Velocity Vector Estimation Using Dual Automotive Radar

Master Thesis

Tijmen Brinkhof

20th of May 2025

MS3
Microwave Sensing
Signals & Systems

TU Delft

Velocity Vector Estimation Using Dual Automotive Radar

Master Thesis

by

Tijmen Brinkhof

20th of May 2025

Student number: 4460502

Thesis committee:		
	Dr. Julian Kooij,	TU Delft, IV Group
	Dr. Francesco Fioranelli,	TU Delft, MS3 Group
	Msc Simin Zhu,	TU Delft, MS3 Group

TU Delft

Contents

1

Abstract

Perception is a fundamental component of autonomous and self-driving vehicles, with reliable object detection and understanding of the environment being critical for safe operation. While lidar and camera based systems are widely used, radar remains a promising option due to its robustness in poor weather conditions and ability to directly measure the radial velocity of objects via the Doppler effect. However, radar's sparse data and resulting limitations have constrained its potential.

This thesis investigates the use of dual automotive radar setups to mitigate these limitations, and use the specific advantages of such a setup to improve full velocity vector estimation methods. A novel algorithm is proposed to achieve more accurate velocity estimation in non-ideal, real world conditions. The work further explores how improved velocity estimation can be used to improve classification performance using graph neural networks. Here it was found that including velocity information via a ground truth method did increase classification performance significantly, though the same result could not be obtained via the previously mentioned velocity estimation method. To support evaluation, a simplified simulation environment and ground truth velocity data for the RadarScenes dataset are developed.

This research aims to close the performance gap between radar and other more data-dense sensors, offering a robust and more cost effective alternative, especially in conditions where optical systems under perform.

2

Introduction

In the search for more intelligent and autonomous vehicles, perception is one of the key areas of attention. To safely navigate in every day environments, autonomous vehicles need to be able to interpret and understand their surroundings. This includes detecting pedestrians, humans, cars, trucks and all other forms of static and dynamic obstacles This needs to be done with high accuracy, since failure often leads to physical harm. This perception serves as the basis for higher-level tasks such a path planning, motion planning, and control. Therefore the design choices made in the perception system of autonomous vehicles plays a critical role in the safety and robustness of autonomous vehicles.

Typically an autonomous vehicle relies on a combination of sensors to gather information about the environment. Usually, camera systems, lidar (light detection and ranging), and radar (radio detection and ranging) are the most commonly used. Each sensor type has distinct advantages and limitations, therefore a combination of sensors is often times used to get the best performance under varying conditions.

Cameras offer high-resolution images and provide the same information as a human can see with their eyes. However, they are susceptible to bad lighting conditions and can be negatively affected by factors such as glare, darkness, and weather like fog or rain. Lidar, on the other hand provides dense 3D spatial information in the form of a point cloud. It gathers this information by emitting laser pulses in varying directions and measuring the time for the pulse to return. This way the sensor can map the environment with a high 3D resolution. Despite the advantages, lidar has limitations in terms of high cost and susceptibility to bad weather conditions. This is because, just like camera systems, lidar makes use of the (near) visible light spectrum.

Finally we arrive at radar systems, which use radio waves to measure range and relative velocity. Due to the different frequencies at which this sensor operates, and the additional velocity information that can be gathered, this system is an interesting choice for autonomous vehicles. Radar is, in contrary to camera and lidar systems, not significantly affected by lighting conditions or other external factors like fog or rain [1][2]. This allows for reliable operation both day and night. In addition, the ability to directly measure the velocity of an object can considerably contribute to the performance of higher-level perception tasks. Contrary to lidar systems which usually create high-density point clouds, data from radar systems conventionally is much more sparse than data from lidar systems.

In recent years, the performance of automotive radar systems has improved significantly, with the development of high-resolution radars and even 4D radar systems that can also measure elevation in addition to range, azimuth and velocity [3]. These improvements have reduced the sparsity gap between radar and lidar, allowing radar to play a larger role in modern autonomous perception systems. Nonetheless, challenges remain. Radar still produces relatively sparse data compared to lidar. Moreover, conventional radar systems measure only the radial component of velocity, which limits their ability to fully perceive the entire environment without integration of other sensors.

To overcome some of these limitations, this thesis explores the use of a dual radar setup in automotive applications. Making use of two radar sensors in a (fully) overlapping configuration can significantly decrease the sparsity gap. Having two sensors doubles the data rate, which translates to denser target information [4]. Additionally, a dual radar system has the potential to measure the full relative velocity of objects instantaneously, including both radial and tangential components[5]. This capability is particularly valuable for time critical systems, such as collision avoidance and adaptive cruise control [6].

Beyond safety related applications, richer radar data can also contribute in other parts of the perception. For instance, improved velocity estimation can enhance object tracking algorithms, support better classification of road users, and assist in more accurate ego-motion estimation. By lowering the performance gap between radar and lidar, dual radar configurations may offer a cost-effective and robust solution for certain perception tasks, particularly in conditions where optical sensors under perform.

2.1. Problem Definition

While velocity estimation using radar systems has been studied extensively [7][8][9], most of this work focuses on (radial) velocity measurements from a single radar. Full velocity vector estimation, especially using dual radar, remains relatively unexplored. The limited research that does exist typically uses controlled conditions not representative for real-world use [10]. This includes having targets that are very close by, or without any other moving objects in the scene. However, in practice many targets are small, far away, or close to other targets making separation difficult. In addition to this, the real world data is often contains a lot of noise and outliers, something that these algorithms are particularly vulnerable to.

The lack of robust solutions for estimating velocity vectors creates a gap, in particular since modern automotive systems depend increasingly on accurate understanding of the environments around them. This thesis addresses this gap by introducing a novel algorithm designed with dual automotive radar in non-ideal conditions in mind. Going beyond the problem of velocity vector estimation alone, this work also investigates the effects of velocity vector estimation on classification performance. This area in particular has received little attention in past academic works.

2.2. Research Questions

This thesis will focus on full velocity vector estimation and its application to classification tasks. This question is subdivided into the following research questions:

- *What are the failure cases of current full velocity vector estimation methods?*
- *How can a dual radar setup be used to improve full velocity vector estimation?*
- *How can full velocity vector estimation be used to improve classification tasks?*

Each of these questions will be investigated and subsequently answered.

2.3. Nomenclature

Between different fields of research different nomenclature is often used for the same or similar concepts. For the sake of consistency, below an overview of some key terms and what they mean in this context is given.

Starting with the definition of a *target*. A target can be described as an object of interest. This changes depending on the type of perception task. For example, when designing a pedestrian detection system, trees are objects but not targets. When doing ego-motion estimation, trees are targets since they provide valuable information about the static environment. Any object can have *Characteristic reflections*. This is a reflection that generates notable Doppler velocities and/or a high *RCS* value by

reflecting off an object's feature like a license plate or wheels. These features can also be not directly visible, like wheel houses on the opposite side of a vehicle. These features can be very beneficial to classify and to segment radar targets [11].

Moving on to how objects are *perceived* by an autonomous system, we arrive at the definition of *detection*. This is the problem of determining if there is a target within a specified area. A single point in the radar point cloud can be referred to as a detection, since it indicates the presence of a target. *Clustering* is then the process of assigning labels of the same kind to each point in a point cloud belonging to the same object. This is then finally followed by *Classification*, this is the process of determining what class the target belongs to after detecting the it's presence.

For an autonomous system to be able to perceive its environment, it needs a sensor system. A *dual radar* system will be used for the purpose of this thesis. The term dual radar is used to describe a system of two of the same radars working together. These radars are physically separated, usually by the width of the car. Despite the distance between them, they purposefully have overlapping fields-of-view, allowing both radars to detect the same objects. In this case the radars are also front-facing, allowing them to obtain the most relevant information.

In Figure 2.1 the difference between a single radar and dual radar in the context of this thesis are visualized.

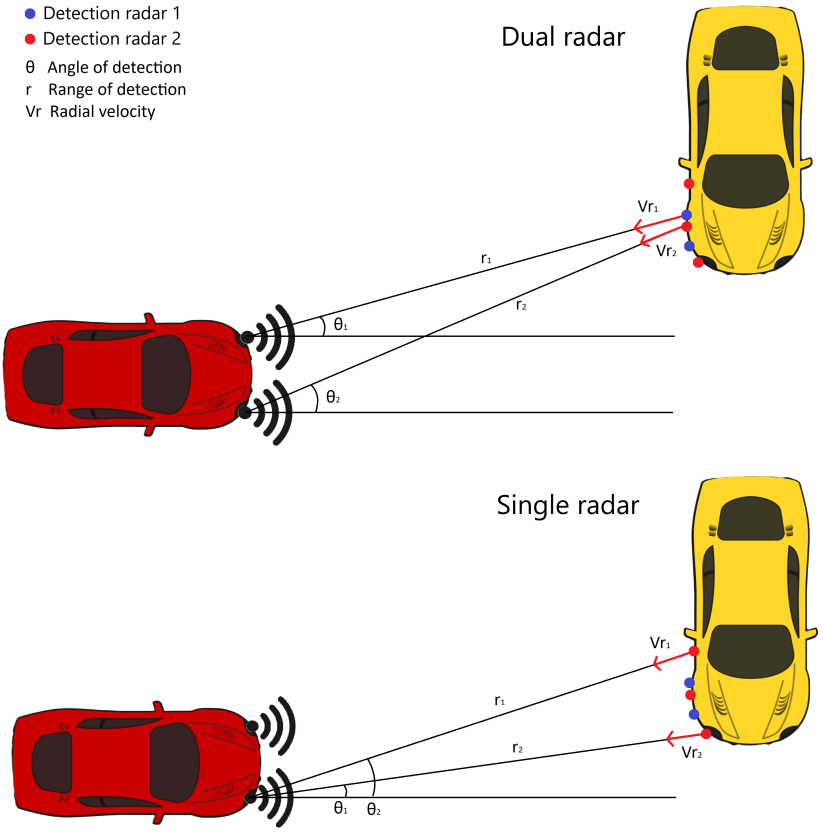

Figure 2.1: Illustration of detections from a single radar compared to detections from a dual radar setup.

In the context of automotive radar systems, it is also important to understand the different types of motion. In this thesis various types of velocity are used, each describing movement from a different perspective. To prevent confusing the reader in later sections, this section introduces and explains the different velocity definitions, which are also visualized in Figure 2.2.

The first concept that needs explaining is that of *full velocity*. This refers to the 2D vector describing its velocity in an earth-fixed cartesian space. This can be in relation to different perspectives, which is what will be addressed next.

Absolute velocity is defined as the full velocity of an object relative to the static environment. This means all motion is observed from to point-of-view of the world, independent of any moving objects.

In contrast, *relative velocity* describes the full velocity vector between two objects from the perspective of the radar. This relative velocity is what the radar sensor typically measures. Without knowing the vehicles own motion, it is impossible to convert the relative velocity to the absolute velocity.

This vehicles own velocity is also known as the *ego-velocity*. This is the absolute velocity of the vehicle on which the radar is mounted. As mentioned, this velocity is crucial in converting relative to absolute velocities.

All these variants of the full velocity are not directly measurable by the radar system, a quantity that can be directly measured is the *radial velocity*. This is the component of the relative velocity in the direction of the radar system. It is determined by measuring the frequency shift in the returning radar signal caused by the velocity of the target, this is also known as the *Doppler effect*.

The velocity perpendicular to the radial velocity is called the *tangential velocity*. From a single view point, this velocity is not directly measurable. This is the case because by definition this velocity has no component in the direction of the radar system.

Another useful velocity is the *lateral velocity*. This velocity is perpendicular to the ego-velocity and can be helpful in describing objects that move across the field-of-view of the observing vehicle.

Up till this point, all described velocities have been *translational*. In some scenario's, it is also necessary to consider an objects *rotational* velocity. It describes the motion around an objects own center axis.

Finally, throughout this thesis the concept of *ground truth velocity* will be used. This refers to any velocity of which its exact value is already known, and can be used to compare to other measured or calculated velocities. Usually these ground truth values are obtained via direct measurement of wheel speed, or tracking systems like GNSS.

A visual explanation of each type of velocity mentioned above is given in Figure 2.2.

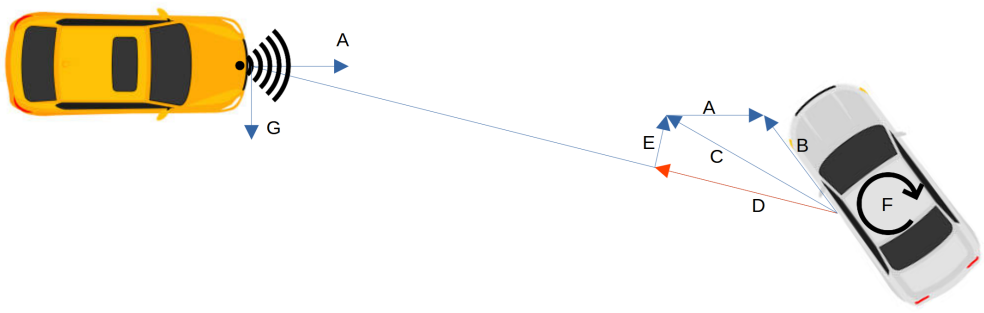

Figure 2.2: Image illustrating the different velocities. (A) Ego-velocity, (B) Absolute velocity, (C) Relative velocity, (D) Radial velocity, (E) Tangential velocity, (F) Rotational velocity (G) Lateral velocity

2.4. Thesis structure

To guide the reader through this document more efficiently, the contents of this thesis will be briefly discussed. Chapter 3 will discuss academic works related to the topics of velocity estimation and classification using automotive radar systems. After this, a novel velocity estimation method will be introduced, which will be tested in the experiments chapter. These tests involve both performance in velocity estimation, and the effectiveness of using this velocity estimation in classification tasks. At the end, the conclusion of this work will be given, and a direction for future work will be proposed.

3

Related Work

To create a starting point for this thesis and provide the reader with more background information, this chapter reviews relevant prior work in the field of automotive radar systems. Doing so is important for several reasons: it provides insights into existing methods, shows the limitations, and therefore identifies the gaps and open challenges that are still to be solved. This allows the research to be more efficiently directed to areas of importance.

To better understand the problem and to create a baseline approach, three key components that are required to answer the research questions will be highlighted:

- Velocity estimation methods
- Target classification methods
- Dataset for training and testing

At the end of this chapter, a suitable option will be chosen from each of these components to be used as a starting point for this thesis.

3.1. Velocity Estimation

Data from the radars first has to be clustered in order to be used for higher-level tasks like classification and tracking. In [12], [13] and [14] the DBSCAN [15] algorithm is proposed for initial clustering. This algorithm clusters points based on the location of the detection, combined with the radial velocity and the timestamp. It is also suggested to include a range-depended value for the minimal amount of points needed to form a cluster to account for the increased sparsity of data points with range.

In [14] another step is added to the clustering process after the initial clustering as described above. As a second step, clusters with two members being closer than some threshold are evaluated as possibly being the same object. For each of the clusters the full velocity is calculated as described in [10]. In short, the radar detections lead to a over-determined linear system of equations. Solving this system will yield the true relative velocity vector. After this the velocity vectors are compared in both magnitude and direction. If the error of these two variables is below some threshold, the clusters are merged. It is also mentioned that this second step could “noticeably benefit” from a better real velocity estimation based on a radars with overlapping fields of view.

This method assumes that all points on the target move with the same velocity. This implies the target is small, or when the target is large, it is rigid and the rotational velocity of the target is small enough so it does not significantly change the measured radial velocity of each data point.

In [10] it is stated that a combination of multiple radars could significantly improve the accuracy of rotational velocity measurements. This method is based on a stationary dual radar setup, whereas in the assumed situation in this thesis the radars are mounted on a moving vehicle.

As an expansion the velocity vector estimation mentioned above, the paper [16] proposes a method to instantaneously measure the angular velocity in addition to the translational velocity. This method is based on exploiting the characteristics of the velocity profile of a moving vehicle as described in [10]. An illustration of a velocity profile is given below:

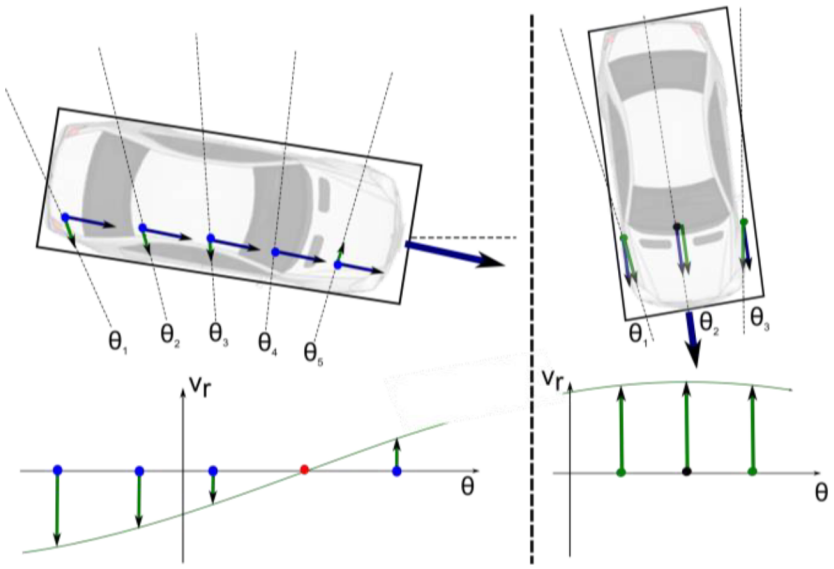

Figure 3.1: Velocity profile of a linear moving vehicle (blue vector). For each observation the radial velocity component is measured by the radar sensor (green vector). Considering one radial cell the velocity describes a cosine (green curve), for a crossing car (left) and a car heading towards the sensor (right). From [10].

Like for the previous method, this paper describes solving an over-determined system of equations. Instead of solving for the full velocity directly, it is assumed the target is rotating around an instantaneous center of rotation (ICR). This leads to a system with three unknowns, the x and y position of the ICR plus the orientation of the vehicle, also meaning now a minimum of three measurements on a single target are required to solve the equation. This system is then solved using Orthogonal Distance Regression to prevent bias due to errors-in-variables. In simulations this method provides better performance over typical (weighted) least-squares solvers [9].

In Figure 3.2 the advantage of stereo vision again becomes clear. By having physically separated radars the total azimuth area of the velocity profile is increased, resulting in more reliable estimations of velocity profile properties [10].

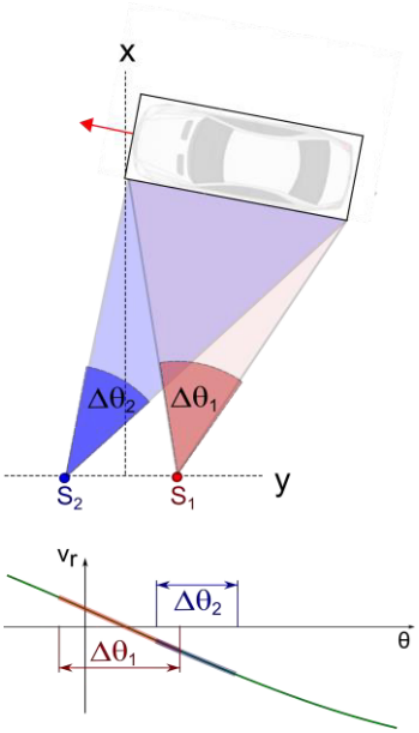

Figure 3.2: Fusion of two radar sensors, resulting in an extended total azimuth area of the velocity profile (different aspect angles). From [10].

Though the addition of the targets angular velocity would provide valuable information, the application of this method would be impractical in some use cases. The paper assumes a situation of a car at 15m distance, with 20 measurements on the vehicle for each radar. Upon inspection of for example the Radar Scenes dataset [17], it quickly becomes clear that these assumptions do not hold in a lot of the situations where the target is far away. Also, this method does not account for other types of targets like cyclists and pedestrians.

To increase the robustness of the aforementioned velocity estimation methods, in [8] [9], and [7] outlier filtering is done by applying the RANSAC [18] to the velocity profile. This method iteratively tries different combinations of points from the cluster, and decides which points create the best fit according to some loss function. These points are then marked as inliers. Applying this outlier detection method before using the over-determined system of equations to solve for the velocity results in a more accurate velocity estimation and makes it more robust to errors made during clustering.

By having multiple viewpoints and more detections on a target, a larger and more varied amount of data is available for relative- and rotational velocity extraction. The resulting, more accurate, estimated velocity can then be used to improve clustering, which in turn could be used to improve other perception tasks like classification and object tracking.

Even though the advantages of having a dual radar become clear from literature, very little work has been done quantifying the results. For the work that has been done, it has mostly been generated in ideal and controlled conditions. This leaves a gap that can be addressed in this thesis.

3.2. Target Classification

Now that velocity estimation methods have been discussed, this section will go into detail about various classification options. One thing to consider when judging the suitability of the models for use in this thesis, is the possibility of integrating full velocity vector information into the model. Depending on how the model function and how it is trained, this might not always be trivial to do.

3.2.1. PointNet

The most prevalent method for object classification nowadays is the use of neural networks. One example of a neural network that is commonly used is PointNet [19]. PointNet is a Deep Learning based 3D classification and segmentation model. In this case, the segmentation model is an extension to the classification network. When released, PointNet provided state-of-the-art performance when tested on the ShapeNet dataset [20] in 2017. In the years after, multiple papers have been published improving PointNet or using it as a building block for new methods [21][22][23]. In [13] a comparison between multiple PointNet based and other (hybrid) deep learning methods is made. Hybrid models are a combination of multiple smaller models of different types, i.e. machine learning, deep learning, or classical methods.

