Robust Multimodal and Multi-Object Tracking for Autonomous Driving Applications

Marc Perez[1,2] and Antonio Agudo[1]

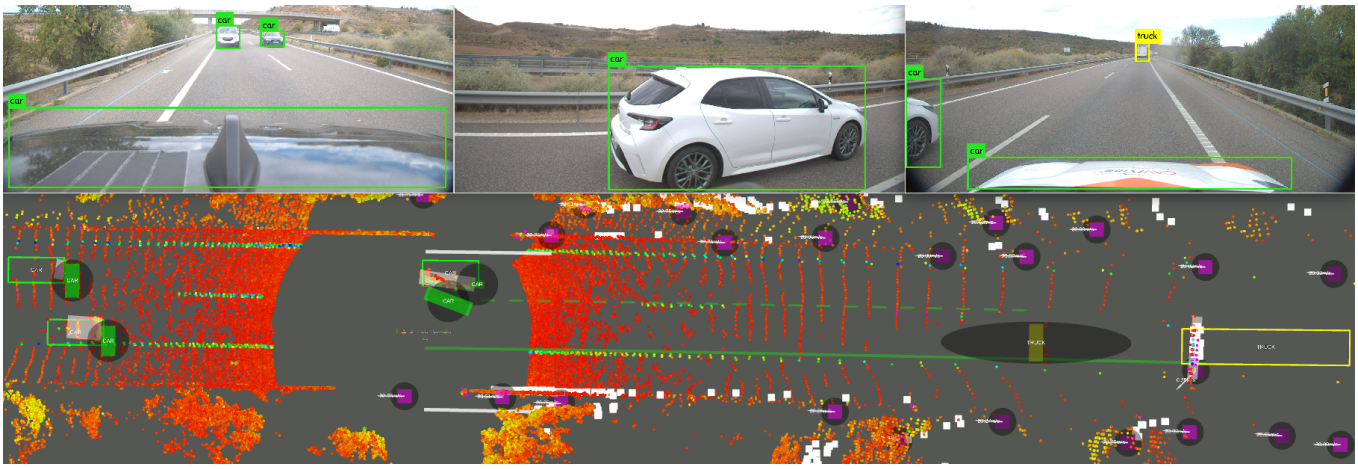


Fig. 1: **Robust multimodal and multi-object tracking.** The raw sensor data and associated detections in a highway scenario; the point cloud from a 360°-lidar is color-coded by intensity with most points in red or yellow. The lidar detections from our method are in grey, the radar is in purple, the camera detections are in yellow for trucks and green for cars, the low-resolution frontal lidar is in white, and the results of our MOT algorithm are the edges of the bounding boxes color-coded as the camera detections. The covariance of each detection is displayed as a shadow ellipse. For reference, the 2D detections overlayed on top of the images are shown on top and the lanes detected by a camera with integrated detections are shown in green.

***Abstract*—In this work, we present a method for Multi-Object Tracking (MOT) that uses unsynchronized multimodal detections from a configurable set of sensors such as cameras, radars and lidars. All the information is processed from the sensors with modality-specific detectors and then combined in the MOT module that incorporates a Kalman filter and tracklet management logic. To be able to deploy our system in real-world driving applications, we handle localization errors, misclassifications and partial bounding-box detections from the object detector in the MOT module. We show promising results and compare them with respect to competing approaches in two challenging real-world scenarios, including a traffic jam chauffeur as well as a traffic monitoring application on highways.**

