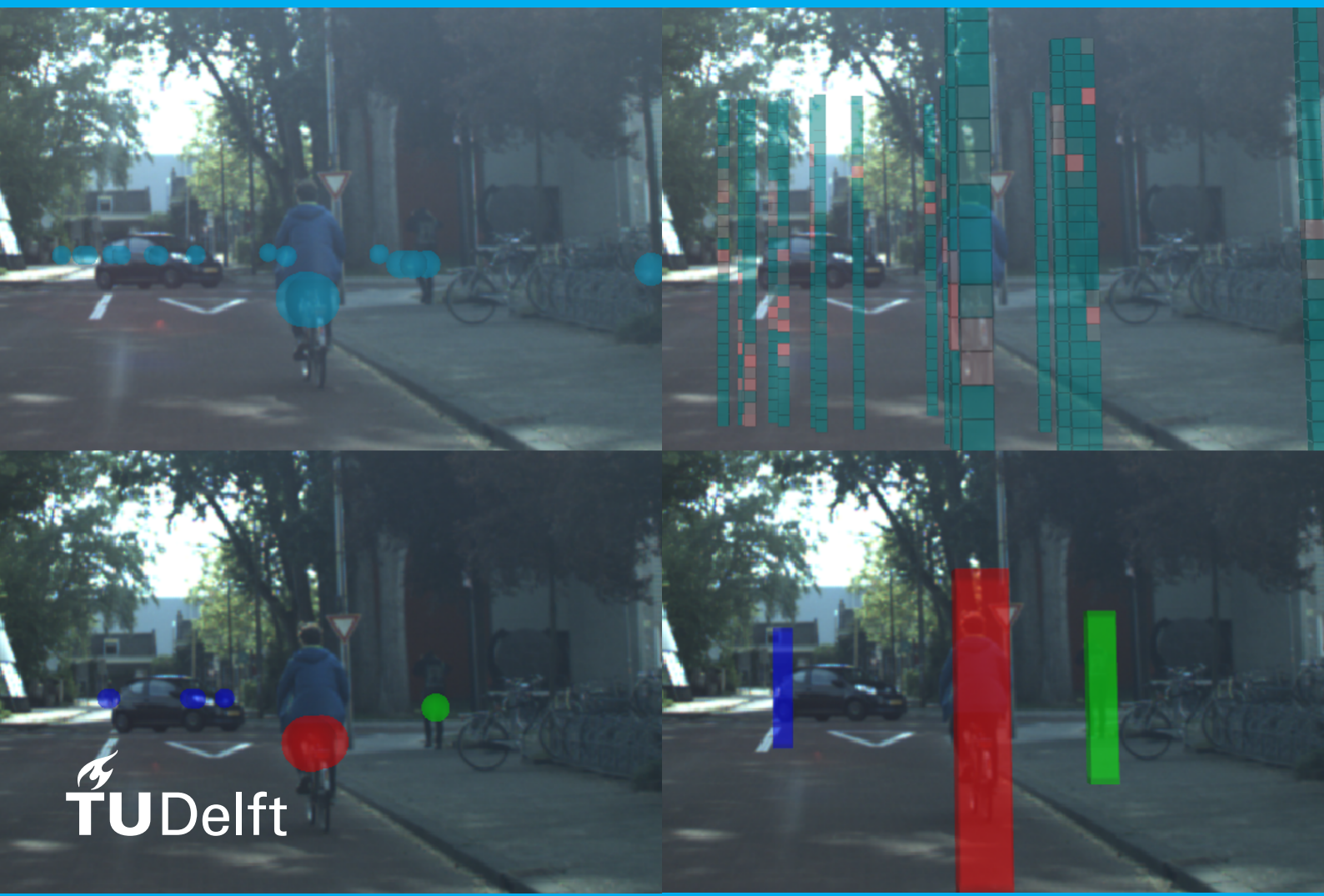
Low-level radar data based road user detection

Jiaao Dong

Academic Supervisor: Prof. dr. D. M. Gavrila
Direct Supervisor: Ir. A. Palffy



TU Delft

Low-level radar data based road user detection

by

Jiaao Dong

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 28, 2019 at 3:00 PM.

Student number:	4756851	
Project duration:	September 4th, 2018 – August 18, 2019	
Thesis committee:	Prof. dr. D. M. Gavrila,	TU Delft, academic supervisor
	Ir. A. Palffy,	TU Delft, direct supervisor
	Dr. J. F. P. Kooij,	TU Delft
	Dr. J. Kober,	TU Delft
	Dr. F. Uysal,	TU Delft

This thesis is confidential and cannot be made public until August 18, 2019.

TU Delft

Contents

List of Figures

List of Tables

Acknowledgements

Studying at Delft University of Technology and working at Cognitive Robotics (CoR) department is the most wonderful and unforgettable experience in my student life.

When I started doing my master thesis, it was such an honor to become a student guided by Prof. Dr. Dariu. M. Gavrila, who is one of the most distinguished scientists in vulnerable road user (VRU) protection domain. Being my supervisor, Prof. Gavrila offers his generous help and always steers my project into the right direction. I have learned not only from the project itself, but also from his attitude towards scientific research, which will always be an intangible and invaluable wealth in my future study and life. Therefore, first and foremost, I would like to thank Prof.Gavrila for his supervision.

I can not wait to say thank you to my daily supervisor Ir. Andras. Palffy. Without his insights, knowledge and ideas, I could not have any chance to overcome the difficulties in my project and get good result. He is the person who helps me to localize the problem, fill in every gap and make things happen. Every time I have problems or questions, he never hesitates to give me feedback and support, no matter where he is and what he is doing. We are supervisor and student, and we are also good friends.

I would also say thank you to all the collegues in CoR. They are warm-hearted, selfless and always willing to help. I truly felt like home when I was working here.

I am grateful to my parents, Biao Dong and Hui Wang, who always support me, despite of the physical distance and time difference. My girlfriend, Iris Zhang, is the person who gives me unconditional love and empowers me to get my MSc degree.

Summary

In order to achieve redundancy and improve the robustness of an autonomous driving system, radar is a suitable choice for road user detection task in severe working conditions (e.g. darkness, bad weather). However, the real-time multi-class radar based road user detection algorithm is less explored compared with camera and LiDAR solutions. To fill this gap, the current thesis proposes a pipeline for radar based road user detection task, which is able to detect pedestrians, cyclists and cars by a single radar sweep. The pipeline effectively utilizes the advantages of radar low-level data by using it as region descriptor and combining it with radar high-level data for region proposal. A novel convolutional neural network structure called LLTnet is designed, in combination with proper pre-processing and post-processing stages. Ensemble learning is used to further improve the inter-class detection accuracy. The LLTnet itself performs radar targets segmentation. If needed, its output can be fused into object-level detection. To better train the network, a real-life dataset containing different moving road users is created during the study by a moving test vehicle, which simulates the real-life urban driving scenarios. After the network is trained, it is firstly evaluated by target-level metrics, such as the classification accuracy and F1 score. Then object-level metrics are used for object-level evaluation, such as the precision, recall and intersection over union (IoU).

Comprehensive experiments are performed which not only evaluate the performance of the proposed model but also test the importance of different stages and features, such as the importance of the ensemble learning and the validity of adding low-level data. The proposed pipeline improves the target-level F1 score from 0.59 of the baseline to 0.64 using LLTnet without ensemble learning. By adding the ensemble learning stage, the target-level F1 score is further improved from 0.64 to 0.70. The object-level recall of pedestrian class greatly improves from 0.37 to 0.68. The validity of adding low-level data to the algorithm is verified by bypassing the low-level data branch of the network. With only high-level branch, the target-level F1 score drops from 0.70 to 0.60. Furthermore, the trained model also shows good generalization ability on unseen data.

1 Introduction

1.1. Context

Over the past few years, self-driving has become a popular research topic in many domains, from computer science, system control to law and philosophy, from universities to companies. Although it is not possible to calculate how much time and energy has been devoted to this area, there is one thing for sure: more and more vehicles equipped with driver assistance system are running on the road and this technology has been changing human's life. In the future, the ultimate goal of self-driving is to get human out of the loop in order to eliminate accidents due to human error. However, this should be achieved without introducing other safety issues. To ensure the car, no matter how autonomous it is, is driving safely, the perception ability is one of the most important parts, because it is the input of the system. Unlike driving on the highway, where most road users are different types of cars, driving in urban area is much more difficult due to the existence of different road users and larger variance of the environment. For example, the most commonly seen road users are pedestrians and cyclists. They are more dynamic and moving with a less predictable behavior. Previously, camera and LiDAR are the most commonly used sensors in this domain.

Camera uses 2D information including geometric and texture details. With the help of stereo camera or RGBD camera, the depth of each pixel can also be measured. Since the camera captures visible light just as how human eyes work, it is the most intensively explored topic and has reached significant accuracy in recent years. Extensive surveys on vision based pedestrian detection can be found in [7] and [10]. However, vision based road user detection has two main drawbacks. First of all, camera struggles in darkness and bad weather conditions, e.g. fog, snow or rain. Secondly, the depth's measurement is less accurate beyond a certain distance. For example, the error of a stereo camera grows quadratically with respect to the distance.

Unlike camera, LiDAR is able to generate a dense point-cloud which directly samples the 3D space. Different methods are proposed for LiDAR object detection task, such as the study in [75] and [64]. With a large amount of points to be processed per frame, LiDAR has the scalability issue while running it in real-time self-driving platform. Furthermore, LiDAR uses narrow light pulses for the measurement, which suffers from dusty and snowy weather conditions. Last but not least, LiDAR is still too expensive to be deployed on production vehicles at the moment. In many use cases its price is higher than the vehicle itself.

Compared with camera and LiDAR, radar uses radio wave to measure the range, azimuth angle and speed of objects. It has some advantages over the other two sensors. Firstly, it is more robust to weather and lighting conditions (e.g. rain, snow, darkness). Secondly, radar can directly measure the speed distribution in space by Doppler effect. This speed distribution provides information both for classifying a road user and predicting its path. Last but not least, radar is cost-effective, which allows several radars to be mounted around the car body. Although the performance of radar will deteriorate inside tunnels and the resolution is not as high as the other two sensors, it is robust and reliable in most of the

cases.

In fact, being used by vehicle active safety system, radar has longer history than the other two sensors. In 1997, the first commercialized adaptive cruise control (ACC) system with a 77 GHz automotive radar was introduced by Mercedes DISTRONIC system in its premium model, S-Class [59].

1.2. Problem statement

Despite of all the advantages and its history, radar is less explored in the domain of road user detection. There are four main reasons. First of all, the elevation angle of an automotive radar is small, which makes it only capable to measure objects in a 2D horizontal plane. Secondly, radar targets are much more sparse than camera pixels or LiDAR point cloud. When a road user is small, such as a pedestrian, only a small number of targets or only one target can be detected. Features extracted from sparse targets are less reliable due to the influence of outliers. An additional reason lies in the fact that the radar spectrum is less interpretable than the other two sensors, which increases the difficulty when tuning the machine learning algorithms based on its output. Last but not least, although the measurement for range and speed are accurate, the angular resolution of radar is low.

The data provided by a commercially available radar is a target list, see Figure 1.1a from a top-down view, which is also called the high-level data. Each target has a 2D ground plane position, a reflectivity and a radial speed relative to the ego-vehicle. The road user detection algorithms using high-level data usually follow a clustering based pipeline [46, 57, 61, 71]. The bottleneck of this pipeline is the clustering stage and the sparsity of the radar targets. When road users are close to each other, e.g. pedestrians in a group, they can be clustered into a single road user. In addition, the statistics from a small amount of targets are sensitive to outliers. Problems caused by the clustering stage can be addressed by end-to-end deep learning methods, while it requires a denser point-cloud as input. Multiple radar frames are aggregated to fulfill this requirement, which introduces a time delay for the prediction. In some studies this time delay reaches up to 0.5 s, which is not acceptable in driving scenario.

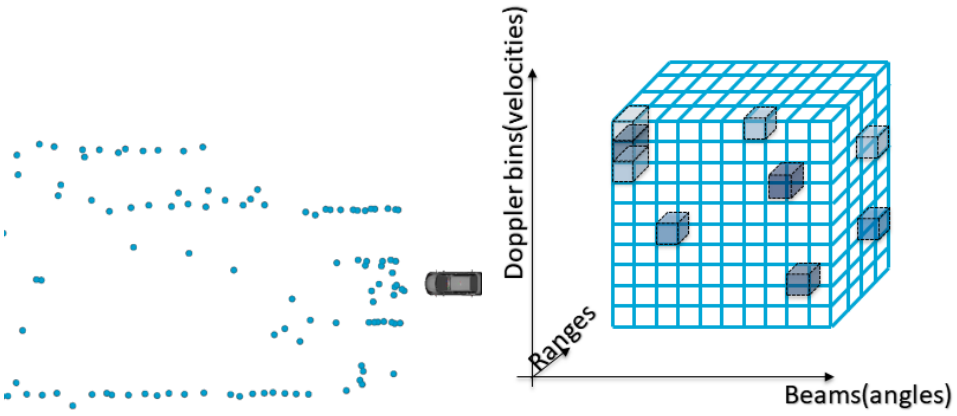


Figure 1.1: (a) This figure shows the target list of a single radar frame from the top-down view. The vehicle is heading towards the left. The blue points are radar targets. (b) The low level data is a 3D matrix with range, speed and angle bins as axes. The entries are the magnitudes of radio wave responses.

Although the resolution of radar high-level data is gradually increasing with the development of semiconductor industry and signal processing techniques, it is still important to avoid information loss when using radar for road user detection. The information loss is mainly caused by the thresholding that generates radar target list from low-level data. The low-level data is a 3D matrix, which is also called the *radar cube*. The axes of the matrix are range, azimuth angle and Doppler bins, see Figure 1.1b. The entries are the magnitudes of radio wave responses at that range-angle-Doppler bin. Due to less thresholding, the low-level data provides the detailed speed distribution for each 2D location in ground

plane. Based on the micro-Doppler signature caused by different road users, many studies utilize this speed distribution for road user classification [4, 44, 45]. However, the positioning accuracy of low-level data is not as good as high-level data, since no sophisticated super resolution algorithms are used. Furthermore, if we look at the ratio between non-zero values and zeros-values inside the radar cube, the non-zero entries are still sparse.

In the current study a new pipeline is proposed that combines the good positioning accuracy of high-level data and detailed speed information of low-level data, see Figure 1.2. This pipeline alleviate the problems of the clustering stage by not using it as an object proposal. The region of interest is provided by the non-static high-level targets. The low-level data cropped by the region proposal is used as the region descriptor. With the help of a suitable convolutional neural network and ensemble learning policy, the information from one single radar frame is enough. The hypothesis is that with a single radar frame, the proposed pipeline will have better performance in terms of the localization accuracy and classification accuracy of a road user detection task. A dataset will be created by an automated annotation pipeline for training and evaluation.

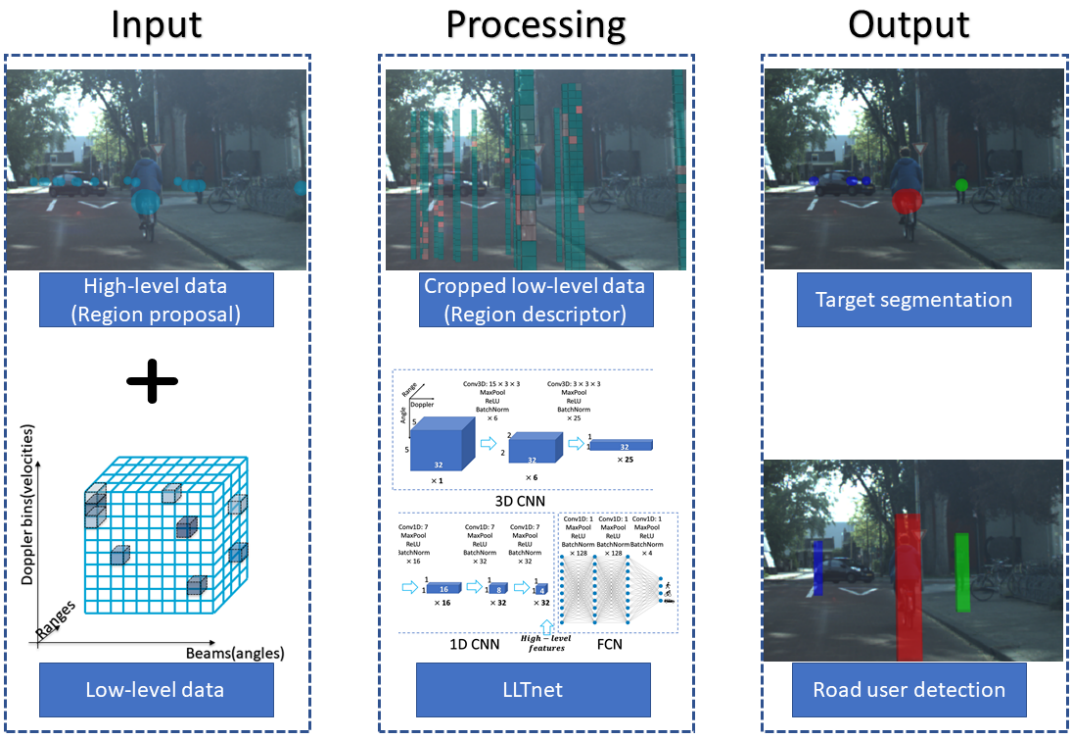


Figure 1.2: The general pipeline of the proposed method. At the first step, the high-level data (i.e. target list) is used for region proposal, where low-level data is cropped. Then the cropped low-level data is used as region descriptor. Each region descriptor is classified by a neural network, thus a label is attached to each target. If needed, the targets can be further fused into road users.

1.3. Thesis outline

In the following chapter, the related work on radar based road user detection is reviewed as well as some general theories. Based on the related work, a new pipeline is proposed in chapter 3. In chapter 4, comprehensive experiments are performed on a real-life dataset in urban scenarios and the results of the proposed method are compared with three baselines including state-of-the-art clustering based methods. This is followed by the conclusion part of the thesis, which also provides some recommendations for the future work.

2

Related Work & Thesis Contributions

Vision based road user detection has been widely researched over the last decade and reached significant accuracy under circumstances where camera can work properly , e.g. with good lighting and weather conditions. Many hand-crafted feature based methods are used for road user detection, such as the methods that use histogram of oriented gradients (HOG) [15] and deformable part model (DPM) [20]. After the popularity of deep learning and the introduction of large scale datasets [17, 35], the general purpose object detection algorithms are empowered, such as Faster R-CNN [23], SSD [36] and YOLO [50]. If trained by a suitable dataset containing enough road users, e.g. EuroCity Persons [9], these algorithms are accurate and efficient for vision based road user detection task.

LiDAR is usually used in high-level driving automation functions due to its ability to generate a dense point-cloud of the 3D world without being influenced by lighting conditions. The LiDAR road user detection algorithms can be found in [18, 75]. Despite of its reliability and accuracy, LiDAR suffers from dusty and snowy weather conditions. Currently, LiDAR is not accessible for production vehicles due to its high price.

Radar is less explored for road user detection task compared with camera and LiDAR. However, due to its robustness under severe working conditions and cost-effectiveness, it is a good sensor for many situations where other sensors are not functioning properly. In this chapter, some general knowledge about radar and deep learning will be introduced, followed by the features used by the radar based road user classification task. The existing radar based road user detection and classification algorithms are following two main streams depending on whether the input is high-level data or low-level data. Therefore, in section 2.2 and section 2.3, the methods regarding radar road user detection and classification are categorized into high-level data algorithms and low-level data algorithms. In the end, the main contributions of the proposed method are introduced in order to fill the gaps found in the related work.

2.1. Theories

2.1.1. FMCW radar

Radar is an acronym for 'Radio Detection And Ranging'. It uses radio wave to measure the range, azimuth angle or velocity of objects. This product was first developed for military use and then became a widely used sensor in various applications such as aviation, remote sensing, speed control and automotive industry. In general, radar uses radio wave and its reflection to detect objects. The wave forms and different digital signal processing schemes depend on the applications.

Among different types of radars, what is most widely used in automotive industry is called FMCW radar, which stands for 'Frequency-Modulated Continuous Wave' radar. As the name indicates, unlike pulse wave radar, the radio wave of an FMCW radar is continuous. The frequency of the continuous wave is modulated over time. These two features enable FMCW radar to measure the range and speed at the same time. If multiple antennas are used, the

direction of the range-speed information can also be recovered. Therefore, the output of an FMCW radar is range-speed-angle information. The workflow of FMCW radar is shown in Figure 2.1.

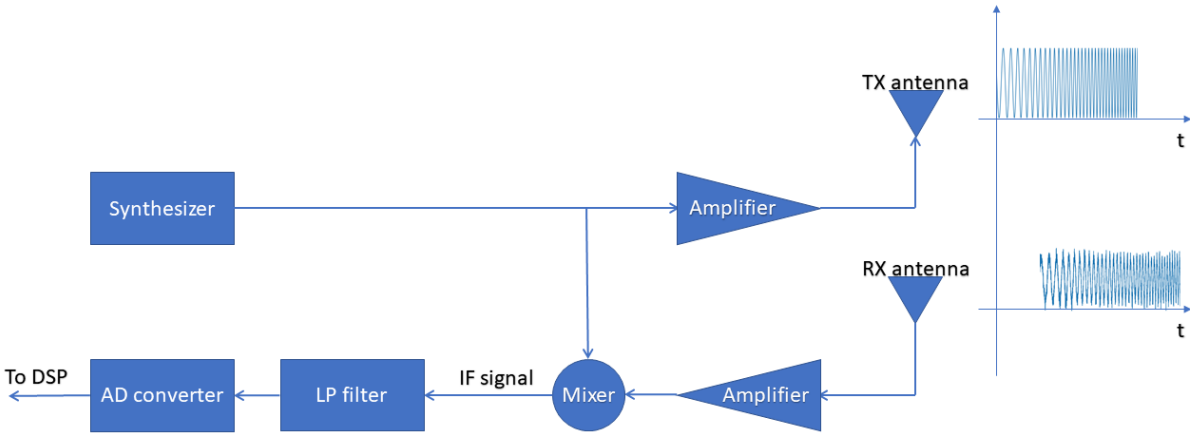


Figure 2.1: The workflow of FMCW radar. The synthesizer generates modulated radio waves transmitted by the TX antenna, which is called TX signal. After the wave is reflected by the objects and received by the RX antenna, the TX signal and RX signal are mixed by the mixer to get the IF (Intermediate Frequency) signal. This signal is sampled by the A-D converter and given to the digital signal processing pipeline.

The synthesizer generates continuous sinusoidal wave with modulated frequency. This signal is transmitted by TX antenna, hence it is called TX signal. The modulation of frequency is usually described by the frequency-time plot ($F-t$ plot). An $F-t$ plot and its correspondent amplitude-time plot ($A-t$ plot) of a typical saw-tooth modulation are shown in Figure 2.2. The radio wave with its frequency increasing linearly from a minimum value to a maximum value is called a chirp. In Figure 2.2, a chirp is characterized by a start frequency F_c, a bandwidth B and a chirp time T_c. In a commercial 77 GHz automotive radar, hundreds of chirps will be transmitted within a second [65].

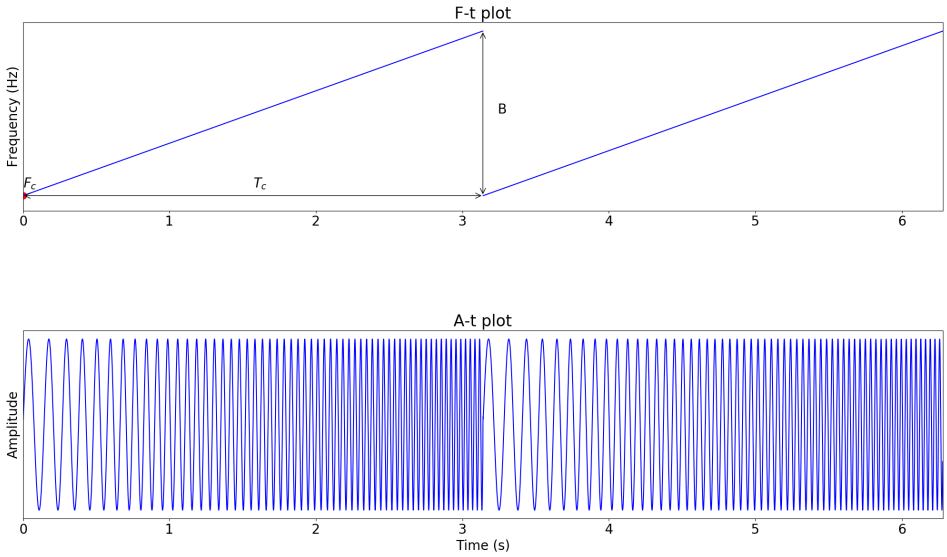


Figure 2.2: $F-t$ plot and $A-t$ plot of radar TX signal. The frequency increases linearly with time. The time period within which a frequency increases from the minimum value to the maximum value is called a chirp. A chirp is defined by a start frequency F_c, a bandwidth B and a chirp time T_c. Modulated frequency is able to measure the traveling time indirectly by frequency difference rather than measuring tiny time difference itself. The latter is difficult.

After the chirps are transmitted by the TX antennas, they are reflected by objects and received by RX antennas. The received signal is called the RX signal. The RX signal and TX

signal are then mixed by a mixer. The resultant signal is called IF (Intermediate Frequency) signal. The frequency and phase of the IF signal are the difference of frequency and phase between TX and RX signal. The IF signal is shown in 2.3.

