2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

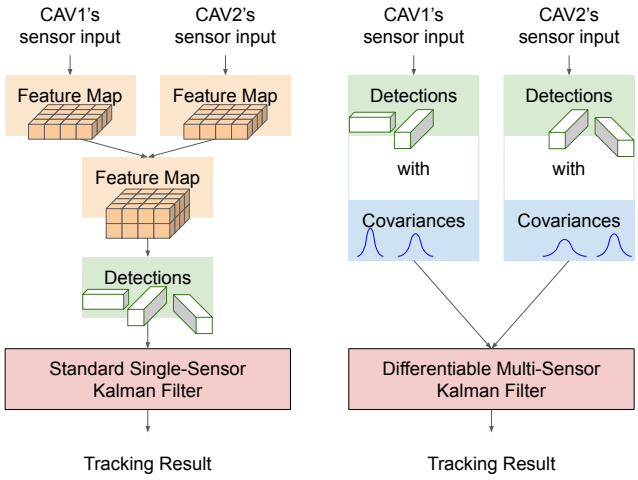
Probabilistic 3D Multi-Object Cooperative Tracking for Autonomous Driving via Differentiable Multi-Sensor Kalman Filter

Hsu-kuang Chiu[1], Chien-Yi Wang[2], Min-Hung Chen[2], and Stephen F. Smith[1]

***Abstract*— Current state-of-the-art autonomous driving vehicles mainly rely on each individual sensor system to perform perception tasks. Such a framework's reliability could be limited by occlusion or sensor failure. To address this issue, more recent research proposes using vehicle-to-vehicle (V2V) communication to share perception information with others. However, most relevant works focus only on cooperative detection and leave cooperative tracking an underexplored research field. A few recent datasets, such as V2V4Real, provide 3D multi-object cooperative tracking benchmarks. However, their proposed methods mainly use cooperative detection results as input to a standard single-sensor Kalman Filter-based tracking algorithm. In their approach, the measurement uncertainty of different sensors from different connected autonomous vehicles (CAVs) may not be properly estimated to utilize the theoretical optimality property of Kalman Filter-based tracking algorithms. In this paper, we propose a novel 3D multi-object cooperative tracking algorithm for autonomous driving via a differentiable multi-sensor Kalman Filter. Our algorithm learns to estimate measurement uncertainty for each detection that can better utilize the theoretical property of Kalman Filter-based tracking methods. The experiment results show that our algorithm improves the tracking accuracy by 17% with only 0.037x communication costs compared with the state-of-the-art method in V2V4Real. Our code and videos are available at the URL and the URL.**

I. Introduction

Autonomous driving vehicles have been deployed for commercial usage in certain regions of the world due to the significant advance of the relevant technologies over the past decade. However, currently deployed large-scale autonomous driving systems mainly rely on the individual autonomous vehicle's perception system. Such a system could have difficulty detecting and tracking other vehicles, pedestrians, cyclists, or any type of vulnerable road users due to occlusion or sensor failure. To address such safety concerns, more and more research datasets [1]–[3] and algorithms [4]–[11] for cooperative perception have been published in recent years. In cooperative perception for autonomous driving, each connected autonomous vehicle (**CAV**) can share its own raw sensor input (**early fusion**), encoded perception features (**intermediate fusion**), or detection results (**late fusion**) with others via vehicle-to-vehicle (V2V) communication in order to achieve better overall perception coverage and accuracy. For the most part, work in cooperative perception has focused on cooperative detection [4]–[11], proposing different types of **intermediate fusion** cooperative detection algorithms and demonstrating promising improvements in detection accuracy over both **no fusion** and **late fusion** baseline methods.



(a) V2V4Real [1] (b) DMSTrack (ours)

Fig. 1: Cooperative tracking architecture comparison.

Much less attention to date has been given to the problem of 3D multi-object cooperative tracking. Some datasets, such as V2V4Real [1], do provide 3D multi-object cooperative tracking benchmarks. But these benchmarks typically apply the aforementioned cooperative detection algorithms and then use the detection results as input to typical 3D multi-object tracking algorithms derived from the standard single-sensor Kalman Filter [12], as shown in Figure 1a. The measurement uncertainty of different sensors from different CAVs may not be properly estimated to utilize the theoretical optimality property of Kalman Filter-based tracking algorithms.

To address this problem, we propose a novel algorithm: Differentiable Multi-Sensor Kalman Filter for 3D Multi-Object Cooperative Tracking (**DMSTrack**), as shown in Figure 1b and Figure 2. Our algorithm is designed to be capable of **estimating observation noise covariance of each detection from different CAVs to better take advantage of the Kalman Filter's theoretical optimality property**: minimizing the expected error of state estimation.

