

# RadarPillars: Efficient Object Detection from 4D Radar Point Clouds

Alexander Musiat<sup>1</sup>, Laurenz Reichardt<sup>1</sup>, Michael Schulze<sup>1</sup> and Oliver Wasenmüller<sup>1</sup>

**Abstract**—Automotive radar systems have evolved to provide not only range, azimuth and Doppler velocity, but also elevation data. This additional dimension allows for the representation of 4D radar as a 3D point cloud. As a result, existing deep learning methods for 3D object detection, which were initially developed for LiDAR data, are often applied to these radar point clouds. However, this neglects the special characteristics of 4D radar data, such as the extreme sparsity and the optimal utilization of velocity information. To address these gaps in the state-of-the-art, we present RadarPillars, a pillar-based object detection network. By decomposing radial velocity data, introducing PillarAttention for efficient feature extraction, and studying layer scaling to accommodate radar sparsity, RadarPillars significantly outperform state-of-the-art detection results on the View-of-Delft dataset. Importantly, this comes at a significantly reduced parameter count, surpassing existing methods in terms of efficiency and enabling real-time performance on edge devices.

## I. INTRODUCTION

In the context of autonomy and automotive applications, radar stands out as a pivotal sensing technology, enabling vehicles to detect objects and obstacles in their surroundings. This capability is crucial for ensuring the safety and efficiency of various autonomous driving functionalities, including collision avoidance, adaptive cruise control, and lane-keeping assistance. Recent advancements in radar technology have led to the development of 4D radar, incorporating three spatial dimensions along with an additional dimension for Doppler velocity. Unlike traditional radar systems, 4D radar introduces elevation information as its third dimension. This enhancement allows for the representation of radar data in 3D point clouds, akin to those generated by LiDAR or depth sensing cameras, thereby enabling the application of deep learning methodologies previously reserved for such sensors.

However, while deep learning techniques from the domain of LiDAR detection have been adapted to 4D radar data, they have not fully explored or adapted to its unique features. Compared to LiDAR data, 4D radar data is significantly less abundant. Regardless of this sparsity, radar uniquely provides velocity as a feature, which could help in the detection of moving objects in various scenarios, such as at long range where LiDAR traditionally struggles [1]. In the View-of-Delft dataset, an average 4D radar scan comprises only 216 points, while a LiDAR scan within the same field of view contains 21,344 points [2]. In response, we propose

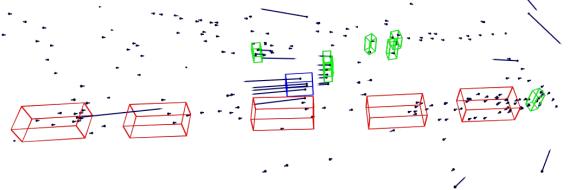


Fig. 1: Example of our RadarPillars detection results on 4D radar. Cars are marked in red, pedestrians in green and cyclists in blue. The radial velocities of the points are indicated by arrows.

our *RadarPillars*, a novel 3D detection network tailored specifically for 4D radar data. Through RadarPillars we address gaps in the current state-of-the-art with the following contributions, significantly improving performance, while maintaining real-time capabilities:

- Enhancement of velocity information utilization: We decompose radial velocity data, providing additional features to significantly enhance network performance.
- Adapting to radar sparsity: RadarPillars leverages the pillar representation [3] for efficient real-time processing. We capitalize on the sparsity inherent in 4D radar data and introduce *PillarAttention*, a novel self-attention layer treating each pillar as a token, while maintaining both efficiency and real-time performance.
- Scaling for sparse radar data: We demonstrate that the sparsity of radar data can lead to less informative features in the detection network. Through uniform network, we not only improve performance but also significantly reduce parameter count, enhancing runtime efficiency.

## II. RELATED WORK

### A. 4D Radar Object Detection

Point clouds can be processed in various ways: as an unordered set of points, ordered by graphs, within a discrete voxel grid, or as range projections. Among these representations, pillars stand out as a distinct type, where each voxel is defined as a vertical column, enabling the reduction of the height dimension. This allows for pillar features to be cast into a 2D-pseudo-image, with its height and width defined by the grid size used for the base of the pillars. This dimensionality reduction facilitates the application of 2D network architectures for birds-eye-view processing. PointPillars-based [3] networks have proven particularly effective for LiDAR data, balancing performance and runtime

<sup>1</sup>Mannheim University of Applied Sciences, Germany  
alex.musiat@gmail.com  
l.reichardt@hs-mannheim.de  
m.schulze@hs-mannheim.de  
o.wasenmueller@hs-mannheim.de

efficiently. Consequently, researchers have begun applying the pillar representation to 4D radar data. Currently, further exploration of alternative representation methods besides pillars for 4D radar data remains limited.

Palffy *et al.* [2] established a baseline by benchmarking PointPillars on their View-of-Delft dataset, adapting only the parameters of the pillar-grid to match radar sensor specifications. Recognizing the sparsity inherent in 4D radar data, subsequent work aims to maximize information utilization through parallel branches or multi-scale fusion techniques. SMURF [4] introduces a parallel learnable branch to the pillar representation, integrating kernel density estimation. MVFAN [5] employs two parallel branches — one for cylindrical projection and the other for the pillar representation — merging features prior to passing them through an encoder-decoder network for detection. SRFF [6] does not use a parallel branch, instead incorporating an attention-based neck to fuse encoder-stage features, arguing that multi-scale fusion improves information extraction from sparse radar data.

Further approaches like RC-Fusion [7] and LXL [8] and GRC-Net [9] opt to fuse both camera and 4D radar data, taking a dual-modality approach at object detection. CM-FA [10] uses LiDAR data during training, but not during inference.

It's worth noting that the modifications introduced by these methods come at the cost of increased computational load and memory requirements, compromising the real-time advantage associated with the pillar representation. Furthermore, none of these methods fully explore the optimal utilization of radar features themselves. Herein lies untapped potential.

