Autolabeling & Semantic Segmentation with 4D Radar Tensors

by

Botao Sun

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday October 31, 2024 at 2:30 PM.

Student number: 5759609

Thesis committee:

Dr. Francesco Fioranelli,	Delft University of Technology, Thesis supervisor
Dr. Julian F. P. Kooij,	Delft University of Technology
MSc Ignacio Roldan,	Delft University of Technology, Daily co-supervisor

TU Delft

Abstract

4D millimeter-wave radar is increasingly important in advanced driver-assistance systems due to its ability to capture Doppler/velocity information and robustness in low-light or adverse weather conditions. Unlike traditional 3D radar, 4D radar provides elevation information, enhancing 3D spatial perception. Most of the perception tasks using 4D radar tensors focus on target classification using bounding boxes. In contrast, although semantic segmentation is typically used to process images and LiDAR point clouds, it has not been well explored for 4D radar tensors.

This MSc thesis aims to bridge the gap in using 4D radar tensors for semantic segmentation and in generating the required labels for supervision. Specifically, it proposes an automatic approach for generating multi-class point-wise labels for automotive datasets by leveraging the complementary information from the synchronized camera and LiDAR data. Then, 4D radar tensors are used as inputs, supervised by the generated labels, to design a radar semantic segmentation network. Promising results are shown by applying both developed parts to the publicly shared RaDelft dataset. The automatic labeling process demonstrates satisfactory quantitative and qualitative results compared with manual labeling results obtained on randomly chosen scenes. The outputs of the semantic segmentation network achieve more than 65% in overall detection probability, improving by +13.1% in terms of vehicle class detection probability, and reducing 0.54 m in terms of Chamfer distance compared to the variants inspired by the literature.

Acknowledgments

Looking back on the entire process of completing my master's thesis, it has been a challenging but rewarding journey. Before starting, I felt more like an undergraduate student with limited research experience in any specific field. I am deeply grateful to my supervisor, Dr. Francesco Fioranelli, for giving me this valuable opportunity, even though I had little prior knowledge about automotive radar. Whenever I encountered difficulties, he always provided insightful perspectives, along with detailed comments and suggestions on both my presentations and reports. Thank you, Dr. Fioranelli, for guiding me throughout this journey.

I am also immensely thankful to my daily supervisor, Ignacio Roldan, who taught me lots of things and proposed several possible approaches from the very beginning of this project. Our frequent weekly meetings provided me with continuous learning opportunities and helped me resolve many challenges throughout the research process. His expertise and deep understanding of the field were crucial to the smooth progress of my research.

I feel honored to have been a member of the MS3 group, where weekly seminars broadened my horizons and taught me much about the group's work. I would like to thank everyone in the group for their valuable suggestions and engaging discussions. Special thanks go to Dr. Olexander Yarovoy for his helpful comments during my mid-term presentation and for asking relevant questions at the EuRAD conference on my behalf. Additionally, I would like to express my gratitude to Dr. Andras Palffy for his insights into the labeling process and his explanation of the *RaDelft* dataset, and to Dr. Yuan Sen for providing me with potential research directions. Lastly, I am thankful to Dr. Julian Kooij for being on my thesis committee and reviewing my work.

Finally, I would like to express my heartfelt gratitude to my parents for their financial support throughout my master's studies and for encouraging me when I faced challenges. My thanks also go to Xinyao Li for her endless emotional support, patience, and care throughout this journey. Her presence provided me with comfort and motivation during the most stressful times. Additionally, I am grateful to my friends, with whom I shared many joyful moments. Their companionship made this journey even more memorable.

Botao Sun
Delft, October 2024

Contents

List of Figures

List of Tables

1

Introduction

In this chapter, the topic of this thesis is introduced, followed by an extensive literature review. First, section 1.1 explains the motivation for selecting the topic of semantic segmentation using 4D radar tensors. Then, subsection 1.2.1 discusses general classification methods using automotive radar in the state of the art, and the types of radar data to be processed. Subsection 1.2.2 critically reviews existing semantic segmentation methods. As the focus of this thesis is on deep-learning-based approaches, subsection 1.2.3 summarizes the available datasets for training and validation, with further details on the RaDelft dataset in section 2.1 of the following chapter. RaDelft will be the dataset to be automatically labeled and utilized in this thesis. Finally, section 1.3 identifies the research gaps from the review and outlines the novel contributions of this thesis.

1.1. Motivation

In the future of vehicle development, autonomous vehicles (AVs) are expected to become dominant. AVs, also known as self-driving cars, are developing rapidly due to the progress of sensing technologies. Multiple sensors are employed simultaneously to perceive surrounding environmental information, enabling AVs to respond effectively to changes in the scenario and surrounding objects. Instead of relying entirely on human decision-making, AVs utilize artificial intelligence in the driving process to predict the behavior of other road users and adapt to changes. This aims to reduce human errors in driving and address congestion challenges, ultimately improving the safety of all road users.

Recent vehicles equipped with multiple sensors including millimeter-wave radars, cameras, and LiDAR assist drivers under specific conditions, such as lane-keeping and collision prevention. Some vehicles can autonomously manage steering, braking, and acceleration in controlled scenarios, such as on highways [1]. These functions correspond to Level 2 in the Society of Automotive Engineers (SAE) Levels of Driving Automation and have become increasingly common in modern vehicles. However, the realization of fully Advanced Driver Assistance Systems (ADAS) has not yet been achieved. One of the most important reasons is that the performance of LiDAR and cameras will be significantly influenced during nighttime or adverse weather conditions, such as dense fog or rain. In these scenarios, the robustness of radar sensors to such environmental factors can mitigate the aforementioned issues.

Millimeter-wave radars, cameras, and LiDAR are commonly used in ADAS for their perception functions. Specifically, ADAS capabilities generally incorporate multiple cameras for a 360° view of the own-vehicle surroundings. Radar sensors transmit and receive millimeter waves to measure the real-time distance, velocity, and azimuth angle of targets. LiDAR can generate three-dimensional (3D) representations of detected items and their surroundings. Table 1.1 briefly illustrates the advantages and disadvantages of using the three most general sensors (radar, camera, and LiDAR) for ADAS.

Sensor	**Advantage**	**Disadvantages**
Radar	1) Working in low light and adverse weather conditions 2) Effective in speed detection	1) Lacking elevation information (Traditional 3D radar) 2) Relatively low angle resolution
Camera	1) Providing a colorful perspective of the environment 2) Effective for reading road signs, traffic signs, and lane detection	1) Without depth information 2) Highly influenced by low light conditions
LiDAR	1) Providing the accurate location of targets 2) Generating 3D information of the surroundings	1) High cost and maintenance compared to other sensors 2) Highly influenced by adverse weather conditions such as rain or fog

Table 1.1: Brief list of advantages and disadvantages of radar, camera, and LiDAR equipped on AVs, inspired by [2]

From the Table 1.1, it can be concluded that radar has its unique function of directly measuring the velocity information via the Doppler effect when detecting dynamic targets. Also, radars are more robust under insufficient light or adverse weather conditions than cameras and LiDAR, which can be an important complement when using other sensors. Compared to other sensors, radars are relatively cheap and can easily be deployed in multiple units to gather multi-view information [3]. These advantages indicate that radar sensors are playing a more and more important role in ADAS.

Despite the robustness of radar sensors, they are still limited in environmental sensing ability due to the lower angle resolution and the lack of elevation information compared to LiDAR sensors. Traditional 3D radar captures the speed, distance, and horizontal/azimuth angle, providing target localization in a two-dimensional (2D) environment without height information. To improve the radar performance in target localization in 3D space, four-dimensional (4D) radar was introduced in 2018 [4]. Unlike 3D radar, 4D radar allows localization in a 3D environment using the range, horizontal/azimuth angle, and

vertical/elevation angle information. This is achieved by increasing the number of transmission and reception channel pairs and integrating more offset antennas in the elevation angle into a chip [5]. This master thesis is heavily based on 4D radar sampled data, with specific details related to the data introduced in section 2.1.

In state-of-the-art ADAS algorithms, deep-learning methods have become more and more common when using cameras as sensors for environmental perception, with semantic segmentation playing an important role. Semantic segmentation assigns each pixel in an image to a class, allowing for accurate scene segmentation [6]. Beyond detecting targets, these methods capture critical information from scenario objects like roads, buildings, and trees, enabling the vehicle to identify drivable areas and avoid obstacles. Compared to standard target classification methods, semantic segmentation approaches provide well-grain information by generating class masks for each pixel.

Semantic segmentation has also been applied to LiDAR point clouds (PCs) in recent years [7]. Each point in the PCs can be classified directly, or the PCs can be transformed into a uniform cube for processing. Using 3D convolution, the fully convolutional network structure in image semantic segmentation can process 3D voxel data [8] to generate class masks. The output also provides 3D information which is difficult to capture with the camera.

Unlike LiDAR PCs, semantic segmentation using radar data is still in its early stages and faces several challenges. One issue is that radar PCs are often too sparse, leading to less informative data compared to LiDAR PCs. Additionally, traditional 3D radar lacks elevation data. However, recent advancements in 4D radar technology with additional height information have improved the feasibility of more accurate spatial perception, albeit with much coarser resolution than with a LiDAR.

In this context, when training a semantic segmentation network using radar tensors, LiDAR PCs with multi-class 3D masks can be utilized as labels, providing rich and detailed ground truth data for supervision. Unlike traditional radar target classification methods that use bounding boxes, semantic segmentation aims to address issues such as variations in target signature size and conflation due to sensor resolution differences. In complex road scenes, targets like vehicles, bicycles, and pedestrians are usually easily bounded by boxes, but buildings, trees, and poles or lampposts are challenging due to their varied shapes and sizes. Semantic segmentation methods are also valuable for open-space segmentation of drivable areas.

Considering the rich content directly provided by radar tensors rather than processed radar PCs, this kind of data format is expected to keep all important information. Therefore, semantic segmentation using 4D radar tensors has the potential for complex road scene understanding in 3D space and will be the general topic of this thesis. This approach is regarded as an important complement to cameras and LiDAR, particularly in various weather and low-light conditions, enhancing fully autonomous driving capabilities.

1.2. Literature review

This section provides a critical literature review of the relevant state of the art, starting from the general signal processing pipeline for automotive radar data. Then, target classification algorithms using automotive radar are summarized, including various data types and corresponding methods. Notably, semantic segmentation on range-azimuth-elevation-Doppler (RAED) tensors generated by 4D radar data remains unexplored. After this, the review examines the application of semantic segmentation from 2D images to 3D space, indicating the feasibility of semantic segmentation on RAED tensors. Finally, considering the thesis' objectives, the review discusses the available radar datasets, identifying a gap in datasets that provide RAED tensors as usable data and 3D multi-class point-wise masks as corresponding labels.

1.2.1. Target classification using automotive radar

Frequency Modulated Continuous Wave (FMCW) radar is the most commonly used radar in automotive vehicles. Radar operates by transmitting radio waves to detect and locate targets through their reflection, requiring signal processing to transform the raw data into meaningful information. Typically, the conventional radar signal processing pipeline starts with the raw data from an Analog-to-Digital Converter (ADC) captured by sensors and processed through three Fast Fourier Transforms (FFTs) to

yield a dense range-azimuth-Doppler (RAD) tensor for each frame, representing range, azimuth, and Doppler along its three axes. The values in this tensor represent the power at each corresponding bin (or cell or voxel). For 4D radar, an additional FFT in elevation produces an RAED tensor. The RAD/RAED tensor can be further processed by compressing along different dimensions: compressing along the Doppler axis generates a range-azimuth (RA) map, and along the azimuth axis generates a range-Doppler (RD) map [9]. A constant false alarm rate (CFAR) detector [10] is then generally applied to the signal, producing sparse radar PCs by retaining only strong responses and rejecting noise and clutter artefacts. Additionally, PCs can generate an occupancy grid map to detect static obstacles [11]. Figure 1.1 illustrates the aforementioned signal processing pipeline. These four types of radar data formats (i.e., RAD tensor, RA/RD map, Radar PCs, and Occupancy grid map) generated from the raw data are used in multiple scenarios and tasks such as detection, classification, and tracking.

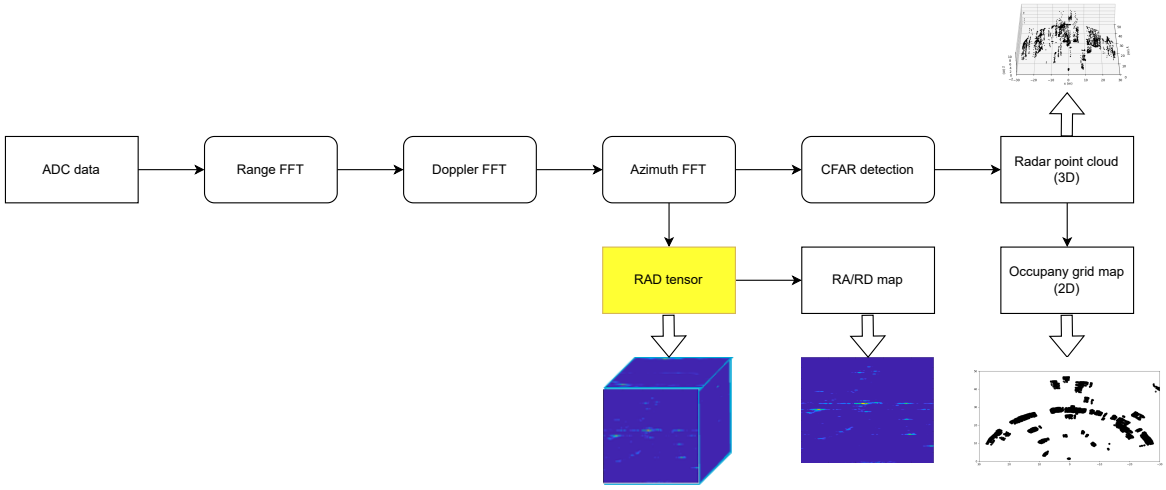

Figure 1.1: Conventional FMCW radar signal processing pipeline. Four types of data (RAD tensor, RA/RD map, Radar PCs, and Occupancy grip map) are mostly used for object detection, classification, and tracking tasks.

The RAD tensor includes more complete information since it is directly generated from raw radar data through three FFTs, and represents power value across RAD bins. This data is expected to be effective for target classification. However, processing the dense RAD tensors often requires a large computational load. To address this issue, researchers compressed the RAD tensor along different axes to create two-dimensional (2D) RA, RD, and azimuth-Doppler (AD) heatmaps, reducing computational load while maintaining key information. Major et al. compressed the 3D RAD tensor by summing over each dimension, followed by convolution and concatenation to extract features. Long Short-Term Memory (LSTM) modules help the model capture the dynamic scene changes. This approach is primarily used to detect dynamic vehicles and provide 2D bounding boxes [12]. Similarly, Gao et al. processed the RAD tensor like Major et al. using 3D inception for multi-level feature extraction, enabling the classification of pedestrians, vehicles, and bicycles [13]. Contrasting with bounding box annotations, Jin et al. utilized camera images as ground truth, inputting RD maps for free-space segmentation and object detection, thus reducing labeling costs [14]. Ouaknine et al. extracted semantic features from RA, RD, and AD maps, combining them to generate segmentation masks for various classes. They also used consecutive frame data to improve accuracy and applied Atrous Spatial Pyramid Pooling (ASPP) modules to capture features at multiple scales [15]. Processing 2D RA, RD, and AD maps is essentially similar to image processing which achieves good performance in 2D scene understanding due to established methods.

With the improvement in computational capabilities, researchers began to process the RAD tensor directly. Wang et al. applied data from both radar and cameras to provide target classes and locations [16]. Their network directly inputs 3D RAD tensors with temporally variable 3D convolution to fuse multi-frame data, in order to extract dynamic features better and address inter-frame alignment issues caused by target motion. Palffy et al. proposed a method that used only low-level radar data to classify road users (pedestrians, bicycles, and vehicles) [17]. Their model employs camera views to supervise

target locations and utilizes whole RAD tensors for target classification, finally providing 3D bounding boxes for targets of interest. Unlike the traditional Convolutional Neural Network (CNN) based methods, Meyer et al. introduced the application of Graph Convolutional Network (GCN) on RAD tensors [18]. In their approach, the graph is constructed directly from the RAD tensor, where each node represents features in a unique range-beam cell. This helps maintain uniform features at different distances.

Target classification using the entire RAD tensor is popular since the Doppler information helps distinguish high-speed targets from low-speed targets. Additionally, there is a dependency between Doppler information and angle estimation for dynamic targets [19]. Despite these advancements, to the best of the author's knowledge there is still no semantic segmentation directly on the RAD tensor due to the lack of suitable datasets, potential computational challenges, and the complexity of incorporating elevation information.

After processing with CFAR detectors, only points with strong responses are retained, resulting in radar PCs. These PCs are more intuitive than RAD tensors as they exist directly in 3D space which can also be used for target classification. Target classification with radar PCs has shifted from traditional methods to deep learning techniques to improve accuracy and suit different scenarios. Initial work involved clustering to identify possible targets, followed by the extraction and classification of manually designed features [20]. However, with the capabilities of deep learning networks that can automatically extract features, researchers gradually give up the manually designed features.

Inspired by Qi et al., who introduced PointNet [21] and PointNet++ [22] for semantic segmentation on PCs, Schumann et al. proposed a recurrent architecture designed for dynamic targets, enhancing semantic extraction from multiple radar sensors [23]. However, radar PCs are usually (much) sparser and noisier than LiDAR PCs, and 3D millimeter-wave radar lacks elevation data. To address these problems, [24] introduced an integration framework combining LiDAR and radar PCs for improved target classification. This framework employs query-based height and Bird's Eye View (BEV) fusion techniques to maximize information utilization from two sensors during training. Recently, increased focus on the graph structure of PCs has led to innovative approaches in target classification. Shi et al. utilized graph construction by connecting spatially proximal points to develop a single-shot detector [25]. After this, the first application of GCN for target classification in radar PCs has achieved competitive results [26]. Overall, due to the similar structure between LiDAR PCs and radar PCs, target classification methods based on radar PCs are relatively mature.