***Index Terms*—Multiple Object Tracking, Sensor Fusion.**

I. INTRODUCTION

An accurate perception of the environment is essential for autonomous driving and Advanced Driver-Assistance Systems (ADAS), as the system needs to be aware of the trajectories of all the other actors in the scene in order to plan its own trajectory. To this end, Multi-Object Tracking (MOT) techniques can be used to combine detections from different sensors over time to estimate these trajectories. A common approach for real-time applications is tracking-by-detection [1]–[3], where a list of existing tracklets is kept, and every time that a new detection is received, it is associated with a tracklet in the list updating it. Detections that are not associated, are used to create new tracklets and are added to the list. Each tracklet represents an object with a state at each time instance. Usually, the state usually contains the position, velocity, orientation, and size of the object. Since common datasets do not include the acceleration, it is not computed in many methods in the literature, being only considered in a variant of CBMOT [2]. However, that information is necessary when deploying these systems in real-world applications, such as a traffic jam chauffeur (as demonstrated in experiment III-A), since the vehicle needs to be able to react to, e.g., a vehicle in front braking suddenly. A tracking-by-detection MOT method needs to solve the next main tasks: 1) Data association (how to decide to which tracklet each detection corresponds), 2) Filtering (how to update the tracklet given the associated detection), and 3) Tracklet management (how to create and delete tracklets). Next, we review previous work on each of these tasks.

1) Data association: Methods can be classified depending on whether they consider all detections from different time steps at once (offline tracking), or the information is

[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain. {mperez, aagudo}@iri.upc.edu
[2]Applus+ IDIADA, Tarragona, Spain.

processed as the data arrive (online tracking). Online tracking [1]–[3] is simpler in terms of data association and may be used in real-time applications, but offline tracking [4] can use future information to achieve more robust and accurate results. Detections and tracklets are associated based on a defined distance, such as the intersection over union, the Euclidean distance and the Mahalanobis one to account for the covariance of the estimations, or even learned distance functions based on data [5]–[7]. A common assumption for driving scenarios is that tracklets and detections are assigned one-to-one, creating a bipartite graph where the associations can be calculated using the Hungarian algorithm [8] or others [9]. In this way, we minimize the total distance of the associations. In some contexts, with many detections of low confidence, it is preferable to iterate through the detections sequentially by confidence and associate each of them to the closest tracklet [2], [3]. Detections can also be combined from different modalities before being associated with a particular tracklet [10]. Recent works have only explored association between objects of the same class, but we have observed that when applying learning-based methods to real-world data, some objects are misclassified. For example, when testing on highways with YOLO [11] trained on the COCO [12] dataset, as we will see in our experiments. We will present a method to overcome these issues and reduce the errors in MOT that misclassifications from the object detector cause.

2) Filtering: Each tracklet has a state. We need a method to predict how this state will evolve in the future, and a method to update the state taking into account the detections associated with it. The most common approach [1], [3], [6], [10] is to use a Bayesian filter for that, such as a Kalman filter (KF) [13] or its extended version [14]. A simpler approach [2], [15], when the detections include the velocity, is to calculate the position of the detection at the last time step, associate it to the tracklet based on the positions at that time, and then replace the state of the tracklet with that of the new detection.

Some sensors, such as radars, or lidar clustering algorithms [16], [17], produce detections with a partial bounding box, but this is not taken into account in previous works as they assume complete bounding boxes as input for the filter, propagating the error from the object detector to the MOT module. In this work, we propose a method to consider these partial detections and reconstruct the complete bounding box in the MOT module.

To improve the robustness of the module, [18] proposes an MOT framework with multiple sensors that can work if one sensor is disabled. Our method can also work when some of the sensors are disabled, but in our case, we try to handle the errors from the detections instead of disabling the sensors.

3) Tracklet management: This subsystem manages the creation, deletion, and output of tracklets. The basic solution is to create tracklets from unassociated detections but do not output them until they have had a minimum number of detections while deleting tracklets that have not been detected for a successive number of time steps [1]. But confidence-based management works better than count-based solutions when a proper function is used to update the confidence of the tracklet taking into account the confidence of the associated detection, as shown in [2].

The main contribution of our work is a detection and tracking pipeline that can work with a large set of sensors and detection methods and that generalizes well to different real-world scenarios, leveraging some novelties we include. First, we propose a simple method to estimate the location error of the depth estimation algorithm used by the camera and take it into account in the MOT module for association and filtering. In this module, we also recover from partial detections (when the detected bounding box is smaller than the real one) and misclassifications from the object detector. Second, we eliminate the need to synchronize sensors by asynchronously updating the same shared tracklet list. We provide experimental results on challenging real-world scenarios, including a traffic jam chauffeur and a traffic monitoring application, where we show the effectiveness of our sensor fusion solution. We compare our results with respect to CBMOT [2] to show how our approach improves the performance of MOT in the presence of noisy detections.