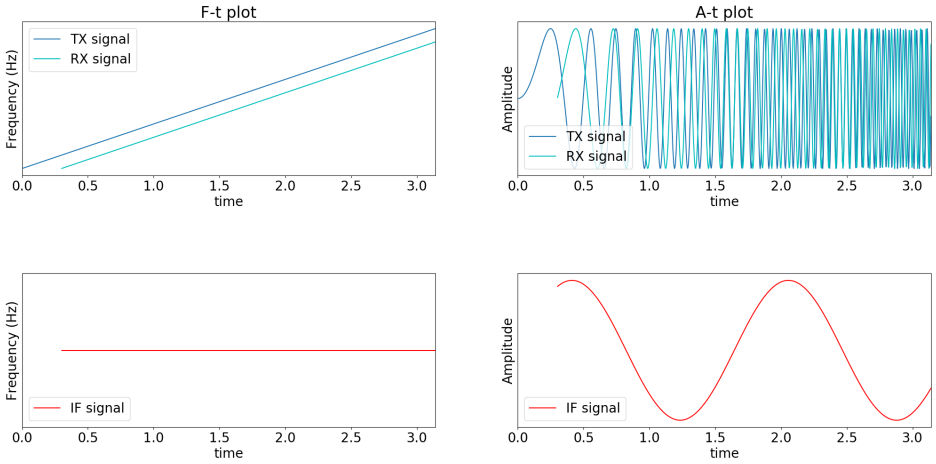


Figure 2.3: The IF signal calculated from one TX chirp and one RX chirp. The dark blue signal is the transmitted signal (TX signal) and the cyan signal is the reflected signal (RX signal). The frequency difference between these two signals are plotted in red (the plot on the bottom left), which is also the frequency of the IF signal. If the object is static or can be considered as static, the frequency difference does not change. Therefore the IF signal is a sinusoidal signal with constant frequency (the signal on the bottom right). In practice, this frequency is recovered by performing Fourier transform to the signal mixed by the mixer.

Then the continuous IF signal is sampled by analog-to-digital converter (A-D converter) and served as the input for the digital signal processing (DSP) pipeline. To better understand different levels of radar data, it is necessary to cover some basics about radar DSP. For more details, please refer to [38, 43, 65, 69].

In the radar DSP pipeline, Fourier transform plays the main role, which converts signals from time domain or space domain into frequency domain. Since the signal in a digital circuit is not continuous, the discrete Fourier transform (DFT) is performed in practice. However, the original DFT is not efficient due to its $O(n^2)$ computational complexity, where n is the number of input samples. This problem is addressed by the well-known fast Fourier transform (FFT) proposed in 1965 [12], which greatly reduces the complexity to $O(nlog(n))$.

In the following subsections the signal processing steps to recover the range, velocity and azimuth angle of objects will be explained.

Range measurement

FMCW radar measures the range of the object by the frequency difference between TX signal and RX signal, i.e. the frequency of IF signal. A typical process is shown in Figure 2.3. For a single object, the RX signal is a delayed version of TX signal. Therefore the frequency of the resultant IF signal does not change within one chirp time, see the red signal in Figure 2.3. The IF frequency is proportional to the range between the objects and radar. The relation is given by Equation 2.1 (d: distance, S: slope of the chirp, c: speed of the light).

$$f_{IF} = \Delta f = S \times \frac{2 \times d}{c} \tag{2.1}$$

When there are multiple objects, if their range are different, multiple reflected chirps will be received with different time delays, as shown by the red signal and green signal in Figure 2.4. The IF signal from these two signals will have multiple frequency components. These frequency components can be recovered by FFT processing. In this way, the range of multiple objects can be resolved. This process is called range FFT in this thesis.

Now that it is clear how is the range measured by chirps, it is necessary to explain the factors that determine the range resolution and the maximum measurable range. In principle, as long as there is a range difference between two objects, there will be a time delay between transmitted and received chirps. However, in real cases, this delay has a lower limit

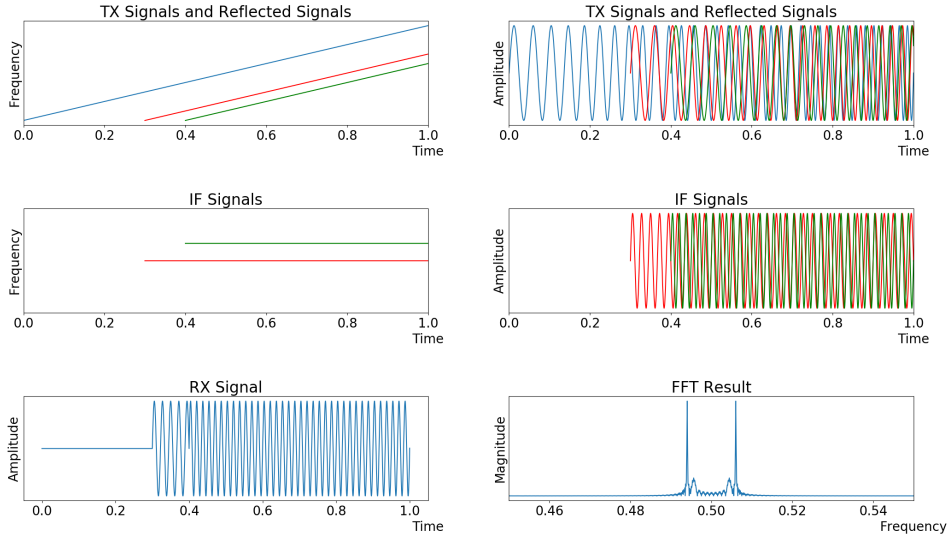


Figure 2.4: The TX, RX and IF signal caused by multiple objects. The 1st row: The blue signal is the transmitted signal. The red signal and green signal are from two different objects. The 2nd row: the green signal and red signal are IF signals caused by two objects. The 3rd row: the IF signal will be recovered from the RX signal by Fourier transform.

which is $\frac{1}{T_c}$ [38]. T_c is the chirp time in Figure 2.5. Any frequency difference smaller than the limit cannot be resolved by range FFT. Using the frequency difference in Equation 2.1, the resolution can be calculated by Equation 2.2. Any objects closer than this resolution cannot be distinguished in range. For example, with a bandwidth of 200 MHz, the range resolution is 0.75 m.

$$\delta d > \frac{c}{2B} \tag{2.2}$$

The maximum range that can be recovered mainly depends on the sampling rate of the following A-D converter. If the sampling rate of A-D converter is F_{IF}, the maximum range is limited by the sampling rate according to Equation 2.3.

$$d_{max} = \frac{F_{IF} \times c}{2d} \tag{2.3}$$

Speed measurement

One of the advantages of FMCW radar is the ability to measure the speed of objects. When an object is moving, the frequency of the radio wave reflected from that object will be further shifted due to the movement. The value of the shift is proportional to the value of the relative speed. This phenomenon is called Doppler effect. Supposing the object is not moving faster than the maximum measurable speed of a radar and its range is smaller than the maximum range limit, the same RX signal caused by the object will appear at two consecutive chirp times. Due to the frequency shift caused by Doppler effect, the IF frequency of two objects are different. However, as shown in Figure 2.5, the frequency difference between the IF signals in two consecutive chirps are too small to be observed. Unlike the frequency, the phase of the IF signal is much more sensitive to such minor changes between consecutive chirps. According to the calculation in [67], if the range changes 1 mm, for a 77 GHz radar, the phase of the IF signal changes by 180° while the frequency only changes by 333 Hz. If two objects are at the same range but different velocities, their responses after range FFT will be gated to the same range bin. To resolve different objects only by their speed, Doppler FFT can be applied over a series of range FFT results from multiple consecutive chirps to get their phase shift over time. After the Doppler FFT, the N-dimensional vector from range FFT becomes an $M \times N$ array with N range bins and M Doppler bins. The image from the 2D-FFT is called range-Doppler image. After getting the range-Doppler image, radar DSP work in short time window is finished. The following steps are performed at a lower data rate [69].

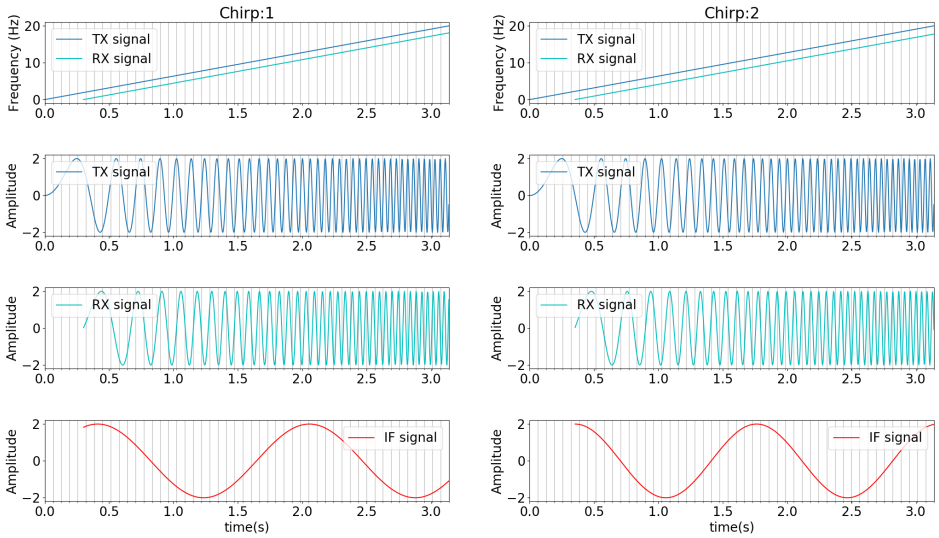


Figure 2.5: The TX signal, RX signal, IF signal in two consecutive chirps

Most automotive radar has multiple antennas. Each antenna has its own range-Doppler image.

Angle estimation

Most objects can be separated after the range FFT and Doppler FFT applied to the IF signal from multiple chirps. However, only knowing their range without accurate position is not adequate for applications in driving scenario. Therefore, resolving them into different azimuth angles is important. One of the most commonly used solutions is digital beam forming, which performs a virtual rotation of the antenna by focusing on the signal from one direction at a time like a physically rotating antenna [69]. Referring to the geometry in Figure 2.6a, signals travel for different distances before they are received by different antennas. The different distances cause a time difference, which introduces a phase shift between IF signals of different antennas. If the distance d between RX antennas is known, the relation between azimuth angle θ of the object, the wavelength λ of the radio wave and the phase shift ω of the IF signal are given by Equation 2.4 and Equation 2.5.

$$\omega = \frac{2\pi d \sin(\theta)}{\lambda} \tag{2.4}$$

$$\theta = \arcsin\frac{\omega\lambda}{2\pi d} = \arcsin\frac{\omega}{2\pi R} \tag{2.5}$$

In practice, FFT is used to calculate the phase shift caused by different angles of arrival. As stated in the previous subsection, each antenna calculates its own range-Doppler image. The complex number pixels at the same position of the range-Doppler image represents signals received by different receivers. To estimate the direction of arrival (DoA) by their different phases, FFT is again used for frequency components analysis. Different antennas can be seen as the sampling of the radio wave in space. As Figure 2.6b shows, before applying FFT, the input values are padded by zeros to avoid 'binning' effects [69]. After padding, FFT is performed separately to the sequences in a single range-Doppler bin.

As is shown in Figure 2.6a and Equation 2.5, the angular resolution is influenced by the spacing d between different antennas, or the ratio between the spacing and the wavelength $R = \frac{\lambda}{d}$. Taking derivatives on both sides of Equation 2.5, the angular resolution $\delta\theta$ is given by Equation 2.6.

$$\delta\theta = \frac{\Delta\omega \times \lambda}{2\pi d \cos\theta} \tag{2.6}$$

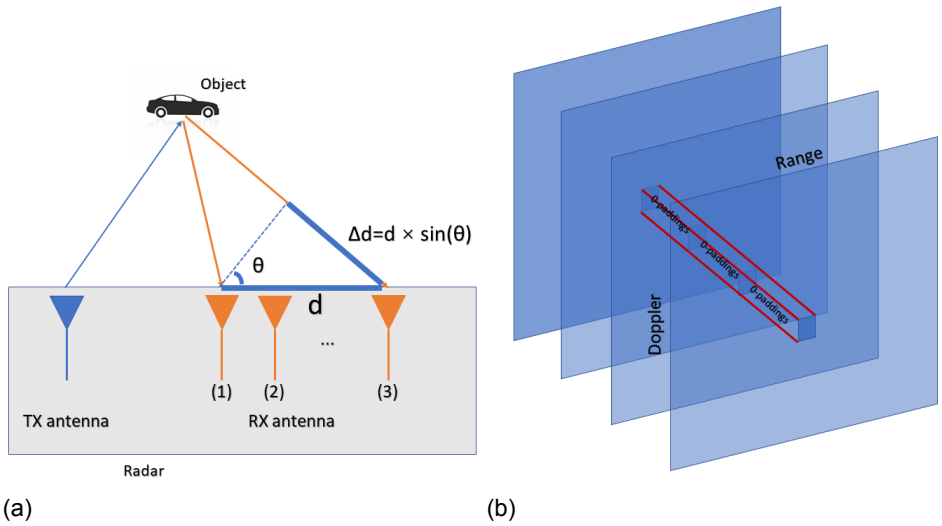


Figure 2.6: (a) Radar uses multiple RX antennas to resolve objects in different angles. (b) The angle FFT applied to multiple range-Doppler images.

Larger spacing will result into a higher angular resolution. However, larger spacing also causes larger phase shift ω with a certain azimuth angle θ. A generic function of radio wave can be written as Equation 2.7.

$$F(x) = A\sin(2\pi ft + \omega) = A\sin(2\pi ft + \omega' + \pi) = A\sin(2\pi ft - (\pi - \omega')), \tag{2.7}$$

If the phase shift of the sinusoidal function exceeds $[-\pi, \pi]$, since it is a periodic function, the phase shift becomes ambiguous. It is not known whether the phase shift is ω (in the $[0, \pi]$ phase) or ω' (in the $[\pi, 2\pi]$ phase). To avoid the ambiguity, in some automotive radars, the phase shift is constrained by Equation 2.8, which decides the maximum possible value of the spacing d_{max} defined by Equation 2.9.

$$\omega = \frac{2\pi d\sin(\theta)}{\lambda} \tag{2.8}$$

$$d_{max} \leqslant \frac{\lambda}{2\sin(\theta)} \tag{2.9}$$

If the radar is required to cover the space from $-90°$ to $+90°$, d_{max} should be less than or equal to $\frac{\lambda}{2}$. Despite of the ambiguity, sometimes the radar uses a larger spacing to increase the angular resolution within the central FOV. If the spacing-wavelength ratio $R = 1.5$, given by the constraint from Equation 2.8, the critical θ value that avoids the ambiguity should be $\theta = \arcsin(1/2R) = 19.47°$. If the object is present at an angle larger than 19.47°, due to Equation 2.7, the reflection of the next angle beam larger than 19.47° will correspond to the FFT result from the most negative beam in $[-\pi, \pi]$ phase shift.

Conventional beamforming is limited in angular resolution. In order to improve the resolution, adaptive beamforming can be used for DoA. One of the most well-known algorithms is MUltiple SIgnal Classification (MUSIC) [58]. MUSIC uses eigenvalue decomposition to calculate the signal subspace and noise subspace. Using the fact that these two spaces are orthogonal to each other, it is possible to search the signal subspace and pick the peaks from the power that represents to what extent are they orthogonal to each other. MUSIC is usually applied to FMCW radar with multiple antennas to increase the resolution. Details can be found in [58].

Target list generation

The range, speed and azimuth angle responses after DSP pipeline are gated into a 3D matrix, which is also called the low-level data or radar cube. The indices of the matrix are the range,

azimuth angle and speed bins. The entries are radio wave responses (usually magnitude). The low-level data is shown by Figure 1.1b in section 1.2.

From this 3D matrix, a target list can be created by thresholding. If an entry is higher than the threshold, it is selected as a target. Its range, azimuth angle and speed are from its range-azimuth-speed bin's index. Its reflectivity is calculated by its magnitude. The list of targets is called high-level data, see Figure 1.1a in section 1.2. The threshold used for generating targets is an important parameter. If the threshold is too low, not only the reflections from real objects but also entries from the background noise will be picked up, which are called false alarms. The systems that have high false alarm rate will keep interrupting normal driving behavior even if there is no obstacle or road users. On the contrary, if the threshold is too high, useful reflections from real road users will be removed. In real life situation, an adaptive threshold is calculated in order to fullfill a constant false alarm rate (CFAR).

Using CFAR as a criterion, different methods are used to calculate the adaptive threshold. One way to calculate the threshold is called cell-averaging. A sliding window is applied to each cell, which is called cell under test (CUT). The power level of cells that are not immediately adjacent to the CUT are averaged. If the power of the CUT is higher than its adjacent cells and higher than this averaged value, a target is declared at this cell. This method is called cell-averaging CFAR (CA-CFAR) [52].

In practice, CFAR is not performed at the last stage where the entire radar cube is formed, but usually at the stage where range-Dopper image is created. CFAR is only calculated in this 2D image, and only the values filtered by CFAR are used as the input for the angle FFT. After the angle FFT, only the highest peak is selected. After CFAR and angle FFT, some sophisticated processing algorithms can be applied in order to get more accurate targets' location.

2.1.2. Deep learning

Machine learning

The task that is performed by a typical machine learning algorithm is to approximate ('learn') a function $\vec{y} = f(\vec{x})$, which maps the input $\vec{x}$ to the output $\vec{y}$. The usual way to do this mapping is learning from examples. These examples are called training data. The performance of the mapping is measured by a value called loss, which is calculated from training data. By minimizing the loss iteratively, the mapping accuracy is gradually improved. Sometimes the input are given to the model without processing, however, most of the cases the input are processed in order to get some features, e.g. the histogram of oriented gradients (HOG) [15] for vision based pedestrian detection. The output depends on the task. If the output is the label of the input, this machine learning task is called classification. The loss will be a value that measures the mis-classification error. If the output is continuous rather than discrete labels, the task is called regression. The loss in this case is a value that measures how far it is from the true output to the predicted output. In general, machine learning algorithms focus on how to better formulate and optimize the mapping according to the task. Most mappings are parameterized by a pre-defined function. Few of them are non-parametric. For classification, there are different classifiers, such as support-vector machine (SVM) [13], random forest [29], k-nearest neighbor (kNN) [2]. For regression task, there are linear regression, logistic regression [14], etc. These methods are usually designed by prior knowledge of the task and making hypothesis to the data distribution. The features given to these methods need to be engineered and selected.

Deep neural networks

Over the past few years, deep learning (DL) has achieved great success in many areas. From simple optical character recognition (OCR) to the most cutting-edge natural language processing (NLP) chatbot, deep learning has pushed artificial intelligence (AI) to a new era. Deep learning is a branch from machine learning topics. It formulates the mapping by using a group of layers to build up a network and learns low-level input features to build up high-level abstraction through its hierarchical structure. The architecture of the network is flexible, which makes it easy to adapt the capacity of the model according to the amount and diversity of the training data. The features are learned by the network automatically, thus the

feature extraction and feature selection steps can be skipped. The principle of the network structure is inspired by biological system in animal brains. The number of layers can vary from 3 (one input layer, one hidden layer and one output layer) to over 100 (ResNet-101 [23]). That's why it is called deep neural networks (DNN).

Similarly to other machine learning algorithms, a loss is defined according to the task. The parameters (also called weights) in different layers are updated by back-propagation [55], which calculates the gradient of the parameters with respect to the loss and performs gradient descent [53] recursively. The only difference from an ordinary gradient descent algorithm is that the weights in the intermediate layers can be reused when they are back-propagated to the previous layer [21] . Depending on the direction of the connection between different layers, there are mainly two types of deep neural networks. If the network does not have feedback connection between layers, it is called feed-forward deep neural networks. If the output of some layers are re-connected to the input of the previous layers, it is called recurrent neural networks (RNN). RNN are out of the scope of the presented thesis.

The building blocks of a deep neural network are different layers. Some of them are generic enough to be seen in many different tasks, such as fully-connected layers and convolutional layers. Some of them are dedicated to some specific tasks such as RoIPool in Faster R-CNN [51] and RoIAlign in Mask R-CNN networks [24] for image object detection. In the following subsections fully-connected layers and convolutional layers will be introduced.

Fully-connected layers

In many cases, the data given to the deep neural network can be converted to one-dimensional vectors called feature vector. Fully-connected layer formulates the mapping from the input vector to the output vector by matrix multiplication followed by non-linearity. Given by Equation 2.10, if the input vector $\vec{x}$ has n elements, and the output vector $\vec{y}$ has m elements, the matrix A should have $m \times n$ elements. These elements are the weights of this layer. In this case, every element of the input $\vec{x}$ has influence on every output in vector $\vec{y}$. This is the reason why it is called 'fully-connected layer'. The non-linearity is a nonlinear function such as sigmoid function or a rectified linear unit (ReLU). Adding non-linearity is necessary, otherwise two matrices multiplied in series are equivalent to a single matrix multiplication, which is unable to do non-linear mapping.

$$\vec{y} = non\text{-}linearity(A \times \vec{x}) \tag{2.10}$$

Convolutional layers

One of the most common and useful applications of machine learning algorithms is to perform image classification. If the resolution of the image is high, it is usually not easy or even impossible to convert the input data to one dimensional vectors and apply fully-connected layers to those vectors. In this case, convolutional neural networks (CNN) is usually used to achieve a sparse connection between its input and output. By definition, the two dimensional convolution applied to a two dimensional image is performed by a kernel with $m \times n$ weights. The kernel slides over the image and calculates the output pixel value based on the weighted average of the pixels covered by the kernel. The output pixel is only influenced by the pixels inside the kernel, which makes the convolutional layers achieve a sparse mapping from the input to the output. Therefore the number of parameters are less. Consequently, the overfitting risk is lower and the generalization ability is better. Furthermore, In combination with max-pooling process, CNN can also achieve local translational invariance. In Figure 2.7, the structure of AlexNet [33] is shown, which is a well-known architecture in the history of CNN.

2.1.3. Features of radar based road user classification

Classification is an important aspect in a detection algorithm. When an algorithm is developed for classification, the features should be considered at first. To do road user detection task, the radar 'sees' the world in a different way compared to camera, LiDAR or human eyes. It does not have the actual shape of a road user in 3D space. What it has is the speed and reflectivity distribution in horizontal plane. In high-level data and low-level data, the representation of the features are different, but the principle behind the features are the same.

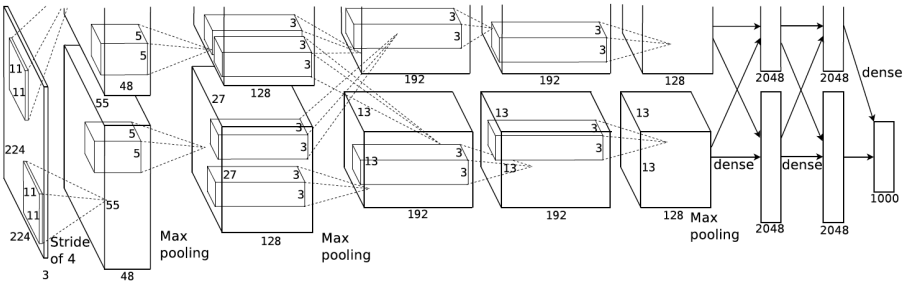


Figure 2.7: The architecture of AlexNet [33], a well-known deep neural network for image classification.

Using them as input, the main features used by radar to discriminate road users are reflectivity distribution, spatial distribution and speed distribution. These features are explained in the following subsections.

Reflectivity

The reflectivity is usually represented by radar cross-section value (RCS) in high-level data and magnitude of the radio wave responses in low-level data. It mainly depends on the material of the object. Metal has a much higher reflectivity than the clothes or vegetation. As a consequence, cars are more reflective than other road users. Cyclists are between cars and pedestrians, due to the combination of human body and bike frame. Pedestrians are the least reflective road users. Sometimes if the pedestrians are in a cluttered background, they will be falsely removed by the intermediate processing stages, e.g. CFAR, which is explained in subsection 2.1.1.

The range of the object also influences the received magnitude of low-level data. According to [4], the received magnitude decreases with $\frac{1}{r^4}$, where r is the range.

Spatial distribution

The spatial distribution of road users in 2D horizontal plane is usually used as a feature for many algorithms [27, 42, 71]. Typically, cars have larger area in space while the area covered by pedestrian or bike is smaller. Using high-level data, this distribution is usually measured by the spread and/or standard deviation of range and azimuth angles. In low-level data, the spatial distribution is measured by the distribution of magnitudes. This distribution is influenced by the range and angle resolution of the radar. A typical range resolution is around 0.5 m with 2° angular resolution at the central FOV. In other words, each 2D grid cell is at the same scale of a road user. In addition, radar uses polar coordinate system to discretize the space. With a constant angle resolution, the width of each grid cell gets larger at a larger range, see Figure 2.8. As a consequence, if the road user is far away from the radar, the non-zero entries in the low-level data will be less.

micro-Doppler signature

For radar road user classification, the speed distribution is usually used, which is also called micro-Doppler signature, see Figure 2.9. These two names are used interchangeably in this thesis. If a moving vehicle is detected by the radar, the speed distribution of the vehicle will have few or only one peak, because it is a rigid body with a single speed. Unlike vehicles and buildings, the human body is deformable. When a pedestrian is walking or a cyclist is riding a bike, different parts are moving at different velocities. The arms and legs of a pedestrian are moving at higher speed. The feet are on the ground with zero speed. The torso is at main speed, which is usually at around 1 m/s. The wheels of the bike have different speed components depending on the position. Consequently, a road user detected by the radar will produce different Doppler speed responses in radar cube or different distribution of targets at the same position with different speed. In addition, if the speed distribution is observed over time, the speed of different road users will show different modulations. For example, the speed of the arms of a pedestrian will be higher when they are reaching out and lower when

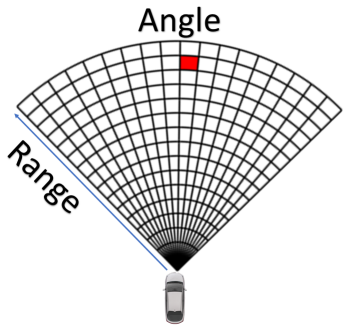


Figure 2.8: The range-angle image of the radar low-level data. The range resolution and azimuth angle resolution does not change with range. Therefore the grid cell gets larger when the range is larger.

they are withdrawn. The micro-Doppler signature caused by a pedestrian moving back and forth is shown by Figure 2.10. The positive speed means the pedestrian is coming closer. The wide band around the bulk translational speed can be observed. If the image is zoomed in, the gait pattern of the pedestrian is also visible.

