Semantic Segmentation on Radar Point Clouds

Ole Schumann,
Markus Hahn
and Jürgen Dickmann
Daimler AG
{ole.schumann, markus.hahn, juergen.dickmann}@daimler.com
Wilhelm-Runge-Str. 11
89081 Ulm, Germany

Christian Wöhler
Technische Universität Dortmund
Image Analysis Group
Otto-Hahn-Str. 4
44227 Dortmund, Germany

***Abstract*—Semantic segmentation on radar point clouds is a new challenging task in radar data processing. We demonstrate how this task can be performed and provide results on a large data set of manually labeled radar reflections. In contrast to previous approaches where generated feature vectors from clustered reflections were used as an input for a classifier, now the whole radar point cloud is used as an input and class probabilities are obtained for every single reflection. We thereby eliminate the need for clustering algorithms and manually selected features.**

I. Introduction

In the last years, image analysis moved from mere classification of a central object in an image and detection of objects or object parts to a single combined task: semantic segmentation. Semantic segmentation describes the task of assigning a class label or a vector of class probabilities to each pixel in an image. Semantic instance segmentation enhances semantic segmentation by differentiating between pixels with the same class label which belong to physically different objects so that in addition to pixel-wise classification also grouping to object instances takes place.

Semantic segmentation is usually accomplished by deep convolutional neural networks [1] which often show an encoder-decoder structure [2], [3]. These architectures all rely on a regular image structure, that is, a rectangular grid with equally spaced pixels. The dimensions of the grid, i.e., the width and the height of the image, may be variable if fully convolutional networks are used. The rectangular grid induces distance and neighborhood relations between the pixels and these relations are exploited by convolution kernels with spatial extensions greater than one pixel. Therefore, these methods work properly if cameras are used as a sensor. For autonomous cars, radar and lidar sensors supplement cameras to maintain functional safety. These additional sensors should not only work complementary but also redundantly. It is hence desirable to gain high semantic understanding of the surroundings also from radar and lidar.

In this article, we will perform semantic segmentation on radar data, i.e., we assign a class label to every measured reflection. We focus on dynamic objects and work with six different classes: `car`, `truck`, `pedestrian`, `pedestrian group`, `bike` and `static` objects. Radar detections obtained after the application of the Constant False Alarm Rate (CFAR) algorithm constitute a point cloud, where a point cloud $\mathcal{P}$ is defined as a set of $N \in \mathbb{N}$ individual points $p_i \in \mathbb{R}^d$, $i = 1, \ldots, N$ in which the order of the points in the point cloud is of no relevance. For each reflection, two spatial coordinates (radial distance r and azimuth angle ϕ), the ego-motion compensated Doppler velocity $\hat{v}_r$ and the radar cross section (RCS) σ are measured. Hence, a $d = 4$ dimensional point cloud has to be processed in the semantic segmentation task. The spatial density of radar reflections can vary drastically so that large scale grid mapping approaches are computationally not feasible. Therefore, the usual network structures used for camera images cannot be applied. The necessity of an algorithm that does not require an image-like input can be read off from Fig. 1, where radar detections collected from four radars during a period of $200\,\mathrm{ms}$ are displayed. In this figure, large areas with no measurements as well as areas with a large number of reflections are visible. A grid map of the whole scene with approximately 2000 individual reflections would have to cover a large spatial area of at least $150\,\mathrm{m} \times 200\,\mathrm{m}$ and even at a very low resolution with cell sizes of $1\,\mathrm{m} \times 1\,\mathrm{m}$, at most $6\,\%$ of the pixels in the grid would have a non zero value.

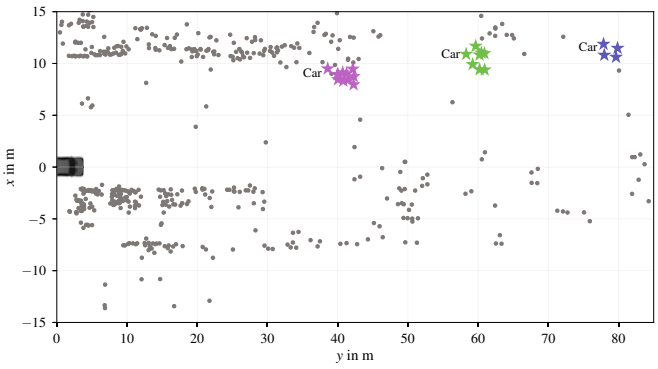

Fig. 1. Radar point cloud accumulated over $200\,\mathrm{ms}$. Reflections of three different cars are highlighted. Only an excerpt of the complete field of view is shown.

We therefore use PointNet++ [4] as a basis for our segmentation algorithm. PointNet++ is capable to work directly on point clouds and it was originally designed to work on 3D

978-0-9964527-7-9 ©2018 ISIF

spatial data from laser scanners. In this article, we modified the architecture to handle two spatial dimensions and two further feature dimensions.