II. Sensor Fusion for Multi-Object Tracking

Our MOT pipeline works with a configurable set of sensors without the need to synchronize them. In this work, without loss of generality, we consider images coming from RGB cameras, 3D detections from radars, and point clouds from lidars. We first process the raw data from the sensors with class-specific object detectors. These 3D detections are then combined in the MOT module using a KF [13] and tracklet management logic, providing 3D tracklets with higher accuracy than the input detections. Our detection and tracking pipeline is displayed in Fig. 2. All the 3D detections, regardless of the modality of origin, are composed of a vector $\mathbf{d}$ and a timestamp. Particularly, $\mathbf{d} = [r_x, r_y, r_z, w, h, l, \phi, c, \beta, \sigma_{r_x,r_x}, \sigma_{r_x,r_y}, \sigma_{r_y,r_y}]^\top$, including the position of the centre of the object (r_x, r_y, r_z), the width w, height h, and length l of the bounding box, the yaw rotation ϕ, the type of object class c, the confidence score β of the detection model, and the 2D-position covariance values $(\sigma_{r_x,r_x}, \sigma_{r_x,r_y}, \sigma_{r_y,r_y})$. For some sensors such as radars, we also include the velocity $\mathbf{v} = [v_x, v_y]^\top$ as well as its covariance entries by the vector $\sigma_{\mathbf{v}} = [\sigma_{v_x,v_x}, \sigma_{v_x,v_y}, \sigma_{v_y,v_y}]^\top$.

We process the images using YOLO [11] trained on the COCO dataset [12] to obtain 2D bounding boxes, and then convert these to 3D detections by using [19] for depth estimation. Since the depth estimation is noisy, each of these 3D detections has a larger frontal positioning error that increases as the object gets further away. This error is captured in the covariance matrix, which the MOT then takes into account and reduces by exploiting other modalities. To obtain this covariance matrix, we consider a fixed pixel error e_p from the object detector and then calculate the difference in range $\Delta\lambda$ and azimuth angle $\Delta\alpha$ that we would obtain from [19] in $p + e_p$ and $p - e_p$, where p is the detected

pixel. Next, we obtain the covariance matrix in terms of $(\sigma_{r_x,r_x}, \sigma_{r_x,r_y}, \sigma_{r_y,r_y})$ by setting the matrix to $\begin{bmatrix} \Delta\lambda & 0 \\ 0 & \Delta\alpha \end{bmatrix}$ and then rotate it by the obtained azimuth α. We use $e_p = 2$ pixels for all our experiments.

To obtain 3D detections from point clouds, we filter the ground and apply a clustering algorithm to get class-agnostic detections with a partial bounding box, corresponding to the visible part of the object. The complete bounding box and class are covered in the MOT module. For lidars and radars, we set the covariance matrix according to sensor specifications.

These detections are taken as input to the sensor fusion module, and we output our estimation consisting in a list of I tracklets at every time step, where the i-th tracklet is coded by a vector $\mathbf{o}_i = [\mathbf{d}^\top, \mathbf{v}^\top, \mathbf{a}^\top, {\sigma_\mathbf{v}}^\top, {\sigma_\mathbf{a}}^\top]^\top$.

For completeness, we also consider the acceleration $\mathbf{a} = [a_x, a_y]^\top$ and the corresponding covariance values $\sigma_\mathbf{a} = [\sigma_{a_x,a_x}, \sigma_{a_x,a_y}, \sigma_{a_y,a_y}]^\top$. These values are not computed in most recent works, but are key in practical applications such as the experiments we provide in sections III-A and III-B.

