Radar-based 2D Car Detection Using Deep Neural Networks

Maria Dreher[1], Emeç Erçelik[1], Timo Bänziger[2], and Alois Knoll[1]

***Abstract*— A crucial part of safe navigation of autonomous vehicles is the robust detection of surrounding objects. While there are numerous approaches covering object detection in images or LiDAR point clouds, this paper addresses the problem of object detection in radar data. For this purpose, the fully convolutional neural network YOLOv3 is adapted to operate on sparse radar point clouds. In order to apply convolutions, the point cloud is transformed into a grid-like structure. The impact of this representation transformation is shown by comparison with a network based on Frustum PointNets, which directly processes point cloud data. The presented networks are trained and evaluated on the public nuScenes dataset. While experiments show that the point cloud-based network outperforms the grid-based approach in detection accuracy, the latter has a significantly faster inference time neglecting the grid conversion which is crucial for applications like autonomous driving.**

I. INTRODUCTION

In autonomous driving, the accurate detection of surrounding objects is a crucial prerequisite in order to perform safe navigation [18]. However, relying on a single type of sensor for safety-critical tasks is not sufficient as each of the sensing technologies currently available has its drawbacks and weaknesses. Thus, multimodality is required to achieve safe and robust navigation [25].

While there are numerous methods performing camera [3] [26] or LiDAR-based [17] [27] object detection, object detection in radar data has not been thoroughly addressed in the literature so far. In addition to being nearly weather independent, it is a fairly cheap, robust and easily deployable sensor that can measure velocities of targets directly. While modern automotive radar sensors generate multiple reflections per object, the resulting point cloud is extremely sparse [2] [1]. Especially compared to LiDAR data, the number of received reflections per object is very low making it difficult to accurately infer the surrounding environment. Additionally, typical automotive radar sensors have a small vertical field of view (FOV), thus they collect little information about the height of objects. Figure 1 shows an exemplary radar point cloud in contrast to LiDAR data, demonstrating the problem of sparsity.

This paper examines algorithms to detect two-dimensional objects given a sparse and noisy radar point cloud. For this purpose, the fully convolutional neural network YOLOv3 [3], which was originally designed for image object detection, is adapted to operate on radar data. This state-of-the-art network is a fast and reliable one-stage object detector (running at up to 45 FPS at 51.5 mAP), a crucial characteristic for safety-critical applications as autonomous driving. In order to apply convolutions, the radar point cloud is transformed into a grid-like structure. To the best of the authors' knowledge, this paper is the first work evaluating object detection in only radar data using CNNs. To observe the impact of this representation transformation, the proposed algorithm is compared to another, point cloud-based approach performing object detection in radar data. This closely related work presented in [1] is based on Frustum PointNets [19], a two-stage network architecture which processes point clouds directly.

(a) Radar Point Cloud (b) LiDAR Point Cloud

Fig. 1: Two-dimensional Radar point cloud in (a) is much sparser compared to the three-dimensional LiDAR point cloud in (b). From [18].

While most publications focusing on radar sensors recorded their own, simplified dataset under controlled conditions [1] [5] [6], this paper provides results on the publicly available nuScenes dataset [18]. The approach presented in [1] is replicated and trained with nuScenes data as well to enable direct comparison.

II. RELATED WORK

In contrast to other typical automotive sensors, radar provides important features like the velocity of detected objects in almost all weather conditions. However, the data is typically noisier compared to LiDAR point clouds and especially the captured position values can be inaccurate. This leads to reflections measured next to the actual object instead of representing the object's precise location [1]. Further, reflections can occur not only at contour points but also at not directly visible parts of objects [12]. For instance, the radar signal can be deflected at the ground and reflected at the bottom of a vehicle. The radar cross section (RCS) information depends on the material of the reflecting target and thus provides valuable information about the object. However, the RCS value is highly affected by

[1]Maria Dreher, Emeç Erçelik, and Alois Knoll are with Chair of Robotics, Artificial Intelligence and Real-time Systems, Technical University of Munich, 85748 Garching b. München, Germany maria.dreher@tum.de, emec.ercelik@tum.de, knoll@mytum.de

[2]Timo Bänziger is with Automated Driving Research at MAN Truck & Bus SE timo.baenziger@man.eu

the orientation of the object resulting in a wide range of intensity values from the same kind of object [28]. Despite the described challenges, recent improvements in range and angular resolution of modern radar systems make the sensor a valuable source for the detection of surrounding objects [1]. This section first reviews object classification and bounding box estimation in radar data as two individual problems and then presents a recently published approach combining both tasks and performing object detection using a single radar sensor. Finally, recent studies of radar fusion are included for completeness.