Radar occupancy grid maps are constructed by projecting radar PCs onto a BEV grid. Since 3D radar PCs are often sparse, projecting them onto a BEV grid increases data density. Kund et al. proposed a model with the input radar 2D grids supervised by LiDAR data to handle occlusions in laser scans [27]. Inspired by image semantic segmentation, networks designed for segmenting camera images can effectively process accumulated radar 2D grids, and finally generate BEV occupancy semantic information [28]. Schumann et al. used a CNN on the 2D grid to detect static targets and a recurrent neural network directly on radar PCs to capture dynamic information, merging these to generate combined semantic PCs [29]. However, due to the relatively simple 2D structure of occupancy grids, their processing may be less effective compared to end-to-end processing of raw radar data.

Summarizing, for the four types of radar data formats considered in this review, it can be briefly concluded that there are 4 main approaches for target classification including cross-sensor supervision, GCN, feature fusion, and semantic segmentation. Among these, both radar occupancy grid maps and RA/RD maps are 2D, which are not appropriate for GCN. Table 1.2 summarizes the radar data types and the main methods for target classification, citing a representative paper for each combination. The first row represents the considered radar data types and the first column represents the methods applied for target classification in the literature. Notably, applying semantic segmentation to RAD tensors remains unexplored, which is the research focus of this thesis.

Methods \ Data types	Radar PCs	**RAD tensors**	Radar Occupancy grid maps	RA/RD maps
Cross-sensor supervision	Tan et al. [30]	Palffy et al. [17]	Kung et al. [27]	Jin et al. [14]
GCN	Svenningsson et al. [26]	Meyer et al. [18]	\	\
Feature fusion	Wang et al. [16]	Yang et al. [31]	Schumaan et al. [29]	Gao et al. [13]
Semantic segmentation	Schumaan et al. [29]		Prophet et al. [28]	Ouaknine et al. [15]

Table 1.2: Radar data formats to be processed in the automotive context and the corresponding methods for target classification, as reviewed for this thesis. Applying semantic segmentation on RAD tensors for target classification, highlighted in yellow, remains under-explored for target segmentation and a targeted research gap in this thesis.

1.2.2. Semantic segmentation methods

Deep learning-based semantic segmentation methods have developed rapidly in recent years. A significant innovation was the introduction of fully convolutional networks (FCNs) by Long et al., solving the computational efficiency and limited receptive fields when using traditional CNN [32]. FCNs consist of convolutional and downsampling layers followed by transposed convolutions for upsampling. This approach enables dense pixel prediction without fully connected layers. Another notable architecture is the encoder-decoder structure used in U-Net, where skip connections are applied to fuse low-level and high-level data [33]. This increases segmentation accuracy through high-resolution feature maps. To address multi-scale object challenges, feature pyramid networks (FPNs) were proposed with top-down structure and lateral connections to merge high-resolution shallow layers with semantically rich deep layers. For the encoder backbone, residual network (ResNet) is often applied to avoid the problems of vanishing gradients. Additionally, the Atrous Spatial Pyramidal Pooling (ASPP) layer in Deeplab v3 expands the receptive field of the network with various dilation rates, effectively learning multi-scale features [34].

Despite these advances in network structure, CNN-based methods are limited by their convolutional kernels which means that only local information is used to understand the input images. To overcome this limitation, transformer-based methods have been adapted from natural language processing to computer vision. Vision Transformer (ViT) was the first proposed transformer to 'tokenize' the input image and then use the self-attention mechanism, obtaining a global perspective at the beginning of the model [35]. Subsequent transformers with different backbones have improved segmentation results. However, these are not suitable here due to their computational complexity and the special radar RAED tensors format that will be used in this thesis.

Based on competitive semantic segmentation results on 2D images, this task is further extended to 3D LiDAR PCs. For the direct process of points, PointNet [21] and PointNet++ [22] are the two most representative methods. PointNet independently extracts features for each point without using neighborhood information. However, this results in ignoring relationships between neighboring points, which significantly impacts the accuracy. PointNet++ addresses this by clustering PCs into subsets and then applying PointNet to each subset to extend the receptive field, but this increases complexity when processing large-scale PCs.

Another approach transforms LiDAR PCs into a uniform 3D cube. In VoxelNet [8], each voxel contains a fixed neighborhood and a variable number of points, creating a sparse tensor. Voxel feature encoding layers were used to combine point-wise features and locally aggregated features. This process should be realized manually in LiDAR PCs. In contrast, radar RAD tensors are already in voxelized form, eliminating the need to transform PCs into a voxelized format. This simplifies the data pre-processing pipeline and indicates the potential of using radar RAD tensors for semantic segmentation. At the same time, the voxelization process in LiDAR PCs loses some information, while radar RAD tensors retain all original data. Moreover, transformer-based methods have shown promising performance on available datasets. The Point Transformer [36] applies a self-attention mechanism to PCs, suggesting that transformers are better suited for PC processing than for natural language and image processing.

Subsequently, more transformers-based methods have achieved good performance on various indoor and outdoor LiDAR datasets.

Existing semantic segmentation methods for radar are limited and primarily operate in the 2D domain. Dimitrievski et al. proposed a weakly supervised method to acquire semantic information of vulnerable road users in a 2D occupancy grid combined with temporal information [37]. This approach separates road targets into vulnerable and non-vulnerable classes, restricting the information predicted from radar RAD tensors, thereby making it only a complement to other sensors. To acquire more information, Schumaan et al. applied two different branches to segment the dynamic objects and static objects from radar PCs and occupancy grids, providing rich information from the BEV view. Temporal information is incorporated into the network when predicting dynamic targets [29]. However, this approach is limited by the loss of input information due to the use of LiDAR PCs and the occupancy grid. To reduce the information loss, Ouaknine et al. applied the ASPP structure and combined the features from RA and RD maps to predict the semantic information on these maps with acceptable results [15]. However, it should be noted that this task remains in the 2D domain without elevation information, and the scene is relatively simple where the maximum number of objects in a scene is only 2. Objects are directly labeled on the RA and RD maps, which are difficult to generate and not intuitive. Currently, to the best of the author's knowledge, there are no available methods for semantic segmentation with 4D radar tensors in complex 3D scenes.

In summary, originated in the field of computer vision, semantic segmentation performs well in 2D scene segmentation. It is also effective for 3D applications like LiDAR PCs in both indoor and outdoor scenes. However, its application in automotive radar is still in the early stages. Unlike LiDAR PCs, radar RAD/RAED tensors are uniform and do not require voxelization. Additionally, direct Doppler information in each voxel eliminates the need for manual feature extraction. Motivated by this gap, this thesis will explore the potential of semantic segmentation in automotive radar perception in 3D space. For this, the availability of suitable datasets is key and the most prominent ones are reviewed in the following section.

1.2.3. Automotive radar datasets

Since this project focuses on underlying automotive radar data, pre-CFAR datasets will be summarized first. Pre-CFAR datasets, including RAD tensors or RA/RD maps, have been widely used in detection and classification tasks because they keep most of the information deriving from raw radar data. To achieve target classification in 3D space using automotive radar, 4D radar datasets are required as 4D radar offers additional elevation data compared to traditional 3D radar. However, the number of datasets for 4D radar is limited, and existing datasets are not tailored for semantic segmentation tasks. They mainly consist of 3D bounding box annotations which are different from the point-by-point annotations required for semantic segmentation. An automatic labeling process is thus necessary for generating semantic labels. This process will leverage synchronous LiDAR PCs and RGB images, requiring a dataset that includes both.

Among existing 3D radar datasets with multi-class masks as annotations, CARRADA [38] offers 2D bounding boxes and masks that can be applied for 2D segmentation in simple scenes, while GhentVRU [37] provides masks specifically for vulnerable road users (VRUs), i.e., only two classes (VRUs and background) are annotated. Due to the lack of elevation information, these datasets cannot extract 3D space information from radar tensors and so will not be considered.

For the 4D automotive radar dataset with RAED tensors, K-radar [39] provides low-level data and 3D bounding box annotations. RADIal dataset [40] offers the most comprehensive data types including raw data, RAED tensors, RA/RD maps, and radar PCs, with open-space masks, that are suitable for free-driving space segmentation tasks. However, it lacks enough classes for multi-class classification tasks. Currently, to the best of the author's knowledge, there is no available 4D radar dataset containing low-level radar data types with multi-class masks as annotations in 3D space for semantic segmentation. This is the gap the proposed labeling process for the *RaDelft* dataset [41] aims to fill. In the *RaDelft* dataset, 16975 frames across seven different scenes were recorded in Delft. A 4D Texas Instrument radar is used to capture data, providing raw radar data and RAED tensors. Synchronous RGB images and LiDAR PCs are also included and calibrated, which will be useful for generating 3D multi-class masks.

Although K-radar and RADIal datasets are 4D radar datasets with synchronous RAED tensors, LiDAR PCs, and RGB images, they are not ideal for the automatic labeling process. For the K-radar dataset, due to the positional differences between Lidar and Radar, targets may appear in the radar view, but not in the LiDAR view [39]. Manual calibration is required to obtain more accurate labels. The RADIal dataset employs low-resolution LiDAR (16-layer) which impacts the label generation accuracy [40], and is currently unavailable. Consequently, validating the proposed labeling on *RaDelft* is recommended to avoid these issues. Table 1.3 summarizes the data and annotation types in existing 3D and 4D automotive radar datasets reviewed in this thesis, including our *RaDelft* dataset.

Dataset	Radar type	Data format	Annotation type
UWCR [13]	3D radar	Raw radar data	2D bounding box
RADDet [42]	3D radar	RAD tensor	3D bounding box
CARRADA [38]	3D radar	RAD tensor, RA/RD map	2D mask, 2D bounding box
GhentVRU [37]	3D radar	RAD tensor	Mask for VRUs
Astyx Hires2019 [43]	4D radar	Radar PCs	3D bounding box
View-of-Delft [17]	4D radar	Radar PCs	3D bounding box
TJ4DRadSet [44]	4D radar	Radar PCs	3D bounding box, Tracking ID
K-radar [39]	4D radar	RAED tensor	3D bounding box, Tracking ID
RADIal [40]	4D radar	Raw radar data, RAED tensor, RA/RD map, radar PCs	Open-space masks
***RaDelft* [41]**	**4D radar**	**Raw radar data, RAED tensor**	**3D multi-class masks**

Table 1.3: Data format and annotation types in existing radar datasets and in *RaDelft*, where the required labels for *RaDelft* are generated by the work proposed in this thesis.

1.3. Research gaps and thesis contributions

Based on the review of general target classification methods using automotive radar, there is a gap in applying semantic segmentation to RAED tensors. RAED tensors are of interest because they can provide advantages such as less information loss than any other types of radar data and additional Doppler information compared to radar PCs and occupancy grid maps. These benefits give RAED tensors the potential for high-quality object detection and target classification. Semantic segmentation at the level of individual voxels can address mismatches caused by sensor resolution differences and challenges from various sizes of scenario objects when using bounding boxes. In ADAS, generated segmented radar PCs complement semantic results from cameras and LiDAR regardless of resolution differences and post-processing, particularly in adverse weather, improving ADAS robustness.

Another gap is the absence of a dataset containing 4D radar RAED tensors and multi-class point-wise labels on LiDAR PCs. Applying semantic segmentation to RAED tensors with height information enables the direct use of LiDAR PCs with semantic information as labels without the need for bounding boxes. Due to the high cost of manual labeling, an automatic annotation generation process is required for datasets with the aid of synchronous RGB images and LiDAR PCs.

Summarizing, the gaps in the existing target classification using automotive radar methods and 4D radar automotive datasets are listed as follows:

- There is currently no available 4D automotive radar dataset that provides RAED tensors along with point-wise multi-class annotations on LiDAR PCs simultaneously.

- The application of semantic segmentation to radar RAED tensors remains unexplored. Semantic segmentation applied directly on radar PCs generated after CFAR has a large performance gap compared with LiDAR results, resulting in substantial target loss.

The two main contributions of this thesis are summarized as:

- Designing an algorithm to automatically generate multi-class point-wise masks for scenario objects, bicycles, pedestrians, and vehicles, and validating it with the *RaDelft* dataset. The generated labels/masks are applicable in subsequent classification and segmentation tasks. Notably, the proposed approach is not limited to this dataset but can be replicated on all kinds of LiDAR PCs with the corresponding RGB images.
- Proposing and training a multi-dimensional convolutional neural network to achieve joint detection and classification in 3D space only using the 4D radar tensors. The output of the proposed network is the voxelized radar PCs in polar coordinates, which can be transformed into Cartesian coordinates, resulting in more accurate radar PCs than using the conventional CFAR method. With this approach, the differences between radar PCs and LiDAR PCs are reduced. Compared to the previous work of our group [45], better detection performance plus additional classification at the level of individual voxels are now performed simultaneously.

This work is currently under review for the 2025 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP. Moreover, the generated labels and related codes are publicly available on *4TU.Federation* and `https://github.com/RaDelft/RaDelft-Dataset` for future research.

1.4. Thesis outline

This thesis begins by outlining the motivation for autolabeling and semantic segmentation on 4D radar tensors. A comprehensive literature review is provided in chapter 1, including target classification methods using automotive radar, existing semantic segmentation methods, and automotive radar datasets.

In chapter 2, the detailed process of the proposed labeling method is introduced, including preliminary object label generation, calibration process, and transformation and voxelization. After the first two steps, multi-class point-wise labels are generated for LiDAR PCs. The final labels are in voxelized format with values ranging from 0 to 4, representing ‘empty’, ‘scenario objects’, ‘vehicles’, and ‘bicycles’ respectively. Additionally, motion state labels for dynamic vehicles are generated through consecutive LiDAR PC frames.

Chapter 3 describes the structure of the proposed semantic segmentation network. The network’s input consists of RAE tensors processed from RAED tensors, and its output has the same dimensions as the labels. The network architecture includes both a 2D backbone and a 3D backbone to efficiently extract features and address the sparsity differences between inputs and outputs.

In chapter 4, both quantitative and qualitative results are presented for the automatic labeling method and semantic segmentation network. The labeling results are compared with manual labeling for reference. For the semantic segmentation network, comparisons are made between the proposed baseline, several modifications based on the baseline, and two variants from the literature.

Chapter 5 discusses potential challenges and cases of interest encountered during the implementation of this project, focusing on both the automatic labeling process and semantic segmentation network. Finally, chapter 6 summarizes the findings in the research and suggests some potential direction for future work.

2

Proposed automatic labeling process

This chapter presents the design of an automatic labeling pipeline for generating multi-class point-wise labels. It begins by introducing the basic information of the RaDelft dataset (section 2.1). Following this, the developed three-step automatic labeling process is introduced in section 2.2, including the preliminary object label generation, calibration, and transformation & voxelization. This process generates the voxelized labels with five distinct classes in the polar coordinates. Additionally, to evaluate the detecting performance of dynamic vehicles, motion state labels are generated for further analysis in section 2.3.

2.1. Introduction of RaDelft dataset

The *RaDelft* Dataset was collected by Roldan et al. and includes synchronous radar raw data, LiDAR PCs, and RGB images for 16975 frames across seven different scenes recorded in Delft [41]. The radar raw data is captured by a Texas Instrument radar board MMWCAS-RF-EVM, which is Multiple-Input Multiple-Output (MIMO) with four cascaded chips resulting in 16 receivers and 12 transmitters, providing high-resolution azimuth and additional height information. The LiDAR PCs are recorded by a RoboSense Ruby Plus Lidar deployed on the roof near the radar. Radar and LiDAR have a frame rate of approximately 10 frames per second (fps). A video camera with a resolution of 1936 $\times$ 1216 pixels and a frame rate of 30 fps is installed on the windshield to capture the RGB images. All sensors have been spatially calibrated using reference targets, enabling projection, translation, and rotation of recorded data across sensors.

Figure 2.1 demonstrates the assembly locations of radar, LiDAR, and camera. The figure shows the LiDAR and radar installed parallel on the car roof, with the camera positioned on the windshield. This configuration ensures that the radar and LiDAR perception areas are similar and can be transformed through translation and rotation. Here, LiDAR PCs are point-wise labeled and serve as the ground truth after translation and rotation to the radar view. RGB images are used for calibration after projecting the LiDAR PCs onto them. Table 2.1 lists some key parameters related to the labeling process of the *RaDelft* dataset.

Figure 2.1: The sensor suite of the vehicle used to collect the *RaDelft* dataset, which includes closely placed radar, LiDAR, and camera, ensuring similar perception areas.

Key parameters	**Value**
Radar frame rate	10 fps
LiDAR frame rate	10 fps
Camera frame rate	30 fps
Radar azimuth view	$-70°$~$70°$

Key parameters	Value
Radar elevation view	$-20°$~$20°$
Radar maximum unambiguous range	$51.4m$
LiDAR elevation view	$-25°$~$15°$
Camera azimuth view	$-34°$~$34°$

Table 2.1: Some key parameters related to the labeling process of the *RaDelft* dataset.

Since the camera frame rate is higher than that of Radar and LiDAR, the closest image should be selected for label generation. The LiDAR's field of view (FoV) is $360°$, whereas the radar azimuth view ranges from $-70°$~$70°$. LiDAR PCs should be cropped to match the radar view. More details about the labeling process will be introduced in section 2.2.

Radar tensors in the *RaDelft* dataset are generated from raw data by radar signal processing, including the execution of two 2D FFTs. Unlike the traditional process mentioned in Subsection 1.2.1, Roldan et al. proposed an additional time division multiple access (TDMA) compensation step between the FFTs to resolve Doppler ambiguity and phase migration effects [45]. To mitigate grating lobes caused by the sparse antenna configuration vertically, the elevation dimension is limited to the $\pm 15°$ range, keeping only the elevation value with the highest power. The resulting radar tensor has two channels: one for the normal RAD tensor with power values and another with a single elevation value from 1 to 34 corresponding to the highest power. Compared to keeping all elevation information for each RAD tensor, this method significantly reduces the data size. Notably, the values in these two channels have different meanings and should be normalized for further processing. Consequently, the dimension of the RAED tensor is $N_C \times N_D \times N_A \times N_R (2 \times 128 \times 240 \times 500)$, representing channel, Doppler, azimuth, and range, respectively.

