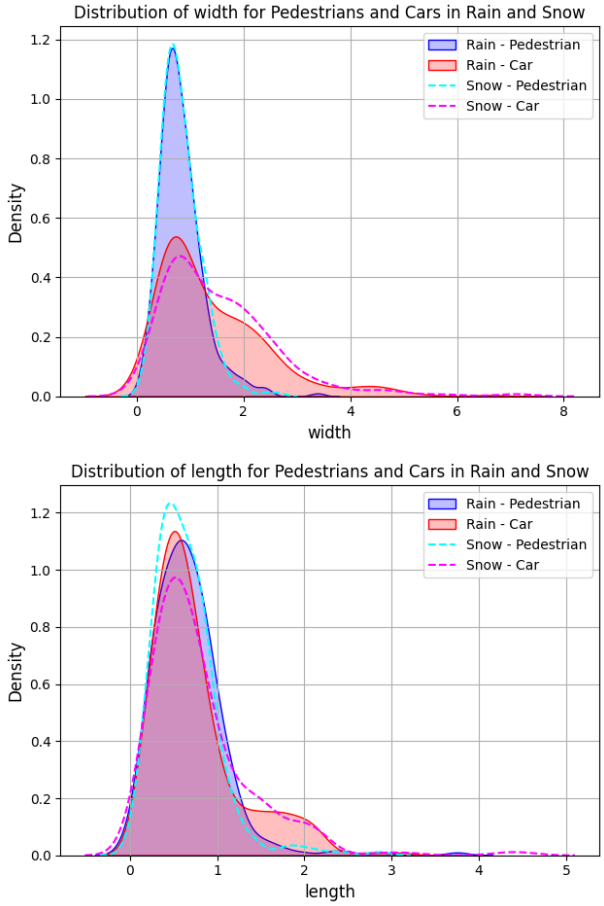

Figure 5.2: The features used for this study are mostly built with the RCS features of the cluster. To prove that the method stays consistent in performance across different weathers, aside from the metric results, the consistency in the distributions of these features is shown.

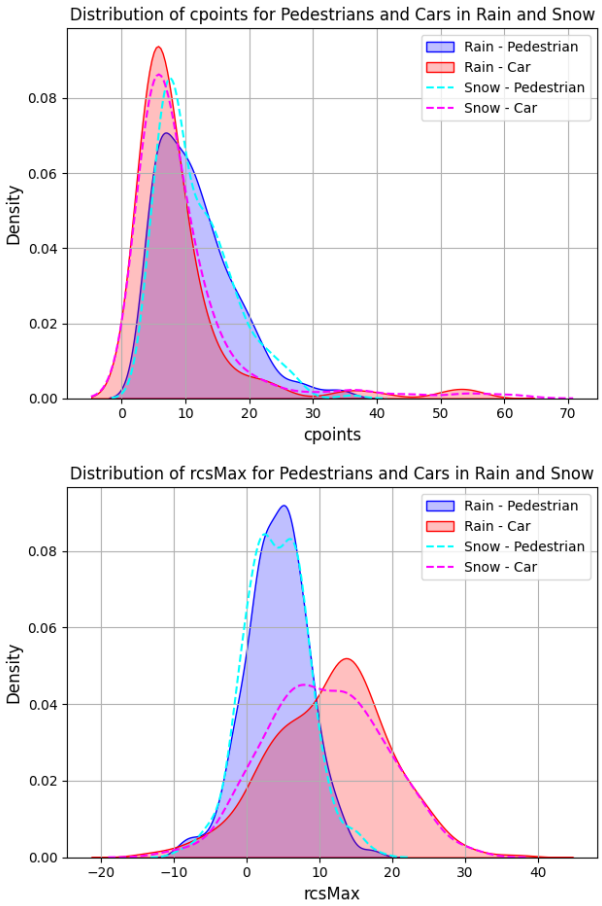

Figure 5.3: The geometrical features that emphasizes on the point density of the radar point cloud. As shown in the figure, this also shows consistency across the weathers.

Table 5.2: Random Forest without HD map features evaluation

Metric / Class	Ped.	Veh.	Noise I	Noise II	Avg
Precision	0.83	0.88	0.50	0.92	0.78
Recall	0.89	0.93	0.11	0.71	0.66
F1 Score	0.86	0.91	0.18	0.80	0.69
mAP	0.78				

5.3.1 Improvements of Radar Precision with HD map Fusion

The primary reason for the inclusion of radar including all its challenges, is its perseverance through all weather conditions. For starters, the features, Eq. (3.22), collected from the dataset, must show consistency accross different weathers. Therefore, the comparison is carried out by comparing the feature distribution in different weather conditions. Looking back into the Eq. 3.22, all of the features are built with RCS and cluster geometrical features, therefore, Figure 5.2, 5.3 compare the distribution of them under different conditions for different classes. The agreement between distributions suggests that the selected features are invariant to weather conditions. Such stability is crucial for real-world deployment, as it ensures consistent performance irrespective of weather changes.

The data collected may not show consistent frequency across different classes. For instance, in the campus ringroad, the majority of the objects are either cars, buses, or pedestrians. Seldom there are bicycles or poles and trash cans. To address this issue in the process of training the radar classifier, the SMOTE method [12] is employed to even out the distribution in the training.

To highlight the importance of HD map fusion used in this study, and its effect in the classifier accuracy, the results are presented under three different conditions. In Table 5.2, Random Forest classifier without HD map related features, e.g., p_x in Eq. (3.3), p_l in Eq. (3.13) that is the results of training $\pi(c|\chi_{C_j})$. In Table 5.3 Random Forest with HD map related features $\pi(c|\chi'_{C_j})$, where $\chi'_{C_j} = [\chi_{C_j}, p_{lj}, \delta_{Rj}, \delta_{Lj}]$, is presented. Finally, in Table 5.4 Random Forest with both the HD map related features and $\pi(c|\bar{d}_j)$ likelihoods in the Eq. (3.26) is delivered. Delving into the results, in the absence of HD map features, the

Table 5.3: Random Forest with HD map features evaluation

Metric / Class	Ped.	Veh.	Noise I	Noise II	Avg
Precision	0.94	0.95	0.86	0.89	0.91
Recall	0.93	0.93	0.67	1.00	0.88
F1 Score	0.93	0.94	0.75	0.94	0.89
mAP			0.90		

Table 5.4: Random Forest with HD map features and likelihoods evaluation

Metric / Class	Ped.	Veh.	Noise I	Noise II	Avg
Precision	0.93	0.94	0.95	0.93	0.94
Recall	0.92	0.88	0.95	0.99	0.94
F1 Score	0.93	0.91	0.95	0.96	0.93
mAP			0.94		

classifier comes to a significant challenge: it frequently misclassified Noise I for vehicles as both of them have similar RCS values. However, it was notably rectified upon the integration of HD map features. A deeper dive into this correction revealed the crucial role of the HD map: most of the misclassified metal objects were located either on the centerline or the sidewalks. Given these specific locations, the probability of encountering a vehicle is intrinsically low, a fact that the HD map data has captured. Furthermore, the results underscore the potential of radars when integrated with HD map information in known environments.

The following is an example of where the HD map features and likelihood improves the classification precision; there is a vehicle on the road with the following features as mentioned in Eq. (3.22),

$$\begin{aligned} \chi_{C_0} = [&0.31, -3.41, 6, -10, 0, 0.01, 0.42, 31.51, 0.0, \\ &0.0, 0.0077, 0.72, 0.21, 0.1, 0.31, 0.05] \\ &p_{l0} = 0, \;\; \bar{d}_0 = 1.92 \end{aligned} \tag{5.1}$$

where for this random forest output is,

$$\begin{aligned}&\pi(\text{Pedestrian}|\chi_{C_0}) = 0.46,\ \ \pi(\text{Vehicle}|\chi_{C_0}) = 0.13\\&\pi(\text{Noise I}|\chi_{C_0}) = 0.17,\ \ \pi(\text{Noise II}|\chi_{C_0}) = 0.23,\end{aligned} \tag{5.2}$$

which misclassified the Vehicle class with Pedestrian. Adding p_l, and p_x from Eq. (3.13) and replacing χ with χ',

$$\begin{aligned}&\chi'_{C_0} = [\chi_{C_0}, p_{l0}, \delta_{R0}, \delta_{L0}]\\&\pi(\text{Pedestrian}|\ \chi'_{C_0}) = 0.3, \pi(\text{Vehicle}|\ \chi'_{C_0}) = 0.24\\&\pi(\text{Noise I}|\ \chi'_{C_0}) = 0.27, \pi(\text{Noise II}|\ \chi'_{C_0}) = 0.18.\end{aligned} \tag{5.3}$$

where the classifier makes a mistake again, however, the difference in the probabilities is quite smaller. In this challenging case, knowing that the object of question is located in 1.92 m right side of the centerline adds extra information for the classifier; at this location the value for $\pi(c|d)$ for each class is,

$$\begin{aligned}&\pi(\text{Pedestrian}|d_0 = 1.92) = 0.5, \pi(\text{Vehicle}|d_0 = 1.92) = 0.98\\&\pi(\text{Noise I}|d_0 = 1.92) = 0.00, \pi(\text{Noise II}|d_0 = 1.92) = 0.00,\end{aligned} \tag{5.4}$$

referring to Eq. (3.26) with $d = 1.92$,

$$\begin{aligned}&\pi(\text{Pedestrian}|\ \chi'_{C_0}, d_0) = 0.15, \pi(\text{Vehicle}|\ \chi'_{C_0}, d_0) = 0.23\\&\pi(\text{Noise I}|\ \chi'_{C_0}, d_0) = 0.00027, \pi(\text{Noise II}|\ \chi'_{C_0}, d_0) = 0.000.\end{aligned} \tag{5.5}$$

In this example, HD map fusion plays a crucial role in accurately classifying the object under consideration. It highlights how leveraging environmental familiarity enriches the feature space for the classifier, while also optimizing memory usage and relying solely on CPU operations instead of resource-intensive GPUs. This approach demonstrates a significant advantage trended deep learning methods that rely solely on large networks. However, it's important to note that this study does not discredit the effectiveness of deep learning, though, it brings in an another perspective that in the case of operation in a known environment, an stochastic approach is more efficient. It is shown in Table 5.2, classic machine learning methods using only radar features have limitations. However, the focus is on the introduction of HD maps into radar features and the fact that it alone

improves results by 12 %, and incorporating the likelihoods of occurrences further boosts performance by 16 %. This underscores the effectiveness of HD maps when integrated into existing radar-based classification systems, showcasing substantial improvements in accuracy and reliability. These advancements are critical for enhancing object classification in dynamic environments, where precise understanding and prediction are paramount.

The last aspect of methodology is its operational efficiency in terms of computational resources and being real-time. Many methods in the literature, rely heavily on a large computing resource to make use of the radar, this study accomplishes reliable navigation using solely CPU resources. This optimization is particularly noteworthy in the context of WATonoBus [5] successfully navigating through challenging weather conditions. The absence of GPU dependency not only reduces the costs but also minimizes power consumption, an essential consideration for electric vehicles, making this method suitable for automotive-grade radar systems.

After the classification in the radar point cloud, all type **II** Noise, false alarms, is removed from the radar point cloud. The resulting point cloud, and the cluster information such as its class and the probability is passed to the next layer, which is the radar-LiDAR-camera-HD map fusion.

5.3.2 Perception Results and Comparative Study

The qualitative results, Figure 5.4, showcased in the images highlight the robust performance of the perception module under challenging weather conditions, particularly heavy snowfall. This capability is crucial for ensuring the safety and reliability of autonomous vehicles in all types of weather. It is demonstrated that even in adverse conditions, the system is able to detect and classify various objects, including pedestrians and unknown entities. In the provided visualizations, the perception module's ability to maintain situational awareness despite the occlusions and noise introduced by heavy snow is observed. The top-down view presents multiple layers of detection data. The radar and LiDAR data, indicated by the concentric circles and scattered green points that are the noise introduced by the snow respectively, show that the system is capable of penetrating through the snow, providing accurate distance and position measurements of objects around the vehicle.

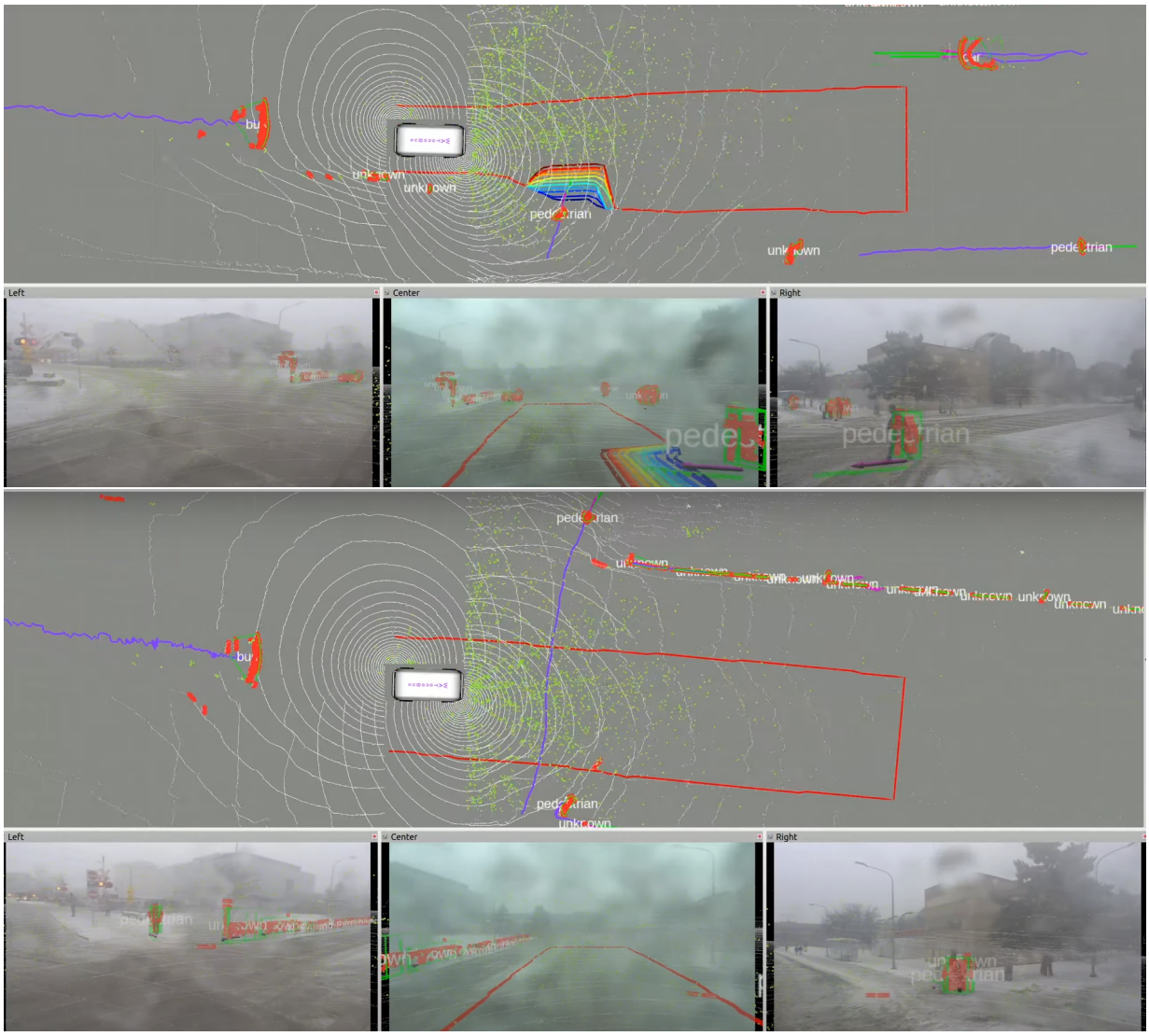

Figure 5.4: The qualitative results of the perception module. In the above, two consecutive frames are shown, and its below, there are the camera frames from front, right, and left side cameras. The purple line is the footprint of each object from tracking, and the green vector is the scale of the velocity. the red points are the objects detected by the perception module.