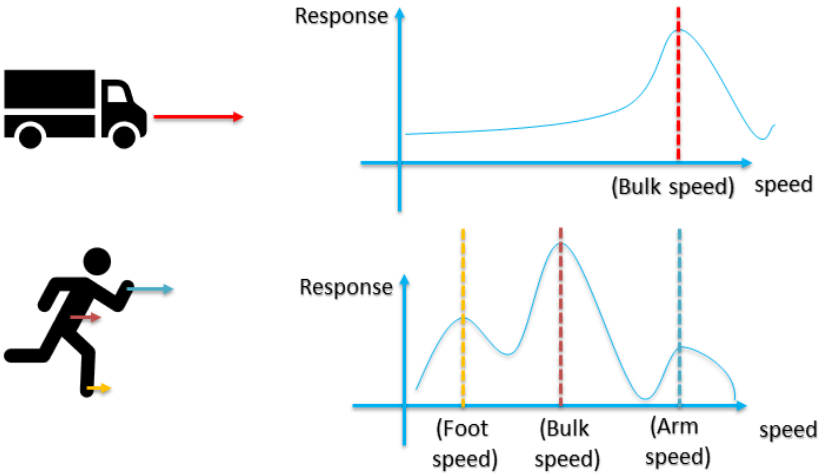


Figure 2.9: The speed distribution of a rigid body (e.g. vehicle) and a deformable body (e.g. pedestrian).

2.2. High-level data algorithms

Radar high-level data is a target list. Each target has a 2D position in the horizontal plane, a Doppler speed relative to the ego-vehicle and an RCS value to measure the reflectivity. In this chapter, these values are denoted by (x, y, v_r, σ). The number of targets per radar frame varies from 100 to 300, see Figure 2.11. These targets can be seen as a point-cloud, but not as dense as LiDAR's. This is the data one can get out-of-the-box from a commercially available radar.

Many radar based road user detection algorithms using high-level data are following a clustering based pipeline [45, 46, 62, 71]. The main stages of this pipeline are shown by the flow chart in Figure 2.12. The details will be covered by the following subsections. The pipeline starts with a clustering method, which clusters targets into object proposals. After the clustering stage, a set of features are extracted from each cluster. These features are usually calculated to measure the distribution of targets' properties in each cluster, such as the speed distribution, the reflectivity distribution and the spatial distribution. For each cluster, these hand-crafted features form a feature vector. This feature vector will be classified by a classifier, such as random forest [29], support-vector machine (SVM) [13], multilayer perceptron (MLP), etc.

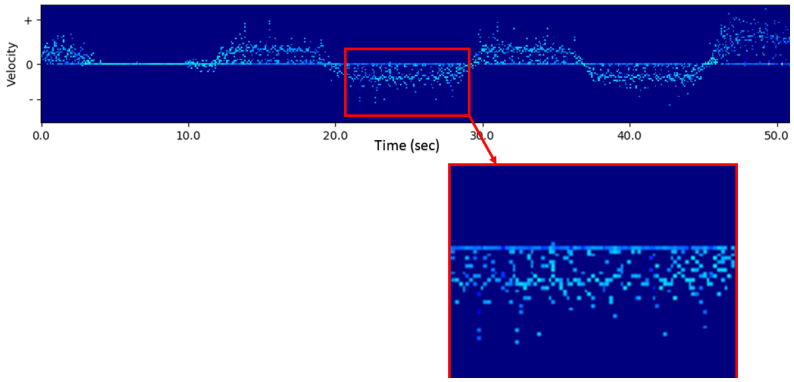


Figure 2.10: The micro-Doppler signature of a pedestrian moving back and forth. The positive value means the pedestrian is coming closer. The wide band around the bulk translational speed can be observed. If the image is zoomed in, the gait pattern of the pedestrian is also visible.

Figure 2.11: This figure shows the target list of a single radar frame from the top-down view. The vehicle is heading towards the left. The blue points are radar targets.

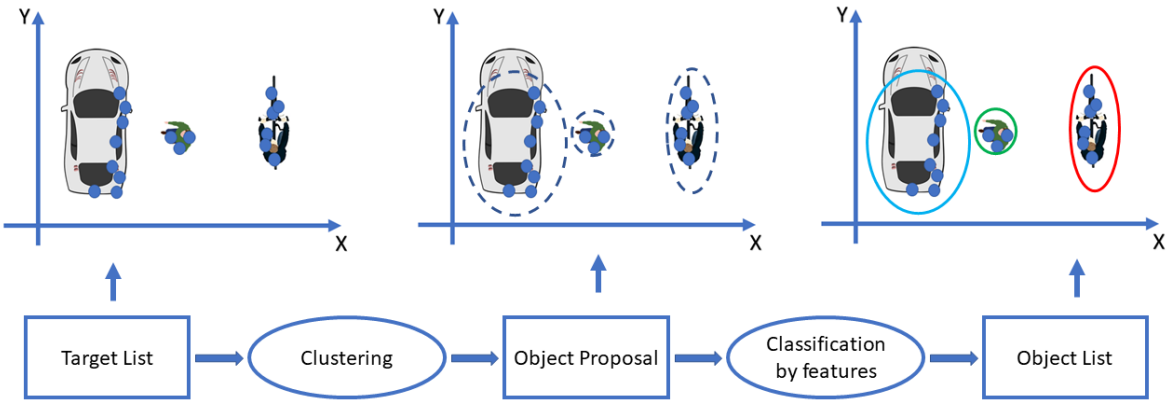


Figure 2.12: Main stages in clustering based pipeline for radar high-level data. Firstly, the targets are clustered into different object proposals. Afterwards, a set of features are extracted from each cluster. These features are given to a classifier. The labels predicted by the classifier will be assigned to the targets inside each cluster.

With the development of high-level data algorithms and deep learning methods for LiDAR point-cloud, some deep learning methods originally for LiDAR point-cloud are adapted for radar target list. These methods will be elaborated in subsection 2.2.5.

Apart from the major steps in the pipeline, there are also some preprocessing steps applied to radar high-level data, such as the ego-motion compensation and multi-frame aggregation. These processing steps are covered by subsection 2.2.1.

2.2.1. Pre-processing

Ego-motion compensation and filtering

Radar is only able to measure the relative speed of each target in radial direction. When the vehicle is moving, targets from static objects such as buildings, trees and street lights will also have a non-zero relative speed, and the measured speed of other moving objects such as pedestrians, cyclists and cars will be shifted due to the ego-motion. The speed with respect to the ground is more meaningful than the relative speed with respect to the radar, which is fluctuating with the ego speed. Therefore, ego-motion compensation is necessary. A typical ego-motion compensation method is proposed by [37].

Given the velocity of the ego-vehicle rotation center v_x^{ego}, v_y^{ego}, the yaw rate ω^{ego} and the mounting position of the sensor with respect to the rotation center s_x, s_y, and assuming that the vehicle is not drifting (i.e. the lateral velocity v_y^{ego} is zero), the linear speed of the sensor can be calculated by Equation 2.11.

$$\begin{pmatrix} v_x^{sen} \\ v_y^{sen} \end{pmatrix} = \begin{pmatrix} v_x - \omega^{ego} s_y \\ \omega^{ego} s_y \end{pmatrix} \tag{2.11}$$

After getting the absolute speed of the sensor, i.e. the ego-motion, an expected measured speed $v_{r_{static}}$ at every target's location is calculated by Equation 2.12. This is the speed measured by the radar if the target at that position is from a static object. Recalling the radar is only able to measure the radial speed, this expected speed depends on the azimuth angle.

$$\begin{pmatrix} v_{r_{static}} \\ v_{t_{static}} \end{pmatrix} = \begin{pmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) \end{pmatrix} \begin{pmatrix} v_x^{sen} \\ v_y^{sen} \end{pmatrix} \tag{2.12}$$

The measured speed v_r of the target is a superposition of the expected speed $v_{r_{static}}$ and the speed of the target itself $\hat{v_r}$ with respect to the ground. Therefore, $\hat{v_r}$ can be calculated by Equation 2.13. The relation between the expected speed $v_{r_{static}}$, the speed with respect to the ground $\hat{v_r}$ and the measured speed v_r are demonstrated in Figure 2.13.

$$\hat{v_r} = v_r - v_{r_{static}} \tag{2.13}$$

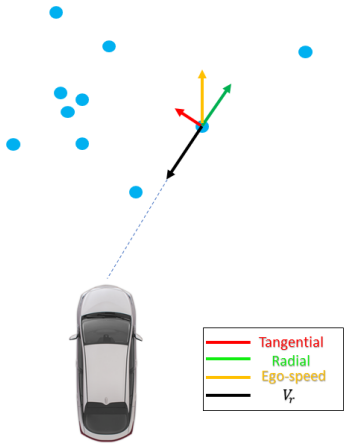


Figure 2.13: The ego-motion compensation geometry. The yellow vector is the ego-motion speed, which is the same with sensor speed. The green vector is the radial component of the ego-motion speed, i.e. the expected speed $v_{r_{static}}$, and the red vector is the tangential component (not measureable by radar). The black vector is the measured speed v_r. The compensated speed $\hat{v_r}$ is the difference between the black vector and green vector.

Micro-Doppler signatures are not observable if the object is not moving. Moreover, the distribution of targets of static objects differs from targets of moving objects. Therefore, after the ego-motion compensation, the targets from static objects are usually removed in radar high-level data pre-processing stage and only dynamic ones are processed. The usual solution is to set a threshold empirically for the compensated speed $\hat{v_r}$, such as the threshold in [60]. The other possible solution is to use the standard deviation of the compensated speed to set the threshold, which is used by [37].

Aggregation of multiple radar frames

The maximum range measured by a radar is usually higher than 100 m and the field of view (FOV) is usually larger than 45°. However, in comparison with the big area covered by FOV, there are only hundreds of targets detected per radar frame. The low density of the targets causes two problems. First of all, most clustering methods are density based [19], which assumes the density inside the cluster is higher than the density outside the cluster. If the targets are so sparse that the distance between targets are at the same level of distance between objects, this assumption does not hold. The extreme case is that when there is only a single target from the object, the parameter setting usually used by the density based clustering method will ignore that single target. Secondly, the statistical features to describe the distribution of targets in each cluster will be less reliable, because statistics of small amount of samples are sensitive to outliers. To address the sparsity problem, multiple radar frames are usually aggregated to generate a more dense point-cloud [61, 62, 71].

The time window to aggregate the frames depends on the requirement of the algorithm. In clustering based methods, such as [61] and [71], two radar frames are aggregated, causing a 150 ms time window. In contrast, the deep neural network in [62] needs more targets in each frame for its multi-scale grouping (MSG) module. Therefore, in that study the data are accumulated over 500 ms. The frames covered by the time window are transformed into the vehicle coordinate system of the earliest measurement.

The models mentioned above do not use time sequence as a feature. In contrast, using recurrent neural networks (RNN) considers the targets from different time steps explicitly. In that case, more radar frames are used for a single prediction. For example, the LSTM algorithm used in [71] takes a sequence of length 8 as its input.

The delay caused by the aggregation varies from 0.15 s to 0.8 s. This is the main disadvantage of aggregation. In real time driving scenario, such delay is not acceptable.

2.2.2. Clustering method

The information from a single target is limited, therefore, in most radar based road user detection and classification algorithms, the targets are firstly grouped into object proposals by a clustering method.

One of the most commonly used clustering methods is called density-based spatial clustering of applications with noise (DBSCAN) [19]. This algorithm assumes that the density inside a cluster should be higher than the density of the background. After the feature space is defined, the targets that have enough neighbouring targets are set as core targets. The minimum number of targets around each core target N_{min} is the first hyper-parameter for this clustering method. Another target is directly reachable from a core target if the distance between them are less than the pre-set minimum distance eps, which is the other important hyper-parameter for DBSCAN. If one target is directly reachable from the other targets, but it is not a core target, this target formulates the edge of a cluster. The main reason for using DBSCAN is that this method does not require the number of clusters as input, which is difficult or even impossible to estimate beforehand in driving scenario. Considering the majority of the clustering methods are different variants of DBSCAN, the current thesis only covers related work that uses DBSCAN for clustering.

In [61], a comprehensive study on DBSCAN method for radar targets is performed. A high resolution radar is used, similar to the one used in [60] by the same authors. Therefore the number of targets detected by the radar will increase compared with targets from a typical commercial radar, which benefits the clustering stage. The feature space for clustering is formed by 2D position and speed. Targets from two time steps are aggregated and then

clustered. The minimum number of targets in a cluster is set as 2 and the minimum distance is 1 in the feature space. The clustering method in this paper is used by many studies such as [45] and [39].

The similar DBSCAN method is used in [46] and [45] by a single radar frame. As is mentioned in [46], if only one radar frame is used, there are many cases where the number of targets of one cluster is equal to 2, which is the minimum targets to formulate a cluster. In other words, many clusters have very few points in it.

In [71], a custom metric to measure the distance in space, time and velocity is used for DBSCAN. After the clustering stage, some targets are mis-clustered. To be specific, some objects (road users) are falsely merged together into one cluster or some targets from background are merged to road users. Some large objects are clustered into several different objects. To compensate for these types of clustering errors, a manual correction is applied to the mis-clustered data. After the correction, targets inside the corrected clusters are re-clustered to form a set of new clusters for data augmentation. From the re-clustered clusters, an additional set of clusters is created by randomly dropping 40% targets. Due to this manual correction, the problems caused by the clustering error is avoided in their training and testing stage. However, this manual correction makes the data not realistic enough for real-life driving scenario, since no manual correction can be applied there.

In [70], the extended DBSCAN (EDBSCAN) with a new size metric is proposed, which is able to deal with the significant density variation in the feature space. However, the clustering method in this study only considers how to effectively cluster pedestrian out of the background. The parameter setting also serves for this purpose. It does not consider how to effectively cluster objects from different classes.

Unlike the manual fine-tuning of hyper-parameters mentioned above, in [63], to use DBSCAN more effectively for different road users, the parameters are learned by a manually clustered training dataset to optimize the performance. On the other hand, manually annotated dataset is limited in size and diversity. Therefore it limits the generalization ability of those learned parameters.

The clustering methods mentioned above are effective in many cases and generates useful object-level information. However, DBSCAN has its limitations under certain conditions. First of all, the minimum number of targets per cluster is different for different road users. A single pedestrian usually has one or two targets while a car usually has multiple targets. In order to cover different classes, this parameter is set based on the smallest class, usually the pedestrian class. Therefore, when multiple road users are moving in a group, e.g. pedestrian group, they will be clustered into the same object. Secondly, the minimum distance to form a cluster also depends on the class of the road user. A car usually has larger extension in space than a pedestrian. If the distance is decided by the scale of the smallest road user, i.e. the pedestrian, the targets from a car will possibly be separated into two clusters. Last but not least, when the targets are sparse, the following feature extraction stage will suffer from outliers.

2.2.3. Feature extraction

From the object list generated by different clustering methods, features are usually extracted based on different speed, reflectivity and space distributions. Methods in [25, 26] use features of the different profiles in range dimension and Doppler dimension and combine these features with other features such as RCS and signal-to-noise ratio. Absolute velocity is also considered with a low weight. The experiment is performed using a 24 GHz radar, of which the resolution is lower than 77 GHz radar.

Following their work, the study in [46] uses 79 GHz radar with a higher resolution than a typical commercial radar. Some features used by [25, 26] are also used by this study, such as the extension, variance and deviation of the speed, etc. There are two significant changes. Firstly, due to the sparsity of radar targets, a feature that measures how much is the object different from a solid body is removed. Secondly, the extension in x- and y-direction within the horizontal plane are summed up to create a new feature.

In [71], more features are extracted from each cluster. The features can be divided into two parts. The first part is directly calculated from the targets' high-level features. For

example, the mean value, standard deviation and the spread of RCS value and ego-motion compensated speed. 18 features are created in total. The second part is indirectly calculated by RCS histogram and speed histogram. The number of bins for each histogram is fixed as $n_{bins} = 8$. In this case, the bare count of an individual bin significantly varies among different clusters. Therefore, the histogram is normalized. The position and count of each bin is encoded into a single value, resulting into 8 values per histogram. The 8 values from each histogram are combined into 16 indirect features. In the end, 34 features from each cluster are used for classification.

In [56], different features are compared and selected for high-level data. In this study, over 50 features are generated. The increase comes from introducing further statistics of the same domains: speed, space and reflectivity distribution. Therefore, many of them are correlated to each other. A feature selection algorithm called backward elimination is performed in order to select a smaller subset. The elimination is done in a greedy fashion. Once a feature is removed, this feature will not be considered anymore. On top of the general feature selection, a classifier-dependant feature selection scheme is provided in [57] by the same authors.

2.2.4. Classification

Many studies choose random forest as the classifier, such as [46, 56, 71]. Random forest is originally proposed in [29], which is a combination of multiple decision trees with bootstrapping and feature bagging in order to improve generalization ability. Decision tree is a tree-like structure, which is formed by different nodes and edges, see Figure 2.14. The nodes that do not have any children node are called *leaves*. The node that does not have parent node is called *root*. Starting from the *root*, one value from the feature vector is tested by an if-then-else question, e.g. If feature $F_1 > 50$ then F_1 goes to $Node_1$, else it goes to $Node_2$. After the testing, the samples are split to two children nodes. The testing stops until all the training data are classified or the minimum number of samples in a node is reached. Then the node becomes a *leaf*. The decision tree classifier has good capacity for different input and output types, e.g. classification, regression. It is easy to understand and visualize. The input to a decision tree does not need careful preprocessing, e.g. normalization. However, decision tree has a big risk to overfit the training data. To solve this problem, random forest is introduced. The main strategy used by random forest is ensemble learning via bootstrapping. Bootstrapping is a procedure that randomly samples the training data repeatedly with replacement into subsets. The subsets are used to train different decision trees. During training, feature bagging is applied, which selects a random subset of features at each candidate split, i.e. node in the tree. At the prediction phase, the predictions of multiple decision trees are either averaged by the number of trees or used for majority voting. Random forest has three advantages. First of all, similarly with decision tree, it is not necessary to normalize the data. Secondly, the training procedure of a random forest does not take a lot of time. Last but not least, it is reliable for different distributions with good generalization ability. In radar road user classification algorithms using high-level data, the random forest classifier performs well. In [71], random forest achieves slightly worse accuracy with much less targets than LSTM. In [46], the ensemble bagged trees achieves the highest accuracy.

Support-vector machine (SVM) [13] is used in [25, 46], which discriminates different classes (usually two) by a decision boundary and optimizes the margin between this boundary and training data. The original SVM is a linear classifier. Its decision boundary is a hyperplane. If kernel functions are used, the decision boundary can also be nonlinear, which makes SVM more flexible. SVM classifiers with different kernel functions are used by [46]. In their study, kernel functions with higher order (cubic and quadratic SVM) are more accurate than the linear SVM. However, SVM with Gaussian kernel does not perform well, which in theory has the infinite dimension map. This result implies one of the disadvantages of SVM classifier. Choosing a suitable kernel function is not easy.

Multi-layer perceptron (MLP) is used in [46], which is a simple neural network with only fully-connected layers. A MLP with one hidden layer achieves a good trade-off between efficiency and accuracy.

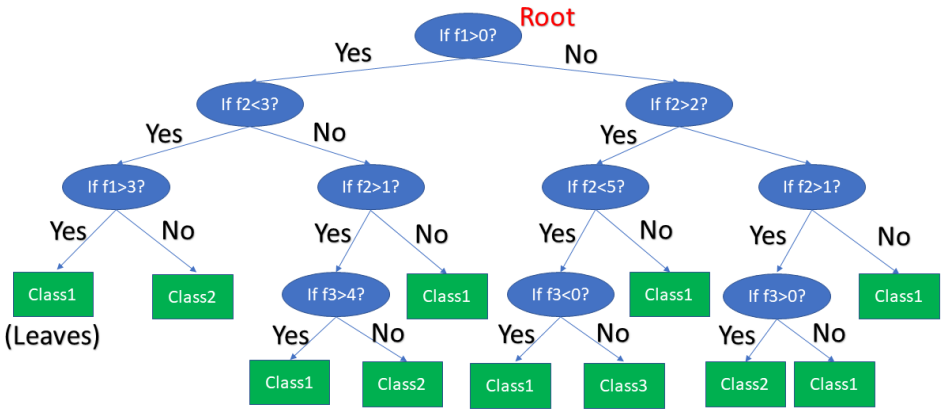


Figure 2.14: The structure of a decision tree.

2.2.5. End-to-end solution

Considering the similarity between radar target list and LiDAR point-cloud, the end-to-end solutions originally for LiDAR point-cloud can be adapted for radar target list. Among them PointNet [47] is the first architecture that uses point-cloud directly as input, instead of projecting points to images [73] or assigning them to voxels [75]. It uses convolutional neural networks in combination with max-pooling to generate a feature vector directly from a point-cloud. Due to max-pooling, the output is not influenced by the order of input points. On top of PointNet, PointNet++ [49] is introduced. Different from PointNet which only extracts features from the entire point-cloud, the points are first sampled into different groups with different radius. This step is called multi-scale grouping (MSG). After each layer for MSG, a feature vector for the group is generated by a small PointNet and this vector becomes a single point for the next MSG layer.

In the study of [62], PointNet++ is used to segment radar point-cloud. The main architecture of PointNet++ is kept. Compared with the original PointNet++, there are two main changes. The first change is the feature space used for neighborhood search. The original PointNet++ uses the 3D position of each LiDAR point, while the PointNet++ for radar only uses 2D ground plane position. The second change is the feature for radar targets segmentation. The features are augmented by the RCS and velocity. This study achieves good target segmentation results on a large dataset. However, in order to produce a dense point-cloud suitable for PointNet++, the radar targets are accumulated over 500 ms. Targets within the 500 ms time window are shifted back to the coordinate system of the first time step.

Inspired by Frustum PointNet [48], [16] extends PointNet++ by adding bounding box proposal and regression for cars. The research uses a single sweep of two radars, which is suitable for car detection, since they usually have several reflections. The authors claim that the method can be extended to multi-class detection tasks. However, it is unknown if a single frame of targets would be enough for smaller road users.

2.3. Low-level data algorithms

The low-level radar data is the data after the angle FFT but before target list generation. For more details about radar DSP and target list generation, please refer to section 2.1.1. As shown by Figure 2.15, the low-level radar data is a three dimensional matrix. A cell inside the matrix corresponds to a certain range, Doppler speed and azimuth angle. Due to its shape, the low-level radar data is also called the radar cube. The length of the cell is the resolution of low-level radar data in each dimension. Each entry of this 3D matrix is the radio wave response (usually magnitude) for the range, speed and azimuth angle indicated by the cell indices.

In most cases, the positioning accuracy of low-level data is usually not as good as high-level data, since sophisticated processing steps to calculate more accurate position and speed of each target only take place after targets are generated from the radar cube. However, low-

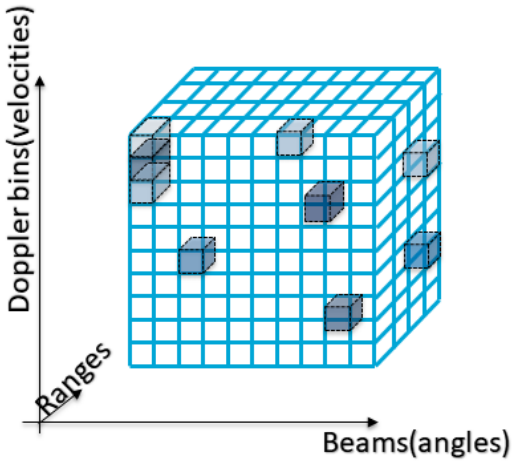


Figure 2.15: Radar low-level data has three dimensions, which are range, Doppler speed and azimuth angle. The entries are magnitudes of radio wave at a certain position with a certain speed.

level radar data includes more information than target list due to less thresholding, which contains characteristic features for road user detection, especially for vulnerable road users. One of the most commonly used features is the micro-Doppler signature. For example, the Figure 2.16a from [60] shows the range-Doppler image of a cyclist, where the wheels and pedals are clearly visible. If the micro-Doppler signature is recorded over time, it will show a modulation by the periodic movement of different parts of a road user, e.g. the different speed of the limbs when a pedestrian is walking. Figure 2.18 shows the micro-Doppler signature of a walking pedestrian.

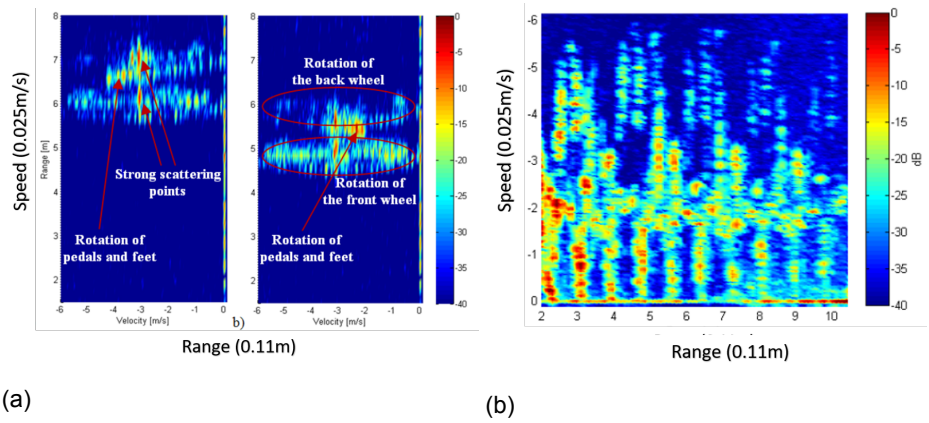


Figure 2.16: (a) The range-Doppler image of a cyclist in [60]. The Doppler signature caused by the wheels, pedals and feet are clearly visible. (b) The range-Doppler image over successive measurement cycles of a radial moving pedestrian [60].

Compared with the target list, the low-level data is less explored. Most research usually focus on a subset of the three dimensions, depending on the experimental setup, their hardware and the problem addressed. For example, if there is only one TX-RX antenna pair, azimuth angle can not be recovered, thus only range-Doppler image is obtained.

Due to the structured format of low-level radar data, an upcoming trend is to apply image processing algorithms. For example, convolutional neural network (CNN) for images are used for range-Doppler image classification [44] and segmentation [74], as well as Doppler-time image classification [3]. For static objects, it is also used for range-Angle image classification [42]. Other than deep learning methods, SURF features [5] are used in [46] for feature extraction. In the following content, different methods will be categorized based on the representation of the low-level data.