2.2. Automatic labeling process

This section describes an automatic pipeline for generating the voxelized LiDAR PCs with class information using LiDAR data and RGB images taken at the same timestamp. The process comprises three steps. First, a pre-trained 3D object detection algorithm is applied to the LiDAR PCs to generate preliminary object labels. Then, these labeled PCs are calibrated using camera data. Finally, the classified LiDAR PCs are transformed to radar view and converted to polar coordinates in the dimension of $N_R \times N_A \times N_E (500 \times 240 \times 34)$, representing range, azimuth, and elevation respectively. Figure 2.2 demonstrates the entire process, starting from the LiDAR PCs and corresponding RGB images, point-by-point labels are generated in the radar view.

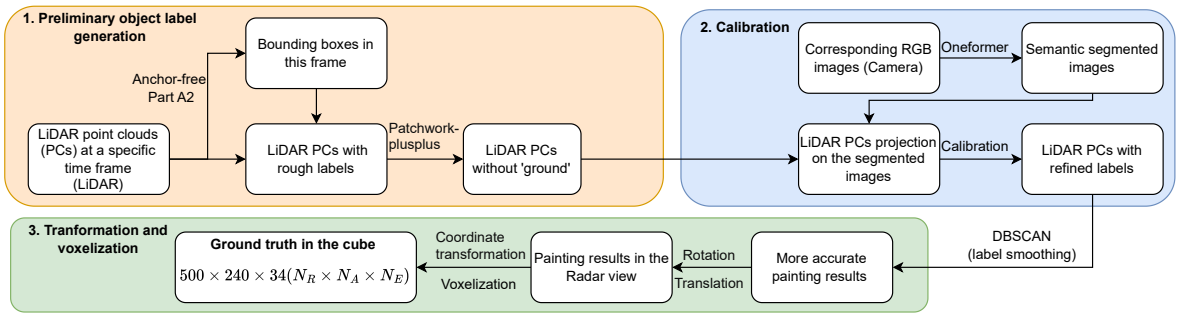

Figure 2.2: Block diagram of the proposed automatic labeling process using LiDAR PCs and RGB images. First, preliminary labels on LiDAR PCs are generated through a pre-trained object detection model. Semantic information from the images is then used to calibrate key labels in the central view, followed by label smoothing using DBSCAN. Finally, point-wise multi-class labels are generated through coordinate transformation and voxelization.

In the LiDAR PCs, the differences between vegetation, poles, traffic signs, and buildings are distinguishable due to high resolution. However, radar has a lower resolution in range, azimuth, and elevation

resolution than LiDAR. At the same time, one of the most important features in radar data is Doppler information, but static objects lack enough Doppler information for classification. Thus, we classify these static objects as 'scenario objects'. Other targets include pedestrians, bicycles (including motorcycles), and vehicles. Consequently, there are five different classes in the ground truth: background, scenario objects, pedestrians, vehicles, and bicycles, represented by values ranging from 0 to 4 in the voxelized LiDAR PCs.

2.2.1. Preliminary object label generation

Benefiting from the rapid development of 3D object detection only using LiDAR PCs, rough object labels can be generated through a pre-trained object detection model applied directly to the LiDAR PCs $L \in \mathbb{R}^{N,4}$, where N is the number of points, and the four dimensions represent the x, y, z coordinates and intensity of each point. In this task, we utilize the anchor-free '**Part-A2**' [46] model trained on the KITTI dataset [47]. The model comprises two modules: the part-aware module and the part-aggregation module. The part-aware module generates foreground segmentation and object locations. Then, the part-aggregation module processes voxelized features through sparse convolution, followed by scoring and regression refinement. The final outputs are the bounding box parameters and the confidence score.

The anchor-free method is suitable for dealing with irregular scenes. Compared to the anchor-based strategy, it is lighter and more memory-efficient. It also avoids the need to set bounding box dimensions before processing. This is particularly important for detecting small-scale pedestrians and bicycles at long distances from the LiDAR. Since the maximum range is 51.4 m, the size of pedestrians and bicycles varies significantly between close and distant targets. Moreover, the anchor-free Part-A2 model outperforms the anchor-based Part-A2 model in detecting pedestrians and bicycles in the KITTI dataset [46].

Considering that the pre-trained model is not specifically designed for the *RaDelft* dataset and trained on the *RaDelft* dataset, its performance is suboptimal. The KITTI dataset applies a 64-layer Velodyne LiDAR, while the *RaDelft* dataset employs a 128-layer RoboSense LiDAR, resulting in denser PCs. This high density particularly affects bicycles close to the sensor which are not bounded as they may be recognized as barriers or buildings. Thus, we first downsample the LiDAR PCs to reduce the density of PCs before using this pre-trained model. A high-confidence threshold of 0.5 is set to ensure detection accuracy. However, this leads to some of the short poles being detected as pedestrians. To address this, a higher confidence of 0.8 is set for pedestrians beyond 30 m from LiDAR. Although some distant pedestrians may be missed, radar also struggles to detect them effectively.

Based on the confidence threshold, only effective bounding boxes with class information are kept. Using the seven bounding box parameters including center coordinates x, y, z, bounding box size d_x, d_y, d_z, and rotation angle $heading$ along the z-axis, the vertex coordinates in the Cartesian coordinate are generated. Points within the bounding box are painted to the corresponding colors. A schematic diagram of this process is shown in Figure 2.3. Visualization results about the bounding boxes and the initial painted results will be demonstrated in the section 4.1.

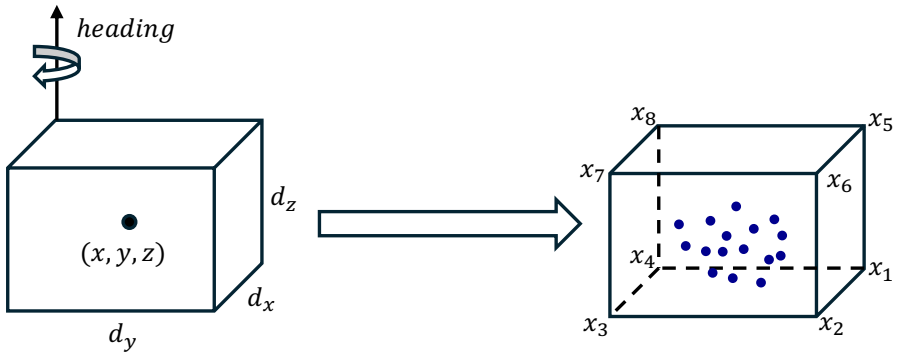

Figure 2.3: The process of transforming bounding box parameters to vertex coordinates. Points within the bounding box are painted to the corresponding color.

Currently, the LiDAR PCs $L \in \mathbb{R}^{N,4+C}$ with N points and C classes are still with 360° FoV and 200 m detection range, and the forward targets are labeled. According to the radar parameters, the whole LiDAR PCs are cropped to fit the radar view with the azimuth view from -70 ~ 70°, the maximum range to 51.4 m, and the elevation view from -15° ~ 15°. The transformation from LiDAR to radar view occurs before the cropping process. After cropping, the PCs are transformed back to the LiDAR view for calibration. Notably, LiDAR also detects the ground points, resulting in dense clusters that can significantly affect radar detection performance. The ground points can be adaptively eliminated by **Patchwork++** [48], resulting in $L_{\text{reduced}} \in \mathbb{R}^{N',4+C}$. After this, only the scenario objects, pedestrians, vehicles, and bicycles are preserved for the further process.

2.2.2. Calibration process

Despite setting a relatively high confidence threshold, some mislabeling situations still occur. To solve this, images captured by the cameras can be used for calibration. As mentioned in Table 2.1, although the camera has a more limited FoV than radar, it is placed directly below the radar. This alignment ensures that the camera's view is the central field of the radar which is more crucial than the side view. Thus, the semantic information provided by the images can be used to calibrate some key targets in the central view.

To extract the semantic information of the images, we use the transformer-based pre-trained model **OneFormer** [49]. This model realizes panoptic segmentation, instance segmentation, and semantic segmentation at the same time, which performs well on multiple public datasets. From the RGB image $I \in \mathbb{R}^{W,H}$ where W represents width and H represents height, as the frame rate of the camera is much higher than LiDAR, the closest image is found for each LiDAR frame. Then, semantic information $S_{\text{semantic}} \in \mathbb{R}^{W,H,C}$ is obtained where C represents the number of classes.

Applying the semantic information S_{semantic}, we execute a projection from the 3D PCs to the 2D images to align the PCs with the pixels in 2D images. Inspired by [50], extrinsic calibration parameters $\mathbf{T}^{\text{LiDAR,Camera}} \in 4 \times 4$ between the camera and LiDAR can be used to estimate the relative rigid transformation, enabling the transformation from the LiDAR view to the camera view. The transformation is given by:

$$\mathbf{T}^{\text{LiDAR,Camera}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

where $\mathbf{R}$ represents the 3×3 rotation matrix, and $\mathbf{t}$ represents the 3D translation vector. To use the transformation, we adjust the value in the strength dimension of L_{reduced} to 1, resulting in $L_{\text{homo}} \in \mathbb{R}^{N',4}$. Then, the transformed points L_{homo} are expressed as:

$$L_{\text{Camera}} = (\mathbf{T}^{\text{LiDAR,Camera}} \cdot L_{\text{homo}}^T)^T$$

where $L_{\text{Camera}} \in \mathbb{R}^{N^{camera},4}$ represents the points in the camera view. Next, using the camera projection matrix $\mathbf{M} \in \mathbb{R}^{4,4}$, 3D points can be projected to 2D images via:

$$L_{\text{projected}} = \mathbf{M} \cdot L_{\text{Camera}}$$

retaining the first two columns in $L_{\text{projected}}$, yeilding $L_{\text{projected}} \in \mathbb{R}^{N^{camera},2}$. According to the image width W and height H, the final points are kept and assigned corresponding classes. Notably, due to the slight errors in the extrinsic calibration parameters $\mathbf{T}^{\text{LiDAR,Camera}}$ and camera projection matrix $\mathbf{M} \in \mathbb{R}^{4,4}$, misalignment may occur, especially for distant points. More details about this will be discussed in chapter 5. Therefore, a 25 m threshold is set for using semantic information from the camera as labels. Consequently, the labels for the points close to the sensors are assigned to L_{reduced}, while the distant points retain their original labels.

Although we integrate the semantic information from two sensors and concentrate on the central view targets, there are still imperfect annotations. For example, a small bounding box may result in partial car mislabeling. To address this, label smoothing is important. We apply the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm on L_{reduced} to identify clusters in the PCs

[51]. In each cluster, the labels of PCs will be assigned to the same class based on majority voting. DBSCAN depends on two key parameters: ϵ (the maximum distance between points) and $minPts$ (minimum number of points in a cluster). After visualizing the cluster results to balance these two parameters, we set $\epsilon = 0.6$ and $minPts = 100$ to detect small clusters and distinguish closely located targets. Finally, PCs with refined labels $L_{\text{final}} \in \mathbb{R}^{N',4+C}$ are generated.

The calibration process can be summarized in the following pseudo-code:

Algorithm 1 Calibration process

Inputs:

- LiDAR PCs with rough labels $L_{\text{reduced}} \in \mathbb{R}^{N,4+C}$ with N' points and C classes
- RGB image $I \in \mathbb{R}^{W,H}$ with W for width and H for height
- Homogeneous transformation matrix $T \in \mathbb{R}^{4,4}$
- Camera projection matrix $M \in \mathbb{R}^{4,4}$
- Number of classes C

Output:

- LiDAR PCs $L_{\text{final}} \in \mathbb{R}^{N',4+C}$ with N' points and C classes

$S_{\text{semantic}} = \text{OneFormer}(I, C)$ ▷ $S \in \mathbb{R}^{W,H,C}$
for $l \in L_{\text{reduced}}$ **do**
  $l_{\text{image}} = \text{PROJECT}(M, T, l_{xyz})$ ▷ $l_{\text{image}} \in \mathbb{R}^2$
  $s = S_{\text{semantic}}[l_{\text{image}}[0], l_{\text{image}}[1], :]$ ▷ $s \in \mathbb{R}^C$
  $l_{\text{Calib}} = \text{Concatenate}(l, s)$ ▷ $l_{\text{Calib}} \in \mathbb{R}^{4+C}$
end for
$L_{\text{final}} = \text{DBSCAN}(L_{\text{Calib}}, \epsilon, \text{minPts})$ ▷ $\epsilon = 0.6, \text{minPts} = 100, L_{\text{final}} \in \mathbb{R}^{N',4+C}$

2.2.3. Transformation and voxelization

To obtain the final labels of radar tensors, we rotate and translate the LiDAR PCs to align with the radar view. Since the radar data is structured as a cube in polar coordinates, it is infeasible to directly use the non-uniform LiDAR PCs in Cartesian coordinates as the labels. Thus, we voxelize the LiDAR PCs L_{final} to convert the PCs into a cube with the dimension $N_R \times N_A \times N_E$. In the voxelization process, the range axis is uniform, with each range cell size equal to $0.1004m$, while a non-uniform voxelization is applied in the azimuth and elevation axis. Consequently, the side bins in the azimuth and elevation axes are sparse, and the central bins are dense, leading to the preservation of more central targets compared to the side targets.

During voxelization, majority voting determines the label for each voxel. If a voxel is empty, it is assigned a value of 0. Other labels are assigned as follows: 1 for 'scenario objects', 2 for 'pedestrians', 3 for 'vehicles', and 4 for 'bicycles'. The final labels have the same timestamp as the LiDAR frames. The voxelization results are highly sparse, with over 99% values being 0. Additionally, the class proportions are unbalanced. The proportions for each class in 7 representative scenes excluding 'background' are shown in Figure 2.4. Visualization of the labels and quantitative evaluation will be shown in chapter 4.

2.3. Motion state labels generation

After exploring the labels across various scenarios, it can be observed that the number of vehicles is significantly higher than that of pedestrians and bicycles, leading to more effective segmentation for vehicles. Among these vehicles, a large portion is static. Thus, whether static vehicles are segmented as 'vehicles' or 'scenario objects' has minimal practical impact since they are treated as barriers to be avoided. However, assigning both static and dynamic vehicles to a single 'vehicles' class can affect the evaluation of detection performance. Static vehicles are more likely to be misclassified as 'scenario objects' which can reduce the overall segmentation performance for vehicles. In this case, motion state labels are generated to specifically evaluate the detection performance of dynamic vehicles.

To achieve this, and inspired by [52], the motion state of vehicles can be generated from LiDAR PCs in frame n and frame n−1 (without the ground points). The process begins with motion compensation between consecutive frames to account for the ego vehicle movement. Next, the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm is applied to detect clusters

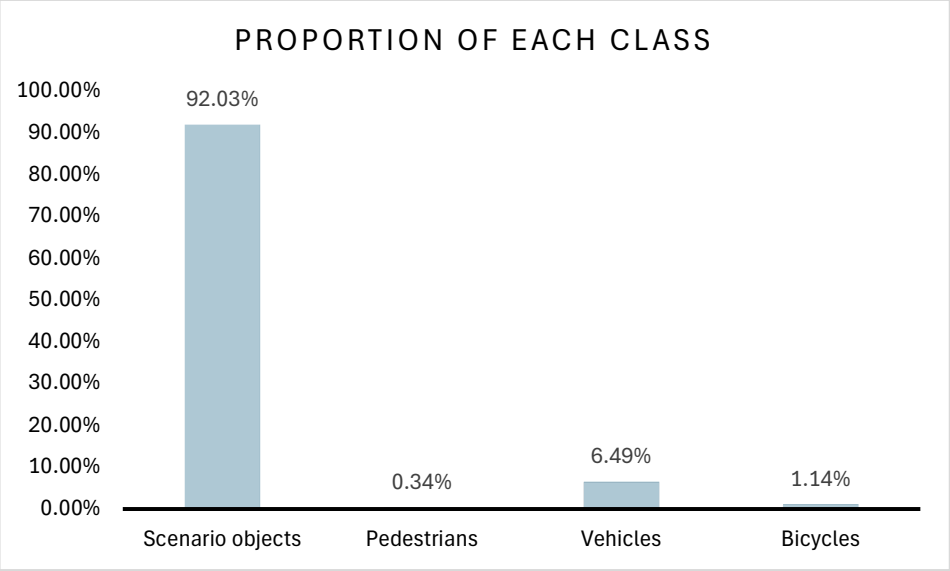

Figure 2.4: Proportion of each class except the background. The proportions of each class are highly unbalanced. Scenario objects class, such as buildings and vegetation, have the largest share due to their large size, while the pedestrians have the smallest proportion for their relatively small size.

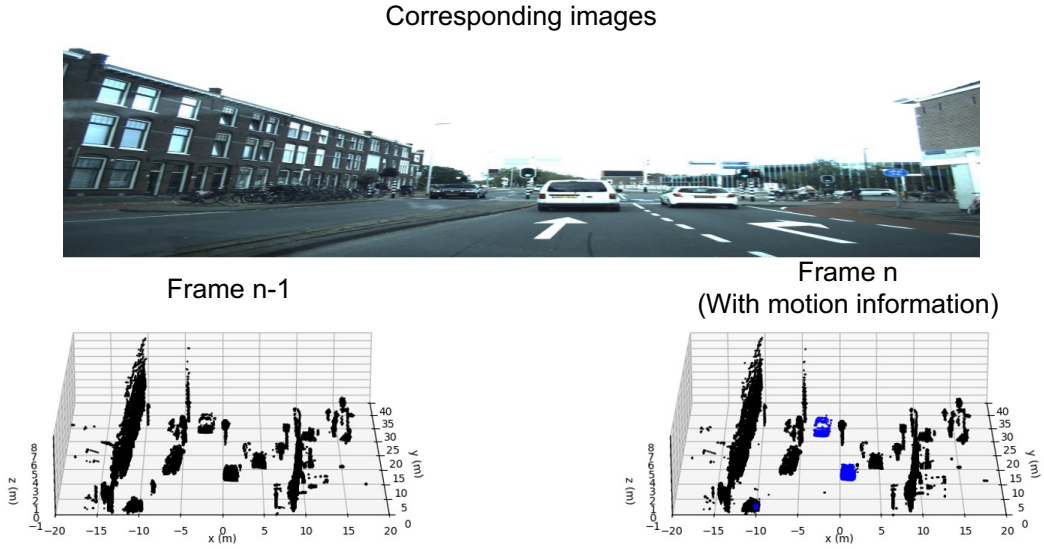

Figure 2.5: Motion state labels generated from two consecutive frames, where the dynamic vehicles are highlighted in blue

in both frames. After cluster identification, the clusters are matched using the method proposed by [52], which employs several metrics: average translation distance between clusters, intersection over union (IoU) of bounding boxes, inliers ratio, and rotation angle. The Hungarian matching algorithm is applied to pair the clusters across frames based on these metrics.