A. Object Classification

Inspired by recent advances in image classification, [4] [6] [5] apply convolutional neural networks (CNN) to classify static objects in occupancy grids, created from accumulated radar data. While the network in [4] assigns class labels to objects, [6] demonstrates that semantic knowledge can be gained from radar measurements. However, CNNs require huge amount of data to train a generalized model. Feature-based classification, on the other hand, achieves good performance with much less data. Therefore, the CNN presented in [4] is combined with a random forest classifier in [5] to improve results. The combined classifier outperforms both individual classification methods. [10] and [11] perform solely feature-based classification focused on moving objects. They define specific velocity and range profiles for pedestrians and vehicles and extract features based on their echo signal characteristics. A support vector machine is then used to classify the extracted features. While feature-based approaches achieve great results, the performance of these algorithms depends on the extraction of relevant features which can be difficult.

B. Object Localization

Bounding box estimation is mostly done by iteratively rotating and adjusting a rectangular [8] or L-shaped [12] model to fit clustered radar reflections. These algorithms are simple and show great results in most cases. However, their performance highly depends on the results of the clustering algorithm. Thus, [9] proposes a template fitting approach as an alternative to oriented bounding box algorithms. In scenarios where only one side of the vehicle is visible to the radar, the template matching approach outperforms the oriented bounding box algorithms while achieving similar performance in standard cases.

C. Object Detection

While there are several publications in the literature covering either object classification [6] [7] or bounding box estimation [8] [9] in radar data, the approach presented in [1] is – to the authors' knowledge – the only work combining both and applying object detection relying solely on radar point clouds. While their two-stage network architecture based on Frustum PointNets [19] shows the great potential of radar sensors for object detection, the presented results are obtained using a severely simplified scenario with only one object in every radar measurement. As described in [1], the data was recorded on a test track and the object of interest was always the same, single target vehicle. Since the recorded dataset was not published, their results cannot be reproduced nor directly compared. Further, they do not provide results in terms of the commonly used mean average precision (mAP) metric.

D. Radar Fusion

The release of nuScenes [18] — the first public dataset providing synchronized radar, LiDAR, and camera measurements — has encouraged the fusion of radar with camera data for 2D object detection as complementary information for images. In [29], a CNN-based camera-radar fusion is proposed to detect 2D objects on the image plane. The inclusion of distance and RCS channels of the accumulated radar data improves results slightly; however, only the fusion of filtered radar points using ground-truth labels provides a promising improvement of the mAP, even though it is not possible to use without ground-truth labels. In [31], authors fuse camera and radar (depth, lateral and longitudinal velocities) data using CNNs for multi-task learning of 2D object detection on the image plane and semantic segmentation. The fusion network improves results comparing to the camera-only network, however, the multi-task framework decreases the accuracy for 2D object detection comparing to the single-task fusion network. [30] proposes a radar-based proposal algorithm to improve object proposals with sparse radar data for 2D object detection. From the range and range-rate information of the radar sensor, authors generate anchor boxes around all radar points, which are used as proposals for the Fast RCNN architecture. This improves speed and accuracy of detection comparing to the Selective Search algorithm for generating object proposals. However, a comparison with CNN-based proposal networks is not provided. In [34] authors compare fusion of camera with radar and fusion of camera with LiDAR on 3D detection using a small dataset. As a result, they find out that the radar combination provides better results than the LiDAR combination. However, the reasons might be having more radar points per frame than provided in one frame of the public nuScenes dataset as well as having less LiDAR points per frame comparing to nuScenes.

In summary, the approaches described above do not adequately address the problem of object detection in real-world scenarios solely based on radar sensors. Thus, in this work a new approach to process radar data with CNNs for Bird's Eye View (BEV) object detection is introduced. Furthermore, a comparison with a point cloud-based network, based on the public nuScenes dataset, is presented.

III. Method

This section presents two approaches for object detection in radar data. While the first algorithm transforms the radar data into a grid representation in order to apply CNN-based 2D object detection in BEV, the second approach based on the work in [1] predicts objects directly from point clouds.

We define the radar point cloud $P = \{p_i | i = 1, ..., N\}$ as a set of N five-dimensional points. Each reflection point $p_i = (x, y, \sigma, v_x, v_y)$ provides x- and y-coordinates, the radar cross section σ as well as the x- and y-components v_x and v_y of the ego-motion compensated radial velocity v. These components are referred to as x- and y-velocities in further sections.

To preserve the shape information in the BEV, the presented object detection algorithms predict oriented bounding boxes, instead of axis-aligned, parameterized by their center position x and y, height h and width w, yaw angle θ as well as a class.

A. Grid-based Object Detection

In contrast to images where the data lies on a regular grid, point clouds are irregular in terms of their density. In addition, being defined as unordered sets, point clouds are invariant to permutations of their elements [13]. This, in particular, makes convolutions as used for image object detection inadequate for point clouds: While images implicitly provide spatial information through their grid-like structure, the position of points in a point cloud is an explicit feature associated with each point.

However, motivated by the outstanding performance of CNN-based image object detection algorithms, 3D point clouds are typically transformed into grid-like structures in order to apply convolutions. Mostly, this is done either by projection to a 2D image plane [14] [15] or quantization into a 3D voxel grid [16] [17] [15]. As the poor elevation information of the radar measurements is neglected in this work, the two-dimensional point cloud can be easily transformed into a 2D BEV grid representation by quantization.

