25^{th} Computer Vision Winter Workshop
Domen Tabernik, Alan Lukežič, Klemen Grm (eds.)
Rogaška Slatina, Slovenia, February 3–5, 2020

Towards Data-driven Multi-target Tracking for Autonomous Driving

Christian Fruhwirth-Reisinger, Georg Krispel, Horst Possegger, Horst Bischof
Graz University of Technology
Institute of Computer Graphics and Vision
{christian.reisinger, georg.krispel, possegger, bischof}@icg.tugraz.at

Abstract. *We investigate the potential of recurrent neural networks (RNNs) to improve traditional online multi-target tracking of traffic participants from an ego-vehicle perspective. To this end, we build a modular tracking framework, based on interacting multiple models (IMM) and unscented Kalman filters (UKF). Following the tracking-by-detection paradigm, we leverage geometric target properties provided by publicly available 3D object detectors. We then train and integrate two RNNs: A state prediction network replaces hand-crafted motion models in our filters and a data association network finds detection-to-track assignment probabilities. In our extensive evaluation on the publicly available KITTI dataset we show that our trained models achieve competitive results and are significantly more robust in the case of unreliable object detections.*

1. Introduction

Multi-target tracking (MTT) aims at jointly estimating the number of targets and their current states from a sequence of unreliable measurements. It is one of the fundamental visual perception tasks for autonomous driving (AD) [21] which allows, for example, reactive navigation or motion planing.

In this work, we address the problem of tracking robustness despite unreliable detections, which strongly degrade tracking performance. This requires, on the one hand, precise state predictions for frames without proper detections and, on the other hand, reasonable track-to-detection assignments. Hence, we train and integrate two recurrent neural networks (RNNs) for these purposes, as illustrated in Fig. 1. The advantage as shown in our evaluations is that data-driven models generalize better on numerous different situations.

Most recent online multi-target tracking approaches, *e.g.* [16, 30, 44, 50] follow the tracking-by-detection paradigm and thus, assume a detected set of possible targets in each frame. For AD, these detections are usually obtained from state-of-the-art object detectors [29, 39, 45] which estimate 3D bounding boxes from LiDAR data or RGB images.

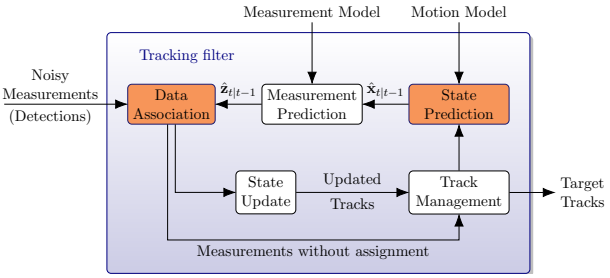

Figure 1. Tracking-by-detection scheme. We exchange the highlighted modules (*i.e.* State Prediction and Data Association) with data-driven recurrent neural networks.

Simultaneously tracking multiple targets is commonly handled by applying a single target tracker (STT) for each object instance, realized by probabilistic filters [1, 6]. These filters predict the current state usually relying on hand-crafted motion models and update the predictions with assigned detections to estimate the *posterior* distribution of each track. A data association step assigns the most reasonable detection to each track and consequently allows the filter updates. Finally, MTT requires track management to initialize and terminate trajectories.

Hence, the main challenges in multi-target tracking are the assignment of detections to tracks and determining whether a track exists or not. The former strongly depends on the detection quality. Many false positive (FP) or false negative (FN, *i.e.* missed) detections require additional knowledge of the tracked targets, *e.g.* their motion behavior, to produce a reasonable trajectory. Additionally, targets at high speed, moving sensors and interacting targets hamper

accurate associations. The question whether a track exists or not, on the other hand, is even harder to answer. Occluded targets and false detections can lead to missing or wrongly initialized tracks, respectively.

Our main contribution is to investigate several tracking aspects for AD within a combined Bayesian filter-based MTT framework. To track traffic participants in 3D world coordinates, we leverage the interacting multiple model (IMM) [8] approach combined with an unscented Kalman filter (UKF) [23, 47], to allow multiple, potentially non-linear motion models. For our investigation, we focus on two tasks: State prediction and data association. We exchange these traditionally hand-crafted parts with learned RNN models and evaluate their impact on the KITTI dataset [17]. Our experiments demonstrate that our data-driven models significantly improve the tracking performance, especially in the case of unreliable or missing detections. Furthermore, in contrast to most recent works which focus exclusively on well-represented object classes (*i.e*. cars and pedestrians), we consider all available object classes. This allows more meaningful conclusions about the tracking capabilities for real-world AD scenarios.

2. Related Work

Because of the large diversity of tracking methods, we focus mainly on online filtering-based and deep learning approaches which have been proposed for traffic scenarios. For a more extensive survey we refer the interested reader to the recent works of Krebs *et al.* [27] and Vo *et al.* [46].

Bayesian Filtering: Most MTT algorithms following the tracking-by-detection paradigm are modeled as parallel STT approaches joined by a data association step. However, even for the simple case of a single target, well-known filtering approaches, *e.g.* linear Kalman filter (KF) [24] or UKF, can not be applied directly [46]. The reason for this are detection origin uncertainties, FP and FN detections.