### B. Transformers in Point Cloud Perception

The self-attention mechanism [11] dynamically weighs input elements in relation to each other, capturing long-range dependencies and allowing for a global receptive field for feature extraction. Self-attention incorporated in the transformer layer has benefited tasks like natural language processing, computer vision, and speech recognition, achieving state-of-the-art performance across domains. However, applying self-attention to point clouds poses distinct challenges. The computational cost is quadratic, limiting the amount of tokens (context window) and hindering long-range processing compared to convolutional methods. Additionally, the inherent sparsity and varied point distributions complicate logical and geometric ordering, thus impeding the adoption of transformer-based architectures in point cloud processing.

Various strategies have been proposed to address these challenges. Point Transformer [12] utilizes k-nearest-neighbors (KNN) to group points before applying vector attention. However, the neighborhood size is limited, as KNN grouping is also quadratic in terms of memory requirements and complexity. On top of grouping, some approaches reduce the point cloud through pooling [13] or farthest-point sampling [14], leading to information loss.

Others partition the point cloud into groups of equal geometric shape, employing window-based attention [15],

[16], [17] or the octree representation [18]. The downside of geometric partitioning is that groups of equal shape will each have a different amount of points in them. This is detrimental to parallelization, meaning that such methods are not real time capable. Despite these efforts, partition based attention is limited to the local context, with various techniques to facilitate information transfer between these groups such as changing neighborhood size, downsampling, or shifting windows. The addition of constant shifting and reordering of data leads to further memory inefficiencies and increased latency. In response to these challenges, Flatformer [19] opts for computational efficiency by forming groups of equal size rather than equal geometric shape, sacrificing spatial proximity for better parallelization and memory efficiency. Similarly, SphereFormer [20] voxelizes point clouds based on exponential distance in the spherical coordinate system to achieve higher density voxel grids. Point Transformer v3 [21] first embeds voxels through sparse convolution and pooling, then ordering and partitioning the resulting tokens using space-filling curves. Through this, only the last group along the curve needs padding, thereby prioritizing efficiency through pattern-based ordering over spatial ordering or geometric partitioning.

These methods often require specialized attention libraries that do not leverage the efficient attention implementations available in standard frameworks.

## III. METHOD

The current state-of-the-art in 4D radar object detection predominantly relies on LiDAR-based methods. As a result, there is a noticeable gap in research regarding the comprehensive utilization of velocity information to enhance detection performance. Despite incremental advancements in related works, these improvements often sacrifice efficiency and real-time usability. To address these issues, we delve into optimizing radar features to improve network performance through enhanced input data quality.

While various self-attention variants have been explored in point cloud perception, their restricted receptive fields, in conjunction with the sparsity and irregularity of point clouds, lead to computationally intensive layers. Leveraging the sparsity inherent in 4D radar data, we introduce PillarAttention, a novel self-attention layer providing a global receptive field by treating each pillar as a token. Contrary to existing layers, PillarAttention does not reduce features through tokenization or need complex ordering algorithms. Additionally, we investigate network scaling techniques to further enhance both runtime efficiency and performance in light of the radar data sparsity.

### A. 4D Radar Features

The individual points within 4D radar point clouds are characterized by various parameters including range ( $r$ ), azimuth ( $\alpha$ ), elevation ( $\theta$ ), RCS reflectivity, and relative radial velocity ( $v_{rel}$ ). The determination of radial velocity relies on the Doppler effect, reflecting the speed of an object

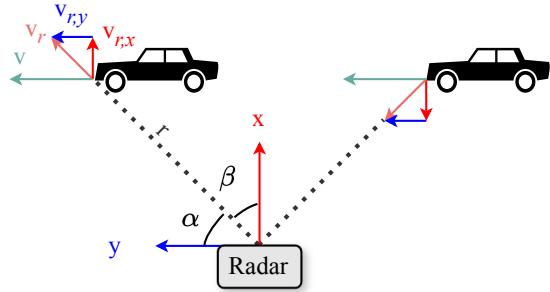


Fig. 2: Absolute radial velocity  $v_r$  compensated with ego motion of 4D radar. As an object moves,  $v_r$  changes depending on its heading angle to the sensor. The car's actual velocity  $v$  remains unknown, as its heading cannot be determined. However,  $v_r$  can be decomposed into its  $x$  and  $y$  components to provide additional features. The coordinate system and nomenclature follows the View-of-Delft dataset [2].

in relation to the sensor's position. When dealing with a non-stationary radar sensor (e.g. mounted on a car), compensating  $v_{rel}$  with the ego-motion yields the absolute radial velocity  $v_r$ . The spherical coordinates  $(r, \alpha, \theta)$  can be converted into Cartesian coordinates  $(x, y, z)$ . While these features are akin to LiDAR data, radar's unique capability lies in providing velocity information. Despite the commonality in coordinate systems between radar and LiDAR, radar's inclusion of velocity remains unique and underutilized. Current practices often incorporate velocity information merely as an additional feature within networks. Therefore, our investigation delves into the impact of both relative and absolute radial velocities. Through this analysis, we advocate for the creation of supplementary features derived from radial velocity, enriching the original data points.

First, we explore decomposing  $v_r$  into its  $x$  and  $y$  components, resulting in vectors  $v_{r,x}$  and  $v_{r,y}$ , respectively. This approach similarly applies to  $v_{rel}$ . This concept is visualized in Figure 2. The velocity vectors of each point can be decomposed through the following equations. Note that the Equation (1) and Equation (2) apply to both  $v_r$  and  $v_{rel}$  in the Cartesian coordinate system, in which  $\arctan\left(\frac{y}{x}\right) = \beta$ .

$$v_{r,x} = \cos\left(\arctan\left(\frac{y}{x}\right)\right) \cdot v_r \quad (1)$$

$$v_{r,y} = \sin\left(\arctan\left(\frac{y}{x}\right)\right) \cdot v_r \quad (2)$$

Secondly, we construct new features by calculating the offset velocities inside a pillar. For this, we first average the velocities inside a pillar and then subtract it from the velocity of each point, to form an additional offset feature. These new features can be calculated for both radial velocities  $v_{rel}, v_r$  and their decomposed  $x, y$  variants. In later experiments we denote the use of these new offset features with subscript  $m$ , for example  $v_{r,m}$  when using the offset velocities for  $v_r$ .