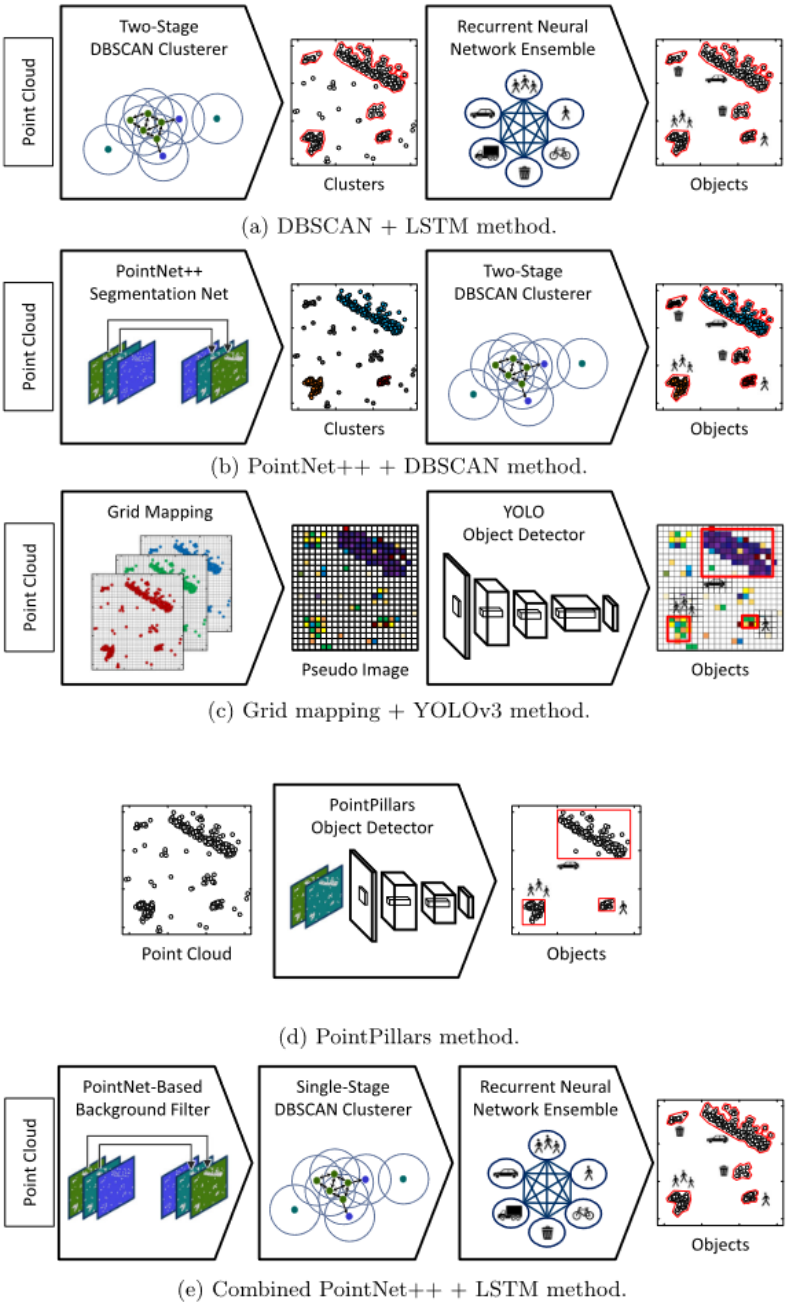

Figure 3.3: Schematic method overview. Five main architectures are compared in this article: a utilizes a clustering algorithm followed by a recurrent neural network classifier. b makes the classification first via semantic segmentation and uses the extra information as additional input to a clusterer. c comprises an image-based object detector made accessible for point clouds via a grid mapping approach. d omits the grid mapping stage by using an object detector optimized for point clouds. Finally, e combines the first two methods in order to utilize the advantages of both architectures. All methods use the same point cloud as input. Due to space constraints, the point cloud are only displayed for the PointPillars [24] method in d. Dependent on the different methods, cluster formations of boxes are returned as object predictions. From [13].

3.2.2. YOLO

From this comparison, it is concluded a You Only Look Once (YOLO) v3[25] approach performs best closely followed by a PointNet++ [21], DBSCAN [15] and LSTM [26] combined model. In the comparison made in [13] multiple occasions of improvements with modular models are mentioned. This creates opportunities for dual radar, since dual radar can contribute to better performance multiple steps of the modular models. It should also be noted that since the release of [13] newer versions of the YOLO model have been released, with the latest being version 9 [27]

3.2.3. Pointillism

In [28] another neural network is proposed that is specifically trained on sparse (dual) radar data. In contrast to the previously mentioned methods, this network is not a classifier but instead is trained to create oriented bounding boxes around objects from a single class, in this case cars. By making use of the dual radar setup, errors in orientation estimation can be significantly reduced. In Figure 3.4 ambiguity in simulated results from different orientations is shown compared to the distance between radars. From Figure 3.5 it becomes clear the optimal radar separation distance is somewhere between 1.5 and 2 meters. Note that this simulation was performed at a maximum distance of 10 meters, and results will be different depending on the type of radar that is used.

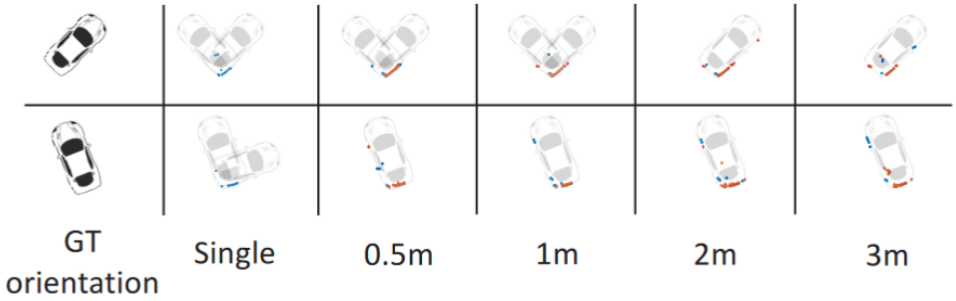

Figure 3.4: Wireless Insite simulations: Point clouds with single radar or smaller separation could lead to ambiguity in pose of the car which gets eliminated by increasing radar separation. Orange and blue points are from 2 different radars. From [28].

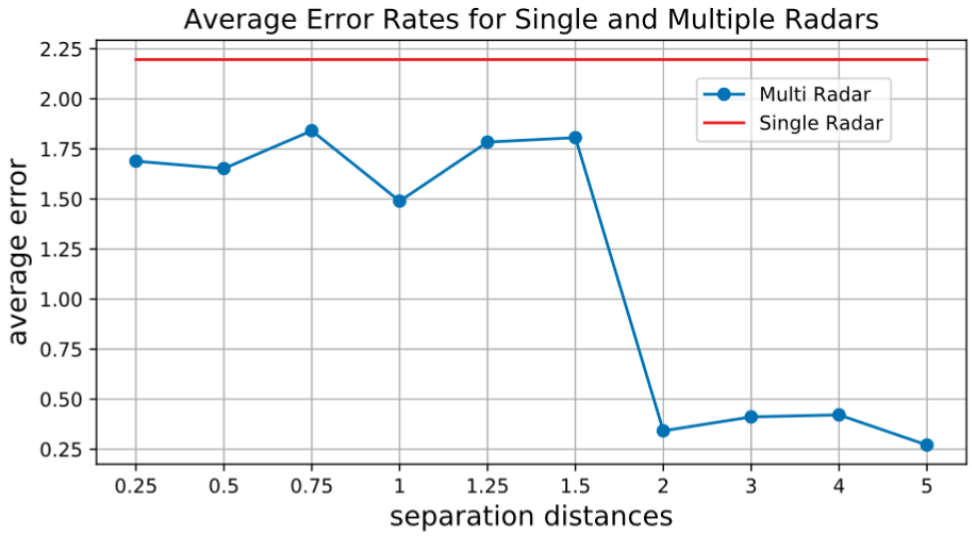

Figure 3.5: Comparison of average error in radian between single and multiple radars. For a separation greater than 1.5m, the performance improves. From [28].

This paper further elaborates on the fusion of the data from the individual radars. Instead of simply merging the radar data, the observation that false alarms are independent of each other in space is leveraged. To filter the data, a space-time coherence framework is used, resulting in a representation of *Cross Potential Point Clouds*. This representation includes a soft probability value of the points coming from an actual object.

Furthermore, in [29] it is stated that having Doppler features from multiple viewing angles results in a significant gain for classification. This is particularly true for low Doppler frequencies and cases with high intraclass variation.

3.2.4. RadarGNN

RadarGNN is a *Graph Neural Network* that is trained specifically for processing radar data. It does this by first converting the radar point cloud into a graph using a *K-Nearest Neighbors* algorithm. This graph is then fed into a neural network, which detects and classifies objects in the data. The full architecture can be seen in Figure 3.6.

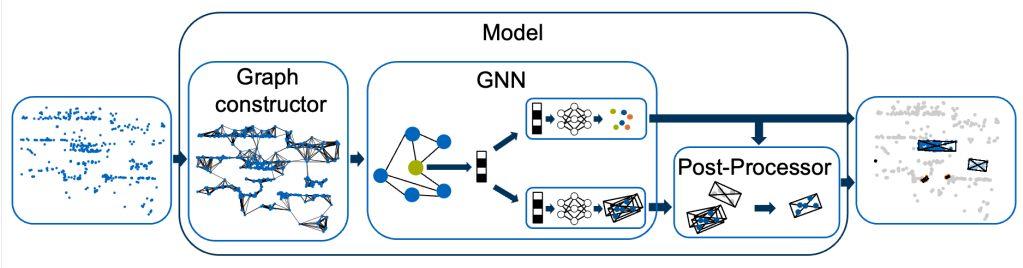

Figure 3.6

This project makes use of the RadarScenes [17] dataset, which will be discussed later in this chapter. It also combines radar frames over a small amount of time, this way the neural net has more information to work with. It also provides a way to combine information from multiple unsynchronised radars.

Although RadarGNN effectively makes use of spatial and Doppler information, it does not make use of any advantages that a dual radar setup might offer even when such a setup is readily available in the RadarScenes dataset. This provides an ideal opportunity for this thesis.

3.2.5. Summary

Hybrid classification methods provide performance close to purely deep learning based methods. Dual radar can benefit these hybrid methods, since dual radar can increase performance of multiple steps in the classification pipeline:

- Better data filtering by making use of *Cross Potential Point Clouds*.
- Using better true relative velocity estimation to improve clustering.
- Increasing classifier performance by providing more “characteristic” reflections.
- Better bounding box (orientation) estimation due to multiple viewpoints.

3.3. Datasets

In this section the different datasets that are available for dual radar purposes are discussed. Firstly, different datasets parameters are discussed. Next, a number of different datasets will be presented. For the datasets which are currently available, the data will be explored.

3.3.1. Radar Specifications

As a first step it is important to understand radar data and its different parameters. These parameters can be divided into three separate categories: radar specifications, radar configuration, and dataset parameters.

- Range [m]
- Framerate [Hz]
- Resolution in range, angle, and velocity [m,degree,km/h]

- Doppler [yes,no]
- Elevation [yes,no]
- Pre-CFAR [yes,no]

Though not a hard cut-off, range is indicated in meters and can be roughly separated into the following brackets: long-range (<250 m), middle-range (<180 m), short-range (<50 m), and ultra-short-range (<25 m).

In theory the resolution is the minimum distance, angle, or velocity between which targets can still be separated. By using i.e. cooperative radars, multi-chip cascaded Multiple-input multiple-output (MIMO) radar or a spinning radar it is possible to increase the (angular) resolution. This has as added benefit that by multiple radars more data is generated. This results in less sparse data, and a higher detection chance of detecting targets further away.

If Doppler data is available this means the radial velocity measurement is included in the data.

Some modern radars also include elevation data in the form of a measured angle. The inclusion of this data can also be indicated as “4D” data: range, horizontal angle (azimuth), radial velocity, vertical angle (elevation).

3.3.2. Radar Setup

- Number of radars [-]
- Overlapping fields of view [yes/no]
- Physical separation between radars [yes/no]

Lastly, dataset parameters in the form of types of scenarios, annotations and other sensors were looked at. These are summarized in Table 3.1.

3.3.3. Dataset Parameters

- Scenarios [Urban/highway/parking lot etc.]
- Target types [trucks/cars/bicycles/pedestrians etc.]
- Weather [clear/rain/fog etc.]
- Size [number of scenarios/time in hours]
- Annotations [2D/3D/point-wise etc.]
- Other sensors [Lidar, (stereo)camera, IMU, GNSS etc.]

For the purpose of this thesis, only datasets with more than one radar (of the same type), with (partially) overlapping fields of view and physical separation for automotive purposes are considered. For each of the datasets found, a full table with parameters will be given.

3.3.4. Search for Datasets

When researching radar related datasets is it hard to miss the “awesome radar perception” github page and the accompanying paper “Towards Deep Radar Perception for Autonomous Driving: Datasets, Methods, and Challenges” [3]. This webpage and paper provide an elaborate overview of radar datasets, see Figure 3.7, and is frequently updated.

The automotive datasets are separated into three different categories:

- Conventional Radar Datasets for Autonomous Driving
- Pre-CFAR Datasets for Detection
- 4D Radar Datasets

Name	Year	Task	Radar Type	Data	Doppler	Range	Other Sensors	Scenarios	Weather	Annotations	Size
Radar Datasets for Detection											
nuScenes [34]	2020	DT	LR	PC	✓	SV	CLO	USH	✓	3D,T	L
PixSet [35]	2021	DT	LR	PC	✓	MR	CLO	USP	✓	3D,T	M
RadarScenes [36]	2021	DTS	HR	PC	✓	SV	CO	USHT	✓	P_w	L
Pointillism [37]	2020	D	2LR	PC	✓	MR	CL	U	✓	3D	M
Zendar [38]	2020	D	SAR	ADC,RD,PC	✓	MR	CLO	U	✗	P_w	S
Dense [43]	2020	D	LR	PC	✓	LR	CLO	USHT	✓	3D	L
RADIATE [44]	2020	LDT	SP	RA	✗	SV	CLO	USHP	✓	2D,T, P_s	M
Radar Pre-CFAR Datasets for Detection											
CARRADA [45]	2020	DTS	LR	RAD	✓	SR	C	R	✗	2D,P_w,M,T	M
RADDet [46]	2021	D	LR	RAD	✓	SR	C	US	✗	2D	M
CRUW [47]	2021	D	LR	RAD	✓	USR	C	USHP	✗	P_o	L
RaDICaL [48]	2021	L	LR	ADC	✓	USR,SR	C_dO	USHIP	✗	2D	L
Ghent VRU [49]	2020	DS	LR	RAD	✓	SR	CL	U	✗	M	M
4D Radar Datasets for Detection											
Astyx [2]	2019	D	HR	PC	✓	MR	CL	SH	✗	3D	S
View-of-Delft [50]	2022	DT	HR	PC	✓	SR	CLO	U	✗	3D,T	S
RADIal [51]	2021	DS	HR	ADC,RAD,PC	✓	MR	CLO	USH	✗	P_o,M	M
TJ4DRadSet [52]	2022	DT	HR	PC	✓	LR	CLO	U	✗	3D, T	M
Radar Datasets for Localisation											
Oxford [53]	2020	L	SP	RA	✗	SV	CLO	U	✓	P_s	L
Mulran [54]	2020	L	SP	RA	✗	SV	LO	US	✗	P_s	M
Boreas [55]	2022	LD	SP	RA	✗	SV	CLO	S	✓	P_s,3D	L
EU Long-term [56]	2020	L	LR	PC	✓	LR	CLO	U	✓	P_s	M
Endeavour [57]	2021	L	LR	PC	✓	5LR	LO	S	✗	P_s	M
ColoRadar [58]	2021	L	HR,LR	ADC,PC	✓	2USR	LO	SIT	✓	P_s	M
Radar Datasets for Other Tasks											
PREVENTION [59]	2019	DT	LR	PC	✓	1LR,2SR	CLO	UH	✓	2D,T	L
SCORP [60]	2020	S	LR	ADC,RAD	✓	USR	C	P	✗	M	S
Ghost [61]	2021	DS	LR	PC	✗	LR	CLO	S	✗	P_w	M

Figure 3.7: Task: D, T, L, and S stand for detection, tracking, localisation, and segmentation; Type: LR, HR, SP, and SAR stand for low-resolution, high-resolution, spinning, and SAR; Range: SV, LR, MR, SR, and USR stand for surrounding view, long-range (<250 m), middle-range (<180 m), short-range (<50 m), and ultra-short-range (<25 m); Other Sensors: C, Cd, L, and O stand for camera, RGBD camera, Lidar, and odometry; Scenarios: U, S, H, P, T, R, and I stand for urban (city), suburban, highway, parking lot, tunnel, race track, and indoors; Size: L, M, and S stand for large, medium, and small; Weather stands for adverse weather; Label: 2D, 3D, T, Pw, Po, Ps, and M stand for 2D bounding box, 3D bounding box, track ID, pointwise detection, object-level point, pose, and segmentation mask.

From this table it becomes clear very few datasets have multiple radars. Even if they do, most of them do not have overlapping fields of view or have no physical separation. Also, none of the available datasets make use of a coherent radar network. This means a mechanism like the “cross-potential point cloud” algorithm from Pointillism[28] or combining multiple frames like in RadarGNN[30] has to be implemented to combine the data from the unsynchronised radars.

The lack of multiple-radar datasets enforces the idea that dual radar is a relatively unexplored area within the automotive industry. There are three datasets that fulfill the dual radar requirement, RadarScenes [17], Pointillism [28] and the recently released Bosch Street Dataset [31]. All of which will be discussed further in the section below.

In addition, the ColoRadar [32] and Endeavour [33] datasets will be briefly looked at. Though these do not fulfill the requirements stated earlier in this section, they have unique characteristics which might bring valuable inspiration.

[H] **Name**	Bosch Street Dataset	RadarScenes	Pointillism	ColoRadar	Endeavour
Radar Specifications					
Range [m]	200m	100m	-	15m	250m
Framerate	15Hz	17Hz	30Hz	5Hz	13Hz
Resolution [m]	-	0.15m	0.067m	0.117	0.39m
Resolution [°]	-	0.5°	-	1.05°,22.05°	1.6°
Resolution [km/h]	-	0.1km/h	0.72km/h	0.91km/h	0.43km/h
Doppler [yes,no]	yes	yes	yes	yes	yes
Elevation [yes,no]	yes	no	no	yes	no
Pre-CFAR [yes,no]	no	no	no	yes	both
Radar Configuration					
Number of radars [-]	9	4	2	2	5
Overlapping radars [yes,no]	Yes	yes	yes	yes	partially
Physical separation [yes,no]	Yes	partially	yes	no	partially
Dataset Parameters					
Scenarios [Urban,highway,parkinglot etc.]	All	All	All	Mostly indoors	Urban
Target types [trucks,cars,bikes etc.]	All	All	All	-	-
Weather [clear,rain,fog etc.]	All	Mostly clear	Mostly clear	-	-
Size [number of scenarios,time in minutes]	13649, 2190 minutes	158, 240 minutes	48, 8 minutes	7, 150 minutes	3, 23 minutes
Annotations [2D,3D,pointwise etc.]	3D bbox	image masks	3D bbox	-	-
Sensors [Lidar,(stereo)camera etc.]	Lidar,camera	camera	Lidar,camera	Lidar,camera	Lidar

Table 3.1: Comparison of different parameters for the RadarScenes, Pointillism, ColoRadar and Endeavour datasets.

3.3.5. RadarScenes

As described in [17], this dataset consists of measurements from four different radars. Where each radar has a field-of-view of approximately 60 degrees. The radars are configured according to Figure 3.8. Due to field of view being somewhat narrow, a large area in front of the car exists which is not covered by both front facing radars. This could potentially cause difficulties when dealing with targets or obstacles close to the vehicle. Full specifications are given in Table 3.1.

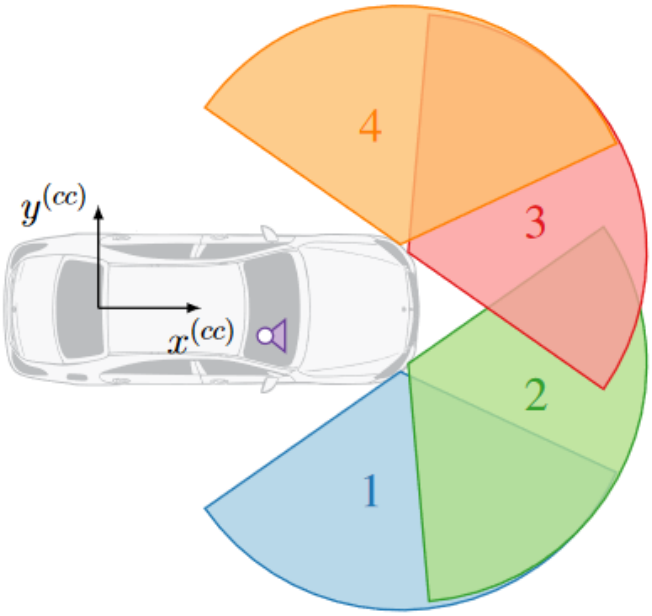

Figure 3.8: Radar set-up in RadarScenes. Note that even though some radars overlap, there is only physical separation between radars two and three.

From this Figure it becomes clear there is only physical separation between radars number two and three.