Given that the LiDAR PCs have a 360° FoV and a longer detection range than necessary, this process ensures that no targets appear in frame n but are missing in frame n−1. To identify vehicles moving faster than 18 km/h, a threshold of 0.5 meters is applied to the average distance between matching clusters, based on the 0.1-second interval between consecutive LiDAR frames. Although this threshold may result in the loss of labels for vehicles moving at very low speeds, it effectively reduces the risk of mislabeling.

Figure 2.5 demonstrates the motion state evaluation in a complex scenario. In the reference image, the white car in front of the ego vehicle and the black car opposite are dynamic, moving due to the green traffic light. In contrast, the white car on the right is static, waiting for the right-turn signal. In the lower-right image, the dynamic vehicles are accurately highlighted in blue, demonstrating the effectiveness of the motion state labeling in distinguishing between dynamic and static vehicles.

3

Proposed semantic segmentation network

After generating the labels for the radar RAED tensors in the RaDelft dataset as described in the previous chapter, a deep-learning network is constructed for semantic segmentation and presented in this chapter. The inputs to the network are RAED tensors with dimension $N_C \times N_D \times N_A \times N_R (2 \times 128 \times 240 \times 500)$*, and the voxelized PCs with dimension* $N_R \times N_A \times N_E (500 \times 240 \times 34)$ *are used for supervision. The network outputs are expected to match the label shape, representing voxelized PCs in the polar coordinates with class information generated by radar. These tensors can be further transformed into Cartesian coordinates for visualization and evaluation.*

This chapter first explains the radar data pre-processing in section 3.1, followed by the architecture of the proposed deep learning network in section 3.2, which is specially designed based on the input structure. Finally, section 3.3 discusses the choice of loss function and training settings.

3.1. Data pre-processing

The original RAED tensor from the *RaDelft* dataset has dimensions of $N_C \times N_D \times N_A \times N_R (2 \times 128 \times 240 \times 500)$, consisting of two channels: one with the power values, and the other one with elevation values from 1 to 34 corresponding to the highest power return bin. The elevation information is embedded in the second dimension, making it less intuitive to use. Additionally, the ground truth generated from LiDAR lacks Doppler information, providing only target locations and classes. To address this, the RAED tensor is compressed into an RAE tensor with dimensions $N_R \times N_A \times N_E (500 \times 240 \times 34)$, representing range, azimuth, and elevation respectively. In this transformation, each voxel in the RAE tensor represents the average of the power values across the Doppler bins and corresponding elevation bins in the RAED cubes. Specifically, for the first range and azimuth bin, there are 128 power values and 128 corresponding elevation bins in the two channels of the original RAED tensors. In the processed RAE tensors, the value at the first range, azimuth, and elevation bin is calculated as the average of all power values where the corresponding elevation bins indicate '1'.

After data preprocessing, the RAE tensor matches the label dimensions, simplifying the data processing in the deep-learning network. The RAE tensor encodes power returns. When a dynamic vehicle exists in the FoV, it is expected to be extremely powerful in some nearby voxels at the corresponding location, serving as useful features for detection or classification. However, over 99% of the RAE tensor's voxels are non-empty, adding significant noise compared to the voxelized LiDAR labels, which are 99% sparse. This disparity complicates the semantic segmentation task.

3.2. Semantic segmentation structure

General semantic segmentation on the 2D RGB image uses input dimensions of $N_H \times N_W \times 3$, representing the height, width, and three color channels (red, green, blue). The segmented output has dimensions $N_H \times N_W \times C_{\text{image}}$, where C_{image} represents the number of classes. In radar semantic segmentation, the preprocessed input has dimensions $N_R \times N_A \times N_E$, with the labels sharing the same dimensions with values ranging from 0 to 4, representing five classes in polar coordinates. The main differences between radar and image data are the use of polar coordinates instead of Cartesian coordinates and the increased sparsity in the outputs. This structure indicates the suitability of semantic segmentation for radar data.

Starting from the RAE tensor, the proposed semantic segmentation network begins with a **2D backbone** to generate two separate feature maps shown in Figure 3.1. The first branch uses a 2D Feature Pyramidal Network (FPN) [53] structure with a ResNet-18 backbone to generate a 3D occupancy latent space with the dimension $N_R \times N_A \times N_E$. FPN fuses multi-scale features from different levels, and the 18-layer residual block structure in Resnet-18 addresses the vanishing gradient problem by enabling more effective training of deeper models [54]. In this tensor, values represent the probability of a target presence at specific locations.

The second branch also applies a 2D FPN structure with a ResNet-18 backbone but outputs a class latent space with dimensions $C \times N_R \times N_A$. In this work, $C = 5$ represents 4 classes of interest and one empty class. Notably, the parameters in these two branches are independent. This feature map can be interpreted as a bird's-eye view containing class information.

These two 2D feature maps can be merged using tensor broadcasting. In this process, the 3D occupancy latent space is expanded to dimensions $1 \times N_R \times N_A \times N_E$, while the class latent space is expanded to dimensions $C \times N_R \times N_A \times 1$. These two tensors are then combined to form a 3D semantic information latent space with dimension $C \times N_R \times N_A \times N_E$. However, this latent space is not the final result, as all elevation bins for the same range and azimuth are predicted to belong to the same class. Therefore, an additional 3D backbone is required to refine the predictions across elevations.

Inspired by [55], a 3D U-Net with five layers is designed for this task, shown in **3D Backbone** in Figure 3.1. The 3D U-Net structure is well-suited for dense segmentation, as it combines local features and global context information. The 3D convolutional structure integrates spatial information to achieve accurate 3D semantic segmentation. Finally, 3D semantic segmentation results are generated with dimensions $C \times N_R \times N_A \times N_E$, representing the probability of each class occurring at the corresponding location.

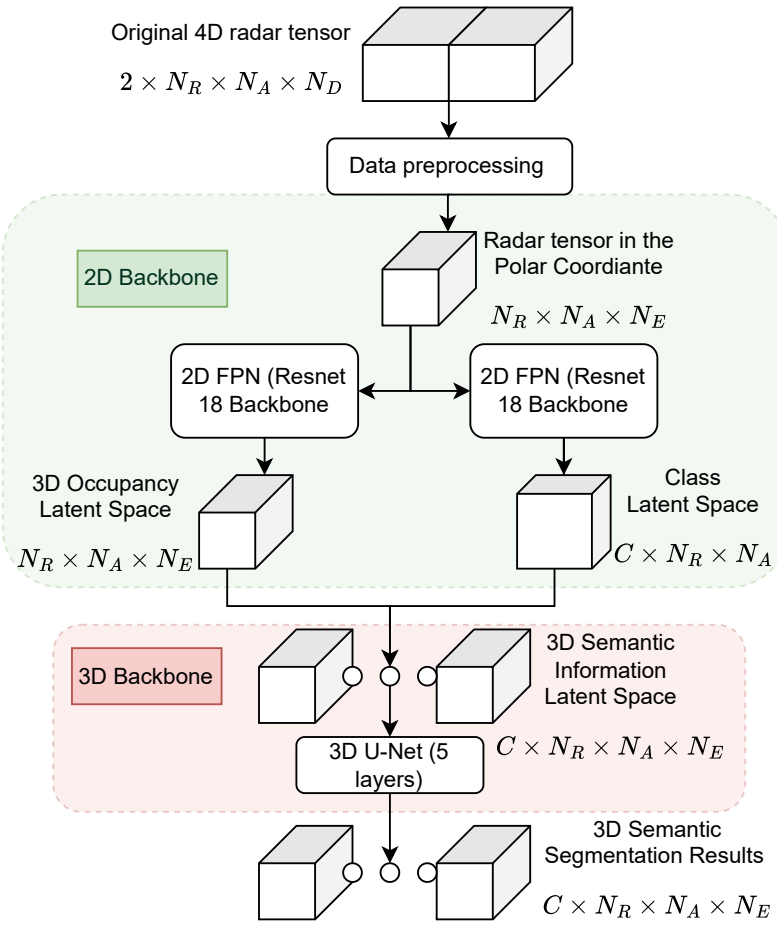

Figure 3.1: Proposed radar semantic segmentation approach. A radar tensor with dimensions $N_R \times N_A \times N_E$ in the polar coordinates is generated after data pre-processing. Next, a 2D backbone with two separate branches is constructed to generate the 3D occupancy latent space and the class latent space, respectively. These latent spaces are then combined through broadcasting to form a new 3D semantic segmentation latent space. Finally, a 3D backbone is used to produce the output, providing the probability of occurrence for each class at every voxel.

This model, combining both the 2D and 3D backbones, is considered as the **baseline**. To further explore performance improvements, several modifications to this baseline have been developed and evaluated to identify the optimal model. The specific modifications are as follows:

- Instead of combining 2D and 3D convolutional networks, a model using only 3D convolutional layers is constructed (**Only 3D**).
- To better integrate 3D information and mitigate vanishing gradient issues in the 3D backbone, a residual structure with additional layers is added to the 3D U-Net (**Baseline + Res**).
- Since consecutive frames often capture similar scenes, similar input values across RAE bins are expected. However, variations exist between consecutive frames. To address this, a causal Gaussian filter is applied after data preprocessing, smoothing inputs across frames (n-2, n-1, and n) (**Baseline + Filter**).
- Inspired by [56], the Convolutional Long Short-Term Memory (ConvLSTM) network, which performs well in radar echo-based precipitation nowcasting, is incorporated into the baseline to integrate temporal information across frames from millimeter-wave automotive radar (**Baseline + LSTM**).

These modifications of the model are trained in the training process and evaluated in subsection 4.2.2.

3.3. Training setting

Considering the unbalanced class proportion and the sparsity differences between the input tensors and the label, a combination of two loss functions is applied in this task. To handle the unbalanced proportion of each class, weighted cross-entropy (wCE) loss is selected to focus more on minority classes and reduce model bias by assigning different weights to each class [57]. Cross-entropy loss predicts the probability of each pixel belonging to a category and calculates the cross-entropy between these probabilities and the actual label. The function of the wCE loss is in the following:

$$\mathcal{L}_{\text{wCE}}(\mathbf{y}, \mathbf{p}) = -\frac{1}{K} \sum_{k=1}^{K} w_k \sum_{(r,a,e)\in\Omega} \mathbf{y}[r, a, e, k] \log \mathbf{p}[r, a, e, k]$$

where $\mathbf{y}$ represents the real label tensor, and $\mathbf{p}$ denotes the predicted probability tensor of the model output. r, a, and e represent the range, azimuth, and elevation bins respectively, and ω_k represents the weight of k class.

The second loss applied is the soft dice (sDice) loss [58]. The dice coefficient is originally designed for the binary data to measure the overlap between two collections, which can be expressed by:

$$\frac{2|A \cap B|}{|A| + |B|}$$

where $|A \cap B|$ represents the common elements. For multi-class segmentation tasks, this parameter can be transformed to the sum of the dice loss of each class, represented by the following equation:

$$\mathcal{L}_{\text{SDice}}(\mathbf{y}, \mathbf{p}) = \frac{1}{K} \sum_{k=1}^{K} \left[1 - \frac{2 \sum_{(r,a,e)} \mathbf{y}[r, a, e, k] \mathbf{p}[r, a, e, k]}{\sum_{(r,a)} \mathbf{y}^2[r, a, e, k] + \mathbf{p}^2[r, a, e, k]} \right]$$

where $\mathbf{y}$ represents represents the real label tensor, and $\mathbf{p}$ denotes the predicted probability tensor of the model output. Notably, although the soft Dice loss is effective for shape segmentation and addressing issues with small foreground classes, it is sensitive to initial values and can affect the stability of gradient descent. To mitigate this, a combination of weighted cross-entropy (wCE) loss and soft Dice (sDice) loss is necessary, balancing the strengths of both losses to improve training stability and performance on unbalanced data.

To better combine these two types of loss, inspired by [59], an auxiliary task is introduced, which is trained alongside the main task. In this auxiliary task, the weight between the two loss functions is dynamically updated after each epoch. The final combined loss function after training is as follows:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{wCE}}(\mathbf{y}, \mathbf{p}) + 2.5 \times \mathcal{L}_{\text{SDice}}(\mathbf{y}, \mathbf{p})$$

where the final weight for wCE loss is 1, and the weight for soft dice loss is 2.5. Additionally, the quantitative results will compare the performance of four approaches: using only wCE loss, using only sDice loss, using an unweighted combination of both losses, and using the dynamically adjusted weighted combination of these two losses. These results will be demonstrated in the result parts.

The semantic segmentation model is trained on the *RaDelft* dataset. Five of the seven scenes are used to train and validate, while the remaining two unseen scenes are reserved for testing. This setup ensures a robust evaluation of the model's performance. Among the training and validation data, 80% is used for training, and the remaining 20% is used for validation.

The network is trained using the DelftBlue Supercomputer [60] on an NVIDIA Tesla A100 80GB GPU with an Intel Xeon E5-6448Y CPU. Detailed training information, including training time for 50 epochs, the number of model parameters, and model size, is summarized in Table 3.1 for the baseline model and its modifications.

Model Name	Training time (h)	Parameters (M)	Model size (MB)
Baseline	16.7	26.6	106

Model Name	Training time (h)	Parameters (M)	Model size (MB)
Baseline + Res	23.3	33.4	133
Baseline + Filter	15.8	26.6	106
Baseline + LSTM	25.1	33.8	135
Only 3D	35.8	28.4	113

Table 3.1: Training details of the proposed baseline network and its several modifications

From Table 3.1, it can be observed that the ‘Baseline + Res’ model has the most parameters while maintaining an acceptable training time. In contrast, the ‘Only 3D’ model has the longest training time although not have too many parameters. This is due to the increased complexity of 3D convolution operations and their backward propagation. Additionally, the ‘Baseline + LSTM’ model has the largest number of parameters because of the extra structure required to process the hidden states from previous frames.

4

Results

This chapter first presents the results of the proposed automatic labeling process in section 4.1, with step-by-step labeling results for better visualization. Manual labeling results are provided to compare with the automatic labeling results from both quantitative and qualitative perspectives. Then, the semantic segmentation results with the proposed network are demonstrated in section 4.2 and compared with the ground truth. The proposed semantic segmentation network is specifically compared with several variants inspired by the literature, and some modifications to the baseline model. Very promising results are shown in terms of the probability of detection and Chamfer distance with the proposed approach.

4.1. Results of designed automatic labeling process

This section provides at first detailed visualization results of each step in the proposed automatic labeling process, where the operations in each step were described in Chapter 2. In steps 1 and 2, preliminary object labels are generated in the LiDAR coordinate system, while final labels appear in the radar view after coordinate transformation and voxelization, leading to differences in angle and position between steps 1, 2, and 3. Quantitative results then compare the performance of automatic and manual labeling. To illustrate each step and its effects, an example from the same complex scenario is provided.

4.1.1. Preliminary object label generation results

Preliminary object labels are generated before removing the ground using 'Patchwork++' in the LiDAR PCs. However, visualizing the target labels is challenging due to the dense ground clusters. Additionally, the original LiDAR PCs have a 360° FoV, making it hard to visualize the entire scene. To improve visualization, the ground is removed, and the data is cropped to fit the radar field of view (FoV). Although cropping is performed in the radar coordinate system, the data is transformed back to the LiDAR coordinate for further processing. Figure 4.1 shows the cropped LiDAR PCs after ground removal and radar view fitting.

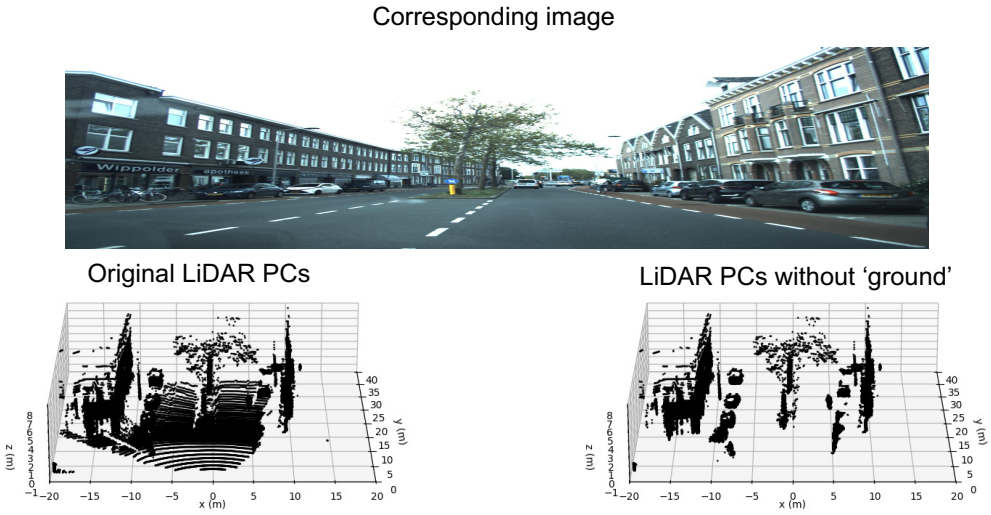

Figure 4.1: Comparison of original LiDAR PCs versus ground-removed LiDAR PCs, cropped to align with the radar FoV in a complex scenario. The LiDAR PCs are subsequently transformed back to the LiDAR coordinate system after cropping.