In a previous work [5], classification was done on feature vectors, which in turn were obtained from clustered radar reflections. With our new approach, we avoid these two pre-processing steps: grouping of radar targets to clusters and generation of pre-defined feature vectors from these clusters is no longer necessary. We show that our new approach outperforms the previous method by a great margin.

The rest of the article is structured as follows: In section II, we comment on related work and other approaches to the topic. Afterwards, our network architecture is described in more detail and we explain our training and testing procedure. In section IV, we display our results and compare them to previous approaches. Finally, we give an outlook to our future work.

II. Related Work

Semantic segmentation is a popular method, when cameras are used as a sensor and most algorithms are tailored to image data. The introduction of fully convolutional networks [1] inspired many similar and later on more advanced neural network structures like SegNet, [2] U-Net [6], R-CNN [7], and its successors Fast R-CNN [8], Faster R-CNN [9], and Mask R-CNN [3]. In order to apply these techniques to radar data, some pre-processing is necessary. Grid maps provide a way to transform the spatially inhomogeneous radar reflections into image like data. The measured reflections are integrated over time and inserted at the respective positions in a map. Different maps can be created with this approach, e.g. occupancy grid maps, which describe the posterior probabilities for cell occupancy, or RCS maps, which provide information about the measured RCS values of the reflections in each cell [10]–[13].

This approach works well for static objects, because only the ego-motion (and not additionally the object's velocity and trajectory) has to be taken into account to insert radar reflections of different times at the correct position in the map. For dynamic objects, which are considered in this work, either precise extended target tracking algorithms are needed or the dynamics of the objects are considered as a feature so that moving objects create an extended tail of reflections in the map. Another difficulty is that for sparse data, grid mapping is not efficient since a potentially large grid is needed to display relatively few measurements.

To the best of our knowledge, semantic segmentation was not done before on automotive radar data of moving objects. Classification was only done on small data sets or large amount of simulated data [14]–[19].

III. Methods

A. Network structure

Qi et al. provide with PointNet [20] and PointNet++ [4] methods to work directly with point clouds so that no previous mapping step is needed. They perform semantic segmentation on 3D point clouds obtained by sampling points from meshes of 3D scans of indoor scenes. We use their architecture as a basis for our approach. However, the radar data we use in our experiments differs from 3D indoor data in the following aspects. Firstly, each radar reflection contains only two instead of three spatial coordinates, but with the two additional values from the ego-motion compensated Doppler velocity and the RCS value, each point p_i of the whole point cloud $\mathcal{P}$ is four-dimensional. Secondly, our data shows much greater differences in density and sampling rate. The 3D scans from the Stanford 3D semantic parsing data set [21] provide high density point clouds in which fine details of office interior is visible, whereas our radar data provides only a few reflections per object so that for smaller or more distant objects not even the object's outline is captured properly, see Fig. 1.

Among others, in PointNet++ a multi-scale grouping module (MSG) and a feature propagation module (FP) are defined. The MSG module considers neighborhoods of multiple sizes around a central point and creates a combined feature vector at the position of the central point that describes these neighborhoods. The module contains three steps: selection, grouping and feature generation. First, N_{sample} points of the input point cloud are selected by farthest point sampling, so that the input point cloud is sampled homogeneously. In the grouping step, for each of the N_{sample} selected points, neighborhoods are created. In our network, neighborhoods are composed of N_{neigh} points that lie within a radius r around the central point. Only the two spatial components of the radar reflections are considered for the neighborhood search. If a reflection has more than N_{neigh} neighbors in the given search radius, only the first N_{neigh} points that were found are used for further computations. If less reflections are found, the first neighbor is repeated to guarantee a fixed size data structure. In each MSG module, multiple neighborhoods with different values for r and N_{neigh} are created. In the final step, features are generated for each of the N_{sample} points by applying convolution layers with filter size 1×1 on the neighborhood tensor with shape $(N_{\text{sample}}, N_{\text{neigh}}, c_{\text{in}})$, where c_{in} is the number of channels. This results in a tensor of size $(N_{\text{sample}}, N_{\text{neigh}}, c_{\text{out}})$ on which a final max pooling layer is applied so that only the contribution of the neighbor with the highest activation for the respective filter is taken into account.

The number of points in the output point cloud after a MSG module is smaller than in the input point cloud so that points in deeper layers contain more and more abstract features which provide information about the surrounding points of previous layers. This process is analogous to convolutional networks for image processing where the image dimensions are reduced in each layer. In Fig. 2, the spatial positions as well as the ego-motion compensated Doppler velocities of radar reflections are shown and the sub-sampling of the input point cloud after each MSG module is depicted. The high dimensional feature vectors that are generated for each point in a MSG module are not depicted in the figures. A camera image of the scene is shown in Fig. 3.