Looking at the data, we see the following information available for each frame:

- $timestamp$: in micro seconds relative to some arbitrary origin
- $sensor\_id$: integer value, id of the sensor that recorded the detection
- $range\_sc$: in meters, radial distance to the detection, sensor coordinate system
- $azimuth\_sc$: in radians, azimuth angle to the detection, sensor coordinate system
- rcs: in dBsm, radar cross section (RCS value) of the detection
- vr: in m/s. Radial velocity measured for this detection
- $vr\_compensated$: in m/s, radial velocity for this detection but compensated for the ego-motion
- x_cc and y_cc: in m, position of the detection in the car-coordinate system (origin is at the center of the rear-axle)
- x_seq and y_seq: in m, position of the detection in the global sequence-coordinate system (origin is at arbitrary start point)
- $uuid$: unique identifier for the detection. Can be used for association with predicted labels and for debugging
- $track\_id$: id of the dynamic object this detection belongs to. Empty, if it does not belong to any.
- $label\_id$: semantic class id of the object to which this detection belongs. passenger cars (0), large vehicles (like agricultural or construction vehicles) (1), trucks (2), busses (3), trains (4), bicycles (5), motorized two-wheeler (6), pedestrians (7), groups of pedestrian (8), animals (9), all other dynamic objects encountered while driving (10), and the static environment (11)

Each frame is given per radar, where the frames are not perfectly synchronised. That is, there is a slight time difference in frames between the different radars. In Figure 3.9 the point clouds from all four radars are plotted for a given frame. In Figure 3.10 the corresponding camera image can be seen.

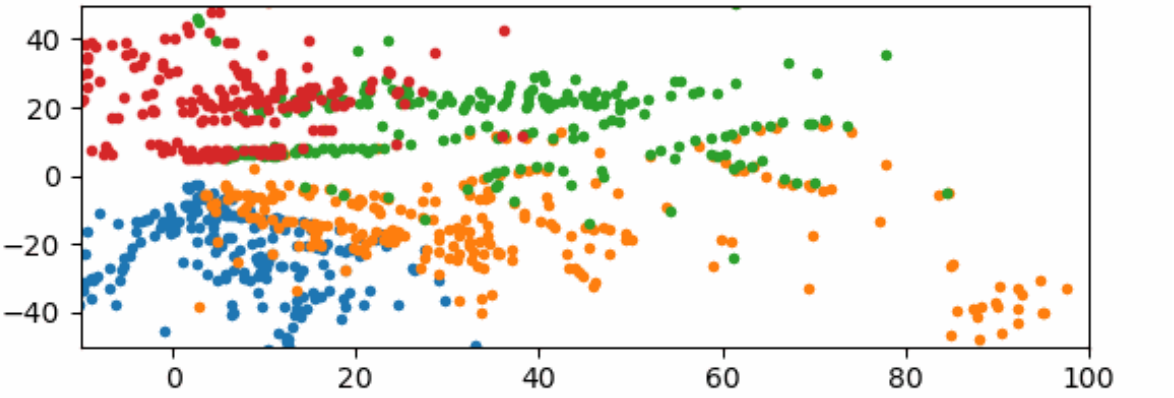

Figure 3.9: Radar frame from the RadarScenes dataset[17] with different color for each radar. Each dot represents a detection, other parameters like magnitude and Doppler velocity are not plotted.

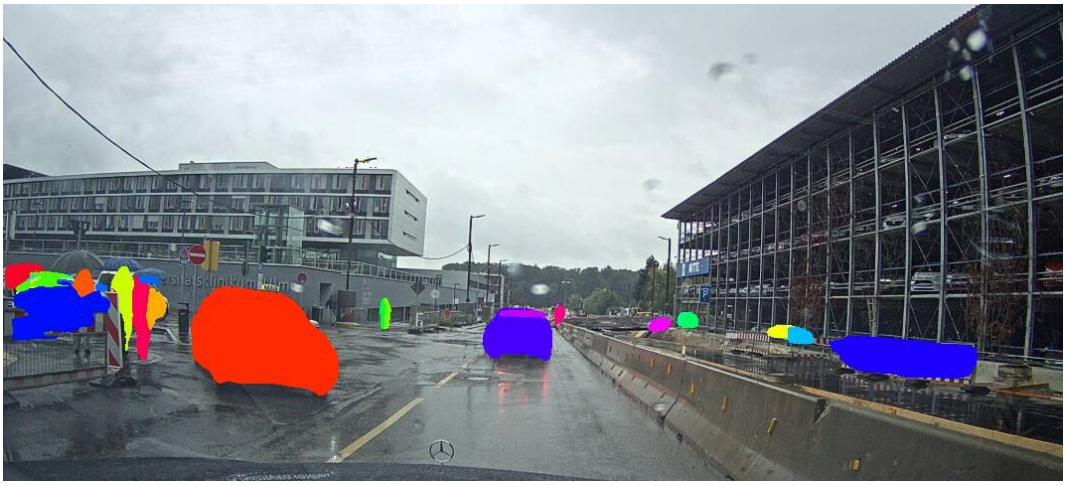

Figure 3.10: RadarScenes camera image corresponding to the radar frame in Figure 3.9.

3.3.6. Pointillism

The Pointillism dataset [28] provides data that is pre-processed by a novel “cross-potential point cloud” algorithm. This algorithm relies on the observation that noise and ghost detections are independent between radars if the radars are placed in (sufficiently) separated physical locations. By matching observations in time and space it is possible to separate noise from actual reflections from the target, leaving a near noiseless point cloud as a result. An example of such a point cloud is given in 3.11. This example is taken from the Pointillism dataset. As can be seen in the same image, 3D bounding boxes are also provided. These are generated by a deep learning architecture called RP-net [34], which is specifically trained to handle sparsity in radar point clouds.

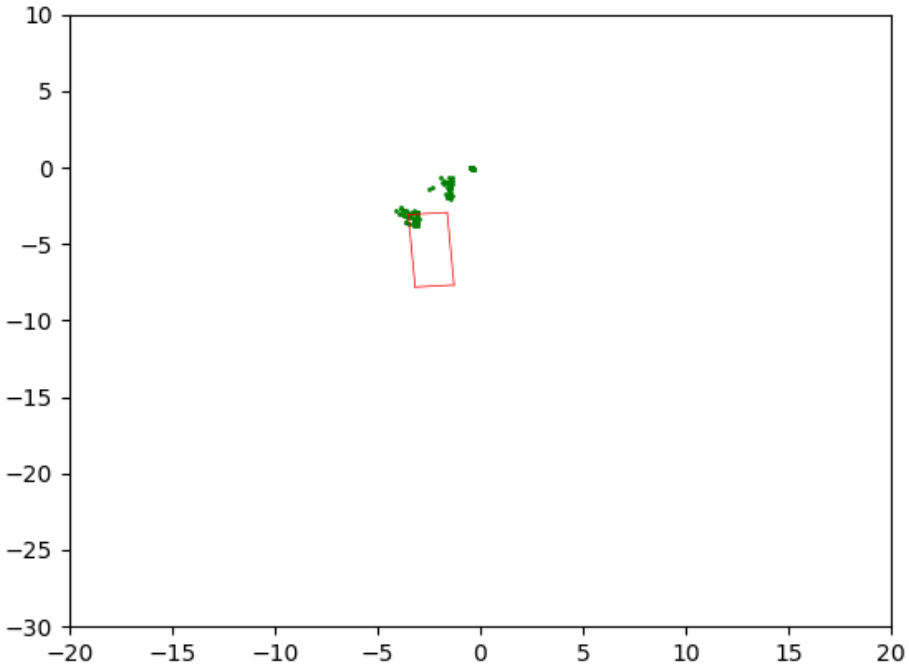

Figure 3.11: Radar frame from the Pointillism dataset. Each dot represents a detection, other parameters like magnitude and Doppler velocity are not plotted. The red rectangle indicates the estimated oriented bounding box predicted by RP-net.

Each detection in the radar data contains the following information:

- $Point\_id$
- X
- Y
- Z
- $Range$
- $Velocity$
- $Doppler\_bin$
- $Bearing$
- $Intensity$

In addition to radar, Lidar and camera data is also available for each scene. Examples for Lidar and camera can be seen respectively in Figure 3.12 and 3.13.

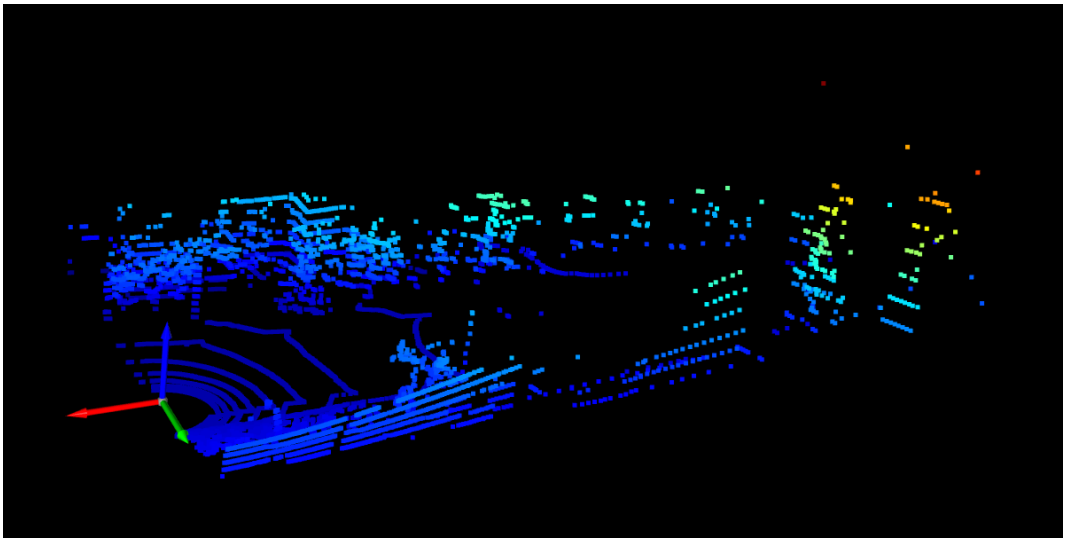

Figure 3.12: Example 3D plotted frame from the Lidar data included in the Pointillism dataset.

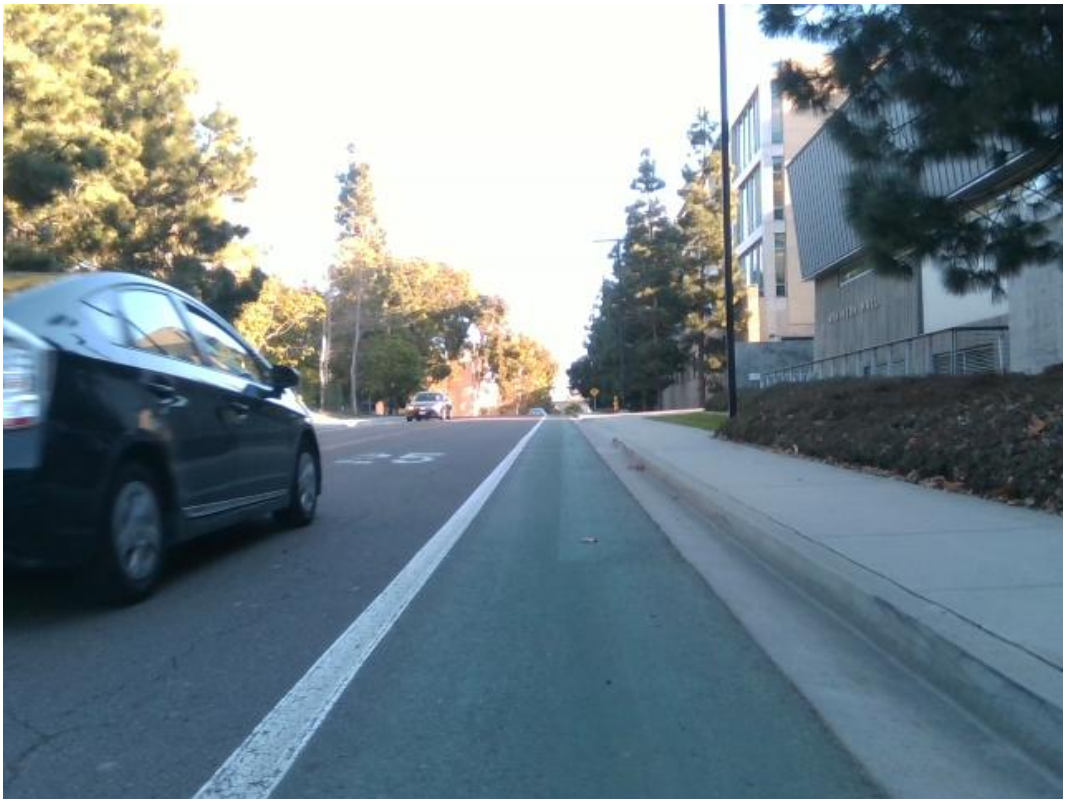

Figure 3.13: Example camera image from the Pointillism dataset.

3.3.7. Endeavour Radar Dataset

The Endeavour dataset [33] consists of data from five ARS430 radars, which can be used in both wide and narrow field-of-view modes, four Lidar sensors, and GNSS data. The radar configuration can be seen in 3.14.

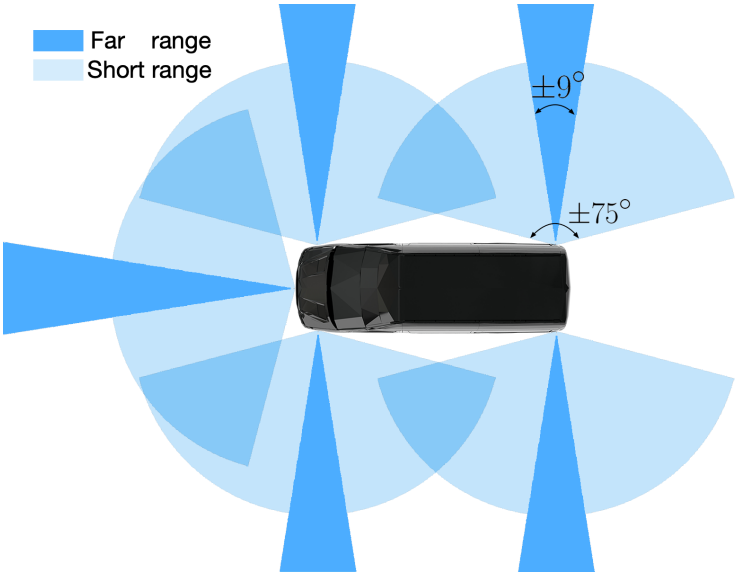

Figure 3.14: Radar configuration in the Endeavour Radar Dataset.

From the configuration it becomes clear there are no front-facing overlapping radars. On the sides of the vehicle there are wide field-of-view radars with large physical separation. This could be an advantage compared to the other previously discussed datasets, since any positive effect of having a dual radar setup will be amplified.

3.3.8. Bosch Street Dataset

The Bosch Street Dataset [31] was released in June of 2024, and is thus the newest dataset of the ones considered. It is also by far the largest dataset in terms of numbers of scenes and total time. It has nine 4D radars with 360 degree coverage around the vehicle, most of which is covered by at least two radars. The full radar setup can be seen below in Figure 3.15.

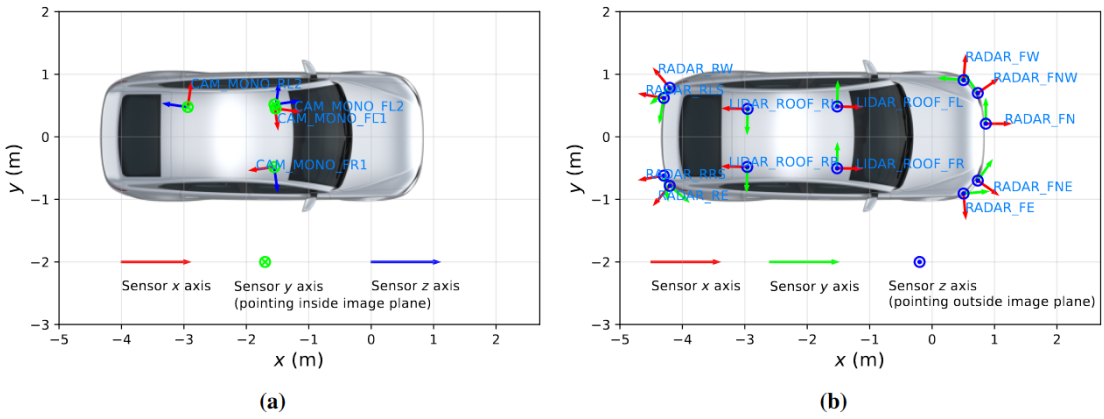

Figure 3.15: Bird's eye view of the sensor setup. (a) shows the cameras and (b) shows the Lidars and the radars. The sensors are depicted by means of of their local coordinate frames. Vehicle not to scale. From [31].

As of the writing of this thesis the Bosch Street Dataset is not publicly available, therefore it has not been possible to do any further data exploration.

3.3.9. Summary

In this section, four different datasets are discussed. Each of these datasets has unique characteristics that make them worth considering. However, current datasets have limitations such as only partially

overlapping fields-of-view, and no ground truth velocity information being available. In the end, only two are suitable for use in this thesis. These being the RadarScenes and Pointillism dataset. This is due to the Endeavour dataset not having two front-facing radars, and the Bosch dataset being unavailable for public use. The lack of ground truth velocity information will be handled later in this thesis.

3.4. Summary and Contributions

Despite improvements in radar based perception, gaps remain in the field of full velocity vector estimation and the use of dual radar in general. Existing methods often assume high quality data, or do not make use of the specific advantages of a dual radar setup. As a result, higher level perception tasks like classification also do not make full use of the motion information that is available. This thesis decreases this academic gap by introducing a novel full velocity vector estimation algorithm that was designed with dual radar in mind, and integrating the motion information into a classification model.

Starting initially as a way to validate the contributions above, a (simplified) simulation environment and ground truth velocity estimation for the RadarScenes dataset are also included in chapter 3 of this thesis.

3.5. Baseline Selection

As mentioned in the beginning of this chapter, a baseline method will have to be chosen to use as a start of the research. This method consists of three components:

- Velocity extraction
- Target classification
- Dataset

The goal of this baseline is to provide a starting point for the thesis from where dual radar based algorithms and approaches can be implemented and tested. Some considerations in this decision making process are ease of implementation, knowledge and support available, and academic potential. these factors taken into account, the RadarGNN [30] project was deemed the most suitable starting point. RadarGNN has support for the RadarScenes dataset, for which there is plenty of knowledge and support available within the MS3 group. Also, it claims state-of-the-art performance, any improvements made would thus be academically relevant. In this method, full velocity vector estimation will be implemented to see how this affects the classifying performance of the model. For the sake of simplicity, rotational velocity will not be considered. For a direct comparison to literature, the overdetermined system of equations method combined with RANSAC filtering will be used as a reference [10].

Having identified the gaps in existing velocity estimation and classification methods, and having selected a suitable baseline approach, the chapter 4 will contain the methodology of this thesis.

4

Methodology

Where the last chapter went into detail about the current state of research, this chapter will continue by describing the process and the methods used to build on that research. The purpose of this is to identify what experiments have to be done, and to create a basis in terms of analysis and mathematical support. The chapter begins by describing the current methods and its failure cases. To support the findings, a simulation environment will be developed that can be used for testing and experimentation. Next a novel algorithm will be introduced, followed by its mathematical proof.

4.1. Problem Formulation

Conventional automotive radar systems provide only radial velocity information, which is not sufficient to fully understand an objects motion without additional assumptions or information from other sensors. Existing full velocity vector estimation methods, such as using a velocity profile together with RANSAC to filter outliers, perform well in ideal conditions but fail in situations with heavy measurement noise or high outlier ratios.

Below the problem is formulated in a more precise manner:

How can the full velocity vector be accurately estimated under noisy, real-world conditions?

In the solving of this problem, the following challenges are identified:

- Dealing with sparse point clouds and limited angular information for target that are far away.
- How to make the estimation robust to outliers.
- Obtaining the velocities for all targets in the scene, without prior knowledge.

By introducing a novel *velocity graph* method, which makes use of pairwise consistency between radar detections, a solution to these challenges will be proposed. This method allows for robust and accurate full velocity vector estimation even in difficult scenarios.

In the remaining sections, the following symbols will be used in relation to the problem.

- θ, the angle of measurement in relation to the radar
- r, the distance of the measurement to the radar
- V_r, the velocity component in the direction of the radar

- V_x, the velocity component in the x direction
- V_y, the velocity component in the y direction

The following section will describe the existing methods and their failure cases, and the proposed solution in detail.

4.2. Current Method and Failure Cases

The most common method to determine the velocity of an object, either the ego-vehicle or a target, is to first extract the detections belonging to the object. This is done by fitting a velocity profile and using RANSAC to determine inliers [9][10]. This method works particularly well for objects that cover a large range of angles, like vehicles that are close (<10m) or the static environment [8]. This is because in these conditions, the velocity profile spans a large range of angles and thus creates a large variety in radial velocity vectors. An example of fitting a velocity profile to the static environment can be seen in Figure 4.1. Here the total range of angles is around 160 degrees.

