基于PointNet++扩展，直接处理radar detection points的deep learning network做object detection，之后再用Kalman filter做tracking的方案(即并不是直接用radar detection points作为tracking的输入，而是先生成object detection的结果，再对object detection的结果作为measurements，来做一次target tracking同时估计ID与target localization refinement)。注意：通常我们认为radar的measurements是有很大noisy的，所以不适合直接以detection作为tracking without ID的结果，换言之，对radar数据设计Object detector得到的object detection结果并不准确(或者radar signal processing直接得到的radar detection points那就更加不准确，甚至一个目标有多个points)，所以我们通常对于radar object detection之后还需要再过Bayesian filter对target的localization信息refinement。而只有Lidar这样的数据我们才直接把object detection的结果作为tracking without ID的结果，所以有的方案是对Lidar的object detection结果像radar的pipeline一样再用tracking filter去做target localization的refinement并估计ID，而有的方案就是直接对Lidar point cloud用Deep Learning同时做object detection(即只做一次target localization的估计，把object detection的结果作为tracking without ID的结果)和tracking(即仅做ID估计，没有target localization refinement)。

Detection and Tracking on Automotive Radar Data with Deep Learning

Julius F. Tilly*, Stefan Haag*, Ole Schumann*, Fabio Weishaupt*,
Bharanidhar Duraisamy*, Jürgen Dickmann* and Martin Fritzsche*

(*) *Research & Development*, Mercedes-Benz AG, Stuttgart, Germany,
Email: firstname.lastname@daimler.com

***Abstract*—Reliable tracking of road users plays a critical part on the way to safe automated driving. In this paper, a machine learning based tracking approach on radar data is presented utilizing the radar target point clouds from multiple time steps as input to detect road users and to predict their tracking information. The detection and tracking of objects is achieved by applying a combination of known feature extractors from lidar and camera detection tasks. The generated feature maps are used as input to two branches - one branch for detection and one for tracking.**

In experiments on an extensive real-world radar data set, the proposed model achieves promising results in tracking performance compared to a basic clustering and a classification assisted tracking approach.

***Index Terms*—radar, tracking, machine learning, fusion, autonomous driving, automotive, sensing**

I. Introduction

Robust tracking of road users is a key element of environment perception for automated driving. To ensure robustness, usually multiple sensor types are employed. One of those sensors is radar, which is reliable in adverse weather conditions and allows instantaneous velocity estimation of measured objects. Commonly used extended object tracking approaches on radar data consist of multiple sub-tasks such as data association, clustering, track management, and predicting object states for future time steps. Thereby, depending on the method, especially correct data association and clustering can prove quite complex and can quickly become computationally expensive (e.g. for multiple hypothesis tracking [1]).

While the machine learning based classification of radar targets is already being done [2], an end-to-end machine learning based tracking system on radar data has not been successfully explored yet. In order to solve the data association and clustering challenge, this work presents a deep neural network (Radar TrackNet) using radar point clouds from multiple time steps to detect relevant objects in road traffic and additionally output their tracking information.

The paper is structured as follows: First, an overview of related work is given, which inspired some of the methods used for the presented approach. Next, the radar data set properties and the generation of ground truth tracking information are described. The third part explains the preprocessing of radar data, the networks architecture, the training process, the generation of consistent track IDs and the calculation of tracking metrics. Finally, the evaluation of the deep learning based method in comparison to more standard tracking approaches on a test set is presented and the importance of radial velocity information is investigated.

II. Related Work

Many research teams are and have been investigating machine learning based tracking on camera data since data sets recorded for image-based tracking are publicly available. For example, [3] provides a standardized benchmark data set for multiple object tracking methods on camera data. In [4] camera-based tracking is realized by first using a Faster R-CNN [5] with ResNet [6] and Feature Pyramid Networks [7] for detection and in a second step regressing detection bounding boxes from the previous frame to the current frame to allow the association via Intersection over Union (IoU) of previous and current boxes to generate consistent track IDs.

In the domain of 3D point clouds, segmentation tasks have been investigated by [8] introducing PointNet++, which is a feature extractor to handle sparse and unordered points without additional mapping steps. Thereby, the points are grouped around center points and those groups are then further processed by parts of PointNet structure [9] to maintain the spatial relation of nearby points.

