A System Implementation and Evaluation of a Cooperative Fusion and Tracking Algorithm based on a Gaussian Mixture PHD Filter

Milos Vasic, David Mansolino, and Alcherio Martinoli

***Abstract*— This paper focuses on a real system implementation, analysis, and evaluation of a cooperative sensor fusion algorithm based on a Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter, using simulated and real vehicles endowed with automotive-grade sensors. We have extended our previously presented cooperative sensor fusion algorithm with a fusion weight optimization method and implemented it on a vehicle that we denote as the *ego* vehicle. The algorithm fuses information obtained from one or more vehicles located within a certain range (that we call *cooperative*), which are running a multi-object tracking PHD filter, and which are sharing their object estimates. The algorithm is evaluated on two Citroën C-ZERO prototype vehicles equipped with Mobileye cameras for object tracking and lidar sensors from which the ground truth positions of the tracked objects are extracted. Moreover, the algorithm is evaluated in simulation using simulated C-ZERO vehicles and simulated Mobileye cameras. The ground truth positions of tracked objects are in this case provided by the simulator. Multiple experimental runs are conducted in both simulated and real-world conditions in which a few legacy vehicles were tracked. Results show that the cooperative fusion algorithm allows for extending the sensing field of view, while keeping the tracking accuracy and errors similar to the case in which the vehicles act alone.**

I. Introduction

The ability to accurately perceive the environment is an important factor contributing to the safety on roads. Situations with limited visibility are common, and it is therefore important to develop methods that can assist the driver by tackling limited sensing range and Field Of View (FOV), sensing reliability, and occlusion problems. One such example of limited visibility is an overtaking scenario, in which the vehicle to be overtaken occludes part of the visibility of the vehicle performing the overtaking maneuver. Accidents while overtaking accounted for 1% of all single- and two-vehicle crashes in the United States in 2013, whereas they were the cause of 1.8% of accidents with fatal consequences [1].

In these situations, information coming from other vehicles or the road infrastructure can substantially improve the decision-making process of the intelligent vehicle. Information sharing is usually achieved using wireless communication links. Communication can be considered as a type of virtual sensor, as suggested in [2]. For example, it can enable vehicles to see behind corners, as the wireless communication does not have the same line-of-sight constraints as most conventional sensors do. Cooperative fusion considers sharing of tracks or perception data (raw or processed) among two or more vehicles. Two major problems related to cooperative fusion are temporal and spatial alignment [3]. Temporal alignment deals with variable delays of wireless communication links. Ideally, vehicle sending observations should help to predict the evolution of the objects' state between the original measurement time and that of reception. Moreover, spatial alignment considers differences in the coordinate systems of the sending and receiving vehicles. The difficulty here arises from the uncertainty of sending and receiving vehicle locations.

Before a vehicle can share its tracks, it needs a filter capable of tracking multiple objects. One possible solution for multi-object tracking is based on Random Finite Set (RFS) models, in which a set of objects of variable cardinality is modeled as a random finite set. Filters based on this theory, such as Probability Hypothesis Density (PHD) filters and Cardinalized PHD (CPHD) filters, deal with the measurement-to-track association implicitly [4]. They can provide higher robustness and accuracy in scenarios where the number of objects is variable and/or not known in advance. Many different implementations of the PHD filter have been developed and used for multi-object tracking; the most common ones are the Gaussian Mixture PHD (GM-PHD) [5] and the Sequential Monte Carlo PHD (SMC-PHD) filter [6]. Given that in cooperative fusion, information needs to be exchanged between vehicles, and that the GM implementation requires less memory for storing the location hypotheses and thus less communication bandwidth than its SMC counterpart, we find approaches based solely on GM better suited for the problem at hand.

When using a PHD filter for tracking, communicating tracks is reduced to communicating the PHD intensities. A method for fusing PHD intensity functions is given in [7]. A distributed fusion of SMC-CPHD filters via exponential mixture densities (EMD) has been presented by Uney et al. in [8]. Battistelli et al. used EMD for distributed fusion of GM-CPHD densities [9]. Common for all three works is the assumption that all PHD filters work in the same domain, i.e., that all agents have entirely overlapping FOV in which objects are sensed. This is a very limiting assumption for the application of the aforementioned methods in the field of (moving) vehicles. In our previous work [10], we proposed a Cooperative GM-PHD (C-GM-PHD) filter, which relaxes this assumption. We validated our cooperative filter in a

The authors are with the Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. firstname.lastname@epfl.ch

This work is financially supported by PSA Groupe and has benefited of the technical and administrative support of the EPFL's Transportation Center. The authors would like to thank to Petar Pjanic from LSP-EPFL for his assistance with data collection.

high-fidelity simulation environment using lidar sensors.

In the core example of the present work we consider two intelligent vehicles, each having a forward-looking vision sensor. The first vehicle runs a tracking algorithm based on the GM-PHD filter, and communicates its PHD intensity to the second vehicle (the *ego* vehicle). With the help of the C-GM-PHD filter, the second vehicle fuses the received intensity with the results of tracking based on its own sensor.

The main contribution of the paper is the real system implementation and validation of our cooperative sensor fusion algorithm in real-world settings and in simulation using automotive-grade sensors. We extend our cooperative algorithm with a fusion weight optimization method, which allows for a more robust algorithm performance. Moreover, instead of lidars, as proposed in our previous work leveraging exclusively high-fidelity simulations, in this paper we use a Mobileye camera as a tracking sensor. It is an automotive-grade monocular vision sensor with embedded image processing and object detection algorithm, which successfully detects vehicles, pedestrians, traffic signs, etc. Instead of with an image, a user is directly provided with the description of detected objects. The Mobileye camera is not very well studied in the literature. One notable exception is the work of Obst et al. [11], who used the Mobileye camera in their sensor fusion algorithm but they came up with different camera parameters and used a measurement model that resembled to a radar sensor, a choice that better suited their fusion framework.