At each time step k, the module keeps a tracklet list of all the objects that are being tracked. Every one of them contains a KF [13], where a constant velocity model is considered. The i-th tracklet has the state $\mathbf{s}_{k,i} = [r_x, r_y, v_x, v_y]^\top \in \mathbb{R}^4$ and its covariance $\mathbf{P}_{k,i} \in \mathbb{R}^{4\times4}$. When the module receives a detection from any modality with a time difference $\Delta t = t_k - t_{k-1}$, all the tracklets are updated using the prediction equation:

$$\hat{\mathbf{s}}_{k,i} = \mathbf{F}(\Delta t)\mathbf{s}_{k-1,i}, \tag{1}$$

where $\mathbf{F} \in \mathbb{R}^{4\times4}$ is defined as:

$$\mathbf{F}(\Delta t) = \begin{bmatrix} \mathbf{I}_2 & \Delta t\mathbf{I}_2 \\ \mathbf{0} & \mathbf{I}_2 \end{bmatrix}. \tag{2}$$

The covariance of the tracklet after the prediction is:

$$\hat{\mathbf{P}}_{k,i} = \mathbf{F}(\Delta t)\mathbf{P}_{k-1,i}\mathbf{F}(\Delta t)^\top + \mathbf{Q}(\Delta t) \in \mathbb{R}^{4\times4}, \tag{3}$$

where the process noise of the prediction $\mathbf{Q}(\Delta t) \in \mathbb{R}^{4\times4}$ is modelled with an impulse of linear velocities as:

$$\mathbf{Q}(\Delta t) = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & b\Delta t\mathbf{I}_2 \end{bmatrix}, \tag{4}$$

where $b = 6$ m/s^2, representing a reasonable maximum for a driving scenario. Then, at each time step k, the J detections received $\{\mathbf{d}_{k,j} \in \mathbb{R}^m \mid j = 0, 1, \ldots, J\}$ are associated to tracklets by minimizing the global distance of the associations by means of [9]. Despite being several distances useful to be applied, we consider the Mahalanobis one $d(\hat{\mathbf{s}}_{k,i}, \mathbf{d}_{k,j})\colon \mathbb{R}^4 \times \mathbb{R}^m \to \mathbb{R}$ defined by:

$$d = \sqrt{(\mathbf{d}_{k,j} - h(\hat{\mathbf{s}}_{k,i}))^\top \mathbf{\Sigma}^{-1}_{k,i,j}(\mathbf{d}_{k,j} - h(\hat{\mathbf{s}}_{k,i}))}, \tag{5}$$

where $h\colon \mathbb{R}^4 \to \mathbb{R}^m$ depends on the sensor and maps the state to the detection space of the sensor. We use the Mahalanobis distance to be able to account for the covariance of the detections, this is especially relevant with camera detections of far away objects, as the frontal error is much larger than the lateral error, and using a simpler distance such as the Euclidean distance would result in incorrect associations. Once the detection $\mathbf{d}_{k,j}$ has been associated to the i-th track, the mean and covariance of the innovation are defined as:

$$\mathbf{y}_{k,i} = \mathbf{d}_{k,j} - h(\hat{\mathbf{s}}_{k,i}) \in \mathbb{R}^m, \tag{6}$$

$$\mathbf{\Sigma}_{k,i,j} = \mathbf{H}\hat{\mathbf{P}}_{k,i}\mathbf{H}^\top + \mathbf{R}_{k,j} \in \mathbb{R}^{m\times m}, \tag{7}$$

where $\mathbf{H} \in \mathbb{R}^{m\times4}$ is the Jacobian of h and $\mathbf{R}_{k,j} \in \mathbb{R}^{m\times m}$ the covariance of the detection.

The detection is used to update the tracklet that has been associated by following the next KF equations:

$$\mathbf{K}_{k,i} = \hat{\mathbf{P}}_{k,i}\mathbf{H}^\top\mathbf{\Sigma}_{k,i,j} \in \mathbb{R}^{4\times m}, \tag{8}$$

$$\mathbf{s}_{k,i} = \hat{\mathbf{s}}_{k,i} + \mathbf{K}_{k,i}\mathbf{y}_{k,i} \in \mathbb{R}^4, \tag{9}$$

$$\mathbf{P}_{k,i} = (\mathbf{I} - \mathbf{K}_{k,i}\mathbf{H})\hat{\mathbf{P}}_{k,i} \in \mathbb{R}^{4\times4}, \tag{10}$$

where $\mathbf{K}_{k,i}$ is the optimal Kalman gain, that weights how much the state changes considering the detection received, and it gives more importance to detections with lower errors. It is worth noting that the detections that have not been associated can be used to create new tracklets to include in the list. As radar sensors have a high rate of false positives, they are not considered for tracklet initialization. Finally, a tracklet removal step is also included to keep only relevant tracklets in the list, removing from the list those tracklets with a confidence below a defined threshold τ or that have not been associated for a given period of time κ. In all our experiments we set $\tau = 0.1$ and $\kappa = 3$ s. Since each sensor updates the shared tracklet list independently, no synchronization is required.