End-to-end learning for detection and tracking on 3D lidar data has been done before in [10]. Here, the lidar point cloud features are extracted by a VoxelNet [11] based approach and processed by 2D CNNs followed by an SSD detection head to output 3D bounding boxes. Bounding boxes for the current and future frames are predicted using multiple past measurement frames. The whole detection and tracking architecture shows high performance in tracking and detection tasks and manages to run at a frame rate of 33 Hz.

The tracking of road users with the assistance of a radar-based classification system has been investigated in [12]. It shows that track stability is increased and errors such as false alarms or missed tracks are reduced by including classification information.

This work uses the insights from the camera and lidar based tracking and detection approaches with deep learning and adapts those to be suitable for radar data based tracking tasks and compares these to standard and classification assisted tracking performance.

TABLE I: Distribution of the annotated targets and objects among the classes.

	Passenger Car	Pedestrian	Pedestrian Group	Two Wheeler	Large Vehicle	Clutter	Static
# labeled targets	1 922 385	424 047	934 994	237 258	913 047	1 379 757	128 828 260
in %	1.43	0.31	0.69	0.18	0.68	1.02	95.68
# labeled objects	3815	1701	1216	379	503	49 236	–
in %	6.71	2.99	2.14	0.67	0.88	86.61	–
observed times (h)	7.28	3.68	4.42	1.10	1.39	6.71	–
in %	29.60	14.98	17.99	4.48	5.63	27.31	–

III. Data Set

A. Recording

Real-world data was recorded with a test vehicle equipped with four automotive series frequency modulated continuous wave (FMCW) radars. The radar mounting positions are on the left and right front corners of the vehicle with tilt angles of 25° and 90°. The radar sensor positions and their respective field of views (FOV) are shown in Fig. 1. By having overlapping FOVs, the probability to miss objects of interest is decreased, since objects are illuminated by the radar's electromagnetic waves from different angles. The radar's characteristics are:

- Operations frequency of $77\,\mathrm{GHz}$,
- Maximum range of $100\,\mathrm{m}$,
- FOV of $\pm 60°$,
- Range accuracy of $\Delta_r = 0.15\,\mathrm{m}$,
- Doppler velocity accuracy of $\Delta_v = 0.1\,\mathrm{km\,h^{-1}}$,
- Angular accuracy ranging from $\Delta_\phi = 0.5°$ to $\Delta_\phi = 2°$.

The angular accuracy is higher for smaller azimuth angles. The radar sensors can not be triggered manually and measure independently from each other with an average time of $60\,\mathrm{ms}$ between two scans. A full scan of all four sensors is assumed to be done in $75\,\mathrm{ms}$ and further referred to as one frame, resulting in a radar frame rate of $13.33\,\mathrm{Hz}$.

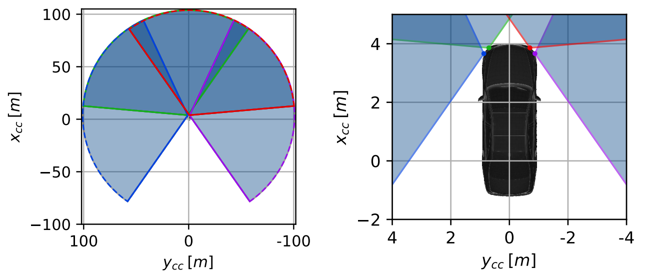


Fig. 1: The field of views of the four radar sensors. Overlapping views are visualized by more saturated areas.

In one frame, the four sensors provide a list of targets generated by a constant false alarm rate algorithm (CFAR), which can return multiple targets per real-world object. As a result, extended object tracking approaches are needed to utilize each target on one object for correct tracking.

Besides radar data, car odometry information and camera data were recorded. While the camera data helps with the annotation of the recorded radar data, the odometry allows to calculate the ego-motion compensated Doppler velocity $\hat{v}_r$ also referable to as radial velocity over ground.