1) Input Representation: The representation transformation of the radar point cloud into a grid with a fixed height I_h, width I_w and cell size s is illustrated in Figure 2. Using a BEV representation, the ego-vehicle is located in the center and oriented towards the top of the grid. The radar reflections in the point cloud are transformed into grid cells according to their x- and y-position. Reflections exceeding the fixed grid size are discarded. Instead of RGB color information as provided in images, the three channels of the input grid are set to the points' RCS value (σ) as well as the velocity values in x- and in y-dimension. Cells where no point was mapped to are set to zero in all three channels.

The original implementation of YOLOv3 [3] operates on squared images by resizing the input. In automotive scenarios, when represented in BEV with the vehicle located in the center, the height of the grid will most commonly be larger than its width due to the shape of a road. In addition, resizing the input grid distorts the spatial locations of the few reflection points provided by the radar. Padding, on the other hand, adds even more empty cells to the already sparse grid. Thus, the network presented in this paper operates directly on rectangular sized inputs of shape $I_h \times I_w$ without resizing to a square format.

2) Network Architecture: We adapt the architecture of the fully convolutional network YOLOv3 [3] for 2D object detection in radar data. The object detection system divides the input into an $O_h \times O_w$ output grid. Caused by the rectangular input grid and in contrast to the original implementation, the output grid of our network is not of square size. Each grid cell in the output grid predicts B bounding boxes with the x- and y-positions of the center, height h and width w as well as yaw angle θ of each box. The orientation estimation is not included in the original YOLOv3 architecture but is essential to preserve the correct shape of objects in the BEV. Furthermore, the network outputs k class probabilities for every box and an additional objectness score which describes how confident the network is that there is an object located in the cell. The final shape of the resulting output grid equals to $O_h \times O_w \times B \times (6 + k)$. The input and output grid and their corresponding channels are illustrated in Figure 3.

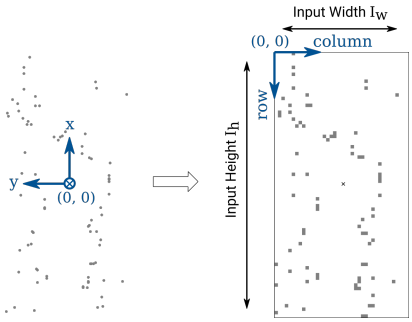

Fig. 2: Radar point cloud in ego vehicle coordinates (left) is transformed to a two-dimensional grid representation (right).

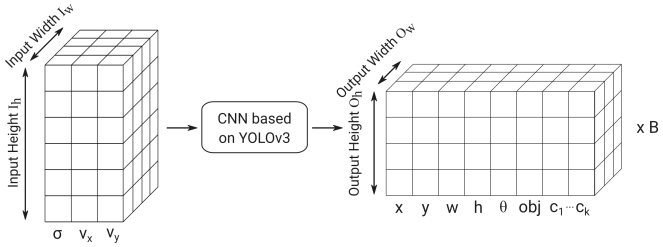

Fig. 3: Overview of input and output grid.

3) Learning and Inference: We regress the position and size of bounding boxes as described in [3]. The additional yaw angle θ is normalized to range from -1 to 1 and then predicted directly. To ensure values in this range, the hyperbolic tangent activation function is applied to the predicted yaw angle. Finally, we optimize the following multi-part loss function:

$$
\begin{aligned}
L = & \sum_{i=0}^{I_h I_w} \sum_{j=0}^{B} 1_{ij}^{obj} \lambda_{ij} [(\hat{x}_{ij} - x_{ij})^2 + (\hat{y}_{ij} - y_{ij})^2 \\
& + (\hat{h}_{ij} - h_{ij})^2 + (\hat{w}_{ij} - w_{ij})^2 + (\hat{\theta}_{ij} - \theta_{ij})^2] \\
& + \sum_{i=0}^{I_h I_w} \sum_{j=0}^{B} H(1_{ij}^{obj}, obj_{ij}) \\
& + \sum_{i=0}^{I_h I_w} \sum_{j=0}^{B} 1_{ij}^{obj} \sum_{l=1}^{k} H(\hat{c}_l, c_l)
\end{aligned}
\tag{1}
$$

The first term calculates the loss for the five bounding box parameters using the sum of squared error where $\hat{*}_{ij}$ are ground-truth values and $*_{ij}$ are the predictions. The indices indicate the grid cell position i and the anchor box j. As the prediction of box parameters should be only penalized if there is a ground-truth object assigned, the summed error term is masked using 1_{ij}^{obj}. This parameter equals to 1 when there is a ground-truth object in cell i and anchor box j is the anchor that has the highest intersection over union (IoU) with it. It is zero otherwise. The weight λ_{ij} penalizes errors in small boxes more than in large ones. The second part of the loss function determines the objectness loss. Following [3], the loss is calculated using the binary cross-entropy. Further, the last term of the loss function describes the classification loss. While the network predicts class scores for k classes, the classification loss is still obtained using the binary cross-entropy with a sigmoid activation function. Hence, the binary cross-entropy loss is calculated for each class. This enables multilabel classification, i.e. an object can have several classes (e.g. $truck$ and $vehicle$). Following [3], no softmax function is used.