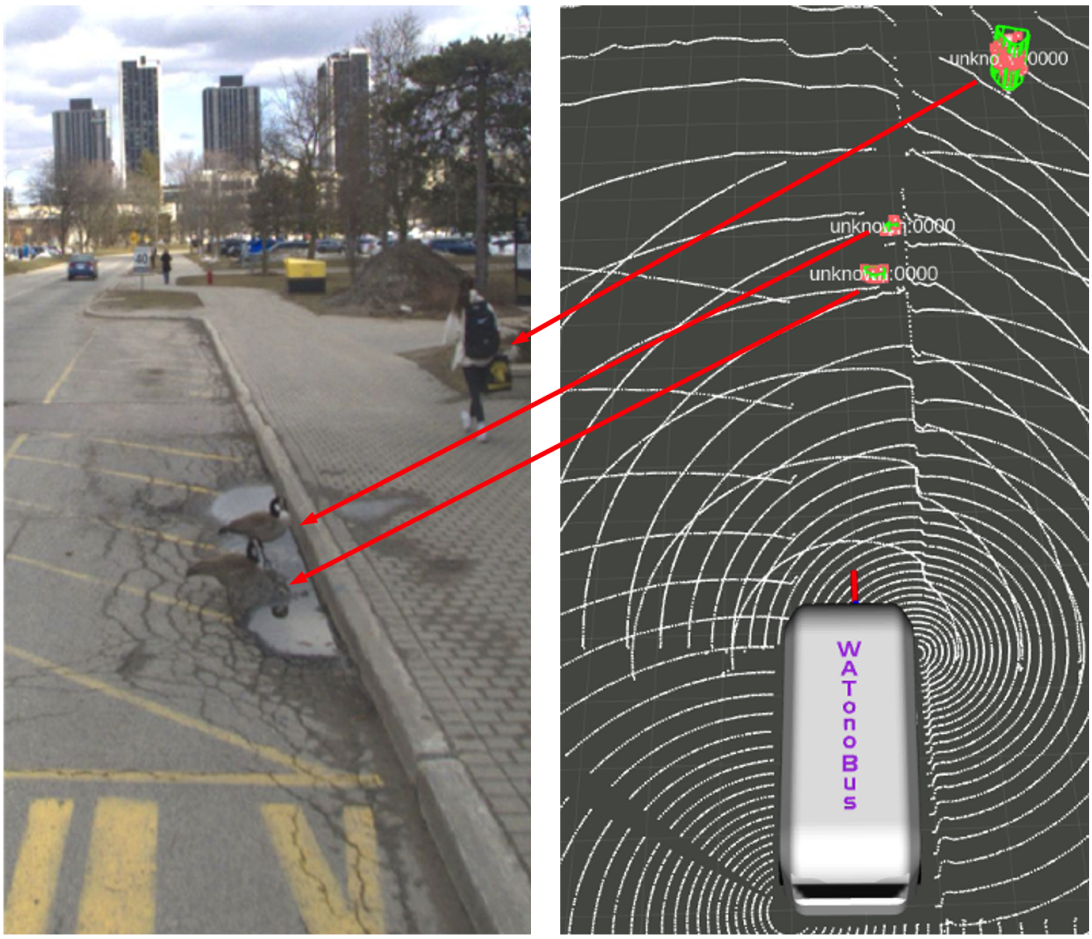

Figure 5.5: Geese detection as an example of irregular small object detection. Though the YOLO was not trained on Geese class, the module detected them, and the bus stopped for them. [5].

The images from different camera angles (left, center, right) further reinforce the module's robustness. Despite the reduced visibility due to the snow-covered lenses, the system successfully identifies and tracks objects, such as pedestrians, which are marked with bounding boxes and labels. This multi-modal approach, combining radar, LiDAR, and camera data, significantly enhances the system's resilience to weather-induced visual degradation.

One notable aspect of the perception module is the ability to classify objects with a degree of uncertainty, labeling some as "unknown." This cautious approach is beneficial in ambiguous scenarios where an object cannot be definitively identified. By doing so, appropriate measures can be taken to ensure safety, such as slowing down or stopping, until the object is cleared off the road. The integration of radar and LiDAR data plays a pivotal role in maintaining the reliability of object detection and tracking. Radar waves, unaffected by snow, provide consistent detection of objects, especially moving ones like pedestrians and vehicles. LiDAR, on the other hand, offers high-resolution 3D spatial data, which is crucial for accurate detection. The fusion of these sensor modalities compensates for the limitations of each, creating a comprehensive perception system. Moreover, it is shown that the system can handle dynamic environments. The accurate tracking of moving pedestrians, even in harsh conditions, indicates the robustness of the tracking algorithm. The long purple line that shows the footprint of the objects, emphasizing on the consistency of the detection and tracking algorithms. This capability is essential for real-time decision-making and navigation in complex urban settings, where the presence of vulnerable road users and other vehicles is common.

The perception module's performance in heavy snow conditions also underscores the effectiveness of the underlying algorithms and data processing techniques. Advanced filtering methods, sensor fusion algorithms, and machine learning models contribute to the system's ability to filter out noise and extract meaningful information from the raw sensor data. This ensures that the detections are reliable and actionable, enhancing the overall safety of the autonomous vehicle. The qualitative results demonstrate the remarkable capabilities of the perception module in detecting and tracking objects under severe weather conditions. The integration of radar, LiDAR, and camera data, along with advanced processing algorithms, enables the system to maintain high levels of accuracy and reliability.

This robustness is critical for the deployment of autonomous vehicles in real-world scenarios, ensuring safety and operational efficiency across a wide range of environmental conditions. The ability to detect and classify objects in heavy snow not only highlights the system's technical sophistication but also its readiness for practical applications in autonomous driving.

Another interesting edge case is the Figure 5.5 where it exemplifies the capability of the perception module to detect irregular small objects, such as geese, which were not part of the YOLO model's original training classes. In this instance, geese are detected as "unknown" objects by the module. Despite this, the perception system effectively identifies the geese's presence and appropriately classifies them as objects requiring attention. The module's successful detection of these small, irregularly-shaped objects highlights its adaptability and precision. The WATonoBus [5], recognizing the geese on its path, stopped to avoid a potential collision, demonstrating the system's ability to ensure safety and handle unexpected scenarios in real-world environments. This detection and response showcase the robustness of the perception system, capable of managing unforeseen obstacles and prioritizing safety even in complex urban settings.

Figure. 5.6 presents a screenshot of the ego vehicle as it approaches an intersection during heavy weather conditions. The pre-trained neural networks failed to detect the pedestrian in front of the ego vehicle. This failure was due to the dense noise points surrounding the pedestrian, which disrupted the detection patterns learned from datasets collected under good weather conditions. In contrast, the proposed method effectively distinguishes between noise and obstacle points. It successfully detects pedestrians and generates the drivable space, thereby enabling the bus to safely stop for pedestrians crossing in front. Returning to the Figure. 5.6. The pre-trained neural networks failed to detect the pedestrian in front of the ego vehicle due to that the dense noise points around the pedestrian may influence the learned detection pattern of the neural networks trained in good weather condition dataset. In contrast, the proposed method correctly distinguish the obstacle points from the noise points, and successfully detect the pedestrian, preventing a severe tragic event to happen.

To facilitate a more detailed analysis, the evaluation metrics were applied separately across various weather conditions. This approach ensures that the performance of the

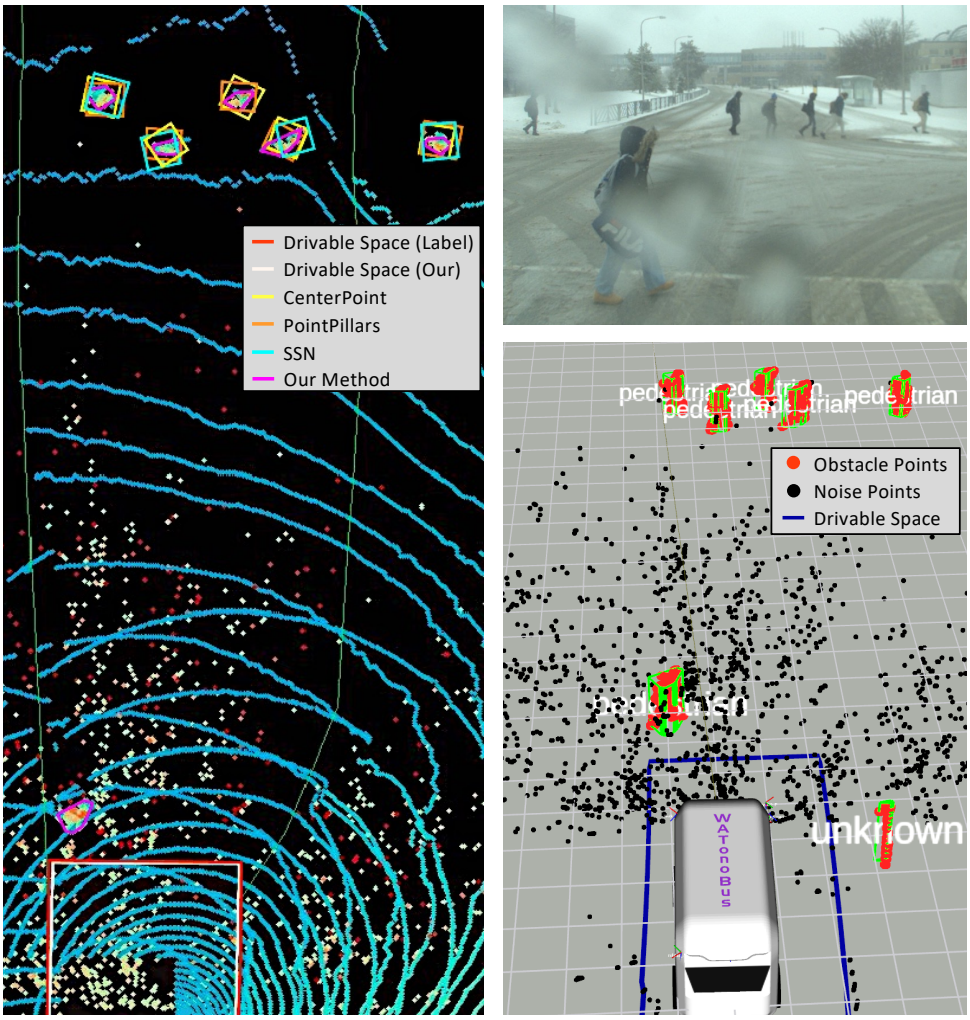

Figure 5.6: Detection performance in heavy snow conditions.

perception module can be accurately assessed under different environmental scenarios. As mentioned in Chapter 1, the primary motivation behind the development of the perception module is its reliability and high precision in object detection. The operational experience gained at MVS and during the operation of the WATonoBus highlighted that most emergency stops were triggered by either missed objects due to adverse weather conditions in Waterloo or by objects that were too small to be detected, such as geese, traffic cones, or similar items. This critical observation prompted the evolution of two essential performance metrics: Miss Rate (MR) and False Alarm Rate (FAR). These metrics are formally defined as follows:

$$MR = \frac{\text{FN}}{\text{FN} + \text{TP}} = 1 - \textbf{Recall} \tag{5.6}$$

$$FAR = \frac{\text{FP}}{\text{TP} + \text{FP}} = 1 - \textbf{Precision} \tag{5.7}$$

For a comprehensive comparative study, the performance of the proposed adaptive ground removal and adaptive DBSCAN clustering methods was benchmarked against several state-of-the-art methods, including CenterPoint [87], PointPillars [37], and SSN [95]. Our extensive experimental results demonstrate a significant enhancement in both miss rate (MR) and false alarm rate (FAR), as shown in Figure. 5.7 and summarized in Table. 5.5. The data clearly shows the superiority of our methods in improving object detection accuracy and reliability.

In the particularly challenging conditions of heavy snow and light snow, which are known to pose significant difficulties for perception systems, the proposed method achieved an impressively low MR of 0.97% and 0.56%, respectively, at a FAR of 1.24% and 0.76%. This performance is substantially better than that of the next best-performing method, CenterPoint (CP), which, at a FAR of 5%, exhibited MRs of 28.85% and 20.73%, respectively. These results are a strong indication of the superior capability of our method under adverse weather conditions, highlighting its robustness and reliability.

Moreover, the implementation of the adaptive ground removal technique plays a crucial role in enhancing the system's ability to accurately differentiate between relevant objects and irrelevant ground elements, thereby reducing the number of missed detections. Similarly, the adaptive DBSCAN clustering method improves the precision of object classification, further contributing to the reduction of false alarms. These advancements collectively

underscore the effectiveness of our approach in tackling the inherent challenges associated with autonomous vehicle perception systems, particularly in dynamic and unpredictable environments.