2.3.1. Range-Doppler image

The range-Doppler image is a 2D image with axes of range bins and speed bins. An example is shown by Figure 2.16a. This image is generated by the radio wave responses from a single RX antenna after applying range FFT and Doppler FFT. In other words, it is one step before the entire 3D radar cube. Each pixel stands for the magnitude of radio wave responses at that range and speed.

A study that focuses on range-Doppler image itself is performed by [60] using a high resolution radar. The range-Doppler images from successive measurement cycles of pedestrians and cyclists in different orientations are obtained and analyzed. In Figure 2.16b, a radial moving pedestrian shows a characteristic wide band of speed responses, from zero speed to three times of the pedestrian's walking speed (2 m/s). For cyclists, the responses caused by the rotation of wheels and pedals are clearly visible when the bike is moving in radial direction, see Figure 2.16a. Although the reflection is weaker and the distribution is narrower when the road user is moving in lateral direction, due to the high resolution in range and speed measurement, the small radial components caused by the extension of body parts in different angle bins can still be recognized.

Single range-Doppler image can be used for image classification. Following the study in [60], the same authors perform road user (pedestrian, cyclist, car) classification using single range-Doppler image [44]. CNN with fully-connected layers allows only one moving object to be classified per radar frame. Frames that contain multiple road users are removed. A similar algorithm can be found in [1], but this study classifies human and robot rather than road users. The other study for single frame range-Doppler image is done by [45], which uses clustered high-level radar data to select region of interest (RoI) from the range-Doppler image and generates SURF features [5] for classification. In conclusion, the single range-Doppler image is used for road user classification because different road users have different distributions in range and speed.

Other than image classification, range-Doppler image is also used for image segmentation. In [74], the range-Doppler image is segmented by U-net [54] to estimate the 3D position of a car in a static environment. The output of the segmentation network has the same resolution with the input. Therefore, unlike image classification algorithms, by image segmentation the number of objects in one range-Doppler image is not limited to one. To estimate the position, direction of arrival (DoA) is needed. As described in subsection 2.1.1, DoA can only be obtained by range-Doppler images of multiple receivers. Therefore, range-Doppler image from different antennas are used together as multiple channels for the input. In this study, the DoA that should have been done by FFT is learned by deep learning. However, the original U-net structure does not have module for such a transformation, which means the transformation should be learned implicitly. The only result of the experiment is the training and validation loss. The prediction accuracy is not evaluated quantitatively. To the best of our knowledge, this is the only image segmentation method applied to radar low-level data.

Instead of using range-Doppler image from a single time step, there are also studies using range-Doppler images from multiple time steps. The study in [31] performs road user (pedestrian, cyclist, vehicle) classification by range-Doppler image sequence using a recurrent convolutional neural network with transposed convolutional layers called C-LSTM. 5 feature maps from 3 C-LSTM are concatenated for each prediction. Using more frames is helpful to improve the classification result. However the time delay caused by multi-frame classification method does not satisfy the real-time processing requirement.

2.3.2. Doppler-time image

Doppler-time image is widely used for radar object detection and classification [6, 66, 70, 72]. It is generated by accumulating micro-Doppler signature over time. The micro-Doppler signature is generated at a certain range if the radar uses continuous radio wave, such as the micro-Doppler signature in [3], which is shown by Figure 2.17. In this study, the range bins are chosen by OS-CFAR [8] and tracked by Kalman filter [30]. In contrast, if a pulse wave radar is used, the micro-Doppler signature can also be generated without knowing the range of the object, such as the micro-Doppler signature in [32]. In that case, the micro-Doppler signature is the radio wave responses of the entire FOV.

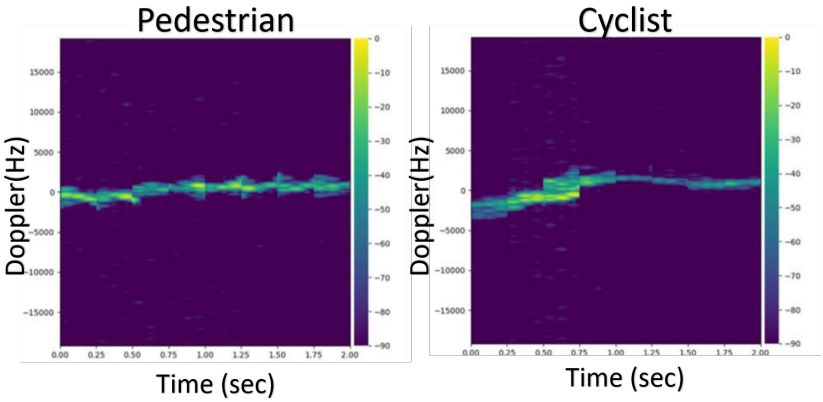


Figure 2.17: The Doppler-time image of a single pedestrian walking (left) and a cyclist (right) [3].

The Doppler-time image contains more information due to larger time window, which benefits pedestrian detection task. When a pedestrian is walking, the micro-Doppler signature will show a gait pattern. For example. the micro-Doppler signature is wider when the legs are opened and narrower when the legs are closed. The arms and torso also have similar effects. The detailed micro-Doppler signature of a walking pedestrian is shown by Figure 2.18 from the study of [6]. To classify a pedestrian by Doppler-time image, the time span of a Doppler-time image is usually longer than 2 s. It is a suitable solution for surveillance [68] which does not require real-time processing. However, for driving scenarios, accumulating signals over 2 s is too long for the vehicle or the driver to react to the situations.

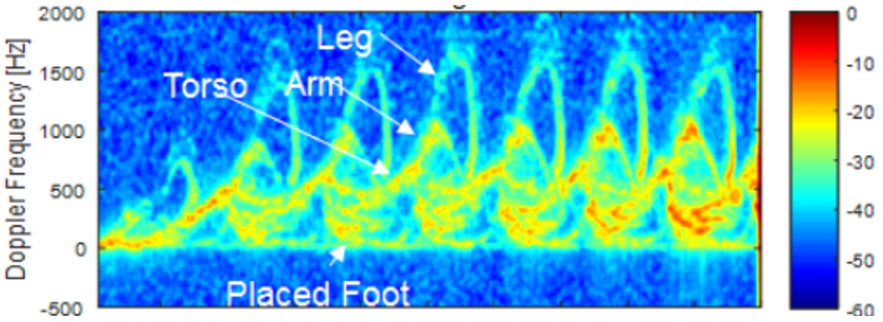


Figure 2.18: The micro-Doppler effect caused by the torso, feet, legs and arms of the pedestrian from the study in [6]. The different parts show different Doppler frequencies and amplitudes.

2.3.3. Range-angle image

Range-angle image contains only grid cells in 2D horizontal plane. An illustration of 2D grid cells is shown by Figure 2.19. This image is a heatmap in 2D space. The values in each grid cell is the reflectivity and/or Doppler speed.

In [42], CNN is applied to range-angle image's RoI for static objects detection, such as car, construction barrier, motorbike, etc. The range-angle image used in the study is shown by Figure 2.20. At the first step, OS-CFAR [8] is applied to the range-Doppler spectrum in order to find potential targets in range and speed dimension. After OS-CFAR, the azimuth spectrum is calculated by angle FFT, resulting in a 3D range-speed-azimuth radar cube. From the radar cube, peaks that are higher than its surroundings are detected. Around each peak, an RoI is cropped. After RoI is created, only a single speed slice is chosen based on the maximum intensity at its range and angle bin. The size of the RoI is 64×66 (range bins $\times$ angle bins). Then this 2D slice is given to CNN for classification.

In [4], range-angle image is used to classify pedestrian out of static objects by a united membership value. The membership value is calculated by the features from a intensity

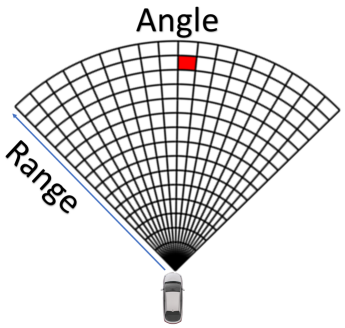


Figure 2.19: The range-angle image of the radar low-level data. The range resolution and azimuth angle resolution does not change. Therefore the grid cell gets larger when the range is larger.

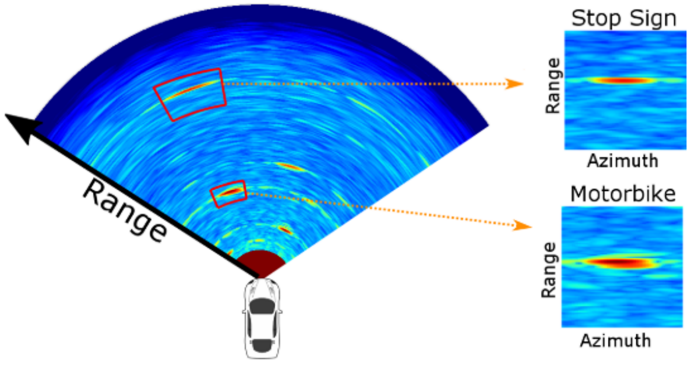


Figure 2.20: The range-angle image used by [42]. Along speed axis of the radar cube, only the slice with maximum intensity is selected as 2D range-angle RoI.

image and a frequency image. Both of them are range-angle images but their pixel represents different features. Intensity image has the intensity value (reflectivity) at each grid cell, while frequency image has the Doppler frequency shift at each grid cell. The latter is proportional to the measured radial speed. Compared with the study in [42] that only uses intensity, the speed in this study will provide extra information. However, adding one speed per grid cell only enables the calculation of the speed distribution in space. The speed distribution inside one 2D grid cell is missing.

Range-angle image can be used to detect pedestrian out of background or detect different static objects, because it has the spatial distribution information. However, the micro-Doppler signature in speed dimension is missing, which is a descriptive feature for road user detection. Furthermore, the grid cell of range-angle image is limited by the range and azimuth angle resolution. The size of a grid cell grows with range and is at the same scale of a pedestrian (see Figure 2.19), which means the low-level data 'samples' the pedestrian in space at a very low sampling rate. In contrast, the resolution in speed dimension is usually better than the resolution in range and azimuth angle. A pedestrian covers more cells in speed dimension compared with the number of cells covered in range or angle dimension. Therefore, including micro-Doppler signature in speed dimension for road user detection is beneficial than only using spatial distribution in range-Angle image.

2.4. Main contributions

Method	Features from	Number of frames	Dataset	Ego-vehicle
Schumann1 [71]	high-level data	2 (150 ms)	real-life	moving
Prophet1 [46]	high-level data	1 (50 ms)	staged	standing
Schumann2 [62]	high-level data	>1 (500 ms)	staged	moving
Prophet2 [45]	range-Doppler image	1	real-life	standing
Perez [44]	range-Doppler image	1	staged	standing
Patel [42]	range-angle image	1	staged	moving
Our method	3D radar cube	1 (75 ms)	moving	moving

Table 2.1: An overview of the most relevant methods. The methods are selected from the related work in the previous sections. The data used for feature extraction is listed in column 'Features from'. The number of frames for a single prediction is listed in 'Number of frames' column. If the actual processing time is mentioned in the study, they are included in the brackets. The 'Dataset' means the recorded road users. The 'Ego-vehicle' means the state of the vehicle used for the recording.

Some most relevant studies from the related work are shown by Table 2.1. Different methods are compared in terms of the data used for feature extraction, the number of frames used for one prediction, the road users in the dataset and the state of the test vehicles. In comparison with other methods, our method uses each single target in high-level data as a region proposal and cropped low-level data as region descriptor. In this way, only single radar frame is used for each prediction, which guarantees the real-time performance. The problems caused by the clustering stage before classification are alleviated. A multi-stage convolutional neural network is used, which is called LLTnet. The network structure is designed for radar based road user detection, which considers the essence of each dimension. The training data contains multiple road users per frame in real life urban scenario recorded by a moving test vehicle. The output of LLTnet is target segmentation. After the segmentation, the label of targets can be fused into an object list. The method is not only evaluated by target-level classification accuracy but also by object-level classification and localization accuracy.

3

Methodology

The proposed pipeline is shown by Figure 3.1. At the first stage, the high-level data and low-level data are preprocessed separately, e.g. ego-motion compensation, noise reduction, etc. After the preprocessing steps, each target from high-level data is mapped to its correspondent grid cell inside the radar cube. This step is called region proposal mapping. Around each mapped target, a small radar cube is cropped. This cropped radar cube is given to a 3D CNN module for multi-scale feature extraction in space and speed, which is followed by a 1D CNN module for one dimensional feature extraction in Doppler speed. After the learnable features are extracted, the feature vector is concatenated with the high-level features and given to the fully-connected layers. In some other studies [62], the high-level features used are $(x, y, \hat{v_r}, \sigma)$, where x and y are coordinates in Cartesian coordinate system. $\hat{v_r}$ is the compensated speed and σ is the RCS value. However, if low-level data is used, the polar coordinate system is more directly linked to the low-level indices, therefore using range and azimuth angle of the targets as high-level position is more meaningful. The high-level features used by the current study are ego-motion compensated speed $\hat{v_r}$, the range r, the angle θ and the RCS σ. By doing low-level feature extraction and high-level feature concatenation, each target has its high-level feature part and low-level feature part. This target is called low-level target, which is also called LLT in the following content. The 3D CNN module, 1D CNN module, concatenation and fully-connected network (FCN) compose the LLTnet, which is introduced in section 3.2. LLTnet performs target segmentation. In order to get object-level information, after each target is given a label, these targets can be fused into road users by a post-clustering stage.

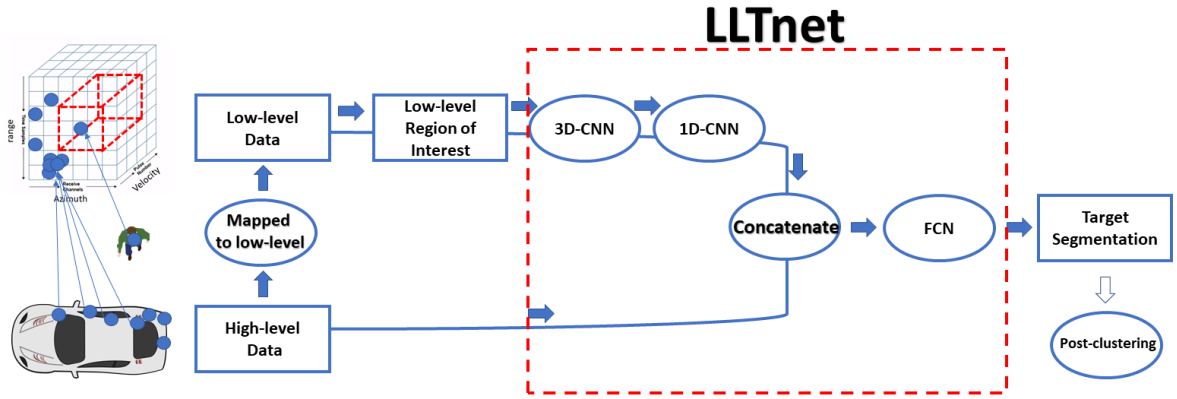


Figure 3.1: The pipeline for the radar low-level data based road user detection algorithm. After the preprocessing steps, each target is mapped to its correspondent grid cell inside the radar cube. A small radar cube around the target is cropped. This cropped radar cube is given to a 3D CNN module for multi-scale feature extraction in space and speed, which is followed by a 1D CNN module for one dimensional feature extraction in speed. After feature extraction by CNN, the low-level feature vector is concatenated with the high-level features and given to FCN for target-level classification, i.e. target segmentation. Afterwards, the segmented targets are given to a post-clustering stage, which fuses the targets into different road users.

3.1. Pre-processing

In this section, the steps for preprocessing are introduced, including ego-motion compensation for high-level data, non-maximum suppression to create low-level data, the region proposal mapping to relate them to each other, the noise reduction and normalization to facilitate training procedure.

3.1.1. Ego-motion compensation

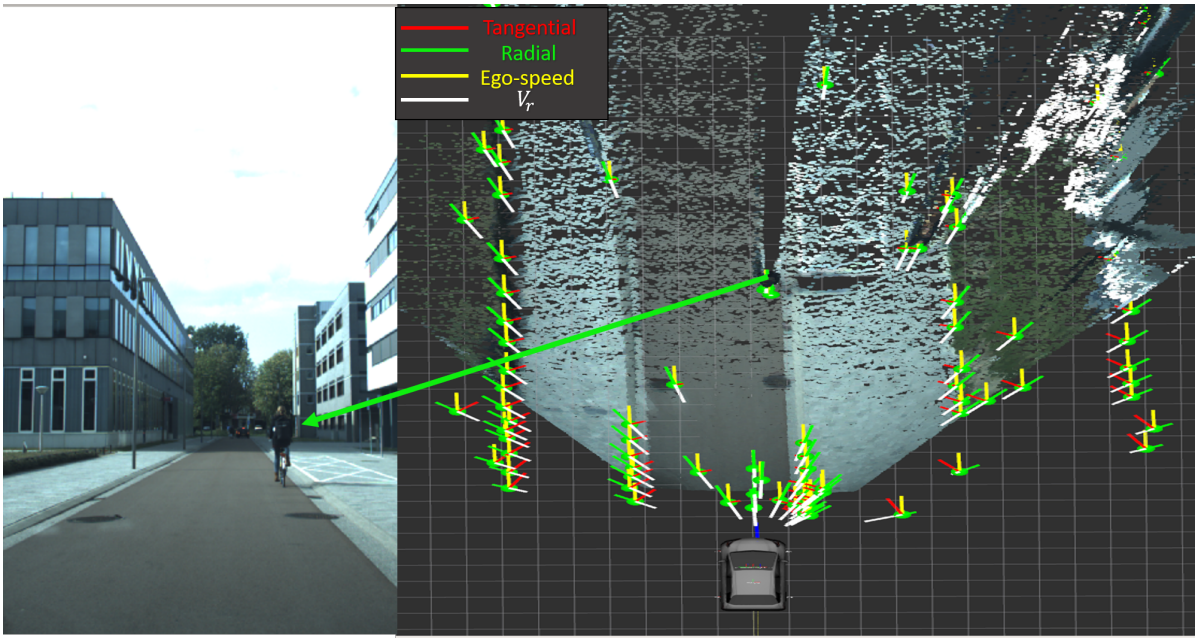


Figure 3.2: The speed components of each target and the corresponding camera image. The yellow vector is the expected speed of each target assuming it is from static objects. The green and red vector are the expected radial and tangential speed at each target position caused by the ego-motion. The white vectors are the relative radial speed between target and radar, i.e. the measured speed. The compensated speed $\hat{v_r}$ should be the difference between the green vector and the white vector. There is a cyclist on the right side in front of the vehicle. The compensated speed $\hat{v_r}$ of the cyclist is similar with the expected radial speed at that position. In this case, the measured radial speed v_r indicated by the white vector is small, which means the relative speed of the cyclist is close to zero.

The compensation rule for the targets is derived similarly as the one in [37], which is explained in subsection 2.2.1. If the yaw rate ω^{ego}, the longitudinal speed v_x and lateral speed v_y at the rotation center of the ego-vehicle are known, the linear velocity of the sensor can be calculated by Equation 3.1. The only difference between the equation in this section and the equation in [37] is from the sign convention. The radar has a sign convention for its targets velocity and the vehicle has a sign convention for its translational speed, rotational speed, lateral position and longitudinal position. These sign conventions will influence the plus and minus signs in Equation 3.1, Equation 3.2 and Equation 3.3.

$$\begin{pmatrix} v_x^{sen} \\ v_y^{sen} \end{pmatrix} = \begin{pmatrix} v_x - \omega^{ego} s_y \\ -\omega^{ego} s_y \end{pmatrix} \tag{3.1}$$

$$\left(v_{r_{static}}\right) = \begin{pmatrix} \cos(\alpha + \beta) & \sin(\alpha + \beta) \end{pmatrix} \begin{pmatrix} v_x^{sen} \\ v_y^{sen} \end{pmatrix} \tag{3.2}$$

$$\hat{v_r} = v_r + v_{r_{static}} \tag{3.3}$$

In Figure 3.2, the compensation for each target is visualized by the image on the right from a top-down view. The yellow vector is the speed of the sensor, which is equal to the expected

speed assuming the target is from static objects. When the vehicle is not turning or drifting, the tangential speed of the sensor is zero, therefore the yellow vector is pointing towards the front. The green and red vector are the radial and tangential component of the sensor speed, which are equal to the expected radial and tangential speed at each target's position. The expected radial speed components of the targets from static objects depend on their azimuth angles. The white vectors are the relative radial speed between target and radar, i.e. the measured speed by the radar. The compensated speed $\hat{v_r}$ should be the difference between the green vector and the white vector. For example, there is a cyclist on the right side in front of the vehicle, where several targets are detected. The compensated speed $\hat{v_r}$ of the cyclist is similar with the expected radial speed at that position. In this case, the measured radial speed v_r indicated by the white vector is small, which means the relative speed of the cyclist is close to zero.

Ego-motion compensation for low-level data is not as trivial as it is for high-level data. Rather than adding or subtracting the expected radial speed for static objects $v_{r_{static}}$ at a certain position, the compensation should be done by shifting entries in the radar cube. The ego-motion compensation for low-level radar data is still an open question. Therefore, in the proposed method, the ego-motion compensation problem of the low-level data is avoided by using the measured speed v_r to crop the low-level features. In other words, at the cropping step, both low-level data and high-level data are without ego-motion compensation, which means the shift caused by the ego-motion does not influence the cropping result.

3.1.2. Non-maximum suppression

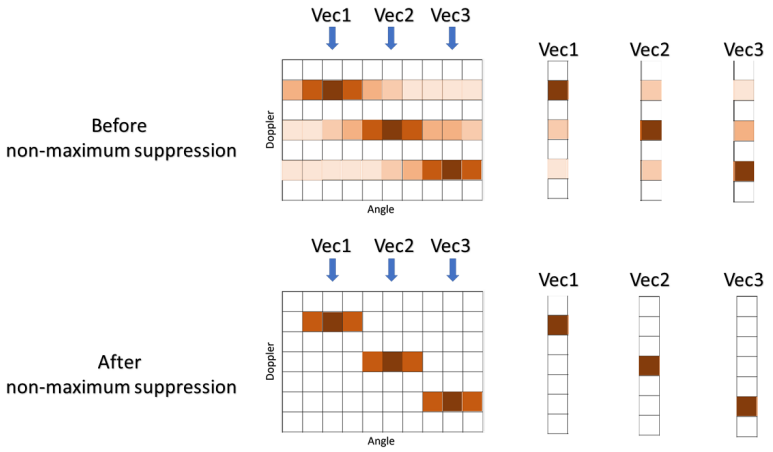


Figure 3.3: The comparison between the cropping before NMS and after NMS. If non-zero entries in all directions are added by different Doppler bins, they interfere with each other and produce very similar speed distributions.

In the low-level data, the FFT is applied separately for each dimension by following the order of range-speed-angle. Details can be found in section 2.1.1. The angle FFT ends up with non-zero values in all directions. These values are relative magnitudes within its own range-Doppler bin to select the most possible angle of arrival. However, when there are two objects at the same range but different angle and Doppler cells, they will interfere each other in Doppler dimension, see Figure 3.3. The Vec1, Vec2 and Vec3 are the micro-Doppler signature at each location. If some values are added by angle FFT in other Doppler bins, this relative distribution will show up elsewhere. Therefore, in the proposed pipeline, only the three largest values from angle FFT are remained for each range-Doppler cell. Other entries at that range-Doppler bin are set as zeros. This process is called non-maximum suppression (NMS). As shown by the second row of Figure 3.3, the noise in micro-Doppler signature at each position is suppressed. The method is similar to what is done by [45]. Instead of selecting only the maximum angle as they suggested, in the proposed method magnitudes from multiple angle bins are remained.

3.1.3. Region proposal mapping

After the preprocessing of high-level data and low-level data, the targets of the high-level data are mapped to the low-level data. This process is called region proposal mapping. Firstly, the range index R_{ind} and speed index V_{ind} of the low-level data counterpart are calculated by Equation 3.4 and Equation 3.5. δr is the range resolution in low-level data and δv is the speed resolution in low-level data. Here the speed of target is the measured speed v_r rather than the compensated speed $\hat{v_r}$ because radar cube is not compensated by ego-motion. For azimuth index, the azimuth angle resolution is not constant, thereby a look-up table is created according to Equation 3.6. If the azimuth angle θ is between θ_i and θ_{i+1}, the azimuth angle index A_{ind} is i. After the target is mapped to the grid cell at (R_{ind}, V_{ind}, A_{ind}), an RoI with a certain size can be cropped. If the number of bins cropped in each dimension are defined as N_{range}, $N_{Doppler}$ and N_{angle}, the RoI around each target is cropped from C_{start} to C_{end} defined by Equation 3.7 and Equation 3.8. After mapping, each target has low-level features. Therefore, in the proposed method, these targets are called low-level targets. LLT is used for abbreviation.

$$R_{ind} = \lfloor \frac{r}{\delta r} \rfloor \tag{3.4}$$

$$V_{ind} = \lfloor \frac{v_r}{\delta v} \rfloor \tag{3.5}$$

$$\theta_i = \arcsin(\frac{i \times N_{phases}}{2 \times N_{beams} \times \frac{d}{\lambda}}), i = -N_{beams}, -N_{beams}+1, \ldots 0, 1, 2 \ldots N_{beams} \tag{3.6}$$

$$C_{start} = (R_{ind} - N_{range}/2,\ V_{ind} - N_{Doppler}/2,\ A_{ind} - N_{angle}/2) \tag{3.7}$$

$$C_{end} = (R_{ind} + N_{range}/2,\ V_{ind} + N_{Doppler}/2,\ A_{ind} + N_{angle}/2) \tag{3.8}$$

The window size (N_{range}, $N_{Doppler}$ and N_{angle}) is set during the experiment, see subsection 4.3.1. If the window size is 1×1, the low-level features are cropped by the first order interpolation of the two nearest range bins. For example, If the target is in the upper half of the range bin, the low-level features are calculated by the linear interpolation of the Doppler responses in ith and $i+1$th range bins. If the window size is larger than 1×1, the low-level features are cropped as they are. Using larger window size in range and angle dimensions makes it possible to consider multiple scales for road users larger than one range-angle bin.