B. Point Cloud-based Object Detection

A representation transformation of unordered point clouds into a regular grid makes convolutional operations applicable to the data but leads to redundant computations especially for sparse radar point clouds. Instead, the network described in this section based on [1] directly operates on point clouds and thereby preserves the actual information provided by the radar sensor without adding unnecessary data.

1) Input Representation: To reduce the search space and efficiently propose regions of interest the point cloud is divided into several patches as described in [1]. A patch with a fixed length and width is created around each radar reflection in the point cloud. As illustrated in Figure 4, the size of the patch is chosen such that it covers the whole object. This approach generates multiple proposals for the same object as well as clutter patches, i.e. patches where no object is located inside. Following [19] and [1], the patches are normalized to a center view as illustrated in Figure 5b. The patch is rotated such that the x-axis points in the direction of the radar reflection which was used to create the patch improving the rotation-invariance of the algorithm. The origin of this coordinate system is still at the position of the ego vehicle. Finally, the center view normalized patch data is used as input for the network. To ensure a fixed number of input points, the radar reflections within a patch are sampled to n points.

2) Network Architecture: The network architecture is similar to the *v1* model proposed in [1] which is based on [13]. As illustrated in Figure 6, the object detection system consists of four different networks. First, a classification network assigns a class to every patch. The predicted class is either one of $k-1$ object classes or class *clutter*, i.e. there is no object inside the patch. The network uses a multilayer perceptron to extract features of n points and generates global features by max pooling them. Based on the global

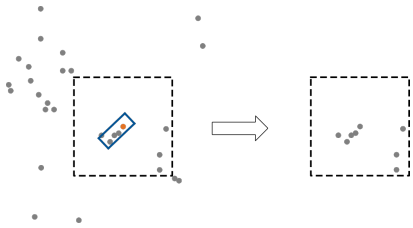

Fig. 4: Generation of regions of interest. A patch is created around every radar reflection (orange). Points located inside the patch are used as input data.

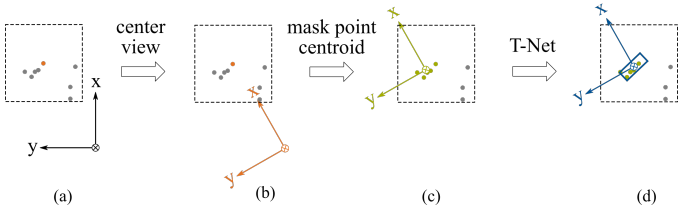

Fig. 5: Coordinate systems. (a) Ego vehicle. (b) Rotation to center view. (c) Translation to local mask coordinate system (origin is centroid of masked points). (d) Object coordinate system (origin is center predicted by T-Net).

information, fully connected layers are used to determine the class of the patch. The classification network finally outputs k class scores for every patch.

If classified as an object class (i.e. not clutter) the segmentation network determines which points in the patch are part of the object by predicting a two-dimensional mask. The specific class of the object, however, was already predicted by the classification network. The segmentation network combines both previously extracted local and global features to predict scores indicating whether a point is an object or a clutter reflection. After segmentation, the points classified as object are masked and the masked points are then normalized to improve translational invariance of the algorithm. As illustrated in Figure 5c this is achieved by a transformation into a local coordinate system which has its origin at the centroid of the masked points. To ensure a fixed number of input points for the bounding box estimation network, the object points are sampled to m points per object patch. Further, a preprocessing transformer network (T-Net) as described in [19] estimates the center given only spatial information (x and y-coordinates) of the m masked object points. The object points are then transformed into an object coordinate system according to the predicted center (see Figure 5d). Finally, a box regression PointNet similar to the one proposed in [19] predicts amodal 2D bounding boxes, i.e. boxes that comprise the entire object even if the radar sensor captures only part of it.

As in [19], the box size as well as the yaw angle is predicted following a hybrid classification and regression approach. Using N_s pre-defined size templates (anchor boxes), the network classifies the size of the box as one of the defined categories yielding N_s class scores. For each size category residuals are additionally predicted creating another $2 \times N_s$

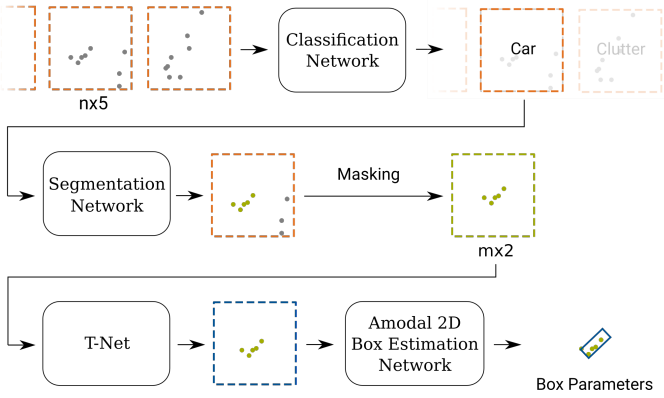

Fig. 6: Overview of the object detection system. While classification and segmentation use the five-dimensional points, only spatial information (x and y-coordinates) is exploited for box center prediction by T-Net and bounding box estimation. Colors of patches indicate the coordinate system in accordance to Figure 5.