B. Annotation

The manual annotation of recorded radar data was performed by assigning labels with the help of camera images to all clusters obtained from manually clustered radar targets belonging to dynamic objects. As long as a moving road user is measured by at least one radar, a cluster with an unique ID is maintained to allow the association of road users over multiple frames. If a road user is not seen for more than $0.5\,\mathrm{s}$ and then measured again, a cluster with a new unique ID is created. The clusters were annotated with seven different class labels:

- *Passenger Car* for normal cars,
- *Pedestrian* for pedestrians,
- *Pedestrian Group* for a cluster inseparable into multiple pedestrian clusters, caused by pedestrians walking close to each other,
- *Two-Wheeler* for bicycles, motorbikes and mopeds,
- *Large Vehicle* for buses, trucks, tractors and construction vehicles,
- *Static* for all targets from objects of the non-dynamic environment - without creation of clusters,
- *Clutter* as rejection class for dynamic targets from measurement noise, errors in the ego-motion compensation or ambiguities in angle and Doppler measurements.

The data set statistics with the total number of different classes, labeled targets, and total observation time is given in Tab. I.

C. Generation of Tracking Ground Truth

Although automotive radars can provide multiple detections per object and scan, further object information such as orientation, size or velocity cannot be determined directly by manual labeling. Hence, the object states are calculated iteratively for each time step where an object is visible by performing extended object tracking on manually labeled data sets. The obtained object states contain information about the object's centroid position, velocity, acceleration and yaw rate. Furthermore, the extent of the objects is approximated as two-dimensional ellipses, determined by length, width and orientation of each object. This information is calculated in three steps. In the first step, forward extended object tracking is performed with a filter that combines the unscented interacting multiple motion model (UIMM) filter and the Random Matrix Model (RMM) filter [13]. The multiple motion model approach and the integration of range-rate measurements allow extended object tracking of profoundly different object types with radar points. The measurement-to-track association task is a priori available by manual labeling. In the second step, the object trajectories are smoothed backward, according to [14], to obtain more precise paths and object behavior. The object extension is recalculated based on the corrected trajectories. The third step varies depending on the object's class types. All object types, except pedestrian groups, are assumed to have static length and width. Therefore, fixed length and width values are calculated from a weighted average of the tracking results for each object. Furthermore, we assume that two-wheelers, cars, and trucks move along their orientation of the major axis. Thereby, the orientation of these objects is corrected to the object's heading direction, if the velocity exceeds a certain threshold.

IV. Methods

A. Point Cloud Preprocessing

The point clouds measured by the radar sensors are both sparse and unordered which makes a preprocessing of the point cloud necessary. Similar to [11], the radar targets are sorted into a grid. Each grid cell has a capacity of up to N_{cp} points. If an excess of points in a cell is found, reservoir sampling is done to not exceed the capacity of the cell. Each radar target provides the following information:

- The radar targets coordinates $\vec{r}_{\mathrm{cc}} = (x_{\mathrm{cc}}, y_{\mathrm{cc}})$ in the ego-vehicle's coordinate frame,
- The ego-motion compensated radial velocity components $\vec{v}_{\mathrm{r}} = (v_{x\mathrm{r}},\, v_{y\mathrm{r}})$,
- A radar cross section (RCS) value σ.

In addition to those measured quantities, the position $\vec{r}_{\mathrm{s}} = (x_{\mathrm{s}}, y_{\mathrm{s}})$ and unit-vector orientation $\vec{o}_{\mathrm{s}} = (o_{x\mathrm{s}}, o_{y\mathrm{s}})$ of the sensor which measured the target in the ego-vehicle's coordinate frame is appended to each target's feature vector.

The feature-values of the points are rescaled to values in range $[0, 1)$ after sorting them into the grid. For the features of the target, this is achieved by applying

$$\begin{aligned}
\hat{x}_{\mathrm{cc}} &= \frac{x_{\mathrm{cc}} - x_{i_{\mathrm{cell}}}}{s_{x_{\mathrm{cell}}}} \\
\hat{y}_{\mathrm{cc}} &= \frac{y_{\mathrm{cc}} - y_{k_{\mathrm{cell}}}}{s_{y_{\mathrm{cell}}}} \\
\hat{\vec{v}}_{\mathrm{r}} &= \frac{1}{v_{\max}} \vec{v}_{\mathrm{r}} \\
\hat{\sigma} &= \frac{\sigma}{\sigma_{\max}},
\end{aligned}$$

with $(x_{i_{\mathrm{cell}}}, y_{k_{\mathrm{cell}}})$ representing the position of the left-bottom corner of a cell with the indices (i, k), $(s_{x_{\mathrm{cell}}}, s_{y_{\mathrm{cell}}})$ giving the cells extensions in x, y-direction and the maximum velocity $v_{\max}$ as well as the maximum RCS value $\sigma_{\max}$ obtained from the whole data set.