The rest of the paper is organized as follows: Sec. II summarizes the multi-object tracking and cooperative fusion algorithms; Sec. III provides details of applying the methodology to our specific problem; Sec. IV presents and discusses the results, while Sec. V gives concluding remarks and an outlook.

II. Method

The cooperative sensor fusion algorithm for tracking of multiple objects contains two main components. The first component is in charge of multi-object tracking. It models object hypotheses set as a GM. The second component enables fusion of two object sets together. The fused set is further used by the tracking component. The two components together form the C-GM-PHD filter [10].

In the remainder of this section, both components are explained in greater details. It is worth noting that the whole algorithm is independent of any specific sensor or state model. In Sec. III we provide details how we apply the algorithm to our specific problem.

A. Multiple object tracking

In this work, for multi-object tracking we use the GM-PHD filter [5]. We summarize the algorithm here for completeness.

In the GM-PHD filter, the multiple-object posterior density is approximated using its first-order statistical moment called *PHD* or *intensity*, which is modeled as a Gaussian Mixture. At time $k-1$ the intensity containing J_{k-1} components with weights w_{k-1}, means m_{k-1} and covariances P_{k-1}, where the weight of a Gaussian component represents the number of objects that are represented using that component, has the form:

$$v_{k-1}(\mathbf{x}) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(\mathbf{x}; m_{k-1}^{(i)}, P_{k-1}^{(i)}) \tag{1}$$

The filter contains the predict and the update step, in which we use a Kalman Filter (KF). To obtain the predicted intensity, the prior is multiplied by the probability of survival p_S, which is a function of the hypothesis state, and the birth intensity γ_k is added. It is given by:

$$v_{k|k-1}(\mathbf{x}) = \sum_{i=1}^{J_{k-1}} p_{S,k}(m_{k|k-1}^{(i)}) w_{k-1}^{(i)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) + \gamma_k(\mathbf{x}) \tag{2}$$

The birth intensity $\gamma_k(\mathbf{x})$ is also a Gaussian Mixture, and represents locations in state space where new objects are likely to appear.

The update step utilizes the set of measurements Z_k and yields a posterior intensity as follows:

$$v_k(\mathbf{x}) = \sum_{i=1}^{J_{k-1}} [1 - p_{D,k}(m_{k|k-1}^{(i)})] w_{k|k-1}^{(i)} \mathcal{N}(\mathbf{x}; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) + \sum_{\mathbf{z} \in Z_k} \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(\mathbf{z}) \mathcal{N}(\mathbf{x}; m_{k|k}^{(j)}(z), P_{k|k}^{(j)}) \tag{3}$$

where

$$w_k^{(j)}(\mathbf{z}) = \frac{p_{D,k}(m_{k|k-1}^{(j)}) w_{k|k-1}^{(j)} q_k^{(j)}(\mathbf{z})}{\kappa_k(\mathbf{z}) + \sum_{l=1}^{J_{k|k-1}} p_{D,k}(m_{k|k-1}^{(l)}) w_{k|k-1}^{(l)} q_k^{(l)}(\mathbf{z})} \tag{4}$$

$$q_k^{(j)}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; H_k^{(j)} m_{k|k-1}^{(j)}, R_k + H_k^{(j)} P_{k|k-1}^{(j)} [H_k^{(j)}]^\top) \tag{5}$$

$$m_{k|k}^{(j)}(\mathbf{z}) = m_{k|k-1}^{(j)} + K_k^{(j)}(\mathbf{z} - H_k m_{k|k-1}^{(j)}) \tag{6}$$

$$P_{k|k}^{(j)} = [I - K_k^{(j)} H_k] P_{k|k-1}^{(j)} \tag{7}$$

$$K_k^{(j)} = P_{k|k-1}^{(j)} H_k^\top (H_k P_{k|k-1}^{(j)} H_k^\top + R_k)^{-1} \tag{8}$$

The parameters used in the update step are the clutter level $\kappa_k(\mathbf{z})$, the probability of detection $p_{D,k}(m_{k|k-1}^{(j)})$ dependent on the mean of the Gaussian component j and computed by the occlusion model (as an object can be occluded or leave the sensor FOV), the observation model H_k and the observation noise covariance R_k. The first sum in (3) represents missed objects, and the second updated objects.

After the update step, the number of Gaussian components increases quadratically. Therefore, all components with a weight less than some threshold T_p are removed. Moreover, components that are close to each other (within Mahalanobis distance U) are merged together and approximated by a single component. Finally, multi-object state estimates are extracted from the intensity by selecting the means of the Gaussians that have weights greater than some threshold T_e.

B. Cooperative fusion

The fusion algorithm is explained in greater details in our previous work [10]. Vehicles located within the communication range of the ego vehicle can share their PHD intensities, i.e., their estimates about objects in their FOV, which have the GM form. In the next paragraph, we explain how the ego vehicle can fuse received PHD intensities with its own intensity, hence increasing its FOV beyond the one of its sensors, and decreasing uncertainty in the areas of overlapping FOVs.