3.1.4. Noise reduction

When the vehicle starts moving, the noise will be higher than the noise in a static radar due to the relative speed between ego-vehicle and static objects caused by ego-motion, see Figure 3.4. To reduce the noise in the low-level data and reduce computational demand, targets from static objects are removed by setting a threshold to the compensated speed $\hat{v_r}$. The threshold should be smaller than the main walking speed of a typical pedestrian (1 m/s). In the current study, the threshold is set as 0.3 m/s. Once the static targets from the high-level data are removed, their correspondent bins in low-level data are also set to zeros. The indices of those bins are calculated by the equations used in subsection 3.1.3. The reason for removing low-level corresponding part of high level static targets is explained as follows.

As shown by Figure 3.4, the central column of a range-Doppler image represents the radio wave responses for 0 m/s. The thick line with a lot of non-zero entries are caused by static objects. If the vehicle is not moving, they are overlapped, as shown by Figure 3.4 (a). When the vehicle starts to move, the responses caused by the static objects will be shifted due to ego-motion. In Figure 3.4 (b) it's shifted to the right because the static objects are approaching the vehicle. However, when the background becomes cluttered, the straight line caused by static objects becomes a curve or multiple curves at near ranges (Figure 3.4 (c)). These curves in this case are caused by a row of static objects distributed longitudinally, e.g.

a row of street lights, a building, a line of parked bikes, etc. These objects have the same relative speed, but they cover a wide band of azimuth angles. The geometry that shows the formation of the curve is in Figure 3.5. When the car is moving in longitudinal direction, the static objects will have the same relative longitudinal speed caused by the sensor (yellow vectors). However, radar is only able to measure their radial component (green vectors). This component is projected by the azimuth angle (blue lines in Figure 3.5). When the objects get closer, the azimuth angle becomes larger, therefore the radial component of the radial speed (green vector) gets smaller. After the difference between the radial speed (green vector) and longitudinal speed (yellow line) is large enough to be seen on the range-Doppler image, a curve diverges from the main thick line. As long as the longitudinal speed of the vehicle does not change, the relative speed of static objects does not change, thereby this diverging point only depends on the azimuth angle. At the same azimuth angle (θ in Figure 3.5), when there are multiple columns of static objects with different lateral positions (the column on the left and the column on the right), their range will be different, which makes the curves in low-level data start to diverge from different range bins.

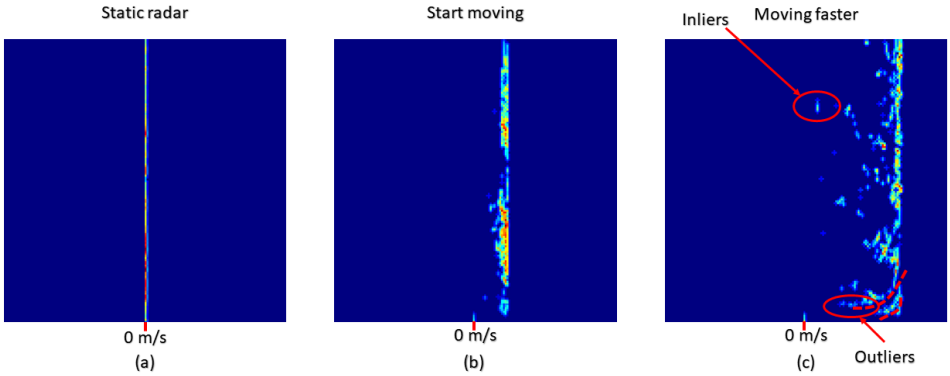


Figure 3.4: The range-Doppler image for different ego-motion speed. the central column of a range-Doppler image indicates $0m/s$ measured speed. The thick line are caused by the static objects. If the vehicle is not moving, these two lines are overlapped, as shown by (a). When the vehicle starts to move, this thick line caused by static objects will be shifted according to the moving speed, as shown by (b). When the background becomes cluttered, this thick straight line becomes a curve or multiple curves at near ranges shown by (c).

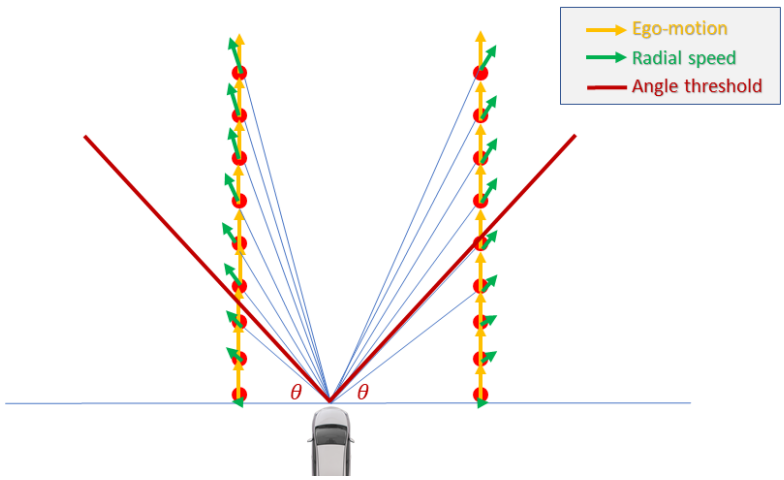


Figure 3.5: The geometry of speed projection that creates the curve in range-Doppler image. When the ego-vehicle moves at a certain speed, the radial relative speed of static objects change with their azimuth angle. When the angle exceeds a certain value, the responses of these static objects diverge from the main responses distribution and generate a curve in range-Doppler image.

This artifact introduces extra noise to the low-level data, because the speed distribution of road users is represented by the non-zero values cropped from Doppler dimension. The inlier

is shown in Figure 3.4 (c). If the Doppler responses outside the thick line are not caused by the real moving objects (inliers) but by the static objects with larger azimuth angle, the latter are outliers (also shown in Figure 3.4 (c)). Setting the low-level data corresponding to the static targets as zero is a solution to alleviate this artifact.

3.1.5. Normalization

The high-level features and low-level features are normalized by subtracting mean and dividing standard deviation. The high-level part is normalized feature-wise. Given the input data, four mean values and four standard deviations of four features are calculated separately from $(r, \theta, \hat{v_r}, \sigma)$. While in low-level part, the mean value and standard deviation are calculated from responses of all the Doppler bins. Therefore only one mean value and one standard deviation for low-level features are calculated from training data. During testing, the mean and standard deviation calculated from training data are used.

3.2. LLTnet

In this section, the network structure of LLTnet is introduced as well as the loss function and ensemble learning policy for the final prediction. The input of the LLTnet is the normalized low-level features cropped by region proposals from high-level targets.

3.2.1. Network structure

LLTnet stands for 'low-level target network', which has a 3D convolutional network module, a 1D convolutional network module and a fully-connected module. The structure of the network is shown by Figure 3.6.

At the first stage, the low-level feature cropped from the radar cube by each target are given to a 3D convolutional network. Each 3D kernel extracts radio wave responses around each 3D cell that it applies to. By max-pooling layer with kernel size 1 in speed dimension, the 3D input is only down-sampled in space (range and angle bins), without changing the number of channels (speed bins). The main purpose of the 3D convolutional network is to encode the micro-Doppler signature from an area into several speed vectors. Each element in the new vector still represents the radio wave responses at a certain speed bin, but it also encodes information from neighboring speed bins. In this way, the essence of the speed dimension is preserved.

The encoded speed vectors are given to a 1D convolutional layers, which down-sample the input vectors in speed dimension and give it more channels at the same time. At the last 1D convolutional layer, using C as the number of channels and D as the number of elements remained from the previous layers, the output is C vectors with D elements. The $D \times C$ elements are flattened into a single vector and concatenated with high-level features. Afterwards, this new vector is given to fully-connected layers. By applying 1D convolution, the number of parameters, i.e. the size of the model becomes less. Moreover, a 1D convolution can achieve translational invariance in the speed dimension. After the fully-connected layers, each target is given a label. In other words, the task performed by LLTnet is target segmentation.

3.2.2. Loss function

The loss function used by LLTnet is a multi-class cross-entropy loss given by Equation 3.9. The $\vec{x}$ in the equation is a vector of softmax score for each class predicted by the model, which is given by Equation 3.10. The loss for different classes are weighted because the training data is unbalanced in number of targets of each class. The weights in Equation 3.9 are calculated by the reciprocal of the number of reflections of each class in the training set, see Equation 3.11.

$$\text{loss}(\vec{x}, class) = weights[class]\left(-\vec{x}[class] + log\left(\sum_j exp(\vec{x}[j])\right)\right) \tag{3.9}$$

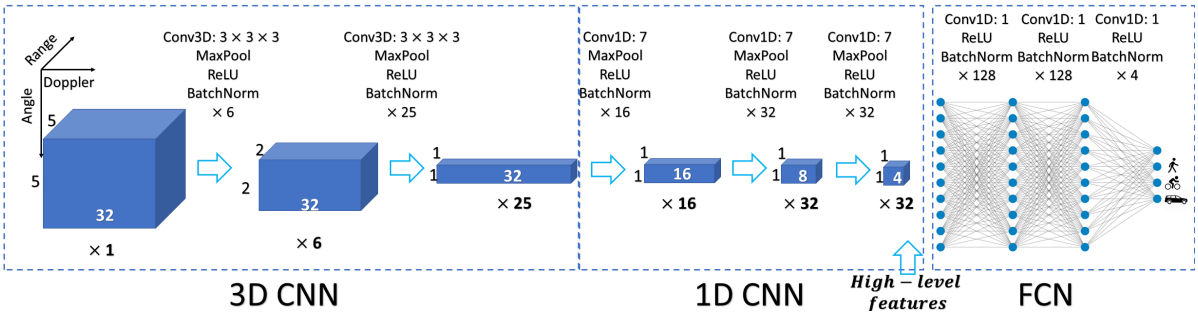


Figure 3.6: The network structure of LLTnet. The 3D CNN down-samples the input in 2D space (i.e. range and angle bins) and encodes the speed information into multiple channels. The 1D CNN is applied to the speed dimension to learn the speed distribution with translational invariance. Finally the fully-connected layers remap the features into different classes through two hidden layers.

$$\text{Softmax}(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{3.10}$$

$$weights = (\frac{1}{N_{class0}}, \frac{1}{N_{class1}}, ...) \tag{3.11}$$

3.2.3. Ensemble learning

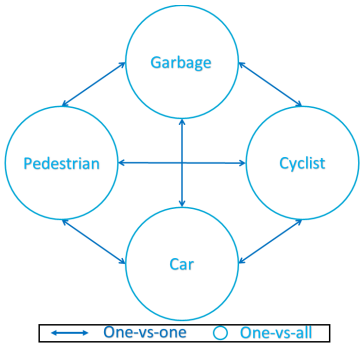


Figure 3.7: The diagram for the ensemble learning. The lines indicate one-vs-one (OVO) models and circles are used for one-vs-all (OVA) models. OVA model classifies a single class against the others, e.g. pedestrian-vs-all, cyclist-vs-all. OVO model distinguishes two classes from each other, e.g. pedestrian-vs-cyclist.

In radar based road user detection task, the difference between different road users is smaller than the difference between road users and background. For example, a laterally moving cyclist with a small speed is similar to a pedestrian. In this case, the wide distribution in speed dimension of the cyclist is caused by different angles. Other than the inter-class similarity, the weights calculated by the reciprocal does not fully compensate for the imbalance of samples from different classes. Further fine-tuning of the weight can be used as an extra compensation. However, once a weight is modified empirically, it influences the other classes in different manners. Therefore, in order to improve the inter-class classification performance, rather than tuning the weights manually, an ensemble learning by binary classification is performed according to the scheme in [57]. Instead of a single multi-class network, two sets of binary models are trained, namely one-vs-all (OVA) models and one-vs-one (OVO) models. Their relations are shown by Figure 3.7. OVA model classifies a single class against the others, e.g. pedestrian-vs-all, cyclist-vs-all. OVO model discriminates two classes from each other, e.g. pedestrian-vs-cyclist. Each OVO model gives a score for the class i against class j as p_{ij}, and the score for the class j against class i can be calculated by $1 - p_{ij}$. Similarly, a OVA model gives a score p_i for class i against the rest classes.

When a new input is given, the scores of it being class i is calculated. The scores for different classes are compared. The class that has the highest score is chosen as the predicted class. This process is given in Equation 3.12

$$\mathrm{id}(x) = \underset{i \in 1,\dots,K}{\operatorname{argmax}} \sum_{j=1, j \neq i}^{K} p_{ij}(x) \cdot (p_i(x) + p_j(x)) \tag{3.12}$$

3.3. Post-clustering

Compared with target segmentation, the object-level information can be beneficial for tracking and decision making. Therefore, after LLTnet performs target segmentation, the proposed pipeline also provides a step to fuse targets into different road users. This is done by a post-clustering step with DBSCAN. The feature space is still made by 2D ground plane position and radial speed. Unlike the clustering step before classification used by the baselines, the label of targets are considered during the parameter setting of DBSCAN. A pedestrian usually has smaller scale and less targets, thus the N_{min} and eps are both smaller. On the contrary, in most cases a car is larger and more targets will be detected. Bigger N_{min} and eps are used for cars.

4

Experiments

4.1. Dataset

There is one publicly available dataset [11] containing radar recordings at the time of writing. However, that dataset only contains high-level radar data. Considering the lack of suitable dataset and limited diversity of the dataset used by previous studies on low-level radar data, for the presented study, a new dataset is created. It is recorded by a moving ego-vehicle and different moving road users in urban scenarios.

4.1.1. Data collection

The dataset is created by two test drives in urban environment. Each of them lasts for about 30 minutes. One of them is used for training. The other is split to two parts. One part is for validation and the other is for testing. The radar used for recording is a 77 GHz automotive radar mounted under the bumper at the front right of the test vehicle. The maximum measureable range is 100 m with a field of view from $-90°$ to $90°$. The range resolution of the low-level data is 0.42 m. The speed resolution of the low-level data varies from 0.11 m/s to 0.13 m/s. The radar works at a frame rate of 13 Hz. The angular resolution of the low-level data of different angle bins are plotted in Figure 4.1. The resolution on the two edges is much lower than it is at the center. A stereo camera is mounted on the windshield which works at a frame rate of 10 Hz and a resolution of 1936×1216 px. The data from GPS, IMU and odometry are also recorded for the estimation of the ego-motion.

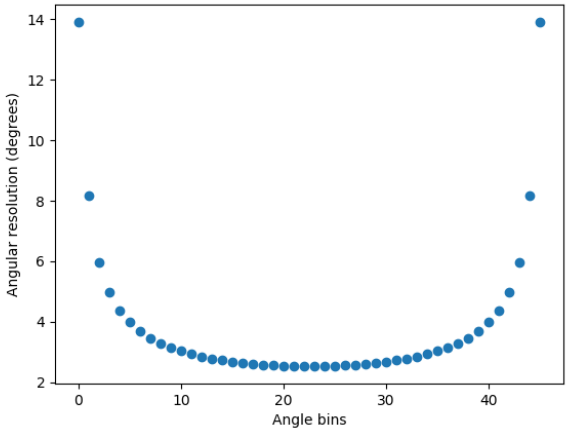


Figure 4.1: The angular resolution of the radar low-level data of different angle bins

4.1.2. Data annotation

Automated annotation pipeline

Since the high-level targets are the input for the region proposal mapping, the annotation aims at giving each high-level target a label. It mainly focuses on three classes, namely pedestrian, cyclist and car. Targets from other objects or road users are annotated as 'others'. The ground truth is provided by a real-time single-shot visual object detection algorithm called SSD [36]. Two SSD models are running simultaneously. One is able to do general object detection. For example, if there is a cyclist, the bike and person will be detected separately. The other SSD is trained by EuroCity Persons (ECP) dataset [9] and is able to discriminate between pedestrians and cyclists, but it does not detect cars. Therefore, during the annotation phase, cars from the first SSD and vulnerable road users from the second SSD are recorded. The output of SSD is 2D bounding boxes in the left image frame of the stereo camera. Depth is estimated by projecting the bounding boxes into the stereo point-cloud computed by the Semi-Global Matching algorithm (SGM) [28], and taking the median distance of the points inside them. Therefore, each 3D bounding box has its distance from the camera, but it does not has thickness in longitudinal direction.

The pipeline of using SSD bounding boxes for annotation is shown by Figure 4.2. Firstly, since the radar high-level targets only have 2D ground plane position, the bounding boxes are projected to the horizontal plane. Different colors are used for different classes (pedestrians in green, cyclists in red and cars in blue). Since the bounding boxes in 3D space only have a distance from the camera, the projected bounding boxes become single lines looked from the top-down view. To annotate the targets, a line should be extended to a region. In the annotation pipeline, this region is created by a 2D viewing frustum in the same 2D plane. A viewing frustum in 3D space is shown by Figure 4.3, which models the field of view of a camera. Since radar receives radio waves just as how camera receives visible light, the frustum can also be used in radar domain, but it only has ground plane coordinates. The extent of the frustum is dependant on the classes. The pedestrian has the smallest extent and the car has the largest. It also depends on the distance, because for larger distance the triangulation of the stereo camera images is less accurate, thus it needs a larger extent to consider a larger uncertainty. y is used to denote the distance between the projected frustum bounding box and the extended upper or lower limit and x is to denote the distance between the bounding box and camera, see Figure 4.2. y_1 is for the upper limit and y_2 for the lower limit. Their relation is given by Equation 4.1 and Equation 4.2. The parameters used in this study are listed in Table 4.1.

$$y_1 = a_1 \times x + b_1 \tag{4.1}$$

$$y_2 = a_2 \times x + b_2 \tag{4.2}$$

	Lower limit		Upper limit	
	a_1	b_1	a_2	b_2
Pedestrian	0.08	1	0.05	0.5
Cyclist	0.1	2	0.06	1
Car	0.12	3	0.07	1.5

Table 4.1: The parameters for calculating the frustum bounding box by Equation 4.1 and Equation 4.2 in 2D horizontal plane of radar targets

Using the 2D frustum, the targets inside each bounding box are labeled. However, there will be some targets that belong to the same object but not covered by the frustum. This happens in the case of cyclist, because the bounding box of the cyclist from SSD does not

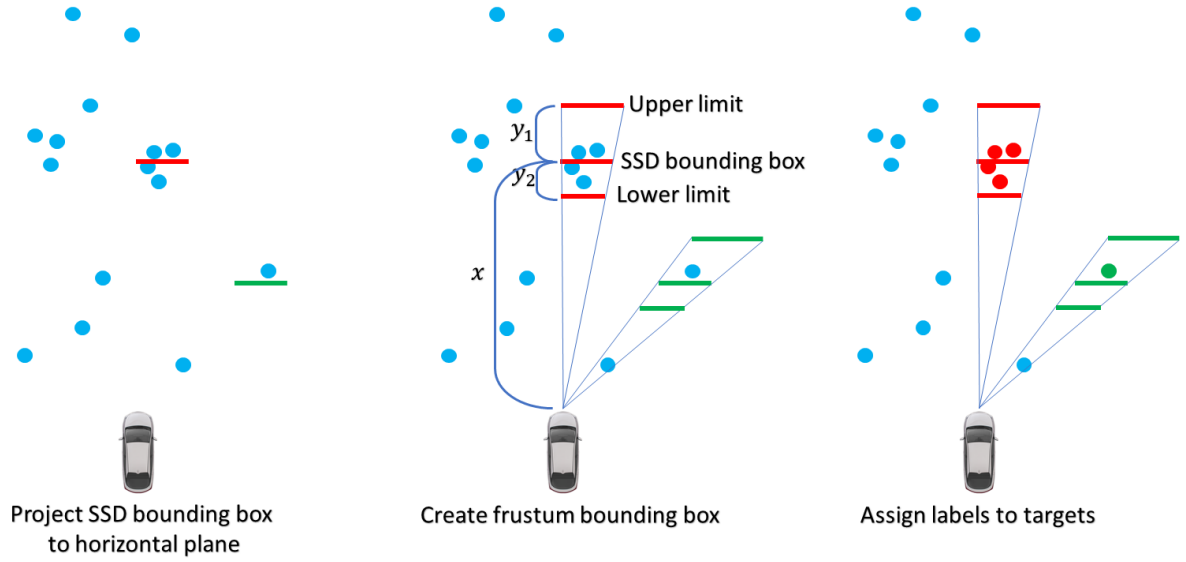


Figure 4.2: The annotation pipeline using SSD bounding boxes. The SSD bounding boxes are projected to the 2D horizontal plane of radar targets. Different colors are used for different classes (pedestrians in green, cyclists in red and cars in blue). Then a frustum bounding box is created around each projected SSD bounding box. The equation and parameters used for creating the frustum bounding boxes are given by Equation 4.1, Equation 4.2 and Table 4.1. The targets inside each bounding box are labeled accordingly.

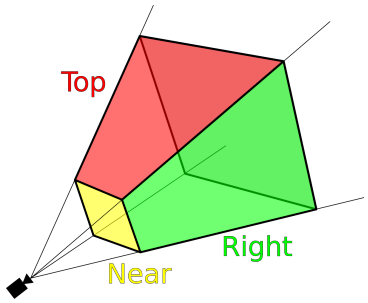


Figure 4.3: The shape of a 3D viewing frustum

enclose the entire bike, as shown by Figure 4.4. Therefore, after being labeled by the frustum bounding box, a label propagation in space is performed. Each labeled target becomes a kernel. The kernel starts from itself and search for its surrounding targets. When some targets are close enough, they will be given the same labels of the kernel. This process can be done iteratively. The targets labeled at the previous iteration will become the kernels again in the next iteration. Since the targets are sparse, after 2-3 iterations, most targets outside bounding boxes are labeled correctly.

Manual correction

SSD has good accuracy for road user detection especially when using it in daylight. In combination with the stereo camera, the vision based automated annotation pipeline work well in most cases. However, it is not perfect. One of the main error sources of SSD detection is the pedestrian-vs-cyclist misclassification. There are two reasons causing this error. In the first case, when the cyclist is in the same orientation with the car i.e. the cyclist and the ego-vehicle are both moving forward, the cyclists are sometimes classified as pedestrians. This is because in such configuration the wheels of the bike are less visible from the back. In the second scenario, when a pedestrian is walking in front of a parked bike, SSD will classify it as cyclist. In addition to pedestrian-vs-cyclist misclassification, there are some pedestrians or cyclists occluded by other objects, such as a pole. They are still visible for SSD and detected correctly, but the triangulation to project the 2D bounding box into 3D is not accurate due

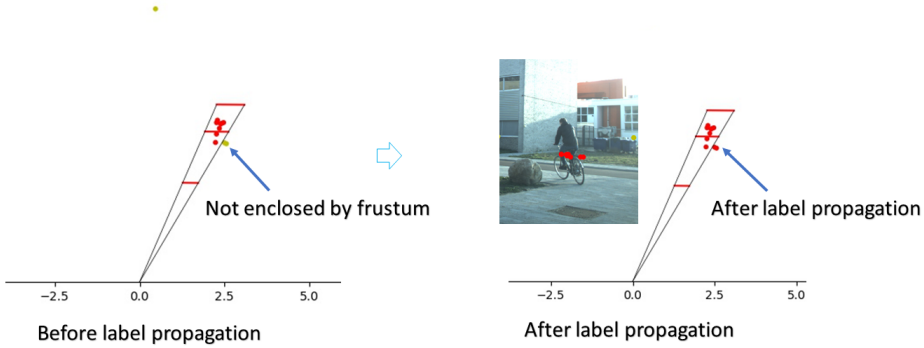


Figure 4.4: The propagation of the labels in space. Sometimes the bike of the cyclist is not covered by the SSD bounding box and the frustum bounding box. The label from the targets inside the bounding box is propagated to the targets that are close to the bounding box.

to the occlusion. These detected road users should be removed. The three cases are shown in Figure 4.5.

Figure 4.5: Misclassifications by SSD

To alleviate the problems caused by SSD, after the automated annotation pipeline, the road users detected by SSD in each frame are cropped from the camera image and exported to different folders with different file names according to their SSD labels, frame ID and object ID. In this way, each class can be quickly checked for mistakenly annotated objects. If such annotation is found, it is fixed manually. For example, if an object is in the folder 'pedestrian' but it is actually a cyclist, this annotation will be moved to the folder 'cyclists'. After this manual correction, the file names in different folders are processed automatically by a program to change the annotation for the relevant frame.

Manual labeling tool development

In addition to the automated annotation pipeline and semi-automated correction for inter-class misclassification, a tool to assist manual labeling is also developed in the current study. This tool has a graphic user interface (GUI) with several functionalities. The interface is shown by Figure 4.7. There are two separate windows in the interface. The window on the left is used for displaying the targets from top-down view and creating a bounding box by dragging and clicking the mouse. The SSD frustum bounding boxes are plotted as a guideline for the user. The targets are projected to the camera image in the window on the right. After

the targets are labeled, they will be given a color. Green is used for pedestrians. Red is for cyclists and blue is for cars. The tool also allows users to jump to a specific frame and set a threshold to the speed. Other functions are listed in appendix C.

4.1.3. Data format

The dataset contains two parts, the recordings and the annotations. The recordings such as high-level data, low-level data and camera images are saved into separate folders with time stamp as their file name. The annotations as well as ego-motion are saved as a list in a json file. Each frame has the 2D annotations in pixel coordinates of the camera image, the 3D annotations in vehicle coordinates, the camera image path, the radar targets path, the labels of each annotation and the ego motion. A typical entry of the annotation list is shown in appendix B. Different messages are synchronized to the time stamp of the high-level data. For example, the camera image used for annotation is the closest camera frame from the frame of the target list. To simulate real-life conditions, e.g. high-level data cannot be mapped to low-level data from the future, the low-level data used by each frame is the latest one.

The statistics of the training data are shown in Table 4.2. The histogram of the number of targets per object is shown in Figure 4.6. It is shown by the distribution that the targets are sparse, with few road users having more than 6 targets. There are many pedestrians only having one target detected by radar. If the clustering based pipeline in the related work in subsection 2.2.2 is applied to these targets, each cluster will have a small number of targets. As a consequence, the features extracted from each cluster will be sensitive to outliers.