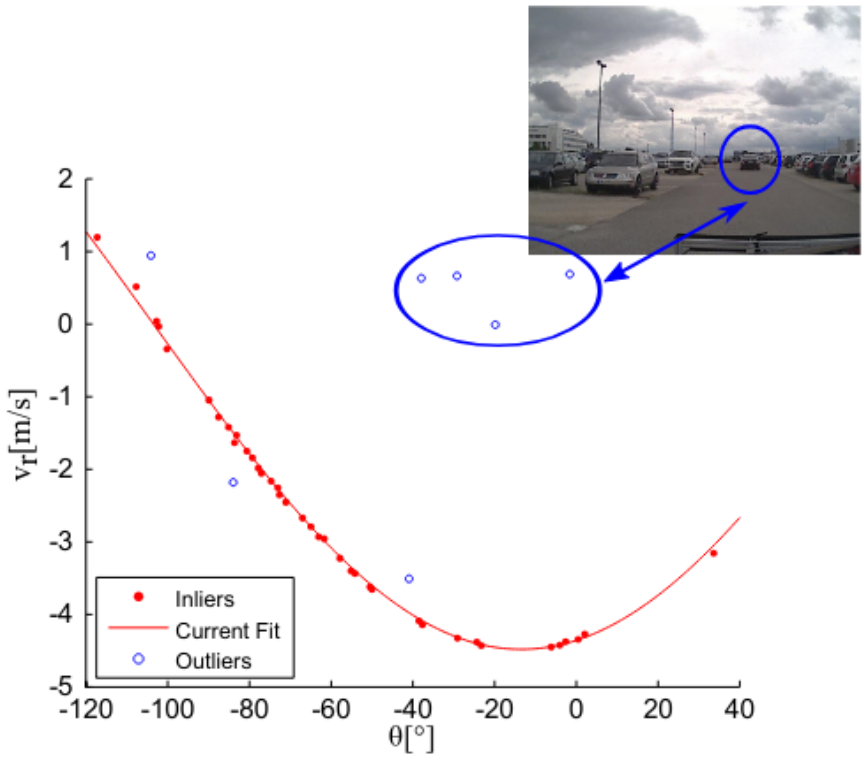

Figure 4.1: This figure shows an ego-motion estimation problem, though the same principle can be applied for moving targets. From [9].

For objects that are farther away and already do not span such a large angle as the static environment, like cars, the velocity profile becomes a straight line and thus reduces the data to only a single dimension. For objects other than the static environment, this occurs already at relatively small distances. An example of this can be seen in Figure 4.2, where a simulated object 5 meters of length was placed 20m in front of the sensors.

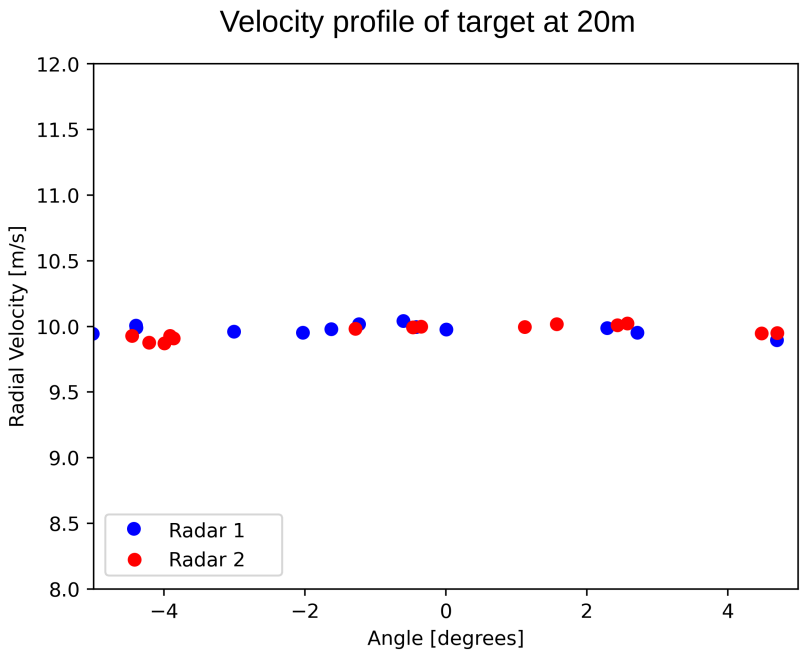

Figure 4.2: Velocity profile of simulated target of 5m at 20m distance. Here it is visible that even at this distance the profile becomes an almost straight line.

When no outliers are present this does not cause any problems since then all points belong to the target and the velocity can still be calculated. However, RANSAC will fail to determine the correct inliers when the data contains a larger percentage of outliers. This can be seen by comparing the data in Figure 4.3.

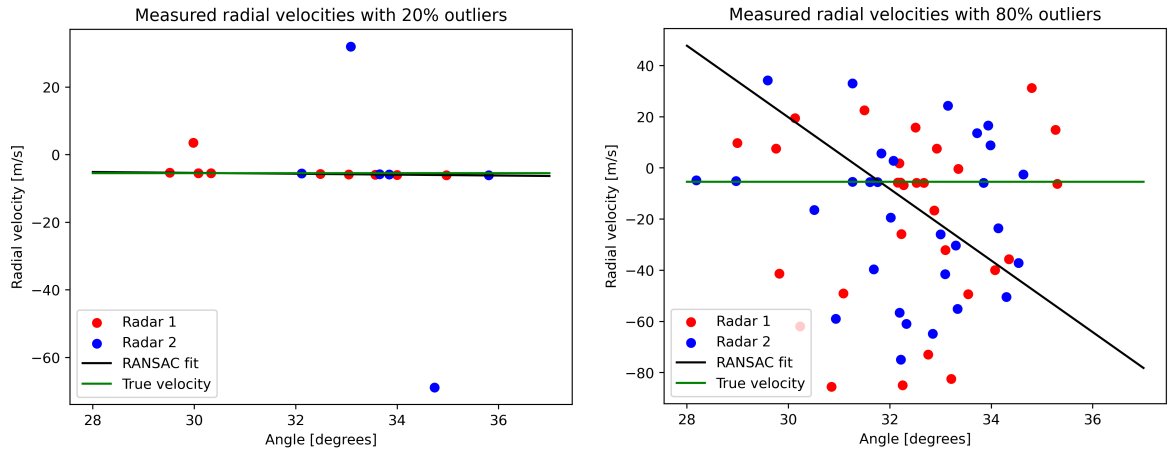

Figure 4.3: Comparison of RANSAC filtering for 20% and 80% outliers respectively. Data generated from a 5m wide target at 20m distance.

After determining the inliers, the velocity of the object can be calculated by constructing an overdetermined system of equations and solving for the full velocity vector according to the equations in Figure 4.4.

$$\begin{bmatrix} v_{r,1} \\ v_{r,2} \\ \vdots \\ v_{r,N} \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) \\ \cos(\theta_2) & \sin(\theta_2) \\ \vdots & \vdots \\ \cos(\theta_N) & \sin(\theta_N) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

Figure 4.4: Overdetermined matrix equation for computing the full velocity vector from a set of N points. From [10].

This matrix can also be computed in Cartesian coordinates, where each line will change to the following:

$$V_r = V \bullet L/|L| = (V_x * L_x + V_y * L_y)/|L| \tag{4.1}$$

where L is the distance vector from the measured point to the sensor, V is the velocity vector, L_x, L_y are the X and Y components of the vector L, respectively, V_x, V_y and V_r are the X,Y and radial components of the velocity vector, respectively.

As will be shown in the in chapter 5, this method works well for data that contains few outliers and is not noisy. In reality, especially for targets that are farther away, this method fails to accurately determine a target's true velocity vector. This is a direct result of RANSAC not being able to distinguish between in- and outliers under these conditions.

4.3. Simulation

For comparison of different methods and to be able to accurately control both system and environmental parameters, a radar point cloud simulation was written. This simulation is able to simulate both dynamic and static targets, with data from multiple radars.

Multiple assumptions are made:

- All variation in measurements follows a normal distribution. This was checked with the RadarScenes dataset to be a valid assumption.
- Targets, both dynamic and static, are represented by a 1-dimensional line. Due to the targets being relatively far away from the radar in this study, the (partially) occluded sides of the targets are of lesser importance. Also, adding reflections from these sides would provide more information to the algorithm. Making this assumption is thus creating more difficult conditions than would be encountered in the real world.
- The number of detections decreases linearly with distance. Due to the same radar angle having to cover a linearly increasing area.
- Points, and thus noise, are generated in polar coordinates. This is because a radar sensor operates in polar coordinates.
- A given percentage of outliers is added to any target. This is done to simulate multi-path reflections and other disturbances.
- System parameters are based on the setup used in the RadarScenes dataset. Estimates of system parameters were used based on the distributions observed in the RadarScenes dataset.
- Like in the RadarGNN paper, multiple frames are combined into a single point cloud. To give the algorithm the same information in the simulation as will be provided in integration with the RadarGNN model.

- Ego-vehicle is always located at (0,0) and looking in the direction of the positive X-axis. This is the same coordinate system as used in the RadarScenes dataset.

From the RadarScenes paper, the specifications of the used radar was taken as a basis for the simplified sensor model. The noise was then modeled by using the specified variation and applying this to the given parameters before constructing the full point cloud. In Figure 4.5 a visualization of the simulated data is given.

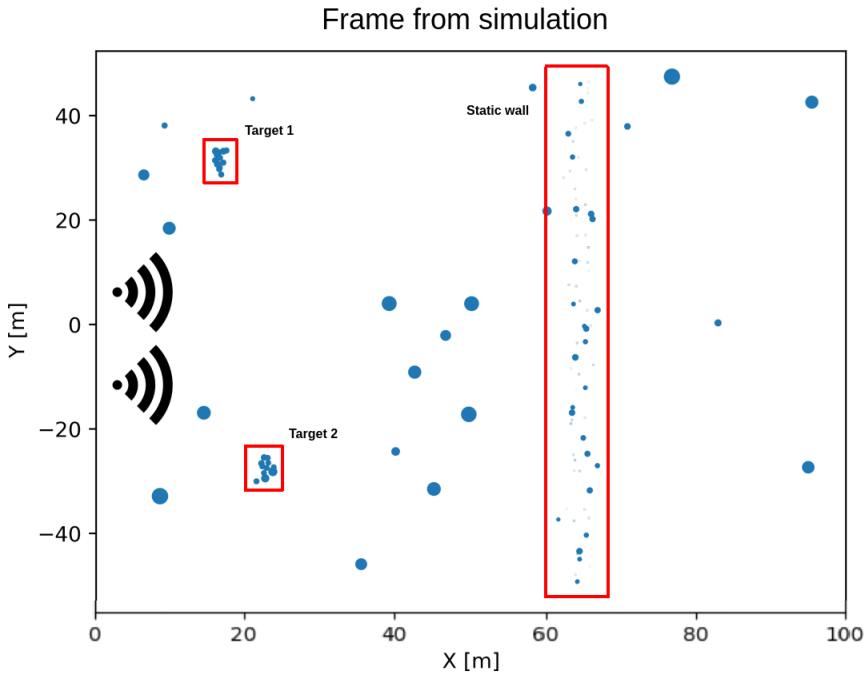

Figure 4.5: Example of simulated unprocessed point cloud showing two targets against a static background. Size of point indicates magnitude of radial velocity. Visual indicators are added to show the general positions of the radars.

In the remaining part of this chapter, the concept of a *velocity graph* will be presented, along with a novel algorithm that makes use of this concept and addresses the shortcomings of the method explained above.

4.4. Velocity Graph

As a result of the shortcomings in existing methods, the search for a better velocity estimation method was started. This search started with the idea that under perfect conditions, calculating the full velocity vector from any two random detections on an object should yield the same result. By doing the reverse of this concept, when every point in the point cloud is combined with every other point, a cluster of consistent velocity vectors should form. This is under the condition that there are no significant rotational velocities in comparison to the translational velocity.

To calculate a single velocity, at least two points are needed since the dimension of the full velocity vector is also two. An example of this calculation is shown in Figure 4.6. This is essentially the same calculation as mentioned before in Figure 4.4, but here the full velocity can be solved directly without using a least squares method.

$$\begin{bmatrix} v_{r,1} \\ v_{r,2} \\ \vdots \\ v_{r,N} \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) \\ \cos(\theta_2) & \sin(\theta_2) \\ \vdots & \vdots \\ \cos(\theta_N) & \sin(\theta_N) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

Figure 4.6: 2 by 2 version of the velocity matrix as used in this concept.

Repeating this process for all combinations of points results in a *velocity graph*. In Figure 4.7 the steps to creating the resulting graph are visualized.

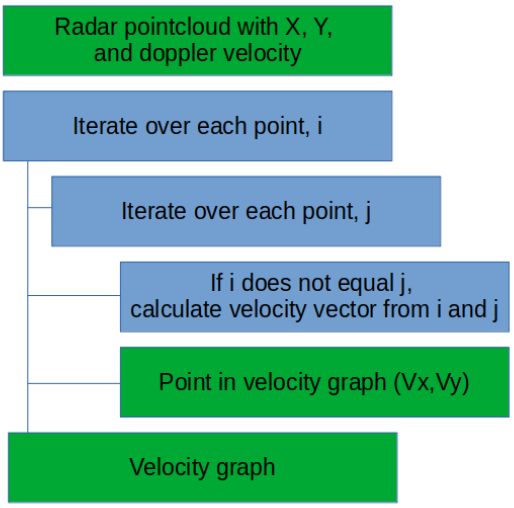

Figure 4.7: Block diagram of steps to generating the velocity graph. Green indicates data, blue indicates a processing step.

Iterating over n points as described in this diagram will lead to a total of n^2 points in the velocity graph. All combinations of points P are shown in Equation 4.2.

$$P = \begin{bmatrix} (0,0) & (0,1) & (0,2) & \dots & (0,i) \\ (1,0) & (1,1) & (1,2) & \dots & (1,i) \\ (2,0) & (2,1) & (2,2) & \dots & (2,i) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (j,0) & (j,1) & (j,2) & \dots & (i,j) \end{bmatrix} \tag{4.2}$$

Here it also becomes clear that some performance increase can be obtained by only using the combination of points from either the upper or lower half of the matrix, since a combination of (i,j) will give the same velocity as a combination of (j,i). The resulting complexity remains at $\mathcal{O}^2$ however.

To test this hypothesis, a single target moving with a constant velocity was simulated.

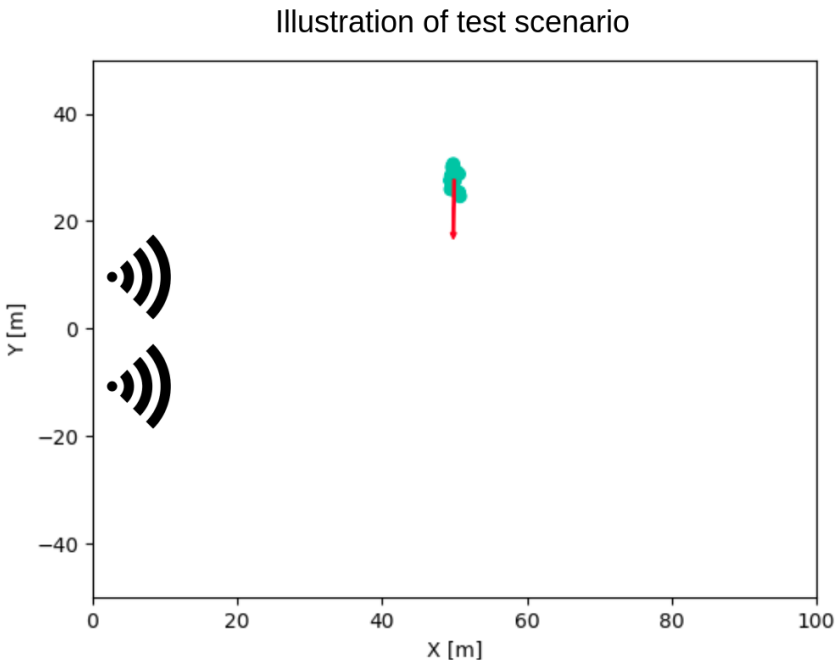

Figure 4.8: Illustration of simulated test scenario.

Next, the velocity vectors were calculated by combining every point with every other point using the equations in Figure 4.6.

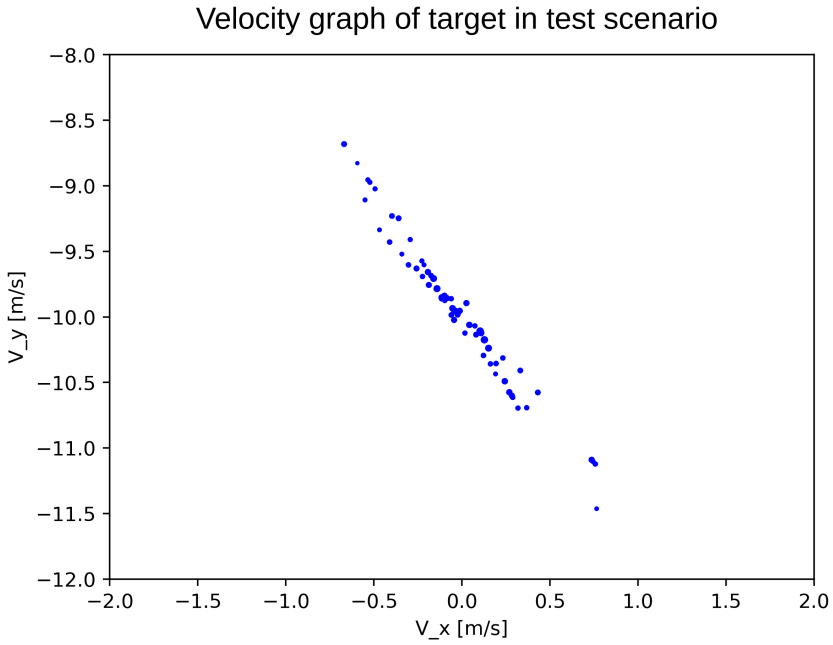

Figure 4.9: First results of generated velocity graph.

Here, it can be observed that the points are distributed along two different lines which cross each other in the center. In Figure 4.10 the points are colored based on the radar they originate from. Since every point in the velocity graph is constructed from two radar detections, the combination of one point belonging to radar one and the other belonging to radar two also exists.

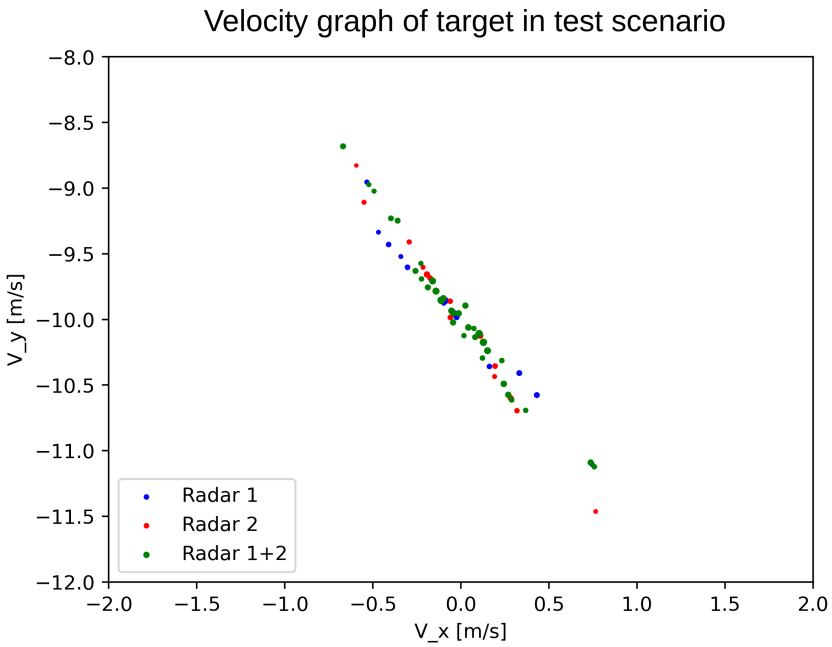

Figure 4.10: Point in velocity graph colored by radar they originate from. Radar 1 is colored blue, radar 2 is colored red, and points calculated from points from both radars are colored green.

In Figure 4.11 the previously discussed RANSAC algorithm is now used to draw two lines through the points belonging to each radar. Since this graph was generated from simulated data, the target velocity is precisely known. From this graph it becomes clear the intersection point of the line is almost exactly equal to this velocity. Also, the points generated from combinations of radar one and radar two, here in green, distribute along the direction of both lines. Why all this is the case will be discussed and mathematically proven later in this chapter.

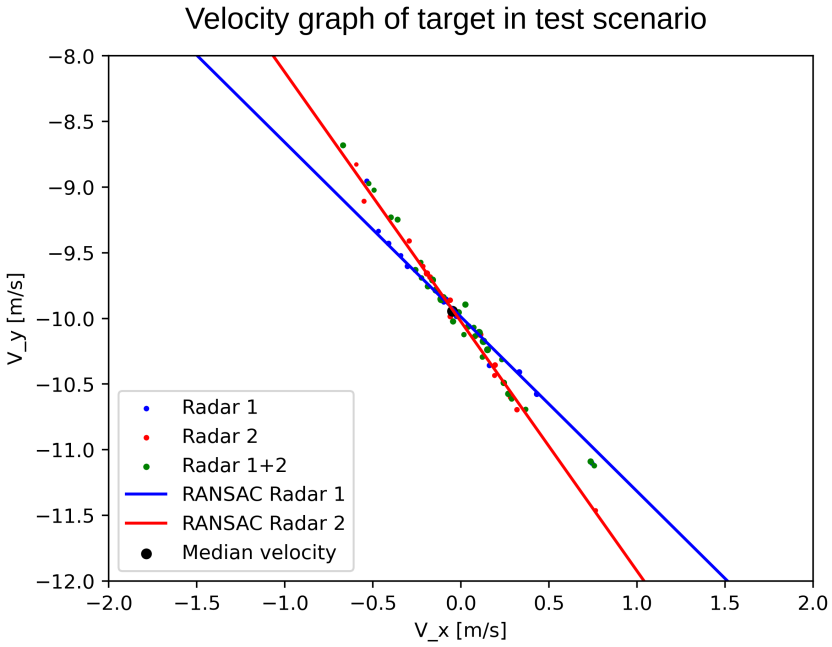

Figure 4.11: Same data and colors as in Figure 4.10, but RANSAC fit plotted through data from radar 1 and radar 2. This clearly shows the distribution of points along these lines.