Table 5.5: Detection Evaluation

Test Case			HeavySnow	LightSnow	Sunny	TrafficCone
MR	FAR5%	CP	28.85	20.73	8.63	87.98
		PP	37.18	13.55	38.78	88.54
		SSN	49.17	38.37	15.56	95.36
	FAR20%	CP	18.89	11.54	7.18	35.42
		PP	30.34	11.03	9.69	70.78
		SSN	42.89	21.02	8.03	82.85
MR	Our Method		0.97	0.56	0.37	3.57
FAR	Our Method		1.24	0.76	0	0

In sunny conditions, our approach achieved an MR of 0.37% with zero false alarms, thereby highlighting the remarkable accuracy of the proposed ground removal method in segmenting almost all obstacle points correctly. In contrast, deep-learning methods encountered false alarms, often caused by inaccurate orientation estimation for objects such as cars, and missed detections, primarily due to the sparsity of points for distant objects. The geese detection cases, e.g. Figure 5.5, was particularly challenging due to irregular shape and class, where it lead to elevated MRs. The proposed method exhibited an MR of 3.57% without any false alarms, demonstrating adeptness in managing varying point density issues for the same objects at different distances. Deep-learning methods performed poorly in this scenario, attributed mainly to the point sparsity, especially concerning the geese. The proposed method demonstrated consistent superiority across various weather conditions.

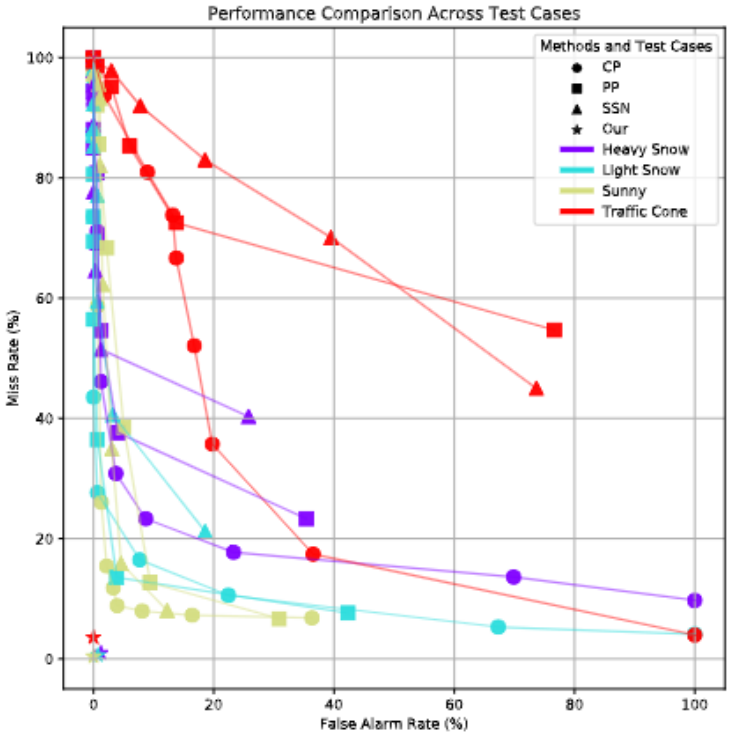

Figure 5.7: Detection performance comparison under various setups.

5.3.3 Localization Results and Comparative Study

To test the localization module, the WATonoBus [5] was driven with the localization system in the loop around the campus ring road. During the ride, the perception module actively processed the environment. Since the localization is a byproduct of the perception module, integrating it into the processing software did not introduce any significant load. The design considerations within the algorithm rendered this additional processing minimal. GPS data [5], was collected under optimal atmospheric conditions, specifically clear and sunny weather. This GPS data is designated as the ground truth due to its renowned precision in favorable conditions, serving as the reliable benchmark for assessing the performance of the localization system.

To assess the precision of the localization system, two principal benchmark evaluation metrics were selected. Firstly, the Mean Absolute Error (MAE),

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{y}_i - \hat{\mathbf{y}}i|, \tag{5.8}$$

and secondly, the Root Mean Squared Error (RMSE),

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum i = 1^{n}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}. \tag{5.9}$$

The MAE was chosen because it provides a straightforward measure of the average magnitude of the errors in the localization system, offering an easily interpretable metric that indicates the average distance between the predicted and actual values. This is particularly useful for understanding the typical performance of the system in practical terms.

The RMSE was selected as it gives more weight to larger errors due to the squaring of the differences before averaging. This property makes RMSE a more sensitive metric to outliers, thus providing a complementary perspective on the system's performance. By emphasizing larger discrepancies, RMSE helps in identifying cases where the localization system may significantly deviate from the ground truth, which is crucial for ensuring the reliability of the system in critical scenarios.

Table 5.6: RMSE and MAE calculated for our localization (EKF) and Kiss-ICP

Measure	State	EKF	KISS-ICP
RMSE	s	0.5267	1.2694
	ψ	1.8017×10^{-5}	7.2291×10^{-6}
	d	0.0502	1.0129
MAE	s	0.5967	0.9443
	ψ	0.0036	0.0024
	d	0.1661	0.7913

Together, these two criteria provide a comprehensive evaluation of the localization method's accuracy, balancing the need to understand both average performance and the impact of larger errors.

To test the approach, one of the most challenging sections of the campus ring-road, characterized by the maximum amount of $\kappa(s)$, was selected. The characteristics of the WATonoBus simplified the equations mentioned in Section 4.3.1. The double steering system of the WATonoBus enforces $\delta_f = -\delta_r$, and with the centroid approximately in the middle of the bus, $l_f \approx l_r$, making $\beta \approx 0$.

Another enforced simplification stems from the road itself. For instance, in Figure 5.12 (j), it is evident that $\partial\kappa/\partial s$ is depicted on a significantly small scale. This is because a substantial increase in $\partial\kappa/\partial s$ would render roads impassable for vehicles. Hence, it is reasonable to approximate $\partial\kappa/\partial s$ as zero. By doing so, the effects of curvature changes along the road become negligible, allowing the model to avoid introducing nonlinearities into the system. This simplification is crucial as it ensures that the equations remain linear, thereby making the computational process more efficient and the system's behavior more predictable. This linear approximation is not only practical but also aligns with the physical constraints of typical road designs, further validating its application in real-world scenarios.

The results depicted in Figures 5.8, 5.9, 5.10, and 5.11 demonstrate the efficacy of the approach in estimating both ν and $\dot{\nu}$. Additionally, these figures illustrate the improvements achieved through the modified kiss-ICP by integrating additional sensors available