The construction of these additional point features is intended to make it easier for the model to learn dependencies from the data in order to increase performance in a way

which does not influence the runtime of the model, beyond its input layer.

### B. PillarAttention

The pillar representation of 4D radar data as a 2D pseudo-image is very sparse, with only a few valid pillars. Due to this sparsity, pillars belonging to the same object are far apart. When processed by a convolutional backbone with a local field of view, this means that early layers cannot capture neighborhood dependencies. This is only achieved with subsequent layers and the resulting increase in the effective receptive field, or by the downsampling between network stages [22], [23]. As such, the aggregation of information belonging to the same object occurs late within the network backbone. However, downsampling can lead to the loss of information critical to small objects. The tokenization and grouping methods of point cloud transformers can have a similar negative effect.

Inspired by Self-Attention [11], we introduce PillarAttention to globally connect the local features of individual pillars across the entire pillar grid. We achieve this by capitalizing on the inherent sparsity of 4D radar data, treating each pillar as a token, allowing our method to be free of grouping or downsampling methods. PillarAttention diverges from conventional self-attention in the manner in which sparsity is handled. Given the largely empty nature of the pillar grid with size  $H, W$ , we employ a sparsity mask to exclusively gather only occupied pillar features  $p$ . Subsequently, we learn key ( $K$ ), query ( $Q$ ), and value ( $V$ ) before applying standard self-attention. Conventionally, sparse values are masked during self-attention calculation. In contrast, our approach reduces the spatial complexity and memory requirements for self-attention from  $(HW)^2$  to  $p^2$ . Nevertheless, it's essential to acknowledge that sparsity, and thus the number of valid pillars, varies between scans. Consequently, the sequence length of tokens fluctuates during both training and inference. Another difference to conventional self-attention is that we did not find the inclusion of position embedding necessary. This can be attributed to the fact that pillar features inherently contain position information derived from point clouds.

Moreover, since pillars are organized within a 2D grid, the order of tokens remains consistent across scans, allowing the model to learn contextual relationships between individual pillars. As such, the use of specialized algorithms for ordering such as octrees and space filling curves is not needed. Also, PillarAttention is not reliant on specialized libraries and benefits from recent developments in the space such as Flash-Attention-2 [24].

We next PillarAttention inside a transformer layer. This layer is encapsulated by two MLPs which control its hidden dimension  $E$ . Following PillarAttention, the transformed features are scattered back into their original pillar positions. The concept of PillarAttention is depicted in Figure 3.

### C. Architecture and Scaling

Our architecture (see Figure 3) is loosely inspired by PointPillars [3]. Similar to PointPillars, we incorporate offset

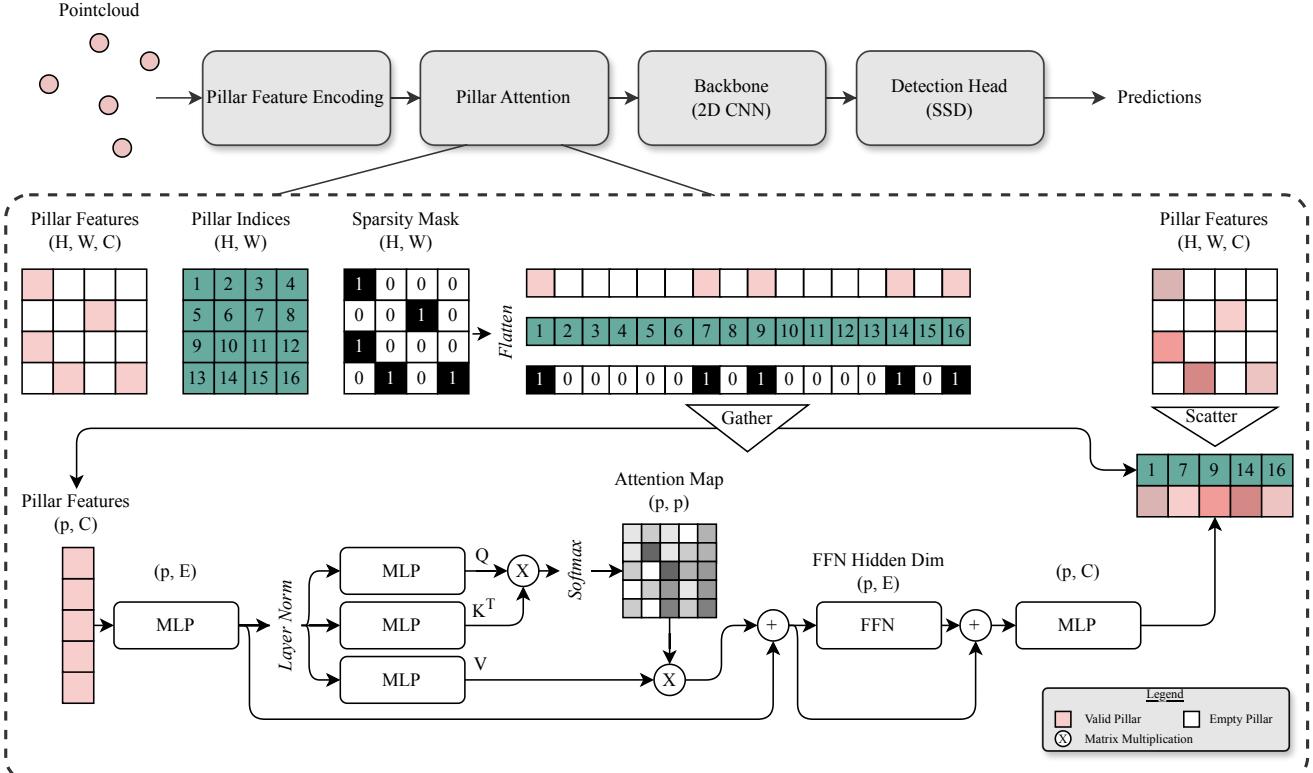


Fig. 3: Overview of our PillarAttention. We leverage the sparsity of radar point clouds by using a mask to gather features from non-empty pillars, reducing spatial size from  $H, W$  to  $p$ . Each pillar-feature with  $C$  channels is treated as a token for the calculation of self-attention. Our PillarAttention is encapsulated in a transformer layer, with the feed-forward network (FFN) consisting of Layer Norm, followed by two MLPs with the GeLU activation between them. The hidden dimension  $E$  of PillarAttention is controlled by a MLP before and after the layer. Finally, the pillar features with  $C$  channels are scattered back to their original position within the grid. Our PillarAttention does not use position embedding.