Before continuing the research, it was first checked whether this phenomena was not simply an artifact of the simulation. To do this, more than one hundred radar frames from RadarScenes were used to manually verify the method. In Figure 4.12 the result from one of these frames is shown. This proves the earlier results are not related to the simulation.

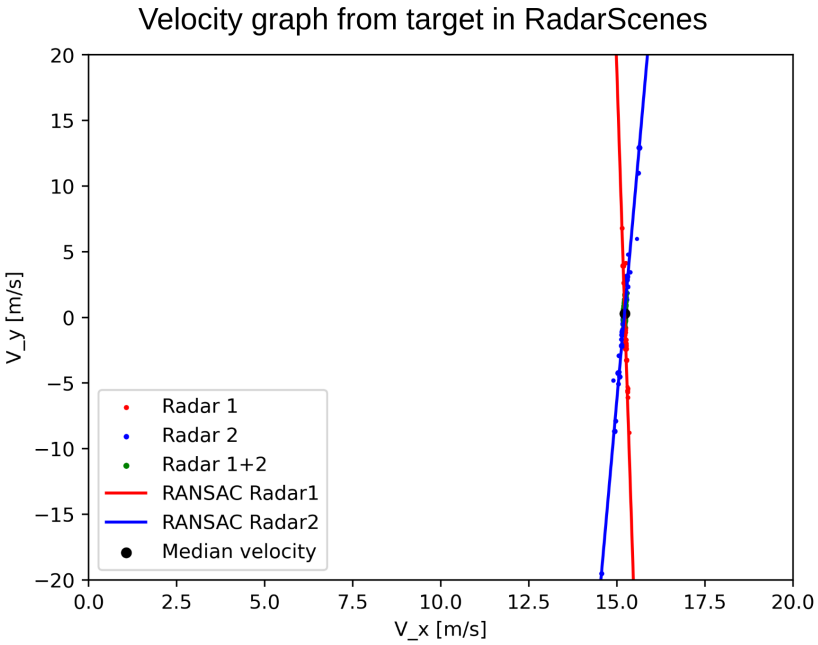

Figure 4.12: Velocity graph from target in RadarScenes. Here the same phenomena are visible as in the simulated velocity graph.

Furthermore, the following relations between the angles in the velocity graph and physical angles at the time of measurement are observed.

- The line equally dividing the large angle between the RANSAC radar lines into two has an angle to the X-axis equal to the angle to the target observed from the ego-vehicle
- The small angle between the radar lines is the same as the angle between the lines drawn from the viewpoints of the radars to the target

These angles are indicated in the velocity graph in Figure 4.13. From these observations it can also be concluded that points close, within some radius r, are constructed from points with low noise.

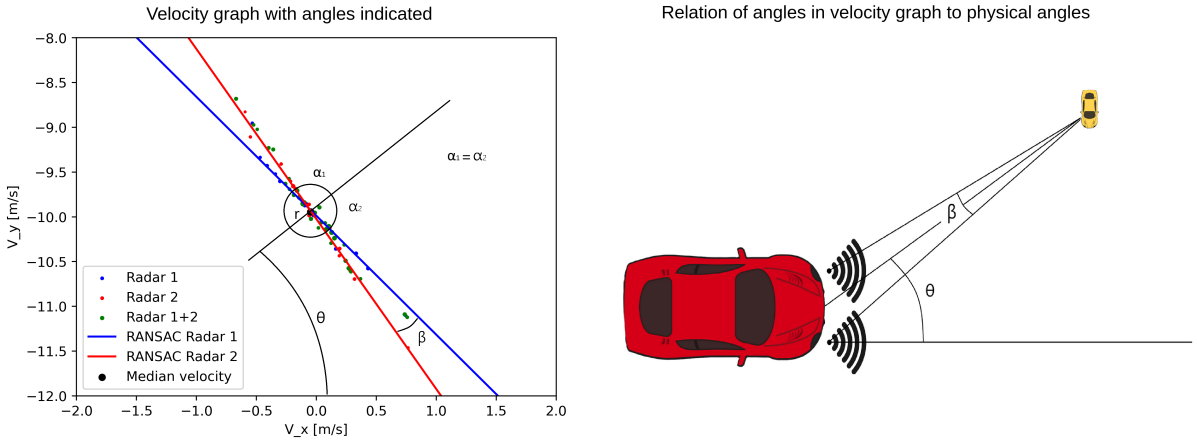

Figure 4.13: Velocity graph with angles beta and theta indicated.

4.5. Algorithm

To compare the performance of the proposed velocity graph method. Test runs were done in the simulator with a target moving laterally across the field-of-view at different distances. From these scenarios, the full velocity vector was reconstructed at every simulated timestep. The simulated test setup is visualized in Figure 4.8. From this point cloud, the velocity graph can be constructed. To extract the velocity, the points are placed in bins of $0.1m/s$. The resulting histogram is then smoothed using a Gaussian kernel, and the peak value is selected as the estimated velocity. An example of such a smoothed histogram is also shown in Figure 4.8.

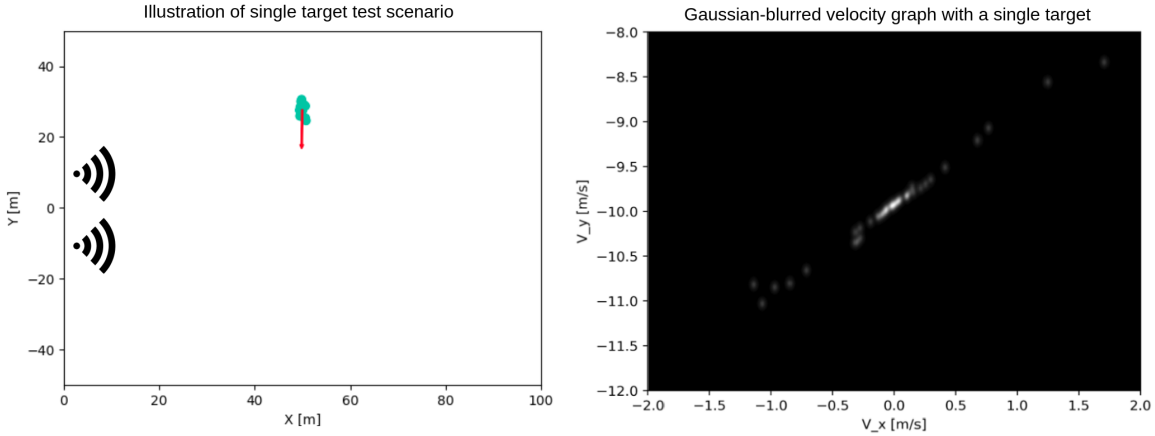

Figure 4.14: Simulation setup and corresponding Gaussian-blurred velocity graph for a single target.

It must be noted that this method does not make explicit use of the 'crossing lines' that are observed. Also, there exist various other methods that might be more suitable for this purpose. This method was chosen for its relative simplicity in this test case. In the next section a different approach will be used when dealing with situations where the point cloud contains multiple objects. For future research, it might prove worthwhile to explore other options, including the training of purpose-built neural networks.

As seen in the earlier in this section, the method of using a Gaussian-blurred histogram works well when the point cloud only contains a single object. For point clouds with multiple objects like the example given in Figure 4.15, this method results in only finding the object which consists of the most points. An example of the Gaussian-blurred histogram containing multiple objects along with the simulated scenario can be seen in Figure 4.15.

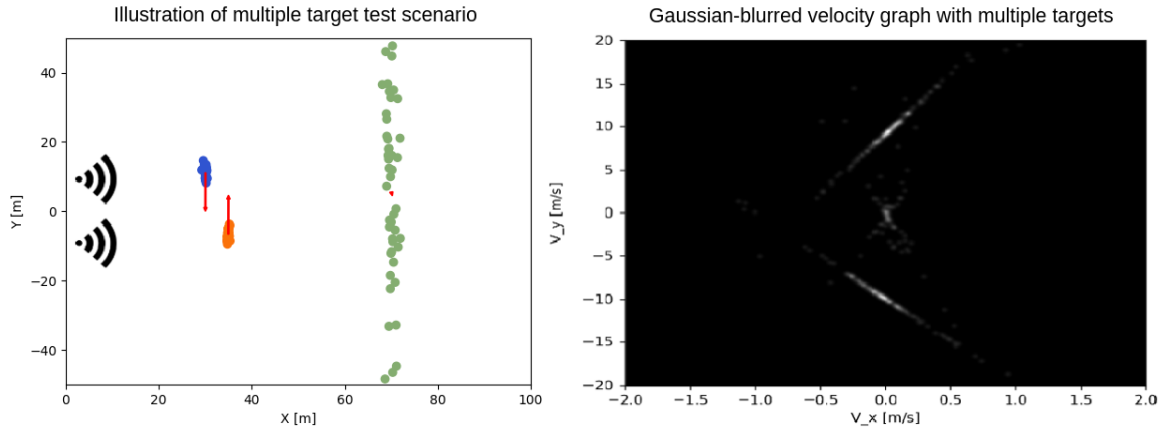

Figure 4.15: Simulation setup and corresponding Gaussian-blurred velocity graph for multiple targets.

To detect multiple objects simultaneously, the velocity graph has to be clustered. This is done by applying DBSCAN to the data, this results in a graph like 4.15. In addition, only combinations of points that are within a certain radius of each other are used to calculate velocities for the velocity graph. This is done to bring down the complexity of the calculation resulting in a significant decrease of required computational power. For the purpose of this research, a radius of 3 meters was chosen. This was done based on the size of targets present in the RadarScenes dataset (e.g. cars and pedestrians). See 4.16 for the resulting velocity graph.

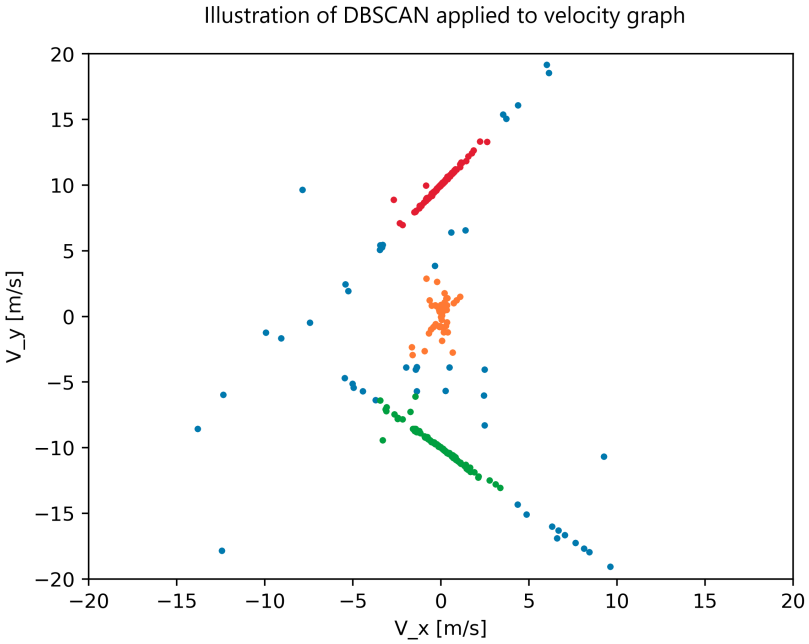

Figure 4.16: Example of DBSCAN applied to the velocity graph with multiple objects, different colors indicate different clusters.

While creating the velocity graph, a list of point combinations and their resulting velocities is saved. This way, clusters in the velocity graph can be used to find the matching points in the original point cloud and cluster them together. This way the algorithm is not only able to estimate target velocities, but also segment the original point cloud.

Once the clusters in the original point cloud are formed, DBSCAN is again applied to separate targets in the physical space. This is done because targets that have a similar velocity vector will end up

in the same cluster in the velocity graph.

As can be seen in the velocity graph in Figure 4.16, not all points are assigned to an object. These velocities are not close enough to any cluster to be considered part of them. By also not using these points to label the original point cloud, it is possible to filter out points that are too noisy. An example of this filtering is shown in Figure 4.17. Here, velocities that fit together to form a consistent full velocity vector are labeled as *core points*. By changing the criteria set in de DBSCAN algorithm, the strictness of this filtering can be changed.

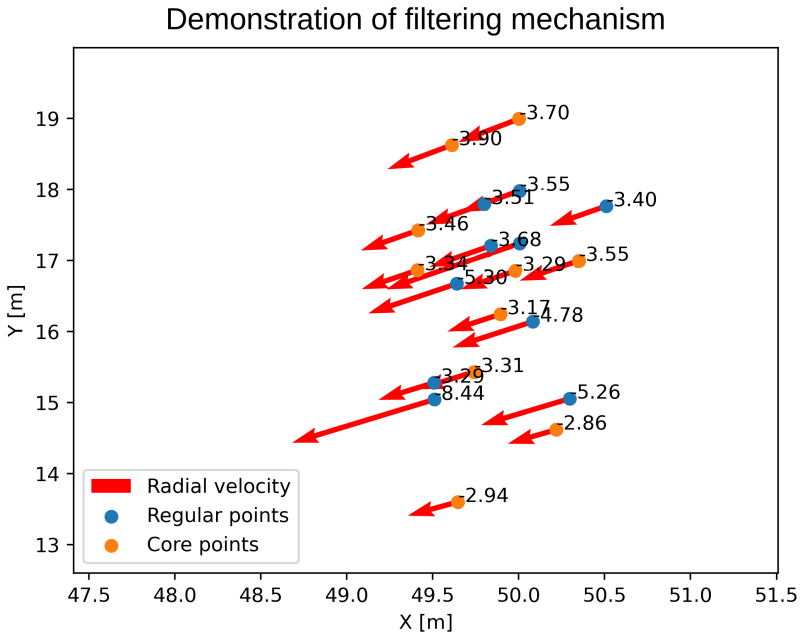

Figure 4.17: Illustration of how the filtering mechanism in the velocity graph approach functions. Points that are consistent with each other are marked in orange, points that cause deviations from the assumed velocity vector are marked in blue.

Figure 4.18 shows the result of applying this method to a point cloud from RadarScenes. In this figure the effects of the filtering are also visible, filtered points are marked as gray circles with no smaller colored inner circle.

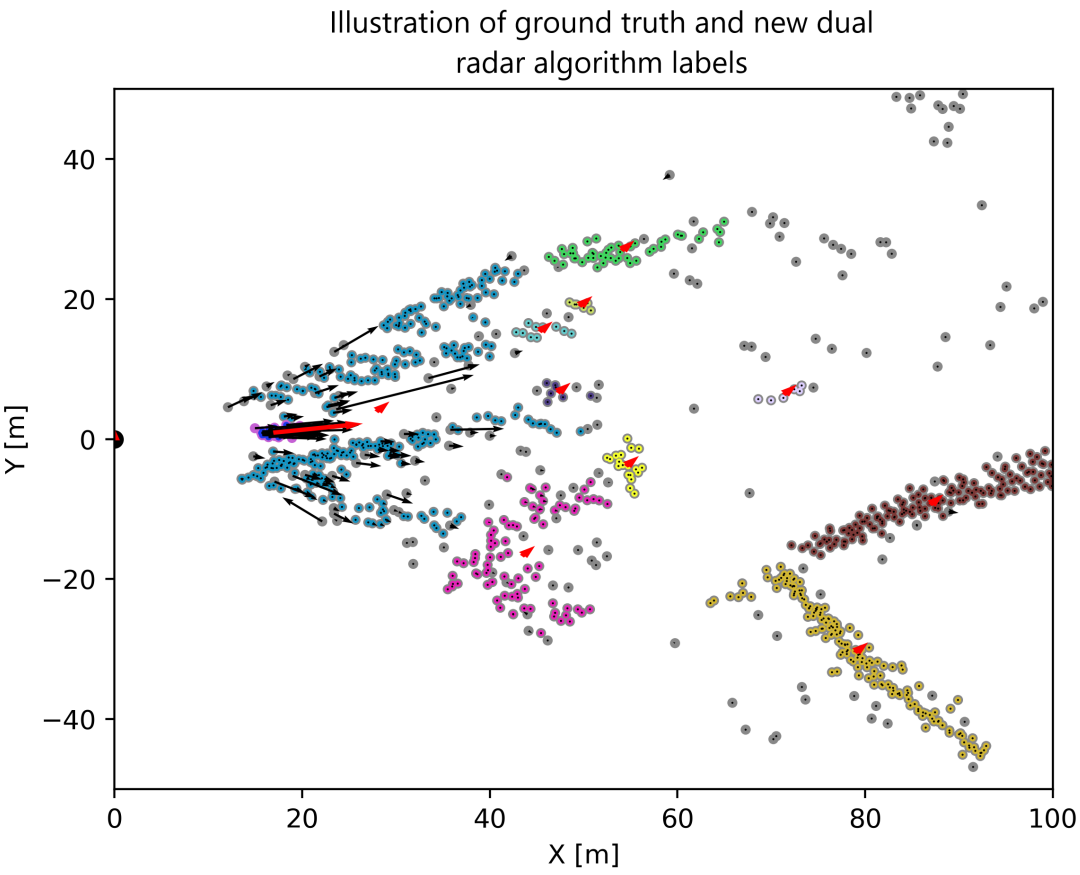

Figure 4.18: Ground truth labels are marked with large circles, velocity graph labels are marked with smaller circles. Points with no inner color are marked as noise by the algorithm, points with same color belong to the same cluster. Black arrows indicate measured doppler velocity, red arrows indicate estimated full velocity vector.

To summarize, the algorithm does three things:

- Cluster the point cloud based on full velocity vector.
- Assign full velocity vectors to the clusters.
- Filter out points that do not belong to clusters.

In Figure 4.19 the block diagram of the full algorithm is given.

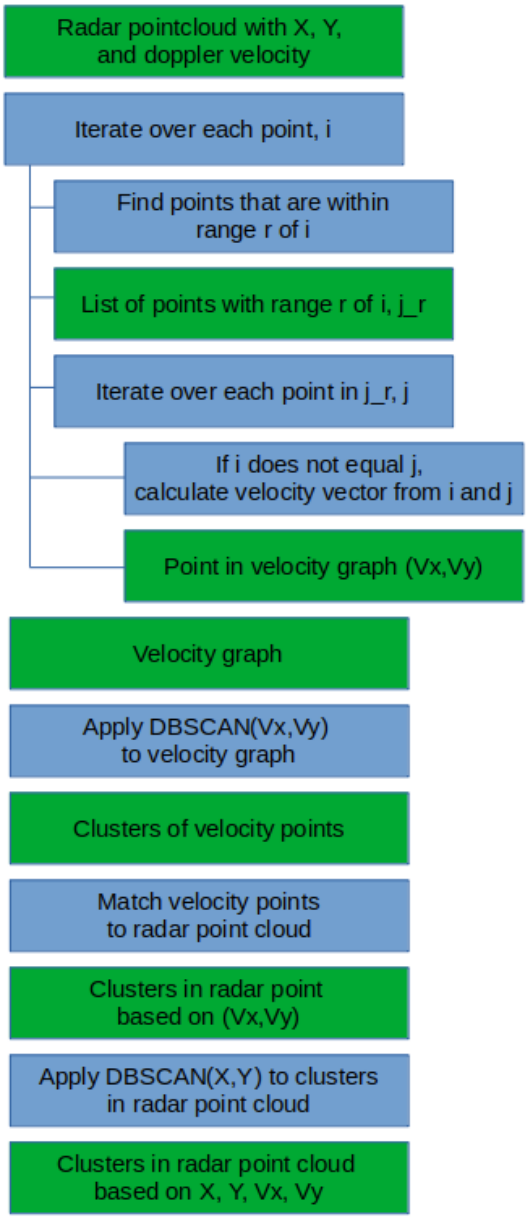

Figure 4.19: Block diagram of steps in velocity graph algorithm. Green indicates data, blue indicates a processing step.

This algorithm addresses the outlier problem described in 4.1 by using pairwise consistency between detections to cluster correct velocity vectors. This way a consensus can be reached about the approximate velocity vector, even in the presence of many conflicting measurements.

4.6. RadarGNN Integration

The final step is to convert the segmented point cloud into a graph so that in can be used in RadarGNN. This means it has to be converted from a point cloud, where each point has multiple properties, to nodes and edges, also with their own properties.

In the original RadarGNN, the graph nodes have the following properties:

- rcs, value for radar cross-section.
- $velocity\_vector$, X and Y components of the Doppler velocity.
- $time\_index$, time value of the measurement.

- $degree$, amount of edges to the node.

And the edges:

- $relative\_position$, position in relation to other connected node. This is required to make the model translation invariant.

Here, the edges are generated via a K-Nearest Neighbours (KNN) algorithm. This means a fixed number of edges, 5 by default, is generated at random to points that are within a certain radius, see the left side of Figure 4.20 for an example.