For semantic segmentation, the information of the sub-sampled point cloud is propagated to the full input point cloud.

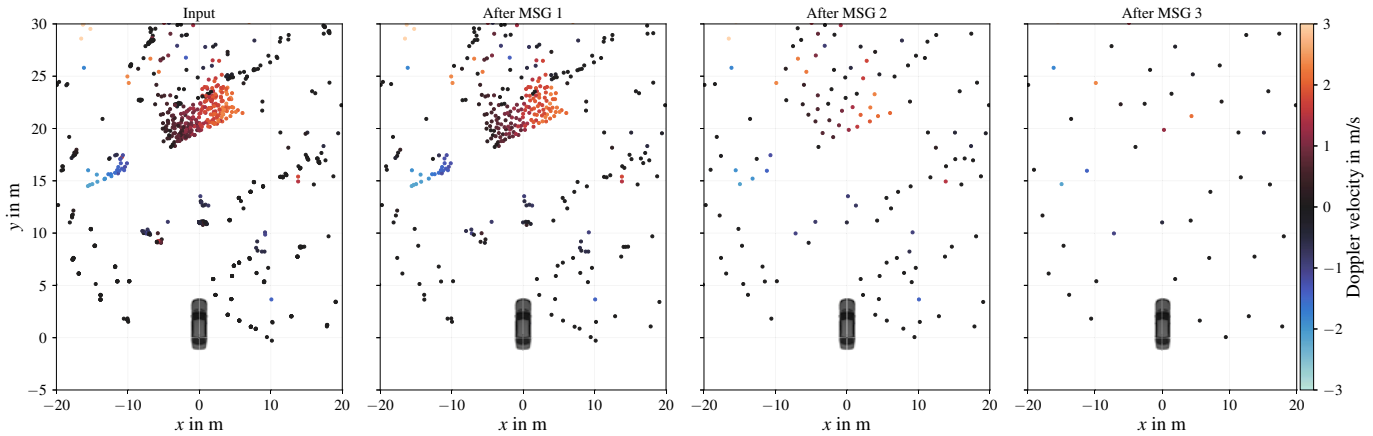

Fig. 2. Excerpt of one example radar point cloud. The spatial coordinates as well as the ego-motion compensated Doppler velocity are plotted. From left to right: point cloud at the input layer and sub-sampled point clouds after the first, second and third MSG module. The data were accumulated over 500 ms. The camera image of this scene can be found in Fig. 3.

Fig. 3. Camera image of the same scene as in Fig. 2.

This task is carried out by the feature propagation modules: k layers of MSG modules are followed by k layers of FP modules which repeatedly propagate the features of the less populated point cloud to the next higher layer. For each point p_i in the denser point cloud, a weighted average of the feature vectors of the three nearest neighbors in the sparser point cloud is computed and – after passing this feature vector through a set of convolution layers – then assigned to the point p_i. Skip connections from the respective level of the MSG module improve the propagation of the features.

Our network structure is depicted in Fig. 4 where also the values of the parameters of the MSG modules are defined.

B. Data set

In this article, we only use real world data that were collected by two different experimental vehicles, vehicle $\mathcal{A}$ and vehicle $\mathcal{B}$. Vehicle $\mathcal{A}$ was equipped with four 77 GHz sensors which were mounted at the two front corners and at the sides of the vehicle. Only the near range mode of the sensor was used so that targets up to a range of 100 m were detected. Each sensor had a field of view of $\pm 45°$. Vehicle $\mathcal{B}$ was equipped with eight radar sensors with the same specifications as the sensors of vehicle $\mathcal{A}$. These eight sensors were mounted on the four corners of the car and at the front left, front right, back left and back right sides of the car.

The data set of vehicle $\mathcal{A}$ ($\mathcal{B}$) contains measurements from over 4.5 h (6.5 min) of driving, that is, over 100 millions (5 millions) of radar reflections were collected of which 3 millions (100 000) belong to 6200 (191) different moving objects. All reflections that belong to the same physical object were manually grouped together and annotated with a label from the following classes: `car`, `truck`, `pedestrian`, `pedestrian group`, `bike` and `static`. The distribution of the reflections among the six classes is shown in Tab. I. In contrast to our previous work [5], clutter was not examined in an extra class but treated as `static`, since in this work we aim at detecting and classifying only real dynamic objects from a raw point cloud. Our previous classifier had to deal with clusters and feature vectors that did not originate from real objects so that a differentiation between a `garbage` class and real objects was necessary. These falsely created clusters and feature vectors were artifacts from imperfect pre-processing steps which we try to avoid here.

TABLE I
DISTRIBUTION RADAR REFLECTIONS AMONG THE SIX CLASSES.