coordinates  $x_c, y_c, z_c$  derived from the pillar center  $c$  as additional features within the point cloud. Subsequently, we employ a PointNet [25] layer to transform the point cloud into pillar features, resembling a 2D pseudo image. These pillar features undergo processing via our novel PillarAttention mechanism, followed by a three-stage encoder. Each encoder stage contains  $3 \times 3$  2D convolution layers, with the ReLU activation function and batch normalization. The first stage employs three layers, while subsequent stages employ five. Additionally, the initial convolution layers in stages two and three downsample features with a stride of two. The output features of each encoder stage undergo upsampling via transposed 2D convolution before being concatenated. Finally, we employ a SSD [26] detection head to derive predictions from these concatenated features.

The sparsity inherent in 4D radar data can severely impact neural network learning. Previous research [22], [23] has demonstrated in the context of LiDAR perception that sparsity propagates between layers, influencing the expressiveness of individual layers. This diminishes the network's capacity to extract meaningful features from the data, where certain neurons fail to activate due to insufficient input. Consequently, a network may struggle to generalize well

to unseen data or exhibit suboptimal performance in tasks such as object detection or classification. Therefore, adapting to data sparsity is crucial for ensuring the robustness and efficiency of neural network-based approaches in 4D radar perception tasks.

In the View-of-Delft dataset, the ratio of LiDAR points to radar points is approximately 98.81. Despite this significant difference, current state-of-the-art 4D radar detection methods employ architectures originally designed for denser LiDAR point clouds. Given the limited points captured by 4D radar, we theorize that networks need less capacity as only a limited amount of meaningful features can be learned.

We propose a solution by suggesting uniform scaling of neural network encoder stages when transitioning from LiDAR to 4D radar data. In the case of RadarPillars, we used the same amount of channels  $C$  in all encoder stages of the architecture. In contrast, networks based on PointPillars double the amount of channels  $C$  with each stage. Our approach is expected to enhance both performance through generalization and runtime efficiency.

#### IV. EVALUATION

We evaluate our network RadarPillars for object detection on 4D radar data on the View-of-Delft (VoD) dataset [2]. As

TABLE I: Comparison of RadarPillars to different LiDAR and 4D radar models on the validation split of the View-of-Delft dataset. R and C indicate the 4D radar and camera modalities respectively for both training and inference. (L) indicates LiDAR during training only.

Model	Modality	Entire Area			Driving Corridor			Frame rate				
		mAP	Car	Pedestrian	Cyclist	mAP	Car	Pedestrian	Cyclist	V100	RTX 3090	AGX Xavier
1-Frame Data												
Point-RCNN* [30]	R	29.7	31.0	16.2	42.1	55.7	59.5	32.2	75.0	23.5	63.2	10.1
Voxel-RCNN* [31]	R	36.9	33.6	23.0	54.1	63.8	70.0	38.3	83.0	23.1	51.4	9.3
PV-RCNN* [32]	R	43.6	39.0	32.8	59.1	64.5	71.5	43.5	78.6	15.2	34.3	4.1
PV-RCNN++* [33]	R	40.7	36.2	28.7	57.1	61.5	68.3	39.1	77.3	9.9	20.1	2.7
SECOND* [34]	R	33.2	32.8	22.8	44.0	56.1	69.0	33.9	65.3	34.6	88.6	11.6
PillarNet* [35]	R	23.7	25.8	11.8	33.6	43.8	56.7	17.0	57.6	42.7	104.0	20.2
PointPillars* [3]	R	39.5	30.2	25.6	62.8	60.9	61.5	36.8	84.5	77.0	182.3	20.6
MVFAN [5]	R	39.4	34.1	27.3	57.1	64.4	69.8	38.7	84.9	-	-	-
<b>RadarPillars (ours)</b>	R	<b>46.0</b>	36.0	35.5	66.4	<b>67.3</b>	69.4	47.1	85.4	86.6	184.5	34.3
CM-FA [10]	R+(L)	41.7	32.3	42.4	50.4	-	-	-	-	-	23.0	-
GRC-Net [9]	R+C	41.1	27.9	31.0	64.6	-	-	-	-	-	-	-
RC-Fusion [7]	R+C	49.7	41.7	39.0	68.3	69.2	71.9	47.5	88.3	-	10.8	-
LXL [8]	R+C	<b>56.3</b>	42.3	49.5	77.1	<b>72.9</b>	72.2	58.3	88.3	6.1	-	-
RCBVE [36]	R+C	49.9	40.6	38.8	70.4	69.8	72.4	49.8	87.0	-	21.0	-
3-Frame Data												
PointPillars*	R	44.1	39.2	29.8	63.3	67.7	71.8	45.7	85.7	75.6	182.2	20.2
<b>RadarPillars (ours)</b>	R	<b>50.4</b>	40.2	39.2	71.8	<b>70.0</b>	70.9	51.4	87.6	85.8	183.1	34.1
5-Frame Data												
SRFF [6]	R	46.2	36.7	36.8	65.0	66.9	69.1	47.2	84.3	-	-	-
SMIFormer [37]	R	48.7	39.5	41.8	64.9	<b>71.1</b>	77.04	53.4	82.9	-	-	-
SMURF [4]	R	<b>51.0</b>	42.3	39.1	71.5	69.7	71.7	50.5	86.9	30.3	-	-
PointPillars* [3]	R	46.7	38.8	34.4	66.9	67.8	71.9	45.1	88.4	78.4	178.4	20.6
<b>RadarPillars (ours)</b>	R	50.7	41.1	38.6	72.6	70.5	71.1	52.3	87.9	82.8	179.1	34.4