To achieve such an algorithm design goal, we reconsider the aforementioned cooperative perception fusion approaches, and determine that, due to its extensibility, the **late fusion** approach particularly fits our algorithm design principle. In this approach, each CAV first performs its individual object detection using its own sensor input. So we can easily reformulate the cooperative tracking problem in

[1] Robotics Institute of Carnegie Mellon University. [2] NVIDIA Research.
This research was funded in part by the CMU Safety21 University Transportation Center, the NSF ACCESS program, and the Google cloud research credit program.

this approach as a state estimation problem in a multi-sensor system and design our algorithm upon the foundation of prior works in this field. The original multi-sensor Kalman Filter [13] work focused on the single-object tracking problem in a multi-sensor system. This research proves that sequentially processing each sensor's measurement on the same object with multiple Kalman Filter's update steps in a discrete timestep also achieves theoretically optimal state estimation results as long as the following precondition holds: each sensor's measurement is synchronized and statistically independent. Inspired by this idea, our first innovation is to extend [13] to have multi-object tracking capability by applying multiple data associations and Kalman Filter update steps to build our cooperative tracking algorithm, as shown in Figure 2. Although the theoretical guarantee property of [13] is for single-object tracking and the precondition may not always hold in cooperative perception driving datasets, our experiment still shows that our proposed new tracking algorithm provides promising performance improvement in comparison to the baseline methods in V2V4Real [1].

The second innovation of our proposed cooperative tracking algorithm is its ability to estimate the observation noise covariance of each detection from different CAVs. The previously developed Backprop Kalman Filter [14] proposes the use of a deep neural network to learn a single observation noise covariance matrix from images for the 2D single-object tracking problem in videos and the visual odometry problem. Compared to these problems, 3D multi-object cooperative tracking for autonomous driving is a much more complex problem. The measurement uncertainty of different bounding box variables of each detection could be related to the sensor system, the 3D object detection model, and the relative pose of the detected object to the sensor. Thus, we propose using local point cloud features and positional features as input to our designed covariance neural network to learn the observation noise covariance. We use the regression loss between the tracking result boxes and the ground-truth boxes to train our proposed differentiable multi-sensor Kalman Filter, as shown in Figure 2 and Figure 3. This approach further improves the cooperative tracking performance in the V2V4Real [1] dataset.

In terms of communication cost, our approach only needs to share bounding box and covariance information for each detection as shown in Figure 1b. So it has a much smaller communication cost compared with the **intermediate fusion** baseline methods in V2V4Real [1], which need to share scene-level Bird-Eye-View (BEV) feature maps with other CAVs as shown in Figure 1a.

In summary, we propose a novel probabilistic 3D multi-object cooperative tracking algorithm for CAVs via differentiable multi-sensor Kalman Filter: **DMSTrack**. Our model learns to estimate the covariance matrix for each detection from each individual CAV. The detection results with covariances are merged via multiple data associations and Kalman Filter update steps to generate the final cooperative tracking results in each timestep. We evaluate our proposed algorithm's performance in the V2V4Real [1] dataset, which is the first large-scale worldwide available cooperative perception real dataset for autonomous driving with 3D multi-object cooperative tracking benchmarks. Our proposed algorithm, **DMSTrack**, improves the tracking accuracy by 17% with only 0.037x communication costs compared with the state-of-the-art method CoBEVT [8] in V2V4Real [1].

II. Related Work

A. Single-Agent Autonomous Driving

Single-agent autonomous driving large-scale datasets [15]–[18] and algorithms [19]–[25] serve as important foundations for cooperative autonomous driving research. For example, we follow V2V4Real [1]'s late fusion approach to use PointPillar [19] as the single-agent 3D object detector in our experiments. Our algorithm also adopts some components from AB3DMOT [21], such as the data association criteria and the track life cycle management.

B. Cooperative Perception with CAVs

Cooperative perception for autonomous driving has gained more attention in the past few years. F-Cooper [4] is the first work that introduces feature-level fusion to improve CAVs' cooperative detection accuracy. V2VNet [6] adopts graph neural networks to aggregate intermediate features. AttFuse [5], V2X-ViT [7], and CoBEVT [8] develop varieties of attention and transformer-based models for cooperative detection. V2V4Real [1] is the first worldwide available public real-world dataset with 3D cooperative detection and tracking benchmarks, and it mainly applies the aforementioned cooperative detection algorithms' results as input to a standard single-sensor Kalman Filter-based tracking algorithm: AB3DMOT [21]. In addition to V2V4Real [1], a simulation dataset V2X-Sim [2] and a real dataset V2X-Seq [3] also provide cooperative tracking benchmarks. However, V2X-Sim [2]'s tracking benchmark degrades detection and tracking evaluation to BEV xy-axis-aligned 2D bounding box format without properly considering the center positions in the z-axis, heights, and orientations. V2X-Seq [3] is a new cooperative perception dataset but is not available worldwide. Therefore, evaluation on these two datasets is out of the scope of our current research. We focus on evaluating our algorithm in the V2V4Real [1] benchmark. Different from the benchmark's best-performing method CoBEVT [8] which is an intermediate fusion approach, our proposed method is a late fusion approach with better tracking accuracy and a lower communication cost.