Vehicle	car	ped.	ped. group	bike	truck	static
$\mathcal{A}$	1 238 849	315 268	739 690	114 153	603 476	97 714 654
	1.23 %	0.31 %	0.74 %	0.11 %	0.60 %	97.01 %
$\mathcal{B}$	69 170	8981	1402	0	16 978	4 742 445
	1.43 %	0.19 %	0.03 %	0 %	0.35 %	98 %

C. Training and Testing

Before we did the actual training, fixing of hyper-parameters was necessary. The number of MSG modules, the number of sample points N_{sample}, the number of neighborhoods in each MSG module with their respective radii r and the number

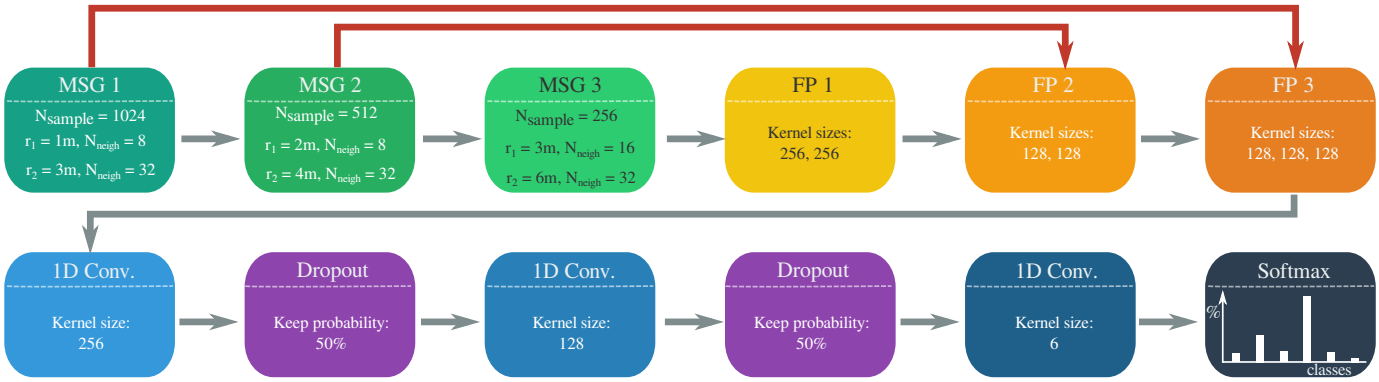

Fig. 4. Structure of our network. The red arrows indicate skip connections through which extracted features from MSG modules are passed to the FP module of the respective layer. The kernel sizes of the three MSG modules are $[[32, 32, 64], [64, 64, 128]]$, $[[32, 32, 64], [64, 64, 128]]$ and $[[64, 64, 128], [64, 64, 128]]$.

of neighboring points N_{neigh} for each sample point as well as the number and size of the convolution layers in each module had to be defined. This was done by examining reasonable configurations on randomly selected validation sets and altering these configurations to optimize the performance of the network even further. A complete sampling of the parameter space is not feasible due to the enormous size of this space and the accompanying computational costs.

The finally chosen and best performing architecture is depicted in Fig. 4.

For evaluation, five-fold cross validation was performed. That is, the data set was split into five folds with 20 % of the data per fold, and each fold was once used for testing while the remaining four folds where used as training data. Only data of vehicle $\mathcal{A}$ was used for training. The measurements from vehicle $\mathcal{B}$ were only used for inspecting the generalization ability of our classifier. The network was training using stochastic gradient descent with a cross-entropy based loss function and the Adam optimization scheme [22]. Parts of the `tensorflow` source code published in [23] were used.

Due to the large imbalance between static and dynamic data (approximately 97 million to 3 million), the weights of the loss function for the `static` class were reduced so that the optimization was stopped from assigning almost all points to the `static` class.

The training was done for 30 epochs during which data augmentation took place: Random noise was applied to each feature dimension, so that the spatial positions of the reflections as well as the measured RCS values and the ego-motion compensated Doppler velocities were altered. The velocity feature was only modified for reflections of dynamic objects. In addition, for each dynamic object a random number $q \in [0, 0.3]$ was generated and each reflection of this object was left out in this epoch with probability q, so that the shapes and densities of the dynamic objects were changed.

The network itself had no notion of the recording time of the individual reflections but during training we provided time windows of length $T = 500\,\text{ms}$ to the network so that the point cloud became more dense and more reflections per object could be considered. The reflections of the different time steps were transformed into the vehicle coordinate system at the time of the earliest measurement. The input size of the point clouds was fixed to 3072 reflections. If more than 3072 reflections were measured during the $500\,\text{ms}$ long time window, reflections from the `static` class were removed and if less than 3072 reflections were measured, one reflection was re-sampled the required amount of times. Due to the max pooling layers in the network structure, this oversampling does not change the outcome of the semantic segmentation.

During testing, the next 3072 reflections were passed through the network, sorted by the measurement time, so that no over- or undersampling was necessary.

Training was done on a Linux workstation equipped with a Nvidia GeForce GTX 1070 GPU.