As shown in Figure 4.1, 'Patchwork++' effectively removes the ground. When using the pre-trained anchor-free 'Part A2' model for object detection, bounding box parameters for 'vehicles', 'pedestrians', and 'bicycles' are generated. After transforming these parameters to the Cartesian coordinate system, Figure 4.2 displays the schematic diagram of the bounding boxes for the same complex scenario, along with the point-wise labels and corresponding reference RGB images.

Although the generated point-wise preliminary labels are generally accurate, certain cases still affect label quality. For example, in the lower-left part of Figure 4.2, multiple closely parked bikes are not correctly bounded. This occurs because the pre-trained object detection model is not tailored to our dataset, making it difficult to differentiate the closely parked bikes from ground barriers.

In addition to closely parked bicycles, vehicles with high roofs also present difficulties in generating accurate bounding boxes due to the limited elevation range of LiDAR mounted on a standard small vehicle. As a result, the tail section of tall vehicles may be incorrectly mislabeled. Figure 4.3 demonstrates an example of this case, where the yellow box partially bounds the vehicle's tail, leading to incomplete labels. This issue also arises with buses or trucks.

Additionally, when a target is partially occluded by vegetation or other objects, the corresponding LiDAR PCs are incomplete, making it more difficult to generate accurate bounding boxes. Due to these limitations in the preliminary object labels, the calibration process is crucial for improving label quality.

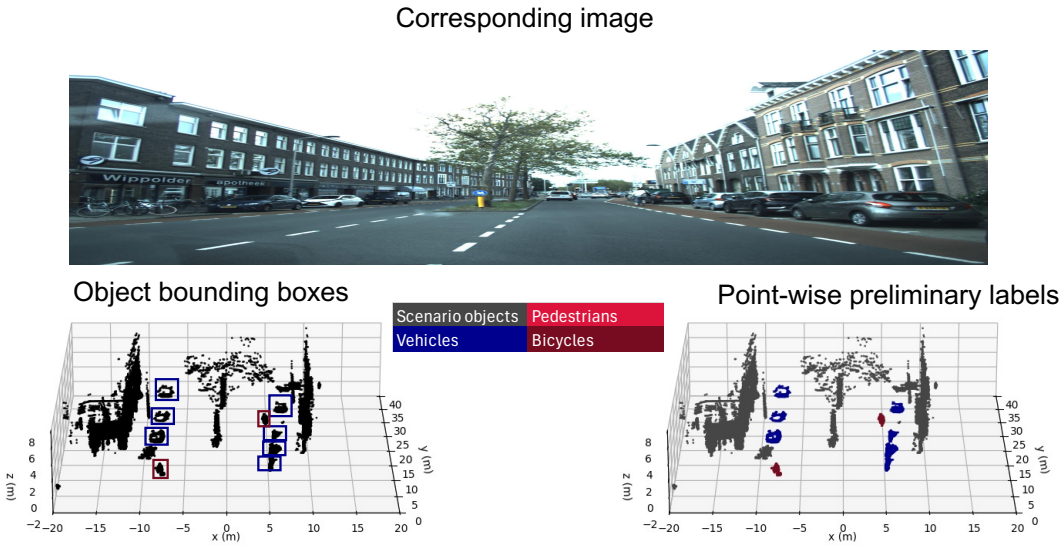

Figure 4.2: Generated object bounding boxes and point-wise preliminary labels for the same complex scenario shown in Figure 4.1. While the labels are generally accurate, there are imperfections in cases involving closely parked bicycles. The color bar indicates the four labeled classes.

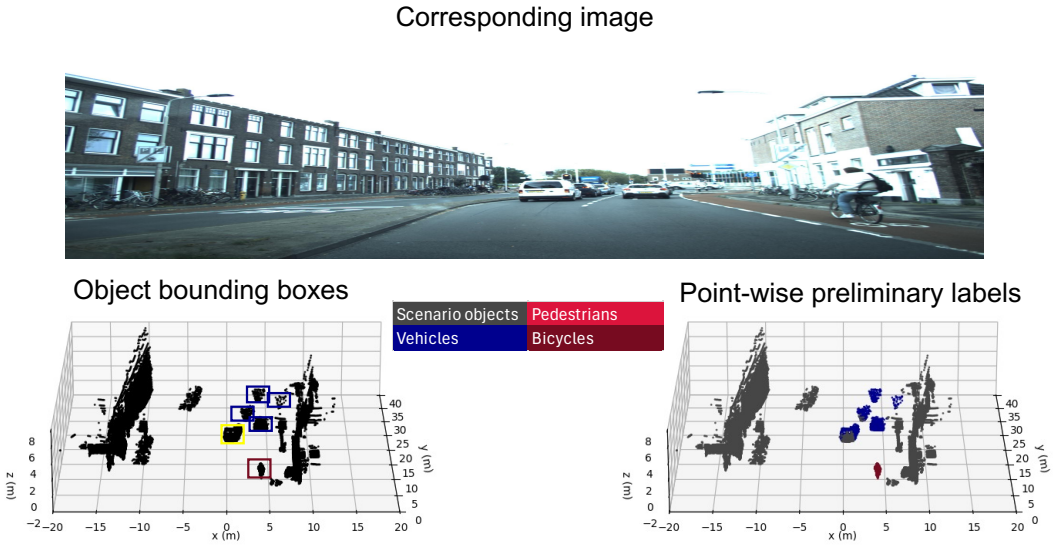

Figure 4.3: Generated object bounding boxes and point-wise preliminary labels for a different scenario. Labels are generally correct, but not perfect for closely parked bicycles and tall-roof vehicles.

4.1.2. Calibration results

The class information generated in the first step is solely from the LiDAR PCs. With the help of synchronized RGB images and advanced image semantic segmentation methods, those images also contribute semantic information to the point cloud. By using transformer-based pre-trained models for image segmentation, as explained in chapter 2, Figure 4.4 illustrates the segmentation results.

Figure 4.4: The original optical image and the corresponding semantic segmentation result for the same complex scenario shown in Figure 4.1. As it can be observed, nearly every pixel is correctly segmented

In Figure 4.4, nearly perfect segmentation is achieved. To utilize the semantic information of each pixel in the RGB images, the LiDAR PCs are projected onto the RGB images. A detailed explanation of the underlying theory is provided in Subsection 2.2.2. It is important to note that, since the transformation matrix between the LiDAR and the camera is known, the LiDAR PCs must be cropped to match the camera's field of view, which is limited to $\pm 34^{\circ}$. Additionally, the distance is constrained to 50 meters, aligning with the radar's effective range. Figure 4.5 illustrates the visualization result when the LiDAR PCs are projected onto the segmented images and assigned to the corresponding pixels.

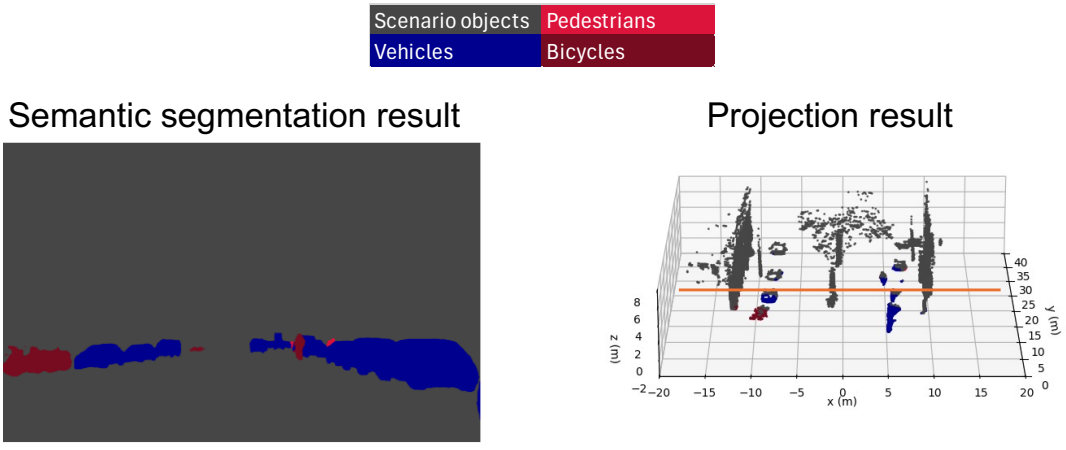

Figure 4.5: The semantic segmentation result and the projection of the LiDAR PCs onto the segmented image. Targets beyond the orange line (approximately 25 m) are not correctly labeled.

It can be observed that while the semantic segmentation result on the image is generally accurate, the labels generated by projecting LiDAR PCs onto the segmented images are less reliable, particularly for distant targets. This issue can be attributed to three factors:

- **Imperfect segmentation of distant targets in RGB images**: Comparing the semantic segmentation with the original images reveals that segmentation accuracy decreases for distant targets,

leading to misclassification of the projected PCs.

- **Timestamp differences between LiDAR PCs and RGB images**: Although the RGB images and LiDAR PCs are synchronized, there are slight timestamp differences. For instance, in scene 2 of the dataset, the time difference between the image and the LiDAR PCs ranges from 3.3 ms to 7.3 ms, with only 876 of the 1700 scenes having a time difference below 4 ms. This discrepancy affects the alignment when projecting LiDAR PCs onto images for semantic labeling.
- **Extrinsic calibration errors**: Even in consecutive frames where the vehicle remains stationary, misalignments persist, likely caused by minor errors in the extrinsic calibration parameters between the camera and LiDAR.

To mitigate the aforementioned problems, a 25-meter threshold is set to leverage information from the images. Beyond this threshold, the accuracy of the semantic labels decreases, so combining semantic information from both RGB images and LiDAR PCs is necessary to generate high-quality labels. Additionally, since the camera's FoV is significantly smaller than the radar's, relying solely on camera-based labels is insufficient for complete scene coverage.

After combining the semantic information from the RGB images with the preliminary labels, the result is shown in the lower-left part of Figure 4.6. However, some car roofs are incorrectly labeled. To address this, the DBSCAN clustering algorithm is applied, leading to correct labeling by assigning the dominant class within each cluster to all points. This process is regarded as a form of 'label smoothing' to improve label consistency, and the results are shown in the lower-right part of Figure 4.6.

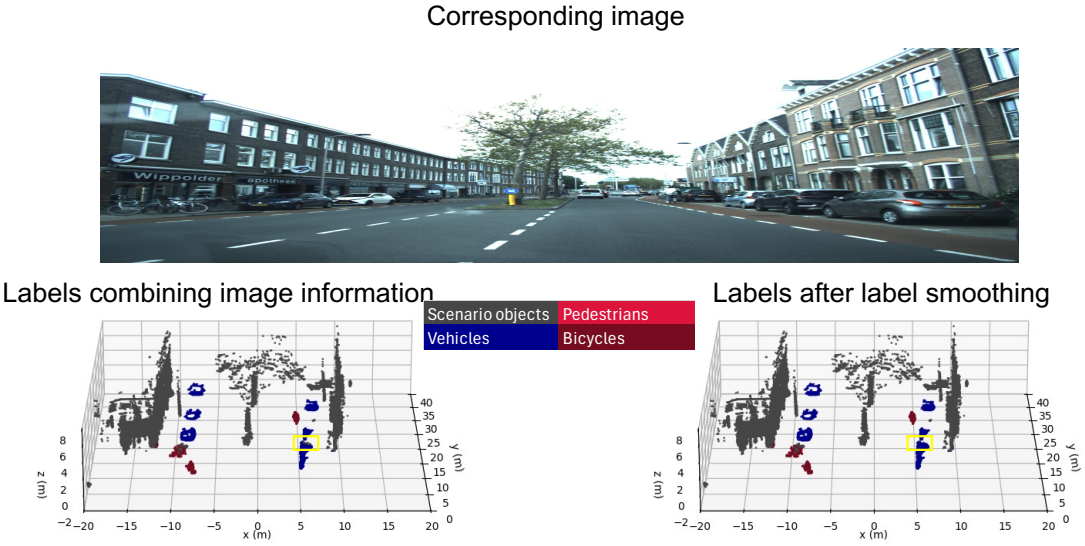

Figure 4.6: Labels combining preliminary point-wise labels and image-based information & labels after label smoothing for the same complex scenario shown in Figure 4.1. In the yellow box, it can be observed that the mislabeling of the car roof is corrected after applying label smoothing.

4.1.3. Transformation and voxelization results (Final labels)

Notably, LiDAR PCs with corresponding labels remain in the LiDAR coordinate and are non-uniform, making them unsuitable for direct use in the radar semantic segmentation task. After translation and rotation, LiDAR PCs in the radar view are shown in the lower-right part of Figure 4.7.

Following this, the location and class information of each point is embedded into a tensor with dimension $N_R \times N_A \times N_E (500 \times 240 \times 34)$ to generate the final labels for radar. Due to the low resolution of radar compared to LiDAR, multiple LiDAR PCs may fall into the same voxel. To ensure each voxel is assigned a unique class, majority voting is applied during this process.

For visualization, this tensor, i.e., data in a cube form in the polar coordinates, is transformed back to the Cartesian coordinates in the point form. In the re-generated PCs, each point is generated from a non-empty voxel in the tensor. The re-generated PCs are sparser than the original LiDAR PCs before

voxelization, while the labels are more accurate since mislabeling in the nearby points is corrected. The ground truth in the Cartesian coordinates is shown in the right bottom part of Figure 4.8.

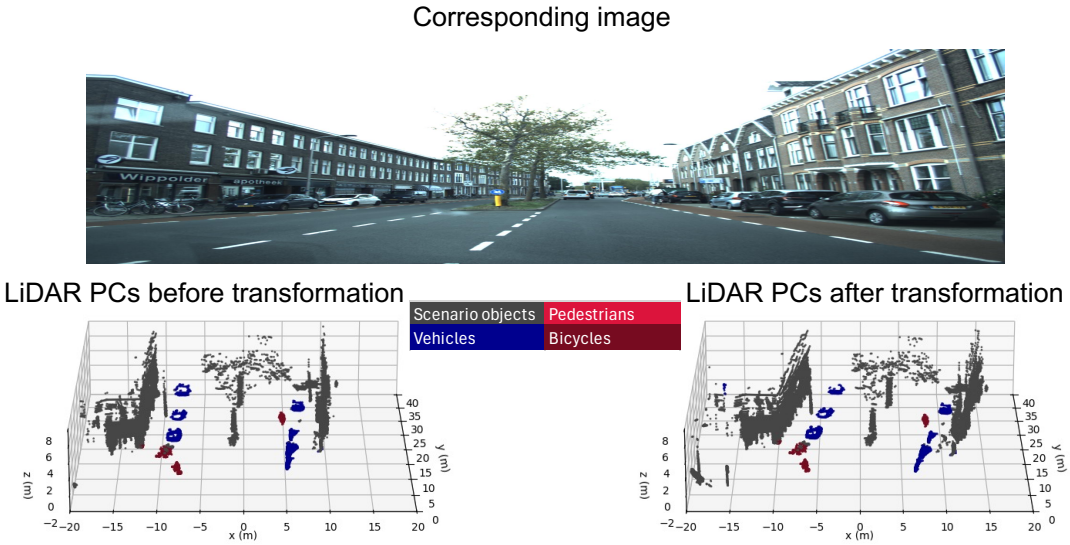

Figure 4.7: LiDAR PCs with point-wise labels in both the LiDAR view and radar view for the same complex scene shown in Figure 4.1. The location of each point has additional translation and rotation to align with the radar view, while the label remains unchanged.

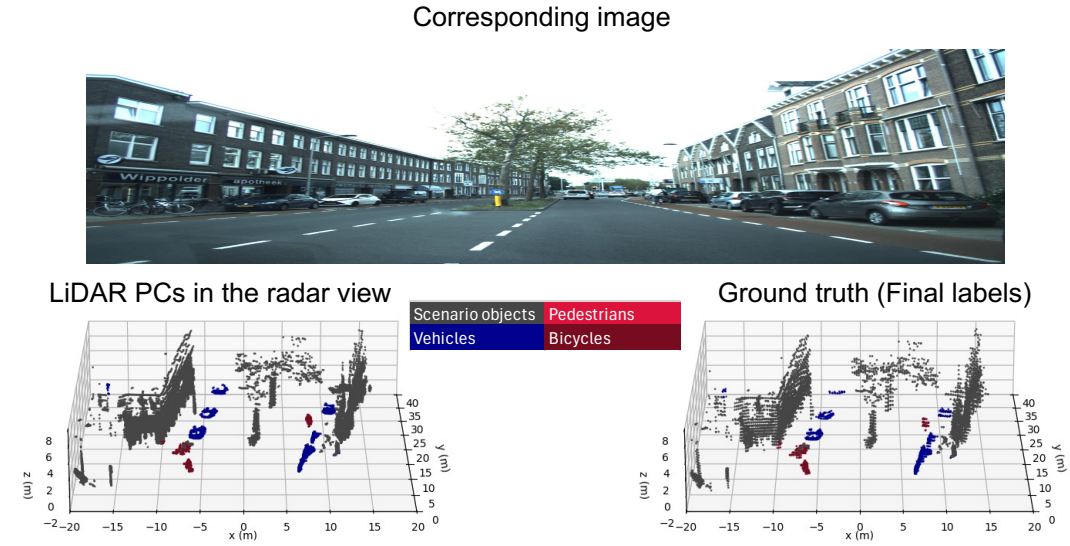

Figure 4.8: LiDAR PCs in the radar view and final labels re-generated from the voxelized LiDAR PCs for the same complex scenario shown in Figure 4.1. Although the PCs in the ground truth are sparser compared to the LiDAR PCs, they maintain the same overall shape.

4.1.4. Automatic versus manual labeling results

To further assess the quality of the generated labels, comparisons between the automatic labeling results and manual labeling results are performed. Since no existing labels are available for the LiDAR PCs, 50 frames from 7 different scenes are randomly selected and manually labeled for comparison with the automatic labeling results. These comparisons were conducted using the regenerated labels derived from the ground truth tensors, which are directly used as labels for the radar semantic segmentation task.

Quantitative results: In order to quantitatively evaluate the performance of the generated labels, three metrics including **Precision**, **Recall**, **F1-score** are used. These metrics are computed for each non-

empty class using the following equations:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

$$\textbf{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where **TP** (True Positive) represents instances where the automatic labeling result matches the manual labeling for a specific class, **FP** (False Positive) represents instances where the model incorrectly labels a different class as the target class, and **FN** (False Negative) represents instances where the model incorrectly labels the target class as another class.