Before fusion, we need to translate the received intensities (states and associated covariances of the object hypotheses) from the frame of the cooperative vehicle, through the global frame, to the coordinate frame of the ego vehicle, and run the prediction model to compensate for the communication delay. In this transformation, the state uncertainties of the coordinate frames of the two cars are integrated in the tracks' covariances. To avoid data incest problems, we use a General Covariance Intersection (GCI) algorithm, which offers a conservative way of fusing two Gaussian Mixtures [12]. The GCI yields a fused intensity $v_{\mathscr{W}}(\mathbf{x})$ by weighting initial intensities using a fusion weight $\mathscr{W}$:

$$v_{\mathscr{W}}(\mathbf{x}) = \frac{v_1^{\mathscr{W}}(\mathbf{x}) v_2^{1-\mathscr{W}}(\mathbf{x})}{\int v_1^{\mathscr{W}}(\mathbf{y}) v_2^{1-\mathscr{W}}(\mathbf{y})\, \mathrm{d}\mathbf{y}} \tag{9}$$

It is shown in [9] that GCI can be approximated by applying Covariance Intersection (CI) pairwise to components from the two intensities. But this approach does not work if the vehicle FOVs do not overlap entirely, because an object seen by only one vehicle does not have its corresponding component in the set of another vehicle, so it likely gets discarded. To address this problem, we apply covariance intersection only to components whose Mahalanobis distance from each other is less than U_F. Take Gaussian components i and j from the intensities v_1 and v_2 respectively; the fused component has the following mean $m_{ij}^{(12)}$, covariance $P_{ij}^{(12)}$, and weight $w_{ij}^{(12)}$:

$$m_{ij}^{(12)} = P[\mathscr{W}(P_i^{(1)})^{-1} m_j^{(2)} + (1-\mathscr{W})(P_i^{(1)})^{-1} m_j^{(2)}] \tag{10}$$

$$P_{ij}^{(12)} = \left[\mathscr{W}(P_i^{(1)})^{-1} + (1-\mathscr{W})(P_j^{(2)})^{-1}\right]^{-1} \tag{11}$$

$$w_{ij}^{(12)} = \left(w_i^{(1)}\right)^{\mathscr{W}} \left(w_j^{(2)}\right)^{1-\mathscr{W}} \kappa(\mathscr{W}, P_i^{(1)}) \kappa(1-\mathscr{W}, P_j^{(2)}) \mathcal{N}\left(m_i^{(1)} - m_j^{(2)}; 0, \frac{P_i^{(1)}}{\mathscr{W}} + \frac{P_j^{(2)}}{1-\mathscr{W}}\right) \tag{12}$$

where

$$\kappa(\mathscr{W}, P) = \frac{[\det(2\pi P \mathscr{W}^{-1})]^{\frac{1}{2}}}{[\det(2\pi P)]^{\frac{\mathscr{W}}{2}}} \tag{13}$$

We keep track of components that have been fused, and we copy the non-fused components to the result set (e.g., objects which are located in FOV of only one vehicle).

As an extension to the fusion algorithm we presented in [10] (where we fixed $\mathscr{W} = 0.5$), in this paper we perform optimization of the fusion weight. Let $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ be the Gaussian mixtures containing selected Gaussian components from v_1 and v_2 which participate in fusion, and $f_{\mathscr{W}}$ the Gaussian mixture that is created by fusing f_1 and f_2 . Further suppose that $J(\mathscr{W})$ is a cost function. Then the goal is to choose $\mathscr{W}^*$ such that

$$\mathscr{W}^* = \arg \min_{\mathscr{W} \in [0,1]} J(\mathscr{W}) \tag{14}$$

The cost function can be defined as

$$J(\mathscr{W}) = \left(D(f_{\mathscr{W}} || f_i) - D(f_{\mathscr{W}} || f_j)\right)^2 \tag{15}$$

The Kullback-Leibler and Renyi divergence have been used as distance metrics in the literature [8], [13]. However, there exists no closed-form solution for these metrics applied to Gaussian mixtures. We therefore use the L_2 distance [14]:

$$D(f||g) = d(f,g) = \int \left(f(\mathbf{x}) - g(\mathbf{x})\right)^2 \mathrm{d}\mathbf{x} \tag{16}$$

for which the closed-form solution is easily derived by applying the following equation:

$$\int \mathcal{N}(\mathbf{x}|\mu_1, \sigma_1)\mathcal{N}(\mathbf{x}|\mu_2, \sigma_2) d\mathbf{x} = \mathcal{N}(\mathbf{0}|\mu_1 - \mu_2, \sigma_1 + \sigma_2) \tag{17}$$

We solve (14) by performing an exhaustive search for different values of $\mathscr{W}$ equidistantly spaced by 0.1.

III. Implementation

The framework presented in the previous section is general enough to be applied to any sensor and any state model. In our previous work [10], we have demonstrated its application to car tracking using lidars. The approach has been experimentally validated in simulation, and simulated data were processed in Matlab.

In this work we use two scenarios, the first containing real-world and the second containing simulated data. We use a different sensing modality for tracking, namely Mobileye cameras instead of lidar sensors. Moreover, in the real-world scenario, we extract ground truth information from lidars as they represent an independent and more accurate source of information than Mobileye cameras. This section describes our experimental setup and models we use to apply the cooperative filter in the real-world and in the simulated scenario. The presented method is demonstrated using one ego and one cooperative vehicle.