IV. Results

Evaluation of our system has been performed based on 6×6 confusion matrices and the macro-averaged F_1 score (from now on only called F_1 score). The F_1 score corresponds to the harmonic mean of precision and recall [24]. In macro averaging, each class contributes equally to the total score – irrespective of the class counts – since an individual F_1 score is calculated for each class and these six values are then averaged.

A. Best Performing Architecture

We first show our results obtained with our best performing architecture. We used only data from vehicle $\mathcal{A}$ for the five-fold cross-validation. In addition to the two spatial coordinates x and y (in vehicle coordinates), we enriched the input point cloud with the ego-motion compensated Doppler velocities and the RCS values. Hence, a four-dimensional point cloud was provided as an input.

The resulting confusion matrix is shown in Fig. 5.

Not surprisingly, the majority class with label `static` shows the highest true positive value. However, one should keep in mind that the distinction between reflections that belong to either moving or non-moving objects is far more difficult than setting a threshold on the Doppler velocity and classifying each reflection with velocity below this threshold

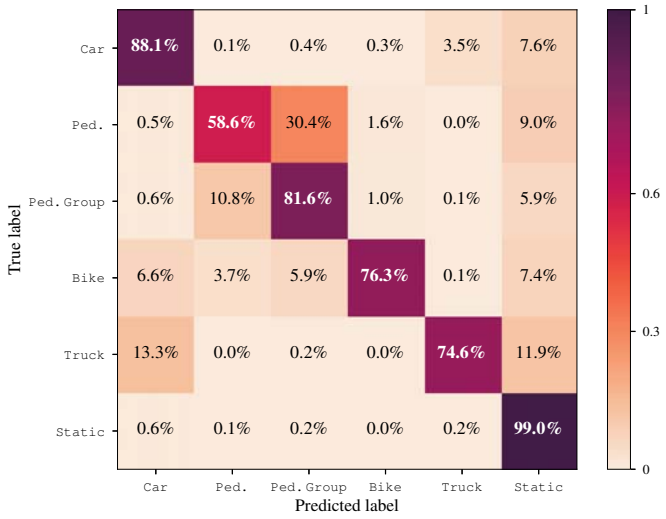

Fig. 5. Relative confusion matrix after 5-fold cross-validation with the network structure as depicted in Fig. 4. Input features of the point cloud: $x, y, \hat{v}_r, \sigma$.

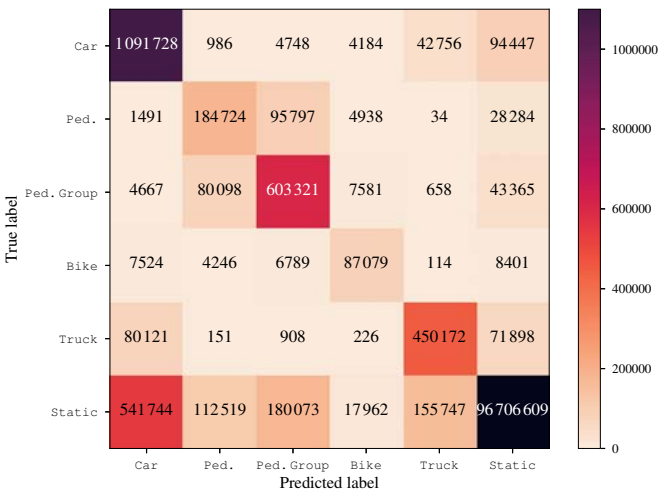

Fig. 6. Absolute confusion matrix after 5-fold cross-validation with the network structure as depicted in Fig. 4. Input features of the point cloud: $x, y, \hat{v}_r, \sigma$.

as static. In real world scenarios, many reflections that do not belong to a moving object show a non-zero ego-motion compensated Doppler velocity, caused by errors in the odometry, sensor misalignment, time synchronization errors, mirror effects or other sensor artifacts. In addition, reflections with zero Doppler velocity do not necessarily belong to a static object, since also the bottom of a rotating car wheel or body parts of a pedestrian that move diametrical to the walking direction may show no radial velocity.

Objects of the class `car` are classified second best, followed by the `pedestrian group`. Members of the `truck` class are often confused with `cars`. Two reasons can explain this confusion: Firstly, at high distances only few reflections can be measured per object so that the spatial extent of an object can hardly be deduced. Secondly, the transition between `car` and `truck` instances is rather smooth, since, e.g., large SUVs are hardly distinguishable from smaller trucks.

Another prominent behavior that can be inferred from the figure is the high confusion between `pedestrians` and `pedestrian groups`. This behavior may be induced by our training data since for human annotators it is sometimes possible to assign the reflections of two nearby pedestrians to the individual persons and hence to create two instances of the class `pedestrian`, but sometimes this is not easily possible and too time demanding so that all reflections are labeled as a single instance of the class `pedestrian group`. So in addition to a complicated task itself, the network also has to struggle with inconsistencies in the ground truth data. For many driving tasks it is not critical to know if there are one or two pedestrians at a certain region, so that the two classes may be merged together yielding over $91\,\%$ true positives.