B. Radar Tracking Network

The Radar TrackNet architecture to perform tracking on a sequence of radar data grids is visualized in Fig. 2. Feature extraction is performed per cell and time step independently by a PointNet++ [8] based approach with farthest point sampling and multi-scale grouping followed by a combination of 2D convolution and max-pooling performed on the mini point clouds found in each cell. This is done in order to calculate features, providing information about structures in the cells. In detail, farthest point sampling is set to select 4 points, and for grouping, the two radii 0.15 and 0.7 with 4 and 8 neighboring points are chosen. The radii are unitless since the grouping is performed on the rescaled point features. Two 2D convolutions processing the point groups are using unit kernel sizes and filter sizes of 16 and 32 before max pooling is performed. In a final step, the tensors from the sampling groups are concatenated and processed by a fully connected layer.

After feature extraction, a tensor with dimensions $T \times W \times H \times F$ (time steps, width, height, feature count) is obtained. This tensor is further processed by an adaption of a feature pyramid network (FPN) [15], [16] to generate high-level semantics, which involves three resolution levels with each level having half the width and height of the previous level. A lower resolution feature map is generated by applying two 2D convolutions on the current feature map - the first one with a $(1, 1)$ kernel and unit stride followed by the second one with a $(3, 3)$ kernel and a stride of two. Merging of the obtained feature maps is achieved by upsampling the lower resolution map via bilinear interpolation, applying a 2D convolution with a $(3, 3)$ kernel with unit stride on the upsampled map and finally concatenating it with the higher resolution map followed by another 2D convolution with a $(1, 1)$ kernel and unit stride. This operation needs to be done twice to merge the three resolution levels starting from the lowest resolution map. The processing of the feature maps with the FPN generating high-level semantics improved the performance of the radar TrackNet when compared to omitting the FPN.

The resulting tensor is used as input for two branches performing merging steps over the time dimension independently by utilizing 3D convolutions with a kernel size of 3 for the time dimension. Depending on T, a total number of merges $n_{\mathrm{tmerge}} = {}^{(T-1)}/_{2}$ is needed. Before each time-merge, 2D convolutions over the width and height dimension are performed. In this work, $T = 5$ was chosen, so that two time-merges are necessary. The 3D convolutions filter count is set to 128. The output of the time merges further processed by a ResNet [6] with two residual blocks in the confidence branch and three residual blocks in the tracking branch. The output of the confidence branch is concatenated with the tensor in the tracking branch before being processed