A. Real-world experimental setup

Our experimental platform consists of two electric Citroën C-ZERO vehicles, that are parked behind each other in one lane, at the relative distance of approximately 15 m. This resembles a scenario in which two cars are driving behind each other at the same speed. The rear vehicle, which we call the ego vehicle ($\mathcal{E}$), is equipped with a forward looking Mobileye camera, and two forward- and two backward-looking Ibeo lidar sensors (that we use for ground truth only and not for tracking). The front vehicle, which we refer to as the cooperative vehicle ($\mathcal{C}$), is endowed as well with a forward-looking Mobileye camera (see Fig. 1). Both vehicles have on-board computers for real-time sensor processing and logging, as well as wireless communication

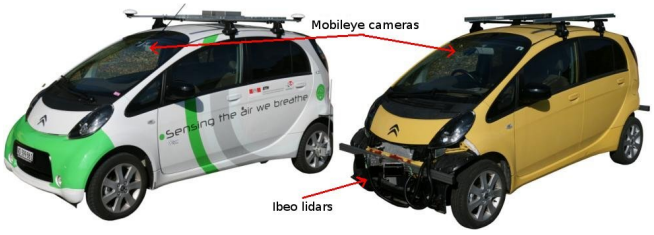

Fig. 1. Prototype Citroën C-ZERO vehicles.

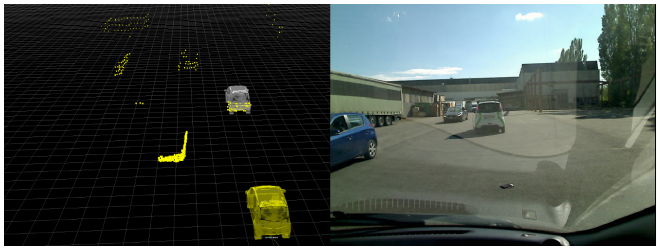

Fig. 2. Experimental setup. Left: ego vehicle (yellow), cooperative vehicle (white), and lidar point cloud of the surrounding of the ego vehicle (legacy vehicles are easily identifiable). Right: View from the web camera installed behind the windshield of the ego vehicle.

equipment. Our scenario includes two more legacy oncoming vehicles ($\mathcal{O}_1$ and $\mathcal{O}_2$) of different type and size, which do not carry neither sensory nor computational equipment. The experimental setup is visualized in Fig. 2.

The entire software pipeline is implemented on in-vehicle computers in the ROS framework [15] using C++. Sensor driver nodes publish data from the sensors to the ROS network. Tracking filter nodes subscribe to sensor data and publish tracks in the form of PHD intensities. A cooperative fusion and tracking node (only present in the ego vehicle) additionally subscribes to the tracks and publishes a fused PHD intensity. Our software architecture is illustrated in Fig. 3. All nodes can be run online (in real-time) during experiments. However, as the emphasis of our work is not on the wireless communication technology, we can afford to log all data using *rosbag* and play it back for processing offline. To avoid time misalignment of vehicle logs, the computer times are synchronized among themselves using a network time synchronization tool before and throughout the experiments.

B. Simulation setup

Our simulated setup is implemented in Webots, a high-fidelity calibrated sub-microscopic simulator for mobile robots and intelligent vehicles [16]. Similar to the real-world setup, we have placed two Citroën C-ZERO vehicles equipped with sensors driving behind each other on a straight road, as well as two legacy vehicles driving in the opposite direction (see Fig. 4).

As a model of the Mobileye camera was not readily available in Webots, we have created it. We analyzed real data in order to achieve results faithful to reality. In addition to the emulated Mobileye, we use simulated GPS and IMU sensors for accurate localization of the sensing cars. A

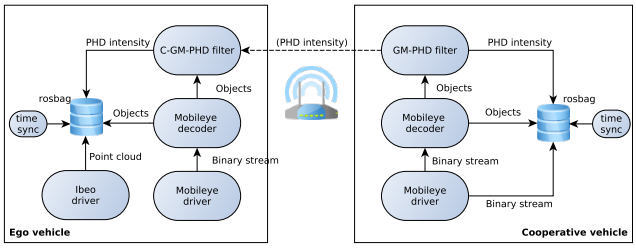

Fig. 3. Software architecture in the ROS framework deployed on the two experimental vehicles' computers.

Fig. 4. Setup in the Webots simulator. The ego car (yellow) drives on a straight road behind the cooperative car (white), while two non-cooperative cars drive in the opposite direction.

supervisor controller is in charge of providing ground truth data.

A software architecture very similar to the one shown in Fig. 3 is used. Webots publishes data from sensors as ROS topics. These topics correspond in type to the ones published by real sensors. Therefore exactly the same implementation of the cooperative tracking and fusion algorithm can be used. One notable difference is that the whole software (deployed in reality on two cars) runs in simulation on a single computer, which simplifies time synchronization between the two cooperating vehicles.

C. State and motion model

We model a car as a rectangle with constant width and length (most cars are relatively similar in size and this model fits them well). The state model suitable for car tracking uses car's position (x, y), velocity (v_x, v_y), orientation θ and turn rate ω:

$$\mathbf{x} = [x, y, v_x, v_y, \theta, \omega]^\top \tag{18}$$

The motion model predicts the evolution of the state through time. The model of our choice is the constant velocity and constant turn-rate, in which the input noise exists on linear and rotational acceleration a and α.

D. Camera model

For this paper, we use Mobileye cameras. They are monocular systems with integrated algorithms for image processing and object recognition. Limited details about the camera system can be found in [17]. After processing each frame using the software embedded in its chip, the camera sends the data describing objects over the CAN bus in multiple consecutive binary encoded messages. We synchronize all

messages related to a single image frame and decode them to create a set Z of measurements $\mathbf{z}$ that relate to separate objects present in that frame:

$$Z = \cup_i \mathbf{z_i} \tag{19}$$

where

$$\mathbf{z} = [x, y, \theta]^\top \tag{20}$$

is a measurement of a single object, and describes its center point and orientation. The Mobileye camera can only detect objects for which it was trained (such as, among others, cars, trucks, and pedestrians). Other classes of objects or the raw camera image are not available. It is important to mention that our version of Mobileye camera only detects cars from the front or from the back (cars facing sideways or under a large angle are not detected). This greatly limits the choice of a scenario. Missed detections and clutter are unavoidable, and they are handled by the tracking algorithm.