C. Kalman Filters

The multi-sensor Kalman Filter [13] provides the theoretical foundation of our algorithm. Backprop Kalman Filter [14] inspires our neural network approach of learning covariance from sensor input to better take the advantage of the theoretical optimality property from [13]. Recent works [26], [27] applying Backprop Kalman Filter [14] only focus on single object tracking. To the best of our knowledge, we are the first to integrate both [13] and [14] to create the differentiable multi-sensor Kalman Filter model and extend it to build our

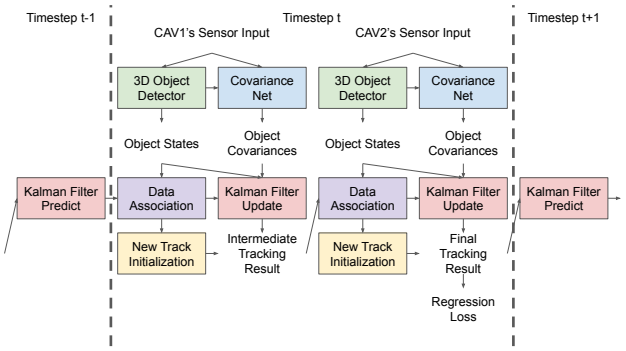


Fig. 2: Architecture diagram of our proposed cooperative tracking algorithm applied to a minimal example with two CAVs.

probabilistic 3D multi-object cooperative tracking algorithm for autonomous driving.

III. Method

Our proposed cooperative tracking architecture is illustrated in Figure 2 with a minimal example of two CAVs. At each timestep t, each CAV first feeds its own sensor input to a 3D object detector and our designed covariance neural network to detect object state means and estimate object state covariances in a decentralized manner. Then each CAV transforms its detected object states from its own local coordinate system to a global coordinate system and transmits the detection information to a centralized agent that maintains our proposed differentiable multi-sensor Kalman Filters to track multiple object states and covariances in the global coordinate system. The centralized agent can be any CAV or an external computing node.

The centralized agent then sequentially performs multiple data associations and Kalman Filter update steps with each CAV's detected object states and covariances, as shown in Figure 2. Once the centralized agent finishes processing the detection information from all CAVs, the latest tracking result is treated as the final cooperative tracking results at this timestep t. A Kalman Filter prediction step is then performed to predict the object states for the next timestep given the Kalman Filter's process model.

In the following subsections, we describe more details of our proposed differentiable multi-sensor Kalman Filter setting for 3D multi-object cooperative tracking, followed by the covariance neural network's input feature formation, model architecture, loss function, and training strategy.

A. Kalman Filter Setting

1) Object State: To apply Kalman Filters [12] to the 3D multi-object cooperative tracking for autonomous driving problem, we follow V2V4Real [1] and AB3DMOT [21] to represent a track state at timestep t by a vector of 10 variables as follows:

$$\mathbf{s}_t = (x_t, y_t, z_t, a_t, l_t, w_t, h_t, dx_t, dy_t, dz_t)^T, \tag{1}$$

where (x_t, y_t, z_t) is the center position of the bounding box, a_t is the rotation angle along the z-axis, (l_t, w_t, h_t) represents the length, width, and height of the bounding box. And (dx_t, dy_t, dz_t) contains the velocity information, which is used to predict the track state in the next timestep.

2) Prediction Step: In the Kalman Filter's prediction step, we use a constant velocity model as the process model.

The Kalman Filter's prediction step can be represented in the following matrix form:

$$\hat{\mu}_t = \mathbf{A}\mu_{t-1} \tag{2}$$

$$\hat{\Sigma}_t = \mathbf{A}\Sigma_{t-1}\mathbf{A}^T + \mathbf{Q} \tag{3}$$

where $\hat{\mu}_t \in \mathbb{R}^{10}$ is the predicted track state mean at timestep t, and $\mu_{t-1} \in \mathbb{R}^{10}$ is the estimated mean of the track state $\mathbf{s}_{t-1}$ at timestep $t-1$. The matrix $\mathbf{A} \in \mathbb{R}^{10\times10}$ is the state transition matrix of the Kalman Filter's process model. The matrix $\hat{\Sigma}_t \in \mathbb{R}^{10\times10}$ is the predicted state covariance at timestep t, and $\Sigma_{t-1} \in \mathbb{R}^{10\times10}$ is the estimated state covariance at timestep $t-1$. The matrix $\mathbf{Q} \in \mathbb{R}^{10\times10}$ is the process model noise covariance. We follow V2V4Real [1] and AB3DMOT [21] and use the same constant values for $\mathbf{Q}$.