values (one for each dimension). The assigned size template class combined with the estimated residuals in x- and y-dimensions result in the actual predicted box size. A similar approach is used to predict the yaw angle. A full rotation of 2π is divided in N_y equally split bins. Then, both class scores and angle residuals are predicted. Together with the center prediction this makes a total number of $2+3\times N_s+2\times N_y$ output values for the bounding box estimation network.

3) Learning and Inference: Following [1], the multi-task loss function described in Equation 2 is used to optimize all four networks involved simultaneously while training.

$$L = \lambda_{cls}L_{cls} + \lambda_{seg}L_{seg} + \lambda_{box}(L_{c1-reg} + L_{c2-reg} + L_{y-cls} + L_{y-reg} + L_{s-cls} + L_{s-reg} + \lambda_c L_c) \quad (2)$$

L_{cls} describes the loss for patch classification and L_{seg} for instance segmentation. Further, L_{c1-reg} and L_{c2-reg} represent the loss for center regression of T-Net and the box estimation network, respectively. The terms L_{y-cls} and L_{y-reg} denote both the classification and regression loss for yaw angle estimation, while L_{s-cls} and L_{s-reg} are for size estimation. As in [1], softmax with cross-entropy loss is used to obtain the loss for all classification and segmentation tasks and smooth-l_1 (huber) loss is used for regression. The individual loss terms are weighted by the λ parameters. Following [1], λ_{box} is set to zero for patches classified as clutter. Thus, only for patches where the classification network recognized an object a bounding box is estimated.

While the previously described loss terms completely parameterize the 2D bounding box, learning is not optimized for the IoU metric due to the separate loss terms for center, size and yaw angle. As explained in [19], in cases where the network predicts center and size accurately but fails in angle estimation, the prediction will be bad in terms of IoU solely due to the angle error. To counteract this, a novel regularization loss called *corner loss* L_c which was proposed in [19] is used to jointly optimize the three terms. It is defined as the sum of the distances between the four corners of a predicted box and the ground truth box.

IV. EXPERIMENTS & RESULTS

A. Metrics

Intersection over union (IoU) is a common metric to evaluate bounding box predictions which can be easily calculated for axis-aligned boxes. To avoid complex intersection calculations due to the orientation, we use the angle-related IoU presented in [20] to compare two boxes A and B (see Equation 3). We first calculate the conventional IoU for B and $\hat{A}$, where $\hat{A}$ is defined to have the same parameters as A except for its orientation. The yaw angle of $\hat{A}$ equals θ_B. Secondly, the obtained IoU value is multiplied with the cosine of the angular difference of the two boxes A and B.

$$ArIOU(A,B) = \frac{area(\hat{A}\cap B)}{area(\hat{A}\cup B)}\,|cos(\theta_A - \theta_B)| \quad (3)$$

A box' angle-related IoU with the ground-truth as well as its class probability determines whether the predicted bounding box is correct. Only if both values pass a defined threshold, the prediction is considered *true positive* in further precision and recall calculations. Lastly, the interpolated mAP defined in the *Pascal VOC Development Kit* [21] is obtained to evaluate the detection performance.

B. Dataset & Training

While there are several publicly available autonomous driving datasets [22] [23] [24] providing camera images and LiDAR measurements, the recently published nuScenes dataset [18] is the first public dataset providing radar point clouds and thus creates a common basis for further research on object detection in radar data.

The dataset consists of 1000 scenes recorded in Boston and Singapore, each of a duration of $20s$. Annotations are provided for keyframes which are sampled from each scene at $2Hz$ resulting in $\sim$ 40 labeled samples per scene. The complete set of scenes is divided into 850 training and validation scenes and 150 testing scenes. However, annotations are only published for the training and validation data.

For thorough evaluation of the presented methods, we only use the annotated training and validation dataset. The 850 scenes are split into $train$, val and $test$ with respect to the recording location in order to avoid overfitting to the static environment. We use 467 scenes from "Boston Seaport" and 85 scenes from "Singapore Holland Village" for training, 115 scenes from "Singapore Queenstown" for validation and 183 scenes from "Singapore One North" for test.

The nuScenes dataset provides an individual point cloud for each of the five radars expressed in a local sensor coordinate system. We fuse these individual measurements into one global point cloud comprising information about the complete environment of the vehicle. The global radar point cloud is expressed in ego vehicle coordinates which has its origin at the midpoint of the rear vehicle axle [18].