In the absence of detailed camera specifications, the important tracking-related camera parameters have been determined experimentally. We compared the output of the camera to a point cloud obtained by a lidar, and determined the useful camera range to be 50 m. Objects beyond 50 m are only sporadically detectable. We also found out that the accuracy of estimated object distance decreases with increasing range, and it is relatively reliable within 50 m from the camera. As we will report below, the choice of the camera range was further motivated by the available ground truth data. The minimum distance at which the camera detects objects is around 15 m. The camera FOV is stated to be 47 deg in [17]. After initial tests, however, we decided to use a symmetrical FOV of 45 deg (from -22.5 to 22.5 deg).

During our experiments we noticed a significant bias between the camera measurements and the ground truth. Although the two cameras we used are identical, they were initially calibrated on vehicles using the automatic calibration procedures, thus their calibration parameters could be slightly different. We have therefore re-calibrated the output of the two cameras for our scenario with respect to the lidar ground truth data. We have determined that a second-order polynomial function provides sufficiently good fit for the longitudinal distance, while for the lateral distance and the orientation only an offset is applied. In order to calibrate the camera on the C-ZERO which has no lidars, its pose relative to the other C-ZERO had to be calibrated first.

Observations collected during camera calibration are used for emulating the sensor in Webots. We noticed that the noise on longitudinal distance is not normally distributed. Instead, for a stationary object, the camera return is typically constant, but it includes a bias. The bias is nevertheless different for different runs, which makes its complete removal through calibration impossible. For our emulated camera we use a linear noise model dependent on the longitudinal distance, the slope of which changes for every simulated camera in every run: the maximal bias e_{max} is sampled at random from a normal distribution with $\mu = 0$ and $\sigma = 1$, and the added noise is in the range $\pm e_{\text{max}}$ from the ground truth value. As a simplification for our filter, however, we assume that the noise is Gaussian, which allows us to use a KF for the state update. A normally distributed noise is added to the y and θ camera returns, with $\sigma_y = 0.1$ m and $\sigma_\theta = 0.01$ rad.

E. Localization

Although localization is not the focus of this paper, it is of great importance for the functioning of the cooperative fusion algorithm to have an accurate relative pose of the cooperative vehicles with respect to the ego vehicle. In the absence of the centimeter-level GNSS devices installed on both experimental vehicles, we tune the pose manually, by using the lidar sensors mounted at the front of our ego vehicle, and matching its point cloud to the CAD model of the Citroën C-ZERO. For this reason, we are forced to keep the two sensing cars stationary in our real-world experiments. In other words, we consider the accurate localization to be a separate problem and assume that it is available. The pose of the center of vehicle $\mathcal{C}$ in the local frame of vehicle $\mathcal{E}$ is determined to be $(x, y, \theta) = (15.45 \text{ m}, -0.25 \text{ m}, -0.0175 \text{ rad})$.

In our simulated setup we have accurate position and orientation sensors. Thus we can afford to move the sensing cars.

F. Ground truth for the real-world setup

In our real-world experimental setup, we do not have accurate localization equipment which could provide ground truth for the tracked vehicles in our scenario. Therefore, we use lidar data to extract the ground truth, taking profit from the fact that a lidar point is accurate to 0.1 m (one σ) and that the experimentally determined accuracy of the Mobileye camera is worse than that.

We extract the ground truth positions of the legacy vehicles from the lidar point cloud. For each experimental run, we determine the contours of the legacy vehicles by visualizing the point cloud in the ROS tool *rviz*. Each ground truth set contains at least 10 manually extracted sample points, and vehicle positions in-between these sample points are linearly interpolated.

The maximal distance of a detectable object in our setting is around 70 m (objects at larger distances were occluded). This is another reason to limit the range of the Mobileye camera to 50 m, as the maximum range of the cooperative system amounts to around 65 m (given the distance between vehicles $\mathcal{E}$ and $\mathcal{C}$).

G. Parameters

The tracking parameters were the same in all runs. The maximum number of Gaussian components per filter J_{max} is limited to 100. The birth model for the GM-PHD filter consists of only one Gaussian component. Its position is in the center of the Mobileye camera's FOV, the heading is set to π rad, whereas the birth velocity and the turn rate are set to zero. The Standard Deviation (SD) of the birth component is 25 m (half of the sensor range) for its position, π rad for the heading, 10 $^{\text{m}}/_{\text{s}}$ for the speed, and 0.2 $^{\text{rad}}/_{\text{s}}$ for the turn rate. The rationale behind these values is that

a new object entering the sensing FOV is very likely to be contained within one SD from the birth component mean.

The SD for the noise in the motion model is set to $1\ {}^{\mathrm{m}}/_{\mathrm{s}^2}$ for the linear acceleration and $0.1\ {}^{\mathrm{rad}}/_{\mathrm{s}^2}$ for the rotational acceleration. The measurement noise is determined empirically and is modeled as a Gaussian with SD $\sigma_x = 0.5$ m, $\sigma_y = 0.3$ m, and $\sigma_\theta = \frac{\pi}{16}$ rad. The clutter model uses a Poisson distribution with an expected cardinality of 1 measurement per sensor return per surveillance area. The $\mathcal{E}$ and $\mathcal{C}$ coordinate frames' uncertainty ($1\,\sigma$) is set to $[0.5\ \mathrm{m}, 0.3\ \mathrm{m}, 0.1\ {}^{\mathrm{m}}/_{\mathrm{s}}, 0.1\ {}^{\mathrm{m}}/_{\mathrm{s}}, 0.0174\ \mathrm{rad}, 0.0174\ {}^{\mathrm{rad}}/_{\mathrm{s}}]^\top$.