Due to the highly imbalanced data set, inspecting only the relative confusion matrix normalized to the class counts may be misleading. We therefore also present the confusion matrix with absolute values in Fig. 6. This visualization highlights that many false positive dynamic objects are created by the network (last row in the figure). This effect becomes most apparent for the class `car`: only $68\,\%$ of the predicted `car` reflections belong to a dynamic object (cf. first column in Fig. 6). However, for automotive applications it could be more desirable to have a high false positive rate for dynamic objects than a high false negative rate. Reducing the weights in the loss function for reflections of the `static` class causes higher false positive values, so that this parameter allows us to tune between false positives and false negatives.

It should be noted that the percentages given for the confusion between dynamic and `static` reflections (last column in the confusion matrix in Fig. 5) do not represent the percentages of overlooked objects. If only one reflection of a dynamic object is classified correctly but the rest of the reflections of the same object are classified as `static`, the object is still detected even though the false negative count increases.

B. Variation of the Input Features

To gain more insight into what information is useful to the network, we repeat the five-fold cross-validation with three different sets of input features $f_1 = \left(x, y, \hat{v}_r\right)$, $f_2 = \left(x, y, \sigma\right)$, $f_3 = \left(x, y\right)$ and compare the results to the original features $f_0 = \left(x, y, \hat{v}_r, \sigma\right)$. In Table II, the F_1 scores for each input configuration are displayed. The following observations can be made from this table. The more input features are presented to the network, the higher the performance. Adding RCS values of each reflection to the input features causes a small increase in the F_1 score (from 0.7303 to 0.7425), whereas including the ego-motion compensated Doppler velocities has a far greater effect, levering the score by almost 0.1. Despite the expected importance of Doppler velocities as a feature, it is interesting to see that for input features f_2 and f_3 the performance of the network is still far above random guess. This implies that the

spatial surroundings of a reflection are very expressive features to the network and build the foundation of the classification step which is then leveraged by the additional features of the velocity and RCS values.

TABLE II
CLASSIFICATION SCORES FOR DIFFERENT INPUT FEATURES.

Input Features	F_1 macro-averaged
$f_0 = (x, y, \hat{v}_r, \sigma)$	0.7425
$f_1 = (x, y, \hat{v}_r)$	0.7303
$f_2 = (x, y, \sigma)$	0.6492
$f_3 = (x, y)$	0.5939

C. Test on Data from Vehicle $\mathcal{B}$

Up to now, only data form vehicle $\mathcal{A}$ was used for training and testing. We now use a network that was trained using only data from vehicle $\mathcal{A}$ to predict the classes of the reflections measured by vehicle $\mathcal{B}$. The differences in this setup are twofold: On the one hand, vehicle $\mathcal{B}$ is equipped with eight instead of four radar sensors so that $360°$ vision around the vehicle is provided instead of the mainly front and side facing setup of vehicle $\mathcal{A}$. On the other hand, the data from vehicle $\mathcal{A}$ was collected in German cities and country roads whereas vehicle $\mathcal{B}$ only collected data in the USA. Different road and street designs as well as on average larger cars constitute challenges for the algorithm.

Applying our best performing network on these new data results in an F_1 score of 0.46 which is distinctly below the value obtained by our five-fold cross validation. If the four sensors at the front of the test vehicle are evaluated independently from the four back-facing sensors, the F_1 score increases to 0.48.

Since the data set from vehicle $\mathcal{B}$ is very small compared to the data set of vehicle $\mathcal{A}$, one has to be cautious with the interpretation of the results. However, it becomes obvious that changing the sensor setup does have an influence on the performance of the classifier.

D. Comparison with Previous Approach

In a previous work [5], we used a combination of DBSCAN [25] for clustering and an LSTM network [26] for classification to generate class labels for sequences of feature vectors. Previously, we measured our performance on *feature vectors* generated on ground truth clusters. In this article, evaluation of this approach is done per-reflection by projecting the class labels of the feature vectors back to the original reflections of the clusters.

We train the LSTM network and our new approach on the same data set and evaluate both methods on an identical test set. For a fair comparison, the LSTM is not tested on the feature vectors of the ground truth clusters but rather on the feature vectors generated from the clusters obtained by applying DBSCAN on the point cloud. In contrast to our current approach, the LSTM also learns to classify feature vectors as `garbage`, if they originate from clusters which do not belong to real objects. If the LSTM rejects such a feature vector, we treat the associated points as `static` in the comparison.