Fig. 7: Exemplary detail view of a moving object in (a) accumulated and (b) non-accumulated radar data. The object is moving in the indicated direction. While in (b) object reflections are located only inside the bounding box, the accumulated data in (a) leads to a trace of reflections behind the object.

TABLE I: 23 nuScenes categories are grouped into five classes.

Class Name	nuScenes Categories
Pedestrian	`human.*` (7), `animal`
Object	`moveable_object.*` (4), `bycicle_rack`
Two-Wheeler	`motorcycle`, `bicycle`
Car	`car`, `emergency.*` (2)
Truck	`truck`, `trailer`, `bus.*` (2), `construction`

The unlabeled radar measurements between annotated keyframes can be used to accumulate the radar data over time in order to create a denser point cloud. On average, there are six additional measurements between two keyframes in the nuScenes dataset. For accumulation, the data captured between the previous keyframe and the current keyframe is fused into one point cloud. However, accumulation distorts the shape of moving objects impeding accurate bounding box estimation. An example showing a moving object in both accumulated and non-accumulated radar data is shown in Figure 7. Both variants are examined in this paper.

The radar data was recorded with $77GHz$ long range radar sensors which can measure within a distance of $250m$. However, due to the highly crowded city scenarios in which the nuScenes data was acquired, this long range cannot be exploited as well as it could be for highway scenes. Especially reflections captured by the radars on the sides of the vehicle are often limited in range due to large buildings along the road. Thus, in this paper only points which do not exceed a distance of $\sim 20m$ from the vehicle in y-direction are included in the fused point cloud. In x-direction only points within a distance of $\sim 40m$ contribute to the final point cloud in order to focus on objects in the immediate environment.

nuScenes provides annotations for objects of 23 different categories which are combined to five distinct classes in this paper. All newly defined classes are described in Table I. Subcategories belonging to the same main category are indicated by $\star$ followed by the number of categories comprised in the main category. For example, both `bus.bendy` and `bus.rigid` are in the main category `bus`.

We augment the dataset by shifting x- and y-coordinates of radar reflections randomly and uniformly following [1]. The ego-motion compensated velocities v_x and v_y are perturbed using random Gaussian noise with zero mean and a standard deviation of 0.2. Gaussian noise with zero mean and a standard deviation of 1 is applied to perturb the RCS values.

Additionally, the point clouds are flipped along the x-axis in the BEV with a probability of 0.5. For the grid-based algorithm, the complete fused radar point cloud is flipped. For the point cloud-based detection, the individual patches are flipped before transforming into center view. In addition to the position of radar reflections, this does also affect the velocity values.

We use a grid cell size of $0.3m$ for the grid-based object detection. The selection is a trade-off between minimizing the number of empty cells and avoiding several reflections to fall into the same cell to prevent information loss as existing information in a grid cell is overwritten when another reflection is mapped to that specific cell. In accordance to the previously described maximum distance of reflections from the ego vehicle of $40m$ in x and $20m$ in y-direction, the input grid is set to be of size $256 \times 128 \times 3$. To decrease the number of empty cells, the occupied grid cells are expanded by setting the same channel values in all eight neighboring cells. Thus, instead of updating only the cell where a reflection falls into, the surrounding cells are stocked with information as well. This is beneficial for convolutional operations due to the reduction of multiplications with zero and significantly increases the fraction of occupied cells in the grid.

As a counterpart to the cell size in the grid-based network, the size of patches used as input for the point cloud-based architecture has to be fixed before training. While it is stated in [1] that a patch should comprise the complete object, the concrete width and height values that were used to obtain the presented results are not specified. We set the patch size to $8m \times 8m$ as 98.66% of boxes in the dataset have a width and height of less than $4m$. This ensures that the patch comprises the complete box even if the radar reflection in the center of the patch is located at the boundary of the object.

Following [3], we use the concept of anchor boxes and generate $B = 9$ box priors. The anchor boxes are generated using the k-means algorithm. Note that only height and width of the boxes are considered for anchor box generation. For better comparison, we also use nine anchor boxes for the point cloud-based approach, i.e. $N_s = 9$. From [1], the number of points per patch n and masked patch m for the training are chosen as 48 and 32, respectively. Further, the full rotation of 2π is divided into $N_y = 12$ equally split bins for residual-based yaw angle estimation following [19].

C. Evaluation

We evaluate the grid-based (III-A) and the point cloud-based (III-B) networks with four different sets of data. As in Table II, "Entire dataset" contains all the class members, "Only vehicles" indicates that only vehicle class members are included for training and validation. "Only moving vehicles" means considering vehicle classes labeled as moving in the

TABLE II: Detection results as mean Average Precision (mAP) with grid-based and point cloud-based networks on different data subsets.

	mAP (with IoU 0.3)	
Data Subset	Grid-based	Point cloud-based
Entire dataset	0.03 %	15.30 %
Only vehicles	4.19 %	16.28 %
Only moving vehicles	7.87 %	16.21 %
Only moving vehicles (Accumulated)	2.65 %	13.38 %

TABLE III: Detection results with varying number of points in ground-truth boxes

Min. # of points	mAP (0.3 IoU)	# of samples with at least one object
≥ 8	0.0%	2457 (11%)
≥ 6	0.0%	4550 (20%)
≥ 4	0.0%	9553 (43%)
≥ 2	7.87%	16076 (72%)

dataset. The last row of the table with "Accumulated" tag means accumulating the radar points over the frames between two successive keyframes. For the evaluation, only ground-truth boxes, enlarged with a $0.3m$ tolerance region, with at least two radar points are taken into account.