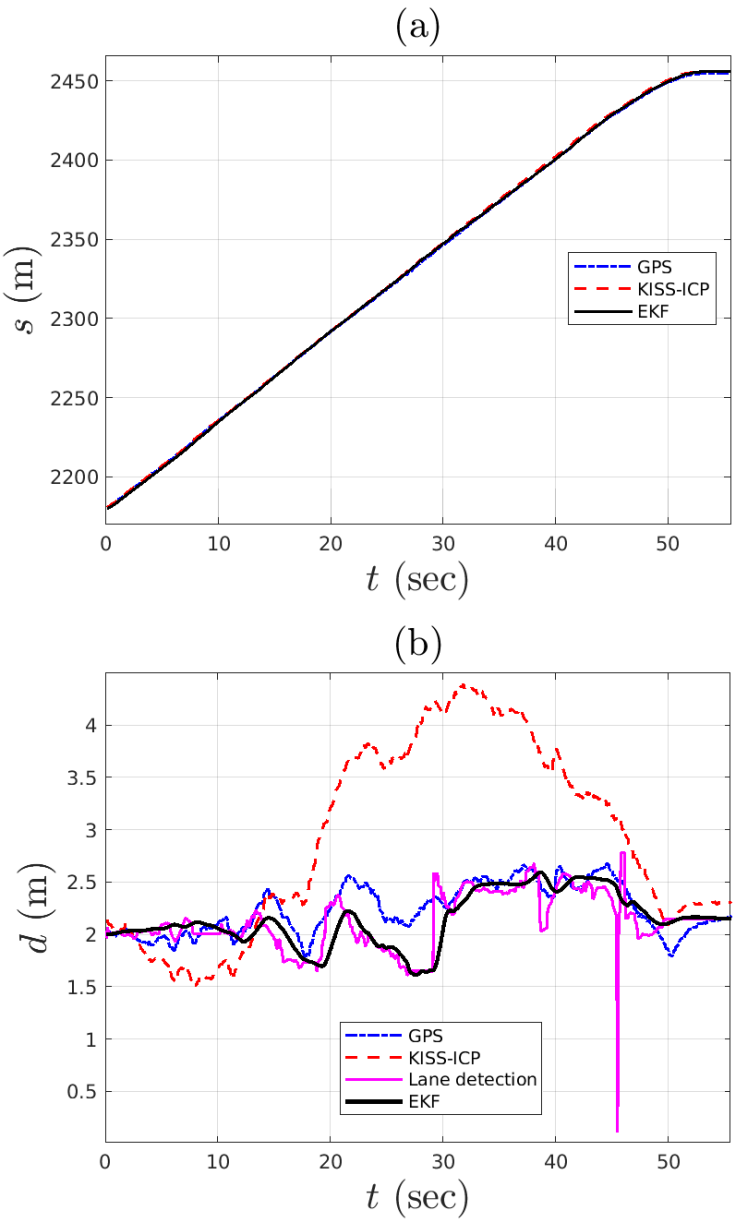

Figure 5.8: Vehicle Frenet coordinate estimate

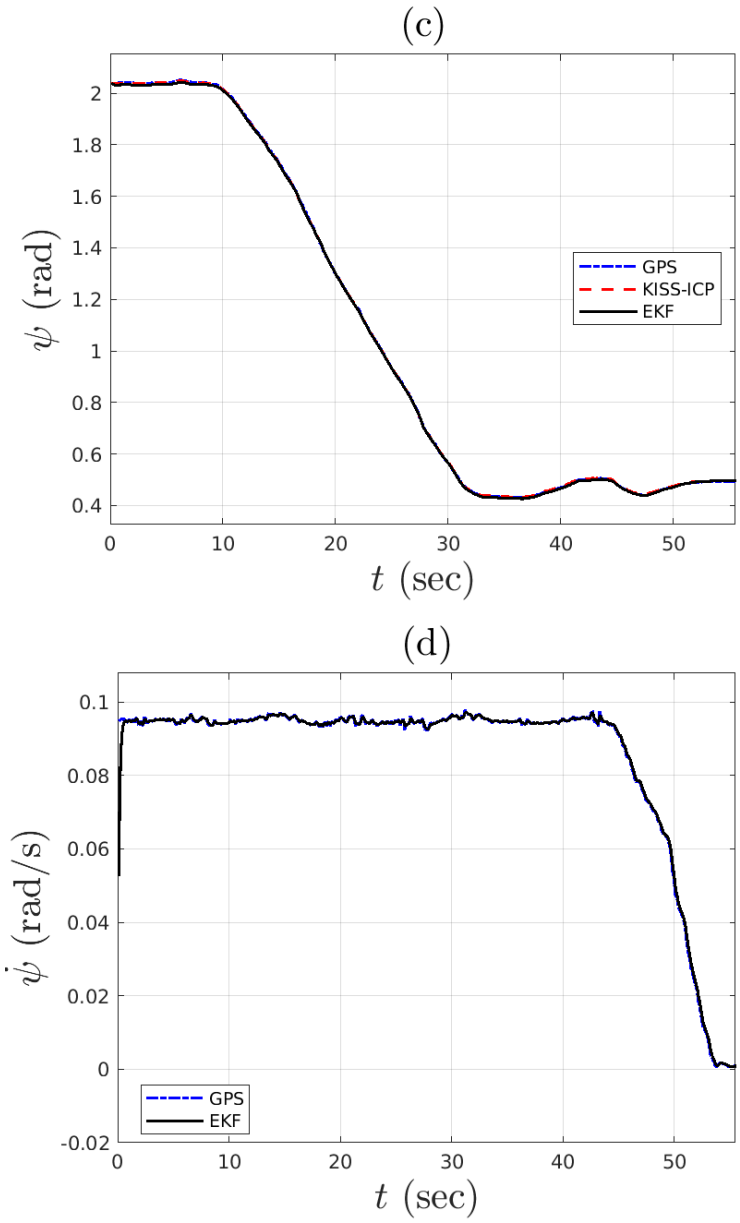

Figure 5.9: Vehicle yaw and yaw rate

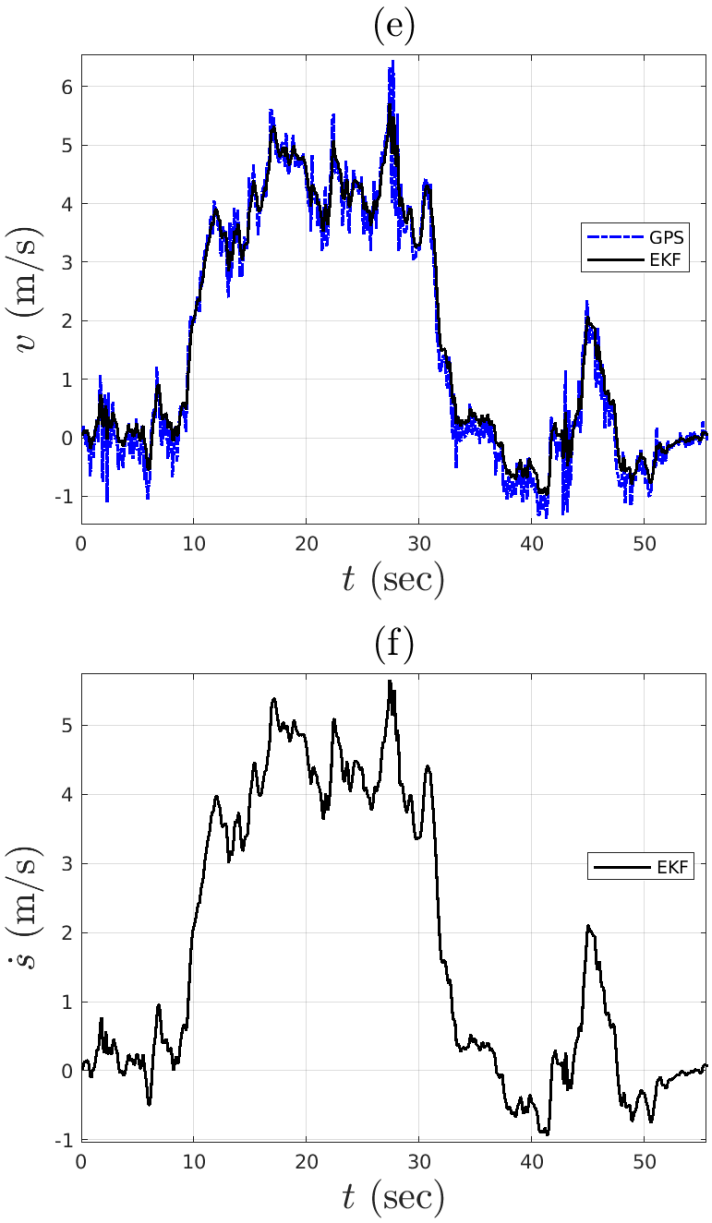

Figure 5.10: Vehicle estimated velocity and arc length rate

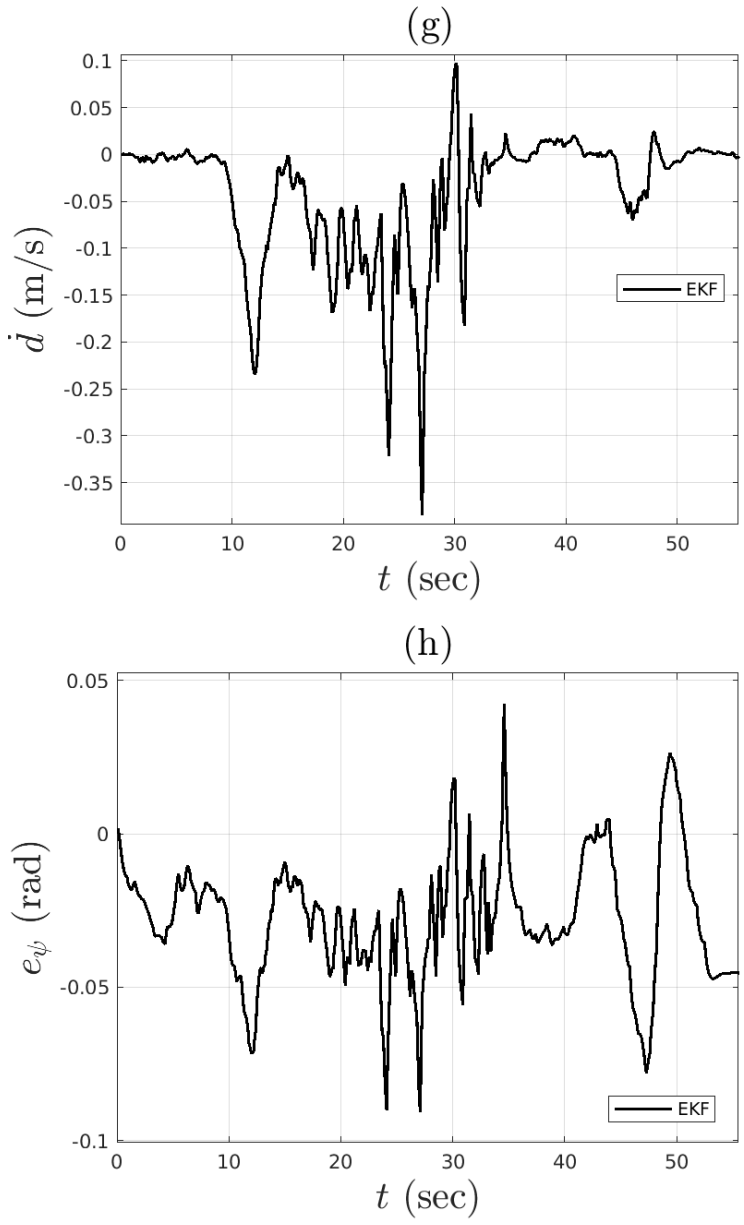

Figure 5.11: Vehicle lateral rates

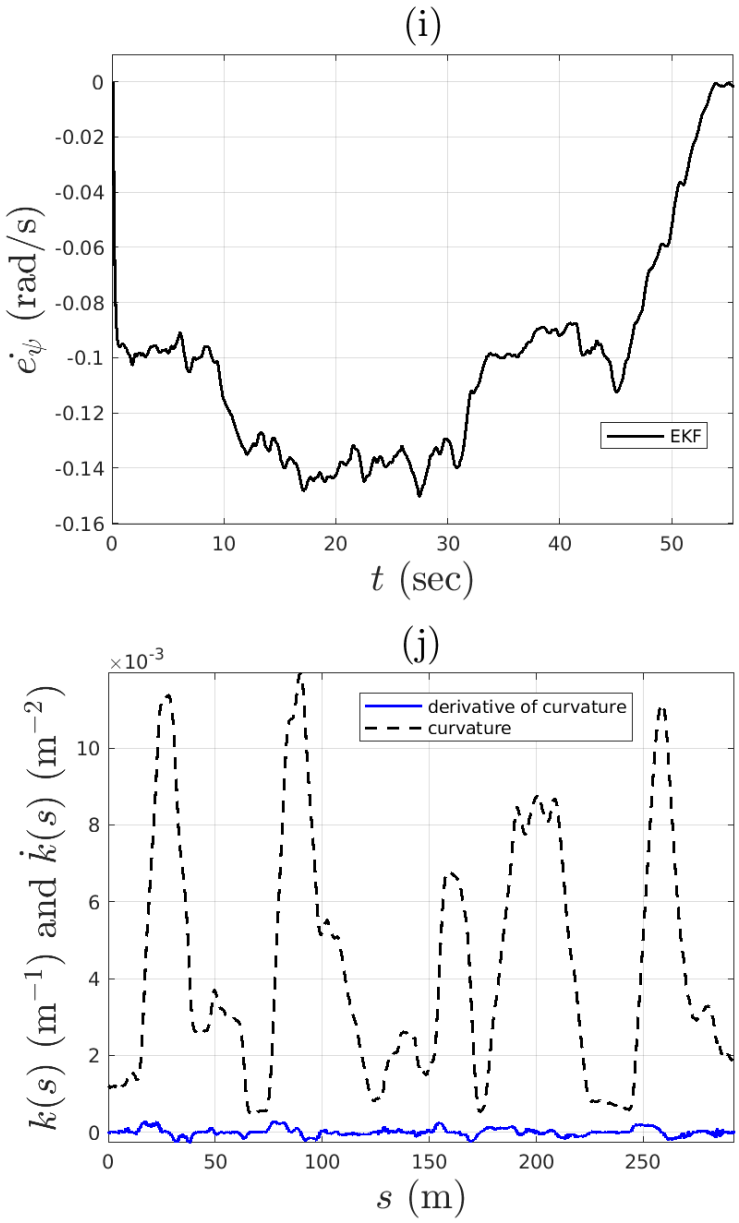

Figure 5.12: Vehicle lateral, curvature, and their rate

on the AV, including encoders, steering encoders, camera, LiDAR, and radar, as presented in Table 5.6. The integration of all available sensors is essential for enhancing the reliability of the approach, as individual sensors may fail under certain conditions. For instance, while the fusion of radar and LiDAR aids in robustly representing the 3D environment by reducing noise from rain and snow, challenges may still arise in environments with limited features, leading to inaccuracies in the ICP algorithm. Similarly, incorporating a camera for lane detection may encounter difficulties, such as failing to detect the centerline, as indicated by the spikes in Figure 5.8 (b) due to impaired vision. Including the vehicle motion model can help mitigate such situations.

Based on the findings depicted in Figure 5.8, 5.9, it appears that estimating the arc length s, yaw angle ψ, and their derivatives poses fewer challenges compared to determining the lateral position d. While the kiss-ICP method exhibits acceptable MAE and RMSE on its own, it encounters difficulties in accurately estimating the lateral position d, which is crucial for maneuvers such as lane changes or pulling over. However, incorporating camera-based lane detection and the motion model significantly enhances this estimation.

A noteworthy improvement in resource utilization is evident in this approach compared to kiss-ICP. The streamlined and effective new representation of the point cloud achieved through radar-LiDAR fusion has led to a reduction in CPU usage during the implementation of the kiss-ICP algorithm. Specifically, while the CPU usage for kiss-ICP on the LiDAR point cloud was measured at 3 cores of a 2.4 GHz Intel Core i7, the modified version only required 0.3 of 1 core for the same model. This efficiency gain is attributed to the utilization of the fused point cloud, which offers a more efficient representation of the environment compared to the dense LiDAR point cloud.

One potential critique of this methodology may center on the extensive sensory input it requires. However, it is crucial to acknowledge that the sensors utilized are standard equipment essential for the operation of an AV. For instance, an AV comes equipped with steering sensors and wheel encoders to facilitate the controller module, along with radar, LiDAR, and cameras for perception purposes. The substantial merit of this approach lies in the strategic utilization of these pre-existing modules, already integral to various AV tasks, to achieve robust localization capabilities. This integration not only optimizes the utilization of available resources but also enhances the overall efficiency and effectiveness

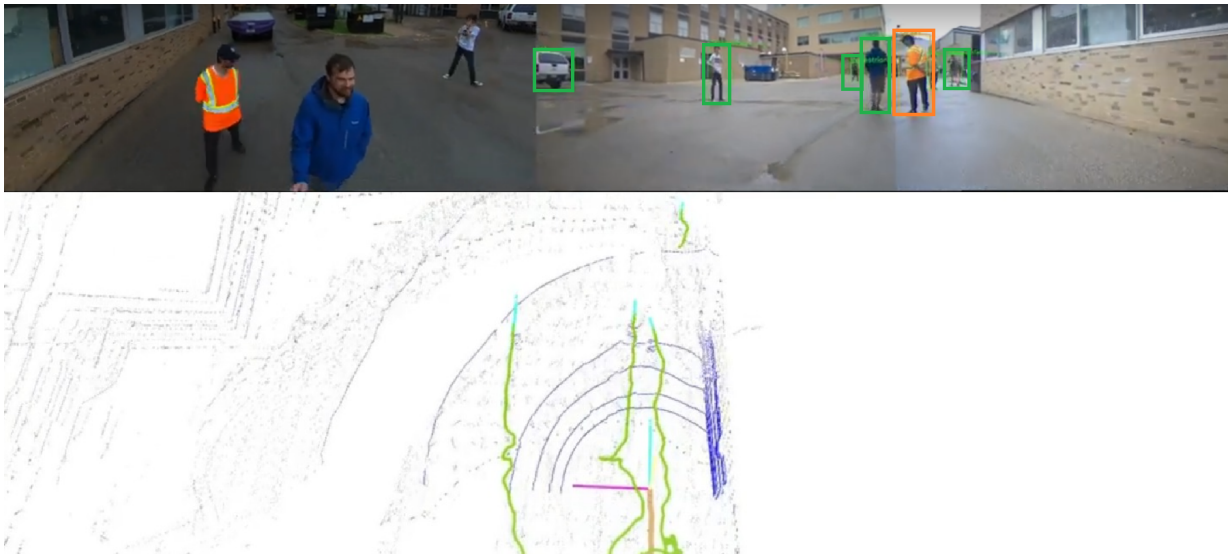

Figure 5.13: Application of the unified Perception-Localization Module in the Human Follower Software

of the vehicle's navigation system.

5.3.4 Unified Localization and Perception Computational Performance

The unified module of perception-localization is implemented in the C++ language known for its optimized performance. The compiler for the code is GNU C++ v9.1.0. The optimization flag is O2, and the nodes are interconnected using Robotic Operating System, ROS, as the middle ware. The nodes run in a cluster configured in an XML file, known as ROS launch file. On the CPU and RAM, the unified module takes 352-410 MB memory and total of 4.3 cores of ARM 2.4 GHz. On the GPU side, the VRAM used is 3.2 GB and 1232 CUDA cores. The runtime for the unified module is 66 ms which is far in margin with real-time standards.

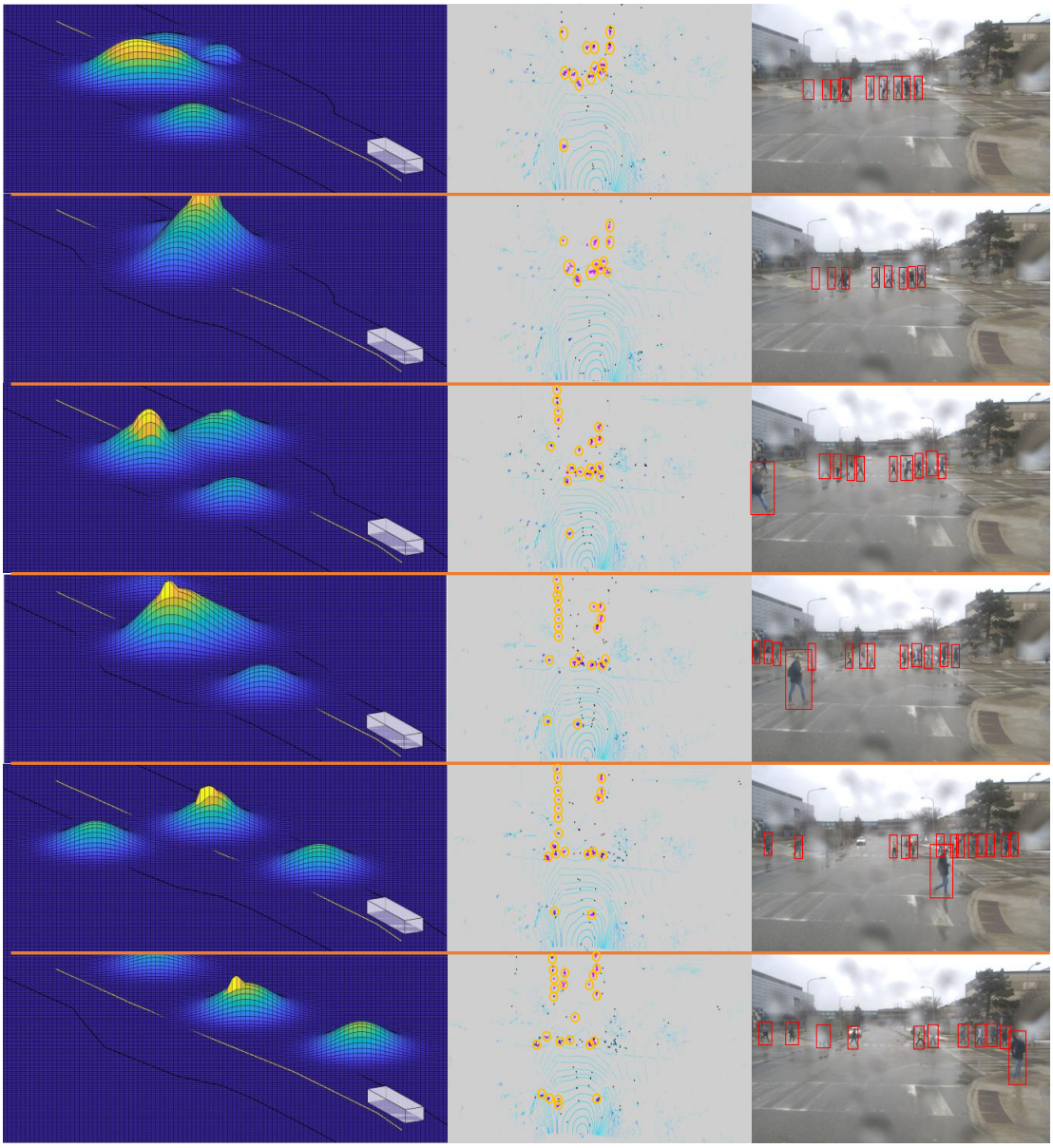

Figure 5.14: The cost-map results from a rainy day on the campus. As it can be seen the vision is completely disturbed, shown on the right, but the radar managed to create a reliable cost-map, shown on the left. The images in the middle are for the radar point cloud put on top of the LiDAR point cloud; the LiDAR points are used here to help visualization of the scene. The black dots are false alarms and clutters. The pink points are the detected objects that here are pedestrians, and metal guard rail by the road, they are better shown by the orange circles.