3) Data Association and Update Step: At timestep t, each track could have zero, one, or more associate detections from multiple CAVs. To have a fair comparison, our algorithm's detection part uses V2V4Real [1]'s **late fusion** baseline's PointPillar [19] model checkpoint as the single-agent 3D object detector. Each detection is represented by 7 variables used as observation in the Kalman Filter as follows:

$$\mathbf{o}_t = (x_t, y_t, z_t, a_t, l_t, w_t, h_t)^T, \tag{4}$$

Our proposed algorithm then sequentially processes all CAV's detections with covariances via multi-sensor Kalman Filters with multiple data associations and update steps, as shown in Figure 2. For data association, we follow V2V4Real [1] and AB3DMOT [21] to use 3D Intersection-over-Union (3D IOU) as the similarity metric and Hungarian algorithm [28]. For each matched pair of a track and a detection, the Kalman Filter's update step works as follows:

$$\mathbf{S}_t = \mathbf{H}\hat{\Sigma}_t\mathbf{H}^T + \mathbf{R} \tag{5}$$

$$\mathbf{K}_t = \hat{\Sigma}_t\mathbf{H}^T\mathbf{S}_t^{-1} \tag{6}$$

$$\mu_t = \hat{\mu}_t + \mathbf{K}_t(\mathbf{o}_t - \mathbf{H}\hat{\mu}_t) \tag{7}$$

$$\Sigma_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\hat{\Sigma}_t, \tag{8}$$

where $\mathbf{S}_t \in \mathbb{R}^{7\times7}$ is the innovation covariance matrix, $\mathbf{H} = [\mathbf{I}\ \mathbf{0}] \in \mathbb{R}^{7\times10}$ is the observation model matrix, $\hat{\Sigma}_t \in \mathbb{R}^{10\times10}$ is the predicted state covariance from equation (3), $\mathbf{R} \in \mathbb{R}^{7\times7}$ is the observation noise covariance matrix for the detection, $\mathbf{K}_t \in \mathbb{R}^{10\times7}$ is the Kalman Gain, $\mu_t \in \mathbb{R}^{10}$ is the estimated state mean at timestep t as the tracking result, $\hat{\mu}_t \in \mathbb{R}^{10}$ is the predicted state mean from equation (2), $\mathbf{o}_t \in \mathbb{R}^7$ is the detection from equation (4), $\Sigma_t \in \mathbb{R}^{10\times10}$ is the estimated state covariance at timestep t, and $\mathbf{I}$ represents an identity matrix. In our algorithm, we assume all observation noise covariance $\mathbf{R} \in \mathbb{R}^{7\times7}$ and track initial state covariance $\Sigma_0 \in \mathbb{R}^{10\times10}$ matrices are all diagonal.

For unmatched detections and tracks, we perform the same track life cycle management approach as in V2V4Real

[1] and AB3DMOT [21]. Our algorithm repeats the update step after each data association to sequentially process the detection results of all CAVs and generates the final tracking results, as shown in Figure 2.

B. Input Feature Formation

To learn the observation noise covariance $\mathbf{R} \in \mathbb{R}^{7\times7}$ in equation (5) for every detection, we use two types of input features: the local BEV feature and the positional feature.

1) Local BEV Feature: Each detection's local BEV feature is extracted from the PointPillar [19] 3D object detector's BEV feature maps. Given an object detection's center position, we first find the corresponding cell in two BEV feature maps right before and after PointPillar [19]'s 2D CNN backbone. We extract two local BEV features respectively and merge them to generate this detection's local BEV feature $\mathbf{F}_{bev} \in \mathbb{R}^{320\times20\times20}$.

2) Positional Feature: Each detection's observation noise covariance could also be relevant to the object's relative position and orientation to the global coordinate system and the corresponding sensor CAV's local coordinate system. The coordinate transformation matrix between the CAV's local coordinate system and the global coordinate system could also provide useful information. Therefore, we propose the following novel positional feature extraction algorithm to capture the geometry information. First, we extract a positional feature vector $\mathbf{f}_{pos}$ with 18 variables for each detection as follows:

$$\mathbf{f}_{global} = (x, y, z, a, l, w, h, \sqrt{x^2+y^2})^T_{global} \quad (9)$$

$$\mathbf{f}_{local} = (x, y, z, a, \sqrt{x^2+y^2})^T_{local} \quad (10)$$

$$\mathbf{f}_{local\_to\_global} = (t_x, t_y, t_z, \theta_{yaw}, \sqrt{t_x^2+t_y^2})^T \quad (11)$$

$$\mathbf{f}_{pos} = \mathbf{f}_{global} \odot \mathbf{f}_{local} \odot \mathbf{f}_{local\_to\_global}, \quad (12)$$

where $\mathbf{f}_{global} \in \mathbb{R}^8$ consists of the detection variables, same as equation (4), appended by its 2D distance to the origin of the global coordinate system. The vector $\mathbf{f}_{local} \in \mathbb{R}^5$ consists of the detection position, orientation, and 2D distance to the origin in the corresponding CAV's local coordinate system. The vector $\mathbf{f}_{local\_to\_global} \in \mathbb{R}^5$ is derived from the coordinate transformation matrix between the corresponding CAV's local coordinate system and the global coordinate system, where $(t_x, t_y, t_z)^T$ represents the translation vector and θ_{yaw} is the rotation angle along the z-axis. Finally, $\mathbf{f}_{pos} \in \mathbb{R}^{18}$ is the positional feature vector of this detection and the notation $\odot$ represents the concatenation operator.