Some terms in the $\mathbf{o}$ vector are not kept in the state of the KF and are instead computed separately. This helps to better generalize to a larger set of sensors that cannot estimate these terms. For instance, we use the orientation of the velocity vector included in the state to calculate the yaw angle or heading when complete and accurate detections are not available. To infer the acceleration, we filter the differences in velocity from the KF state over time. First, we calculate the instant acceleration $\hat{\mathbf{a}}_k$ from the velocity and time differentials between the current time step t_k and the previous one t_{k-1} as $\hat{\mathbf{a}}_k = \max(\min((\mathbf{v}_k - \mathbf{v}_{k-1})/(t_k - t_{k-1}), b), -b)$, where max and min indicate the enterwise max/min operators, respectively. We then filter the acceleration with respect to previous accelerations as $\mathbf{a}_k = \gamma\mathbf{a}_{k-1} + (1-\gamma)\hat{\mathbf{a}}_k$, for $\gamma \in [0, 1]$. For all our experiments we set $\gamma = 0.8$.

We follow the multiplication score update function from [2] to update the confidence score of the tracklets and propose new approaches to calculate the width, height, length, and class, that are more robust to detections with partial bounding boxes and misclassifications. Next, we present our solutions.

A. Handling misclassifications

Learning-based object detectors have recently achieved very accurate results [15]. However, these approaches tend

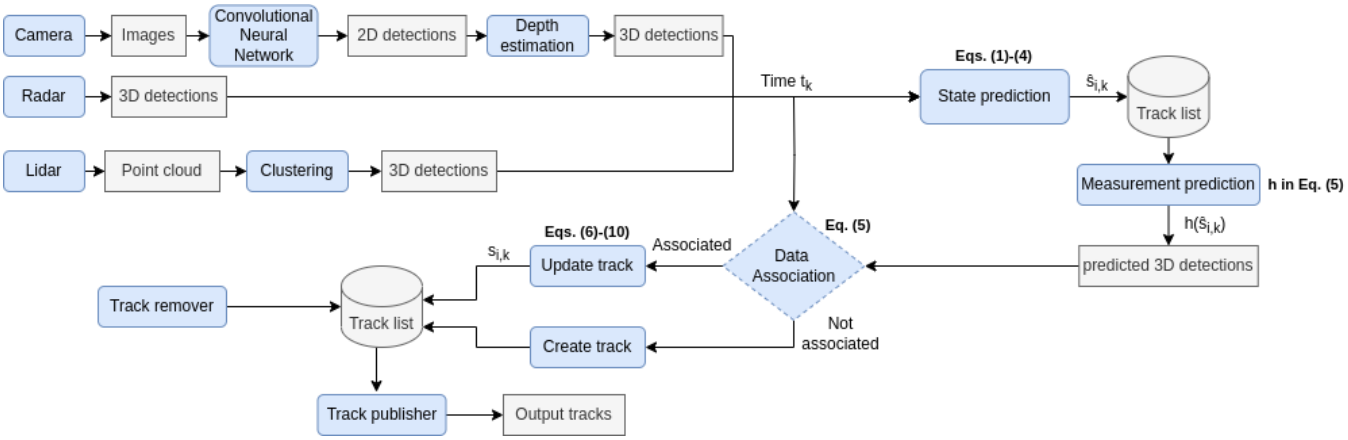


Fig. 2: **Multi-object tracking from sensor fusion.** The proposed system receives inputs from a configurable set of sensors (camera, radar, and lidar, in our particular configuration). Modality-specific object detectors generate 3D detections that are then combined by means of a KF to obtain more robust and stable 3D tracks. The corresponding equations and terminology used in the text can be found next to each module.