A simple solution to this problem is the nearest neighbor (NN) filter [3, 9], which uses the closest detection in terms of spatial distance to each predicted state, *e.g.* [1]. However, such a setup is prone to lose tracks in case of wrong detection-to-track assignments due to FPs and FNs. An improved version is the probabilistic data association (PDA) filter [2, 4]. It uses assignment probabilities of certain detections in each frame and applies the state estimation filter with weighted detections to all targets individually [3, 46]. This improves the results in cluttered environments. However, both filters, NN and PDA, are designed for STT and should be used in MTT problems with clearly separable targets only.

In contrast to this local data assocation, global strategies consider all detections and tracked targets in every frame. The most common approaches are global nearest neighbor (GNN) and joint probabilistic data association (JPDA) [15]. While the former solves a minimization problem on given costs w.r.t. distance, intersection over union (IoU) or likelihood, the latter is an extension to the PDA filter. It performs a weighted update including all detections within a certain gating region simultaneously regarding all tracks. For example, Choi *et al.* [10] combined GNN association with a linear KF. Their association criterion is based on a weighted sum of target distance and size. In contrast, Sharma *et al.* [44] proposed a tracker without filtering which solves the association problem via the Hungarian [37] algorithm. They devised several costs from 3D cues, which are directly learned from monocular images.

Commonly used filtering approaches typically employ a single linear motion model, *e.g.* constant velocity (CV). However, traffic participants do not always act in a linear way. Hence, the interacting multiple model (IMM) [8] approach enables switching various models representing different motion patterns. This includes, for example, the *coordinated turn* modeled by the constant turn rate velocity (CTRV) model and static or slow movement modeled by the random model [31]. Rachman [40] proposed a tracker for traffic scenarios based on unscented Kalman filtering with an IMM and JPDA, which inspired our baseline MTT framework. However, his evaluation is restricted to a specific environmental scenario with few selected KITTI sequences and thus, complicates deducing insights on the general applicability for autonomous driving.

The main drawback of these approaches is the missing ability to handle a variable number of tracks. Stacking all tracks in a single state vector, on the one hand, restricts trackers to a fixed number of targets. Initiating a new filter for each target, on the other hand, requires an external track management. Alternatively, the multiple hypothesis tracker (MHT) [7, 41] builds hypotheses for all track-to-measurement assignments over time including the possibility of initiating and terminating tracks. However, the complexity of MHT grows exponentially with each time

step, which requires complexity reduction, *e.g.* via hypothesis pruning or track merging [11, 12, 25].

Sequential Monte Carlo (SMC) methods [14, 13, 36], like the particle filter [18, 34] can also be directly used for state estimation. This filter usually performs better in non-linear/non-Gaussian environments because it approximates the *posterior* PDF by a finite set of particles. One drawback of these approaches, however, is the high computational complexity.

Deep Learning: A more recent research direction is to leverage deep neural networks (DNNs) for MTT. They have gained a lot of attention within the last years because of their impressive performance in many computer vision challenges, especially object detection and classification [28].

Despite the fact that AD requires tracking in 3D world coordinates because vehicles depend on spatial information, a lot of tracking approaches are designed and evaluated in the image space. Sharma *et al.* [44], for example, leverage 2D and 3D cues from monocular images to perform tracking in the 2D image space. Another 2D approach was proposed by Gündüz and Acarman [19]. They exploit image features to find similarities between consecutive frames. In contrast, Zhang *et al.* [50] used 3D information from point clouds fused with image features even though they perform tracking in 2D.

On the other hand, one way to track in 3D world coordinates from monocular images only, is to estimate the distances between ego-vehicle and detected targets [42]. End-to-end trainable models using 3D LiDAR data [33] and additional RGB images [16] were also proposed. Hu *et al.* [20] estimate 3D bounding boxes from a sequence of images and track them with a trained long short-term memory (LSTM) network. A universal tracking approach based on RNNs was proposed by Milan *et al.* [35]. Their end-to-end trainable network represents equal structures like well-known Bayesian filters and processes simple bounding box inputs.

In contrast to these, we leverage the best of both worlds, *i.e.* combining learned motion/association models and well understood filtering techniques.

3. Multiple Object Tracker

Given a sequence of noisy 3D bounding boxes, we want to track all objects of interest trough time. Each target is represented by its current state which is modeled by a random variable. Such a task can be seen as a dynamic state estimation problem. For this purpose, we model a single tracker for each target with the state-space approach which allows state estimation from noisy detections. Thus, a dynamical model describes the state transition over time and a measurement model relates detections to the state.

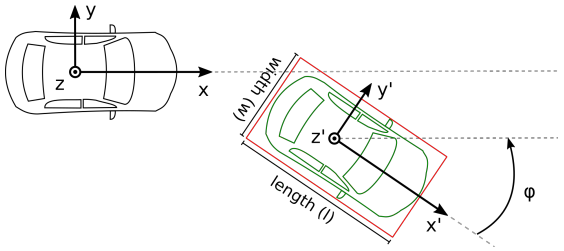

Figure 2. Sensor model [17] with ego-vehicle in black and an exemplary target car in green.

A well-developed framework for this problem is the Bayesian filter for which computational traceable solutions (*i.e.* KF, UKF) exist. It can be applied in a recursive manner to handle incoming detections for each time frame which is crucial for an online tracker. In the following we discuss the specific combination of such filters we exploit for target tracking. Additionally, we replace parts of the filters/trackers with data-driven RNN models.

3.1. Bayesian Tracking Framework

Representations: Object detections are represented by 3D bounding boxes, relative to the ego-vehicle's center coordinates x, y, z as illustrated in Fig. 2.