The GM-PHD merging parameter is set to $U = 10$, and a Gaussian component is pruned if its weight is less than $T_p = 10^{-5}$. The extraction threshold T_e is set to 0.5. The fusion distance parameter U_F is empirically set to 30. The probability of survival in the joint FOV is set to 0.99. The maximal probability of detection is $p_D^{max} = 0.9$. The shape of the object (tracked vehicles) is assumed to be rectangular, with length of 3.5 m and width of 1.5 m.

IV. Experimental Results

The experimental evaluation of the cooperative fusion and tracking algorithm is performed on a real-world dataset consisting of 15 distinctive runs. In each of the runs, cars $\mathcal{E}$ and $\mathcal{C}$ were positioned at the same spot, while cars $\mathcal{O}_1$ and $\mathcal{O}_2$ were driven manually in the opposite direction at the speed of approximately 20 ${}^{\mathrm{km}}/_{\mathrm{h}}$. In addition, a similar scenario is designed in simulation, which demonstrates the performance of the cooperative fusion when the cars $\mathcal{E}$ and $\mathcal{C}$ move at 30 and 27 ${}^{\mathrm{km}}/_{\mathrm{h}}$, respectively. The two legacy cars in simulation move at respective speeds of 21 and 26 ${}^{\mathrm{km}}/_{\mathrm{h}}$. For the sake of comparison, for each run both C-GM-PHD (includes cooperative fusion) and regular GM-PHD filter (does not include cooperative fusion) were run on the ego vehicle[1].

Fig. 5 shows a trajectory of the car $\mathcal{O}_1$ obtained during the first run. When $\mathcal{O}_1$ is only observed by the car $\mathcal{C}$, the estimation uncertainty is large as the uncertainty of the relative localization between cars $\mathcal{E}$ and $\mathcal{C}$ needs to be integrated into the track estimation uncertainty. Once the $\mathcal{O}_1$ enters the FOV of the car $\mathcal{E}$, the uncertainty reduces progressively. Around $x = 30$ m we observe that, as expected, the track estimate gets closer to the $\mathcal{E}$ measurements, since its uncertainty is significantly smaller than the uncertainty of the track received from the car $\mathcal{C}$ (it does not contain the localization uncertainty). In the case when no cooperative fusion is used (Fig. 5 bottom), the track uncertainty only depends on the motion and sensor uncertainty, as the tracking is performed in the local frame of the car $\mathcal{E}$.

The rest of the results use the Optimal Sub-Pattern Assignment (OSPA) metric [18] for performance evaluation. OSPA is commonly used for evaluation of multi-object tracking performance. It consists of two components: the first component reflects the average error in position across all tracked objects, whereas the second component reflects a cardinality error (mismatch between the number of objects in the ground truth and in the estimated object set). We set the position sensitivity parameter $p = 1$, i.e., we compute the position error using the Euclidean distance between the center of the estimated and the ground truth object). The cut-off parameter for cardinality error penalties is set to $c = 10$.

[1]A video showing the experiments can be viewed at http://disal.epfl.ch/research/NetworkedIV

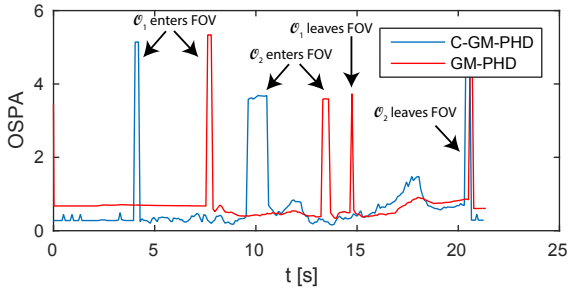

Fig. 6. OSPA over time for a single run using real-world data. Performance of the cooperative filter is compared against ground truth in the union of FOVs of the cars $\mathcal{E}$ and $\mathcal{C}$, while for non-cooperative filter only ground truth objects in the FOV of the car $\mathcal{E}$ are considered.

The OSPA performance for the first run is shown in Fig. 6. For the cooperative version of the filter, the ground truth objects are considered in the union of the two vehicles' FOVs. On the other hand, to enable a fair comparison, for the non-cooperative filter only ground truth objects in the FOV of the ego car are taken into account. At the beginning, only vehicle $\mathcal{C}$ is tracked, until the car $\mathcal{O}_1$ enters the FOV. Spikes indicate cardinality penalties. Wrong cardinality estimates can be due to delay in object birth or death, loosing track of objects for short periods of time or counting twice the same object for short amounts of time. In Fig. 6, all spikes represent birth/death delays. This leaves us with an impression that the performed fusion weight optimization allows for a more robust algorithm performance compared to [10], in a sense that the algorithm loses track of the objects less frequently.