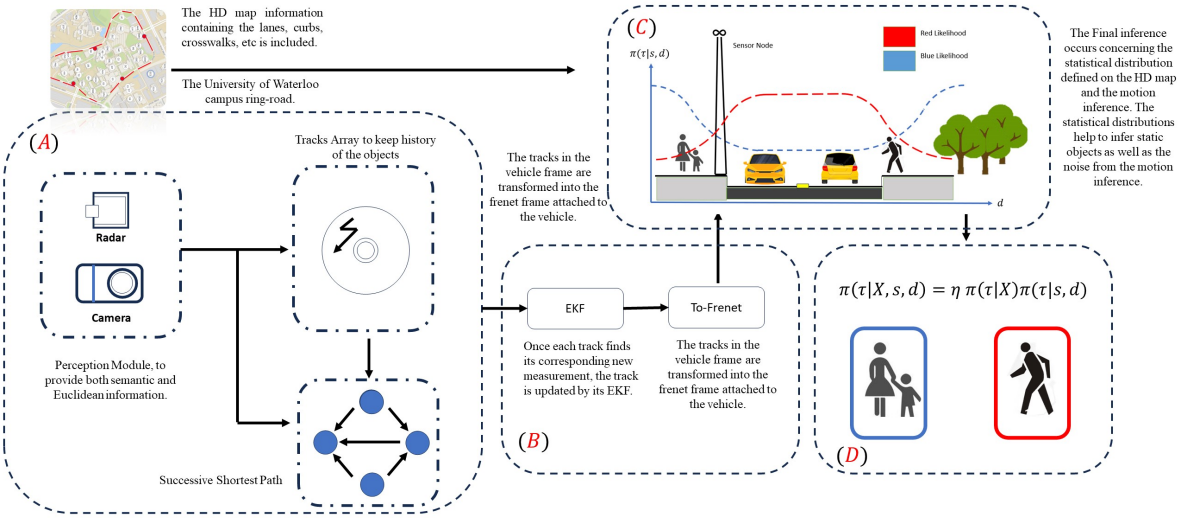

Figure 5.15: The schematics for the Red-VS-Blue approach, in which the tracking and prediction are utilized to classify the actors to those who come into interaction with the vehicle and those who doesn't.

5.4 Applications

The unified perception-localization algorithm enabled multiple applications in the areas of decision making, lane keeping, and human following. Some of the highlighted applications are:

- *Cost-Map Creation from Radar-HD map results*: The radar detections enabled a robust cost-map presented in Figure 5.15. Moreover, the cost-map also served as a smoothing layer, as it mitigates the flickering effects inherent in radar detections. The cost-map introduces a layer of continuity and establishes a safety zone, contrasting with the discrete, probability-based clusters. This distinction is crucial, as the discrete nature of radar detections and the integration of probabilistic information pose substantial challenges for path planning and control algorithms, which typically do not process such inputs. In response to this challenge, the cost-map depicted in Figure 5.15 facilitates a seamless transformation of information for the planning and control systems.

- *Game-theory Motion Planning with Perception Uncertainty and Right-of-Way Con-*

straints: Enabling a Game-theoric motion planning approach through utilizing the uncertainty developed from the perception-localization EKF tracking module. The distribution updates from the EKF is plugged into the Bayesian equilibrium aspect of the framework incorporates probabilistic beliefs and updates them based on sensor measurements, allowing the AV to make informed decisions in the presence of uncertainty in sensory system.

- *Game-theory in Practice: Application to Motion Planning and Decision Making in an Autonomous Shuttle Bus*: Enabling the motion planning through a Game-theory approach, through providing the velocity and the acceleration of other actors. The velocity and the acceleration of the other actors enables a naive prediction soley based on the motion states, that improves the motion planning algorithm safety.

- *Eyes On The Red, Watch For The Blue; An Stochastic Approach To Predict And Classify Traffic Participants*: Enabling an infrastructure-augmented algorithm for predicting traffic participant behavior. The algorithm not only produces prediction results but also identifies critical traffic participants that require the attention of autonomous vehicles. Using the velocity and the acceleration of other actors, the perception-localization module reduces the number of entities to be considered through a red-blue classification approach. The actors that are likely to interact with the vehicle are called red, while those that are less likely to interact are called blue.

- *Adaptive Cruising For WATonoBus*: The accurate velocity estimation from the model, enable the WATonoBus for an adaptive lane keeping module. While keeping the lane, it is possible for other objects to be in the same lane as the bus. To enable an adaptive cruising application, knowing the velocity of the leading vehicle, helped the WATonoBus in keeping a safe distance from the front vehicle, while keeping the lane.

- *Predictive Human Following*: The accurate object detection and tracking, enabled the development of a human following module. In this module, the detected human position and its velocity, helps the vehicle to follow the human, while maintaining a safe distance. Furthermore, knowing the velocity and acceleration of the human,

the module is able to predict the future trajectory and therefore, enabling an agile following module.

Chapter 6

Conclusion

6.1 Conclusion

In this study, a novel unified perception-localization method was presented. Initially the radar point cloud was improved with the HD map fusion, and the significant effect of HD map fusion was demonstrated through an ablation study. Subsequently, the challenges related to irregularly shaped objects were addressed by employing HD maps and adaptive ground removal in the LiDAR point cloud. Additionally, radar-LiDAR fusion and the adaptive DBSCAN approach effectively mitigated noise introduced by snow and rain in inclement weather. The overall perception module exhibited robust performance, as evidenced by qualitative results showing the vehicle operation in heavy snow and a comparative study highlighting the module's performance with respect to the state-of-the-art solutions.

The processed perception module output was then utilized to formulate a novel localization algorithm, which outperformed state-of-the-art methods such as Kiss-ICP by integrating data from vehicle wheel sensors. A comprehensive study revealed that the unified perception-localization algorithm is both effective and efficient in resource consumption, optimally utilizing all common sensors on an autonomous vehicle.

6.2 Future Work

The following future works are suggested as extensions for this study:

- Enhanced Clustering for Radar Point Clouds: The current clustering approach considers only scaled velocity and Cartesian positions. However, points belonging to the same object typically exhibit consistent Radar Cross Section (RCS) values. The range of RCS values differs from Cartesian coordinates and velocity, necessitating a more detailed study. This can be achieved through an adaptive search radius for the DBSCAN algorithm, tailored specifically for radar data.

- Advanced Statistical Fusion: The statistical fusion between radar point clouds and HD maps could be enhanced by incorporating arc length as an additional feature. Combining arc length with lateral distance can generate a heat map for different classes of objects, enabling more sophisticated fusion techniques such as convolutional neural networks.

- Utilizing RCS for Orientation and Velocity Estimation: Observations from this study indicate that RCS values vary with object orientation relative to the radar. This feature can be leveraged for improved orientation and velocity estimation. Creating a heatmap of RCS values and employing the association algorithm used in this study can produce a richer feature set for a fully convolutional network to estimate object orientation.

- Radar-Specific Map Creation for Localization: The localization algorithm's computation and accuracy can be significantly enhanced with the creation of a radar-specific map. This map could capture RCS values as indicators of radar reflectors in the environment, providing unique signatures to address the kidnapping problem in localization.

- Integrated Calibration of Radar, Camera, and LiDAR: The calibration process for radar, camera, and LiDAR can be streamlined using a specialized checkerboard with corner reflectors. The checkerboard's plane and pattern can be used for LiDAR

and camera calibration, while corner reflectors at the board's corners can be utilized for radar calibration. Plane estimation can then be formulated as an optimization problem to determine the calibration parameters.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Rafael Arnay, Nestor Morales, Antonio Morell, Javier Hernandez-Aceituno, Daniel Perea, Jonay T. Toledo, Alberto Hamilton, Javier J. Sanchez-Medina, and Leopoldo Acosta. Safe and reliable path planning for the autonomous vehicle verdino. *IEEE Intelligent Transportation Systems Magazine*, 8(2):22–32, 2016.

[3] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019.

[4] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine vision and applications*, 25(3):727–745, 2014.

[5] Neel P. Bhatt, Ruihe Zhang, Minghao Ning, Ahmad Reza Alghooneh, Joseph Sun, Pouya Panahandeh, Ehsan Mohammadbagher, Ted Ecclestone, Ben MacCallum, Ehsan Hashemi, and Amir Khajepour. Watonobus: An all weather autonomous shuttle. *arxiv*, 2023.

[6] Mattias Brännström, Erik Coelingh, and Jonas Sjöberg. Model-based threat assessment for avoiding arbitrary vehicle collisions. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):658–669, 2010.

[7] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[8] Lukas Brunke. Learning model predictive control for competitive autonomous racing. *arXiv preprint arXiv:2005.00826*, 2020.

[9] Alexander Buyval, Aidar Gabdullin, Konstantin Sozykin, and Alexandr Klimchik. Model predictive path integral control for car driving with autogenerated cost map based on prior map and camera image. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2109–2114, 2019.

[10] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[11] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[13] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6526–6534, 2017.

[14] Y-T Chiu, S Srinara, M-L Tsai, and K-W Chiang. Improvement of lidar-slam-based 3d ndt localization using fault detection and exclusion algorithm. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43:189–195, 2022.

[15] Jian Deng and Krzysztof Czarnecki. Mlod: A multi-view 3d object detection based on robust feature fusion method. In *2019 IEEE intelligent transportation systems conference (ITSC)*, pages 279–284. IEEE, 2019.

[16] Tianchen Deng, Hongle Xie, Jingchuan Wang, and Weidong Chen. Long-term visual simultaneous localization and mapping: Using a bayesian persistence filter-based global map prediction. *IEEE Robotics & Automation Magazine*, 30(1):36–49, 2023.

[17] Wenjie Ding, Limeng Qiao, Xi Qiu, and Chi Zhang. Pivotnet: Vectorized pivot learning for end-to-end hd map construction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3672–3682, October 2023.

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[19] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[20] Jin Fang, Dingfu Zhou, Xibin Song, and Liangjun Zhang. Mapfusion: A general framework for 3d object detection with hdmaps. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 3406–3413. IEEE Press, 2021.

[21] Di Feng, Ali Harakeh, Steven L Waslander, and Klaus Dietmayer. A review and comparative study on probabilistic object detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):9961–9980, 2021.

[22] Taisei Fujimoto, Satoshi Tanaka, and Shinpei Kato. Lanefusion: 3d object detection with rasterized lane map. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 396–403, 2022.

[23] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[24] Alberto Y Hata and Denis F Wolf. Feature detection for vehicle localization in urban environments using a multilayer lidar. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):420–429, 2015.

[25] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Epnet: Enhancing point features with image semantics for 3d object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 35–52. Springer, 2020.

[26] Jinyong Jeong, Younggun Cho, and Ayoung Kim. Hdmi-loc: Exploiting high definition map image for precise localization via bitwise particle filter. *IEEE Robotics and Automation Letters*, 5(4):6310–6317, 2020.

[27] Shihao Ji, S. Vishwanathan, Nadathur Satish, Michael Anderson, and Pradeep Dubey. Blackout: Speeding up recurrent neural network language models with very large vocabularies. 11 2015.

[28] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[29] Phillip Karle, Maximilian Geisslinger, Johannes Betz, and Markus Lienkamp. Scenario understanding and motion prediction for autonomous vehicles—review and comparison. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):16962–16982, 2022.

[30] Aino Keitaanniemi, Petri Rönnholm, Antero Kukko, and Matti T Vaaja. Drift analysis and sectional post-processing of indoor simultaneous localization and mapping (slam)-based laser scanning data. *Automation in Construction*, 147:104700, 2023.

[31] Dominik Kellner, Jens Klappstein, and Klaus Dietmayer. Grid-based dbscan for clustering extended objects in radar data. In *2012 IEEE Intelligent Vehicles Symposium*, pages 365–370, 2012.

[32] Eugene F. Knott, John F. Shaeffer, and Michael T. Tuley. *Radar Cross Section*. SciTech Publishing, Raleigh, NC, 2nd edition, 1993.

[33] Jennifer Kreklow, Björn Tetzlaff, Benjamin Burkhard, and Gerald Kuhnt. Radar-based precipitation climatology in germany—developments, uncertainties and potentials. *Atmosphere*, 11(2):217, 2020.

[34] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[35] Sampo Kuutti, Saber Fallah, Konstantinos Katsaros, Mehrdad Dianati, Francis Mccullough, and Alexandros Mouzakitis. A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. *IEEE Internet of Things Journal*, 5(2):829–846, 2018.

[36] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, 2019.

[37] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[38] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *2010 IEEE international conference on robotics and automation*, pages 4372–4378. IEEE, 2010.