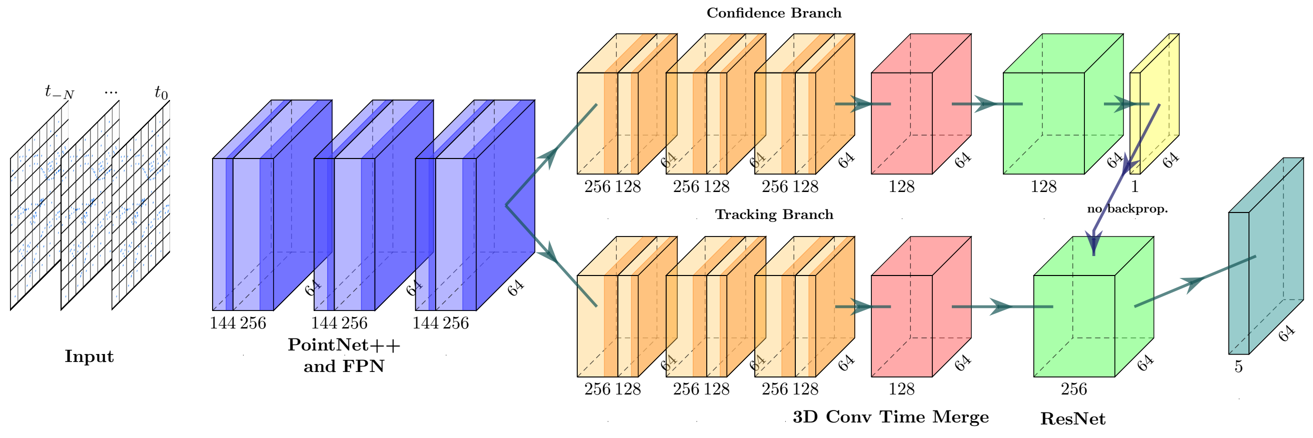


Fig. 2: Network architecture used for tracking on radar data. An example architecture utilizing measurement frames from $N = 3$ consecutive time-steps and an input grid with 64 cells in width and height is visualized.

by the tracking branch's residual blocks. The concatenation is performed by a connection that blocks backpropagation, to avoid backpropagating errors from the tracking branch to the confidence branch. This two branch approach was chosen to allow a more independent training of confidence and track information output but allow a propagation of confidence information to the tracking branch.

Overall, the network has two outputs: a confidence tensor with the dimensions $W \times H \times 1$ which contains probabilities of finding a relevant object in a cell encoded as logits and a tensor with track information such as translation and velocities per cell. The track information tensor has the dimensions $W \times H \times 5$. The network in this configuration is only able to output one track per cell, which can be problematic in scenarios where objects are really close together and consequently are located in the same cell. A possible solution is altering the output tensors to allow multiple tracks per cell, but in the scope of this work only a single track per cell was investigated.

The implementation of the network is done with the libraries TensorFlow [17] and Keras [18] in Python.

C. *Training*

The loss functions chosen for training are binary cross-entropy loss for the confidence branch and root mean squared error (RMSE) loss terms for the track information like the 2D translation and velocity. The RMSE for position and velocity are added without any weighting to obtain the final loss of the tracking branch.

A two-step process is used to train the network. First, the loss for track translation and velocity output is weighted with a factor of $^1/_{100}$ compared to the loss for confidence to focus on detecting road users before calculating further tracking information. In the second step, the learning rate is reduced by a factor of ten, and the weight for the track output loss is reset to one. This two-stepped training process improved the overall performance of the radar tracking network when compared to directly training with equal loss weights. Adam [19] was chosen as the optimizer with an initial learning rate of 0.001 while other parameters were left with their default values as proposed in [19].

D. *Track Association and Metrics*

In order to assign consistent IDs to tracks that are predicted by the network, linear Kalman filters are used. The outputs are associated with tracks already present by calculating a cost matrix with the Mahalanobis distance between each track and output. A linear sum assignment problem (LSAP) is then solved on the cost matrix to assign IDs to outputs and create or remove tracks. Tracks already present from previous frames are updated with matched velocity and translation information from the network. An overview of the track management is visualized in a flow chart in Fig. 3. With the obtained consistent IDs, it is possible to evaluate the networks performance with CLEAR MOT metrics [20] such as *Multiple Object Tracking Accuracy (MOTA)* and *Multiple Object Tracking Precision (MOTP)*. Those metrics are calculated by

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t} \tag{1}$$

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}, \tag{2}$$

with the number of track matches c_t, distance $d_{i,t}$ for track i, and number of misses m_t, false positives fp_t, mismatches mme_t and total number of ground truth tracks g_t for time t. The *MOTA* score range is $(-\infty, 1]$. A score of one signals perfect association results and negative values are obtained if the number of errors made by the tracker exceeds the number of ground truth tracks found in the scene. The value of *MOTP* gives the mean error between ground truth and the associated predicted tracks. *MOTP* should only be used as a secondary indicator for tracking performance if *MOTA* exceeds zero since the *MOTP* value can be close to zero even if the tracker misses nearly all objects and, for example, only matches one with good precision.