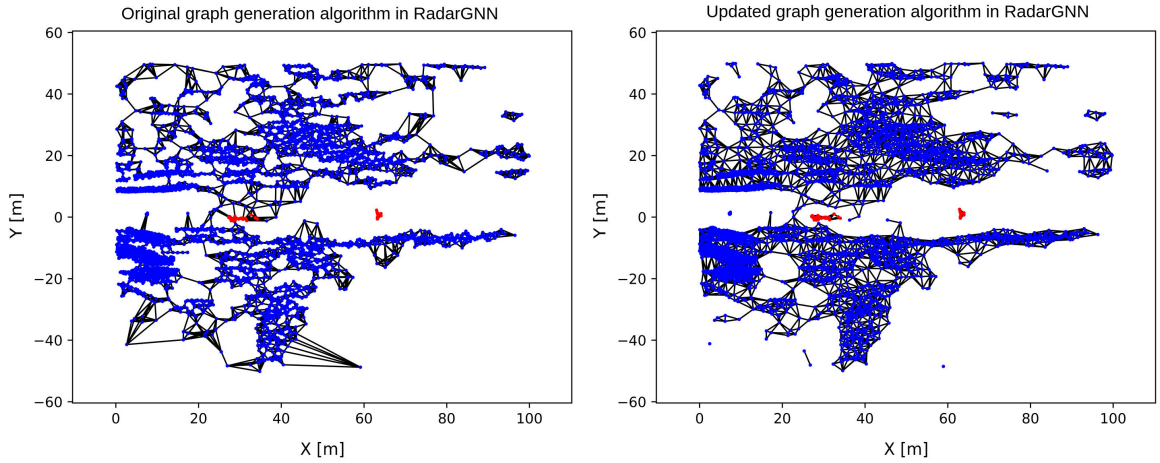

Figure 4.20: Original and updated graph generation methods. The new method creates edges between points with a similar full velocity vector, the old method generates these randomly. Static environment is marked in blue, vehicle is marked in red.

For the implementation of the full velocity vector, node features were kept mostly the same as in the original RadarGNN. The only change made was that the $velocity\_vector$, which are X and Y components of the Doppler velocity, was replaced by the X and Y components of the full velocity vector. The edge features were not changed.

To generate the edges, the KNN algorithm was changed out for Delaunay triangulation. In the new method, points that are within a given radius and have the same full velocity vector are connected. This is done to more effectively implement the knowledge gained by calculating the full velocity vector. This new implementation is shown on the right side in Figure 4.20.

4.7. Ground truth generation

Since the RadarScenes dataset does not include ground truth data for the velocity vector of objects, and alternative had to be found to generate this data. To do this, the baseline method selected in chapter 3 was taken as a starting point. This method is proven to work reliably and accurately when given data with few outliers. Fortunately, RadarScenes does provide point-wise labeling, which can be given to the baseline method. This way accurate velocity information can still be generated without it being available in the dataset itself. Even though the labeling is near-perfect, the measurements are of course not. Because of this, this method sometimes generates (large) errors in situations where the measurement data is of very poor quality.

4.8. Proof

Let's first assume that any of the measured parameters consist of two parts, the true value and a noise value. This approach makes it possible to study how noise in the measurements affects the final calculated velocity components.

$$V_{r1} = V_{r1}^{true} + V_{r1}^{noise} \tag{4.3}$$

$$V_{r2} = V_{r2}^{true} + V_{r2}^{noise} \tag{4.4}$$

$$\theta_1 = \theta_1^{true} + \theta_1^{noise} \tag{4.5}$$

$$\theta_2 = \theta_2^{true} + \theta_2^{noise} \tag{4.6}$$

$$r_1 = r_1^{true} + r_1^{noise} \tag{4.7}$$

$$r_2 = r_2^{true} + r_2^{noise} \tag{4.8}$$

4.8.1. Full Velocity Vector

The radial velocity components are projections of the full velocity vector onto the directions of each radar. These can be expressed in terms of the full velocity vector as follows:

$$\begin{bmatrix} V_{r1} \\ V_{r2} \end{bmatrix} = \begin{bmatrix} cos(\theta_1) & sin(\theta_1) \\ cos(\theta_2) & sin(\theta_2) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} \tag{4.9}$$

We can now invert this equation to compute the full velocity vector from the radial components:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} cos(\theta_1) & sin(\theta_1) \\ cos(\theta_2) & sin(\theta_2) \end{bmatrix}^{-1} \begin{bmatrix} V_{r1} \\ V_{r2} \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \frac{1}{cos(\theta_1)sin(\theta_2) - sin(\theta_1)cos(\theta_2)} \begin{bmatrix} sin(\theta_2) & -sin(\theta_1) \\ -cos(\theta_2) & cos(\theta_1) \end{bmatrix} \begin{bmatrix} V_{r1} \\ V_{r2} \end{bmatrix} \tag{4.11}$$

Using the identity

$$cos(\theta_1)sin(\theta_2) - sin(\theta_1)cos(\theta_2) = sin(\theta_2 - \theta_1) \tag{4.12}$$

the expression can be simplified:

$$\vec{V} = \frac{1}{sin(\theta_2 - \theta_1)} \begin{bmatrix} sin(\theta_2) & -sin(\theta_1) \\ -cos(\theta_2) & cos(\theta_1) \end{bmatrix} \vec{V_r} \tag{4.13}$$

4.8.2. Noise Sensitivity

It is now possible to analyze how uncertainty, or noise, in each measurement influences the calculated full velocity vector. Taking the derivatives with respect to $V_{r1}^{noise}, V_{r2}^{noise}, \theta_1^{noise}, \theta_2^{noise}, r_1^{noise}, r_2^{noise}$, starting with V_{r1}^{noise}:

$$\frac{\partial \vec{V}}{\partial V_{r1}^{noise}} = \begin{bmatrix} sin(\theta_2)\frac{1}{sin(\theta_2 - \theta_1)} \\ -cos(\theta_2)\frac{1}{sin(\theta_2 - \theta_1)} \end{bmatrix} \tag{4.14}$$

This derivative describes the direction and magnitude of the change in $\vec{V}$ cause by a noise in V_{r1}. Interestingly, this direction is constant for any θ_2.

The slope of the this line, a, can be calculated:

$$a = -\frac{cos(\theta_2)}{sin(\theta_2)} \tag{4.15}$$

Following the same procedure for V_{r2}^{noise} yields:

$$a = -\frac{cos(\theta_1)}{sin(\theta_1)} \tag{4.16}$$

Where the direction of change in $\vec{V}$ is constant for any θ_1.

Doing the same for θ give the same result as for the noise in radial velocity:

$$\frac{\partial \vec{V}}{\partial \theta_1^{noise}} = \begin{bmatrix} \frac{1}{sin(\theta_1-\theta_2)}\left(\frac{cos(\theta_1-\theta_2)}{sin(\theta_1-\theta_2)}(V_{r1}sin(\theta_2) - V_{r2}sin(\theta_1)) + V_{r2}cos(\theta_1)\right) \\ \frac{1}{sin(\theta_1-\theta_2)}\left(\frac{cos(\theta_1-\theta_2)}{sin(\theta_1-\theta_2)}(V_{r2}cos(\theta_1) - V_{r1}cos(\theta_2)) + V_{r2}sin(\theta_1)\right) \end{bmatrix} \tag{4.17}$$

$$\frac{\partial \vec{V}}{\partial \theta_1^{noise}} = \begin{bmatrix} cos(\theta_2)\frac{1}{sin(\theta_1-\theta_2)^2}(V_{r2} - V_{r1}cos(\theta_1-\theta_2)) \\ -sin(\theta_2)\frac{1}{sin(\theta_1-\theta_2)^2}(V_{r2} - V_{r1}cos(\theta_1-\theta_2)) \end{bmatrix} \tag{4.18}$$

Again, the direction of change of $\vec{V}$ is constant for any value of θ_2. The slope can thus also be calculated in the same manner.

$$a = -\frac{cos(\theta_2)}{sin(\theta_2)} \tag{4.19}$$

Following the same procedure for θ_2^{noise} yields:

$$a = -\frac{cos(\theta_1)}{sin(\theta_1)} \tag{4.20}$$

Finally, examining how range noise affects the velocity vector, we find:

$$\frac{\partial \vec{V}}{\partial r_1^{noise}} = \vec{0} \tag{4.21}$$

$$\frac{\partial \vec{V}}{\partial r_2^{noise}} = \vec{0} \tag{4.22}$$

This means that noise in the range value does not influence the calculated velocity, which makes sense since r is also not present in Equation 4.9.

4.8.3. Geometrical Interpretation

To further understand what this means, lets look at the general formula for a first order polynomial:

$$y = ax + b \tag{4.23}$$

A line tangential to this line can be formulated as:

$$y = -(1/a)x + b \tag{4.24}$$

Now take the angle between the line and the X-axis, θ, and write the gradient of the line as a function of this angle:

$$a = \frac{sin(\theta)}{cos(\theta)} \tag{4.25}$$

For the tangent line this is then:

$$a = -\frac{cos(\theta)}{sin(\theta)} \tag{4.26}$$

Which is the same as Equations 4.15, 4.16, 4.19 and 4.20. This proves that for a variation in the measured radial velocity or angle of one point, the calculated velocity coordinates form a line at an angle tangent to the measured physical angle of the other point as visualized in Figure 4.13.

5

Experiments

In the previous chapter the concept and mathematical foundation for the velocity graph algorithm were presented. In this chapter, these findings will first be validated in a simulated environment. The velocity graph will then be tested against the baseline method and itself, but only making use of a single radar. The experiments are designed to compare these methods in situations where they are expected to behave differently from one another. This means that some of the testing conditions are sometimes unlikely to occur in real-world road conditions, but it will also clearly show the limits of these methods. Lastly, the classification results after integrating the full velocity vector into RadarGNN will be presented.

5.1. Method Validation

This method is compared to three other methods:

- RANSAC and overdetermined system of equations
- velocity graph with only one radar, but same amount of points
- velocity graph with only one radar, half the amount of points

The methods described above were tested at different distances. In the simulation, a car driving across the field-of-view at 30m, 50m, 70m and 90m were simulated. In addition, for every distance, ten different amounts of outlier percentages were added to the target to test robustness to noise. These percentages ranged from 0% to 90%, with 10% points increases. The amount of inlier points were kept constant, this means the total amount of points increases with the increase in outliers.

In Figure 5.1, the test results for 30m and 90m are shown. The average and variation of the full set of simulations can be found in Appendix A.

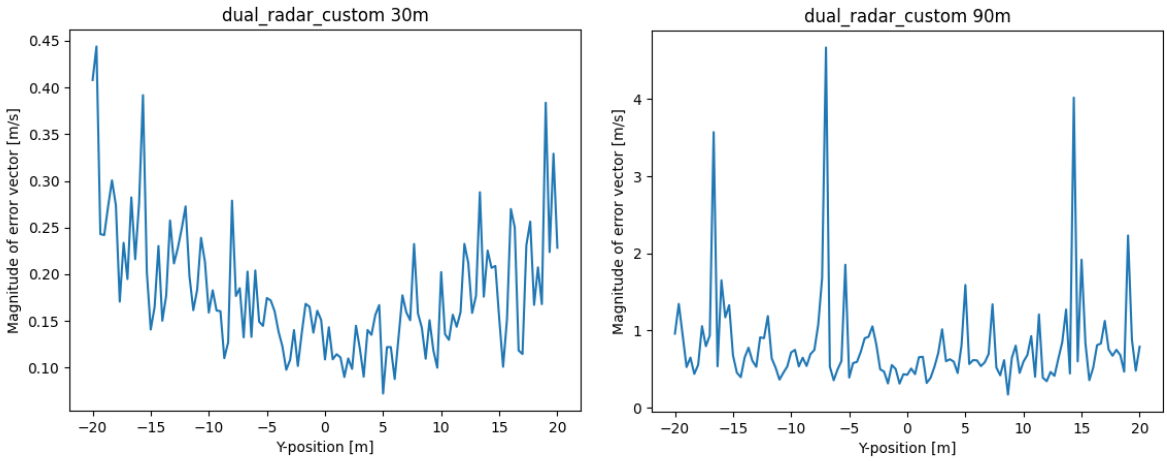

Figure 5.1: Comparison of the single target velocity graph method with a simulated target passing at 30m and 90m distances in front of the radar setup.

From these two graphs, already a few observations can be made:

- Unlike the RANSAC method, performance of the velocity graph method is much less dependent on the percentage of outliers.
- Performance seems to be best exactly in front of the two radars. Since this effect is also visible for the single radar experiments, this is most likely not a result of the crossing-radar-lines-effect. Therefore it is most likely caused by target being further away due to the vertical distance component.
- This effect disappears at greater distances. This is because the distance is now more or less the same, independent of lateral position.

In the following figures key findings from the full set of experimental data are presented. In Figure 5.2 the previously mentioned methods are compared in relation to their robustness to outlier percentage. Here it becomes clearly visible that the new dual radar method outperforms the RANSAC method in terms of robustness. Another thing to note is that using two radars outperforms using a single radar, but that this effect is mostly caused by having double the amount of data points to work with. This is confirmed by also running the simulation with the viewpoint from only a single radar, but by using the same number of points as would be produced by two radars. In the figure below, this setup has nearly the same performance as a dual radar setup.

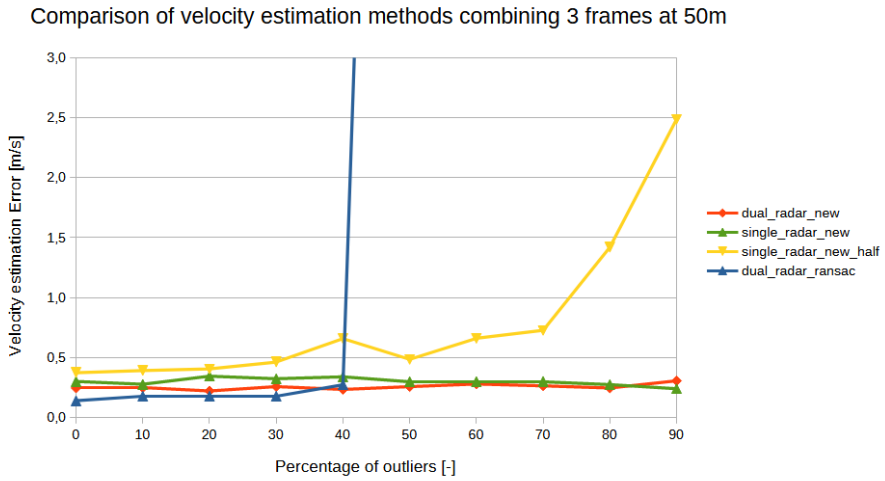

Figure 5.2: Comparison of robustness to noise of different methods. In red the velocity graph method with two radars, in green the velocity graph method with one radar but with the same amount of points, in yellow the velocity graph method with a single radar, and lastly in blue the baseline RANSAC method using two radars.

Figure 5.3 compares the combining of different amounts of frames into a single point cloud. This experiment was performed to test the effects of this mechanism as it is used by the RadarGNN model. As shown in Figure 5.2, the new velocity graph algorithm is barely affected by an increase in outliers. This seems to be the case as long as there is a minimum amount of inliers available, once this number drops below a certain threshold the performance drops. This minimum number of inliers can be controlled by combining multiple frames together, when this minimum cannot be obtained from a single frame.

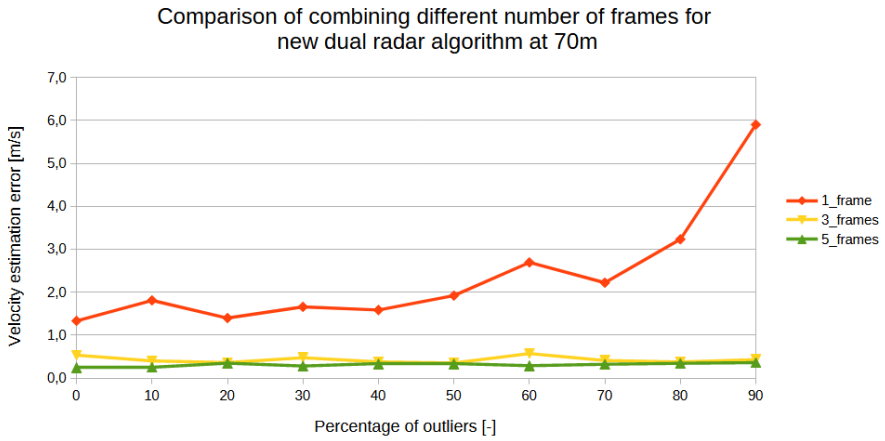

Figure 5.3: Comparison for the velocity graph algorithm using different amount of frames. By combining frames the data sparsity problem can be addressed. Here it is visible the method significantly benefits from a denser point cloud.

Lastly, Figure 5.4 shows the effect of increased distance. Again, the performance is not significantly affected by the increase in outliers. The performance does however degrade with an increase in distance, likely because of the decrease in total points in the point cloud.

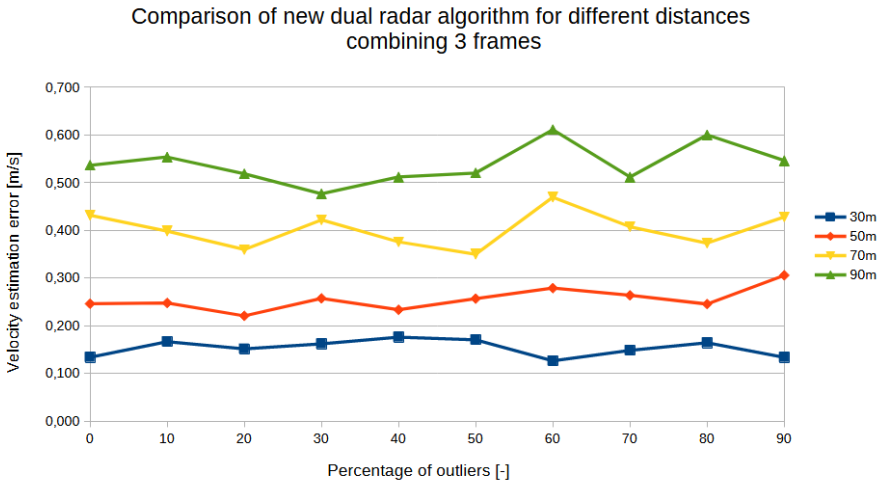

Figure 5.4: Comparison of the velocity graph method with different outlier percentages, and at different distances. Accuracy increases when the target it closer due to the denser point cloud.

To summarize, from this section the following can be concluded:

- The velocity graph method is not significantly affected by an increase in outliers, as long as there is a minimum number of inliers available.
- The increase in performance from using two radars is mainly due to having a twice as large point cloud. Though another roughly 10% performance increase is a result of the crossing-radar-lines-effect.
- Combining multiple radar frames helps increase the number of inliers, extending the effective range of the sensors.

These are the results from the experiments by using the method described at the beginning of this chapter. It is expected that by using a method that exploits the crossing-radar-lines-effect will increase performance. Doing so should also increase the gap between using a dual radar setup, and a single radar with the same size point cloud.

5.2. Parameter Optimization on RadarScenes

To tune the algorithm for the radar setup used in RadarScenes, a grid search on the most important parameters was performed. These parameters are the eps and $min\_samples$ inputs for the DBSCAN algorithm in the velocity graph. They determine the permitted error in the velocity vector during the forming of clusters, and the minimum amount of points needed in the velocity graph to form a cluster respectively. In other words, they determine how noisy a point is allowed to be, and the minimum number of points that are needed to label something as an object. From the scikit-learn documentation[35] on DBSCAN:

- eps, float, default=0.5
  The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.
- $min\_samples$, int, default=5
  The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself. If $min\_samples$ is set to a higher value, DBSCAN will find denser clusters, whereas if it is set to a lower value, the found clusters will be more sparse.

First a rough estimate of the parameters was determined to start the grid search. This was done based on trial and error. This resulted as a value of 1 for eps and a value of 50 for $min\_samples$ as a starting point for the grid search. Values of 0.5, 1, and 2 were tested for eps. For $min\_samples$, 25, 50, 75, and 100 were tested.

Additionally, RadarGNN was also configured in such a way that data from roughly three frames were combined into a single frame, since experiments showed that any number above three resulted in diminishing returns.

Performance was tested by evaluating the precision, recall, and accuracy of the velocity estimation. Figure 5.6, 5.5, and 5.8 show these results respectively.

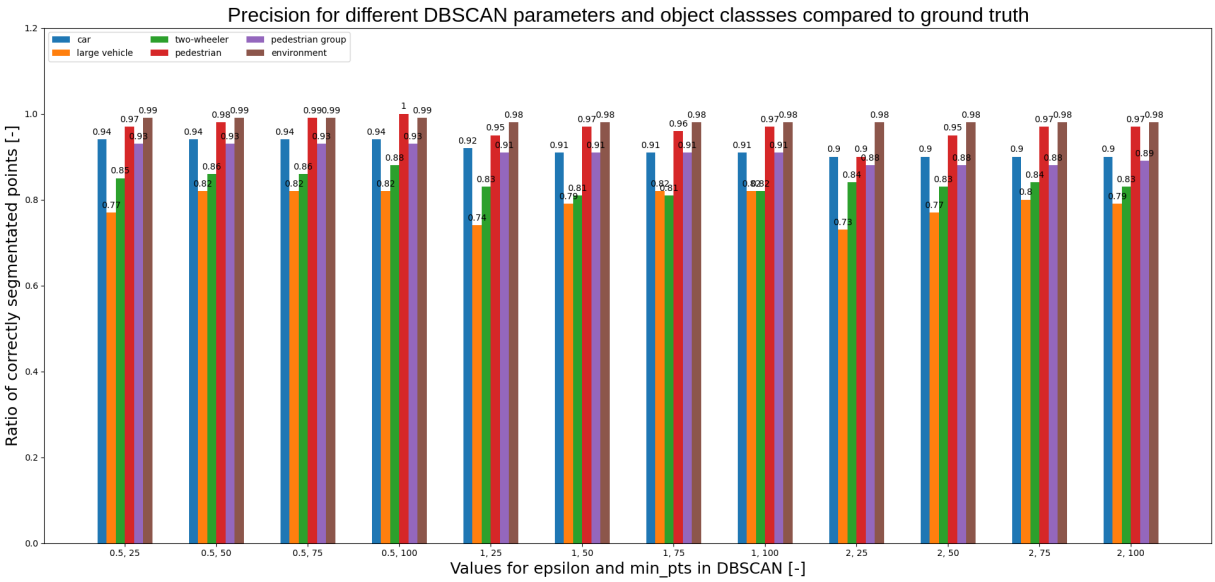

Figure 5.5: Precision for different combinations of parameter values. Higher is better.