[39] Yicheng Li, Feng Feng, Yingfeng Cai, Zhixiong Li, and Miguel Angel Sotelo. Localization for intelligent vehicles in underground car parks based on semantic information. *IEEE Transactions on Intelligent Transportation Systems*, 25(2):1317–1332, 2024.

[40] You Li and Javier Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, 2020.

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[42] Hengbo Ma, Yaofeng Sun, Jiachen Li, and Masayoshi Tomizuka. Multi-agent driving behavior prediction across different scenarios with self-supervised domain knowledge. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3122–3129. IEEE, 2021.

[43] Enrique Marti, Miguel Angel de Miguel, Fernando Garcia, and Joshue Perez. A review of sensor technologies for perception in automated driving. *IEEE Intelligent Transportation Systems Magazine*, 11(4):94–108, 2019.

[44] Kaouther Messaoud, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi. Relational recurrent neural networks for vehicle trajectory prediction. In *2019 IEEE intelligent transportation systems conference (ITSC)*, pages 1813–1818. IEEE, 2019.

[45] Xiaoyu Mo, Yang Xing, and Chen Lv. Graph and recurrent neural network-based vehicle trajectory prediction for highway driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1934–1939. IEEE, 2021.

[46] Liz Murphy, Steven Martin, and Peter Corke. Creating and using probabilistic costmaps from vehicle experience. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4689–4694. IEEE, 2012.

[47] Liz Murphy and Paul Newman. Risky planning on probabilistic costmaps for path planning in outdoor environments. *IEEE Transactions on Robotics*, 29(2):445–457, 2012.

[48] Ramin Nabati and H Qi. Centerfusion: Center-based radar and camera fusion for 3d object detection. *IEEE Conference on Applications of Computer Vision (WACV)*, 2020.

[49] Ramin Nabati and Hairong Qi. Rrpn: Radar region proposal network for object detection in autonomous vehicles. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3093–3097, 2019.

[50] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and M Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. *IEEE Conference on Sensor Data Fusion*, 2019.

[51] Felix Nobis, E. Shafiei, Phillip Karle, Johannes Betz, and M. Lienkamp. Radar voxel fusion for 3d object detection. *MDPI*, 2021.

[52] Barrett O'Neill. *Elementary Differential Geometry*. Elsevier/Academic Press, 2 edition, 2006.

[53] Andras Palffy, Ewoud A. I. Pool, Srimannarayana Baratam, Julian F. P. Kooij, and D. Gavrila. Multi-class road user detection with 3+1d radar in the view-of-delft dataset. *IEEE Robotics and Automation Letters*, 2022.

[54] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10386–10393. IEEE, 2020.

[55] Ignacio Parra, Miguel Angel Sotelo, David Fernández Llorca, and Manuel Ocaña. Robust visual odometry for vehicle localization in urban environments. *Robotica*, 28(3):441–452, 2010.

[56] Jan-Hendrik Pauls, Kürsat Petek, Fabian Poggenhans, and Christoph Stiller. Monocular localization in hd maps by combining semantic segmentation and distance transform. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4595–4601, 2020.

[57] Ying Peng, Yechen Qin, Xiaolin Tang, Zhiqiang Zhang, and Lei Deng. Survey on image and point-cloud fusion-based object detection in autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):22772–22789, 2022.

[58] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[59] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[60] Louis L. Scharf. *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison Wesley, New York, NY, 1991.

[61] Dan Shen, Yaobin Chen, Lingxi Li, and Stanley Chien. Collision-free path planning for automated vehicles risk assessment via predictive occupancy map. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 985–991. IEEE, 2020.

[62] Pengcheng Shi, Zhikai Zhu, Shiying Sun, Xiaoguang Zhao, and Min Tan. Invariant extended kalman filtering for tightly coupled lidar-inertial odometry and mapping. *IEEE/ASME Transactions on Mechatronics*, 2023.

[63] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019.

[64] Keqi Shu, Ngc-Dũng ào, Weisen Shi, and Amir Khajepour. Group frenet frame cav path planning on highways. *IEEE Internet of Things Journal*, 2023.

[65] Keqi Shu, Reza Valiollahi Mehrizi, Shen Li, Mohammad Pirani, and Amir Khajepour. Human inspired autonomous intersection handling using game theory. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[66] Keqi Shu, Huilong Yu, Xingxin Chen, Long Chen, Qi Wang, Li Li, and Dongpu Cao. Autonomous driving at intersections: A critical-turning-point approach for left turns. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[67] Keqi Shu, Huilong Yu, Xingxin Chen, Shen Li, Long Chen, Qi Wang, Li Li, and Dongpu Cao. Autonomous driving at intersections: A behavior-oriented critical-turning-point approach for decision making. *IEEE/ASME Transactions on Mechatronics*, 27(1):234–244, 2021.

[68] Merrill Ivan Skolnik. *Radar Handbook*. McGraw-Hill Education, 3rd edition, 2008. Chapter 21: FMCW Radar Fundamentals and Equations.

[69] John Smith and Sarah Johnson. Understanding the universal transverse mercator frame in gps. *GPS World*, 25(2):45–52, 2018.

[70] Jae Kyu Suhr, Jeungin Jang, Daehong Min, and Ho Gi Jung. Sensor fusion-based low-cost vehicle localization system for complex urban environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1078–1086, 2016.

[71] Chen Sun, Yaodong Cui, Yukun Lu, Yifeng Cao, Dongpu Cao, and Amir Khajepour. Robust learning for autonomous driving perception tasks in cyber-physical-social systems. In *2023 IEEE 3rd International Conference on Digital Twins and Parallel Intelligence (DTPI)*, pages 1–7. IEEE, 2023.

[72] Chen Sun, Cong Wang, Zejian Deng, and Dongpu Cao. Dimensionless model-based system tracking via augmented kalman filter for multiscale unmanned ground vehicles. *IEEE/ASME Transactions on Mechatronics*, 26(2):600–610, 2020.

[73] Chen Sun, Ruihe Zhang, Yukun Lu, Yaodong Cui, Zejian Deng, Dongpu Cao, and Amir Khajepour. Toward ensuring safety for autonomous driving perception: Standardization progress, research advances, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[74] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[75] DE Vakmann. *Sophisticated signals and the uncertainty principle in radar*, volume 4. Springer Science & Business Media, 2012.

[76] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp – simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 8(2):1029–1036, 2023.

[77] Gregory K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[78] Huayou Wang, Changliang Xue, Yanxing Zhou, Feng Wen, and Hongbo Zhang. Visual semantic localization based on hd map for autonomous vehicles in urban scenarios. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11255–11261, 2021.

[79] Sufang Wang and Weicun Zhang. Slam algorithms for autonomous mobile robots. In *Modeling, Identification, and Control for Cyber-Physical Systems Towards Industry 4.0*, pages 115–134. Elsevier, 2024.

[80] Yusheng Wang, Yidong Lou, Weiwei Song, Bing Zhan, Feihuang Xia, and Qigeng Duan. A lidar-inertial slam tightly-coupled with dropout-tolerant gnss fusion for autonomous mine service vehicles. *IEEE Transactions on Instrumentation and Measurement*, 2023.

[81] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1742–1749. IEEE, 2019.

[82] Erik Ward and John Folkesson. Vehicle localization with low cost radar sensors. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 864–870. IEEE, 2016.

[83] Guotao Xie, Hongbo Gao, Lijun Qian, Bin Huang, Keqiang Li, and Jianqiang Wang. Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models. *IEEE Transactions on Industrial Electronics*, 65(7):5999–6008, 2017.

[84] Liang Xie, Chao Xiang, Zhengxu Yu, Guodong Xu, Zheng Yang, Deng Cai, and Xiaofei He. Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12460–12467, 2020.

[85] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–253, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[86] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 29–31 Oct 2018.

[87] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[88] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2019.

[89] Ao Zhang, F. Nowruzi, and R. Laganière. Raddet: Range-azimuth-doppler based radar object detection for dynamic road users. *IEEE Conference on Robots and Vision (CRV)*, 2021.

[90] Guoqiang Zhang, Hao Li, and Fabian Wenger. Object detection and 3d estimation via an fmcw radar using a fully convolutional network. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[91] Yujia Zhang, Jungwon Kang, and Gunho Sohn. Pvl-cartographer: Panoramic vision-aided lidar cartographer-based slam for maverick mobile mapping system. *Remote Sensing*, 15(13):3383, 2023.

[92] Yuxiao Zhang, Alexander Carballo, Hanting Yang, and Kazuya Takeda. Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:146–177, 2023.

[93] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[94] Hao Zhu, Mengyin Fu, Yi Yang, Xinyu Wang, and Meiling Wang. A path planning algorithm based on fusing lane and obstacle map. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1442–1448, 2014.

[95] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 581–597. Springer, 2020.

APPENDICES

Appendix A

Technical Concerns Regarding Radar

A.1 Automotive Radars

Automotive radars have become a critical component in the development of advanced driver-assistance systems (ADAS) and autonomous vehicles. These radars play a vital role in enhancing vehicle safety, improving navigation, and enabling various autonomous functionalities. This section provides an overview of automotive radars, their operating principles, applications, and future prospects.

Automotive radars operate by transmitting radio waves and analyzing the reflections from objects in the environment. The basic principle involves emitting a signal that bounces off targets and returns to the radar sensor. The time delay between transmission and reception, along with the frequency shift (Doppler effect), allows the radar to determine the distance, velocity, and angle of objects. This information is crucial for real-time decision-making and collision avoidance.

Radars typically operate in the millimeter-wave frequency bands, such as 24 GHz and 77 GHz. The 77 GHz band is preferred for automotive applications due to its higher resolution and shorter wavelength, which allows for more precise detection of objects. The radar system includes components like the transmitter, receiver, signal processor, and antenna array, which work together to detect and track objects.

Automotive radars are integral to various ADAS features and autonomous driving technologies. Some key applications include:

- **Adaptive Cruise Control (ACC):** Radars monitor the distance and relative speed of vehicles ahead, allowing the system to maintain a safe following distance by automatically adjusting the vehicle's speed.
- **Collision Avoidance Systems:** Radars detect potential obstacles and alert the driver or autonomously apply the brakes to prevent collisions.
- **Blind Spot Detection (BSD):** Radars scan the areas alongside and behind the vehicle to identify vehicles in blind spots, warning the driver during lane changes.
- **Cross Traffic Alert (CTA):** Rear radars detect approaching vehicles from the sides when reversing out of parking spaces, alerting the driver to potential hazards.
- **Lane Change Assist (LCA):** Radars monitor adjacent lanes to ensure safe lane changes by detecting fast-approaching vehicles.

A.2 Installation Concerns and Calibration process

Automotive radars have become a critical component in the development of advanced driver-assistance systems (ADAS) and autonomous vehicles. These radars play a vital role in enhancing vehicle safety, improving navigation, and enabling various autonomous functionalities. This section provides an overview of automotive radars, their operating principles, applications, and future prospects.

A.2.1 Operating Principles

Automotive radars operate by transmitting radio waves and analyzing the reflections from objects in the environment. The basic principle involves emitting a signal that bounces off targets and returns to the radar sensor. The time delay between transmission and reception,

along with the frequency shift (Doppler effect), allows the radar to determine the distance, velocity, and angle of objects. This information is crucial for real-time decision-making and collision avoidance.

Radars typically operate in the millimeter-wave frequency bands, such as 24 GHz and 77 GHz. The 77 GHz band is preferred for automotive applications due to its higher resolution and shorter wavelength, which allows for more precise detection of objects. The radar system includes components like the transmitter, receiver, signal processor, and antenna array, which work together to detect and track objects.

A.2.2 Applications in ADAS and Autonomous Vehicles

Automotive radars are integral to various ADAS features and autonomous driving technologies. Some key applications include:

- **Adaptive Cruise Control (ACC):** Radars monitor the distance and relative speed of vehicles ahead, allowing the system to maintain a safe following distance by automatically adjusting the vehicle's speed.
- **Collision Avoidance Systems:** Radars detect potential obstacles and alert the driver or autonomously apply the brakes to prevent collisions.
- **Blind Spot Detection (BSD):** Radars scan the areas alongside and behind the vehicle to identify vehicles in blind spots, warning the driver during lane changes.
- **Cross Traffic Alert (CTA):** Rear radars detect approaching vehicles from the sides when reversing out of parking spaces, alerting the driver to potential hazards.
- **Lane Change Assist (LCA):** Radars monitor adjacent lanes to ensure safe lane changes by detecting fast-approaching vehicles.

A.3 Installation Concerns and Calibration Process

The successful implementation of automotive radar systems depends significantly on proper installation and precise calibration. These steps are crucial to ensure the radar's optimal performance, accuracy, and reliability. This section discusses the primary concerns associated with installing automotive radars and outlines the calibration process necessary to maintain system integrity.

A.3.1 Installation Concerns

1. Placement and Alignment: The positioning of radar sensors on the vehicle is critical. Radars are typically installed in the front grille, bumpers, or rear of the vehicle, depending on their intended function (e.g., adaptive cruise control, blind spot detection). The placement should ensure an unobstructed field of view to prevent any interference from vehicle components or external elements. Incorrect alignment can lead to inaccurate measurements and reduced system effectiveness.

2. Mounting Stability: The stability of the radar sensor mount is another essential consideration. Sensors must be securely fastened to minimize vibrations and movements that could affect their readings. Vibration dampening materials and robust mounting brackets are often used to maintain the stability of the radar unit.

3. Environmental Factors: Radar sensors must be designed and installed to withstand various environmental conditions, including extreme temperatures, moisture, dirt, and debris. Protective housings and coatings can help shield the sensors from these elements, ensuring long-term durability and performance.

4. Electromagnetic Interference (EMI): Automotive radars can be susceptible to electromagnetic interference from other electronic systems within the vehicle or from external sources. Careful consideration of the sensor's placement relative to other electronic components and the use of EMI shielding can mitigate these issues.

5. Legal and Regulatory Compliance: Radar installations must comply with regional regulations regarding electromagnetic emissions and automotive safety standards.

Ensuring compliance with these regulations is essential for both legal operation and the safety of the vehicle's occupants and other road users.

A.3.2 Calibration Process

Calibration of radar and LiDAR systems using a corner reflector is a precise and widely-used method to ensure accuracy in sensor measurements. A corner reflector, typically made of three mutually perpendicular metal plates forming a trihedral structure, provides a strong, consistent, and well-defined reflective surface for electromagnetic and laser signals. During calibration, the corner reflector is strategically placed within the sensor's field of view to create a known reference point for aligning the radar and LiDAR outputs. This process helps correct systematic errors, synchronize coordinate systems, and verify the accuracy of the sensor's range, angle, and intensity measurements. Calibration with corner reflectors is essential in applications like autonomous vehicles, where sensor fusion and high-precision object detection are critical.