We further define the state vector at time t as $\mathbf{x}_t = (\text{pos}_t, \omega_t, v_t, \dot{\omega}_t, \text{bb}_t, \varphi_t)^T$, where pos_t denotes the center coordinates x_t, y_t and z_t, bb_t denotes the bounding box dimension, *i.e.* width w_t, length l_t and height h_t, and φ_t denotes the bounding box orientation. Additionally, the state vector contains non-observable parameters: Steering angle ω_t, velocity v_t and turn rate $\dot{\omega}_t$. Notice, we use different parameters for steering angle and bounding box orientation to support scenarios where ego-motion measurements are not available.

The measurement vector $\mathbf{z}_t$ contains observable parameters which can be obtained from the object detector at time t, *i.e.* $\mathbf{z}_t = (x_t, y_t, z_t, w_t, l_t, h_t, \varphi_t)^T$.

Unscented Kalman Filter (UKF): is a computationally tractable solution of the Bayesian filter which allows non-linear dynamical models. The main idea of the UKF is to propagate a fixed number of appropriately chosen weighted sample points – so-called sigma points – through a non-linear func-

tion by using the Unscented Transformation (UT). This process does not need an analytical derivation of dynamic and measurement functions. We leverage the scaled UT [22] which ensures a positive semi-definite covariance matrix.

Thus, we first need to determine $2n+1$ sigma points $\mathcal{X}_{i,t-1} \in \mathbb{R}^n$ with $i \in \{0,\dots,2n\}$ for n state variables, weights $\mathbf{w}^{(m)} \in \mathbb{R}^n$ for state mean $\hat{\mathbf{x}}_{t-1|t-1}$, and weights $\mathbf{w}^{(c)} \in \mathbb{R}^n$ for the state covariance matrix $\hat{\mathbf{P}}_{t-1|t-1}$. These weights depend on the scaling factor $\lambda = \alpha^2(n+\kappa) - n$, where α controls the spread of the sigma points around the mean. The remaining parameters κ and β represent another scaling and prior knowledge about the state distribution, respectively. To avoid sampling non-local effects under strong nonlinearities, the parameter should be $0 \leq \alpha \leq 1$. Furthermore, positive semi-definiteness can be guaranteed by choosing the parameter $\kappa \geq 0$. A typically good choice for state estimation problems is $\kappa = 0$. Finally, for Gaussian distributions $\beta = 2$ is optimal, otherwise it should be non-negative.

We further use the scaling parameters to sample scaled sigma points $\mathcal{X}_{i,t-1} \in \mathbb{R}^n$ with $i \in \{0,\dots,2n\}$ as in [22] from the previous *posterior* state $\hat{\mathbf{x}}_{t-1|t-1}$ and covariance $\mathbf{P}_{t-1|t-1}$. Afterwards, we propagate the sigma points through a potentially non-linear function $f(\cdot)$ representing the dynamic model and use the propagated sigma points

$$\mathcal{X}_{i,t|t-1} = f(\mathcal{X}_{i,t-1}) \quad \forall i \in \{0,\dots,2n\}, \tag{1}$$

to calculate the predicted mean $\hat{\mathbf{x}}_{t|t-1}$ and covariance $\mathbf{P}_{t|t-1}$ as

$$\hat{\mathbf{x}}_{t|t-1} = \sum_{i=0}^{2n} \mathbf{w}_i^{(m)} \mathcal{X}_{i,t|t-1}, \quad \text{and} \tag{2a}$$

$$\mathbf{P}_{t|t-1} = \sum_{i=0}^{2n} \mathbf{w}_i^{(c)} \mathbf{v}_x^T \mathbf{v}_x^T + \mathbf{Q}_t, \tag{2b}$$

with innovation $\mathbf{v}_x = \left(\mathcal{X}_{i,t|t-1} - \hat{\mathbf{x}}_{t|t-1}\right)$ and process noise covariance matrix $\mathbf{Q}_t \in \mathbb{R}^{n\times n}$.

The update step requires a new set of $2n+1$ sigma points $\mathcal{X}_{i,t} \in \mathbb{R}^n$ with $i \in \{0,\dots,2n\}$ for n state variables and weights for predicted state mean $\hat{\mathbf{x}}_{t|t-1}$ and corresponding covariance matrix $\hat{\mathbf{P}}_{t|t-1}$. Afterwards, we propagate these sigma points through the measurement function $h(\cdot)$

$$\mathcal{Z}_{i,t|t-1} = h\left(\mathcal{X}_{i,t}\right) \quad \forall i \in \{0,\dots,2n\}, \tag{3}$$

and build a weighted sum to obtain predicted *a priori* measurements $\hat{\mathbf{z}}_{t|t-1}$, the corresponding innovation covariance matrix $\mathbf{S}_t$ and the cross covariance $\mathbf{C}_t$ as

$$\hat{\mathbf{z}}_{t|t-1} = \sum_{i=0}^{2n} \mathbf{w}_i^{(m)} \mathcal{Z}_{i,t|t-1}, \quad \text{and} \tag{4a}$$

$$\mathbf{S}_t = \sum_{i=0}^{2n} \mathbf{w}_i^{(c)} \mathbf{v}_z \mathbf{v}_z^T + \mathbf{R}_t, \quad \text{and} \tag{4b}$$

$$\mathbf{C}_t = \sum_{i=0}^{2n} \mathbf{w}_i^{(c)} \left(\mathcal{X}_{i,t|t-1} - \hat{\mathbf{x}}_{t|t-1}\right) \mathbf{v}_z^T, \tag{4c}$$

with innovation $\mathbf{v}_z = \left(\mathcal{Z}_{i,t|t-1} - \hat{\mathbf{z}}_{t|t-1}\right)$ and measurement noise covariance matrix $\mathbf{R}_t \in \mathbb{R}^{q\times q}$.