These metrics are computed for each frame, and the average values across the 50 frames are presented in Table 4.1.

Table 4.1: Quantitative metrics of the designed automatic labeling results compared to manual labeling results considered as the ground truth to derive the metrics.

Class	Precision	Recall	F1-score
Scenario objects	0.98	0.99	0.99
Vehicles	0.95	0.81	0.88
Bicycles	0.79	0.87	0.83
Pedestrians	0.67	0.72	0.69

The results indicate that the performance for ‘scenario objects’ and ‘vehicles’ is satisfactory, while the performance for ‘bicycles’ and ‘pedestrians’ is slightly lower, which is expected due to increased difficulty in automatically labeling these targets. The challenges for this arise from several factors:

- **Side-view misalignment**: Since the data is collected from a car driving on a motorized lane, non-motorized lanes, where pedestrians and bicycles are usually located, are often at the side in most frames. This side-view positioning increases the cases of misalignment during the calibration process.
- **Densely packed bicycles**: Closely packed bicycles are difficult to isolate into separate bounding boxes. Even after calibration, this issue persists when bicycles are far from the sensors.
- **Small object size**: The generally small size of pedestrians and bicycles makes them more prone to partial misclassification, making it difficult to generate accurate clusters for the label smoothing process.

Qualitative results: Comparisons between the proposed automatic labeling results and the manual labeling results are shown in Figure 4.9 for the same scenario discussed earlier. The figure demonstrates the labeling results intuitively. Most points are correctly labeled through the automatic process, with the exception of certain points within the green box. These errors are due to the close proximity of parked bicycles, which makes it difficult to generate optimal bounding boxes, and these points exceed the effective range of the calibration process, as previously discussed.

In another example, Figure 4.10 presents a frame where pedestrians are correctly labeled, even when one is standing near a building, demonstrating the effectiveness of the automatic labeling process in such cases. However, the bicycles parked on the sidewalk, highlighted by green boxes, remain incorrectly labeled in this frame, illustrating the ongoing challenge with labeling bicycles in side-view positions or parked closely to each other.

Figure 4.11 provides another example frame for the vehicles not being perfectly labeled. In this frame, part of a parked car in the opposite direction (highlighted with green boxes) is not segmented perfectly.

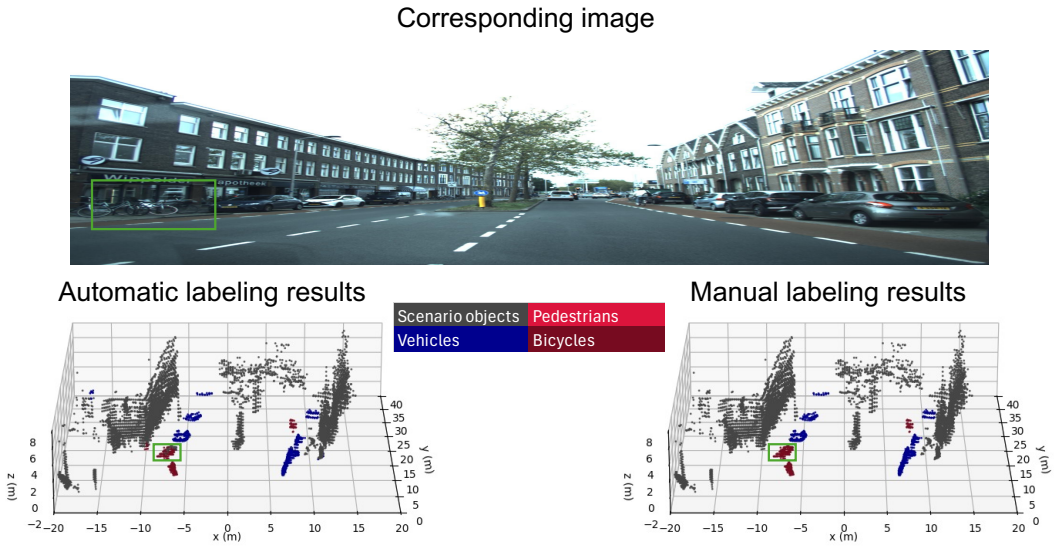

Figure 4.9: Automatic vs manual labeling results for the same complex scene shown in Figure 4.1, with a reference camera image provided. In the green bounding boxes, bicycles parked on the sidewalk are incorrectly labeled. The color bar differentiates the four labeled classes.

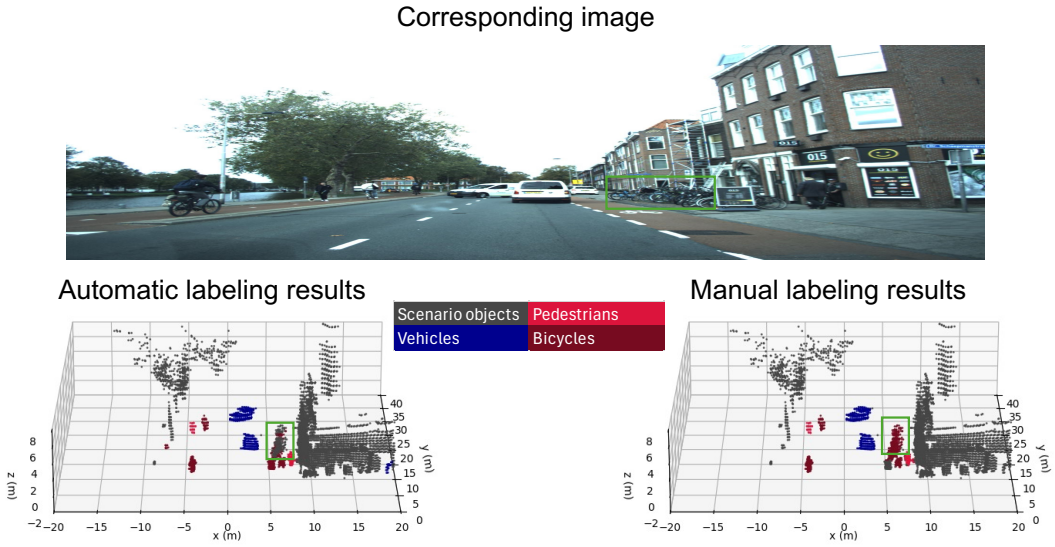

Figure 4.10: Automatic vs manual labeling results for another scene where the automatic labeling performs well for pedestrians, with a reference camera image provided. The color bar distinguishes the four labeled classes.

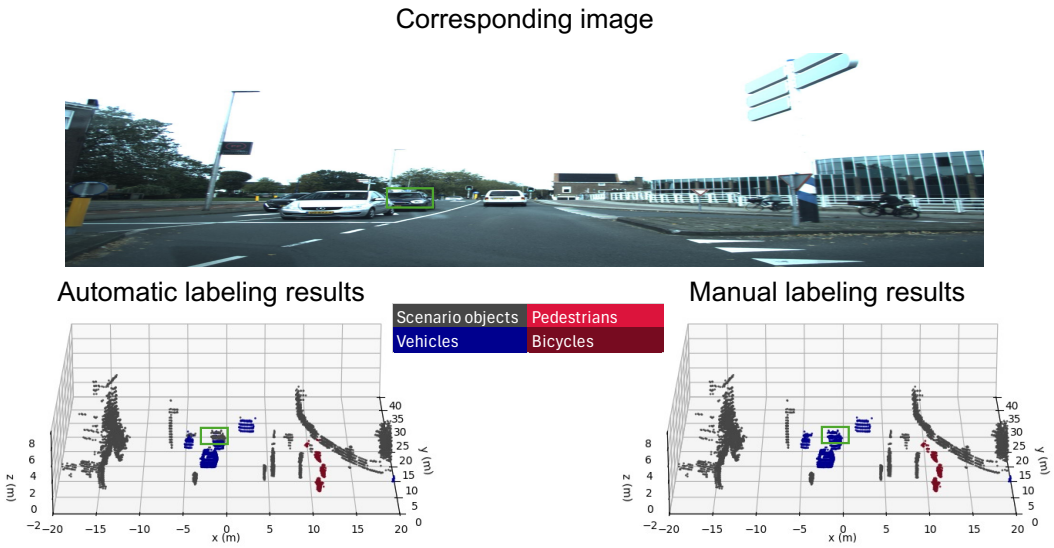

Figure 4.11: Automatic vs manual labeling results for another scene where the automatic labeling is imperfect for vehicles, with a reference camera image provided. The color bar differentiates the four labeled classes.

In summary, while the proposed automatic labeling process demonstrates overall strong performance, it is not as precise as manual labeling. However, it significantly reduces labeling costs and time and addresses potential issues to a large extent. After analyzing multiple frames, the imperfect labeling typically occurs in the following scenarios that should be double-checked:

- **Closely parked bicycles on the sidewalk**: These are often misclassified as scenario objects due to their larger-than-normal shape when parked closely.
- **Vehicles with tall roofs at a distance**: Vehicles with tall roofs may not be perfectly enclosed by the bounding box, and when located far from the sensors, they can exceed the effective range of camera calibration, leading to partial misclassification.
- **Motorbikes or bicycles with large rear boxes**: The pre-trained model used for preliminary object detection is not specifically designed for the LiDAR in the *RaDelft* dataset, occasionally resulting in imperfect bounding boxes for these targets.
- **Pedestrians far from the sensors**: Distant pedestrians may be ignored in the labeling process due to their small size in the LiDAR PCs, where only a few points are detected by the LiDAR.

4.2. Semantic segmentation results

To evaluate the performance of the semantic segmentation with the proposed network, both quantitative and qualitative results are presented in this section. For the quantitative evaluation, two implementations of existing methods in the literature are compared against the proposed method. For the qualitative assessment, the radar semantic segmentation results are compared with the ground truth and corresponding images to visually analyze the model's performance.

4.2.1. Qualitative results

The predicted tensor with dimensions $C \times N_R \times N_A \times N_E$ is first processed using the Softmax function. For each location, the class with the highest probability is selected, resulting in a 3D tensor with dimensions $N_R \times N_A \times N_E (500 \times 240 \times 34)$, where values range from 0 to 4 representing different classes. Applying the same visualization strategy as applied to the automatic labeling results in voxelized form, the segmented radar PCs are recovered by transforming the polar coordinates into Cartesian coordinates for visualization.

Figure 4.12 illustrates a complex frame of generated radar PCs with class information, the corresponding LiDAR PCs ground truth, and the reference RGB images. The results indicate that the generated radar PCs successfully detect most targets within the FoV. Close vehicles are accurately segmented,

and dynamic vehicles bounded by the purple box are successfully segmented as well benefiting from the radar's Doppler information. For static bicycles on the left side bounded by the orange box, some are correctly segmented into their class, whereas others are misclassified as 'vehicles' or 'scenario objects' due to their dense arrangement. Two cars waiting at the traffic lights, located in front of the white car and bounded by the blue box, are classified as 'scenario objects'. This misclassification arises due to insufficient power values in the RAE radar tensor after pre-processing, as static objects tend to have low power values in the corresponding voxels. These low values are often interpreted as noise, leading to their misclassification as 'scenario objects'.

Figure 4.13 demonstrates another scenario where the ground truth is not entirely accurate, particularly for the bicycles bounded by the green boxes. Despite this, the generated radar PCs still achieve good segmentation performance for the bicycles. This indicates that although the generated labels are imperfect, it is valuable to train and use semantic segmentation networks. The network correctly segments parked bicycles and vehicles near the sidewalk, though some vehicles may be misclassified as bicycles or pedestrians.

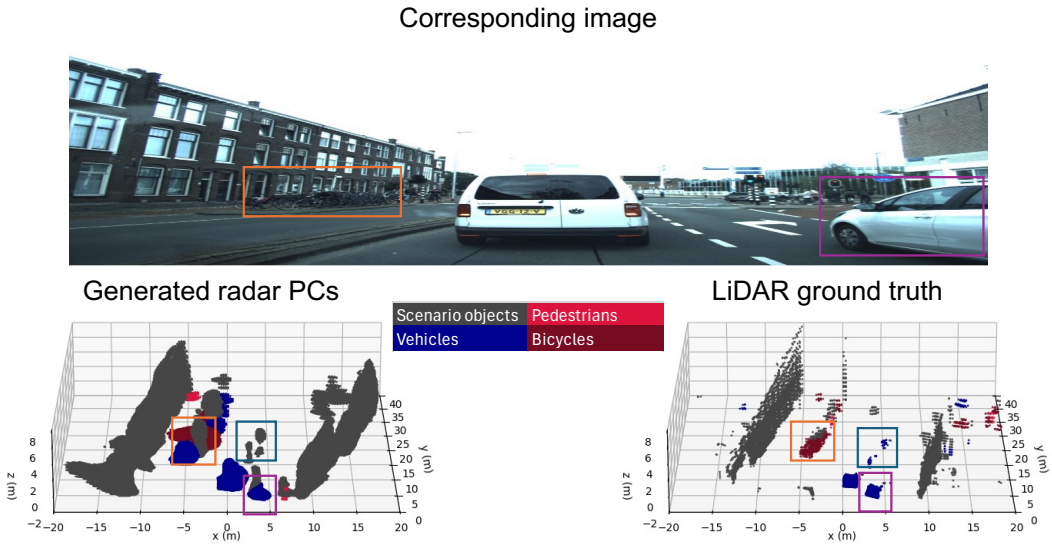

Figure 4.12: Generated radar PCs with class information and LiDAR ground truth are presented for a complex scene, where dynamic vehicles are successfully segmented. A corresponding RGB image is provided for reference, with the color bar consistent with the one used in the labeling process throughout the thesis.

In summary, after analyzing the visualization results qualitatively, radar demonstrates strong performance in detection targets in 3D space, while the segmentation performance is limited especially for static targets or distant objects. The following errors frequently occur in multiple scenarios:

- Static targets are often misclassified as scenario objects.
- Pedestrians and bicycles with low velocity, especially when located closely together, are frequently misclassified as vehicles due to their not typical shape and limited Doppler information.
- Tall poles or side scenario objects may be missed due to the layout and positioning of the radar and camera sensors.

4.2.2. Quantitative results

From the visualization results of segmented radar PCs transformed from the output tensors, radar PCs are noisier than LiDAR PCs due to the lower spatial resolution and the multipath effect caused by reflections from targets and/or ground. However, radar has the advantage of detecting optically occluded objects, which LiDAR cannot perceive, leading to much denser radar PCs than LiDAR PCs. As a result, traditional semantic segmentation metrics like Intersection over Union (IoU) are not suitable, since the union of radar and LiDAR PCs would be significantly larger than their intersection. Even if the radar PCs successfully segment objects, the size of the segmented radar PCs may be larger, making IoU an unreliable metric for evaluation.

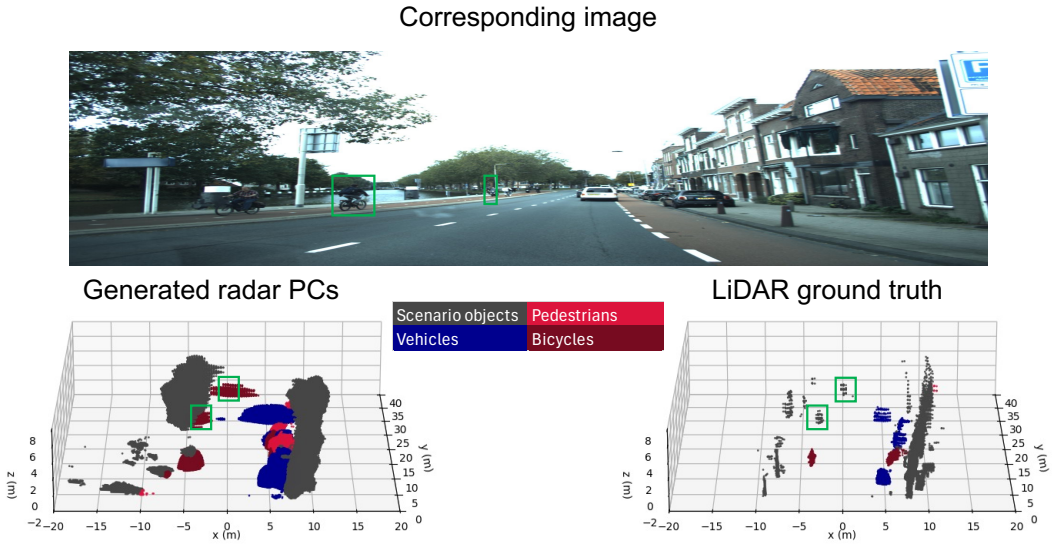

Figure 4.13: Generated radar PCs with class information and LiDAR ground truth are presented for a complex scene, where bicycles are successfully segmented. A corresponding RGB image is provided for reference, with the color bar consistent with that used in the labeling process throughout the thesis.

To simultaneously evaluate detection and segmentation performance, we calculate instead the detection probability (P_d) and false alarm rate (P_{fa}) for all classes and for each individual class. Notably, we can group the 'pedestrians' and 'bicycles' classes into a single class to assess the detection and segmentation performance of vulnerable road users (VRUs). These two metrics are computed directly from the voxelized data.

Additionally, the Chamfer distance (CD) between the predicted results and the ground truth is computed for all classes, 'scenario objects' only, and all targets to measure the similarity between their PCs using the following equation:

$$CD(S_{1_k}, S_{2_k}) = \frac{1}{|S_{1_k}|} \sum_{x \in S_{1_k}} \min_{y \in S_{2_k}} \|x - y\|_2 + \frac{1}{|S_{2_k}|} \sum_{y \in S_{2_k}} \min_{x \in S_{1_k}} \|x - y\|_2$$

where S_{1_k} and S_{2_k} denote the predicted and ground truth point sets for class k. A lower Chamfer distance indicates higher similarity between the two sets. This distance is calculated after transforming the voxelized data into Cartesian coordinates in point format.