Statistics of tracking error aggregated over 15 runs are shown using the box plot in Fig. 7. We can see that a median error of the cooperative filter is slightly lower than its non-cooperative counterpart. However, their performance is very similar. The absence of statistically significant improvement of the cooperative filter in the intersecting FOV can be attributed to the conservative nature of the GCI method. It is important to note that here comparison is performed only in the FOV of the car $\mathcal{E}$, as the non-cooperative filter cannot track objects outside of this FOV. Moreover, the localization uncertainty plays an important role in the accuracy of the cooperative tracking algorithm, as the uncertainty of $\mathcal{E}$ and $\mathcal{C}$ coordinate frames gets integrated in the track estimation uncertainty. In Fig. 5 we can see that it takes a significant amount of time for the uncertainty to be reduced when the cooperative fusion is employed. Consequently, it is not unexpected that the cooperative filter tracks objects outside of the ego car's FOV with less accuracy than the objects in

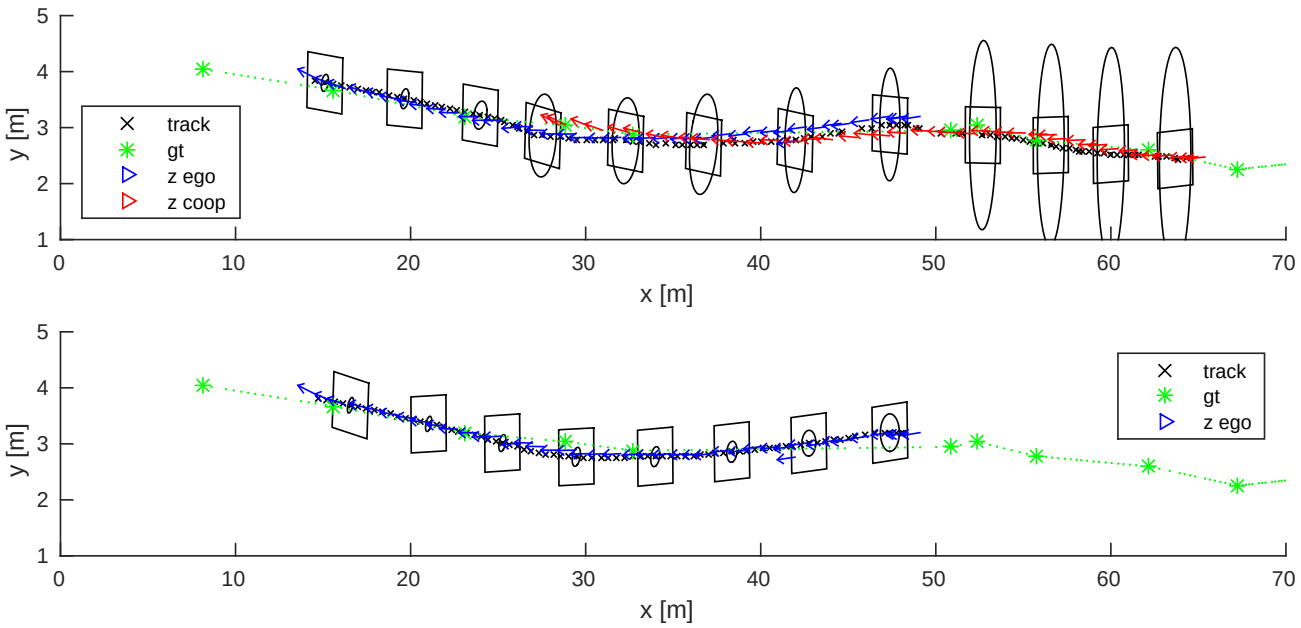

Fig. 5. Trajectory of the car $\mathcal{O}_1$ during the first run of the real-world experimental produced by the C-GM-PHD filter (top) and the GM-PHD filter (bottom). $\mathcal{O}_1$ starts from far away (70 m) and approaches the tracing vehicles. Black crosses indicate the estimated track positions. Vehicle rectangle including its orientation is showed every 10 filter update cycles (rectangles have been scaled down for clarity). Ellipses in black indicate one standard deviation for (x, y) uncertainty. Ground truth trajectory sample points are shown using green asterisks, and interpolated trajectory is shown in green dots. Ground truth does not contain orientation information. Blue and red arrows represent Mobileye measurements (position and orientation). Every second measurement is shown to avoid the image being very cluttered. Note the difference between x and y spatial scales.

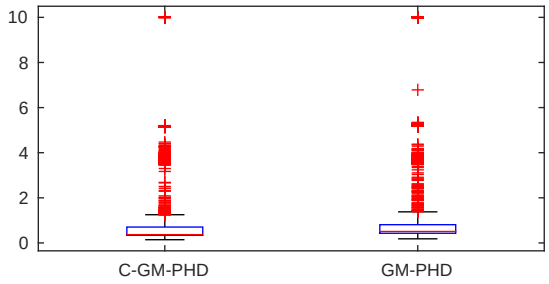

Fig. 7. Comparison of OSPA performance accumulated during 15 experiment runs between the cooperative and non-cooperative filter.

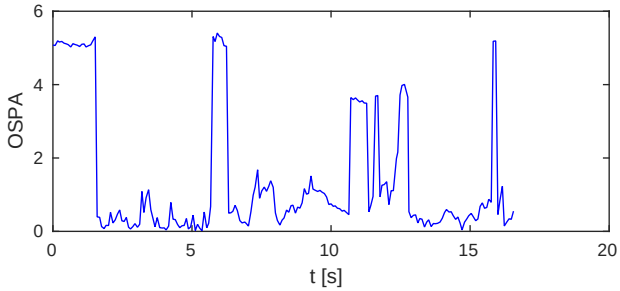

Fig. 8. OSPA over time for one run of the C-GM-PHD filter in the Webots simulator.

the ego FOV, thus comparing the accuracy of the two filters over the union of the FOVs would be unfair. We believe that cooperative localization techniques, such as map registration, can be beneficial, as they can significantly reduce the coordinate frame uncertainty. With smaller uncertainty of the received intensity, the error and uncertainty of the fused intensity should decrease.