The grid-based network cannot learn to distinguish between five different classes with a almost zero mAP (Table II). YOLOv3-based CNNs are known to struggle detecting very small objects in images [32]. Due to the sparse and noisy radar data, small objects and pedestrians are not represented in the feature maps of BEV to be detected. On the other hand, the point cloud-based network performs well on the entire dataset, and its performance is better for the vehicles. It predicts bounding boxes based on individual points, which prevents the network from the loss of features for small objects. Accumulating radar reflections over time reduces performances of both networks as seen in the last row of the Table II. A possible reason is that accumulated reflections become unrelated with the considered object through time as seen in Figure 7. Accumulating smaller number of frames in addition to the object velocity compensation might be helpful for the representation problem of accumulated data and for the sparsity of the radar reflections.

We also consider training the grid-based network using ground-truth boxes with varying number of radar reflections. In Table III, we show that increasing minimum number of reflections in a ground-truth box decreases number of samples that the network can be trained with. Therefore, we only see a meaningful result when the network is trained with ground-truth boxes with at least 2 points.

Similarly, we vary the minimum number of radar reflections in vehicle and clutter patches for the point cloud-based network (Table IV). In the first row, we only include vehicle patches with at least 8 reflections and clutter patches with at least 20. As a result 92% of the patches are assigned to the vehicle class and the rest are clutter patches. Even though

TABLE IV: Detection results with varying number of points in vehicle and clutter patches

Min. # of Vehicle/Clutter Points	mAP (0.3 IoU)	# of Vehicle Patches
$\geq$8 / $\geq$20	15.81%	16043 (92%)
$\geq$6 / $\geq$16	19.61%	33014 (76%)
$\geq$4 / $\geq$12	18.59%	74946 (57%)
$\geq$2 / $\geq$8	16.21%	114135 (30%)

increasing minimum reflections per object improves the point cloud-based network's detection performance, no learning is achieved for the grid-based network as seen in Tables III and IV. Setting four or more reflections per ground-truth object as minimum causes a highly imbalanced dataset (Table III last column), which results in overfitting. In contrast, the data is oppositely imbalanced for the point cloud-based algorithm. As shown in Table IV, there are more object than clutter patches. While this initially has a positive effect on the detection results, the performance drops again when the imbalance becomes too high (i.e. 92% positive patches). Further, the problem of sparsity can be observed here: Only 1472 patches hold more than 20 reflections. It is important to keep in mind that a patch covers an area of $64m^2$.

The grid-based network regresses the orientation directly from input, whereas the point cloud-based network uses 12 equally split bins. The mean-squared error (MSE) between the estimated yaw angles and the ground-truth angles are calculated only considering the correctly detected objects as 12.04 for the direct and 7.57 for the residual regression. The residual orientation estimation shows a smaller error compared to the direct regression of the yaw angle. While these values surely depend on the overall performance of the object detector, the direct regression of the yaw angle has in fact a major disadvantage due to the singularities [33]. This means that even though π and $-\pi$ values are similar, they are too different for a neural network for regression.

The grid-based and point cloud-based networks run with 6.27 and 0.27 frames per second respectively, measured without run-time optimization on a NVIDIA Quadro GV100GL. Despite the considerably better detection results, the one-stage grid-based algorithm is significantly faster than the two-stage point cloud-based detector. However, the inference time on point cloud-based network highly depends on the number of patch proposals.

V. CONCLUSION

Object detection using radar is a challenging problem, especially in an urban environment with dense traffic and fast changing situations [12] [18], though its importance with robust measurements in changing weather and illumination conditions. Therefore, we aimed to analyze how much radar can contribute to inferring object shapes and locations in the BEV in this study. We adapted the fully-convolutional neural network YOLOv3 to operate on sparse and noisy radar data. In addition, we replicated a point cloud-based network

to have a comparison on the public nuScenes dataset. The point cloud-based network achieves a 19.61% mAP on nuScenes, whereas the YOLOv3-based network only reaches 7.87% mAP. The evaluation of the proposed object detection algorithms approves that the representation transformation of the point cloud into a grid has a large impact on the detection results. The highly imbalanced ratio between empty and occupied cells in the grid amplifies the overfitting of the network.

In conclusion, results of this study demonstrate the general potential of neural networks to perform object detection in radar data. Given the extremely challenging conditions of dense city traffic situations, especially the point cloud-based network achieves a decent detection performance. In this work only existing neural networks are adapted to radar data, but no new models are introduced, which is subject to further research. Also, the network could be extended using 3D convolutions or long short-term memory layers to take the temporal dimension of the radar data into account.