Once we have the positional feature vector $\mathbf{f}_{pos}$, we then generate the positional feature matrix $\mathbf{F}_{pos}$ using our modified version of positional encoding [29]. In our algorithm, each element x in $\mathbf{f}_{pos}$ is transformed to a positional encoded vector $\mathbf{x}$ by the following equations:

$$\bar{x} = normalize(x) \quad (13)$$

$$\mathbf{x}_{2i} = sin(\frac{\bar{x}}{2^{\frac{i}{d}}}) \ \forall \ i = 0, 1, ..., d-1 \quad (14)$$

$$\mathbf{x}_{2i+1} = cos(\frac{\bar{x}}{2^{\frac{i}{d}}}) \ \forall \ i = 0, 1, ..., d-1, \quad (15)$$

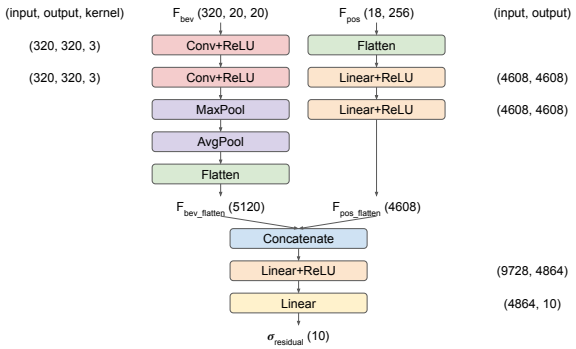


Fig. 3: Covariance Neural Network. The numbers represent features' tensor shapes and trainable layers' input, output, and kernel sizes.

where $\bar{x} \in [-\pi, \pi]$ is the normalized value of x by considering the min and max possible values of this variable, $d = 128$ is a constant number, $\mathbf{x} \in \mathbb{R}^{256}$ is the positional encoded vector, the subscripts $2i$ and $2i+1$ represent the indices of even and odd elements of the vector. We use the same approach to encode each element in $\mathbf{f}_{pos}$ and generate the final positional feature matrix $\mathbf{F}_{pos}$ as follows:

$$\mathbf{F}_{pos} = PositionalEncoding(\mathbf{f}_{pos}), \quad (16)$$

where $\mathbf{F}_{pos} \in \mathbb{R}^{18\times256}$ is the final positional feature of this detection.

C. Covariance Neural Network

Our covariance neural network takes each detection's local BEV feature $\mathbf{F}_{bev}$ and positional feature $\mathbf{F}_{pos}$ as input and generates output to estimate this detection's covariance, as shown in Figure 3. The estimated covariance is used to generate the observation noise covariance $\mathbf{R} \in \mathbb{R}^{7\times7}$ during the Kalman Filter's update step in equation (5) or the state covariance matrix $\Sigma_0 \in \mathbb{R}^{10\times10}$ for a newly initialized track.

The input features $\mathbf{F}_{bev} \in \mathbb{R}^{320\times20\times20}$ and $\mathbf{F}_{pos} \in \mathbb{R}^{18\times256}$ have very different numbers of elements. To better balance the contributions from these two features during training, we first use the same number of trainable layers to process them individually and make the processed flatten features $\mathbf{F}_{bev\_flatten} \in \mathbb{R}^{5120}$ and $\mathbf{F}_{pos\_flatten} \in \mathbb{R}^{4608}$ more equivalent in size.

Then we concatenate these two processed flatten features and use two linear layers with a rectified linear unit to generate the final network output. Instead of directly generating the covariance values, the network generates the residuals of standard deviations in order to make the training more numerically stable as follows:

$$diag(\mathbf{R}) = (sqrt(diag(\mathbf{R}_{def})) + \sigma_{residual}[1:7])^2, \quad (17)$$

where $diag$ is the operator that accesses diagonal elements of a matrix, $\mathbf{R} \in \mathbb{R}^{7\times7}$ is the detection's observation noise covariance matrix used in the Kalman Filter update step's equation (5), $sqrt$ represent the element-wise square root function, $\mathbf{R}_{def}$ is the constant default observation noise covariance matrix set to be an identity matrix, $\sigma_{residual} \in$

$\mathbb{R}^{10}$ is the covariance neural network output, the notation $[1:7]$ means accessing the first 7 elements of a vector, and the superscript 2 represents the element-wise square operator.

Similarly, when a new track is initialized from an unmatched detection, we set the track's initial state covariance matrix as follows:

$$diag(\Sigma_0) = (sqrt(diag(\Sigma_{0_{def}})) + \sigma_{residual})^2, \quad (18)$$

where $\Sigma_0 \in \mathbb{R}^{10\times10}$ is the state covariance matrix of a newly initialized track, $\Sigma_{0_{def}}$ is the constant default state covariance matrix set to be an identity matrix.