Finally, we compute the *posterior* mean $\hat{\mathbf{x}}_{t|t}$ and covariance matrix $\mathbf{P}_{t|t}$ as

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{C}_t \mathbf{S}_t^{-1} (\mathbf{z}_t - \hat{\mathbf{z}}_{t|t-1}), \quad \text{and} \tag{5a}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{C}_t \mathbf{S}_t^{-1} \mathbf{C}_t^T. \tag{5b}$$

Interacting Multiple Model: The original UKF implementation supports only a single dynamic model which is often also referred to as motion model. Mostly, a single model is not able to cover the motion behavior of various traffic participants. One solution to this problem is the interacting multiple model (IMM). It is a traceable approximation to the intractable multiple model optimal Bayes filter [43], which is modeled as jump Markov non-linear system. Besides the states of a system, such a filter estimates mode probabilities, which defines how likely a motion model matches the system's behavior.

The multiple model optimal Bayes filter and its approximation assumes a fixed set $\mathcal{M} = \{M_j\}_{j=1}^r$ of r models, each processed by a recursive filter, *e.g.* linear KF or UKF. Model state transitions within the IMM are modeled by a first-order Markov chain represented by a state transition probability matrix $\Pi = [p_{i,j}] \in \mathbb{R}^{r\times r}$, where $p_{i,j}$ denotes the probability of a state transition from model i to model j. Hence, the main diagonal $p_{i,i}$ contains the probabilities to stay in the same model state.

Basically, a full cycle of the recursive IMM filter contains four steps: interaction, prediction, update and combination. First, we perform a probabilistic mixing with the *posterior* state estimate $\hat{\mathbf{x}}_{i,t-1|t-1}$ and covariance estimate $\mathbf{P}_{i,t-1|t-1}$ of the previous stage for each filter j as

$$\hat{\mathbf{x}}^{\star}_{j,t-1|t-1} = \sum_{i=1}^{r} \mu_{i|j,t-1}\, \hat{\mathbf{x}}_{i,t-1|t-1}, \quad \text{and} \tag{6a}$$

$$\mathbf{P}^{\star}_{j,t-1|t-1} = \sum_{i=1}^{r} \mu_{i|j,t-1} \left(\mathbf{P}_{i,t-1|t-1} + \mathbf{v}_i \mathbf{v}_i^T\right), \tag{6b}$$

with innovation $\mathbf{v}_i = (\hat{\mathbf{x}}_{i,t-1|t-1} - \hat{\mathbf{x}}^{\star}_{j,t-1|t-1})$. This results in a single initial state $\hat{\mathbf{x}}^{\star}_{j,t-1|t-1}$ and covariance $\mathbf{P}^{\star}_{j,t-1|t-1}$. The mixing probabilities $\mu_{i|j,t-1}$ can be calculated as

$$\mu_{i|j,t-1} = \frac{p_{i,j}\ \mu_{i,t-1}}{\mu^{-}_{j,t}}, \text{ with } \mu^{-}_{j,t} = \sum_{i=1}^{r} p_{i,j}\mu_{i,t-1}, \quad (7)$$

where $\mu^{-}_{j,t}$ is the predicted mode probability for filter j at the current time step, $\mu_{i,t-1}$ the mode probability of the previous time step, and $p_{i,j}$ the transition probability to switch from model i to j. In summary, the previous filters with their mode probability and the transition probability directly influence the initial state of each filter.

Afterwards, each of the j filters performs a separate prediction step as in Eq. (2) to obtain predicted states $\hat{\mathbf{x}}_{j,t|t-1}$ and corresponding covariance matrices $\mathbf{P}_{j,t|t-1}$. Additionally, this step yields predicted measurements $\hat{\mathbf{z}}_{j,t|t-1}$ and corresponding innovation covariance matrices $\mathbf{S}_{j,t}$ (see Eq. (4)). In order to obtain the *posterior* state $\hat{\mathbf{x}}_{j,t|t}$ and covariance matrix $\mathbf{P}_{j,t|t}$ each filter updates its state individually (see Eq. (5)). Within an IMM filter cycle, we then update the mode probabilities

$$\mu_{j,t} = \frac{\mathcal{L}_{j,t}\ \mu^{-}_{j,t}}{\sum_{i=1}^{r} \mathcal{L}_{i,t}\ \mu^{-}_{i,t}}, \quad \mathcal{L}_{j,t} = \mathcal{N}\left(\mathbf{z}_t; \hat{\mathbf{z}}_{j,t|t-1}, \mathbf{S}_{j,t}\right), \quad (8)$$

where $\mathcal{L}_{j,t}$ denotes the likelihood of a model fitting the assigned measurement $\mathbf{z}_t$.

Finally, we obtain the *posterior* state $\hat{\mathbf{x}}_{t|t}$ and its covariance $\mathbf{P}_{t|t}$ by combining the output of each filter, weighted by the mode probability

$$\hat{\mathbf{x}}_t = \sum_{j=1}^{r} \mu_{j,t}\ \hat{\mathbf{x}}_{j,t|t}, \quad \text{and} \quad (9a)$$

$$\mathbf{P}_t = \sum_{j=1}^{r} \mu_{j,t|t}\left(\mathbf{P}_{j,t|t} + \mathbf{v}_j\mathbf{v}_j^T\right), \quad (9b)$$

with innovation $\mathbf{v}_j = (\hat{\mathbf{x}}_{j,t|t} - \hat{\mathbf{x}}_{t|t})$. Note that this final result is not part of the filter recursion itself.