REFERENCES

[1] Danzer, A., Griebel, T., Bach, M., and Dietmayer, K. (2019, October). 2d car detection in radar data with pointnets. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 61-66). IEEE.

[2] J. Schlichenmaier, F. Roos, M. Kunert, and C. Waldschmidt. Adaptive clustering for contour estimation of vehicles for high-resolution radar. In 2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pages 14. IEEE, may 2016.

[3] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767, apr 2018.

[4] J. Lombacher, M. Hahn, J. Dickmann, and C. Wohler. Potential of radar for static object classification using deep learning methods. In 2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pages 14. IEEE, may 2016.

[5] J. Lombacher, M. Hahn, J. Dickmann, and C. Wohler. Object classification in radar using ensemble methods. In 2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pages 8790. IEEE, mar 2017.

[6] J. Lombacher, K. Laudt, M. Hahn, J. Dickmann, and C. Wohler. Semantic radar grids. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 11701175. IEEE, jun 2017.

[7] O. Schumann, C. Wohler, M. Hahn, and J. Dickmann. Comparison of random forest and long short-term memory network performances in classification tasks using radar. In 2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF), pages 16. IEEE, oct 2017.

[8] F. Roos, D. Kellner, J. Klappstein, J. Dickmann, K. Dietmayer, K. D. Muller-Glaser, and C. Waldschmidt. Estimation of the orientation of vehicles in high-resolution radar images. In 2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pages 14. IEEE, apr 2015.

[9] J. Schlichenmaier, N. Selvaraj, M. Stolz, and C. Waldschmidt. Template matching for radar-based orientation and position estimation in automotive scenarios. In 2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), pages 9598. IEEE, mar 2017.

[10] S. Heuel and H. Rohling. Pedestrian classification in automotive radar systems. In 2012 13th International Radar Symposium (IRS), pages 3944. IEEE, may 2012.

[11] S. Heuel and H. Rohling. Pedestrian recognition in automotive radar sensors. In 2013 14th International Radar Symposium (IRS), pages 732-739. IEEE, 2013.

[12] F. Roos, D. Kellner, J. Dickmann, and C. Waldschmidt. Reliable Orientation Estimation of Vehicles in High-Resolution Radar Images. IEEE Transactions on Microwave Theory and Techniques, pages 29862993, sep 2016.

[13] C. R. Qi, H. Su, M. Kaichun, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7785. IEEE, jul 2017.

[14] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 945953. IEEE, dec 2015.

[15] C. R. Qi, H. Su, M. NieBner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and Multi-view CNNs for Object Classification on 3D Data. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 56485656. IEEE, jun 2016.

[16] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 19121920. IEEE, jun 2015.

[17] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 922928. IEEE, sep 2015.

[18] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027, 2019.

[19] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 918927. IEEE, jun 2018.

[20] L. Liu, Z. Pan, and B. Lei. Learning a Rotation Invariant Detector with Rotatable Bounding Box. arXiv preprint arXiv:1711.09405, 2017.

[21] M. Everingham and J. Winn. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit. Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep, pages 132, 2011.

[22] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research, pages 12311237, 2013.

[23] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon. KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving. IEEE Transactions on Intelligent Transportation Systems, pages 934948, mar 2018.

[24] A. Patil, S. Malla, H. Gang, and Y.-T. Chen. The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes. In 2019 International Conference on Robotics and Automation (ICRA), pages 95529557. IEEE, may 2019.

[25] H. Winner, S. Hakuli, and F. Lotz. Handbuch Fahrerassistenzsysteme. Springer Fachmedien Wiesbaden, Wiesbaden, 2015.

[26] R. Girshick. Fast R-CNN. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 14401448. IEEE, dec 2015.

[27] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D Object Detection Network for Autonomous Driving. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 65266534. IEEE, jul 2017.

[28] M. I. Skolnik. Radar Handbook (second edition). 1990.

[29] Nobis, F., Geisslinger, M., Weber, M., Betz, J., and Lienkamp, M. (2019, October). A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection. In 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF) (pp. 1-7). IEEE.

[30] Nabati, R., and Qi, H. (2019, September). RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles. In 2019 IEEE International Conference on Image Processing (ICIP) (pp. 3093-3097). IEEE.

[31] John, V., Nithilan, M. K., Mita, S., Tehrani, H., Sudheesh, R. S., and Lalu, P. P. (2019, November). SO-Net: Joint Semantic Segmentation and Obstacle Detection Using Deep Fusion of Monocular Camera and Radar. In Pacific-Rim Symposium on Image and Video Technology (pp. 138-148). Springer, Cham.

[32] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[33] Simony, M., Milzy, S., Amendey, K., and Gross, H. M. (2018). Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 0-0).

[34] Meyer, M., Kuschk, G. (2019, October). Deep learning based 3d object detection for automotive radar and camera. In 2019 16th European Radar Conference (EuRAD) (pp. 133-136). IEEE.