So far, all results refer to the real-world experiments, in which the two sensing cars are static. Fig. 8 shows the OSPA metric for a simulated experiment. We remind the reader that, in this scenario, the sensing cars move at different speeds. We observe the similarity in tracking performance between the simulated and the real-world experiments. The OSPA error is slightly larger in the simulated experiment. The reason for this could lie in the uncertainty introduced by the motion of the sensing cars, or in the noise of the emulated Mobileye camera.

V. Conclusion and Outlook

In this paper, we presented a real system implementation of our Cooperative GM-PHD filter [10], which we extended with a fusion weight optimization routine. We evaluated it using real vehicles and automotive-grade sensors in real-world settings. The scenario included two stationary sensing vehicles (Citroën C-ZERO) called the cooperative and the ego vehicle, as well as two non-cooperative legacy vehicles. The cooperative vehicle ran the GM-PHD filter and it used the Mobileye camera for tracking. The ego vehicle also had a Mobileye camera, but it additionally fused the PHD intensity it received from the cooperative vehicle with its own PHD intensity (using the C-GM-PHD filter). Results were as well generated for the cases when the ego vehicle did not fuse the remote information. We compared the estimated trajectory of one legacy vehicle in absence and presence of

cooperation between the sensing vehicles. Results showed a clear added value of cooperation resulting in an increased range and FOV. We have further compared the performance obtained during 15 experimental runs to the ground truth data obtained from the accurate lidar sensor. The cooperative fusion algorithm was validated by comparing its performance with the non-cooperative tracking algorithm.

We reproduced the aforementioned scenario in simulation, with the difference that the two sensing vehicles were no longer stationary. Our algorithm showed similar performance, provided that the sensing vehicles are endowed with accurate localization systems.

As noted already, one of the biggest assumptions of this work is that accurate relative localization exists between the ego and the cooperative vehicle. In the absence of the centimeter-level localization equipment on both real vehicles, in this paper we had to solve the localization manually using the lidar point cloud. To overcome this problem, we plan to look into methods that can provide accurate relative positioning, such as feature-based localization and map matching. This would in turn allow us to carry out experiments in real-world scenarios where all vehicles move.

Moreover, in this paper a physical communication link between vehicles was only used to synchronize the time of in-vehicle computers. To make this aspect realistic, we intend to establish real-time sharing of PHD intensities between vehicles, execute the cooperative fusion algorithm online, and assess the tracking performance in scenarios where the communication lag varies with the distance between vehicles and amount of data communicated per unit of time.

REFERENCES

[1] National Highway Traffic Safety Administration (NTHSA), "2013 Traffic Safety Facts FARS/GES," Annual Report 812139, 2015.

[2] G. Thomaidis, K. Vassilis, P. Lytrivis, M. Tsogas, G. Karaseitanidis, and A. Amditis, "Target tracking and fusion in vehicular networks," in *IEEE Intelligent Vehicles Symposium*, 2011, pp. 1080–1085.

[3] A. Rauch, F. Klanner, R. Rasshofer, and K. Dietmayer, "Car2X-based perception in a high-level fusion architecture for cooperative perception systems," in *IEEE Intelligent Vehicles Symposium*, 2012, pp. 270–275.

[4] R. P. S. Mahler, *Statistical Multisource-Multitarget Information Fusion*. Norwood, MA, USA: Artech House, Inc., 2007.

[5] B.-N. Vo and W.-K. Ma, "The Gaussian mixture probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.

[6] B. Ristic, D. Clark, and B.-N. Vo, "Improved SMC implementation of the PHD filter," in *13th Conference on Information Fusion*, 2010. doi: 10.1109/ICIF.2010.5711922

[7] D. Clark, S. Julier, R. Mahler, and B. Ristic, "Robust multi-object sensor fusion with unknown correlations," in *Sensor Signal Processing for Defence*, Sep. 2010. doi: 10.1049/ic.2010.0233

[8] M. Uney, D. Clark, and S. Julier, "Distributed fusion of PHD filters via exponential mixture densities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 521–531, Jun. 2013.

[9] G. Battistelli, L. Chisci, C. Fantacci, A. Farina, and A. Graziano, "Consensus CPHD filter for distributed multitarget tracking," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 508–520, Jun. 2013.

[10] M. Vasic and A. Martinoli, "A collaborative sensor fusion algorithm for multi-object tracking using a Gaussian mixture Probability Hypothesis Density filter," in *IEEE Intelligent Transportation Systems Conference*, 2015, pp. 491–498.

[11] M. Obst, L. Hobert, and P. Reisdorf, "Multi-sensor data fusion for checking plausibility of V2V communications by vision-based multiple-object tracking," in *IEEE Vehicular Networking Conference*, 2014, pp. 143–150.

[12] R. P. S. Mahler, "Optimal/robust distributed data fusion: a unified approach," in *Signal Processing, Sensor Fusion, and Target Recognition IX*, vol. 4052, 2000, pp. 128–138.

[13] M. Hurley, "An information theoretic justification for covariance intersection and its generalization," in *Proceedings of the Fifth International Conference on Information Fusion*, vol. 1, 2002, pp. 505–511.

[14] B. Jian and B. Vemuri, "Robust point set registration using Gaussian Mixture Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, Aug. 2011.

[15] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[16] S. Gowal, Y. Zhang, and A. Martinoli, "A realistic simulator for the design and evaluation of intelligent vehicles," in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 1039–1044.

[17] E. Dagan, O. Mano, G. Stein, and A. Shashua, "Forward collision warning with a single camera," in *IEEE Intelligent Vehicles Symposium*, 2004, pp. 37–42.

[18] D. Schuhmacher, B.-T. Vo, and B.-N. Vo, "A consistent metric for performance evaluation of multi-object filters," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3447–3457, Aug. 2008.