The distribution of the testing set is shown by Table 4.3.

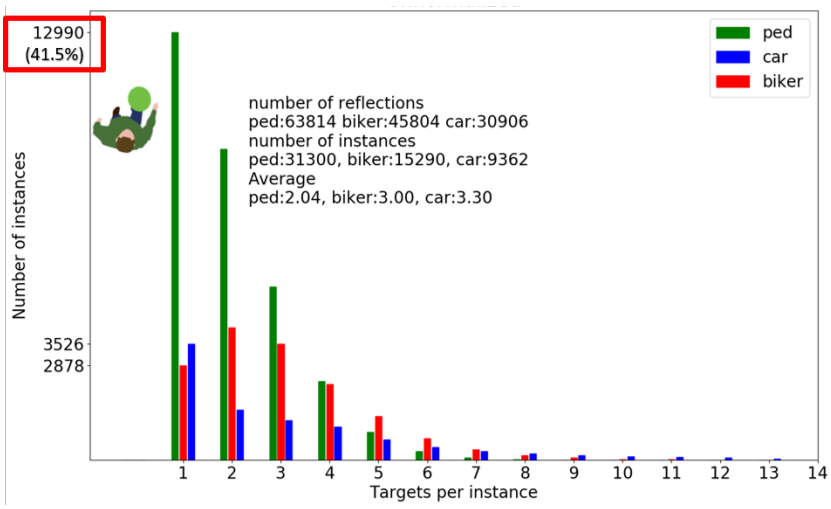


Figure 4.6: The distribution of the reflections in the training set. It can be seen from the statistics, there are many pedestrians that only have one target. Those pedestrians cannot be detected by clustering-based method because the minimum number of targets per cluster cannot be lower than 2 if DBSCAN is used. From the overall statistical distribution, there are few road users having more than 6 targets.

Classes	Others	Pedestrians	Bikers	Cars
Instances		31300	15290	9362
Targets	394606	63814	45804	30906
Targets per instance		2.04	3.00	3.30

Table 4.2: The distribution of the training set.

Classes	Others	Pedestrians	Bikers	Cars
Targets	199609	21983	43459	44214

Table 4.3: The distribution of the testing set

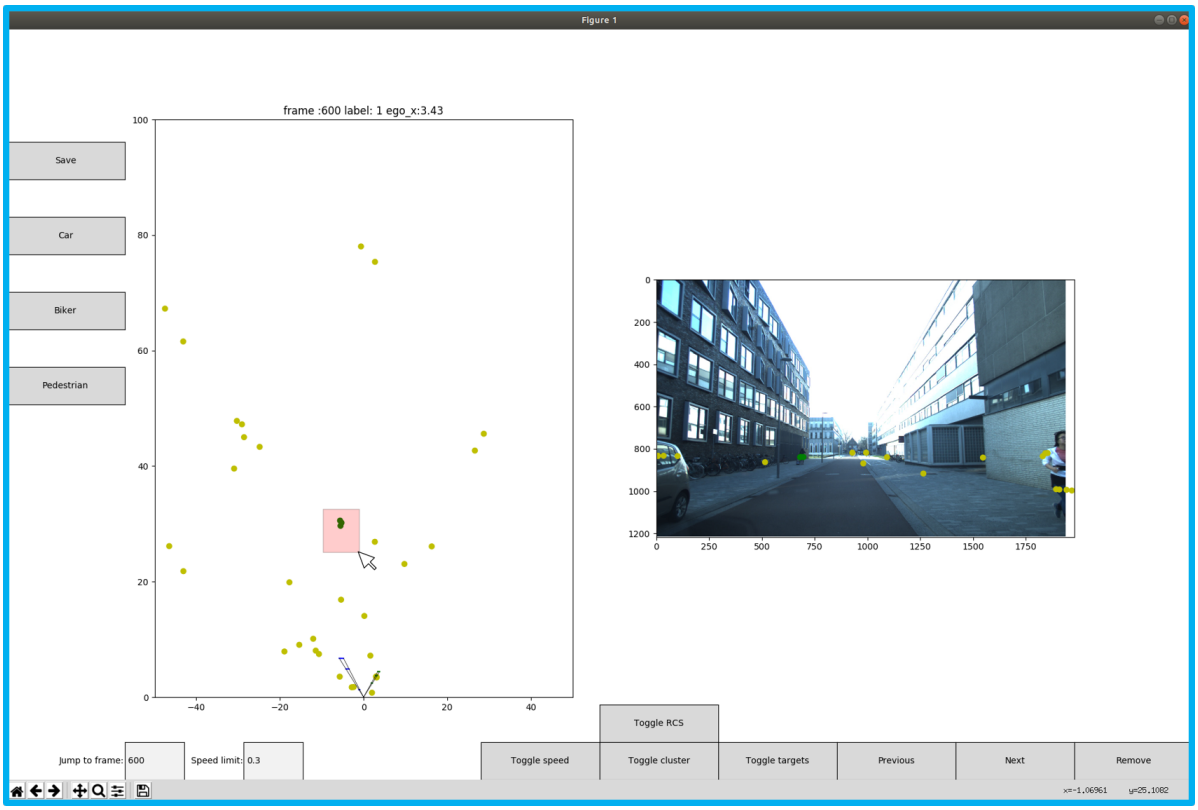


Figure 4.7: The GUI tool for radar targets annotation tool. There are two separate windows in the interface. The window on the left is used for showing the targets from top-down view and creating a bounding box by dragging the mouse. The SSD frustum bounding boxes are plotted as a guideline for the user. The targets are projected to the camera image in the window on the right. After the targets are labeled, they will be given a color. Green is for pedestrians. Red is for cyclists and blue is for cars. The tool also allows users to jump to a specific frame and set a threshold to the speed. Other functions are listed in Table C.1 in appendix C.

4.2. Baselines

To evaluate the proposed method, three methods are set as baselines. The Schumann method is from the study in [71] and the Prophet method is from the study in [45]. These two methods are the only published real-time, multi-object, multi-class detectors using high-level data at the time of writing. The last baseline is to prove the validity of using the low-level data and LLTnet. It only uses high-level features with the same fully-connected layers used by LLTnet. During the experiment, these baselines are trained and tested by the dataset described in section 4.1. In addition to the baselines, some variants are also tested, e.g. using high-level features without speed.

4.2.1. Schumann method

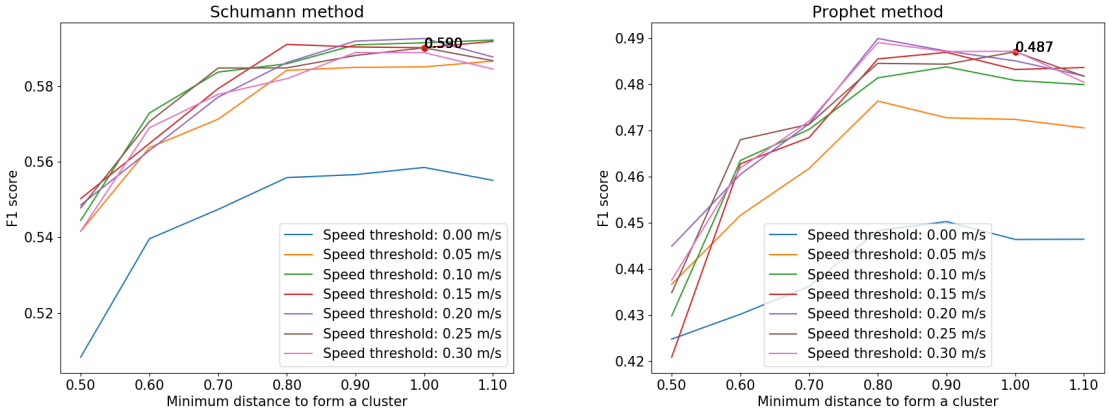
The Schumann method is a typical clustering based pipeline. Its details are described in [71] and section 2.2. In the original study, there are four radars mounted on the test vehicle and targets from two radar frames are aggregated. In the current study, in order to compare its performance with our method under the same conditions, one radar frame is used. In the original study, the accurate parameter setting for DBSCAN and its metric are not given. The DBSCAN set-up in [61] is taken as a reference. The minimum number of targets per cluster N_{min} can not be 1, otherwise some features are not valid (e.g. the covariance of the

2D coordinates). Considering the number of targets from a pedestrian in Figure 4.6, it is better not to set a number higher than 2. Therefore, 2 is used in this study. Using the parameter setting in [61], the minimum distance for a core point to look for its neighboring points eps is set as 1. The other parameter setting is not from DBSCAN, but from the static objects removal stage. In the original set-up, the static objects are not removed. However, the clustering stage suffers from the different density in static targets and moving targets. The classification of the clusters are also biased to the targets from class 'others' due to the ratio between the static targets and moving targets. Due to these two reasons and in accordance with the proposed method, a threshold is set to remove static targets. The threshold for the speed of targets is set as 0.3 m/s in consistency with the proposed method. An optimal search is performed on the speed threshold and eps. The eps is searched from 0.5 to 1.1 with a step size of 0.1. The speed threshold is searched from 0.05 m/s to 0.3 m/s by a step size of 0.05 m/s. The result of the optimal search is shown in Figure 4.8a. Based on the result, the selected value is a reasonable set-up and a good trade-off between accuracy and efficiency. In the original study, the clusters are manually corrected. In the presented baseline, no manually correction is applied.

After the clustering stage, 34 features are generated, which are the same with features in [71]. 18 features come directly from the statistics of the high-level features. The others are from the histogram of the RCS and velocity. After feature extraction, in the original study, two classifiers are compared. One is random forest and the other is LSTM recurrent neural networks. For LSTM, 8 frames are used to form a sequence, which does not meet the requirement for real-time processing. In the current study, only random forest is used. The parameters for the random forest is listed in Table 4.4.

Number of trees	Criterion	Min_samples_split	Min_samples_leaf
10	Gini impurity	2	1

Table 4.4: The parameter setting for the random forest classifier. Min_samples_split means the minimum number of samples required to split an internal node. Min_samples_leaf means The minimum number of samples required to be at a leaf node.



(a) Schumann method (b) Prophet

Figure 4.8: The F1 score of the baselines with different parameter settings. The horizontal axis represents for the minimum distance to form a cluster by DBSCAN, i.e. eps. Different plots are from different speed threshold used for filtering the targets. The red dot is the selected parameter setting.

4.2.2. Prophet method

Prophet method proposed by [46] is also a clustering based method using DBSCAN. Different from the original study where the data is recorded by a static radar with one road user at a time, the current study is done by a moving ego-vehicle and most frames have multiple road

users. In the original study, the parameter setting for DBSCAN is following the study in [61]. The metrics is Euclidean distance in speed and space. The eps is set as 1. The N_{min} is set as 2. In the current study, similarly with Schumann method in subsection 4.2.1, the minimum distance in the feature space to look for neighboring targets and the speed threshold to remove static targets are optimized by a naive search. The result of the optimization is shown in Figure4.8b. The optimized result is similar with the result by using 0.3 m/s as threshold and 1 as eps. In order to make the set-up consistent to the proposed method and its original study, 0.3 m/s and 1 are used.

After the clustering stage, 11 out of 13 features proposed in the original study are used for classification. The velocity resolution is removed, because the feature in the original study is used for denoting which of their two radars are used. In the current study, this feature is useless since only one radar is used. The other removed feature is the existence of a static target. The static targets are removed by setting speed threshold, this flag is always 0. Removing it does not influence the final result.

Many classifiers are used by the original study, which are mainly from three classes. Some of them are from trees based methods, such as decision trees and ensemble bagged trees. Some of them are from support-vector machine and its variants. The last method is a multi-layer perceptron with one hidden layer. The classifier used in the current study is random forest. There are two reasons. The first reason is to make it consistent with the Schumann method. The second reason is the ensemble bagged tree method yields the best result in the original study. The parameter setting of the random forest classifier is the same with the one used in subsection 4.2.1.

4.2.3. High-level multi-layer perceptron

To compare the performance improvement made by adding low-level features, a multi-layer preceptron (MLP) for high-level data is trained and tested. The network structure of the MLP is the same as the fully-connected layers used by LLTnet. The input high-level data is normalized in the same way as mentioned in the subsection 3.1.5. By using the MLP directly for classification, the modules for low-level features in LLTnet are bypassed.

4.3. Training

4.3.1. Training set-up

The training pipeline, testing pipeline and the structure of the LLTnet are implemented by PyTorch framework [41]. Two test drives are recorded for the study. The first recording is used for training. From the second recording, 20% frames are randomly chosen for validation. The rest of the frames from the second recording are used for testing. Note that the cross-validation is not applicable in this case, because the frame rate of sensors are much higher than the frequency of road users' movement. Therefore the consecutive frames in the recording are highly correlated to each other. In other words, each road user moves very small distance with very small speed change between two consecutive frames. Therefore, If some frames for validation or for testing are sampled from the first recording for a cross-validation, the data used for testing and validation will be almost the same with the training set, which causes a high risk of overfitting.

The hyper-parameters for training and testing is listed in Table 4.5. The validation loss is used for early stopping. After each epoch, the loss on the validation set is evaluated and a checkpoint is only updated if the validation loss is smaller than the previous smallest validation loss.

To further test the importance of different factors, not only the baselines and the proposed LLTnet with ensemble learning but also many of their variants are trained and evaluated. The different set-ups are listed in Table 4.6. The baselines mentioned in section 4.2 are Schumann, Prophet and LLTnet-FCN. The proposed method is LLTnet-ensemble.

The features used by different methods are listed in column 'Features'. The high-level features means $(r, \theta, \hat{v_r}, \sigma)$ from the radar target list, except for Schumann method and Prophet method, which use the extracted high-level features from each cluster. The classifier for target segmentation used by each method is listed in the column 'Classifier'. The column

Starting learning rate	0.001
N_{epoch}	10
Batch size	1024
Optimizer	Adam
$N_{Doppler}$	32
N_{angle}	5
N_{range}	5

Table 4.5: The hyper-parameters for training

'Ensemble' means whether the ensemble learning policy for multi-class classification is used. The column 'speed' means whether the compensated speed $\hat{v_r}$ is included in the high-level data. The column 'Window' is only applicable to methods that use LLT, which is the size for cropping low-level features.

Method	Features	Classifier	Ensemble	Speed	Window
Schumann	High-level	Random Forest	No	Yes	None
Prophet	High-level	Random Forest	No	Yes	None
HL-RF	High-level	Random Forest	No	Yes	None
LLT-RF1	Low-level	Random Forest	No	Yes	1x1
LLT-RF5	Low-level	Random Forest	No	Yes	5x5
LLTnet-FCN-multiclass	High-level	FCN of LLTnet	No	Yes	None
LLTnet-FCN-multiclass*	High-level	FCN of LLTnet	No	No	None
LLTnet-FCN-ensemble	High-level	FCN of LLTnet	Yes	Yes	None
LLTnet-FCN-ensemble*	High-level	FCN of LLTnet	Yes	No	None
LLTnet-multiclass	Low-level	LLTnet	No	Yes	5x5
LLTnet-multiclass*	Low-level	LLTnet	No	No	5x5
LLTnet-ensemble	Low-level	LLTnet	Yes	Yes	5x5
LLTnet-ensemble*	Low-level	LLTnet	Yes	No	5x5

Table 4.6: The detailed information of each model in the experiments. The features used by different methods are listed in column 'Features'. The high-level features means $(r, \theta, \hat{v_r}, \sigma)$ from the radar target list, except for Schumann method and Prophet method, which use the extracted high-level features from each cluster. The classifier for target segmentation used by each method is listed in the column 'Classifier'. The column 'Ensemble' means whether the ensemble learning policy for multi-class classification is used. The column 'speed' means whether the compensated speed $\hat{v_r}$ is included in high-level data. The column 'Window' is only applicable to methods that use LLT, which is the size for cropping low-level features.

4.3.2. Data augmentation

The radar signals from different road users are not influenced by mirroring the objects to the other side. Therefore, the training data are mirrored by adding a minus sign before its azimuth angle (i.e. from $(r,\ \theta,\ \hat{v_r},\ \sigma)$ to $(r,\ -\theta,\ \hat{v_r},\ \sigma)$) . A Gaussian random noise with zero mean and 0.05 standard deviation is added to the normalized range and normalized speed of high-level features.

	Pedestrian	Cyclist	Car
N_{min}	1	2	3
eps	1	2	5

Table 4.7: The class-wise parameter setting for DBSCAN method in post-clustering stage. N_{min} is the minimum number of targets to form a cluster. eps is the minimum distance for a core point to look for its neighboring points in DBSCAN.

4.4. Post-clustering set-up

The post-clustering stage uses DBSCAN clustering method. The minimum number of targets N_{min} around each core point, the minimum distance within which a point searches its neighbors eps for each class are listed in Table 4.7.

4.5. Results

In this section, the methods are evaluated quantitatively and qualitatively. For quantitative evaluation, firstly the loss and processing time are measured. Then the target segmentation results and road user detection results are measured separately. The metrics used for target segmentation evaluation are F1 score and confusion matrix. The metrics used for road user detection evaluation are the precision, recall, F1 score and intersection over union (IoU). They are calculated on object-level which considers the localization accuracy and the classification accuracy. For qualitative evaluation, a real-time program for demonstration is implemented, which are shown by figures in the thesis.

4.5.1. Quantitative evaluation

Loss trajectory

The validation loss trajectory of each model in LLTnet-ensemble are shown in Figure 4.9.

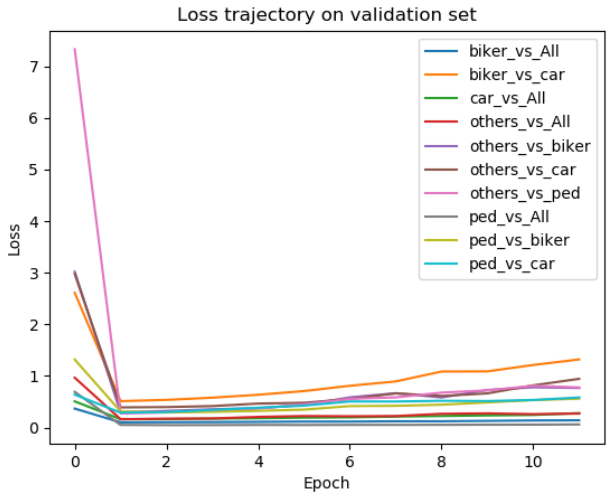


Figure 4.9: The validation loss trajectory of the models used for the ensemble learning. After each epoch, the loss on the validation set is evaluated and a checkpoint is only updated if the validation loss is smaller than the previous smallest validation loss.

The processing time

The processing time with different speed threshold for the targets are shown by Table 4.8.

Target-level result

Precision and recall are two metrics that are usually used for the evaluation of a classifier or target segmentation model. Precision means how many samples are predicted correctly

Speed threshold (m/s)	Pre-processing (s)	Inference (s)	Total (s)	Frame rate
0	0.08	0.10	0.18	5.49
0.1	0.06	0.05	0.11	8.93
0.2	0.06	0.04	0.10	9.80
0.3	0.06	0.04	0.10	9.80

Table 4.8: The processing time of each stage in the proposed pipeline. The first columns represents the different speed thresholds for the targets. The last column is the frame rate of the pipeline. The columns in between are the preprocessing time, the inference time and the total time for a single prediction.

among all the positive predictions, which is calculated by Equation 4.3. Recall means how many samples are correctly predicted from all the positive ground truth, calculated by Equation 4.4. Here TP means 'True Positives'. FP means 'False Positives'. FN means 'False Negatives'. These two metrics measure the accuracy in two different aspects. In other words, neither a good precision nor a good recall is adequate for a good classification. For example, if all the samples in the presented study are predicted as 'others', the recall of 'others' class will be 1. However, the precision will decrease at the same time. Therefore, to consider both precision and recall, the harmonic average of precision and recall is used in the current study, which is also called F1 score. The calculation of F1 score is defined by Equation 4.5. In a multi-class classification task, the macro-average F1 score is the unweighted mean of the F1 score calculated for each single class. In the presented study, the class-wise F1 score and macro-average F1 score of the baselines, the proposed LLTnet-ensemble and their variants are listed in Table 4.9.

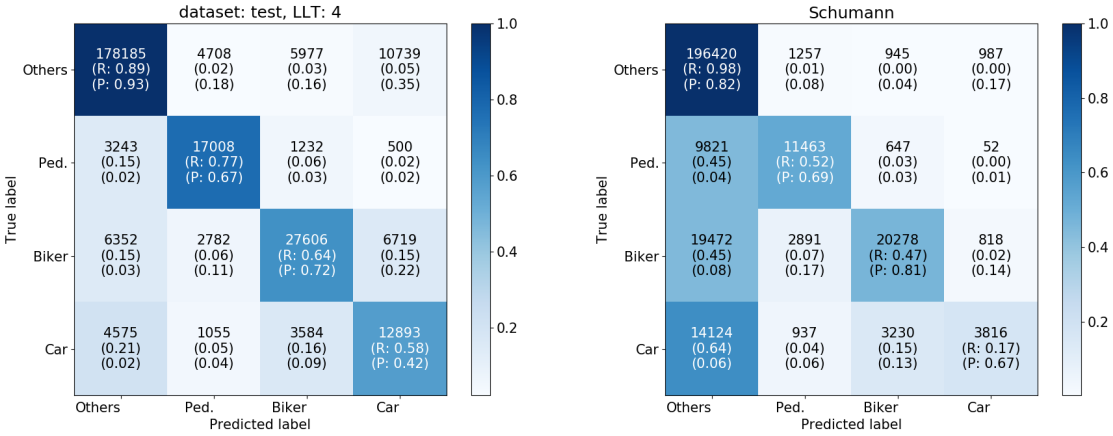
$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4.3}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4.4}$$

$$F_1 = \frac{Precision \times Recall}{Precision + Recall} \tag{4.5}$$

Method	Pedestrian	Cyclist	Car	Others	Macro-Average
Schumann	0.60	0.59	0.28	0.89	0.59
Prophet	0.48	0.43	0.16	0.88	0.49
HL-RF	0.49	0.51	0.26	0.89	0.53
LLT-RF1	0.53	0.52	0.26	0.89	0.55
LLT-RF5	0.49	0.35	0.16	0.87	0.47
LLTnet-FCN-multiclass	0.54	0.63	0.29	0.83	0.57
LLTnet-FCN-multiclass*	0.79	0.34	0.33	0.23	0.42
LLTnet-FCN-ensemble	0.55	0.64	0.35	0.89	0.61
LLTnet-FCN-ensemble*	0.35	0.33	0.16	0.87	0.43
LLTnet-multiclass	0.67	0.64	0.41	0.85	0.64
LLTnet-multiclass*	0.63	0.62	0.33	0.82	0.60
LLTnet-ensemble	**0.72**	**0.67**	**0.49**	**0.91**	**0.70**
LLTnet-ensemble*	0.66	0.63	0.37	0.89	0.64

Table 4.9: The F1 score of the baselines, the proposed LLTnet-ensemble and their variants. The set-up of each method are listed in Table 4.6.



(a) The confusion matrix of LLTnet-ensemble model (b) The confusion matrix of Schumann method

Figure 4.10: The confusion matrix of the proposed method and Schumann method. The first value in the bracket are the value normalized by the summation of numbers in each row. The second value in the bracket are the normalized value by the summation of numbers in each column. Therefore, on the diagonal line, 'R' means recall and 'P' means precision.

The confusion matrix is also a widely used measurement for radar target segmentation task. The confusion matrix of LLTnet-ensemble method is shown in Figure 4.10a in comparison with Schumann method in Figure 4.10b. In the confusion matrix used by the current study, each row of the matrix represents the true label of each target while each column represents the predicted labels. These entries are the number of targets. The values on the diagonal are true positive predictions. By using confusion matrix, the specific misclassification error between two classes is revealed. For example, the value 1232 in the second row of Figure 4.10a means there are 1232 targets from pedestrians predicted as bikers. The value in the first bracket is normalized per each row and the value in the second bracket is normalized per each column. In diagonal entries, 'R' stands for recall and 'P' stands for precision. The confusion matrices of the other methods are listed in Appendix A.

Object-level result

A road user detection algorithm has two aspects; the localization and the classification. In the proposed radar based road user detection algorithm, the metrics used by visual object detection algorithms that consider localization accuracy and classification accuracy are adapted for radar targets prediction.

The output of the post-clustering stage are a set of targets with class label and object ID, see the 'prediction' in Figure 4.11. Each predicted object is compared with the ground truth object. The intersection of two objects are the common targets of the predicted object and the ground truth object. The union are all the targets covered by the ground truth and the predicted object. The intersection over union (IoU) is calculated to evaluate the localization accuracy. If the IoU is equal to or higher than 0.5, it is counted as a true positive prediction. The precision and recall are hence calculated by Equation 4.6 and Equation 4.7. If the precision and recall are known, the F1 score can be calculated in the same way of target-level F1 score. The overall localization accuracy is evaluated by the mean IoU of each class. The results are shown by Table 4.10.

$$precision = \frac{TP}{Number\ of\ detections} \tag{4.6}$$

$$recall = \frac{TP}{Number\ of\ ground\ truth} \tag{4.7}$$

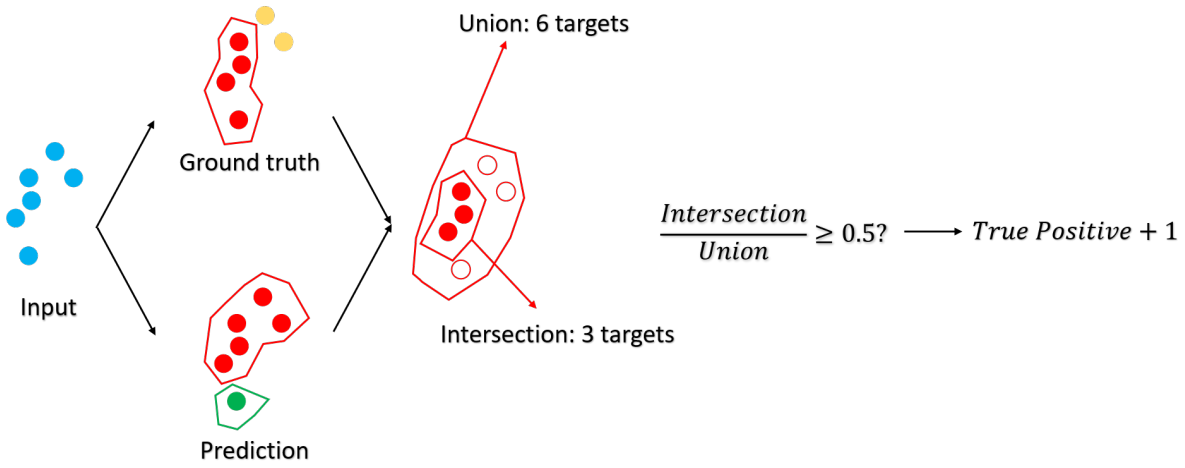


Figure 4.11: The calculation of a true positive prediction based on the intersection over union (IoU). The intersection is calculated by the common targets of the predicted object and the ground truth object. The union is calculated by all the targets covered by ground truth and the prediction. If IoU is larger than 0.5, it is counted as a true positive prediction.

