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 Fritzscbe*

(*) 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 [11]).

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.

<table>
<thead>
<tr>
<th></th>
<th>Passenger Car</th>
<th>Pedestrian</th>
<th>Pedestrian Group</th>
<th>Two Wheeler</th>
<th>Large Vehicle</th>
<th>Clutter</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td># labeled targets</td>
<td>1922385</td>
<td>424047</td>
<td>934994</td>
<td>237258</td>
<td>913047</td>
<td>1379757</td>
<td>128828260</td>
</tr>
<tr>
<td>in %</td>
<td>1.43</td>
<td>0.31</td>
<td>0.69</td>
<td>0.18</td>
<td>0.08</td>
<td>1.02</td>
<td>95.68</td>
</tr>
<tr>
<td># labeled objects</td>
<td>3815</td>
<td>1701</td>
<td>1216</td>
<td>379</td>
<td>503</td>
<td>49236</td>
<td>–</td>
</tr>
<tr>
<td>in %</td>
<td>6.71</td>
<td>2.99</td>
<td>2.14</td>
<td>0.67</td>
<td>0.88</td>
<td>86.61</td>
<td>–</td>
</tr>
<tr>
<td>observed times (h)</td>
<td>7.28</td>
<td>3.68</td>
<td>4.42</td>
<td>1.10</td>
<td>1.39</td>
<td>6.71</td>
<td>–</td>
</tr>
<tr>
<td>in %</td>
<td>29.60</td>
<td>14.98</td>
<td>17.99</td>
<td>4.48</td>
<td>5.63</td>
<td>27.31</td>
<td>–</td>
</tr>
</tbody>
</table>

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 GHz,
- Maximum range of 100 m,
- FOV of ±60°,
- Range accuracy of $\Delta r = 0.15$ m,
- Doppler velocity accuracy of $\Delta v_r = 0.1$ km h$^{-1}$,
- Angular accuracy ranging from $\Delta \phi = 0.5^\circ$ to $\Delta \phi = 2^\circ$.

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 ms between two scans. A full scan of all four sensors is assumed to be done in 75 ms and further referred to as one frame, resulting in a radar frame rate of 13.33 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 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 $\vec{r}_{cc} = (x_{cc}, y_{cc})$ in the ego-vehicle’s coordinate frame.
- The ego-motion compensated radial velocity components $\vec{v}_{r} = (v_{xr}, v_{yr})$.
- A radar cross section (RCS) value $\sigma$.

In addition to those measured quantities, the position $\vec{r}_{s}$ = $(x_{s}, y_{s})$ and unit-vector orientation $\vec{a}_{s} = (a_{xs}, a_{ys})$ 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

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

with $(x_{cell}, y_{cell})$ representing the position of the left-bottom corner of a cell with the indices $(i, k)$, $(s_{xcell}, s_{ycell})$ 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 $T_{merge} = (T-1)/2$ 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.
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 \( \frac{1}{100} \) 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 Precision (MOTP). Those metrics are calculated by

\[
MOTP = \frac{\sum_{i,t} d_{t,i}}{\sum_{t} e_t}
\]

\[
MOTA = 1 - \frac{\sum_{t} (m_t + fp_t + mme_t)}{\sum_{t} g_t}
\]

with the number of track matches \( e_t \), distance \( d_{t,i} \) 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.

![Network architecture](image)
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 UI/M-M 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).

<table>
<thead>
<tr>
<th></th>
<th>MOTA</th>
<th>MOTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar TrackNet</td>
<td>62.0 %</td>
<td>5.09</td>
</tr>
<tr>
<td>BC Tracker</td>
<td>52.4 %</td>
<td>12.45</td>
</tr>
<tr>
<td>CA Tracker</td>
<td>47.2 %</td>
<td>10.06</td>
</tr>
</tbody>
</table>

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 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.

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

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

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.
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.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>TrackNet MOTA</th>
<th>TrackNet MOTP</th>
<th>CA Tracker MOTA</th>
<th>CA Tracker MOTP</th>
<th>BC Tracker MOTA</th>
<th>BC Tracker MOTP</th>
<th>FP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>64.0 %</td>
<td>3.75</td>
<td>379</td>
<td>3594</td>
<td>35.4 %</td>
<td>7.52</td>
<td>5653</td>
<td>2165</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>57.4 %</td>
<td>3.38</td>
<td>34</td>
<td>1873</td>
<td>70.1 %</td>
<td>6.26</td>
<td>2</td>
<td>1384</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>82.5 %</td>
<td>14.90</td>
<td>92</td>
<td>259</td>
<td>66.3 %</td>
<td>2.14</td>
<td>176</td>
<td>473</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>55.7 %</td>
<td>9.47</td>
<td>152</td>
<td>806</td>
<td>60.1 %</td>
<td>16.63</td>
<td>432</td>
<td>486</td>
</tr>
<tr>
<td>Scenario 5</td>
<td>45.4 %</td>
<td>7.70</td>
<td>1170</td>
<td>4023</td>
<td>43.8 %</td>
<td>20.55</td>
<td>181</td>
<td>6094</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>S.</th>
<th>#Tracks</th>
<th>#Pedestrian</th>
<th>#Pedestrian Group</th>
<th>#Bike</th>
<th>#Car</th>
<th>#Truck</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>39</td>
<td>0</td>
<td>37</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>136</td>
<td>30</td>
<td>0</td>
<td>14</td>
<td>92</td>
<td>0</td>
</tr>
</tbody>
</table>

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.
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