Our new method reaches an F_1 score of 0.734 on this selected test set, whereas the DBSCAN+LSTM approach results in a score of only 0.597. The new approach creates far fewer false positive dynamic objects and has higher true positive counts in all classes. The most appealing feature is that three times less reflections are erroneously considered as static so that possibly less objects are overlooked. The confusion of reflections originating from dynamic objects with ones from the `static` class does not only stem from bad classification results of the LSTM, but mostly because of insufficient clustering so that the LSTM has never the chance to classify certain reflections.

E. Visualizations

It is informative to visualize the outputs of different network layers during a forward pass of one scene. Fig. 2 displays the spatial positions as well as the Doppler velocities of one example scene at the input level and after the three MSG modules.

It is difficult to visualize the convolution kernels of the different layers, since only 1×1 convolutions are performed and hence meaningful images of the filters themselves do not exist. However, we passed different scenes through the network and collected the network outputs before the last convolution layer. From this output, we randomly selected 1000 points from each class along with their 128-dimensional feature vectors and passed this high dimensional point cloud through the t-SNE dimensionality reduction algorithm [27] to obtain a two-dimensional point cloud. This is visualized in Fig. 7, where four distinct clusters for the classes `car`, `truck`, `bike` and `static` can be observed. The reflections stemming from `pedestrians` or `pedestrian groups` are not well separated, in accordance with the confusion matrix in Fig. 5. Reflections from the `car` and `bike` class enrich the center of the point cloud, displaying points which are hard to classify.

Finally, Fig. 8 displays the same scene as in Fig. 2, but now the predicted class labels are shown instead of the Doppler velocity. All three pedestrians, the truck and the car were identified correctly. However, some clutter behind the rightmost pedestrian was falsely classified as a pedestrian group and a few reflections behind the car were also erroneously labeled as members of the `car` class. Despite that, the semantic information of the scene is well represented.

V. CONCLUSION AND OUTLOOK

In this paper, we provided results for semantic segmentation on radar data using a variant of PointNet++ as our classification algorithm. We showed that our new approach outperforms our previous method which included two now obsolete preprocessing steps, namely clustering and feature generation. In addition, we demonstrated that utilizing both, the RCS value and the ego-motion compensated Doppler velocity, improves

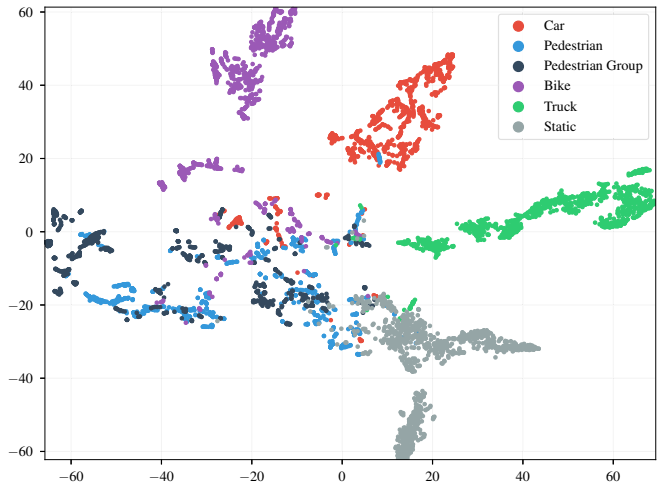

Fig. 7. Two dimensional embedding of the 128 dimensional feature vectors of the second to last convolutional layer in our network. The non-linear t-SNE method was used for the embedding.

the classification results, whereby the Doppler velocity has a greater impact on the results.

In future works, we will focus on two different aspects. On the one hand, it seems beneficial to incorporate time information into the network. The temporal evolution of objects is a descriptive feature that should at least improve the distinction between static and dynamic class instances. One possible way to achieve this goal is to integrate a recurrent neural network structure into PointNet++. A simpler way would be to present the measurement time stamp as an additional feature. On the other hand, an extension to semantic *instance* segmentation is desirable. Currently, only class labels are provided for each reflection without notion of the object instance this reflection belongs to. We therefore do not know how many different objects exist in a scene but only have knowledge about the amount of reflections that belong to an object class. Class-aware clustering algorithms are one possibility to generate instances from the reflections but combined learning of instances and class affiliation may yield a higher total performance.