Method	Ped.	Biker	Car
Schumann	**0.67**	**0.68**	0.36
Prophet	0.52	0.62	0.32
LLTnet-ensemble	0.51	0.66	**0.53**

(a) Precision

Method	Ped.	Biker	Car
Schumann	0.37	0.42	0.15
Prophet	0.30	0.27	0.06
LLTnet-ensemble	**0.68**	**0.51**	**0.42**

(b) Recall

Method	Ped.	Biker	Car
Schumann	0.48	0.52	0.21
Prophet	0.38	0.38	0.10
LLTnet-ensemble	**0.58**	**0.58**	**0.47**

(c) F1 score

Method	Ped.	Biker	Car
Schumann	0.42	0.42	0.16
Prophet	0.31	0.27	0.09
LLTnet-ensemble	**0.54**	**0.50**	**0.38**

(d) Mean intersection over union (mIoU)

Table 4.10: The object-level results on road user detection.

4.5.2. Qualitative evaluation

The proposed pipeline is implemented by a node in Robot Operating System (ROS) for qualitative evaluation. Under an environment that simulates the in-vehicle experiment, this ROS node subscribes to the radar high-level data, low-level data and the ego-motion of the vehicle. Then it uses the trained LLTnet-ensemble model to make prediction. At the same time, a point-cloud is created by the stereo camera to visualize the scene in front of the vehicle. The results of the target segmentation step are projected to 3D space for visualization. The height of targets is the height of the mounting point of radar. Different colors are used to show targets predicted as different road users, see Figure 4.12. After target segmentation stage, the road users by post-clustering are visualized by thin cubes using the same color map, see Figure 4.13. The position of the cube is the mean value of x and y coordinates of targets from each road user. In addition, this ROS node also supports micro-Doppler signature visualization. The micro-Doppler signature at the position of each target is visualized by drawing several 3D cubes stacked perpendicularly to the horizontal plane, see Figure 4.14. Each 3D cube represents the Doppler responses at that position at a certain speed bin. The block at the height of radar represents the speed bin corresponding to the measured speed v_r of that target. Other than the ROS node for on-board processing, the targets are also projected to the 2D camera image with different colors for offline qualitative evaluation, see Figure 4.15.

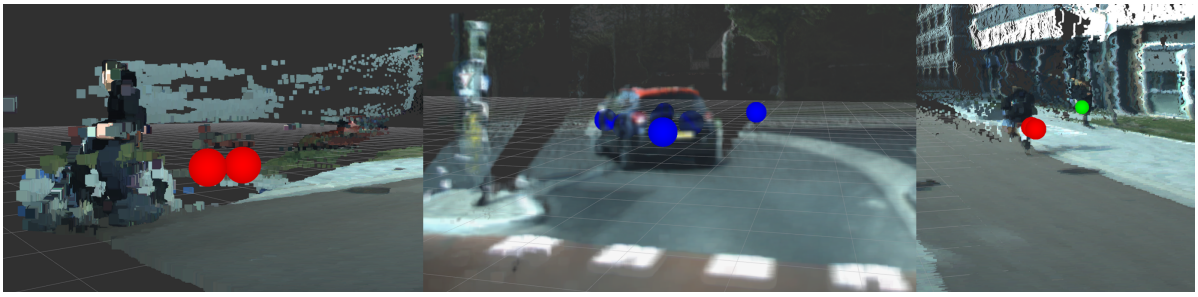


Figure 4.12: The targets that are predicted as road users are projected to 3D space. Green is used for pedestrian. Red is used for cyclist and blue is used for cars.

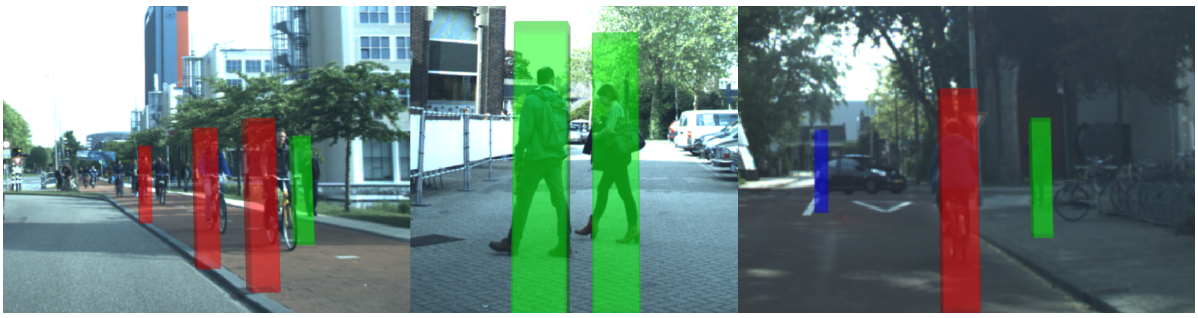


Figure 4.13: The road user detection visualized by cubes in different colors. Green denotes the pedestrian. Red is used for cyclist. Blue is for cars.

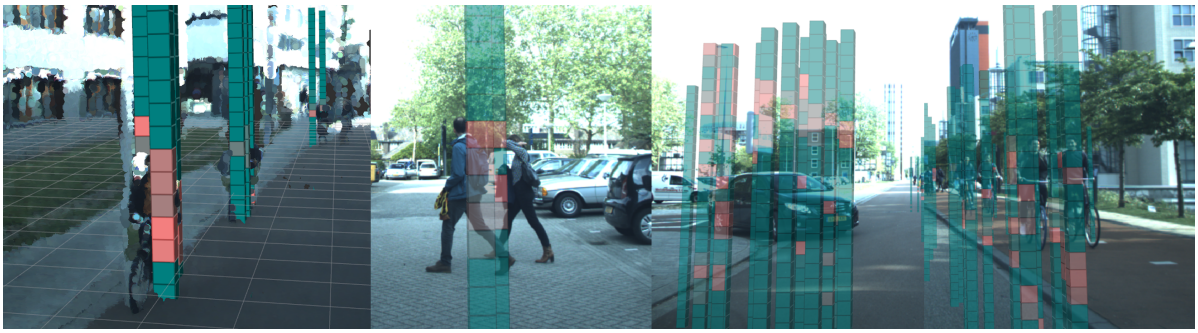


Figure 4.14: The Doppler signature visualization of different road users. Each 3D cube represents the Doppler responses at that position at a certain speed bin. The block on the ground plane represents the speed bin corresponding to the measured speed of that target.

Figure 4.15: Targets predicted as road users are projected to 2D image. Green is used for pedestrian. Red is used for cyclist and blue is used for cars.

4.6. Discussion

In this section, the results presented in section 4.5 are discussed from different aspects.

4.6.1. Clustering based methods

In general, clustering based pipeline obtains satisfactory result if the clustering step groups targets properly. Schumann method re-implemented in the current study obtained an F1 score of 0.59. As a reference, in the study [62] by the same authors of [71], the DBSCAN+LSTM method achieves an F1 score of 0.597.

For Prophet method, although it uses the same parameter setting, the performance is worse than Schumann method. The only difference between them is the different features of each cluster. Features used by Prophet are only calculated directly from the statistical distribution. In contrast, features used by Schumann method are also calculated from the normalized histograms of speed and RCS values. It is shown by the result that only using statistical features is not reliable in a fast changing environment, e.g. environment with moving vehicle and moving road users.

To isolate the influence of the clustering stage, a confusion matrix only for the clustering stage is shown by Figure 4.16. The targets that belong to actual road users are annotated as 'should be clustered' and targets that are given a cluster label by DBSCAN are predicted as 'clustered'. Only about 70% targets are correctly clustered. There are more than 18000 targets belonging to the actual road users are not clustered. Since the minimum number of targets to form a cluster is set as 2 in this case, the misclusteredd targets are mainly from the road users that only have one single target detected by radar.

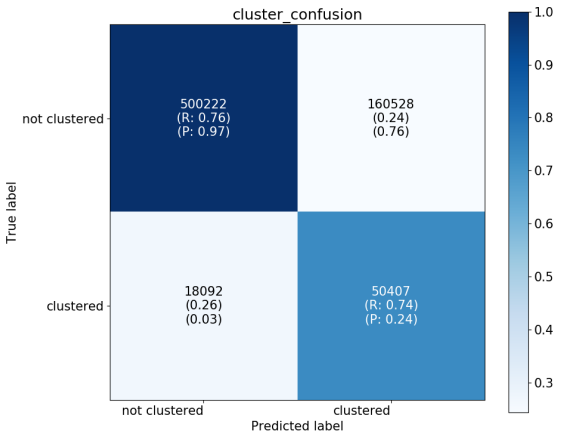


Figure 4.16: The confusion matrix of the clustering stage. This matrix only calculates the clustering accuracy by DBSCAN.

4.6.2. Improvement brought by low-level data and LLTnet

If other conditions in Table 4.6 are kept as the same (classifiers, whether to add ensemble learning, whether to use speed, the window size) but only the features are different, the importance of adding low-level data can be verified by the result in Table 4.9. For example, LLTnet-FCN-ensemble uses the same fully-connected layers of LLTnet-ensemble model and the same policy for ensemble learning. However, the former has a macro-average F1 score of 0.60 while the LLTnet-ensemble has 0.70. The other example is LLT-RF1 and HL-RF. Even only using the random forest classifier, adding low-level data still improves the performance. If the ensemble learning method is not used, the improvement from LLTnet-FCN-multiclass (0.567) to LLTnet-multiclass (0.640) also proves the benefit brought by adding low-level data and LLTnet.

4.6.3. Improvement brought by ensemble learning

In this task, the positive samples and negative samples are highly imbalanced. In addition, the difference between different road users is less obvious than the difference between road users and background. Therefore, using binary models on every possible combination of two classes with an effective voting policy can improve the inter-class classification performance. This is proved by the results in Table 4.9. For example, in Table 4.9 the method LLTnet-multiclass uses the same network structure and the same input as LLTnet-ensemble. However, by only adding ensemble learning stage, the macro-average F1 score improves from 0.64 to 0.70. The ensemble learning experiments on high-level data makes the argument stronger, because adding ensemble stage to a fully-connected network also improves the macro-average F1 score from 0.57 of LLTnet-FCN-multiclass to 0.60 of LLTnet-FCN-ensemble.

4.6.4. Importance of the high-level speed

Speed is an important feature in both high-level data and low-level data. If only high-level data is used, after the high-level speed (the speed of the target with respect to the ground) is removed, the performance greatly drops. The decreased performance shows the important role played by the high-level speed in high-level data based method. To test the importance of the high-level speed in the proposed pipeline, the LLTnet is also trained by setting high-level speed as zeros (LLTnet-multiclass* and LLTnet-ensemble*). In other words, the network does not 'see' the high-level speed. Without the help of high-level speed, the performance also drops, which is expected. However, the performance is still better than the state-of-the-art clustering based method. This experiment proves LLTnet is not trapped by a local minimum, i.e. using high-level speed as the only criterion.

4.6.5. Generalization ability of the method

During the experiment, after the model is trained and tested by the automatically annotated dataset, it gives higher false positives from background than expected. To look into the problems, the frames which have false positive predictions are visualized by high-level data scatter plot and camera image. After the visualization, it turns out that many false positive predictions are from some actual road users not annotated by the automated annotation pipeline. In other words, the false positives are not really false positives, but are from the misclassification of SSD node. This phenomenon means the LLTnet-ensemble is not only superior than the high-level data methods, but also has good generalization ability. Furthermore, it also proves radar can be a supplementary sensor for other vision based sensors. It is able to learn useful information against reasonable noise. Due to this reason, the testing set is re-annotated manually. For better evaluation, the results shown in Table 4.9 are evaluated by this manually corrected testing set. About 6000 frames are manually corrected. Only by repairing the testing set and the model is trained by the training set without correction, the macro-average F1 score is greatly improved by 0.04.

4.6.6. The result of the post-clustering

The post-clustering results are evaluated on the dataset without manual correction mentioned in subsection 4.6.5, since that correction is only performed on target level. Therefore, the measured object-level precision is lower than the actual precision. With the imperfect ground truth in testing set, the object-level results of the proposed pipeline are still better than the baseline by a large margin.

4.6.7. Error analysis

The error analysis is mainly done on target segmentation results. There are two reasons. The first reason is that the target segmentation is the direct output of the LLTnet-ensemble, which is the main contribution of the proposed method. The second reason is that the result of target segmentation is evaluated by a manually annotated testing set, which is more accurate.

From the confusion matrix of LLTnet-ensemble in Figure 4.10a, it is observed that the target-level misclassification are mainly from three sources: car-vs-biker area, false positive

region (the first row) and false negative region (the first column).

car-vs-biker misclassification

The car-vs-biker misclassification is mainly caused by two reasons. The first reason is that there are many bikers moving together in groups, see Figure 4.17. Those bikers are moving at similar speed and having strong reflection due to grouping. Radar only looks at the speed distribution and reflection for classification. Therefore, both high-level and low-level data of these biker groups are similar to the data from cars.

The second reason is from the distribution of targets inside a car. In an urban scenario, there are many cars slowly crossing the road in front of the ego-vehicle. A single radar is only able to measure the radial speed. Even if the speed of the entire car body is similar, their radial components measured by different angle bins are different. Furthermore, when the car is turning, the car body not only has bulk translational speed, but also has rotational speed. The radius in this case is the distance from the measured point to the rotation center. Therefore, the speed distribution around a car also becomes wide, similarly to bikers. Those cars are sometimes predicted as bikers, see Figure 4.18.

False positives from the background

False positives in this context means the targets that are not from any road user but are predicted as road users, i.e. the first row of the confusion matrix. First of all, the testing set is still not perfect. The manual correction for the testing set mentioned in subsection 4.6.5 is only done for vulnerable road users. Therefore, the false positive for cars also include cars that are not detected by SSD. This is verified by exporting the false positive frames shown by Figure 4.19. The subplot on the right is the predicted labels from the top-down view. The subplot on the left is the ground truth projected to 2D camera image. Yellow means the targets from the bus are annotated as 'others'. These targets are predicted as cars by the LLTnet-ensemble correctly, but they are mistakenly counted as false positives.

Secondly, there are some false positives caused by the ghost targets. For example, when the ego-vehicle is close to a building or a parked vehicle, there will be targets 'inside' the vehicle or the building caused by the reflection of the ego-vehicle itself. This is shown by Figure 4.20. In this case, the parked vehicle in front of the ego-vehicle is like a 'mirror' in the world seen by radar, which causes the problems similar to the problems caused by a mirror to vision based road user detection algorithms.

The third reason for the false positive prediction is the noise from static objects. When the vehicle is moving, the measured speed is compensated by the ego-motion. The ego-motion is measured by odometry, which is not perfect. Using a small threshold for speed filtering (0.3 m/s) will keep some targets of objects that are not actually moving but having a measured speed due to imperfect ego-motion estimation. The example of this error is shown in Figure 4.21.

False negative detection

The false negatives are the targets that should be predicted as road users but predicted as targets from 'others'. There are mainly two reasons.

First of all, in the training data, there are many road users not annotated correctly by SSD due to the lighting conditions, field of view of the camera, capacity of the SSD model, etc. Therefore, the training data that are annotated as 'others' also have some targets from actual road user which will mislead the model.

Secondly, although the weights for the loss function and ensemble learning scheme are used to address the problem caused by imbalanced dataset, due to the fact that the number of targets from 'others' is 10 times more than targets from road users, the model is still biased by the targets from 'others'. Further fine-tuning of the weights will help improve the performance, but leaving the weights defined only by the number of training data is a more robust solution, otherwise every time a new model is trained with different ratio of targets from different classes, the weights should be tuned again.

The error caused by the post-clustering stage

The object-level road user detection error not only comes from the target segmentation step but also comes from the post-clustering stage. The post-clustering stage uses different hyper-parameter settings for different road users. However, it lacks the ability to consider the instance-level relation between different road users. See Figure 4.22, if there are some targets at a car predicted as bikers, those targets will also be fused into a biker if they met the requirement to form a cluster at the post-clustering stage. It does not filter the results based on relative position of their targets, e.g. there can not be a biker 'inside' a car.

Secondly, the recall of the cyclist is decreased compared to its target segmentation result. The decrease is mainly because of the minimum number to form a cyclist cluster is set as 2. Although cyclists have larger scale than the pedestrian, there are still some cyclists only having a single target detected by radar. Those targets are ignored with this set-up. However, according to the number of cyclists having one target in Figure 4.6, the hypothesis for minimum target equal to 2 is reasonable for a good trade-off between precision and recall.

The other reason that influences the F1 score is the IoU threshold to assign a true positive detection. The 50% IoU is a reasonable threshold in visual object detection algorithm, while it remains a question whether this value also holds in radar road user detection task. For example, if there is a pedestrian with three targets detected by radar while only one target is predicted as a pedestrian, whether this prediction can be counted as a true positive? If so, the recall will further increase.

Figure 4.17: The cyclists in group with similar speed are predicted as cars. Green dot means car. Red dot means cyclist.

Figure 4.18: The cars that are moving slowly in lateral direction are predicted as bikers. Red dot means bikers and blue dot means car.

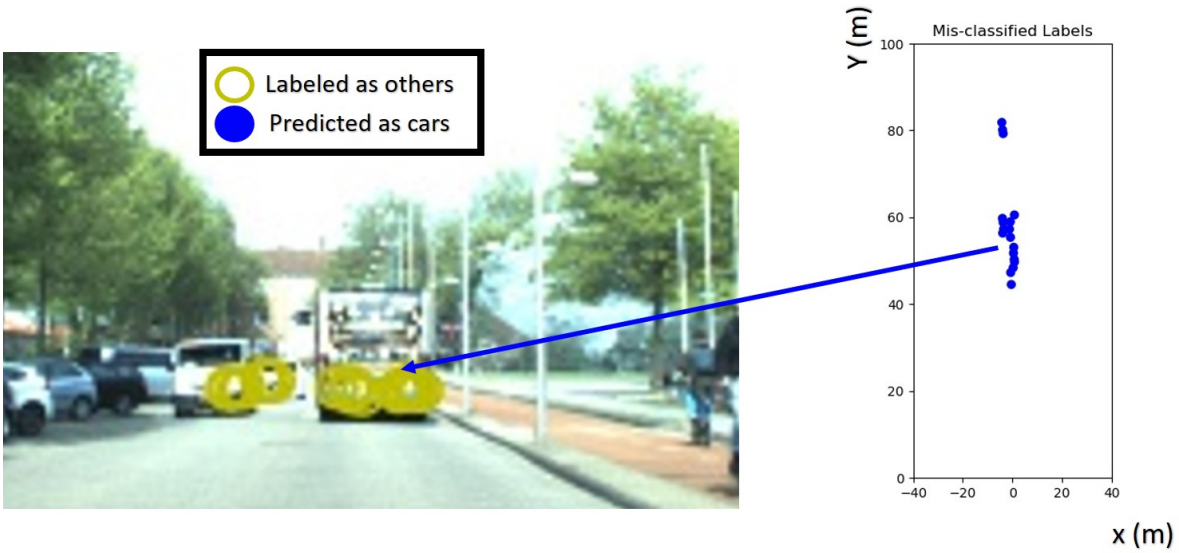


Figure 4.19: The subplot on the right is the predicted targets from top-down view. Blue means they are predicted as cars. The annotation is projected in the camera image on the left. Yellow means targets are annotated as 'others'. In the confusion matrix, these targets are counted as false positives. However, it turns out that the ground truth used for evaluation is imperfect and the prediction made by LLTnet-ensemble is correct.

Figure 4.20: The ghost targets caused by the reflection of the ego-vehicle when there is a parked vehicle in front of it. Red dot means biker and blue dot means car.

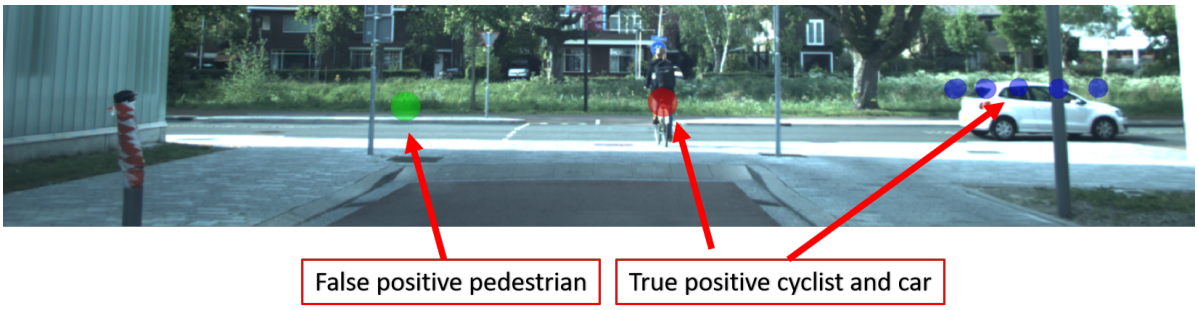


Figure 4.21: The false positive pedestrian caused by the background noise. Green dot means pedestrian. Red dot means biker and blue dot means car.

Figure 4.22: The error caused by the post-clustering stage. If some targets 'inside' a car are predicted as bikers, they will possibly also fused into a biker.

5 Conclusion

5.1. Conclusion

In this thesis, a new pipeline is proposed to perform low-level radar data based road user detection by effectively combining high-level radar data and low-level radar data. Road user detection task has two aspects: the localization and classification. High-level data is used as region proposal, which has better accuracy for localization and less computational demand. Low-level data is used as additional features for accurate classification of road users from these proposals. By not only using targets for object-level information and using single target as input, the clustering stage before classification is removed and features are more descriptive. A novel convolutional neural network structure for target segmentation is designed by considering the essence of low-level data. After target segmentation, the targets can be further fused into road users by post-clustering stage. Only using single frame and single radar, the proposed method outperforms the 3 baselines including the state-of-the-art high-level data based algorithm. Post-clustering stage is able to convert the target segmentation result into different road users. The object-level result is also superior than the baselines. The target segmentation result reaches an F1 score of 0.70 and object-level road user detection has achieved a 0.68 recall on pedestrian class. For this task, a dataset of radar low-level data is created by real-life test drive and automated annotation pipeline. The model is not only tested on the dataset offline but also deployed to a ROS node to perform real-time detection for demonstration.

5.2. Recommendation

5.2.1. Improve synchronization

Three input messages should be further synchronized, which are camera image, radar high-level data and radar low-level data. In the experiment, the radar high-level data is used as the reference for other messages. The target list is always using the latest radar cube to generate its low-level counterpart. The closest camera frame is used because the camera image is only useful for annotation purpose. The camera images and radar low-level data come at a frame rate of 10 Hz, while the radar high-level data has a frame rate of 13 Hz. These messages are recorded in their own time line, therefore there will be a time offset between messages of a single annotated frame. This offset in time will result in an offset in space when both the environment and ego-vehicle is moving. For example, in Figure 5.1 the radar targets from a laterally moving cyclist mismatch its closest camera frame.

The synchronization between radar high-level data and low-level data is also important. The high-level data can only be mapped to the latest low-level data. The time difference between these two messages will cause an offset in space between targets and its corresponding low-level data. In other words, most targets are usually closer to the ego-vehicle than the radar cube because it comes later. One way to address this problem is to add some compensation for ego-motion by the time difference.

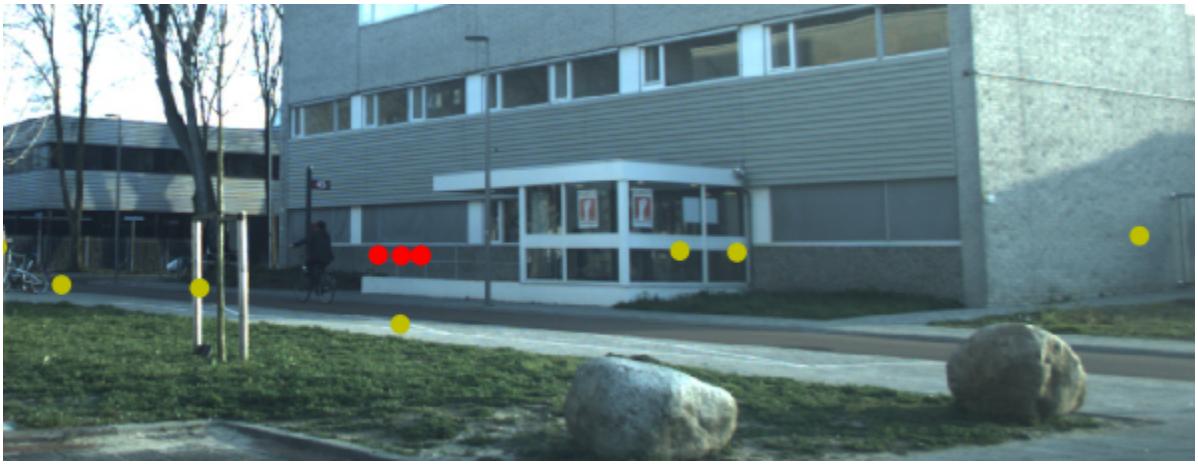


Figure 5.1: The offset between camera image and radar targets. The red targets are from the cyclist moving in lateral direction. The frame rate of camera image is only $10\ Hz$, less than the frame rate of radar high-level data, which is $13\ Hz$. When the biker is moving quickly, there will be a mismatch between camera image and targets.