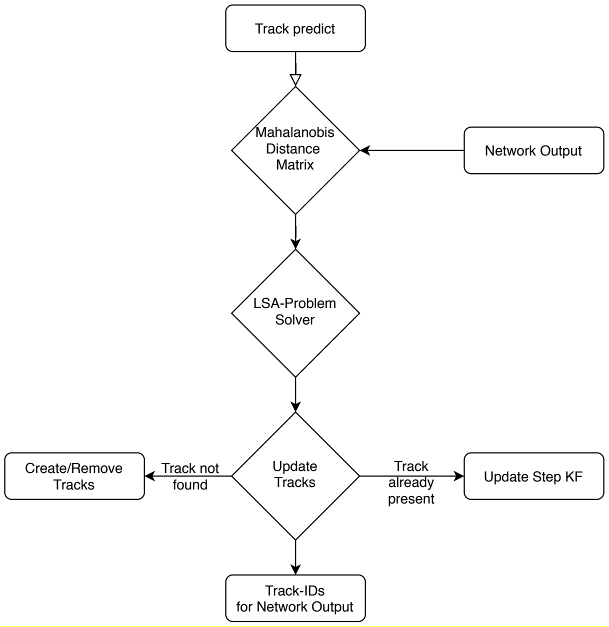


Fig. 3: Flow Chart of track management to assign IDs to the network output.

V. Results

A. Evaluation

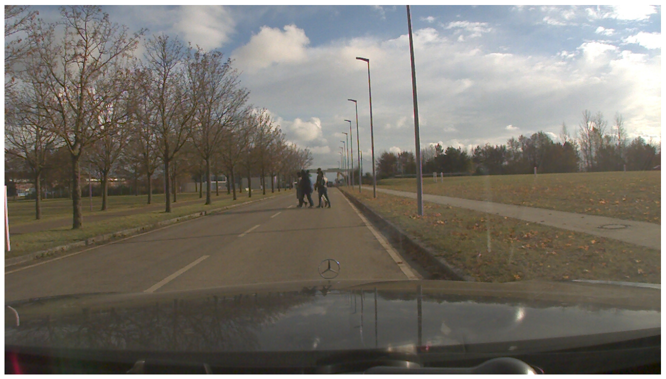
The *MOTA* and *MOTP* scores evaluated on the test set are given in Tab. II. The test set consists of five scenarios. It should be noted that the classification assisted tracker (CA tracker) and the basic clustering tracker (BC Tracker) use the same UIMM-RMM filter, as described in Sec. III-C, that is used for the ground truth generation as well, which gives them an advantage in comparison to the Radar TrackNet. Basic Clustering stands for utilizing grid-based DBSCAN with parameters adjusted to the sensor resolution [13]. Cluster to Track association is performed with the global nearest neighbour approach. New tracks are created from non-assigned clusters that contain at least two measurements that exceed a velocity threshold. Classification Assisted Tracking [12] utilizes clustering only for track creation. Radar detections are directly assigned to existing tracks exploiting position, velocity and classification information that was gained from a point net. Further, the scores for the Radar TrackNet are calculated by directly taking the velocities and positions from the TrackNet output. Only the track IDs are obtained from the approach described in Sec. IV-D.

TABLE II: Tracking scores for the whole test set for the presented network, a classification assisted tracker (CA Tracker) [12] and a basic clustering tracker (BC Tracker).

	MOTA	*MOTP*
Radar TrackNet	62.0 %	5.09
BC Tracker	52.4 %	12.45
CA Tracker	47.2 %	10.06

The Radar TrackNet shows the highest total performance in terms of *MOTA* and *MOTP* values compared to the CA tracker and the BC Tracker when evaluated on all scenarios of the test set. The *MOTP* value is calculated by using the euclidean distance of position and velocity as the distance measure between ground truth tracks and track hypothesis. The input of positions is in m and velocities in $\mathrm{m/s}$.

A better assessment of the different tracking approaches is obtained when the scores for each scenario are calculated independently. The scores for the different scenarios are given in Tab. III separately and show that scenarios vary in difficulty for the tracking approaches. In the first scenario, for example, a group of pedestrians walking on the street in front of the ego-vehicle, as shown in Fig. 4, proves difficult for the CA Tracker, which only reaches a *MOTA* score of 35.4 % while TrackNet and BC Tracker reach about 29 percentage points more.