Within our tracking framework each tracker is initialized with three motion models as in [40]. First, the constant velocity (CV) model for straight motion. Second, the constant turn rate velocity (CTRV) model for coordinated turns, *e.g.* at cross ways. And third, the random (RAND) model represents static or slowly moving targets.

3.2. Data Association and Track Management

Following the tracking-by-detection scheme, our approach requires an association mechanism which joins tracks and detections in each time frame. To this end, we leverage two different approaches. A global exclusive method, *i.e.* Hungarian [37] algorithm, on the one hand, and a joint probabilistic approach, *i.e.* JPDA [2, 15] on the other hand.

For the global exclusive approach we use the negative intersection over union value of targets and detections to fill a cost matrix. Afterwards, the Hungarian algorithm finds associations by minimizing the total cost. Tracks without assigned detection and vice versa are managed by the track management.

The JPDA approach takes all tracks and detections of a certain time frame into account. First, it performs a gating mechanism based on the Mahalanobis distance between tracks and detections. Afterwards, it calculates association probabilities for each detection within a certain gating region. Finally, the used filter performs an update weighted by these association probabilities. Detections which do not belong to a gating region and tracks without assigned detections are also managed by the track management.

A simple track manager takes care of unassigned detections and tracks. Based on fixed thresholds, a track missing τ_m updates gets terminated. On the other hand, detections without assignment are used to initialize new tracks. Such newly initialized tracks are considered active after receiving τ_u updates.

3.3. RNN Models

Our modular framework allows to exchange different parts with data-driven models. This is, on the one hand, the IMM with hand-crafted motion models which can be replaced by a RNN trained for state prediction. Data association, on the other hand, depends on the intersection over union between tracks and detections. Hence, we train an encoder-decoder RNN to find association probabilities.

State Prediction: We leverage a two-layer LSTM network with 256 hidden units each and a fully connected output layer with two nodes and linear activation as shown in Table 1. The network takes the

Layer	Type	Input	Output	Activation
1	LSTM	4	256	-
2	LSTM	256	256	-
3	FC	256	2	identity

Table 1. Prediction network architecture.

center coordinates x_t, y_t of the bounding box ground area and the ego-vehicle movement in x and y direction as an input. The output is then the predicted

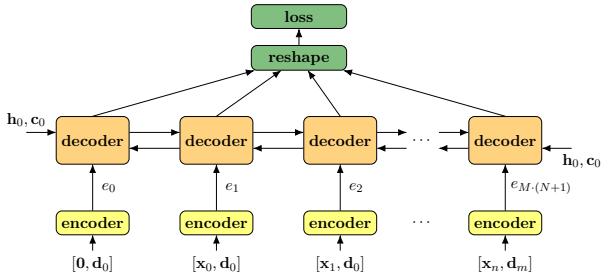

Figure 3. Encoder-Decoder structured network for data association using a bidirectional LSTM decoder [49].

center position of the bounding box ground area.

The model is trained with sequences of length 5 and optimized by the Adam [26] optimizer with a learning rate of 0.003 and the mean squared error loss. For inference, we replace the IMM by propagating the sigma points of the UKF through the trained model (see Eq. (1)).

Data Association: Matching a variable number of tracks and detections can be handled by an encoder-decoder model [49]. We adapted this approach for our purpose and thus, replace the encoder with a RNN and the MSE loss with a cross entropy error loss. Furthermore, we also learned the initial internal states $\mathbf{h}_0$ and $\mathbf{c}_0$ of the decoder while training.

The input to this network are permutations of all matching combinations between N tracks and M detections including the case that a detection belongs to no track. Fig. 3 shows the model structure. Each permutation contains the last 5 states $\mathbf{x}$ of a track and one detection $\mathbf{d}$. All entries contain the center position x, y of the bounding box ground area and its top-view dimension, *i.e.* width w and length l.

The encoded permutations $\mathbf{e}_i$ of size 64 are the input of the decoder, which is composed of one bidirectional LSTM layer with 64 hidden units and two fully-connected layers resulting in a single output for each permutation pair. After softmax activation and reshaping the output, we get a cost matrix representing the detection-to-track assignment probability.

Because the network is not able to learn the one-to-one constraint between tracks and detections, we apply the Hungarian algorithm on the cost matrix at inference time. Furthermore, we remove assignments with a probability < 0.5 and assignments to the dummy track with probability > 0.5.

Training is also done with the Adam optimizer and a learning rate of 0.003. To avoid overfitting, we apply early stopping and data augmentation, *i.e.* mirroring along the x-axis and adding noise.

4. Experimental Results

Dataset: To demonstrate our multi-target tracking framework for AD scenarios, we evaluate it on the publicly available KITTI dataset [17] which provides various environment categories, *i.e. City*, *Residential* and *Road*. The dataset contains RGB image sequences and LiDAR data with corresponding 3D bounding box annotations for eight different object classes, *i.e. Car*, *Pedestrian*, *Cyclist*, *Van*, *Tram*, *Person sitting*, *Truck* and *Misc*.