\* Re-implemented

there there is no public benchmark or test-split evaluation, we follow established practice and perform all experiments on the validation split. Following VoD, we use the mean Average Precision (mAP) across both the entire sensor area and the driving corridor as metrics. During training, we augment the dataset by randomly flipping and scaling the point cloud. Data is normalized according to the mean and standard deviation. We adopt a OneCycle schedule [27] with a starting learning rate of 0.0003 and a maximum learning rate of 0.003. For loss functions, we utilize Focal Loss [28] for classification, smooth L1-Loss for bounding box regression, and Cross Entropy loss for rotation. Our RadarPillars use a backbone size of  $C = 32$  for all encoder stages, a hidden dimension of  $E = 32$  for PillarAttention, and  $v_{r,x}$ ,  $v_{r,y}$  as additional features. This puts RadarPillars at only  $0.27 M$  parameters with  $1.99 GFLOPS$ . Our pillar grid size is set to  $320 \times 320$  for 1-, 3- and 5-frame data. We set the concatenated feature size for the detection head to  $160 \times 160$ . We implement our network in the OpenPCDet framework [29], training all models on a Nvidia RTX 4070 Ti GPU with a batch size of 8 and float32 data type.

Our ablation studies in Sections IV-B, IV-C and IV-D are carried out for 1-frame detection. In each ablation study, we only study the impact of a single method. We cover the combination of our methods to form our final model in Section IV-A.

### A. RadarPillars

We present a comprehensive evaluation of our RadarPillars against state-of-the-art networks, detailing results in Table

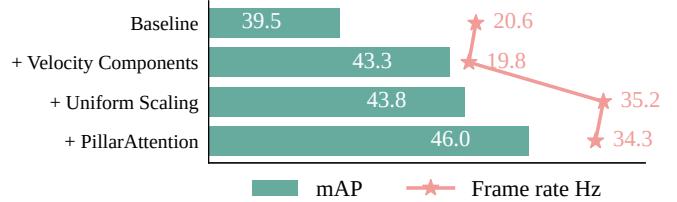


Fig. 4: Combination of our proposed methods forming RadarPillars, in comparison to the baseline PointPillars [3]. Results for 1-frame object detection precision for the entire radar area on the View-of-Delft dataset [2]. The frame rate was evaluated on a Nvidia AGX Xavier 32GB.

I. Given the nascent stage of 4D radar detection, we establish additional benchmarks by training LiDAR detection networks for 4D radar data: PV-RCNN [32], PV-RCNN++ [33], PillarNet [35], Voxel-RCNN [31], and SECOND [34]. For these networks, we utilize the settings as Palfy *et al.* [2] used in their adaption of PointPillars [3]. Following other work, we evaluate frame rate performance on an Nvidia Tesla V100, Nvidia RTX 3090 and Nvidia AGX Xavier 32GB.

Our comparison highlights the remarkable superiority of our RadarPillars over the current state-of-the-art. These findings firmly establish RadarPillars as a lightweight model with significantly reduced computational demands, outperforming all other 4D radar-only models. While RadarPillars matches SMURF [4] in precision (with a margin of +0.8 for the driving corridor and -0.3 for the entire radar area), its advantage in frame rate is seismic, outperforming SMURF by a factor of 2.73. Considering this difference, SMURF would likely struggle to achieve real-time capabilities on an embedded device such as an Nvidia AGX Xavier, whereas RadarPillars excels in this regard. In the 3-frame and 5-frame settings, RadarPillars performs on-par or better than the state of the art in terms of precision, while exceeding other methods in terms of frame rate. However, accumulating radar frames requires trajectory information. The accumulated data is already preprocessed in the View-of-Delft dataset. In a real-world application, waiting on and processing frames of multiple timesteps before passing them to the network, would incur a delay in detection predictions. Such a delay could be detrimental depending on the application, such as reacting to a pedestrian crossing the street. Because of this, the 1-frame setting can be considered as more meaningful. Despite its simplicity compared to complex network architectures, RadarPillars sets a new standard for performance, even surpassing established LiDAR detection networks in both frame rate and precision. Compared to PointPillars, our network showcases a significant improvement in both mAP (+6.5) and frame rate (+13.7 Hz), accompanied by a drastic reduction in parameters (-94.4 %) from  $4.84 M$  to  $0.27 M$ . Furthermore, the computational complexity is reduced by (-87.9 %) from  $16.46 GFLOPS$  to  $1.99 GFLOPS$ . These results establish RadarPillars as the new state-of-the-art for 4D radar-only object detection in terms of both performance and run-time. While they are not directly com-

TABLE II: Comparison of the results for the features that are additionally generated from the radial velocities.

X,Y,Z, RCS	Features						Entire Area				Driving Corridor								
	$v_{rad}$	$v_r$	$v_{rel,xy}$	$v_{rel,xy}$	$v_{rel,m}$	$v_{pm}$	$v_{rel,ym}$	$v_{rel,ym}$	mAP	Car	Pedestrian	Cyclist	mAP	Car	Pedestrian	Cyclist			
									$AP_{50}$	$AP_{25}$	$AP_{25}$	$AP_{25}$				$AP_{50}$	$AP_{25}$	$AP_{25}$	
✓	✓	✓							39.5	30.2	25.6	62.8	60.9	61.5	36.8	84.5			
✓	✓								32.3	33.7	20.1	43.2	58.6	70.8	29.6	75.4			
✓		✓							38.6	33.7	24.7	57.6	62.9	68.8	37.0	83.0			
✓	✓	✓	✓	✓					41.8	37.9	26.0	61.4	64.2	69.4	37.8	85.4			
✓	✓	✓	✓	✓	✓				43.3	37.4	29.6	62.9	65.9	70.6	40.9	86.0			
✓	✓	✓	✓	✓	✓	✓			37.9	31.7	25.9	56.1	61.6	68.6	38.5	77.8			
✓	✓	✓	✓	✓	✓	✓			39.9	36.8	24.8	58.0	62.0	69.5	35.4	81.1			
✓	✓	✓	✓	✓	✓	✓			39.3	35.7	25.7	56.7	63.8	70.3	37.1	84.1			
✓	✓	✓	✓	✓	✓	✓			39.6	36.0	26.7	56.2	63.8	70.0	37.5	83.8			
✓	✓	✓	✓	✓	✓	✓			38.4	31.6	26.5	57.1	61.4	66.9	36.9	80.3			
✓	✓	✓	✓	✓	✓	✓			39.9	31.2	28.9	59.6	62.0	65.2	39.1	81.8			
✓	✓	✓	✓	✓	✓	✓			38.6	33.1	26.7	56.0	64.3	69.7	39.2	84.0			