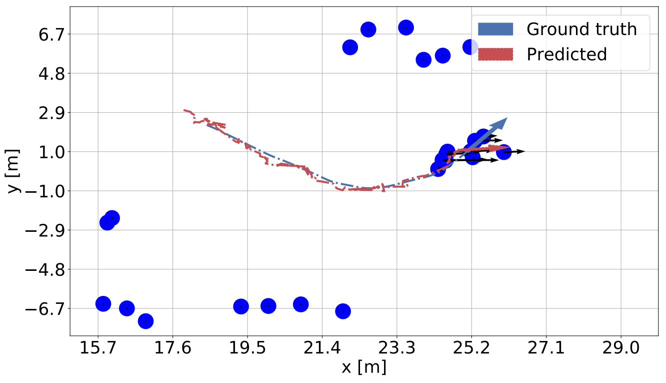


Fig. 4: Scenario 1: A group of pedestrians walking in front of the ego vehicle. The ground truth track is visualized in blue and the predicted track in red. Black arrows indicate the measured Doppler velocities of the radar targets which are depicted by blue circles.

While the BC tracker achieves high scores in scenarios with a non-moving ego-vehicle, it fails when the ego-vehicle moves, as shown by the *MOTA* score of −22.6 % in scenario four and a score of 30.0 % in scenario five. In those cases, the movement of the ego-vehicle increases the number of static

TABLE III: Tracking scores for the presented network and the classification assisted tracker [12] analyzed for the different scenarios. The best scores for each scenario are highlighted in bold font. In addition absolute values for false positives (FP) and false negatives (FN) are given.

	TrackNet				CA Tracker				BC Tracker				
	MOTA	*MOTP*	FP	FN	*MOTA*	*MOTP*	FP	FN	*MOTA*	*MOTP*	FP	FN	Number of Frames
Scenario 1	64.0 %	**3.75**	379	3594	35.4 %	7.52	5653	2165	**65.0 %**	8.67	1272	2995	6531
Scenario 2	57.4 %	**3.38**	31	1873	**70.1 %**	9.66	2	1384	55.4 %	11.01	0	2080	1366
Scenario 3	**82.5 %**	4.90	92	259	66.3 %	2.14	176	473	79.3 %	**0.66**	40	373	2011
Scenario 4	55.7 %	**9.47**	152	806	**60.1 %**	16.63	432	486	−22.6 %	18.71	2549	333	1150
Scenario 5	**45.4 %**	**7.70**	1170	4023	43.8 %	20.55	181	6094	30.0 %	30.14	6703	682	2436

radar targets with a false non-zero radial velocity after the ego-motion compensation, which leads to more false positives when the BC Tracker is used. That is due to the fact that it clusters targets with non-zero radial velocities and opens tracks for the resulting clusters. This limitation makes the BC Tracker unreliable in most scenarios when the ego vehicle is moving. Here the CA Tracker and the Radar TrackNet show a much higher radial velocity clutter tolerance, as both approaches are at least partially machine learning based and trained with samples that include radial velocity clutter. Still, less clutter improves the results for all approaches. This is observable in the third scenario involving a bicycle and a car as visualized in 5, for which high scores are achieved. An overview of the number and types of object tracks in the different scenarios is given in table IV.

Regarding the number of false positives (FP) and false negatives (FN) in the five test scenarios, the TrackNet exhibits a higher probability to miss a real track than to output false tracks. This probability can be tuned by changing the threshold used in the confidence branch to decide if an object is detected or not. For the results given in this work a threshold of 0.6 was chosen. Depending on the application a lower threshold resulting into a higher number of false positives might be desirable.

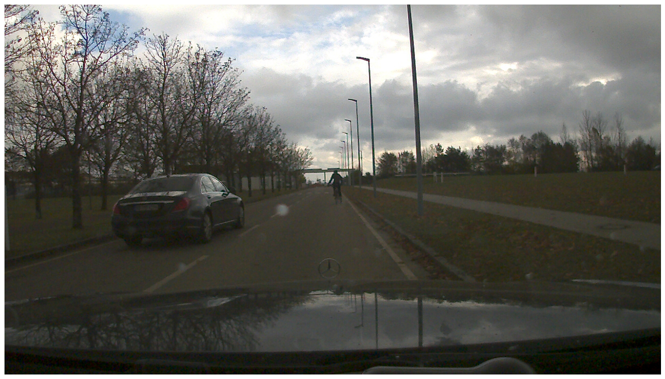
TABLE IV: Number and type of object tracks in the test scenarios.