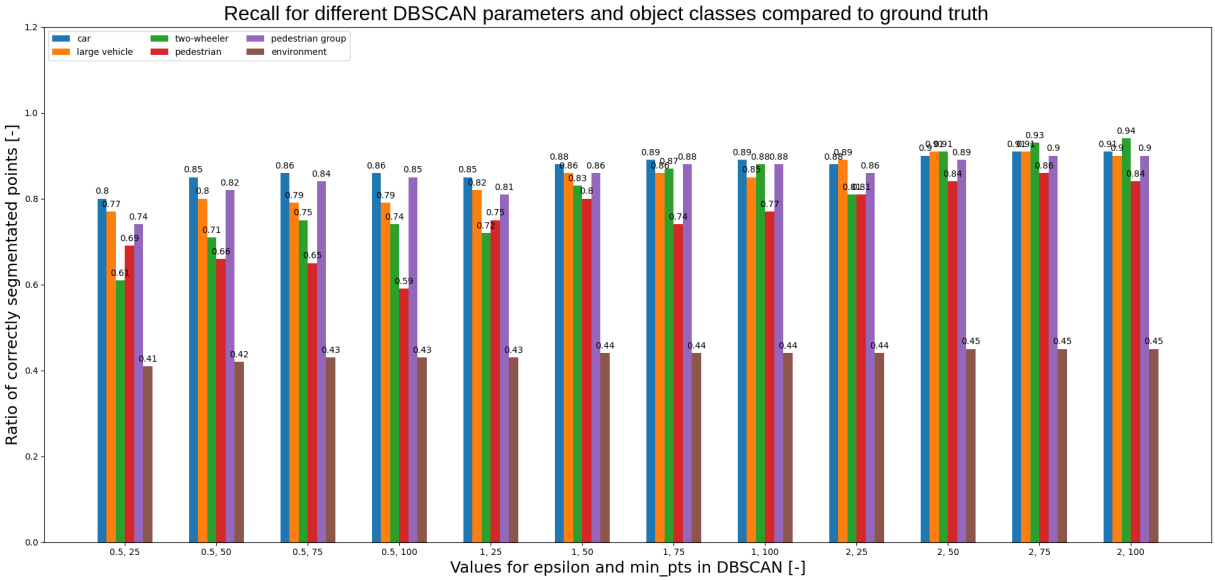

Figure 5.6: Recall for different combinations of parameter values. Higher is better.

During the evaluation of the velocity vector performance, it became clear that in some cases the ground truth velocity estimation does not always produce a good estimate. This is the result of low

quality data in some of the frames in RadarScenes. The magnitude of the error between the estimated vector and the ground truth vector is shown in Figure 5.7.

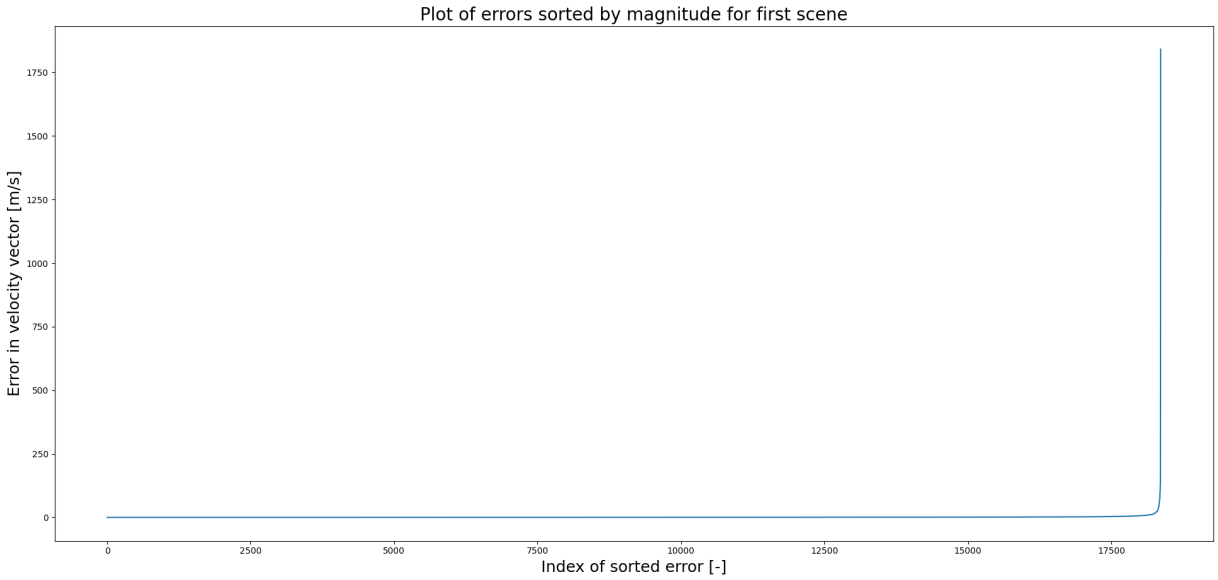

Figure 5.7: Error in velocity vector estimation compared to ground truth sorted by magnitude.

To still create a fair comparison, different metrics were taken over different percentiles of data. This way the large errors caused by low quality data do not skew the comparison. In Figure 5.8 the results are visualized.

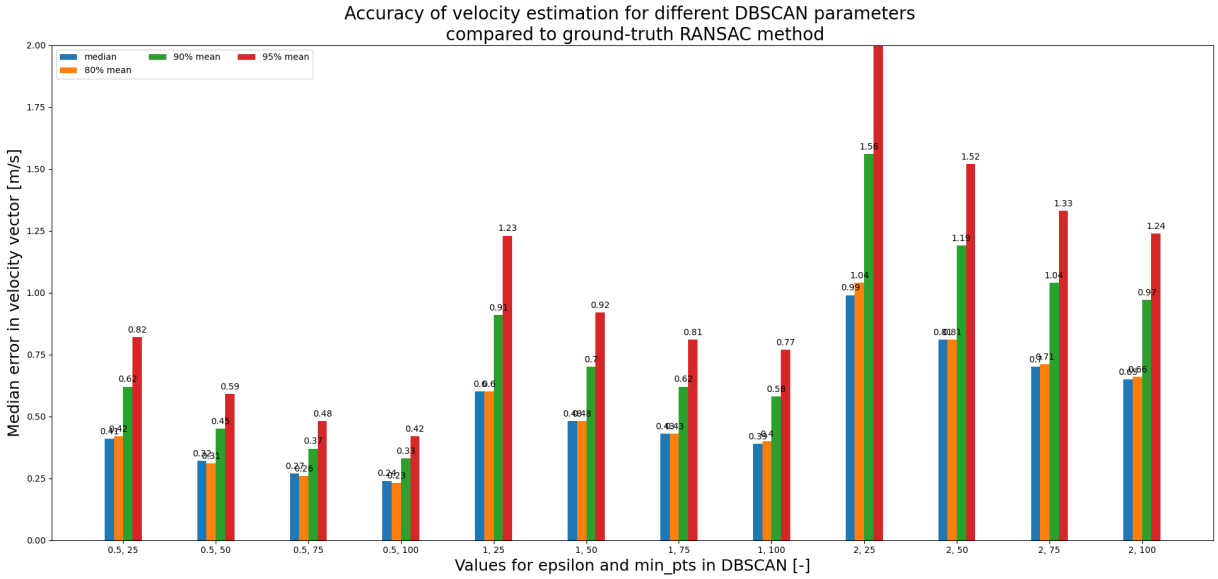

Figure 5.8: Magnitude of error in velocity vector. Lower is better.

5.3. Classification

Considering the extra conditions added to the test environment compared to the original RadarGNN paper, the model was first retrained under these conditions to create a baseline comparison. This model was then compared to a model trained with ground truth estimation, and to the model trained with the velocity graph method.

The training was done using the parameters in the Table 5.1. These are the default parameters from the RadarGNN project.

Table 5.1: Table with parameters used to train the RadarGNN model.

MODEL_ARCHITECTURE	
node_feature_dimension	3
edge_feature_dimension	2
conv_layer_dimensions	[224, 224, 128, 64, 32]
classification_head_layer_dimensions	[6]
regression_head_layer_dimensions	[16, 5]
initial_node_feature_embedding	True
initial_edge_feature_embedding	True
node_feature_embedding_layer_dimensions	[32, 64, 128, 224]
edge_feature_embedding_layer_dimensions	[4, 8, 16]
conv_layer_type	”MPNNConv”
batch_norm_in_mlps	False
TRAINING	
dataset	radarscenes
bg_index	5
learning_rate	0.001
epochs	30
batch_size	5
shuffle	True
deterministic	True
seed	123
exponential_lr_decay_factor	0.95
bb_loss_weight	0.5
regularization_strength	0.000005
adapt_orientation_angle	True

The resulting confusion matrices can be seen in Figure 5.9.

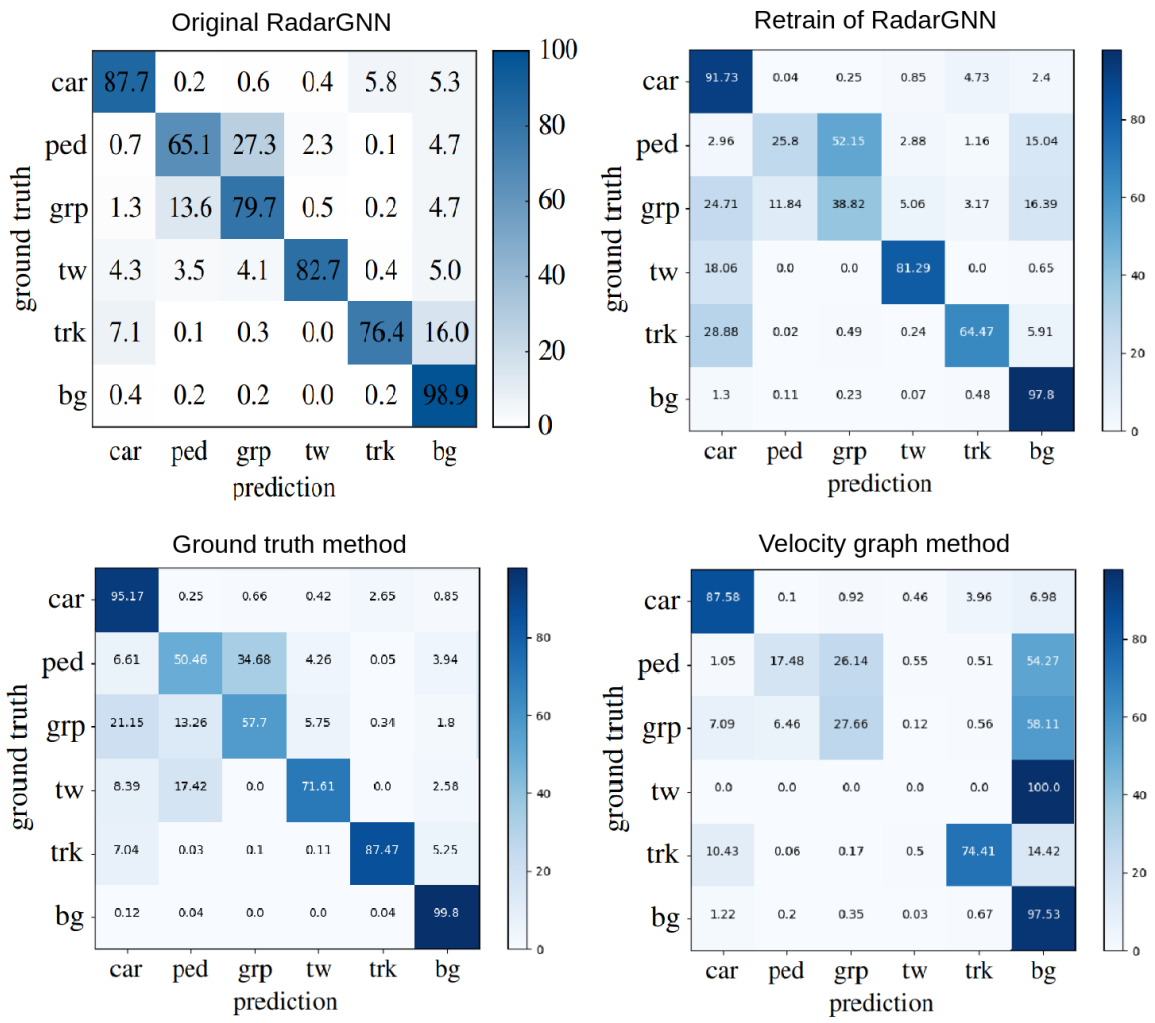

Figure 5.9: Confusion matrices of different models.

From this comparison, a couple observations can be made.

- The retrained model performs worse than the original from the paper.
- The ground truth method performs significantly better than the retrained original model.
- The velocity graph method performs poorly.

Considering the velocity graph method performed well in previous tests, improvements in classification performance were expected. Especially since the ground truth method does show an increase in performance.

There could be several origins of this under-performance. The first options is the data that the model uses to learn. Due to the modifications made, this data could simply not be enough in quantity, or the class balance could be disrupted due to the cropping. Secondly something could have gone wrong in the training of the model, the model might be too complex, or the loss functions might not be properly formulated. Lastly, the velocity graph algorithm might not perform as expected. After some consideration, it was deemed worthwhile to investigate the following possible causes:

- Overfitting due to smaller dataset.
- Class unbalance due to cropping of field-of-view or filtering out points that do not belong to clusters.
- Low frequency but high magnitude errors cause poor learning.

In the next section, each of these possible reasons for the poor performance will be addressed. At the end of the section a new model will be evaluated which incorporates attempted fixes for the problems mentioned in this section.

5.4. Improvements

In this section, some areas of concern will be discussed. At the end of the section a new updated model that attempts to solve these concerns will be presented.

5.4.1. Overfitting

As can be seen from the previous section, performance is not quite as expected. A quick look into the training curves as seen in Figure 5.10 showed that the model is likely overfitting on the training data.

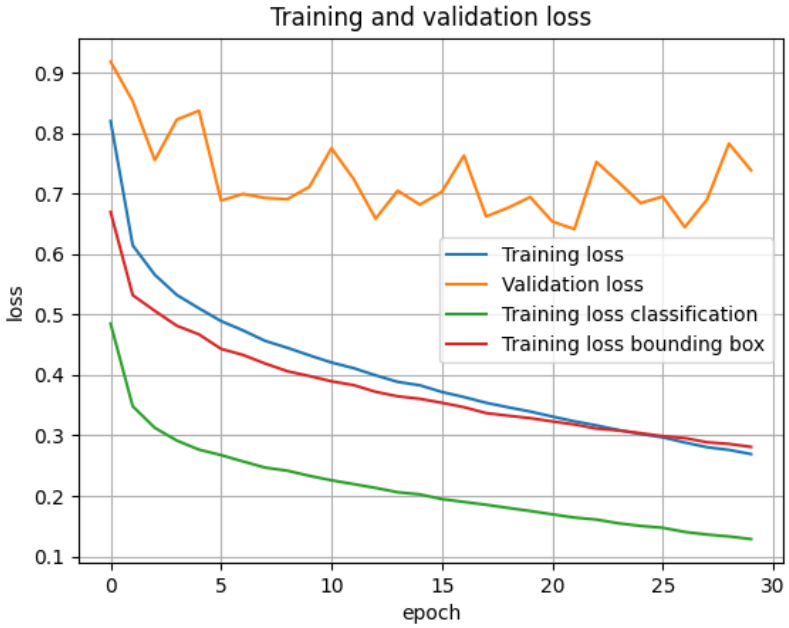

Figure 5.10: Loss curve of training original model on validation data.

Evaluating the model on the training data indeed shows that this is the case, in Figure 5.11 it is clearly visible that there quite a big performance gap in comparison to the results from the validation data.

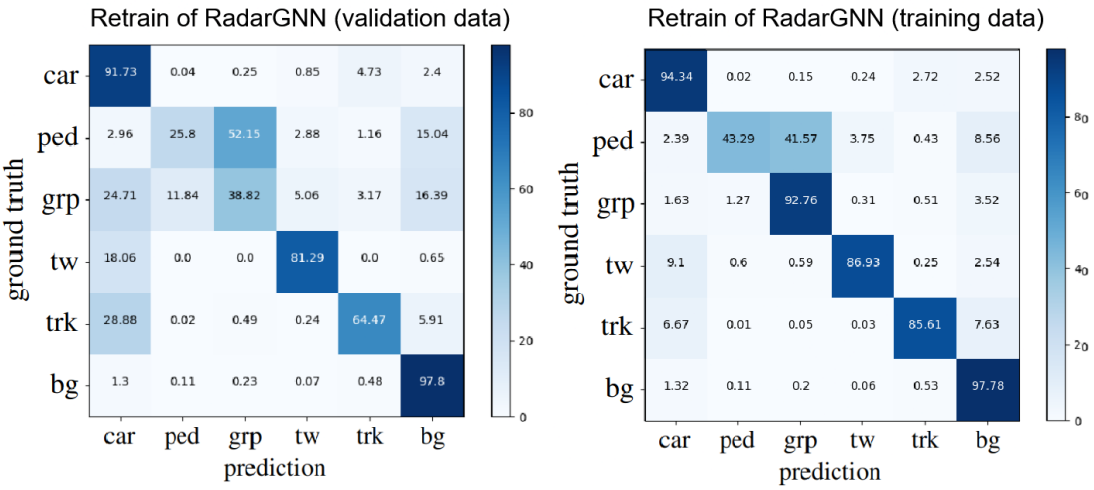

Figure 5.11: Confusion matrices for the original retrained model evaluated on the validation and training set respectively.

In Figure 5.10 it is visible that the training loss keeps dropping even after the validation loss stagnates. To hopefully reduce overfitting, in the next version a model will be selected from an earlier epoch.

5.4.2. Class Unbalance

The second possibility for the poor performance is sought in the class (un)balance of the dataset. Due to the selection of the smaller subset of 64 scenes, the cropping of the field-of-view, and the filtering of points that do not belong to a cluster, it is possible for the classes to become unbalanced in the training data.

To gain insight in this problem, the dataset was processed and metrics for each class were saved. In Figure 5.12 the data from the velocity graph method with and without the filtering feature are compared to the orignal values from the RadarScenes paper. The classes are expressed as a ratio of the total amount of point labels, where the sum of the classes equals one.

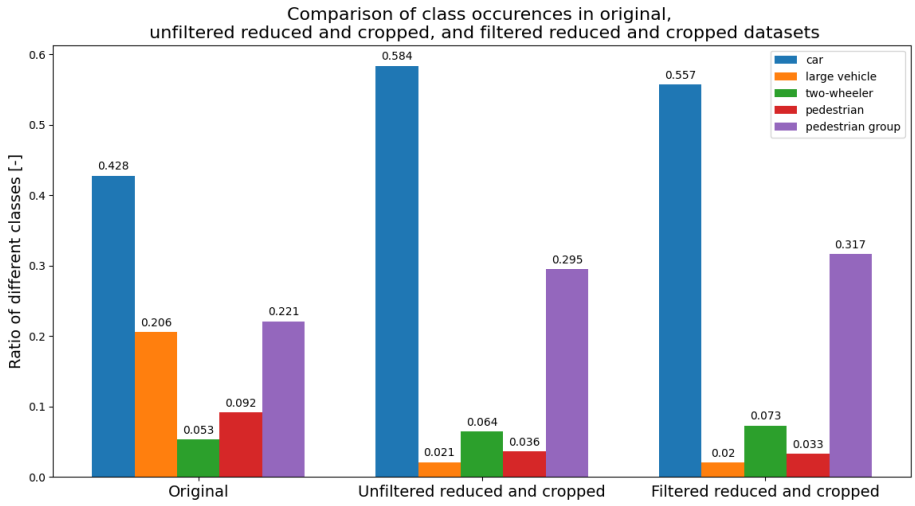

Figure 5.12: Percentages of label occurrences for different classes in different processing methods. Original RadarScenes dataset on the left, forward cropped version with selected 64 scenes in the middle, and forward cropped filtered version of the selected 64 scenes on the right.

Looking at these percentages, it is deemed possible that the observed unbalance might be cause

for the poor performance. This is especially the case for the pedestrian classes, since they are already under represented in the original dataset.

5.4.3. Large Errors

The last possible cause for the poor performance might be that large errors in the velocity estimation cause the training of the network to be disturbed. As can be seen in Figure 5.7, a small percentage of estimations have massive errors. To prevent this, a limit was set on the maximum magnitude of the velocity vector. For the purpose of this test, this limit was set at 25m/s.

5.4.4. Updated Model

With all the improvements suggested in the previous section implemented, the model was once again retrained. This led to the model as picture in Figure 5.13.