parable, RadarPillars achieves competitive results to multi-sensor methods fusing camera and radar data for detection. Interestingly, RadarPillars outperforms the precision of GRC-Net [9] without fusing image data and CM-FA [10] which uses LiDAR point clouds for training.

RadarPillars' performance stems from several key design choices, notably the decomposition of the compensated radial velocity  $v_r$  into its  $x$  and  $y$  components as additional features, choosing a uniform channel size of  $C = 32$  for all stages of the backbone, and incorporating PillarAttention. Figure 4 illustrates the impact of each method on model performance. Notably, the introduction of  $x$  and  $y$  components of radial velocity yields a substantial mAP boost of (+3.8) without significant runtime overhead. We theorize that this leads to more meaningful point feature encoding, before these are grouped and projected, in turn leading to more meaningful pillar features. Furthermore, downscaling the backbone architecture through uniform scaling significantly enhances frame rate without compromising performance. Finally, PillarAttention contributes to an increase in mAP (+1.6) at only a slight runtime increase. We delve into our design choices through the subsequent ablation studies.

### B. 4D Radar Features

The results of our proposed construction of additional point features from the radial velocities are shown in Table II. For a description of our methods, please refer to Section IV-B. The first finding of note is, that the performance of the model is strongly dependent on the compensation of the radial velocity  $v_{rel}$  through ego motion (leading to  $v_r$ ). If  $v_r$  is not used as a feature, the detection precision of the model drops by 7.2. On the other hand, if the relative radial velocity  $v_{rel}$  is not used as a feature, the precision of the model only drops by 0.9. This can naturally be explained by the fact that the measured relative radial velocities  $v_{rel}$  are dependent on the ego motion of the recording vehicle. As the vehicles driving velocity changes during a recording, the characteristic velocity profiles of the road users are distorted by their relative measurement to the vehicle velocity. Furthermore, the results show that the decomposing of the radial velocities  $v_{rel}$ ,  $v_r$  into their respective  $x$  and  $y$  components lead to an increase in performance. The best

TABLE III: Comparison of different implementations of self-attention on the validation split of the View-of-Delft dataset.

Method	Entire Area				Driving Corridor			
	mAP	$AP_{50}$	$AP_{25}$	$AP_{25}$	mAP	Car	Pedestrian	Cyclist
None (Baseline)	39.5	30.2	25.6	62.8	60.9	61.5	36.8	84.5
Point-Attention (unmasked)	40.6	36.6	25.9	59.4	62.4	68.6	36.6	81.9
Point-Attention (masked)	41.6	37.8	26.7	60.4	63.4	69.6	37.4	83.1
Pillar-Attention	<b>42.9</b>	38.1	28.1	62.4	<b>64.2</b>	68.5	40.0	84.2
Feature-Attention	41.3	37.7	28.2	58.1	62.5	70.5	34.2	82.9

result is achieved by constructing the  $x$  and  $y$  components of only the compensated radial velocity  $v_r$ , which leads to a significant increase in precision of 3.8. Further processing of the velocities in the form of constructing an offset feature (denoted by the subscript  $m$ ) to the average values within a pillar does not show any clear improvements.

### C. PillarAttention

We investigate how our PillarAttention layer described in Section III-B affects detection precision. Equal settings are used for all layers for fair comparison. The experimental results are summarized in Table III.

We first contrast PillarAttention with what we describe as PointAttention. In PointAttention point features are grouped (but not projected) by their pillar index, with each group zero-padded to a group size of 10. Then, standard self-attention inside a transformer layer is applied to these point features, before pillar-projecting the result as pillar-features. To assess the impact of padding, we also train a masked version of PointAttention. In both PointAttention versions, self-attention is computed among all radar points in the point cloud, treating each point as a token, similar to PillarAttention. Thirdly, we compare with implementing self-attention between the concatenated features of all encoder stages, before processing by the detection head and similar in concept to SRFF [6]. In this scenario, the concatenated feature maps are flattened prior to self-attention calculation.

The results of Table III show that Pillar-Attention leads to the greatest increase in detection precision. Using attention directly on the points is less beneficial for both the masked and unmasked versions of PointAttention. We theorize that a cause of this could be that, while there is some ordering by pillar grouping, the points inside one of these groupings are still unordered. In contrast, Pillar-Attention has a defined orders for every token, while still providing fine grained detail. This result is shared by the use of late attention, indicating that a global receptive field is advantageous early on for 4D radar data. In a further experiment, we investigate the choice of the hidden dimension  $E$  of the PillarAttention layer. The results from Table IV show that the best precision is achieved with an embedding dimension of  $E = 128$  channels.

### D. Backbone Scaling

We study the uniform scaling strategy of RadarPillars, setting all three encoder stages to the same amount of

TABLE IV: Results for different embedding dimensions  $E$  of the PillarAttention module on the validation split of the View-of-Delft dataset.