D. Loss Function and Training Strategy

To train the covariance neural network, we divide the training set sequences into sub-sequences, each of which has a length $T = 10$ timesteps. We run our proposed cooperative tracking algorithm on each sub-sequence and calculate the loss, as shown in Figure 2. When calculating the loss, we only consider a subset of tracks whose closest ground-truth object is within 2 meters. Assuming we have N_t tracks satisfy this condition at timestep t, and we use L2-norm to calculate the loss as follows:

$$L = \frac{1}{\sum_{t=1}^{T} N_t} \sum_{t=1}^{T} \sum_{i=1}^{N_t} ||(\mu_t^i[1:7] - gt_t^i)||_2, \quad (19)$$

where $L \in \mathbb{R}$ is the loss, $T = 10$ is the length of each training sub-sequence, N_t is the number of tracks satisfy the condition, $\mu_t^i[1:7] \in \mathbb{R}^7$ represents the first 7 variables of the ith track's state mean $\mu_t^i \in \mathbb{R}^{10}$ at timestep t, and $gt_t^i \in \mathbb{R}^7$ represents the track μ_t^i's closest ground-truth object variables. During training, this loss L can back-propagate through every timestep $t \in [1, T]$ to all covariance neural networks used by all CAVs via our proposed differentiable multi-sensor Kalman Filter. We use Adam [30] optimizer to train our model, with an initial learning rate of 0.001 and a weight decay of 0.00001. We clip the gradient norm to 1 and train our model for 20 epochs on the training split. During our development, we found that clipping the gradient norm is particularly important to avoid the exploding gradient problem and make the training of our proposed differentiable multi-sensor Kalman Filter numerically stable.

IV. Experimental Results

A. Datasets

We train our model and evaluate our cooperative tracking algorithm on the V2V4Real [1] dataset, which is the first worldwide available public real-world dataset for vehicle-to-vehicle perception with 3D tracking benchmarks. Each of the training, validation, and testing split has a total of $14,210$, $2,000$, and $3,986$ frames of data respectively from two vehicles driven simultaneously together. The data released to the public and used in V2V4Real [1] paper's benchmarks are the LiDAR point clouds and ground-truth annotations of the training and testing splits. The training split has 32 driving sequences and a total of 7105 timesteps of data, and the testing split has 9 driving sequences and a total of 1993 timesteps of data. The frame rate is 10Hz.

B. Evaluation Metrics

We use the same evaluation metrics from V2V4Real [1] and AB3DMOT [21]. The primary metric is the Average Multi-Object Tracking Accuracy (AMOTA). This AMOTA metric considers detection confidence scores and numbers of true positives, false positives, false negatives, and identity switches in a spectrum of different recall levels. Secondary evaluation metrics include Average Multi-Object Tracking Precision (AMOTP), scaled Average Multi-Object Tracking Accuracy (sAMOTA), Multi-Object Tracking Accuracy (MOTA), Mostly Tracked Trajectories (MT), and Mostly Lost Trajectories (ML). We use AB3DMOT [21]'s default evaluation criteria: 3D Intersection-over-Union with a threshold value of 0.25 when calculating the metrics.

C. Quantitative Results

1) Tracking Performance: Our proposed cooperative tracking algorithm (DMSTrack)'s performance can be seen in Table I. Since V2V4Real [1] only releases its cooperative detection code and mentions that it uses AB3DMOT [21] tracking algorithm without releasing its tracking code so far (as of ICRA2024 submission deadline), we implement our own V2V4Real [1] baseline by integrating CoBEVT [8] model checkpoint's cooperative detection results as the input to AB3DMOT [21]'s tracking code to have a fair comparison with our algorithm. From Table I, we can see that our implementation of CoBEVT [8] baseline has higher AMOTA than the V2V4Real [1] paper shows. More importantly, our proposed cooperative tracking method (DMSTrack) significantly outperforms our implemented CoBEVT [8] baseline in AMOTA by 6.36, which is a 17% relative improvement.

To figure out which parts of our algorithm contribute to the cooperative tracking performance gain, we conduct ablation experiments on variants of our algorithm, as shown in Table II. Table II's second row (**Constant covariance**) shows the result of our multi-sensor Kalman Filter tracking method but with AB3DMOT [21]'s default constant observation noise covariance matrix R and initial track state covariance Σ_0. We can see that this variant already outperforms V2V4Real [1]'s CoBEVT [8] baseline. We also show two more variants that use either only the local BEV feature $\mathbf{F}_{bev}$ or only the positional feature $\mathbf{F}_{pos}$ as input of the covariance neural network. We can see that each feature can provide useful information to let the network learn covariance matrices for different detections and achieve better tracking accuracy compared with the CoBEVT [8] baseline and our **Constant covariance** baseline. The best cooperative tracking performance is achieved when both types of features are used.