The KITTI dataset contains two partly overlapping datasets: The *tracking* dataset and the *raw* dataset. The former consists of 21 training sequences and 29 test sequences and the latter contains 38 sequences, sorted by environment categories. For evaluations on the *tracking* dataset, we apply the widely used train/validation split [38] since there are no public annotations for the corresponding test data. For evaluations on the *raw* dataset, we carefully select a set of sequences[1] containing all environmental categories as well as under-represented object classes, *i.e. Cyclist*, *Truck* and *Tram*. We further ensure that no training sequence is part of the validation set.

Performance measures: We employ the widely used CLEAR measures [5], namely Multiple Object Tracking Precision (MOTP) and Accuracy (MOTA). MOTP reflects the tracker's precision *wrt.* object locations and dimensions, whereas MOTA states the overall tracking ability. MOTA can be described as the consistent labeling of objects over time and takes false positive trajectories (FP), false negative trajectories (FN) and identity switches (IDS) into account. Additionally, we report the common track quality measures [32] which describe the coverage of tracks as either mostly tracked (MT), partly tracked (PT) or mostly lost (ML).

Baseline: In addition to our combined filter model, we implement a 3D version of SORT [6]. Concurrently to our work, a similar SORT extension [48] was submitted to KITTI[2] (approximately 5–6 % lower MOTA than the current leaders). This allows us to compare our evaluations to state-of-the art approaches listed in this leaderboard.

Notation: For each experiment, we denote a tracker configuration by the respective data associa-

[1] We use sequences 0001, 0005, 0014, 0018, 0060, 0084, 0020, 0039, 0064, and 0070 of KITTI *raw* as validation set.

[2] `http://www.cvlibs.net/datasets/kitti/eval_tracking.php`

	IDS	MOTA	MOTP	FPS
3D SORT	53	76.5%	**75.5%**	**412**
GNN	85	75.9%	74.6%	74
JPDA	99	72.1%	73.4%	50
RNN_{DA}	**48**	76.2%	74.3%	33
RNN_{Pr} – GNN	68	**77.2%**	71.5%	40
RNN_{Pr} – RNN_{DA}	78	75.6%	71.5%	34

Table 2. Evaluation of all tracker configurations on KITTI *raw* dataset regarding precision, accuracy, ID switches and runtime (without the object detection step) in frames per second (FPS). **Bold** scores denote the best results.

tion scheme, *e.g.* GNN denotes the IMM-UKF baseline with GNN [37]. RNN_{DA} denotes the learned data association network. Configurations which use the data-driven state prediction model are denoted by the additional prefix RNN_{Pr}.

4.1. Realistic Traffic Scenario

MTT for AD must perform well for all kinds of object classes and environments. Hence, we evaluate our trackers on the KITTI *raw* dataset, considering all traffic participants for which annotated ground truth is available. Note that we use the corresponding training set to optimize all parameters which are then fixed for all further experiments.

KITTI *raw* dataset: The results in Table 2 show that data-driven models improve the tracking performance in different ways. The learned data association model, on the one hand, produces the lowest number of ID switches and the prediction model outperforms the baseline by approx. 0.7% regarding MOTA on the other hand. Fig. 4 reveals that the

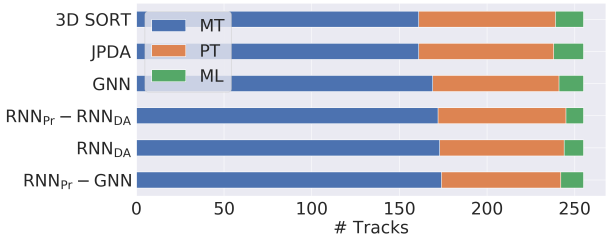

Figure 4. Track coverage of all approaches on the selected sequences of the KITTI *raw* dataset.

learned prediction model ensures also a high coverage of tracks. Additionally, it shows that the learned association model minimizes the number of totally lost tracks. Notwithstanding, we observe that the threshold values regarding initiation and termination of tracks highly influence the number of MT targets, although this only slightly affects the overall MOTA.

	IDS	MOTA	MOTP	FPS
3D SORT	77	63.1%	73.1%	**657**
GNN	46	71.6%	**73.2%**	98
JPDA	63	60.0%	67.8%	77
RNN_{DA}	**43**	**72.1%**	71.2%	54
RNN_{Pr} – GNN	65	72.0%	66.8%	52
RNN_{Pr} – RNN_{DA}	72	71.7%	65.9%	48

Table 3. Evaluation on KITTI *raw* dataset, where we omit detections of every second frame.

Fig. 5 shows qualitative results for RNN_{Pr} – RNN_{DA}.

We observed that evaluating solely on the main object classes (*i.e.* cars, pedestrians and cyclists) shows only a minor performance improvement (overall 1–2 % MOTA). Nevertheless, we evaluate on all object classes since they are all important for reliable perception in AD.

Dropping Detections: In order to evaluate the state prediction quality of all trackers, we drop detections for selected frames. Because of periodically missing detections, updates in consecutive time frames are not possible. Thus, we adapt the thresholds for initialization and termination of tracks within our track management. We set the threshold for initialization to 1 which causes an immediate initialization and increase the threshold for termination by the number of frames without detection.

Table 3 shows the results for omitting all detections in every second frame. We notice a significant decrease w.r.t. MOTA for the baseline approach and the tracker with JPDA. While the former is limited

Figure 5. Qualitative results of sequence '0005' from the KITTI *raw* dataset. The illustration shows colored 3D bounding boxes, each color representing a track instance.