To comprehensively assess the model performance, comparisons with other existing methods are necessary. However, to the best of the author's knowledge, no multi-class semantic segmentation method exists that uses only 4D radar tensors as input and LiDAR PCs as ground truth in 3D space; hence, direct comparisons are challenging. As an alternative, two adapted methods from the literature are implemented and trained on the *RaDelft* dataset for comparison:

- **Variant 1**: The first adapted method is based on the network proposed by [41]. This original work designs a data-driven detection network to generate high-quality radar PCs that are closely similar to LiDAR PCs. The network includes a DopplerEncoder, a backbone, and a temporal coherence network, with the input data from the *RaDelft* dataset consisting of two channels (one elevation axis and one power axis). To modify this model for semantic segmentation, the backbone is adjusted to output tensors with dimensions $C \times N_R \times N_A \times N_E$, creating the version referred to as 'Variant 1'.
- **Variant 2**: The second adapted method is based on the network proposed by [39]. In their original work, sparsified radar RAE tensors are used as inputs for 3D object detection. Their network comprises pre-processing, a 3D sparse convolutional backbone, and a neck & head structure. To adapt this model for semantic segmentation network, the neck & head structure is replaced with a segmentation head to produce outputs with dimensions $C \times N_R \times N_A \times N_E$. Additionally,

the input tensors are sparsified, retaining only the top 10% of power return bins, constructing the model referred to as 'Variant 2'.

In addition to the two aforementioned variants inspired by the literature, specific modifications according to the proposed **baseline** include **Only 3D**, **Baseline + Res**, **Baseline + Filter**, and **Baseline + LSTM**, as described in Chapter 3. These are also trained and evaluated to compare the results of the proposed semantic segmentation network. Additionally, for each network, we also train a version where 'pedestrians' and 'bicycles' are merged into a 'VRU' class where the output channel decreases from 5 to 4. This modification is based on the fact that 'pedestrians' and 'bicycles' share similar size and often velocity characteristics. Given the radar's limited resolution and reliance on Doppler information as a key feature, distinguishing between these two classes can be challenging. By merging them into a 'VRU' class, the networks are expected to achieve better segmentation performance for VRUs and reduce the complexity of segmenting other classes.

Table 4.2 presents the metrics averaged across all test data for the aforementioned networks. The metrics applied are listed in the following:

- $P_{d_{\text{All}}}$: Detection probability for all points.
- $P_{d_{\text{Scenario}}}$: Detection probability for scenario objects.
- $P_{d_{\text{Vehicles}}}$: Detection probability for vehicles.
- $P_{d_{\text{VRU}}}$: Detection probability for VRUs.
- $P_{fa_{\text{All}}}$: False alarm rate for all points.
- CD_{All}: Chamfer distance for all points.
- CD_{Targets}: Chamfer distance for all targets except scenario objects.

Table 4.2: Performance of the proposed semantic segmentation network vs alternative variants inspired from the literature. Results in () are for versions of each network trained with the classes 'pedestrians' & 'bicycles' combined into the 'VRU' class. The best performance for each metric is marked in bold.

Methods	$P_{d_{\text{All}}}$	$P_{fa_{\text{All}}}$	$P_{d_{\text{Scenario}}}$	$P_{d_{\text{Vehicles}}}$	$P_{d_{\text{VRU}}}$	CD_{All}	CD_{Targets}
Baseline (VRU)	63.1%	3.55%	60.7%	34.8%	23.1%	2.45m	6.73m
	(65.1%)	(3.09%)	**(61.6%)**	**(47.9%)**	(25.3%)	(2.15m)	(5.79m)
Baseline + Res (VRU)	63.0%	3.41%	60.9%	35.9%	24.2%	1.97m	5.95m
	(63.2%)	(2.98%)	(58.4%)	(42.6%)	**(25.8%)**	**(1.77m)**	**(5.33m)**
Baseline + Filter (VRU)	59.4%	3.25%	55.8%	33.1%	21.1%	2.05m	6.71m
	(61.6%)	(3.02%)	(58.2%)	(38.3%)	(23.2%)	(1.96m)	(5.56m)
Baseline + LSTM (VRU)	53.9%	3.30%	50.5%	32.0%	22.5%	2.32m	8.38m
	(58.4%)	(3.38%)	(54.9%)	(37.9%)	(23.6%)	(2.31m)	(7.54m)
Only 3D (VRU)	52.8%	4.87%	48.9%	34.8%	21.1%	2.49m	8.25m
	(57.1%)	(3.99%)	(52.7%)	(36.6%)	(22.4%)	(2.32m)	(7.82m)
Variant 1 [41] (VRU)	54.2%	3.05%	49.9%	32.9%	21.3%	2.34m	7.53m
	(60.4%)	(3.38%)	(57.9%)	(34.7%)	(22.4%)	(2.31m)	(7.40m)
Variant 2 [39] (VRU)	42.4%	**2.43%**	39.1%	20.8%	13.7%	2.81m	8.14m
	(46.8%)	(2.77%)	(42.1%)	(28.7%)	(16.8%)	(2.69 m)	(7.79m)

From the Table 4.2, it can be observed that the VRU version of the baseline achieves the highest $P_{d_{\text{All}}}$, $P_{d_{\text{Scenario}}}$ and $P_{d_{\text{Vehicles}}}$, with values of 65.1%, 61.6% and 47.9% respectively. Meanwhile, the VRU version of 'baseline + Res' achieves the best performance in $P_{d_{\text{VRU}}}$, CD_{All} and CD_{Targets}. The relatively high $P_{d_{\text{All}}}$ and $P_{d_{\text{Vehicles}}}$ indicate that the proposed network effectively segments vehicles and objects with an acceptable $P_{fa_{\text{All}}}$. The generally low $P_{d_{\text{VRU}}}$ can be attributed to the following factors:

- **Low proportions of pedestrians and bicycles**: The small number of labels for these classes increases the difficulty of achieving accurate segmentation.
- **Closely parked bicycles**: This arrangement of the bicycles significantly modifies the shape and motion information of bicycles, leading some to be misclassified as scenario objects.

- **Low radar resolution**: The limited resolution of the radar makes it difficult to detect VRUs at greater distances. For the VRUs within 30 m, the $P_{d_{\text{VRU}}}$ of the VRU version of model 'Baseline + Res' increases to 42.8%

The low CD_{All} indicates that the generated radar PCs are similar to the LiDAR PCs; although there might be some slight misalignments with the closed points, the Chamfer distance still will not be influenced much, validating the approach for detection.

The relatively high Chamfer distance for the targets, regardless of the kinds of models used, can be attributed to the fact that the dataset was recorded in realistic road scenes, which include multiple parked cars, bicycles, and static pedestrians. They will be mostly predicted as scenario objects, but their labels are mostly correct, leading to the higher Chamfer distance. This mismatch contributes to the higher Chamfer distance. The static targets are indeed difficult for radar to perceive as their power return is much smaller than the dynamic ones.

According to the quantitative results provided in Table 4.2, comparisons between several modifications and the baseline, and analysis of the results are outlined as follows:

- **Baseline + Res**: This modification increases the network depth by incorporating residual blocks in the 3D backbone, which enhances feature extraction for reconstructing radar PCs. The lower Chamfer distances compared to the baseline indicate better similarity with the corresponding LiDAR PCs. However, this version shows a decrease in detection probabilities for all targets, vehicles, and scenario objects. Despite this, due to the more balanced results, it can still be considered a more effective option compared to the baseline.
- **Baseline + Filter**: In this modification, the temporal smoothing of input across consecutive frames improves consistency in the generated radar PCs, as reflected in the lower Chamfer distance compared to the baseline. However, the temporal Gaussian filter reduces the high power returns while increasing lower power returns, making it harder to extract features during segmentation. For instance, reducing the high power return when detecting dynamic vehicles makes vehicle segmentation more challenging compared to the original input format.
- **Baseline + LSTM**: Although this modification is designed to capture temporal information across consecutive frames, it underperforms relative to the baseline. This may be due to the input data format: the values in the input RAE tensor represent the average power across the Doppler channel, meaning that some information from the original RAED tensor is lost. Additionally, even when the ego-vehicle remains static, radar data still fluctuates due to environmental noise and multi-path effects. As a result, the hidden states from previous frames do not necessarily provide useful information for detection and segmentation.
- **Only 3D**: This version produces a higher $P_{fa_{\text{All}}}$ compared to the others, as using 3D convolution fails to handle the sparsity differences between the input tensors and labels effectively. This results in a denser output, despite its complex convolutional process. Additionally, it should be noted that this approach increases the computational load and extends training and test time.
- **Variant 1**: This variant uses a Doppler encoder to combine two channels in the RAED tensors. However, the differences in meaning and range between these two channels make it more difficult to utilize Doppler information and extract elevation information effectively. Consequently, this reduces segmentation and detection performance compared to the proposed semantic segmentation network.
- **Variant 2**: In this method, only the top 10% of power bins are selected for input, with the other remaining bins set to 0 for sparse convolution. Although this approach results in the lowest $P_{fa_{\text{All}}}$, the detection and segmentation performance is the worst among all the models. The reduced sparsity difference between the input and labels helps avoid noisy outputs, but it also leads to significant information loss, limiting the model's ability to detect and segment objects accurately.

Summarizing, the proposed semantic segmentation network achieves the highest $P_{d_{\text{All}}}$ (65.1%), $P_{d_{\text{Vechicles}}}$ (47.9%) and the lowest $CD_{All} = 1.77m$. The performance of the 'VRU' version of the proposed semantic segmentation network is better than individual segmentation of pedestrians and bicycles, particularly in the improvement of $P_{d_{\text{Vechicles}}}$ and $P_{d_{\text{VRU}}}$. The results of baseline and 'baseline + Res' modifications outperform the two variants, proving the effectiveness of the proposed semantic segmentation model.

Among these, the ‘Baseline + Res’ modification provides the most balanced performance with the lowest chamfer distance and an acceptable detection probability. When focusing on the ability to segment vehicles, the ‘VRU’ version of the baseline achieves the highest performance.

Although the $P_{d_{\text{Vechicles}}}$ of the ‘VRU’ version of the baseline is 47.9%, this metric includes the detection probability of both dynamic and static vehicles. To further investigate the relationship between radar detection and segmentation performance based on the motion state of vehicles, the models are evaluated using labels that include vehicle motion states, as described in section 2.3, to analyze the performance in detecting only dynamic vehicles.

Table 4.3 presents the metrics averaged across all test data using the two best models and two variants inspired by the literature. In addition to the previously discussed $P_{d_{\text{Vechicles}}}$, three new metrics are introduced to further evaluate performance:

- $P_{d_{\text{Vehicles (< 30 m)}}}$: Detection probability for vehicles closer than 30 m.
- $P_{d_{\text{Dynamic Vehicles}}}$: Detection probability for vehicles labeled as dynamic at all distances.
- $P_{d_{\text{Dynamic Vehicles (< 30 m)}}}$: Detection probability for vehicles labeled as dynamic closer than 30 m.

Table 4.3: Performance of the proposed semantic segmentation network vs alternative variants inspired from the literature on segmenting dynamic vehicles. The performance for segmenting vehicles closer than 30 m is also provided.

Methods	$P_{d_{\text{Vehicles}}}$	$P_{d_{\text{Vehicles (< 30 m)}}}$	$P_{d_{\text{Dynamic Vehicles}}}$	$P_{d_{\text{Dynamic Vehicles (< 30 m)}}}$
Baseline (VRU)	34.8% **(47.9%)**	41.8% **(57.5%)**	43.5% **(56.7%)**	49.0% **(64.3%)**
Baseline + Res (VRU)	35.9% (42.6%)	43.4% (50.9%)	46.1% (54.9%)	52.4% (62.3%)
Variant 1 [45] (VRU)	32.9% (34.7%)	40.3% (38.6%)	40.6% (42.3%)	48.6% (50.7%)
Variant 2 [39] (VRU)	20.8% (28.7%)	27.2% (32.1%)	34.2% (37.4%)	39.9% (43.1%)

From the Table 4.3, it can be concluded that the detection probability for dynamic vehicles is significantly higher than for all vehicles. This is due to the stronger power returns typically generated when detecting dynamic vehicles. Additionally, the proposed semantic segmentation network shows improved detection performance for vehicles within 30 m, which aligns with the assumptions regarding radar’s effectiveness at closer targets.

5

Discussion

This chapter proposes a deeper discussion on open questions related to the proposed automatic labeling process and semantic segmentation network, supported by related images and an analysis of potential reasons for unexpected or unsatisfactory behaviours.

5.1. Discussion related to the automatic labeling process

Why not directly use the semantic segmentation methods on LiDAR PCs to provide labels?

The objective of the automatic labeling process is to generate point-wise multi-class labels for LiDAR PCs, which are then transformed and voxelized to match the radar view and be the ground truth. Given the strong performance of semantic segmentation methods for LiDAR PCs on open-source datasets, the state-of-the-art pre-trained model, Point Transformer v3 (PTv3), is applied to LiDAR PCs for comparison in various indoor and outdoor 3D perception tasks during this research [61]. The labeling results of the proposed method and PTv3 are demonstrated in Figure 5.1.

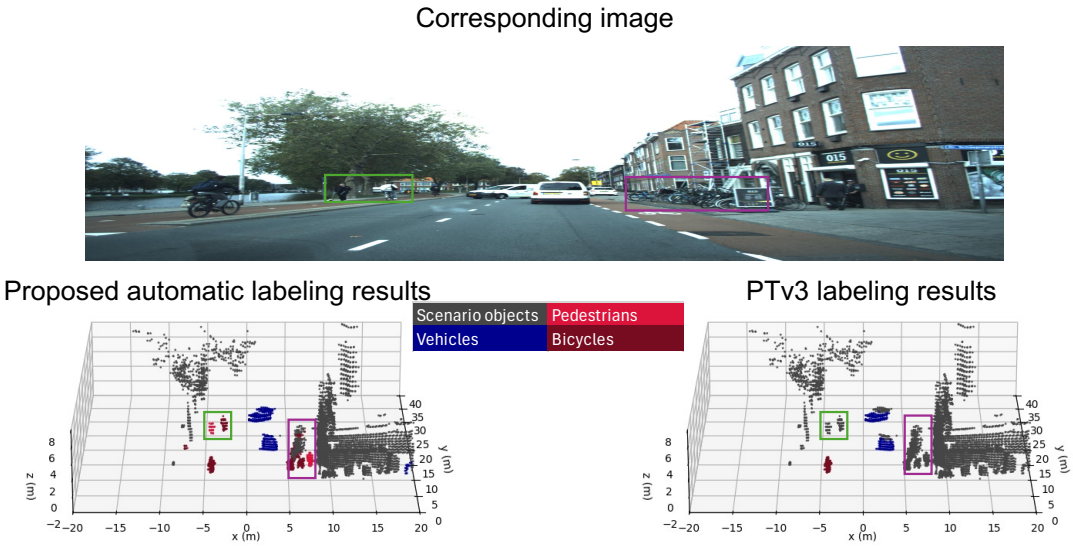

Figure 5.1: Labeling results using the proposed automatic labeling method and the pre-trained PTv3 model. The color bar is consistent with that used in the labeling process.

In Figure 5.1, it is evident that labeling using pre-trained PTv3 is less accurate than the proposed automatic labeling pipeline, particularly for bicycles and pedestrians, as highlighted in the green and purple boxes. These labeling imperfections stem from the fact that the PTv3 model was trained on the *NuScenes* dataset [62], where the LiDAR differs from the one used in the *RaDelft* dataset. As PTv3 is a transformer-based model, large-scale manually labeled data is required for fine-tuning, which significantly increases the labeling cost. For this reason, semantic segmentation methods like PTv3 are not directly applied to LiDAR PCs to generate labels in this study.

Why the calibration process mentioned in subsection 2.2.2 is not enough for providing labels?

In subsection 2.2.2, a calibration process is proposed to refine the preliminary labels by using the semantic information from the camera images. In this process, LiDAR PCs are projected onto the 2D segmented images and then assigned to the corresponding classes similar to the concept proposed by [63] recently. To evaluate the effectiveness of this approach, a comparison between the radar view results of the same scenario using the proposed automatic labeling method and the calibration process alone for final labels is provided in Figure 5.2.

The lower-left image in the Figure 5.2 shows the labeling results only using the camera for semantic information, while the lower-right images represents the proposed automatic labeling results. The camera-based method also applies a clustering technique for label smoothing. However, it is clear that the camera-based method has limited FoV which is smaller than the radar's. In contrast, the proposed automatic labeling method benefit from LiDAR's 360° FoV, providing wider coverage. Moreover, significant mislabeling occurs when targets are far from the sensor in the camera-based method. These two factors make the camera-based labeling method impractical for generating reliable labels.

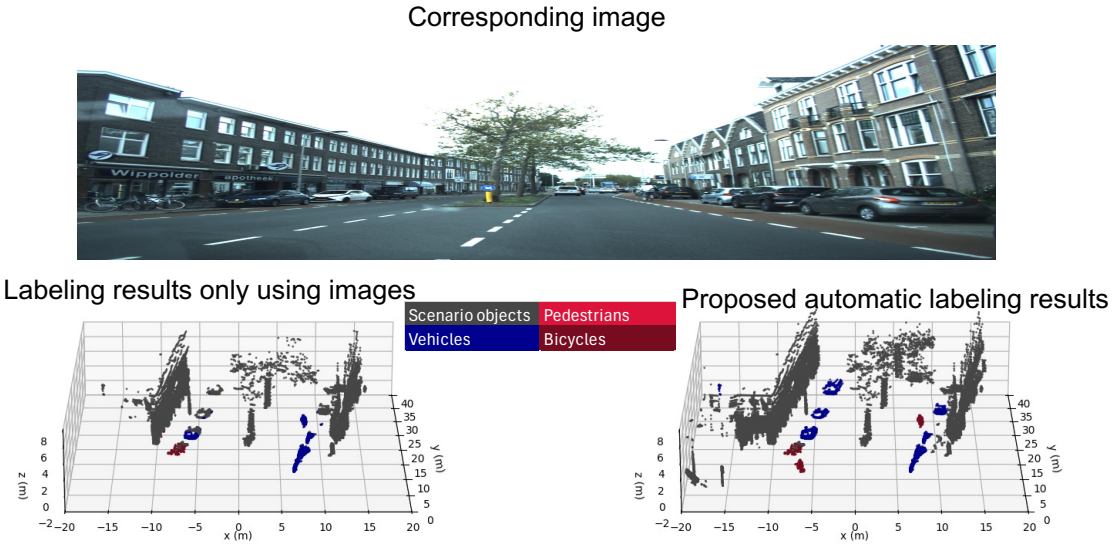

Figure 5.2: Labeling results using the proposed automatic labeling method and only the calibration process. The color bar is consistent with that used in the labeling process.