	#Tracks	#Pedestrian	#Pedestrian Group	#Bike	#Car	#Truck
S. 1	39	0	37	0	1	1
S. 2	16	0	15	0	1	0
S. 3	4	0	0	3	1	0
S. 4	20	3	0	0	15	2
S. 5	136	30	0	14	92	0

Overall, the TrackNet is able to detect road users and output their track information with low velocity and positioning errors, with a *MOTP* value of 5.09 compared to 10.06 for the CA Tracker and 12.45 for the BC Tracker.

B. *Influence of the Doppler Velocity Information*

In order to measure the impact of the radial velocity information provided for every radar target, the Radar TrackNet model was trained again with all velocity information set to zero. The evaluation on the test scenarios shows a degradation of performance when no velocity information is provided, with the score for *MOTA* falling to 17.1 % and a *MOTP* value of 17.78. This indicates that the Doppler velocity is essential information for the network and the time information provided by five consecutive radar measurement frames with only position and RCS information per target can not compensate for the left out information.



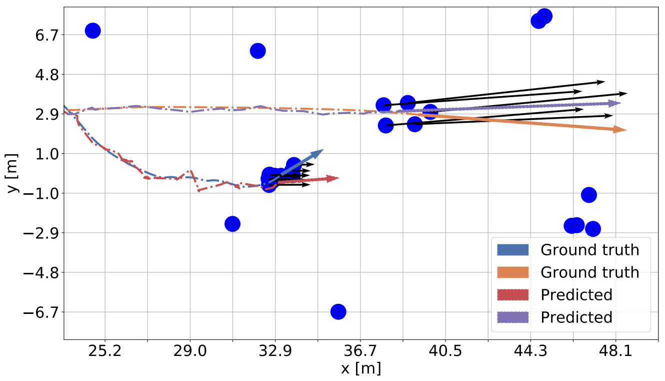


Fig. 5: Scenario 3: A bicycle and a passenger car driving in front of the ego vehicle. The ground truth tracks are visualized in blue for the bicycle and orange for the car, while the predicted tracks are red and purple respectively. Black arrows indicate the measured Doppler velocities of the radar targets which are depicted by blue circles.

VI. Conclusion

In this work, a deep neural network architecture (Radar TrackNet) was introduced, which takes radar point clouds from multiple time steps as input to detect road users and to calculate their tracking information. The network was trained and tested on an extensive real-world radar data set. A comparison to a classification assisted tracker [12], and a basic clustering tracker is done by calculating *MOTA* and *MOTP* scores for five tracking scenarios in which the Radar TrackNet can achieve the best total score of the three approaches. Future work involves having the network predict tracks multiple time steps into the future, increasing the number of samples of underrepresented classes in the data set like bicycles and large vehicles and extend the network to be able to predict multiple tracks per cell.

References

[1] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4696–4704.

[2] O. Schumann, J. Lombacher, M. Hahn, C. Whler, and J. Dickmann, "Scene understanding with automotive radar," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 188–203, 2020.

[3] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," 2016.

[4] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2019.00103

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, p. 11371149, Jun 2017. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2016.2577031

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[7] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.106

[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 51055114.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016. [Online]. Available: http://arxiv.org/abs/1612.00593

[10] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 3569–3577.

[11] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 4490–4499.

[12] S. Haag, B. Duraisamy, W. Koch, and J. Dickmann, "Classification assisted tracking for autonomous driving domain," in *2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, Oct 2018, pp. 1–8.

[13] S. Haag, B. Duraisamy, F. Govaers, W. Koch, M. Fritzsche, and J. Dickmann, "Extended object tracking assisted adaptive clustering for radar in autonomous driving applications," in *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, Oct 2019, pp. 1–7.

[14] ——, "Baas: Bayesian tracking and fusion assisted object annotation of radar sensor data for artificial intelligence application," in *Submitted and accepted by 2020 IEEE Radar Conference (RadarConf)*, 2020.

[15] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollàr, "Learning to Refine Object Segments," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 5957–5966, 2016. [Online]. Available: http://arxiv.org/abs/1603.08695

[16] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, 2017, pp. 936–944.

[17] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[18] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[19] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," *International Conference on Learning Representations 2015*, pp. 1–15, 2015.

[20] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *Eurasip Journal on Image and Video Processing*, vol. 2008, 2008.