5.2.2. Improve radar resolution

The range resolution of the radar low-level data in this study is around $0.5\ m$ and the speed resolution is around $0.12\ m/s$. The range resolution is important for the low-level radar data based road user classification. In Figure 5.2 is the comparison of the range-Doppler image cropped around a biker from the radar used in this study and the high resolution radar used in [60]. The image on the left is the biker in this study. Responses from multiple range bins are recovered, but the micro-Doppler signatures for different parts are not clearly visible. In contrast, the micro-Doppler signatures from the pedals and wheels of the cyclist are clearly visible in the image on the right. Therefore, it is expected that using a radar with higher resolution will improve the performance of the proposed method. The proposed method is applicable to different resolutions. If the resolution is higher, more bins can be cropped to form the LLT.

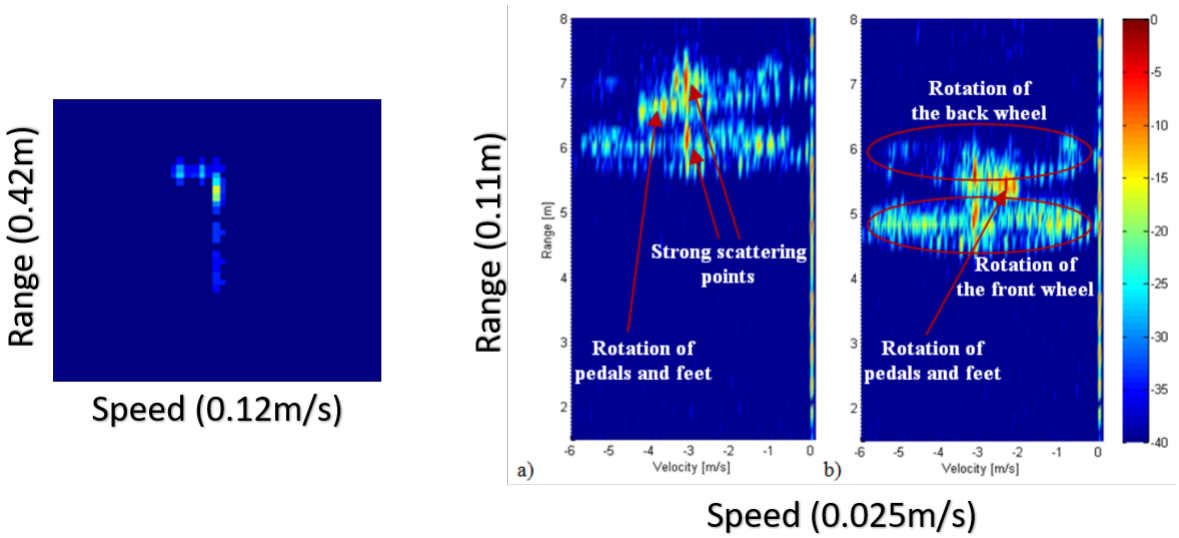


Figure 5.2: The comparison of the cyclist in the low-level data (range-Doppler image) of the radar used in this study and the radar used by [60].

5.2.3. Use more sophisticated scheme to do the post-processing

In the proposed pipeline, the object-level result is obtained by post-clustering stage. The consistency of the labels of targets are not considered when the post-clustering is done separately for each class. Majority voting can be applied to consider consistency. Moreover, there are some more sophisticated algorithms that can also be used in this stage. For example, the target segmentation result is a suitable input for extended object tracking [22], which tracks targets over multiple time steps and also fuses targets into objects. Targets with labels can also be a good input for particle filter. In [40], the particle filter is applied to radar targets for tracking. The other possible solution is to use an additional module in the end of the LLTnet to learn object-level information, such as using a graph convolutional networks, similarly with the one used in [34].

A

Confusion matrices

This is the appendix for the confusion matrices of each method that is tested.

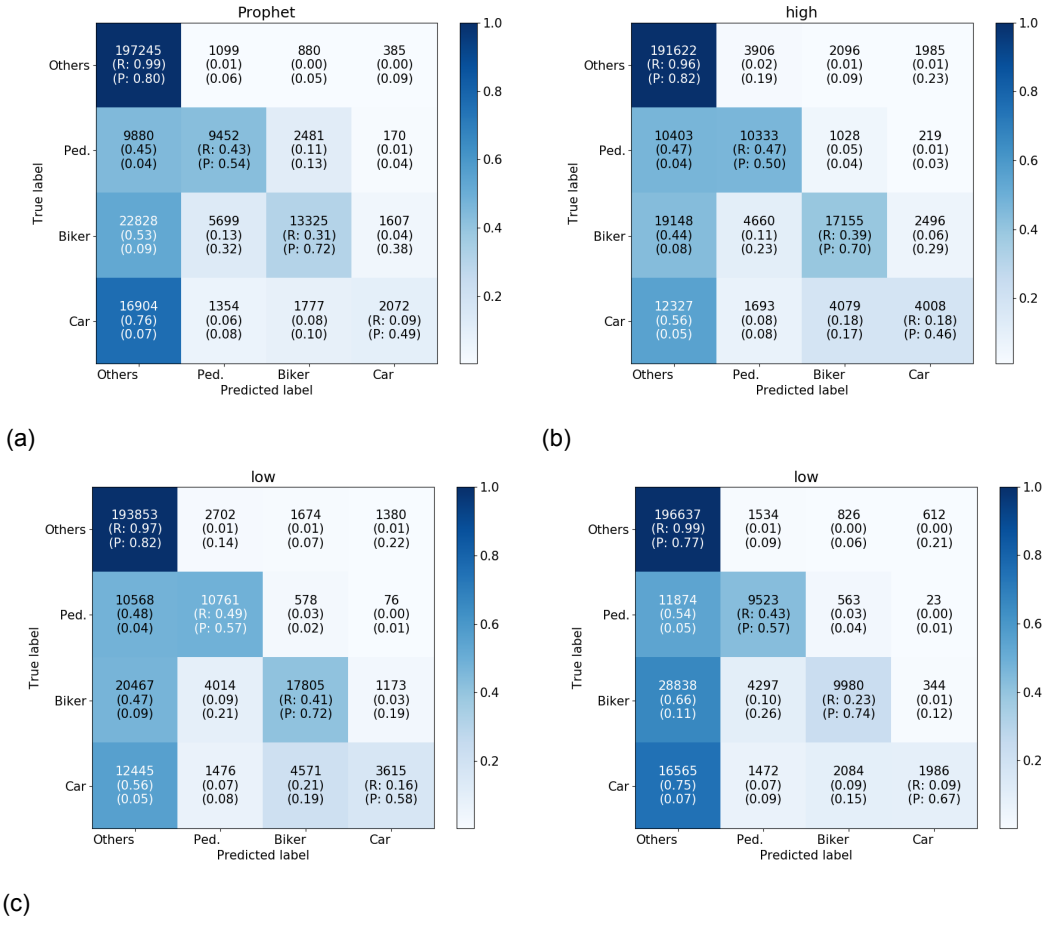


Figure A.1: The confusion matrix of different tests listed in Table 4.6. (a) Prophet (b) HL-RF (c) LLT-RF1 (d) LLT-RF5

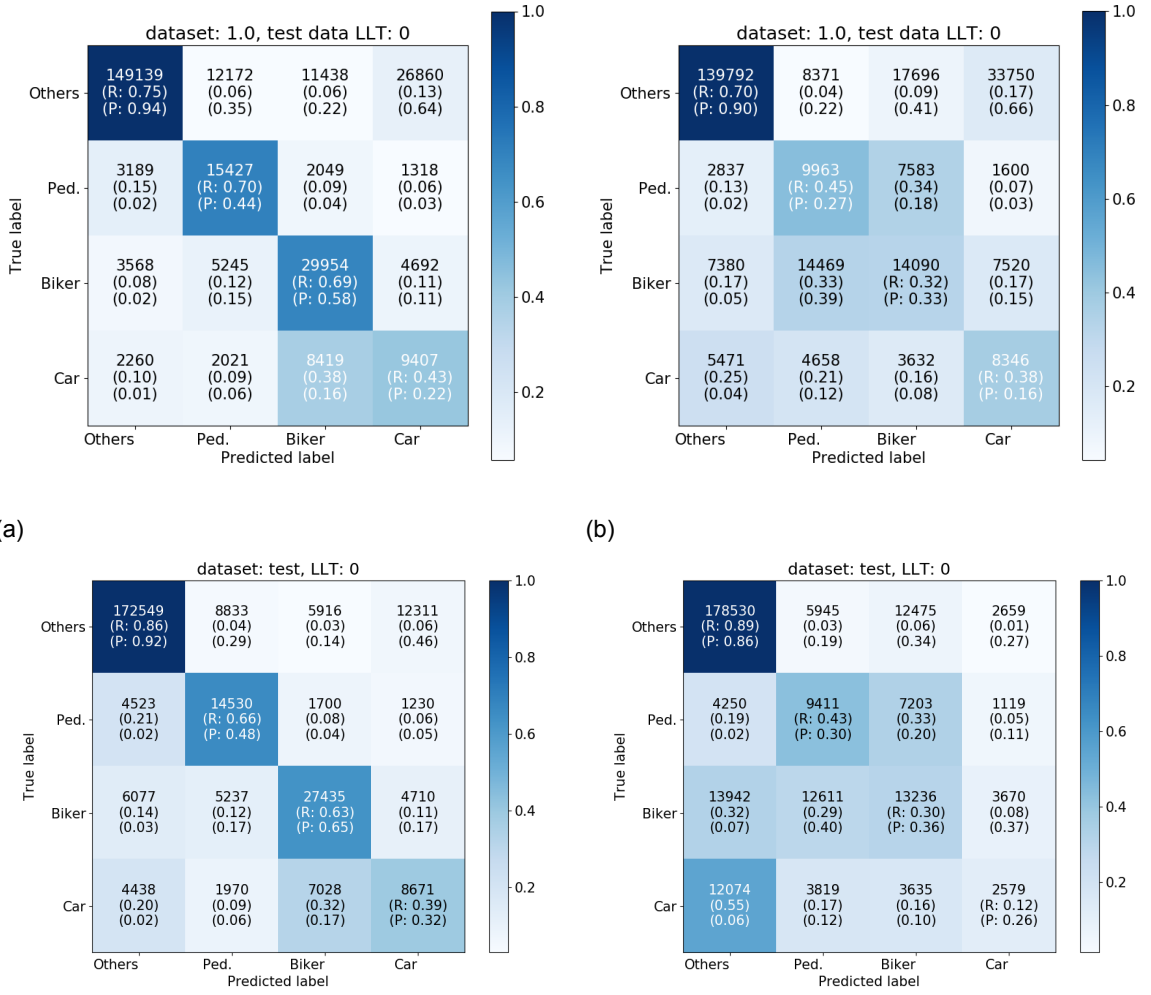


Figure A.2: The confusion matrix of different tests listed in Table 4.6. (a) LLTnet-FCN-multiclass (b) LLTnet-FCN-multiclass* (c) LLTnet-FCN-ensemble (d) LLTnet-FCN-ensemble*

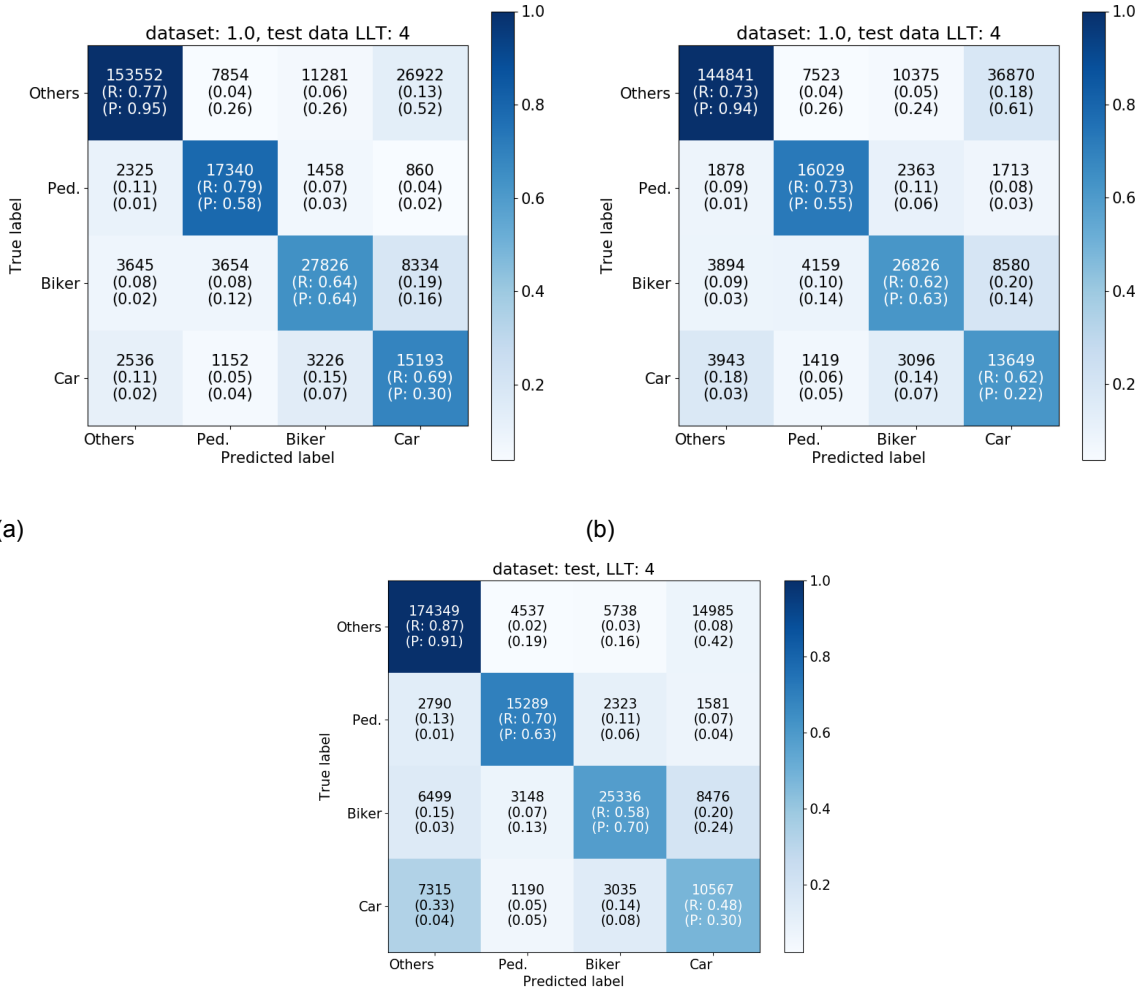


Figure A.3: The confusion matrix of different tests listed in Table 4.6. (a) LLTnet-multiclass (b) LLTnet-multiclass* (c) LLTnet-ensemble*

B

Data format


```
{
    "Doppler_resolution": 0.1147531196475029,
    "anno_2D_tl_br": [
        [
            847.295654296875,
            770.4639282226562,
            1143.8294677734375,
            1022.10546875
        ],
        [
            1384.3057861328125,
            671.7341918945312,
            1575.11767578125,
            1081.8585815429688
        ]
    ],
    "anno_3D_long_latmin_latmax": [
        [
            7.50309148629769,
            0.8193309585507842,
            0.8705795262037058,
            7.471578269037758,
            -0.96991590786273,
            0.8579269802145458
        ],
        [
            3.850627042921245,
            -1.3605099630057207,
            1.132129963566272,
            3.8383509156737303,
            -2.0575197931393605,
            1.127201102660487
        ]
    ],
    "cube": "",
    "ego_angular": 0,
    "ego_linear": [
        0,
        0,
        0
    ],
    "filename_cam": "camera_1553878299372222978.jpg",
    "filename_radar_targets": "radar_targets_1553878299308520797.npy",
    "labels": [
        "vehicle",
        "rider"
    ],
    "lidar": "",
    "time_stamp": 1553878299308520797
```


Figure B.1: An example of the annotation. The annotations are saved as a list of frames in a json file. Each frame has a dictionary of the speed resolution of the radar cube, the 2D and 3D annotations for cars and VRUs, the camera image path, the radar targets path, the labels of each annotation and the ego motion.

C

Table for manual annotation tool

Buttons/Text Boxes	Functionalities
Save	To save the annotaitons of the frames that are finished to a json file. This file can be used as annotations for the processing pipeline or as checkerpoints to continue working next time.
Car/Biker/Pedestrian	To choose a class for the currently selected targets.
Jump to frame	To jump to a specific frame.
Speed limit	To set a speed limit for the compensated speed of the targets.
Toggle speed	To project the compensated speed of the targets to the camera as their height.
Toggle RCS	To plot the targets in different colors depending on their RCS value.
Toggle cluster	To plot the targets in different colors depending on their DBSCAN result.
Toggle targets	To choose whether to plot the targets on the image.
Previous/Next	choose the previous (Space) or next frame (Left).
Remove	Remove the last annotation of the current frame (Esc).

Table C.1: The functionalities of each button and text box

Bibliography

[1] Sherif Abdulatif, Qian Wei, Fady Aziz, Bernhard Kleiner, and Urs Schneider. Micro-doppler based human-robot classification using ensemble and deep learning approaches. In *2018 IEEE Radar Conference (RadarConf18)*, pages 1043–1048. IEEE, 2018.

[2] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] Aleksandar Angelov, Andrew Robertson, Roderick Murray-Smith, and Francesco Fioranelli. Practical classification of different moving targets using automotive radar and deep neural networks. *IET Radar, Sonar & Navigation*, 12(10):1082–1089, 2018.

[4] A Bartsch, F Fitzek, and RH Rasshofer. Pedestrian recognition using automotive radar sensors. *Advances in Radio Science*, 10(B. 2):45–55, 2012.

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[6] Domenic Belgiovane and Chi-Chih Chen. Micro-doppler characteristics of pedestrians and bicycles for automotive radar sensors at 77 ghz. In *2017 11th European Conference on Antennas and Propagation (EUCAP)*, pages 2912–2916. IEEE, 2017.

[7] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *European Conference on Computer Vision*, pages 613–627. Springer, 2014.

[8] Stephen Blake. Os-cfar theory for multiple targets and nonuniform clutter. *IEEE transactions on aerospace and electronic systems*, 24(6):785–790, 1988.

[9] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila. Eurocity persons: A novel benchmark for person detection in traffic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1844–1861, Aug 2019.

[10] Antonio Brunetti, Domenico Buongiorno, Gianpaolo Francesco Trotta, and Vitoantonio Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300:17–33, 2018.

[11] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[12] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20 (3):273–297, 1995.

[14] Jan Salomon Cramer. The origins of logistic regression. 2002.

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. 2005.

[16] Andreas Danzer, Thomas Griebel, Martin Bach, and Klaus Dietmayer. 2d car detection in radar data with pointnets. *arXiv preprint arXiv:1904.08414*, 2019.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[18] Ahmad El Sallab, Ibrahim Sobh, Mahmoud Zidan, Mohamed Zahran, and Sherif Abdelkarim. Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds. 2018.

[19] M Ester, HP Kriegel, J Sander, and X Xu. Density-based clustering algorithms for discovering clusters. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR, USA*, pages 2–4, 1996.

[20] Pedro F Felzenszwalb, David A McAllester, Deva Ramanan, et al. A discriminatively trained, multiscale, deformable part model. In *the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, page 7, 2008.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[22] Karl Granstrom, Marcus Baum, and Stephan Reuter. Extended object tracking: Introduction, overview and applications. *arXiv preprint arXiv:1604.00970*, 2016.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[25] Steffen Heuel and Hermann Rohling. Pedestrian classification in automotive radar systems. In *2012 13th International Radar Symposium*, pages 39–44. IEEE, 2012.

[26] Steffen Heuel and Hermann Rohling. Pedestrian recognition in automotive radar sensors. In *2013 14th International Radar Symposium*, volume 2, pages 732–739. IEEE, 2013.

[27] Michael Heuer, Ayoub Al-Hamadi, Alexander Rain, Marc-Michael Meinecke, and Hermann Rohling. Pedestrian tracking with occlusion using a 24 ghz automotive radar. In *2014 15th International Radar Symposium*, pages 1–4. IEEE, 2014.

[28] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[29] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[30] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[31] Sangtae Kim, Kwangjin Lee, Seungho Doo, and Byonghyo Shim. Moving target classification in automotive radar systems using transposed convolutional networks. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 2050–2054. IEEE, 2018.

[32] Youngwook Kim and Taesup Moon. Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks. *IEEE geoscience and remote sensing letters*, 13(1):8–12, 2015.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[34] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[36] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[37] Jakob Lombacher, Kilian Laudt, Markus Hahn, Jürgen Dickmann, and Christian Wöhler. Semantic radar grids. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1170–1175. IEEE, 2017.

[38] Sanoal Machado and Santiago Mancheno. Automotive fmcw radar development and verification methods.

[39] Frank Meinl, Martin Stolz, Martin Kunert, and Holger Blume. An experimental high performance radar system for highly automated driving. In *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 71–74. IEEE, 2017.

[40] Andras Palffy, Julian F P Kooij, and Dariu M Gavrila. Occlusion aware sensor fusion for early crossing pedestrian detection. *2019 IEEE Intelligent Vehicles Symposium (IV)*, (30th IEEE Intelligent Vehicles Symposium):1558–1564, 2019.

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[42] Kanil Patel, Kilian Rambach, Tristan Visentin, Daniel Rusev, Michael Pfeiffer, and Bin Yang. Deep Learning-based Object Classification on Automotive Radar Spectra.

[43] Sujeet Milind Patole, Murat Torlak, Dan Wang, and Murtaza Ali. Automotive radars: A review of signal processing techniques. *IEEE Signal Processing Magazine*, 34(2):22–35, 2017.

[44] Rodrigo Pérez, Falk Schubert, Ralph Rasshofer, and Erwin Biebl. Single-frame vulnerable road users classification with a 77 ghz fmcw radar sensor and a convolutional neural network. In *2018 19th International Radar Symposium*, pages 1–10. IEEE, 2018.

[45] Robert Prophet, Marcel Hoffmann, Alicja Ossowska, Waqas Malik, Christian Sturm, and Martin Vossiek. Image-based pedestrian classification for 79 ghz automotive radar. In *2018 15th European Radar Conference (EuRAD)*, pages 75–78. IEEE, 2018.

[46] Robert Prophet, Marcel Hoffmann, Martin Vossiek, Christian Sturm, Alicja Ossowska, Waqas Malik, and Urs Lübbert. Pedestrian classification with a 79 ghz automotive radar sensor. In *2018 19th International Radar Symposium*, pages 1–6. IEEE, 2018.

[47] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

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

[49] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[50] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[51] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[52] Mark A Richards. *Fundamentals of radar signal processing*. Tata McGraw-Hill Education, 2005.

[53] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[55] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[56] Nicolas Scheiner, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. Radar-based feature design and multiclass classification for road user recognition. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 779–786. IEEE, 2018.

[57] Nicolas Scheiner, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. Radar-based road user classification and novelty detection with recurrent neural network ensembles. *arXiv preprint arXiv:1905.11703*, 2019.

[58] Ralph Schmidt. Multiple emitter location and signal parameter estimation. *IEEE transactions on antennas and propagation*, 34(3):276–280, 1986.

[59] Martin Schneider. Automotive radar-status and trends. In *German microwave conference*, pages 144–147, 2005.

[60] Eugen Schubert, Frank Meinl, Martin Kunert, and Wolfgang Menzel. High resolution automotive radar measurements of vulnerable road users–pedestrians & cyclists. In *2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4. IEEE, 2015.

[61] Eugen Schubert, Frank Meinl, Martin Kunert, and Wolfgang Menzel. Clustering of high resolution automotive radar detections and subsequent feature extraction for classification of road users. In *2015 16th International Radar Symposium*, pages 174–179. IEEE, 2015.

[62] Ole Schumann, Markus Hahn, Jürgen Dickmann, and Christian Wöhler. Semantic segmentation on radar point clouds. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2179–2186. IEEE, 2018.

[63] Ole Schumann, Markus Hahn, Jürgen Dickmann, and Christian Wöhler. Supervised clustering for radar applications: on the way to radar instance segmentation. In *2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4. IEEE, 2018.

[64] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *European Conference on Computer Vision*, pages 197–209. Springer, 2018.

[65] Christian Sturm, Gang Li, Gerd Heinrich, and Urs Lübbert. 79 ghz wideband fast chirp automotive radar sensor with agile bandwidth. In *2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–3. IEEE, 2016.

[66] Dave Tahmoush. Review of micro-doppler signatures. *IET Radar, Sonar & Navigation*, 9 (9):1140–1146, 2015.

[67] TI Training. Intro to mmWave Sensing : FMCW Radars. URL `https://training.ti.com/intro-mmwave-sensing-fmcw-radars-module-1-range-estimation`.

[68] RP Trommel, RIA Harmanny, L Cifola, and JN Driessen. Multi-target human gait classification using deep convolutional neural networks on micro-doppler spectrograms. In *2016 European Radar Conference (EuRAD)*, pages 81–84. IEEE, 2016.

[69] Dmitry Utyansky. Digital Signal Processing for Frequency- Modulated Continuous Wave RADARs.

[70] Thomas Wagner, Reinhard Feger, and Andreas Stelzer. Radar signal processing for jointly estimating tracks and micro-doppler signatures. *IEEE Access*, 5:1220–1238, 2017.

[71] Christian Wöhler, Ole Schumann, Markus Hahn, and Jürgen Dickmann. Comparison of random forest and long short-term memory network performances in classification tasks using radar. In *2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–6. IEEE, 2017.

[72] Honghui Yan, Wolfgang Doerr, Alexander Ioffe, and Henrik Clasen. Micro-doppler based classifying features for automotive radar vru target classification. In *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2017.

[73] Zhishuang Yang, Wanshou Jiang, Bo Xu, Quansheng Zhu, San Jiang, and Wei Huang. A convolutional neural network-based 3d semantic labeling method for als point clouds. *Remote Sensing*, 9(9):936, 2017.

[74] Guoqiang Zhang, Haopeng Li, and Fabian Wenger. Object detection and 3d estimation via an fmcw radar using a fully convolutional network. *arXiv preprint arXiv:1902.05394*, 2019.

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