5.2. Discussion related to the proposed semantic segmentation network

Why are advanced deep learning architectures, such as transformer-based methods, not applied in the design of semantic segmentation networks?

In the proposed semantic segmentation network, traditional CNN-based structures are applied to extract the features and generate semantic results from radar tensors. Since this thesis is the first research on the semantic segmentation of 4D radar tensors, there are few existing references for directly applying semantic segmentation to 4D radar RAED tensors. Although transformer-based methods have shown success in semantic segmentation for RGB images and LiDAR PCs, they do not perform as well with radar tensors for several reasons:

- The transformer is designed to process sequential data without spatial structure. Due to the disparity between dense radar tensors as inputs and sparse voxelized LiDAR PCs as labels, the transformer struggles to capture complex spatial relationships.
- The self-attention mechanism in transformers is optimized for capturing long-time/distance dependencies. However, when targets are far away and features in the radar tensor are concentrated in only a few voxels, the transformer tends to overlook local details.
- Transformers require calculating attention weights for every element in the input tensor relative to all other elements. For a radar tensor with dimensions $500 \times 240 \times 34$, the computational cost is significantly higher than for processing images via CNNs, leading to inefficiencies.
- Radar tensors often contain noise and multi-path effects, which complicate the transformer's ability to differentiate between noise and relevant targets. The self-attention mechanism may incorrectly assign weights to noise, negatively impacting model performance.

Why the motion state labels are only used for evaluation, not in the training process?

In this thesis, motion state labels for vehicles faster than 18 km/h are used to further evaluate the model performance. However, during the training process, the motion state labels are not used as a new class in the semantic segmentation model or assign vehicles slower than 18 km/h to 'scenario objects'. This decision was based on the following reasons:

- Some dynamic vehicles are not correctly labeled due to limitations in the motion state labeling process. If these imperfect labels were used in training, some high-power voxels might not be labeled as dynamic. For instance, in Figure 5.3, two dynamic vehicles are present, but the LiDAR

PCs for those vehicles are partially occluded, making it difficult to form clusters that match across consecutive frames. As a result, the motion labels for some dynamic vehicles are incomplete or inaccurate.

- Vehicles faster than 18 km/h represent a relatively small proportion of the dataset. Adding a new class for these dynamic vehicles further increases the overall class imbalance, making the semantic segmentation task even more challenging.

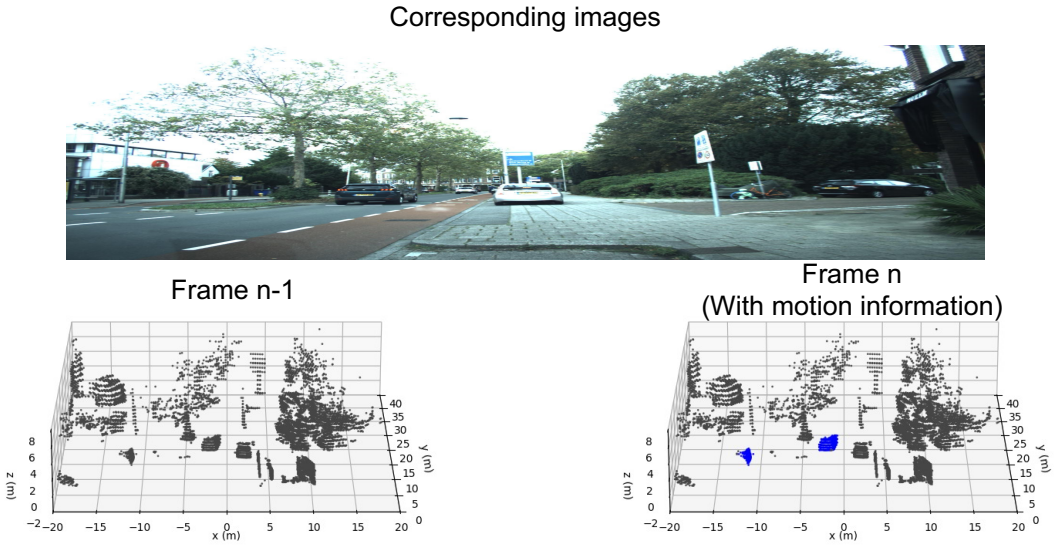

Figure 5.3: Motion state labels generated from two consecutive frames, where some dynamic vehicles do not appear in the LiDAR PCs. Dynamic vehicles are highlighted in blue.

6

Conclusion and future work

6.1. Conclusion

In this thesis, an automatic approach for generating multi-class point-wise labels is proposed for automotive datasets by utilizing complementary information from synchronized camera and LiDAR data. Additionally, a new radar semantic segmentation network is designed, using 4D radar tensors as inputs, supervised by the generated labels. These two novel contributions effectively address the research gaps identified in the literature review in section 1.3.

The automatic labeling process begins with the generation of preliminary object labels using only LiDAR PCs, followed by a calibration step incorporating semantic information from camera RGB images. Finally, voxelized labels ranging from 0 to 4 representing five classes are created to match the format of the input radar tensors. This process also includes motion state labels for vehicles, which can support the evaluation of the proposed semantic segmentation model.

The proposed semantic segmentation network employs both a 2D and 3D backbone to generate voxelized radar PCs with class information. This architecture effectively extracts features from the radar tensors while addressing the sparsity differences between inputs and labels. It also reduces computational complexity compared to using a 3D backbone alone.

Promising results are shown by applying both proposed contributions to the publicly shared *RaDelft* dataset. The automatic labeling process demonstrates satisfactory quantitative and qualitative results compared with manual labeling results of randomly chosen scenes. The outputs of the semantic segmentation network achieve more than 65% in overall detection probability, improving by +13.1% in vehicle detection probability and reducing 0.54 m in Chamfer distance compared to the alternative variants inspired by the literature.

In conclusion, this thesis proposes an efficient method for labeling LiDAR PCs that can serve as ground truth for various automotive radar perception tasks. It also explores the extended capabilities of automotive radar RAED tensors, demonstrating their potential beyond current applications. The point-wise multi-class labels for the *RaDelft* dataset, along with the related codes have been uploaded to *4TU.Federation* and `https://github.com/RaDelft/RaDelft-Dataset` for future research. Additionally, the results of this thesis have been submitted for publication to the ICASSP 2025 conference.

6.2. Future work

In the process of researching the automatic labeling and semantic segmentation of the 4D radar tensors, some aspects were left unexplored due to various limitations. To address these challenges and improve both model performance and labeling quality, several suggestions and improvements for future research are outlined. These suggestions are divided into two areas: the automatic labeling process and the semantic segmentation network.

Suggestions for automatic labeling process

1. **Improving Bounding Box Quality**:

   One key improvement in the automatic labeling process is refining the quality of bounding boxes during preliminary object label generation. In this thesis, a pre-trained model was applied from a large-scale dataset, but the LiDAR type and road scenery differed from that dataset. Due to the lack of manual labels, model refinement was not possible to conduct. By refining this pre-trained model on relevant data, more accurate bounding boxes can be produced.

2. **Enhancing Calibration Process**:

   In the *RaDelft* dataset, only a synchronous camera with a limited FoV is available, which restricts calibration of side-view targets. Additionally, the calibration process has a limited effective range due to slight extrinsic parameter errors between the camera and LiDAR. Future research should explore using multiple cameras with wider FoV and more accurate extrinsic calibration to better assign semantic information from images to LiDAR PCs for improved calibration.

3. **Incorporating Full Motion State Information**:

   In this thesis, only vehicles moving faster than 18 km/h are labeled as dynamic targets, and this information is not used in training. Ideally, by leveraging consecutive LiDAR frames with motion compensation, the scene flow of each point could be inferred. With a full understanding of each point's motion state, confusion between parked vehicles, bicycles, and other scenario objects can be better resolved during training.

Suggestions for proposed semantic segmentation network

1. **Improving Temporal Information Integration**:

   Although temporal information from previous frames was integrated into the network through both input data and network architecture, quantitative results did not exceed the baseline. It can also be observed that predictions vary across consecutive frames and lack the stability seen in the ground truth. Future work could explore more efficient signal processing methods for radar data and integrate techniques designed specifically for temporal information in automotive radar signal processing, such as incorporating attention mechanisms across time frames to better capture temporal dependencies and improve performance.

2. **Preserving Doppler Information**:

   The 4D radar tensors used in the *RaDelft* dataset retain only one channel for the Doppler axis, which results in information loss. While an alternative approach that kept the top three power values in the Doppler axis was tested, it failed to retain essential features and introduced noise. Future research should focus on preserving more meaningful Doppler information while balancing computational load to enhance performance.

3. **Combining Radar Tensors and RGB Images**:

   Although the location of generated radar PCs closely aligns with LiDAR PCs in terms of FoV, segmentation performance still needs to improve. On the other hand, images perform well in segmentation but lack detailed depth information. By combining radar tensors with images, higher resolution radar point clouds can be generated along with more accurate class information, potentially making radar an alternative to LiDAR in ADAS.

References

[1] Manzoor Ahmed Khan et al. “Level-5 autonomous driving—are we there yet? a review of research literature”. In: *ACM Computing Surveys (CSUR)* 55.2 (2022), pp. 1–38.

[2] Zhangjing Wang, Yu Wu, and Qingqing Niu. “Multi-sensor fusion in automated driving: A survey”. In: *Ieee Access* 8 (2019), pp. 2847–2868.

[3] Arthur Venon et al. “Millimeter wave fmcw radars for perception, recognition and localization in automotive applications: A survey”. In: *IEEE Transactions on Intelligent Vehicles* 7.3 (2022), pp. 533–555.

[4] Martin Stolz et al. “A New Antenna Array and Signal Processing Concept for an Automotive 4D Radar”. In: *2018 15th European Radar Conference (EuRAD)*. 2018, pp. 63–66. DOI: 10.23919/EuRAD.2018.8546603.

[5] Zeyu Han et al. “4D millimeter-wave radar in autonomous driving: A survey”. In: *arXiv preprint arXiv:2306.04242* (2023).

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[7] Jiaying Zhang et al. “A review of deep learning-based semantic segmentation for point cloud”. In: *IEEE access* 7 (2019), pp. 179118–179133.

[8] 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*. 2018, pp. 4490–4499.

[9] Bence Major et al. “Vehicle Detection With Automotive Radar Using Deep Learning on Range-Azimuth-Doppler Tensors”. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019, pp. 924–932. DOI: 10.1109/ICCVW.2019.00121.

[10] Hermann Rohling. “Radar CFAR thresholding in clutter and multiple target situations”. In: *IEEE transactions on aerospace and electronic systems* 4 (1983), pp. 608–621.

[11] Jakob Lombacher et al. “Object classification in radar using ensemble methods”. In: *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. 2017, pp. 87–90. DOI: 10.1109/ICMIM.2017.7918863.

[12] Bence Major et al. “Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019.

[13] Xiangyu Gao et al. “Ramp-cnn: A novel neural network for enhanced automotive radar object recognition”. In: *IEEE Sensors Journal* 21.4 (2020), pp. 5119–5132.

[14] Yi Jin et al. “Cross-Modal Supervision-Based Multitask Learning With Automotive Radar Raw Data”. In: *IEEE Transactions on Intelligent Vehicles* 8.4 (2023), pp. 3012–3025. DOI: 10.1109/TIV.2023.3234583.

[15] Arthur Ouaknine et al. “Multi-view radar semantic segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15671–15680.

[16] Yizhou Wang et al. “RODNet: A real-time radar object detection network cross-supervised by camera-radar fused object 3D localization”. In: *IEEE Journal of Selected Topics in Signal Processing* 15.4 (2021), pp. 954–967.

[17] Andras Palffy et al. “Multi-class road user detection with 3+ 1D radar in the View-of-Delft dataset”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4961–4968.

[18] Michael Meyer, Georg Kuschk, and Sven Tomforde. “Graph convolutional networks for 3d object detection on radar data”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3060–3069.

[19] Wei Zhang et al. “Super resolution DOA based on relative motion for FMCW automotive radar”. In: *IEEE Transactions on Vehicular Technology* 69.8 (2020), pp. 8698–8709.

[20] Ole Schumann et al. “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)*. 2017, pp. 1–6. DOI: 10.1109/SDF.2017.8126350.

[21] Charles R. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[22] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems* 30 (2017).

[23] Ole Schumann et al. “Semantic Segmentation on Radar Point Clouds”. In: *2018 21st International Conference on Information Fusion (FUSION)*. 2018, pp. 2179–2186. DOI: 10.23919/ICIF.2018.8455344.

[24] Yingjie Wang et al. “Bi-LRFusion: Bi-directional LiDAR-radar fusion for 3D dynamic object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13394–13403.

[25] Weijing Shi and Raj Rajkumar. “Point-gnn: Graph neural network for 3d object detection in a point cloud”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 1711–1719.

[26] Peter Svenningsson, Francesco Fioranelli, and Alexander Yarovoy. “Radar-pointgnn: Graph based object recognition for unstructured radar point-cloud data”. In: *2021 IEEE Radar Conference (RadarConf21)*. IEEE. 2021, pp. 1–6.

[27] Pou-Chun Kung, Chieh-Chih Wang, and Wen-Chieh Lin. “Radar occupancy prediction with lidar supervision while preserving long-range sensing and penetrating capabilities”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2637–2643.

[28] Robert Prophet et al. “Semantic segmentation on 3D occupancy grids for automotive radar”. In: *IEEE Access* 8 (2020), pp. 197917–197930.

[29] Ole Schumann et al. “Scene understanding with automotive radar”. In: *IEEE Transactions on Intelligent Vehicles* 5.2 (2019), pp. 188–203.

[30] Bin Tan et al. “3d object detection for multi-frame 4d automotive millimeter-wave radar point cloud”. In: *IEEE Sensors Journal* (2022).

[31] Yanlong Yang et al. “RaLiBEV: Radar and LiDAR BEV fusion learning for anchor box free object detection system”. In: *arXiv preprint arXiv:2211.06108* (2022).

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015, pp. 234–241.

[34] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[35] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[36] Hengshuang Zhao et al. “Point transformer”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 16259–16268.

[37] Martin Dimitrievski et al. “Weakly supervised deep learning method for vulnerable road user detection in FMCW radar”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–8.

[38] Arthur Ouaknine et al. “Carrada dataset: Camera and automotive radar with range-angle-doppler annotations”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 5068–5075.

[39] Dong-Hee Paek, Seung-Hyun Kong, and Kevin Tirta Wijaya. “K-radar: 4d radar object detection for autonomous driving in various weather conditions”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 3819–3829.

[40] Julien Rebut et al. “Raw high-definition radar for multi-task learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17021–17030.

[41] I. Roldan et al. *RaDelft Dataset: a large-scale, real-life, and multi-sensor automotive dataset (Version 1)*. 4TU.ResearchData, 2024. URL: `https://doi.org/10.4121/4E277430-E562-4A7A-ADFE-30B58D9A5F0A.V1`.

[42] Ao Zhang, Farzan Erlik Nowruzi, and Robert Laganiere. “Raddet: Range-azimuth-doppler based radar object detection for dynamic road users”. In: *2021 18th Conference on Robots and Vision (CRV)*. IEEE. 2021, pp. 95–102.

[43] Michael Meyer and Georg Kuschk. “Automotive Radar Dataset for Deep Learning Based 3D Object Detection”. In: *2019 16th European Radar Conference (EuRAD)*. 2019, pp. 129–132.

[44] Lianqing Zheng et al. “Tj4dradset: A 4d radar dataset for autonomous driving”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2022, pp. 493–498.

[45] Ignacio Roldan et al. “See Further Than CFAR: a Data-Driven Radar Detector Trained by Lidar”. In: *arXiv preprint arXiv:2402.12970* (2024).

[46] Shaoshuai Shi et al. “From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.8 (2020), pp. 2647–2664.

[47] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[48] Seungjae Lee, Hyungtae Lim, and Hyun Myung. “Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3D point cloud”. In: *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* 2022, pp. 13276–13283.

[49] Jitesh Jain et al. “Oneformer: One transformer to rule universal image segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 2989–2998.

[50] Joris Domhof, Julian FP Kooij, and Dariu M Gavrila. “A joint extrinsic calibration tool for radar, camera and lidar”. In: *IEEE Transactions on Intelligent Vehicles* 6.3 (2021), pp. 571–582.

[51] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[52] Yancong Lin and Holger Caesar. “Icp-flow: Lidar scene flow estimation with icp”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 15501–15511.

[53] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[54] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[55] Özgün Çiçek et al. “3D U-Net: learning dense volumetric segmentation from sparse annotation”. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*. Springer. 2016, pp. 424–432.

[56] Xingjian Shi et al. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. In: *Advances in neural information processing systems* 28 (2015).

[57] Haibo He and Edwardo A Garcia. “Learning from imbalanced data”. In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[58] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. “V-net: Fully convolutional neural networks for volumetric medical image segmentation”. In: *2016 fourth international conference on 3D vision (3DV)*. Ieee. 2016, pp. 565–571.

[59] Lukas Liebel and Marco Körner. “Auxiliary tasks in multi-task learning”. In: *arXiv preprint arXiv:1805.06334* (2018).

[60] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. `https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2`. 2024.

[61] Xiaoyang Wu et al. “Point Transformer V3: Simpler Faster Stronger”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 4840–4851.

[62] Holger Caesar et al. “nuScenes: A multimodal dataset for autonomous driving”. In: *arXiv preprint arXiv:1903.11027* (2019).

[63] Max Heidbrink et al. “Concept for Automatic Annotation of Automotive Radar Data Using AI-Segmented Camera and Lidar Reference Data”. In: *2024 21st European Radar Conference (EuRAD)*. IEEE. 2024.