A.4 Radar models used in this study

The two radar used for this study are AWR1462, and ARS401-28. The AWR1462 is a highly integrated single-chip mmWave radar sensor from Texas Instruments, designed specifically for automotive and industrial applications. Operating in the 76-81 GHz frequency band, the AWR1462 offers high range resolution and accuracy, making it ideal for advanced driver assistance systems (ADAS) and various industrial uses. This chip combines a Digital Signal Processor (DSP), a Microcontroller Unit (MCU), and a radar accelerator, significantly reducing the size, power consumption, and cost of radar systems. It supports multiple chirp configurations, allowing for flexible system design. Additionally, the AWR1462 includes built-in safety features and diagnostics to comply with automotive safety standards. Developers can leverage TI's evaluation modules (EVMs), software development kits (SDKs), and reference designs to accelerate their development process and bring robust radar solutions to market efficiently.

The ARS401-28 is a sophisticated automotive radar sensor developed by Continental, operating in the 77 GHz frequency band. This long-range radar sensor is designed to enhance vehicle safety and automation, making it ideal for applications such as adaptive cruise control (ACC), automatic emergency braking (AEB), and forward collision warning (FCW). The ARS401-28 excels in providing high resolution and accuracy, capable of precise measurements of distance, speed, and angle of multiple objects simultaneously. Its robust performance in various weather conditions, including rain, fog, and snow, ensures consistent and reliable operation. The sensor meets stringent automotive safety standards, featuring self-diagnostics and fault detection for safe and dependable usage. With comprehensive support and documentation from Continental, the ARS401-28 facilitates seamless integration into advanced driver assistance systems (ADAS), contributing significantly to enhanced driving safety and convenience.

A.5 Common Automotive Radar Sensor Interfaces

The common interfaces used for automotive radar sensors are USB, Ethernet, and CAN. The most common one due to the widespread use of CAN interface in the vehicles, is CAN.

- USB interface: The USB interface transmits the data through serial communication. The module usually waits for a initialize message that should be sent in bytes. Then it starts to send the detection array back to the computer that with the datasheet are translated.
- Ethernet interface: The Ethernet interface is reliable and fast, but less common in vehicles. Interacting is through UDP or TCP/IP packets, that while knowing the IP of the sensor, it only requires to define a socket that listens to that IP and port.
- CAN interface: The most common interface in automotive radars is CAN bus. The CAN bus, sends the 8 byte frames on a CAN address. The driver, uses CAN socket to receive those packets, and then based on the databse file, it translates those packets.

A.6 CAN interface

The Controller Area Network (CAN) interface is a robust vehicle bus standard designed to enable microcontrollers and devices to communicate with each other without a host computer. Initially developed by Bosch in the 1980s for automotive applications, CAN has become a crucial technology in various embedded systems, particularly in the automotive industry, due to its reliability, efficiency, and cost-effectiveness.

A.6.1 Overview of CAN Interface

The CAN interface facilitates communication between different electronic control units (ECUs) in a vehicle. Each ECU can send and receive messages, making it possible for multiple systems within a vehicle to operate cohesively. CAN operates on a multi-master, message-oriented protocol, meaning that any node can initiate a transmission if the bus is free. This decentralization enhances the system's robustness and flexibility.

A.6.2 Technical Specifications

- **Bit Rate:** CAN supports bit rates up to 1 Mbps for standard CAN and up to 5 Mbps for CAN FD (Flexible Data-rate).
- **Message Frames:** CAN messages are transmitted in frames that include an identifier, control bits, data bytes (up to 8 bytes for standard CAN and up to 64 bytes for CAN FD), a cyclic redundancy check (CRC), and acknowledgment bits.
- **Error Detection:** CAN includes several error detection mechanisms such as bit stuffing, monitoring, cyclic redundancy check, and acknowledgment checking, ensuring high reliability.
- **Bus Arbitration:** CAN uses a non-destructive bitwise arbitration method, allowing higher-priority messages to be transmitted first without losing lower-priority messages.

A.6.3 CAN Protocol Layers

CAN protocol consists of two layers: the Data Link Layer and the Physical Layer.

1. Data Link Layer: This layer is responsible for message framing, arbitration, acknowledgment, error detection, and error signaling. It ensures that messages are transmitted reliably and efficiently across the network.

2. Physical Layer: This layer defines the actual transmission of data over the network. It includes specifications for electrical signal characteristics, bit timing, and physical media (e.g., twisted pair cables).

A.6.4 Applications in Automotive Systems

In automotive systems, CAN interfaces play a pivotal role in managing communications between various subsystems, including:

- **Engine Control Unit (ECU):** Communicates with sensors and actuators to manage engine performance, emissions, and fuel efficiency.
- **Transmission Control Unit (TCU):** Coordinates with the ECU to optimize gear shifts and improve driving dynamics.
- **Anti-lock Braking System (ABS):** Exchanges data with wheel speed sensors and the ECU to prevent wheel lockup during braking.
- **Infotainment Systems:** Integrates multimedia, navigation, and communication features by connecting to various audio, video, and data sources.
- **Advanced Driver Assistance Systems (ADAS):** Facilitates the operation of systems such as adaptive cruise control, lane-keeping assistance, and collision avoidance by sharing sensor data and control commands.

A.6.5 Advantages of CAN Interface

The CAN interface offers several advantages that have led to its widespread adoption:

- **Robustness:** The protocol's error detection and fault confinement capabilities ensure reliable communication in noisy environments.
- **Scalability:** CAN networks can easily be expanded by adding new nodes without significant changes to the existing architecture.
- **Cost-Effectiveness:** The simplicity and efficiency of the CAN protocol reduce the need for complex wiring and lower overall system costs.
- **Real-Time Performance:** The deterministic nature of CAN's arbitration process ensures timely delivery of high-priority messages, essential for real-time control applications.

A.7 CAN driver

Writing of the CAN driver depends highly on the database file known as .dbc file. In this file, the architecture of the CAN frames are presented, with which it is understood how to manipulate every bit of the 8 byte frame of the CAN. Here is the CAN driver example for the ARS 401 radar used for this study.

```
#include "decode_radars/decoder.h"
#include <string>

#include <boost/program_options.hpp>
```

```
decoder::decoder(int id_, int mode_){

    id=id_;
    mode=mode_;

    count=0;

    std::string pub_topic_status="/list_messages";
    std::string pub_topic_content="/list_contents";

    objects_list.objects.resize(100);



    status_can_sub=nh.subscribe("/can_tx",100,
                                &decoder::status_callback,this);
    content_can_sub=nh.subscribe("/can_tx",100,
                                &decoder::content_callback,this);

    radar_points=std::make_shared<
                            std::vector<pcl::PointXYZI, Eigen::aligned_allocator<pcl::P
                            >(250);

    if ( mode == _cluster)


    {
```

```
    std::string domain="_cluster_";

    std::string topic_status_cluster=pub_topic_status+domain+
                                     std::to_string(id);
    status_decode_pub=nh.advertise<decode_radars::ClusterList>(
        topic_status_cluster, 10
    );

    std::string topic_content_cluster=pub_topic_content+domain+
                                      std::to_string(id);

    content_decode_pub=nh.advertise<decode_radars::ClusterRadar>(
        topic_content_cluster, 10
    );

    std::string pointcloud_topic="/radar/pointcloud_"+std::to_string(id);

}

else if (mode == _object)
{
    std::string domain="_object_";

    std::string topic_status_object=pub_topic_status+domain+
                                     std::to_string(id);

    status_decode_pub=nh.advertise<decode_radars::ContiList>(
        topic_status_object, 50
    );
```

```
        std::string topic_content_object=pub_topic_content+domain+
                                            std::to_string(id);

        content_decode_pub=nh.advertise<decode_radars::ContiRadar>(
            topic_content_object, 50
        );

        object_list_pub=nh.advertise<decode_radars::ContiRadarList>(
            topic_content_object+"_list", 100
        );


    }


    else{
        std::cerr<<"Error! please choose either\
                    of the available modes"<<std::endl;
    }

}









void decoder::status_callback(const can_msgs::Frame& frame){
```

```
int id_status_msg;



// this is for the cluster mode
if (mode == _cluster){



    id_status_msg=1536 + id*16;

    if (frame.id != id_status_msg) return;

    decode_radars::ClusterList output;

    output.nof_targetsnear=frame.data[0];
    output.nof_targetsfar=frame.data[1];

    Byte b2,b3;


    b2.byte=frame.data[2];
    b3.byte=frame.data[3];


    output.meas_counter= b3.bit1*pow(2,0)+b3.bit2*pow(2,1)+
                         b3.bit3*pow(2,2)+b3.bit4*pow(2,3)+
                         b3.bit5*pow(2,4)+b3.bit6*pow(2,5)+
                         b3.bit7*pow(2,6)+b3.bit8*pow(2,7) +
```

```
                        b2.bit1*pow(2,8)+
                        b2.bit2*pow(2,9) +b2.bit3*pow(2,10)+
                        b2.bit4*pow(2,11)+b2.bit5*pow(2,12)+
                        b2.bit6*pow(2,13)+b2.bit7*pow(2,14)+
                        b2.bit8*pow(2,15);

    output.interface_version=frame.data[4];



    status_decode_pub.publish(output);

}



// this is for the object mode
else if (mode == _object){


    id_status_msg=1546 + id*16;

    if (frame.id != id_status_msg) return;

    decode_radars::ContiList output;

    output.nof_objects=frame.data[0];

    Byte b1,b2;
```

```
    b1.byte=frame.data[1];
    b2.byte=frame.data[2];

    output.meas_counter= b2.bit1*pow(2,0)+b2.bit2*pow(2,1)+
                         b2.bit3*pow(2,2)+b2.bit4*pow(2,3)+
                         b2.bit5*pow(2,4)+b2.bit6*pow(2,5)+
                         b2.bit7*pow(2,6)+b2.bit8*pow(2,7) +

                         b1.bit1*pow(2,8)+
                         b1.bit2*pow(2,9) +b1.bit3*pow(2,10)+
                         b1.bit4*pow(2,11)+b1.bit5*pow(2,12)+
                         b1.bit6*pow(2,13)+b1.bit7*pow(2,14)+
                         b1.bit8*pow(2,15);

    output.interface_version=frame.data[3];




    status_decode_pub.publish(output);
}



else{
    std::cerr<<"none of the modes activated"<<std::endl;
    return;
}
```

```
}

void decoder::content_callback(const can_msgs::Frame& frame){

    int id_content_msg;


     if (mode==_cluster){



        id_content_msg=1793 + id*16;

        if (frame.id!=id_content_msg) return;

        decode_radars::ClusterRadar output;

        output.target_id=frame.data[0];

        Byte b1,b2,b3,b4,b5,b6;

        b1.byte=frame.data[1];
        b2.byte=frame.data[2];
        b3.byte=frame.data[3];
        b4.byte=frame.data[4];
        b5.byte=frame.data[5];
        b6.byte=frame.data[6];

        output.longitude_dist=TARGET_DIST_LONG_MIN+ TARGET_DIST_RES*(
```

```
    pow(2,0)*b2.bit4 + pow(2,1)*b2.bit5 + pow(2,2)*b2.bit6 +
    pow(2,3)*b2.bit7 + pow(2,4)*b2.bit8 +
    pow(2,5)*b1.bit1 + pow(2,6)*b1.bit2 + pow(2,7)*b1.bit3 +
    pow(2,8)*b1.bit4 + pow(2,9)*b1.bit5 +
    pow(2,10)*b1.bit6 + pow(2,11)*b1.bit7 + pow(2,12)*b1.bit8
);

output.lateral_dist=TARGET_DIST_LAT_MIN + TARGET_DIST_RES*(
    pow(2,0)*b3.bit1+pow(2,1)*b3.bit2+
    pow(2,2)*b3.bit3+pow(2,3)*b3.bit4+
    pow(2,4)*b3.bit5+pow(2,5)*b3.bit6+pow(2,6)*b3.bit7+
    pow(2,7)*b3.bit8+pow(2,8)*b2.bit1+pow(2,9)*b2.bit2);

output.longitude_vel=TARGET_VREL_LONG_MIN + TARGET_VREL_RES*(
    pow(2,0)*b5.bit7+pow(2,1)*b5.bit8+pow(2,2)*b4.bit1+
    pow(2,3)*b4.bit2+pow(2,4)*b4.bit3+pow(2,5)*b4.bit4+
    pow(2,6)*b4.bit5+pow(2,7)*b4.bit6+pow(2,8)*b4.bit7+
    pow(2,9)*b4.bit8);

output.lateral_vel= TARGET_VREL_LAT_MIN + TARGET_VREL_RES*(
    pow(2,0)*b6.bit6+pow(2,1)*b6.bit7+
    pow(2,2)*b6.bit8+pow(2,3)*b5.bit1+pow(2,4)*b5.bit2+
    pow(2,5)*b5.bit3+pow(2,6)*b5.bit4+pow(2,7)*b5.bit5+
    pow(2,8)*b5.bit6);

output.rcs=TARGET_RCS_MIN + TARGET_RCS_RES*(frame.data[7]);

output.header.frame_id="radar_";
output.header.stamp=frame.header.stamp;

(*radar_points)[output.target_id].x=output.longitude_dist,
```

```
    (*radar_points)[output.target_id].y=output.lateral_dist,
    (*radar_points)[output.target_id].z=output.longitude_vel,
    (*radar_points)[output.target_id].intensity=output.rcs;



    content_decode_pub.publish(output);

}


else if (mode == _object){
    id_content_msg=1547 + id*16;


    if (frame.id!=id_content_msg) return;

    decode_radars::ContiRadar output;

    output.obstacle_id=frame.data[0];

    Byte b1,b2,b3,b4,b5,b6;

    b1.byte=frame.data[1];
    b2.byte=frame.data[2];
    b3.byte=frame.data[3];
    b4.byte=frame.data[4];
    b5.byte=frame.data[5];
    b6.byte=frame.data[6];

    output.longitude_dist = OBJECT_DIST_LONG_MIN+
```

```
OBJECT_DIST_RES*(pow(2,0)*b2.bit4+pow(2,1)*b2.bit5+
pow(2,2)*b2.bit6+
pow(2,3)*b2.bit7+pow(2,4)*b2.bit8+pow(2,5)*b1.bit1+
pow(2,6)*b1.bit2+pow(2,7)*b1.bit3+pow(2,8)*b1.bit4+
pow(2,9)*b1.bit5+
pow(2,10)*b1.bit6+pow(2,11)*b1.bit7+pow(2,12)*b1.bit8);

output.lateral_dist = OBJECT_DIST_LAT_MIN+
OBJECT_DIST_RES*(pow(2,0)*b3.bit1+pow(2,1)*b3.bit2+
pow(2,2)*b3.bit3+pow(2,3)*b3.bit4+pow(2,4)*b3.bit5+
pow(2,5)*b3.bit6+pow(2,6)*b3.bit7+pow(2,7)*b3.bit8+
pow(2,8)*b2.bit1+pow(2,9)*b2.bit2+pow(2,10)*b2.bit3);

output.longitude_vel = OBJECT_VREL_LONG_MIN+
OBJECT_VREL_RES*(pow(2,0)*b5.bit7+pow(2,1)*b5.bit8+
pow(2,2)*b4.bit1+pow(2,3)*b4.bit2+pow(2,4)*b4.bit3+
pow(2,5)*b4.bit4+pow(2,6)*b4.bit5+pow(2,7)*b4.bit6+
pow(2,8)*b4.bit7+pow(2,9)*b4.bit8);

output.lateral_vel = OBJECT_VREL_LAT_MIN+
OBJECT_VREL_RES*(pow(2,0)*b6.bit6+pow(2,1)*b6.bit7+
pow(2,2)*b6.bit8+pow(2,3)*b5.bit1+pow(2,4)*b5.bit2+
pow(2,5)*b5.bit3+pow(2,6)*b5.bit4+pow(2,7)*b5.bit5+
pow(2,8)*b5.bit6);

output.rcs=OBJECT_RCS_MIN+OBJECT_RCS_RES*frame.data[7];




output.header.frame_id="radar_front";
```

```
        output.header.stamp=frame.header.stamp;

        objects_list.objects[output.obstacle_id]=output;

        content_decode_pub.publish(output);
    }

    else{
        return;
    }
}
```