	IDS	MOTA	MOTP	FPS
3D SORT	65	55.7%	**72.5%**	**826**
GNN	**28**	63.0%	71.5%	112
JPDA	54	48.4%	61.7%	107
RNN_{DA}	44	67.2%	68.9%	73
RNN_{Pr} – GNN	48	63.8%	61.1%	60
RNN_{Pr} – RNN_{DA}	53	**67.3%**	58.7%	56

Table 4. Evaluation on KITTI *raw* dataset, where we omit detections of every second and third frame.

by a linear motion model, the latter suffers from its static configuration. In contrast, configurations with our learned components lose only $\approx 4\% - 5\%$ while SORT loses more than 13% accuracy. This results in an increase of ML tracks as illustrated in Fig. 6.

Table 4 shows the results for omitting all detections in every second and third frame. Again, SORT and the JPDA tracker perform worse and the number of ML tracks for these trackers is two times higher as for our best performing approach.

4.2. Different Detectors

KITTI *tracking* dataset: For a better comparability to state-of-the-art approaches, we also evaluate on the widely used validation split [38] of the KITTI *tracking* dataset. Note, however, that a direct comparison is not possible as most approaches evaluate in 2D image space and only consider the three most well-represented object classes (cars, pedestrians and cyclists). Table 5 shows results for all object classes which are approximately 9% worse in comparison to our previous evaluations. This can be contributed to the larger number of highly crowded pedestrian scenes which cause frequent detection errors due to heavy occlusions.

So far, all our reported results leverage Frustum PointNets [39] detections. Table 6 demonstrates the effect of using PointRCNN [45] instead. The results are similar to Frustum PointNets (Table 5). However,

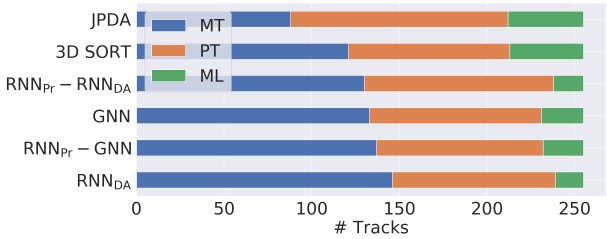

Figure 6. Track coverage of all approaches on the selected sequences of the KITTI *raw* dataset assuming omitted detections of each second frame.

	IDS	MOTA	MOTP	FPS
CIWT [38] (cars)	26	74.38%	82.85%	-
CIWT [38] (ped.)	41	61.87%	78.85%	-
3D SORT	160	67.2%	71.6%	**358**
GNN	191	**68.9%**	71.0%	50
JPDA	**103**	57.8%	**72.2%**	34
RNN_{DA}	185	64.5%	70.3%	27
RNN_{Pr} – GNN	229	68.7%	67.0%	30
RNN_{Pr} – RNN_{DA}	240	64.0%	66.1%	27

Table 5. Evaluation on the validation split of the KITTI *tracking* dataset. Note that [38] only evaluate on selected, well-represented object classes.

	IDS	MOTA	MOTP	FPS
3D SORT	64	66.4%	80.9%	**400**
GNN	67	**68.5%**	81.2%	62
JPDA	**48**	54.8%	**81.4%**	45
RNN_{DA}	77	60.2%	**81.4%**	31
RNN_{Pr} – GNN	91	68.2%	76.7%	36
RNN_{Pr} – RNN_{DA}	128	62.5%	76.3%	32

Table 6. Evaluation on the validation split of the KITTI *tracking* dataset using PointRCNN [45] detections.

we observe a performance decrease for models with RNN_{DA}. This can be contributed to the model training, since RNN_{DA} was trained using Frustum PointNets detections. Additionally, the overall improved MOTP results reveal a significantly better bounding box orientation estimation of PointRCNN compared to Frustum PointNets.

5. Conclusion

In this paper, we investigate several tracking aspects for AD within a combined Bayesian filter-based MTT approach. In particular, we leverage the IMM combined with UKF, as well as different data association methods. In order to increase tracking robustness despite unreliable detections, we exchange the state prediction and data association with data-driven models. Our evaluations show competitive results to state-of-the-art approaches and an improved robustness on the challenging KITTI dataset.

Acknowledgements

This project was partially funded by the Austrian Research Promotion Agency (FFG) under the project DGT (860820) and the Christian Doppler Laboratory for Embedded Machine Learning.

References

[1] A. Asvadi, P. Peixoto, and U. Nunes. Detection and Tracking of Moving Objects Using 2.5D Motion Grids. In *Proc. ITSC*, 2015. 1, 2

[2] Y. Bar-Shalom, F. Daum, and J. Huang. The Probabilistic Data Association Filter: Estimation in the presence of measurement origin uncertainty. *IEEE Control Sys.*, 29(6):82–100, 2009. 2, 5

[3] Y. Bar-Shalom and X.-R. Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing, first edition, 1995. 2

[4] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975. 2

[5] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *JIVP*, 2008(1):1–10, 2008. 6

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple Online and Realtime Tracking. In *Proc. ICIP*, 2016. 1, 6

[7] S. S. Blackman. Multiple Hypothesis Tracking for Multiple Target Tracking. *IEEE MAES*, 19(1 II):5–18, 2004. 2

[8] H. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE TAC*, 33(8):780–783, 1988. 2

[9] S. Challa, M. R. Morelande, D. Musicki, and R. J. Evans. *Fundamentals of Object Tracking*. Cambridge University Press, first edition, 2011. 2