to degrade their performance due to the distribution shift that is produced when they are applied to sensors and scenarios not considered in the training distribution [20]. For example, in our experiments on proving grounds and open roads, we can observe many misclassification errors between similar classes, such as bicycles and motorcycles, or buses and trucks. These errors tend to appear in some of the frames, while the majority of the sequence is correctly classified, making it possible to recover from the error by exploiting temporal consistency. To handle this misclassification error in the MOT module, we associate tracklets and detections from similar classes that are not associated after the same class association. We can extract the pairs of similar classes from a confusion matrix (when available) taking as pairs the classes that are confused in over 1% of all data. We could also set them empirically, depending on our application and set of classes. This allows us to correctly associate detections to tracklets even when they contain misclassification errors. Each tracklet has a counter for each class with the number of times that it has been associated with a class, and then we output the class with the highest counter.

B. Handling partial detections

Some sensors and detection methods [15] are capable of detecting the full bounding box of the objects even when the object is partially visible. However, other sensors (like some radars) detect only the visible part of the object or do not provide accurate information on the size of the detected object. To reconstruct the complete bounding box from partial detections such that they can be associated with tracklets, we define minimum values for width, height, and length for each class. These values can be set empirically or based on the 20-th percentile of the values in the training set, when available.

Then, we assume that the closest part of the bounding box is detected and if any dimensions are lower than the minimum for the class, we calculate what the centre of the detection would be if those dimensions were equal to the minimum for the class, so that the closest point to the sensor remains the same. To this end, we define the closest point as the intersection of the bounding box with the line from the sensor to the centre of the box. We then resize the box and calculate the new centre of the box, so that the closest point remains the same, as it is shown in Fig. 3.

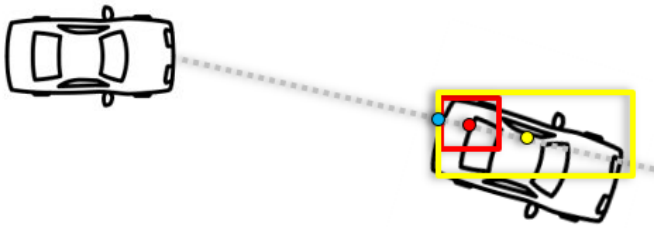


Fig. 3: **Handling of partial detections.** We consider a sensor mounted at the front bumper of the left car that detects the car in front with the partial bounding box in red. We then extract the closest point (in blue) and the new centre (in yellow), such that the blue point is the closest point for the bounding box with minimum size for a car (in yellow) that maintains the detected orientation. The orientation will be corrected in the MOT module.

To improve the accuracy of the data association when the orientations of the detections are not reliable or unavailable, we set the orientation of the bounding box to that of the tracklet that we are comparing it to. This reconstruction is especially important for classes with a large bounding box, such as trucks or buses. Some sensors will only detect the rear part and, if the full bounding box is not reconstructed, we would not be able to correctly associate the detections or update the tracklets, as we will see in the experiments.

III. EXPERIMENTS

We now present experimental results using our MOT method with different sensors on different real and challenging scenarios. We compare our results with the state-of-the-art CBMOT [2] tracker, since it has an open code base and it is a tracking-by-detection method that can be easily integrated into our experiments.

A. MOT for a traffic jam chauffeur

We first test our method in proving grounds, by integrating our MOT technique as the perception module of a traffic jam chauffeur function. To generate the ground truth data in order to analyze the performance of the system, we equip our ego vehicle and the target vehicle in each scenario with the OxTS RT3000, high-precision differential GNSS devices that are in constant communication so that they can estimate their relative position with an accuracy of 1 cm. Since we want to test an ADAS function that is currently present in commercial vehicles and to be able to extract comparable results, we use a front radar (Continental ARS 408-21) and a front camera with integrated detections (Mobileye 630) as sensors.