A.8 Radar Full-State Observer Real-Time C++ Code

```
#include <decode_radars/clustering.hpp>
#include <decode_radars/kalman.hpp>
#include <decode_radars/utilities.h>
#include <decode_radars/data_association.hpp>
```

```
class track
{

    public:


    int label;
    uint no_measurement_count;
    uint measurement_count;

    Eigen::VectorXf X;

    Eigen::VectorXf x;

    uint id;
    ros::Time last_stamp;

    bool matched=false;


    std::vector<Eigen::VectorXf,
                Eigen::aligned_allocator<Eigen::VectorXd>> footprint;

    std::shared_ptr<KalmanFilter> ekf;
```

```
    track();

    track(Point point_,
    ros::Time stamp_, uint id_, int label_);

    void naive_prediction(std::vector<Eigen::VectorXf,
                          Eigen::aligned_allocator<Eigen::VectorXf>>& predictions);

    bool get_match();

    bool cpr(ros::Time new_stamp);

    bool sanity_check(const decode_radars::ContiRadar obj);

    void set_match(bool match_);

    void do_update_with_measurement(
    Eigen::VectorXf meas, ros::Time stamp_);
    void do_update_without_measurement();


};


bool track::cpr(ros::Time time_)
{
    double dt=time_.toSec() - last_stamp.toSec();

    if (dt>= 0.5)
        return false;
    else
```

```
        return true;
}


void track::do_update_with_measurement(Eigen::VectorXf meas, ros::Time stamp_)
{
    measurement_count++;
    no_measurement_count=0;

    float dt=stamp_.toSec() - last_stamp.toSec();

    Eigen::MatrixXf A(6,6);

    A<<1, dt, pow(dt,2)/2 , 0.0, 0.0, 0.0,
        0, 1,  dt         , 0.0, 0.0, 0.0,
        0, 0,  1.0        , 0.0, 0.0, 0.0,
        0, 0,  0          , 1.0, dt, pow(dt,2)/2,
        0, 0,  0          , 0.0, 1.0, dt,
        0, 0,  0          , 0.0, 0.0, 1.0;


    last_stamp=stamp_;

    this->ekf->A=A;
    this->ekf->update(meas ,stamp_);

    footprint.push_back(this->ekf->get_state());

}

void track::do_update_without_measurement()
```

```
{
    no_measurement_count++;
}

track::track(Point point_, ros::Time stamp_, uint id_, int label_)
{
    label=label_;
    x.resize(6);
    x(0)=point_.x;
    x(1)=0.0f;
    x(2)=0.0f;

    x(3)=point_.y;
    x(4)=0.0f;
    x(5)=0.0f;

    footprint.push_back(x);

    last_stamp=stamp_;

    Eigen::MatrixXf A(6,6);
    Eigen::MatrixXf P(6,6);
    Eigen::MatrixXf Q(6,6);
    Eigen::MatrixXf R(2,2);
    Eigen::MatrixXf C(2,6);

    P=0.1*Eigen::MatrixXf::Identity(6,6);
    Q=0.1*Eigen::MatrixXf::Identity(6,6);
    R=0.1*Eigen::MatrixXf::Identity(2,2);

    C<<1.0f, 0.0, 0.0, 0.0, 0.0, 0.0,
```

```
       0.0, 0.0, 0.0, 1.0, 0.0, 0.0;

    float dt=0.07;

    A<<1, dt, pow(dt,2)/2 , 0.0, 0.0, 0.0,
        0, 1,  dt         , 0.0, 0.0, 0.0,
        0, 0,  1.0        , 0.0, 0.0, 0.0,
        0, 0,  0          , 1.0, dt, pow(dt,2)/2,
        0, 0,  0          , 0.0, 1.0, dt,
        0, 0,  0          , 0.0, 0.0, 1.0;

    id = id_;

    ekf=std::make_shared<KalmanFilter>( A, C, Q, R, P);

    ekf->init(x);
}


typedef std::shared_ptr<track> trackptr;

class tracking
{
    public:

    bool first_round;

    std::vector<uint> used_ids;
    std::vector<uint> unused_ids;

    std::unordered_map<uint, trackptr> track_map;
```

```
    ros::NodeHandle nh;

    ros::Subscriber radar_cluster_sub;
    ros::Publisher tracks_pub;


    std::vector<trackptr> tracks;


    tracking();


    void cluster_callback(
    const sensor_msgs::PointCloud2::ConstPtr& clptr2);
};


tracking::tracking()
{
    used_ids.reserve(100);
    unused_ids.reserve(100);

    tracks.reserve(100);

    first_round=false;

    for (uint i=0; i<200; i++) unused_ids.push_back(i);


    radar_cluster_sub=nh.subscribe("/radar/fused_with_camera",
```

```
    10, &tracking::cluster_callback, this);

    tracks_pub=nh.advertise<visualization_msgs::Marker>("/radar/tracks", 10);

}



void tracking::cluster_callback(
const sensor_msgs::PointCloud2::ConstPtr& clptr2)
{


    pcl::PointCloud<pcl::PointXYZ>::Ptr clptr( new pcl::PointCloud<pcl::PointXYZ>());

    pcl::fromROSMsg(*clptr2, *clptr);

    ros::Time stamp_=clptr2->header.stamp;


    std::vector<Point> clpts_new;
    for (auto clpt:clptr->points) clpts_new.push_back(Point{clpt.x, clpt.y});




    if (tracks.size()==0)
    {
        uint clpts_idx=0;
        for (auto clpts:clpts_new)
```

```
    {

        uint id_=unused_ids.at(0);
        trackptr new_track=std::make_shared<track>(clpts, stamp_, id_);


        used_ids.push_back(id_);
        unused_ids.erase(unused_ids.begin());

        tracks.push_back(new_track);


        clpts_idx++;
    }
    return;
}


std::unordered_map<int,int> track_2_new;
std::unordered_map<int,int> new_2_track;

for (auto trackptr_:tracks) trackptr_->ekf->predict();

std::vector<Point> trackpts;
for (auto trackptr_:tracks)
    trackpts.push_back(
    Point( trackptr_->ekf->get_state()(0),
    trackptr_->ekf->get_state()(3)));
```

```
for (auto trackpt: trackpts)
    std::cout<<"the track is: "<<trackpt.x<<", "<<trackpt.y<<std::endl;
for (auto clpt_new:clpts_new)
    std::cout<<"the new is :"<<clpt_new.x<<", "<<clpt_new.y<<std::endl;

std::vector<std::vector<bool>>
can_match(clpts_new.size(), std::vector<bool>(trackpts.size(), true));

bidirectional_assignment(
trackpts,
clpts_new,
track_2_new,
new_2_track,
can_match, 10.0f);

// old tracks
uint tracker_idx=0;
for (auto tracker_it=tracks.begin(); tracker_it < tracks.end(); tracker_it++)
{
    if (track_2_new.find(tracker_idx) != track_2_new.end())
    {
        Eigen::VectorXf meas(2);
        meas<<clpts_new.at(track_2_new[tracker_idx]).x, clpts_new.at(track_2_new[tr
        (*tracker_it)->do_update_with_measurement(meas, stamp_);
    }
    else
    {
        (*tracker_it)->do_update_without_measurement();
    }

    tracker_idx++;
```

```
    }

    // new tracks
    uint clpts_idx=0;
    for (auto clpts:clpts_new)
    {
        if (new_2_track.find(clpts_idx) != new_2_track.end())
        {
            std::cout<<"found match for... "<<clpts.x<<" , "<<clpts.y<<std::endl;
            std::cout<<"which is ...."<< tracks[new_2_track[clpts_idx]]->ekf->get_state
            tracks[new_2_track[clpts_idx]]->ekf->get_state()(3)<<std::endl;
            clpts_idx++;
            continue;
        }


        std::cout<<"new track... "<<clpts.x<<" , "<<clpts.y<<std::endl;

        uint id_=unused_ids.at(0);
        trackptr new_track=std::make_shared<track>(clpts, stamp_, id_);


        used_ids.push_back(id_);
        unused_ids.erase(unused_ids.begin());
//
        tracks.push_back(new_track);


        clpts_idx++;
    }
```

```
// maintanance
for (auto iter=tracks.begin(); iter<tracks.end(); iter++)

{
    if ((*iter)->cpr(stamp_))
    {
        continue;
    }

    uint id_=(*iter)->id;

    auto iter_=std::find(used_ids.begin(), used_ids.end(), id_);

    used_ids.erase(iter_);
    unused_ids.push_back(id_);

    auto iter_alias=iter;
    iter--;
    tracks.erase(iter_alias);
}


for (auto iter=tracks.begin(); iter<tracks.end(); iter++)

{

    //if ( (*iter)->footprint.size() < 4) continue;
```

```
        visualization_msgs::Marker marker;

        marker.type=visualization_msgs::Marker::ARROW;
        marker.action=visualization_msgs::Marker::ADD;
        marker.header.frame_id="map";
        marker.header.stamp=ros::Time::now();

        marker.ns="tracks";
        marker.color.a=1.0f;
        marker.color.r=1.0f;
        marker.scale.y=marker.scale.z=0.1;

        float vx=(*iter)->ekf->get_state()(1);
        float vy=(*iter)->ekf->get_state()(4);

        marker.scale.x=std::hypot(vx,vy);

        float azmth=atan2(vy, vx);

        marker.pose.orientation.w=cos(azmth/2);
        marker.pose.orientation.z=sin(azmth/2);


        marker.pose.position.x=(*iter)->ekf->get_state()(0);
        marker.pose.position.y=(*iter)->ekf->get_state()(3);
marker.pose.position.z=1.0f;
        marker.pose.position.z=0.0f;

        marker.id=(*iter)->id;

        marker.lifetime=ros::Duration(0.1);
```

```
    tracks_pub.publish(marker);

}

for (auto iter=tracks.begin(); iter<tracks.end(); iter++)

{

    //if ( (*iter)->footprint.size() < 4) continue;


    visualization_msgs::Marker marker;

    marker.type=visualization_msgs::Marker::TEXT_VIEW_FACING;
    marker.action=visualization_msgs::Marker::ADD;
    marker.header.frame_id="map";
    marker.header.stamp=ros::Time::now();

    marker.ns="velocities";
    marker.color.a=1.0f;
    marker.color.r=1.0f;
    marker.scale.x=marker.scale.y=marker.scale.z=1.0;

    float vx=(*iter)->ekf->get_state()(1);
    float vy=(*iter)->ekf->get_state()(4);



    float azmth=atan2(vy, vx);

    marker.pose.orientation.w=1.0f;
```

```
    marker.pose.position.x=(*iter)->ekf->get_state()(0);
    marker.pose.position.y=(*iter)->ekf->get_state()(3);

    marker.pose.position.z=2.0f;

    marker.id=(*iter)->id+50;

    marker.lifetime=ros::Duration(0.1);

    marker.text=std::to_string(std::hypot(vx,vy));
    tracks_pub.publish(marker);

}


for (auto iter=tracks.begin(); iter<tracks.end(); iter++)

{
    //if ( (*iter)->footprint.size() < 4) continue;

    visualization_msgs::Marker path_marker;
    path_marker.header.frame_id = "map";
    path_marker.header.stamp=ros::Time::now();
    path_marker.type = visualization_msgs::Marker::LINE_STRIP;
    path_marker.action=visualization_msgs::Marker::ADD;
    path_marker.scale.x = 0.1;
    path_marker.color.r = 0.0;
    path_marker.color.g = 1.0;
    path_marker.color.b = 0.0;
```

```
        path_marker.color.a = 1.0;
        path_marker.lifetime = ros::Duration(0.1);
        path_marker.id=(*iter)->id + 100;
        path_marker.ns="footpring";
        path_marker.pose.position.z=0.0;
        path_marker.pose.orientation.w=1.0f;


        for (auto pred:(*iter)->footprint)
        {
            geometry_msgs::Point ps;
            ps.x=pred(0);
            ps.y=pred(3);
            ps.z=1.0;
            path_marker.points.push_back(ps);
        }

        tracks_pub.publish(path_marker);
    }
}

int main(int argc, char** argv)
{

    ros::init(argc,argv,"tracking");

    tracking tracking_;


    ros::spin();
```

```
    return 0;
}
```