[10] J. Choi, S. Ulbrich, B. Lichte, and M. Maurer. Multi-Target Tracking using a 3D-Lidar sensor for Autonomous Vehicles. In *Proc. ITSC*, 2013. 2

[11] I. J. Cox. A review of statistical data association techniques for motion correspondence. *IJCV*, 10(1):53–66, 1993. 3

[12] I. J. Cox and S. L. Hingorani. Efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE TPAMI*, 18(2):138–150, 1996. 3

[13] B. Fortin, R. Lherbier, and J.-C. Noyer. A Model-Based Joint Detection and Tracking Approach for Multi-Vehicle Tracking with Lidar Sensor. *IEEE TITS*, 16(4):1883–1895, 2015. 3

[14] B. Fortin, J. C. Noyer, and R. Lherbier. A Particle Filtering Approach for Joint Vehicular Detection and Tracking in LiDAR data. In *Proc. I2MTC*, 2012. 3

[15] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *J. Ocean. Eng.*, 8(3):173–184, 1983. 2, 5

[16] D. Frossard and R. Urtasun. End-to-end Learning of Multi-sensor 3D Tracking by Detection. In *Proc. ICRA*, 2018. 1, 3

[17] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 32(11):1231–1237, 2013. 2, 3, 6

[18] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F*, 140(2):107, 1993. 3

[19] G. Gündüz and T. Acarman. A Lightweight Online Multiple Object Vehicle Tracking Method. In *Proc. IV*, 2018. 3

[20] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krähenbühl, T. Darrell, and F. Yu. Joint Monocular 3D Detection and Tracking. In *Proc. ICCV*, 2019. 3

[21] J. Janai, F. Güney, A. Behl, and A. Geiger. Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. *arXiv CoRR*, abs/1704.05519, 2017. 1

[22] S. J. Julier. The scaled unscented transformation. In *Proc. ACC*, 2002. 4

[23] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proc. IEEE*, 92(3):401–422, 2004. 2

[24] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.*, 82(1):35, 1960. 2

[25] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple Hypothesis Tracking Revisited. In *Proc. ICCV*, 2015. 3

[26] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. ICLR*, 2015. 6

[27] S. Krebs, B. Duraisamy, and F. Flohr. A survey on leveraging deep neural networks for object tracking. In *Proc. ITSC*, 2017. 2

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NeurIPS*, 2012. 3

[29] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proc. CVPR*, 2019. 1

[30] P. Lenz, A. Geiger, and R. Urtasun. FollowMe: Efficient online min-cost flow tracking with bounded memory and computation. In *Proc. ICCV*, 2011. 1

[31] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. Part I: Dynamic models. *IEEE TAES*, 39(4):1333–1364, 2003. 2

[32] Y. Li, C. Huang, and R. Nevatia. Learning to associate: HybridBoosted Multi-target Tracker for Crowded Scene. In *Proc. CVPR*, 2009. 6

[33] 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 *Proc. CVPR*, 2018. 3

[34] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proc. ICCV*, 1999. 3

[35] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online Multi-Target Tracking Using Recurrent Neural Networks. In *Proc. AAAI*, 2017. 3

[36] N. Morales, J. Toledo, L. Acosta, and J. Sanchez-Medina. A Combined Voxel and Particle Filter-Based Approach for Fast Obstacle Detection and Tracking in Automotive Applications. *IEEE TITS*, 18(7):1824–1834, 2017. 3

[37] J. Munkres. Algorithms for the Assignment and Transportation Problems. *JSIAM*, 5(1):32–38, 1957. 2, 5, 7

[38] A. Osep, W. Mehner, M. Mathias, and B. Leibe. Combined Image- and World-Space Tracking in Traffic Scenes. In *Proc. ICRA*, 2017. 6, 8

[39] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *Proc. CVPR*, 2018. 1, 8

[40] A. S. A. Rachman. 3D-LIDAR Multi Object Tracking for Autonomous Driving. Master's thesis, Delft University of Technology, Center for Systems and Control, 2017. 2, 5

[41] D. Reid. An algorithm for tracking multiple targets. *IEEE TAC*, 24(6):843–854, 1979. 2

[42] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granstrom. Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering. In *Proc. IV*, 2018. 3

[43] M. Schreier. *Bayesian environment representation, prediction, and criticality assessment for driver assistance systems*. PhD thesis, Technische Universität Darmstadt, Department of Electrical Engineering and Information Technology, 2017. 4

[44] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna. Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking. In *Proc. ICRA*, 2018. 1, 2, 3

[45] S. Shi, X. Wang, and H. Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. In *Proc. CVPR*, 2019. 1, 8

[46] B.-N. Vo, M. Mallick, Y. Bar-shalom, S. Coraluppi, R. Osborne, R. Mahler, and B.-T. Vo. *Multitarget Tracking*, pages 1–15. John Wiley & Sons, Inc., first edition, 2015. 2

[47] E. A. Wan and R. van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proc. AS-SPCC*, 2000. 2

[48] X. Weng and K. Kitani. A Baseline for 3D Multi-Object Tracking. *arXiv CoRR*, abs/1907.03961, 2019. 6

[49] K. Yoon, D. Y. Kim, Y. C. Yoon, and M. Jeon. Data association for multi-object tracking via deep neural networks. *Sensors*, 19(3):1–15, 2019. 6

[50] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy. Robust Multi-Modality Multi-Object Tracking. In *Proc. ICCV*, 2019. 1, 3