2) Communication Cost: Our cooperative tracking algorithm's communication cost is closer to V2V4Real [1] **late fusion**'s cost. In addition to transmitting 7 variables per detection box, our algorithm needs to transmit 10 more variables of the covariance residual values as shown in Figure 2. Therefore, our communication cost is $\frac{7+10}{7} \approx 2.43$ times of the V2V4Real [1]'s **late fusion**'s cost. So our communication cost is 0.0073 MB, which is just 0.037 times of the cost of V2V4Real [1]'s CoBEVT [8].

TABLE I: Cooperative tracking performance in the testing split of V2V4Real [1] dataset in comparison with baseline methods. The baseline methods' results are from the V2V4Real [1] paper. In each column, the best-obtained results are typesetted in boldface. (* Our baseline implementation of applying CoBEVT [8] detection results as input to AB3DMOT [21] tracking algorithm. Our code is available at this URL.)

Method	AMOTA ↑	AMOTP ↑	sAMOTA ↑	MOTA ↑	MT ↑	ML ↓	Cost (MB) ↓
No Fusion	16.08	41.60	53.84	43.46	29.41	60.18	**0**
Late Fusion	29.28	51.08	71.05	59.89	45.25	31.22	0.003
Early Fusion	26.19	48.15	67.34	60.87	40.95	32.13	0.96
F-Cooper [4]	23.29	43.11	65.63	58.34	35.75	38.91	0.20
AttFuse [5]	28.64	50.48	73.21	63.03	46.38	28.05	0.20
V2VNet [6]	30.48	54.28	75.53	64.85	48.19	27.83	0.20
V2X-ViT [7]	30.85	54.32	74.01	64.82	45.93	26.47	0.20
CoBEVT [8]	32.12	55.61	77.65	63.75	47.29	30.32	0.20
CoBEVT [8] (*)	37.16	57.20	84.54	84.14	57.07	15.83	0.20
DMSTrack (ours)	**43.52**	**57.94**	**91.50**	**88.32**	**68.35**	**13.19**	0.0073

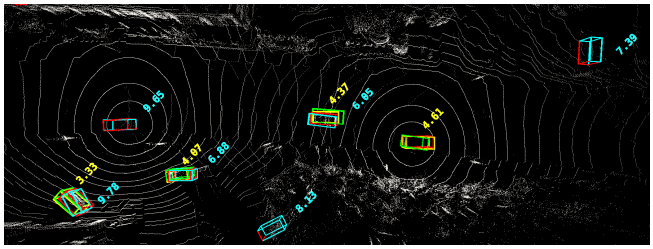
TABLE II: Ablation analysis on variants of our cooperative tracking algorithm.

Method	AMOTA ↑	AMOTP ↑	sAMOTA ↑	MOTA ↑	MT ↑	ML ↓	Cost (MB) ↓
CoBEVT [8] (*)	37.16	57.20	84.54	84.14	57.07	15.83	0.20
Constant covariance	41.51	55.87	89.37	**88.68**	66.67	13.67	**0.003**
Local BEV feature only	43.48	**58.05**	91.48	88.46	66.91	13.43	0.0073
Positional feature only	43.45	57.85	91.44	88.38	67.63	13.67	0.0073
DMSTrack (ours)	**43.52**	57.94	**91.50**	88.32	**68.35**	**13.19**	0.0073

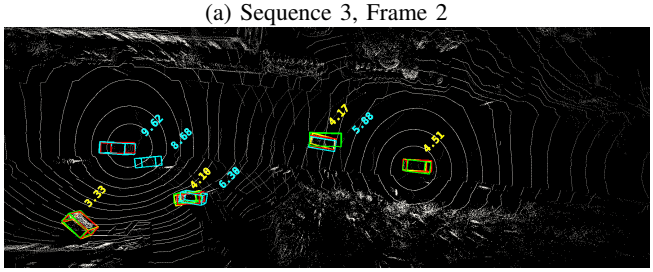
D. Qualitative Results

Figure 4 shows our qualitative results of a testing sequence's 3 sample frames. Green and Red 3D bounding boxes are the ground-truth and the final tracking result. Yellow and Cyan boxes are the individual detections from CAV1 and CAV2 respectively. CAV1 and CAV2 are located at the left and right centers of the two circular point cloud patterns respectively. The numbers represent the average of diagonal elements of our model's estimated observation noise covariance matrix $\mathbf{R}$ of each detection.