Dim.	Entire Area				Driving Corridor			
	$mAP$	Car	Pedestrian	Cyclist	$mAP$	Car	Pedestrian	Cyclist
	$AP_{50}$	$AP_{25}$	$AP_{25}$		$AP_{50}$	$AP_{25}$	$AP_{25}$	
$E = 16$	37.6	33.3	23.5	56.0	61.6	68.6	32.0	84.1
$E = 32$	39.6	36.3	23.4	59.1	62.6	69.7	34.7	83.6
$E = 64$	39.9	36.1	24.9	58.6	62.7	69.1	37.4	81.5
$E = 128$	<b>42.9</b>	38.1	28.1	62.4	<b>64.2</b>	68.5	40.0	84.2
$E = 256$	39.1	33.8	25.5	58.0	60.7	67.4	35.8	78.9
$E = 512$	37.5	33.0	20.3	59.1	59.9	68.9	29.4	81.3

channels  $C$ . This we compare to the common practice of doubling the amount of channels with each encoder stage, as is the case in PointPillars, leading to a backbone with  $C$ ,  $2C$  and  $4C$  channels. Experimental results are shown in Table V. Uniform scaling with  $C = 64$  leads to a precision increase of 3.1, outperforming the double-scaling baseline with  $C = 64$ , while reducing network parameters by ( $-83.6\%$ ) from  $4.84 M$  to  $0.79 M$ . This also results in reduced computational effort, which is reflected in the frame rates achieved. The increase in precision is consistent across all choices of  $C$ , indicating that uniform scaling is superior for 4D radar data. With real-time performance in mind we choose  $C = 32$  for RadarPillars, reducing precision by 0.6, but increasing the frame rate by  $15.5\text{ Hz}$  on a Nvidia AGX Xavier 32GB. This further reduces parameter count to  $0.26 M$ .

We theorize that this phenomenon stems from the extreme sparsity of radar data, providing only little input for a neural network. As such, the network can only form weak connections during training, leaving most feature maps without impact. To provide additional context to strengthen this assumption, we perform a weight magnitude analysis. For this analysis, we first clip the weight values at a minimum of 0, as ReLU is used as the activation function in RadarPillars. Next, we divide by the maximum weight in the entire layer. This scales the weights of all layers independently of each other into a normalized magnitude range between 0 and 1. We then remove dead weights by using a minimum magnitude threshold of 0.001. The remaining weight magnitudes are plotted in a box plot to enable comparison independent of parameter counts in Figure 5. Outlier weights are not depicted for visual clarity, as these number in the thousands. The box plot shows that smaller backbones with less channels learn stronger connections, offering a possible explanation as to why a reduced parameter count is so beneficial. In conclusion, adapting LiDAR networks requires the downscaling of their backbones to adapt to the sparsity of the 4D radar data, as shown by the effectiveness of RadarPillars. In preliminary investigations we have also tried removing an encoder stage or adding an additional stage, however both were detrimental to performance and using three encoder stages was optimal for precision.

TABLE V: We show that the uniform backbone scaling of RadarPillars outperforms traditional double-scaling in terms of precision and frame rate. All of our choices of channels  $C$  outperform this double-scaling strategy with  $C = 64$ .

Features	$M$	Entire Area				Driving Corridor				F.Rate
		$mAP$	Car	Pedestrian	Cyclist	$mAP$	Car	Pedestrian	Cyclist	
			$AP_{50}$	$AP_{25}$	$AP_{25}$		$AP_{50}$	$AP_{25}$	$AP_{25}$	
(512, 512, 512)	37.12	40.2	32.1	26.3	62.2	63.8	67.5	39.5	84.5	4.2
(256, 256, 256)	9.72	40.9	33.5	26.8	62.3	64.7	70.2	37.9	85.9	9.3
(128, 128, 128)	2.74	41.9	36.4	27.1	62.3	64.6	70.2	38.8	84.6	17.7
(64, 64, 64)	0.79	<b>42.6</b>	36.3	28.6	63.0	<b>65.0</b>	69.1	39.7	86.1	28.3
(32, 32, 32)	0.26	42.0	33.4	30.4	62.3	64.8	69.1	42.6	82.7	36.1
(16, 16, 16)	0.11	40.2	31.8	28.4	60.5	61.0	65.8	38.8	78.3	35.9
Baseline [3]	4.84	39.5	30.2	25.6	62.8	60.9	61.5	36.8	84.5	20.6
(64, 128, 256)										

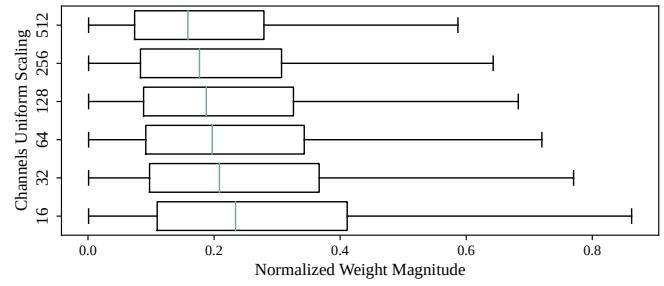


Fig. 5: Weight magnitude analysis comparing various channel sizes for uniformly scaling RadarPillars. Results show that the weight strength increases with decreased network size. This visualization excludes dead weights and outliers.

## V. CONCLUSION

This work introduces RadarPillars, our novel approach for object detection utilizing 4D radar data. As a lightweight network of only  $0.27 M$  parameters and  $1.99\text{ GFLOPS}$ , our RadarPillars establishes a new benchmark in terms of detection performance, while enabling real-time capabilities, thus significantly outperforming the current state-of-the-art. We investigate the optimal utilization of radar velocity to offer enhanced context for the network. Additionally, we introduce PillarAttention, a pioneering layer that treats each pillar as a token, while still ensuring efficiency. We demonstrate the benefits of uniform scaled networks for both detection performance and real-time inference. Leveraging RadarPillars as a foundation, our future efforts will focus on enhancing runtime by optimizing the backbone and exploring anchorless detection heads. Another avenue of research involves investigating end-to-end object detection using transformer layers with PillarAttention exclusively or adapting promising LiDAR methods [38], [39] to benefit radar. Lastly, we propose the potential extension of RadarPillars to other sensor data modalities, such as depth sensing or LiDAR.

## ACKNOWLEDGMENT

This research was partially funded by the Federal Ministry of Education and Research Germany in the project PreciRaSe (01IS23023B).