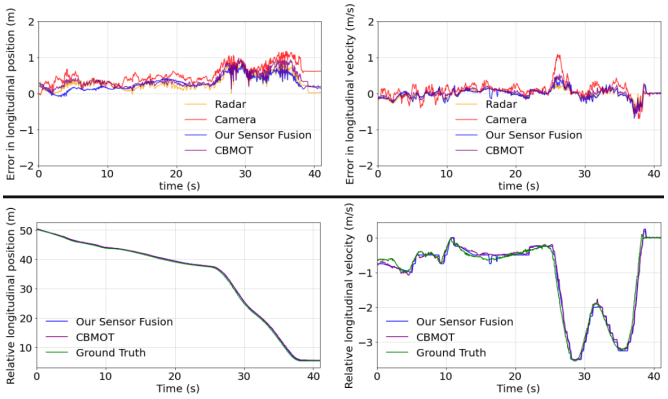
The radar is more accurate in estimating the longitudinal position and velocity of the target but has too many false positives, so we rely on the camera for tracklet initialization. We report results for different car-to-car scenarios, where the ego vehicle with the perception system and the target vehicle in front drive in the same lane. For the parametrizations S_1 and S_2, the ego vehicle approaches while driving faster than the target vehicle in front and then adapts to the velocity of the target vehicle. For S_3, S_4, and S_5 both vehicles are driving at 50 km/h, with some oscillations, then the vehicle in front brakes at different deceleration values and the ego vehicle reacts accordingly by breaking until both vehicles are at a full stop. In Tab. I, we show the average error in position and velocity. We can see that our sensor fusion outperforms both sensors and CBMOT [2]. In the scenarios with rain, S_4 and S_5, we can see that the camera is more affected than the radar and the error in longitudinal position is much larger than in clear conditions. To complement these quantitative results, Fig. 4 shows the evolution of the error in longitudinal position and velocity of each sensor over time in the last three scenario parametrizations S_{3-5}, along with the reference position and velocity, and the estimations of our sensor fusion and CBMOT [2]. Here we can see that the output of our sensor fusion is smoother than the result of CBMOT [2] and the input sensors. CBMOT [2] does not consider the error in the detections, and therefore each detection received has a similar effect on the state of the KF. We overcome this limitation by considering the location error of the detections so that the KF can update its state giving more weight to the detections with a lower error and less to the detections with a higher error. Finally, Fig. 5 shows more details for a time instant in S_4, where we can see an image from a reference camera and the detections from each sensor along with our sensor fusion's tracklet, the CBMOT [2] tracklet, and the corresponding ground truth. The error of CBMOT [2] is larger than that of our method as it does not take into account the error of the sensors and treats them equally, we are also capable of reconstructing the original bounding box, whereas CBMOT [2] keeps the partial bounding box provided by the sensors.

		S_1	S_2	S_3	S_4	S_5	Avg
Sensor Fusion (Ours)	$\bar{e}_x$	**0.20**	**0.18**	**0.30**	0.17	**0.24**	**0.22**
	$\bar{e}_y$	**0.29**	**0.28**	**0.43**	0.33	**0.54**	**0.37**
	$\bar{e}_{v_x}$	**0.10**	**0.12**	**0.11**	0.15	0.25	**0.15**
	$\bar{e}_{v_y}$	**0.15**	**0.15**	**0.30**	**0.20**	**0.58**	**0.28**
CBMOT [2]	$\bar{e}_x$	0.42	0.57	0.48	0.38	1.33	0.64
	$\bar{e}_y$	0.40	0.52	0.51	0.35	0.65	0.49
	$\bar{e}_{v_x}$	0.17	0.13	0.12	0.13	0.26	0.18
	$\bar{e}_{v_y}$	0.17	0.18	0.32	**0.20**	0.62	0.30
Radar	$\bar{e}_x$	0.85	0.74	0.47	**0.17**	0.34	0.51
	$\bar{e}_y$	0.63	0.60	0.84	0.43	0.68	0.64
	$\bar{e}_{v_x}$	0.23	0.15	0.13	**0.12**	**0.22**	0.17
	$\bar{e}_{v_y}$	0.20	0.20	0.33	0.21	0.61	0.31
Camera	$\bar{e}_x$	0.21	0.20	0.48	1.11	1.58	0.72
	$\bar{e}_y$	0.36	0.36	0.47	**0.31**	0.63	0.43
	$\bar{e}_{v_x}$	**0.10**	0.13	0.18	0.24	0.31	0.19
	$\bar{e}_{v_y}$	0.17	0.18	0.31	**0.20**	0.64	0.30
Parameters	v_{ego}	60	50	50	50	50	
	v_{tar}	20	20	50	50	50	
	a_{tar}	-	-	-1	-2	-4	
	rain	no	no	no	yes	yes	