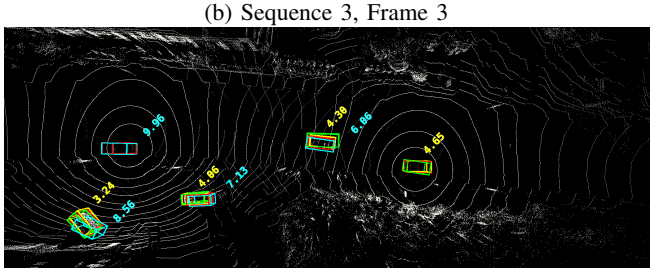
For the object in the bottom left corner of Figure 4a and 4c, we can see that CAV1's detection (Yellow) is closer to the ground-truth (Green), compared with CAV2's detection (Cyan). The CAV2's lower detection accuracy on that object is potentially due to partial occlusion and a longer distance from the object to CAV2's sensor. Our covariance neural network is able to learn a larger observation noise covariance for CAV2 on that detection than CAV1 given the positional features and the local BEV features. And our differentiable multi-sensor Kalman Filter-based tracking algorithm is able to generate the final tracking result box (Red) closer to CAV1's detection (Yellow) and the ground-truth (Green). Our full-length tracking result videos are available at this URL.



(a) Sequence 3, Frame 2



(b) Sequence 3, Frame 3



(c) Sequence 3, Frame 4

Fig. 4: Qualitative tracking results of our algorithm.

V. Conclusion

We propose a novel probabilistic 3D multi-object cooperative tracking algorithm via a differentiable multi-sensor Kalman Filter. For each detection from each CAV, our model learns to use local BEV features and positional features to estimate observation noise covariance. The estimated uncertainty can be better used to exploit the theoretical property of Kalman Filter-based tracking methods. Our cooperative tracking algorithm achieves better tracking accuracy with lower communication cost in comparison to the state-of-the-art method in V2V4Real [1]. For future work, we would like to explore more advanced tracking algorithms and distributed training approaches: training separate covariance nets independently and chaining them together at tracking inference.

REFERENCES

[1] R. Xu, X. Xia, J. Li, H. Li, S. Zhang, Z. Tu, Z. Meng, H. Xiang, X. Dong, R. Song, H. Yu, B. Zhou, and J. Ma, "V2v4real: A real-world large-scale dataset for vehicle-to-vehicle cooperative perception," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[2] Y. Li, D. Ma, Z. An, Z. Wang, Y. Zhong, S. Chen, and C. Feng, "V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 914–10 921, 2022.

[3] H. Yu, W. Yang, H. Ruan, Z. Yang, Y. Tang, X. Gao, X. Hao, Y. Shi, Y. Pan, N. Sun, J. Song, J. Yuan, P. Luo, and Z. Nie, "V2x-seq: A large-scale sequential dataset for vehicle-infrastructure cooperative perception and forecasting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[4] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. Association for Computing Machinery, 2019, p. 88–100.

[5] R. Xu, H. Xiang, X. Xia, X. Han, J. Li, and J. Ma, "Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[6] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, J. Tu, and R. Urtasun, "V2vnet: Vehicle-to-vehicle communication for joint perception and prediction," in *European Conference on Computer Vision (ECCV)*, 2020.

[7] R. Xu, H. Xiang, Z. Tu, X. Xia, M.-H. Yang, and J. Ma, "V2x-vit: Vehicle-to-everything cooperative perception with vision transformer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[8] R. Xu, Z. Tu, H. Xiang, W. Shao, B. Zhou, and J. Ma, "Cobevt: Cooperative bird's eye view semantic segmentation with sparse transformers," in *Conference on Robot Learning (CoRL)*, 2022.

[9] Y. Li, S. Ren, P. Wu, S. Chen, C. Feng, and W. Zhang, "Learning distilled collaboration graph for multi-agent perception," in *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[10] Y.-C. Liu, J. Tian, N. Glaser, and Z. Kira, "When2com: Multi-agent perception via communication graph grouping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Y.-C. Liu, J. Tian, C.-Y. Ma, N. Glaser, C.-W. Kuo, and Z. Kira, "Who2com: Collaborative perception via learnable handshake communication," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6876–6883.

[12] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, 1960.

[13] D. Willner, C. B. Chang, and K. P. Dunn, "Kalman filter algorithms for a multi-sensor system," in *1976 IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes*, 1976, pp. 570–574.

[14] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, 2016, p. 4383–4391.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[17] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[18] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 9710–9719.

[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[21] X. Weng, J. Wang, D. Held, and K. Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[22] X. Weng, Y. Wang, Y. Man, and K. Kitani, "Gnn3dmot: Graph neural network for 3d multi-object tracking with multi-feature learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[23] H.-k. Chiu, A. Prioletti, J. Li, and J. Bohg, "Probabilistic 3d multi-object tracking for autonomous driving," *arXiv preprint arXiv:2001.05673*, 2020.

[24] H.-k. Chiu, J. Li, R. Ambrus, and J. Bohg, "Probabilistic 3d multi-modal, multi-object tracking for autonomous driving," *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[25] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[26] M. A. Lee, B. Yi, R. Martín-Martín, S. Savarese, and J. Bohg, "Multimodal sensor fusion with differentiable filters," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 444–10 451.

[27] A. Kloss, G. Martius, and J. Bohg, "How to train your differentiable filter," in *Autonomous Robots*, vol. 45, 2021, pp. 561–578.

[28] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015.