## REFERENCES

- [1] M. Fürst, O. Wasenmüller, and D. Stricker, “LrpD: Long range 3d pedestrian detection leveraging specific strengths of lidar and rgb,” in *Intelligent Transportation Systems Conference (ITSC)*, 2020.
- [2] A. Palffy, E. Pool, S. Baratam, J. F. Kooij, and D. M. Gavrila, “Multi-class road user detection with 3+ 1d radar in the view-of-delft dataset,” *Robotics and Automation Letters (RA-L)*, 2022.
- [3] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [4] J. Liu, Q. Zhao, W. Xiong, T. Huang, Q.-L. Han, and B. Zhu, “Smurf: Spatial multi-representation fusion for 3d object detection with 4d imaging radar,” *Transactions on Intelligent Vehicles (T-IV)*, 2023.
- [5] Q. Yan and Y. Wang, “Mvfan: Multi-view feature assisted network for 4d radar object detection,” in *International Conference on Neural Information Processing (ICONIP)*, 2023.
- [6] L. Ruddat, L. Reichardt, N. Ebert, and O. Wasenmüller, “Sparsity-robust feature fusion for vulnerable road-user detection with 4d radar,” *Applied Sciences*, 2024.
- [7] L. Zheng, S. Li, B. Tan, L. Yang, S. Chen, L. Huang, J. Bai, X. Zhu, and Z. Ma, “Rcfusion: Fusing 4d radar and camera with bird’s-eye view features for 3d object detection,” *Transactions on Instrumentation and Measurement (TIM)*, 2023.
- [8] W. Xiong, J. Liu, T. Huang, Q.-L. Han, Y. Xia, and B. Zhu, “Lxl: Lidar excluded lean 3d object detection with 4d imaging radar and camera fusion,” *Transactions on Intelligent Vehicles (T-IV)*, 2023.
- [9] L. Fan, C. Zeng, Y. Li, X. Wang, and D. Cao, “Grc-net: Fusing gat-based 4d radar and camera for 3d object detection,” SAE Technical Paper, Tech. Rep., 2023.
- [10] J. Deng, G. Chan, H. Zhong, and C. X. Lu, “Robust 3d object detection from lidar-radar point clouds via cross-modal feature augmentation,” *International Conference on Robotics and Automation (ICRA)*, 2024.
- [11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [12] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, “Point transformer,” in *International Conference on Computer Vision (ICCV)*, 2021.
- [13] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, “Point transformer v2: Grouped vector attention and partition-based pooling,” *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [14] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, “Point-bert: Pre-training 3d point cloud transformers with masked point modeling,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [15] Y.-Q. Yang, Y.-X. Guo, J.-Y. Xiong, Y. Liu, H. Pan, P.-S. Wang, X. Tong, and B. Guo, “Swin3d: A pretrained transformer backbone for 3d indoor scene understanding,” *arXiv preprint arXiv:2304.06906*, 2023.
- [16] P. Sun, M. Tan, W. Wang, C. Liu, F. Xia, Z. Leng, and D. Anguelov, “Swformer: Sparse window transformer for 3d object detection in point clouds,” in *European Conference on Computer Vision (ECCV)*, 2022.
- [17] X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi, and J. Jia, “Stratified transformer for 3d point cloud segmentation,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [18] P.-S. Wang, “Octformer: Octree-based transformers for 3d point clouds,” *ACM Transactions on Graphics (TOG)*, 2023.
- [19] Z. Liu, X. Yang, H. Tang, S. Yang, and S. Han, “Flatformer: Flattened window attention for efficient point cloud transformer,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [20] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia, “Spherical transformer for lidar-based 3d recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [21] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, “Point transformer v3: Simpler, faster, stronger,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [22] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, “Sparsity invariant cnns,” in *International Conference on 3D Vision (3DV)*, 2017.
- [23] L. Reichardt, P. Mangat, and O. Wasenmüller, “Dvmn: Dense validity mask network for depth completion,” in *Intelligent Transportation Systems Conference (ITSC)*, 2021.
- [24] T. Dao, “Flashattention-2: Faster attention with better parallelism and work partitioning,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [27] L. N. Smith and N. Topin, “Super-convergence: Very fast training of neural networks using large learning rates,” in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019.
- [28] T.-Y. Ross and G. Dollár, “Focal loss for dense object detection,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [29] O. D. Team, “Openpcdet: An open-source toolbox for 3d object detection from point clouds,” <https://github.com/open-mmlab/OpenPCDet>, 2020.
- [30] S. Shi, X. Wang, and H. Li, “Pointrnn: 3d object proposal generation and detection from point cloud,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [31] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, “Voxel r-cnn: Towards high performance voxel-based 3d object detection,” in *Conference on Artificial Intelligence*, 2021.
- [32] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, “Pv-rcnn: Point-voxel feature set abstraction for 3d object detection,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [33] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, “Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection,” *International Journal of Computer Vision (IJCV)*, 2023.
- [34] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” *Sensors*, 2018.
- [35] G. Shi, R. Li, and C. Ma, “Pillarnet: Real-time and high-performance pillar-based 3d object detection,” in *European Conference on Computer Vision (ECCV)*.
- [36] Z. Lin, Z. Liu, Z. Xia, X. Wang, Y. Wang, S. Qi, Y. Dong, N. Dong, L. Zhang, and C. Zhu, “Rcbvdet: Radar-camera fusion in bird’s eye view for 3d object detection,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [37] W. Shi, Z. Zhu, K. Zhang, H. Chen, Z. Yu, and Y. Zhu, “Smiformer: Learning spatial feature representation for 3d object detection from 4d imaging radar via multi-view interactive transformers,” *Sensors*, 2023.
- [38] L. Reichardt, N. Ebert, and O. Wasenmüller, “360deg from a single camera: A few-shot approach for lidar segmentation,” in *International Conference on Computer Vision (ICCV)*, 2023.
- [39] L. Reichardt, L. Uhr, and O. Wasenmüller, “Text3daug – prompted instance augmentation for lidar perception,” in *International Conference on Intelligent Robots and Systems (IROS)*, 2024.