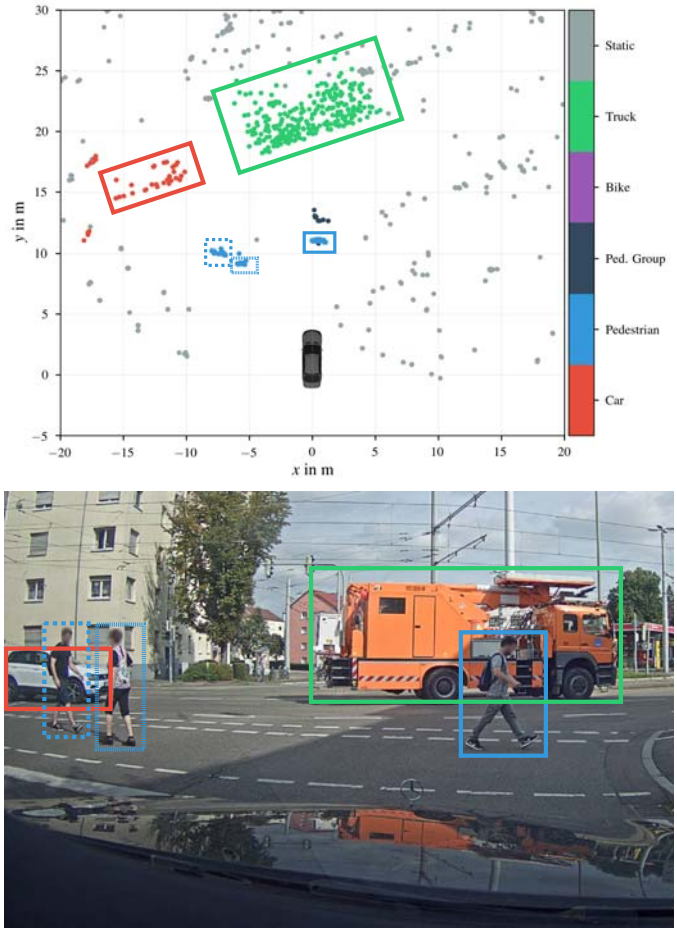

Fig. 8. Predicted class labels for each reflection for one example scene. The bounding boxes were added manually for association between the point cloud and the camera image.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1411.4038

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *arXiv preprint*, nov 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[3] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://arxiv.org/pdf/1703.06870.pdf

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, dec 2017. [Online]. Available: https://arxiv.org/pdf/1612.00593.pdfhttp://arxiv.org/abs/1612.00593

[5] O. Schumann, M. Hahn, J. Dickmann, and C. Wöhler, "Comparison of random forest and long short-term memory network performances in classification tasks using radar," in *Sensor Data Fusion: Trends, Solutions, Applications (SDF)*. IEEE, oct 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8126350/

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv preprint*, may 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. [Online]. Available: http://arxiv.org/abs/1311.2524

[8] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Neural Information Processing Systems (NIPS)*, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[10] J. Lombacher, M. Hahn, J. Dickmann, and C. Wohler, "Detection of arbitrarily rotated parked cars based on radar sensors," in *16th International Radar Symposium (IRS)*. IEEE, jun 2015, pp. 180–185. [Online]. Available: http://ieeexplore.ieee.org/document/7226281/

[11] ——, "Potential of radar for static object classification using deep learning methods," in *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. IEEE, may 2016, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/7533931/

[12] ——, "Object classification in radar using ensemble methods," in *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. IEEE, mar 2017, pp. 87–90. [Online]. Available: http://ieeexplore.ieee.org/document/7918863/

[13] J. Lombacher, K. Laudt, M. Hahn, J. Dickmann, and C. Wohler, "Semantic radar grids," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jun 2017, pp. 1170–1175. [Online]. Available: http://ieeexplore.ieee.org/document/7995871/

[14] M. Özcan, S. Z. Gürbüz, A. R. Persico, C. Clemente, and J. Soraghan, "Performance Analysis of Co-Located and Distributed MIMO Radar for Micro-Doppler Classification," in *EuRAD 2016*, 2016, pp. 85–88.

[15] E. Schubert and M. Kunert, "Human RCS measurements and dummy requirements for the assessment of radar based active pedestrian safety systems," *14th International Radar Symposium (IRS)*, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6581669

[16] S. Heuel and H. Rohling, "Pedestrian recognition based on 24 GHz radar sensors," in *11th International Radar Symposium (IRS)*. Vilnius: IEEE, 2010, pp. 1–6.

[17] ——, "Two-stage pedestrian classification in automotive radar systems," *12th International Radar Symposium (IRS)*, pp. 477–484, 2011.

[18] ——, "Pedestrian classification in automotive radar systems," in *13th International Radar Symposium (IRS)*. Warsaw: IEEE, 2012, pp. 39–44.

[19] P. Molchanov and A. Vinel, "Radar frequency band invariant pedestrian classification," *14th International Radar Symposium (IRS)*, pp. 1–6, 2013.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *arXiv preprint*, jun 2017. [Online]. Available: http://arxiv.org/abs/1706.02413

[21] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D Semantic Parsing of Large-Scale Indoor Spaces," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[23] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++, Tensorflow Implementation," 2017. [Online]. Available: https://github.com/charlesq34/pointnet2

[24] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2007.

[25] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," in *Advances in Knowledge Discovery and Data Mining*. Springer, Berlin, Heidelberg, 2013, pp. 160–172. [Online]. Available: http://link.springer.com/10.1007/978-3-642-37456-2{_}14

[26] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: http://www7.informatik.tu-muenchen.de/{~}hochreit

[27] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.