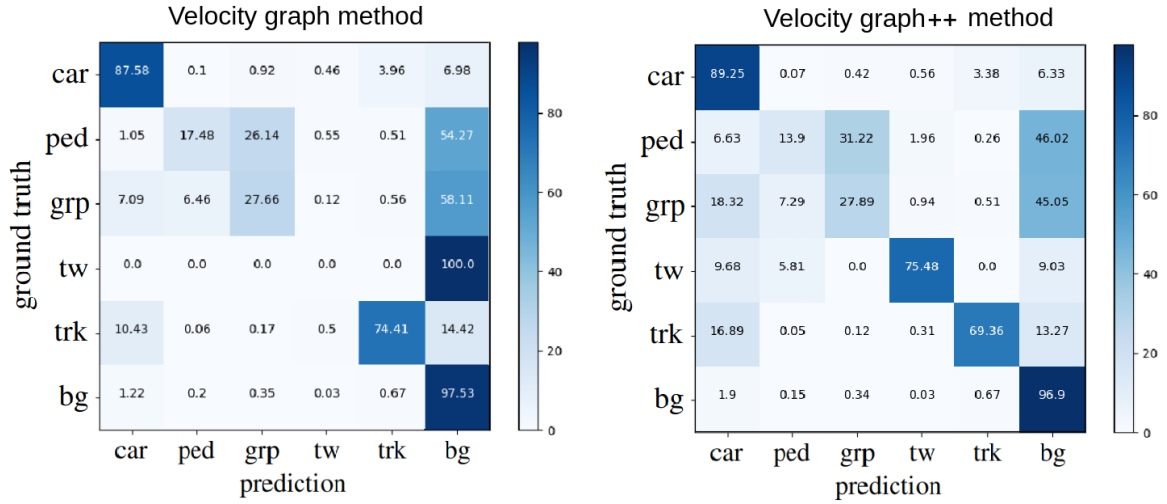

Figure 5.13: Comparison of velocity graph models with and without further improvements.

5.5. Conclusion

In the last part of this study, the four most relevant versions of the dual radar classification models are evaluated by analyzing the F1 and mAP scores for each model. These models are the original re-trained model on both the validation and training data, the model incorporating the ground truth velocity information, and the model using the velocity graph with further improvements. The results are given below in Table 5.2.

Table 5.2: F1 and mAP scores for the different models for different classes.

		original_validation	original_train	ground_truth	velocity_graph++
F1	Car	0.75	0.83	0.93	0.71
	Pedestrian	0.23	0.39	0.50	0.11
	Pedestrian group	0.14	0.67	0.45	0.08
	Two wheeler	0.06	0.82	0.19	0.10
	Large vehicle	0.68	0.78	0.91	0.70
	Background	0.99	0.99	1.00	0.98
mAP	Car	0.68	0.75	0.79	0.71
	Pedestrian	0.06	0.11	0.21	0.02
	Pedestrian group	0.05	0.67	0.16	0.03
	Two wheeler	0.05	0.53	0.40	0.16
	Large vehicle	0.63	0.74	0.82	0.65

From the results it can be concluded that including the full velocity vector information significantly improves the classification performance. For example, the model trained with ground truth velocity information showed consistent improvements among several classes, most notably for ”car” and ”large vehicle” with F1 scores increasing from 0.75 to 0.93 and from 0.68 to 0.91, respectively.

It also shows that the current models are most likely over fitted on the training data, since the model evaluated on validation data performs significantly worse than the model evaluated on the training data. This is especially the case for the ”pedestrian”, ”pedestrian group” and ”two wheeler” classes.

However, the integration of the velocity graph algorithm did not show the expected improvements. Despite using an improved version of the model, the performance did not show any substantial gains. For example, the F1 score for ”car” decreased from 0.75 (original validation) to 0.71 (velocity graph++), and the mAP for ”large vehicle” showed an increase from 0.63 to 0.65.

6

Conclusion

In this last chapter, the main motivations, methods, and findings of this thesis are presented. It reflects on how the original research questions were answered, and provides the reader with the key contributions that are made. Finally, it will provide potential directions for future research.

6.1. Motivation and Findings

Reliable perception is critical in safe operation of autonomous vehicles. While radar offers some distinct advantages over other types of sensors, its sparse data and lack of the ability to directly measure an objects full velocity have limited its use. This thesis is aimed at addressing these problems and improving full velocity vector estimation by using a dual radar setup, with the broader goal of making radar a competitive alternative to other sensors like lidar and cameras.

To tackle these problems, a novel *velocity graph* algorithm was introduced. This method makes use of pairwise consistency between radar detections to estimate the 2D motion of objects in a robust manner, even in conditions of heavy noise. A simplified radar point cloud simulation environment was created to test this method in controlled conditions. In addition, a ground truth velocity method for the RadarScenes dataset was created, allowing for validation in real-world conditions. The full velocity vector information was then integrated into the RadarGNN model to quantify the effects of improved velocity information on classification tasks.

The proposed method yielded several important results:

- Improved velocity estimation: The velocity graph method significantly outperforms traditional RANSAC-based velocity profile fitting methods in simulation, specifically in conditions where the percentage of outliers is above 40%.

- Better clustering and filtering: By clustering detections based on consistent velocity vectors, the method was able to simultaneously segment the scene and filter out noisy points.

- Gains in classification performance: Integrating ground truth full velocity vectors into RadarGNN led to improved object classification (i.e. mAP for cars from 0.68 to 0.79), though this same result could not be achieved using the velocity graph method.

These conclusions directly answer the research questions stated at the beginning of this thesis; the failure cases of current methods were analyzed and addressed, dual radar setups were shown to reliably estimate full velocity vectors through the proposed velocity graph method, and lastly, accurate full velocity vector information, though using the ground truth, improved classification performance, showing the wider application of this work.

6.2. Future Work and Recommendations

While this thesis showed the effectiveness of the velocity graph method and its wider use cases, several directions of research are still left to be explored further.

The current method assumes negligible rotational velocity of targets. Expanding the algorithm to include rotational velocity would be an interesting addition. By using three equations per point instead of two, it should be possible to create a three-dimensional velocity graph.

Also, in this thesis DBSCAN was used as a basis for target detection in the velocity graph. Doing more research into different algorithms and clustering methods could also still significantly improve the clustering performance of the algorithm.

Another potential direction is to train a neural network that both performs clustering and velocity estimation simultaneously. Such a model could learn to recognize patterns in the velocity graph directly, potentially resulting in better performance.

In conclusion, this thesis demonstrated the usefulness of dual radar systems in improving velocity estimation and classification performance. With further research, these systems might prove a valuable complement to current sensor systems used in automotive applications.

A

Data

1 frame	dual_radar_custom				single_radar_custom				single_radar_custom_half				dual_radar_ransac			
Mean 95%	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)	30m (4.7)	50m (3)	70m (2.5)	90m (2*)	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)
Outliers - 0%	0.253	0.511	1.330	1.753	0.320	0.444	1.231	2.178	0.372	1.111	1.746	1.653	0.223	0.404	0.843	1.408
10%	0.210	0.535	1.808	1.807	0.275	0.621	1.286	2.466	0.524	1.257	1.493	1.651	0.218	0.318	0.529	199.366
20%	0.306	0.674	1.396	1.754	0.282	0.587	1.782	1.934	0.903	1.488	1.810	1.590	0.240	11.002	23.850	70.013
30%	0.277	0.625	1.657	1.989	0.307	0.844	1.276	2.499	0.690	1.506	1.765	1.841	0.268	22.388	148.122	311.788
40%	0.288	0.901	1.585	2.279	0.283	0.862	1.995	2.445	0.815	1.490	1.290	1.794	25.724	79.974	96.448	292.305
50%	0.270	1.020	1.919	2.822	0.384	0.758	1.472	2.353	1.213	1.615	1.525	2.121	30.190	66.014	356.999	391.331
60%	0.310	0.645	2.693	2.230	0.323	0.910	1.417	1.908	1.427	1.356	2.183	2.840	56.738	129.160	325.579	356.843
70%	0.278	0.952	2.220	3.316	0.280	0.794	2.408	3.041	1.463	2.263	2.183	2.907	120.159	165.189	381.541	518.952
80%	0.286	2.638	3.231	4.253	0.531	2.595	4.345	3.699	4.993	3.436	12.985	3.118	161.267	237.089	449.072	492.882
90%	0.402	4.184	5.904	7.113	0.386	3.301	7.841	7.826	5.935	11.221	14.257	7.035	163.094	275.741	418.238	493.076
Var 95%																
Outliers - 0%	0.043	0.200	2.394	3.688	0.057	0.194	1.487	4.049	0.094	1.673	5.282	3.788	0.035	0.133	1.602	4.017
10%	0.027	0.282	4.498	3.066	0.045	0.385	2.308	11.888	0.336	2.149	4.079	5.786	0.033	0.059	0.249	284,663.950
20%	0.052	0.458	2.651	2.894	0.046	0.285	4.342	3.736	2.447	5.015	5.239	7.866	0.052	2,124.699	12,625.303	52,047.996
30%	0.078	0.612	7.037	4.548	0.054	1.265	1.705	16.720	0.639	8.352	4.758	4.947	0.148	5,677.888	126,205.211	340,202.236
40%	0.044	2.033	3.052	9.749	0.067	0.795	9.902	11.943	1.738	5.952	3.530	6.012	6,916.507	32,636.835	54,832.620	338,886.109
50%	0.058	1.839	12.902	16.836	0.125	1.313	4.810	13.598	4.665	4.115	6.855	14.253	6,672.429	16,307.095	171,635.761	299,495.877
60%	0.072	1.038	16.228	12.270	0.070	1.745	3.206	8.714	9.783	3.299	18.124	19.210	10,111.487	44,218.169	150,890.768	220,463.651
70%	0.086	2.410	16.246	21.880	0.048	2.597	19.725	22.660	10.513	15.659	8.219	13.571	18,711.494	35,200.136	105,994.867	237,571.140
80%	0.095	29.185	27.222	44.467	1.502	20.445	49.398	22.519	68.135	26.130	100.595	22.065	18,644.746	35,650.090	96,938.484	159,611.677
90%	0.712	57.608	54.212	69.380	0.279	39.471	85.044	73.252	83.754	102.393	87.802	56.254	13,213.470	32,393.196	79,592.513	129,699.676

Figure A.1: Simulation results for different methods using a single frame.

3 frames	dual_radar_custom				single_radar_custom				single_radar_half				dual_radar_ransac			
Mean 95%	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)	30m (4.7)	50m (3)	70m (2.5)	90m (2*)	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)
Outliers - 0%	0.134	0.246	0.531	0.536	0.148	0.300	0.396	0.601	0.207	0.374	0.602	0.652	0.095	0.139	0.257	0.391
10%	0.166	0.248	0.398	0.554	0.183	0.277	0.362	0.578	0.245	0.391	0.602	0.827	0.098	0.178	0.228	0.291
20%	0.151	0.221	0.360	0.518	0.144	0.344	0.424	0.673	0.246	0.404	0.913	1.251	0.094	0.177	0.252	0.308
30%	0.162	0.257	0.472	0.477	0.160	0.323	0.482	0.482	0.262	0.462	1.165	1.709	0.093	0.177	0.215	0.265
40%	0.176	0.233	0.376	0.512	0.167	0.339	0.411	0.535	0.279	0.657	1.818	2.465	0.122	0.272	0.524	10.326
50%	0.170	0.257	0.350	0.520	0.172	0.299	0.498	0.530	0.237	0.484	1.270	1.962	3.309	16.290	59.159	61.059
60%	0.126	0.279	0.569	0.611	0.149	0.300	0.474	0.557	0.328	0.659	1.165	2.379	32.857	30.065	76.766	109.659
70%	0.148	0.263	0.408	0.512	0.163	0.301	0.400	0.461	0.268	0.726	1.367	2.402	57.117	63.930	186.921	272.220
80%	0.164	0.245	0.373	0.600	0.174	0.274	0.462	0.588	0.230	1.419	2.234	3.892	66.976	152.718	240.957	342.373
90%	0.134	0.306	0.428	0.546	0.159	0.239	0.455	0.563	0.291	2.487	3.510	6.874	158.028	251.161	319.480	402.104
Var 95%																
Outliers - 0%	0.009	0.034	0.132	0.185	0.013	0.041	0.088	0.195	0.018	0.090	0.185	0.265	0.005	0.013	0.039	0.071
10%	0.015	0.037	0.085	0.220	0.013	0.044	0.096	0.158	0.025	0.096	0.335	0.784	0.005	0.014	0.025	0.051
20%	0.013	0.027	0.081	0.143	0.012	0.066	0.090	0.270	0.042	0.089	0.591	1.784	0.005	0.019	0.031	0.049
30%	0.011	0.027	0.122	0.138	0.013	0.064	0.131	0.152	0.040	0.192	2.327	4.459	0.005	0.018	0.025	0.040
40%	0.017	0.026	0.085	0.148	0.014	0.048	0.094	0.232	0.040	0.387	5.893	13.577	0.020	0.328	1.360	1,928.537
50%	0.017	0.033	0.134	0.124	0.021	0.048	0.152	0.151	0.027	0.294	3.716	6.410	37.114	1,378.762	18,226.125	16,137.572
60%	0.009	0.031	0.284	0.232	0.017	0.042	0.152	0.171	0.072	0.758	1.420	12.790	1,957.258	2,060.635	17,376.436	32,153.897
70%	0.010	0.035	0.074	0.169	0.013	0.036	0.084	0.207	0.033	1.132	4.361	8.303	5,311.165	5,414.392	42,805.190	69,277.956
80%	0.018	0.034	0.083	0.240	0.016	0.039	0.148	0.335	0.037	8.702	19.077	28.837	4,497.510	19,004.548	37,808.679	103,537.955
90%	0.012	0.043	0.152	0.250	0.015	0.045	0.125	0.276	0.062	28.631	34.951	72.639	12,618.640	36,684.018	48,115.801	59,942.692

Figure A.2: Simulation results for different methods using three frames.

5 frames	dual_radar_custom				single_radar_custom				single_radar_half				dual_radar_ransac			
Mean 95%	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)	30m (4.7)	50m (3)	70m (2.5)	90m (2*)	30m (7.9)	50m (4.9)	70m (3.5)	90m (3)
Outliers - 0%	0.101	0.213	0.243	0.308	0.130	0.230	0.297	0.390	0.146	0.256	0.435	0.497	0.080	0.114	0.187	0.261
10%	0.112	0.201	0.250	0.334	0.149	0.164	0.269	0.508	0.152	0.240	0.533	0.494	0.081	0.119	0.229	0.230
20%	0.100	0.210	0.345	0.374	0.145	0.199	0.283	0.363	0.148	0.310	0.476	0.523	0.067	0.109	0.196	0.195
30%	0.131	0.152	0.279	0.277	0.105	0.236	0.274	0.382	0.161	0.408	0.581	0.596	0.091	0.126	0.175	0.216
40%	0.101	0.182	0.327	0.337	0.125	0.194	0.282	0.329	0.175	0.373	0.475	1.257	0.125	0.183	0.986	0.743
50%	0.080	0.171	0.331	0.415	0.145	0.210	0.336	0.472	0.183	0.308	0.814	1.519	0.852	3.098	6.317	1.489
60%	0.088	0.178	0.286	0.353	0.139	0.198	0.267	0.379	0.151	0.408	0.596	1.029	8.065	10.528	48.534	44.847
70%	0.085	0.152	0.320	0.385	0.131	0.232	0.314	0.383	0.189	0.300	1.399	1.763	29.610	46.414	133.153	142.498
80%	0.135	0.238	0.342	0.433	0.089	0.186	0.249	0.393	0.156	0.325	1.265	4.239	66.307	107.490	156.024	193.415
90%	0.112	0.213	0.358	0.433	0.120	0.253	0.319	0.417	0.200	0.339	2.167	2.741	105.188	159.919	239.583	246.990
Var 95%																
Outliers - 0%	0.005	0.019	0.029	0.070	0.008	0.021	0.054	0.109	0.008	0.031	0.112	0.131	0.003	0.006	0.014	0.031
10%	0.011	0.017	0.026	0.054	0.008	0.013	0.055	0.115	0.010	0.038	0.190	0.151	0.003	0.007	0.035	0.022
20%	0.006	0.022	0.044	0.073	0.012	0.024	0.033	0.056	0.012	0.057	0.129	0.196	0.002	0.007	0.017	0.023
30%	0.009	0.014	0.039	0.050	0.004	0.035	0.033	0.062	0.019	0.101	0.243	0.241	0.005	0.009	0.017	0.022
40%	0.006	0.024	0.063	0.046	0.007	0.021	0.034	0.077	0.019	0.085	0.142	3.323	0.018	0.064	11.257	4.681
50%	0.004	0.020	0.055	0.100	0.015	0.021	0.052	0.101	0.017	0.045	0.661	3.624	2.173	31.939	259.592	10.949
60%	0.006	0.016	0.037	0.054	0.010	0.018	0.034	0.097	0.012	0.117	0.538	1.961	72.390	234.404	3,577.561	6,994.947
70%	0.004	0.009	0.049	0.087	0.007	0.024	0.042	0.067	0.019	0.060	3.953	6.666	729.075	2,084.611	22,246.300	21,512.938
80%	0.009	0.024	0.065	0.080	0.005	0.018	0.040	0.101	0.019	0.060	6.566	58.236	3,688.639	7,785.257	24,025.488	28,248.949
90%	0.007	0.028	0.040	0.159	0.007	0.027	0.068	0.121	0.023	0.096	15.972	27.526	7,362.532	12,631.553	25,938.413	34,319.661

Figure A.3: Simulation results for different methods using five frames.

Bibliography

[1] Xinxin Du et al. “A General Pipeline for 3D Detection of Vehicles”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3194–3200. DOI: 10.1109/ICRA.2018.8461232.

[2] Michael Darms, Paul Rybski, and Chris Urmson. “Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments”. In: *2008 IEEE Intelligent Vehicles Symposium*. 2008, pp. 1197–1202. DOI: 10.1109/IVS.2008.4621259.

[3] Yi Zhou et al. “Towards Deep Radar Perception for Autonomous Driving: Datasets, Methods, and Challenges”. In: *Sensors* 22 (2022). DOI: 10.3390/s22114208.

[4] Markus Bühren and Yang Bin. “Automotive Radar Target List Simulation based on Reflection Center Representation of Objects”. In: (Mar. 2006).

[5] Marcel Hoffmann et al. “Instantaneous ego-motion estimation using a coherent radar network”. In: (2022), pp. 321–324.

[6] Bin Yang et al. “RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects”. In: (2020). DOI: 10.48550/arXiv.2007.14366.

[7] Johannes Schlichenmaier et al. “Instantaneous Actual Motion Estimation with a Single High-Resolution Radar Sensor”. In: *2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. 2018, pp. 1–4. DOI: 10.1109/ICMIM.2018.8443553.

[8] Dominik Kellner et al. “Instantaneous full-motion estimation of arbitrary objects using dual Doppler radar”. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 324–329. DOI: 10.1109/IVS.2014.6856449.

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

[10] Dominik Kellner et al. “Instantaneous lateral velocity estimation of a vehicle using Doppler radar”. In: *Proceedings of the 16th International Conference on Information Fusion*. 2013, pp. 877–884.

[11] Andreas Danzer et al. “2D Car Detection in Radar Data with PointNets”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 61–66. DOI: 10.1109/ITSC.2019.8917000.

[12] Eugen Schubert et al. “Clustering of high resolution automotive radar detections and subsequent feature extraction for classification of road users”. In: *2015 16th International Radar Symposium (IRS)*. 2015, pp. 174–179. DOI: 10.1109/IRS.2015.7226315.

[13] Nicolas Scheiner et al. “Object detection for automotive radar point clouds – a comparison”. In: *AI Perspectives* 3 (Nov. 2021). DOI: 10.1186/s42467-021-00012-z.

[14] Nicolas Scheiner et al. “A Multi-Stage Clustering Framework for Automotive Radar Data”. In: *AI Perspectives* (2021). DOI: 10.48550/arXiv.1907.03511.

[15] Martin Ester et al. “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. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[16] Dominik Kellner et al. “Instantaneous full-motion estimation of arbitrary objects using dual Doppler radar”. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 324–329. DOI: 10.1109/IVS.2014.6856449.

[17] Ole Schumann et al. “RadarScenes: A Real-World Radar Point Cloud Data Set for Automotive Applications”. In: *2021 IEEE 24th International Conference on Information Fusion (FUSION)*. 2021, pp. 1–8. DOI: 10.23919/FUSION49465.2021.9627037.

[18] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[19] Charles R. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Stanford University* (2017).

[20] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: 1512.03012 [cs.GR]. URL: https://arxiv.org/abs/1512.03012.

[21] Charles R. Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Stanford University* (2017).

[22] Yin Zhou and Oncel Tuzel. “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: *Apple Inc* (2017).

[23] Charles R. Qi et al. “Frustum PointNets for 3D Object Detection from RGB-D Data”. In: *Stanford University, Nuro, Inc., UC San Diego* (2018).

[24] Alex H. Lang et al. “PointPillars: Fast Encoders for Object Detection from Point Clouds”. In: *nuTonomy: an APTIV company* (2019).

[25] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV]. URL: https://arxiv.org/abs/1804.02767.

[26] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[27] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*. 2024. arXiv: 2402.13616 [cs.CV]. URL: https://arxiv.org/abs/2402.13616.

[28] Kshitiz Bansal et al. “Pointillism: accurate 3D bounding box estimation with multi-radars”. In: *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 2020, pp. 340–353.

[29] Nicolai Kern, Timo Grebner, and Christian Waldschmidt. “PointNet+LSTM for Target List-Based Gesture Recognition With Incoherent Radar Networks”. In: *IEEE Transactions on Aerospace and Electronic Systems* 58.6 (2022), pp. 5675–5686. DOI: 10.1109/TAES.2022.3179248.

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

[31] Karim Armanious et al. *Bosch Street Dataset: A Multi-Modal Dataset with Imaging Radar for Automated Driving*. June 2024.

[32] Andrew Kramer et al. *ColoRadar: The Direct 3D Millimeter Wave Radar Dataset*. 2021. arXiv: 2103.04510 [cs.RO]. URL: https://arxiv.org/abs/2103.04510.

[33] Institute of Artificial Intelligence and Xi’an Jiaotong University Robotics. *Endeavour Radar Dataset*. 2021. URL: https://www.zhejiangglory.eu.org/2021/06/25/Endeavour_Radar_Dataset.html.

[34] *RP-net GitHub page*. URL: https://github.com/Kshitizbansal/pointillism_rp_net.

[35] *scikit-learn page on DBSCAN*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.