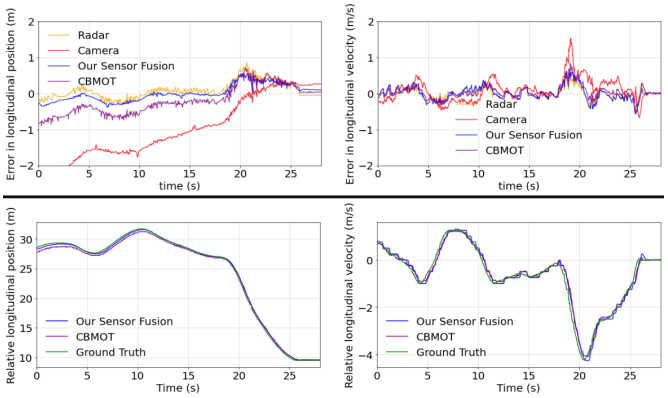
TABLE I: **Car-to-car average errors**. The table reports the Mean Absolute Error (MAE) in longitudinal and lateral position $\bar{e}_x$, $\bar{e}_y$ in meters [m], as well as the MAE in longitudinal and lateral velocity $\bar{e}_{v_x}$, $\bar{e}_{v_y}$ in meters/second [m/s]. We evaluate each sensor independently, our sensor fusion algorithm and the competing approach CBMOT [2], all of them for a scenario with 5 different parametrizations, S_{1-5}, where the ego vehicle drives at a speed of v_{ego} km/h and has a target vehicle in front driving at v_{target} km/h, then the target vehicle breaks at a_{target} m/s^2. We also include the average error across scenario parameterizations, and we highlight the best results in bold. S_{1-3} are in clear conditions and S_{4-5} raining.

B. MOT for traffic monitoring

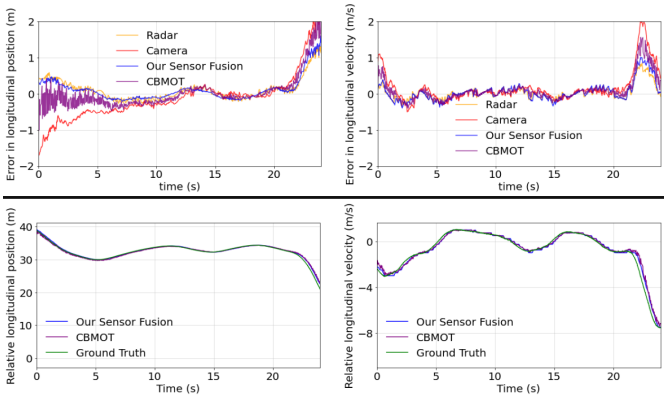
Finally, we test our system with a larger and more diverse set of sensors and apply it to traffic monitoring on highways. We equip two vehicles with three cameras (Basler a2A1920-51gcBAS), one radar (Continental ARS 408-21), one frontal lidar (ibeo LUX 4L), and one 360° lidar (Ouster OS2-128). The goal of this application was to determine the effect on other highway users of a truck platoon (a group of trucks communicating with each other and driving at close distances with automated speed control). We placed one equipped vehicle in front of the platoon and the other one behind the platoon. We then processed the data recorded from the sensors offline to generate the trajectories (position, velocity, and acceleration at every time step) of all other road users. We recorded data with the platoon active, meaning the trucks were driving at a closer distance, and with the platoon inactive, with the drivers driving as in normal conditions. In



$\mathbf{S_3}$: $v_{ego} = 50$ km/h, $v_{tar} = 50$ km/h, $a_{tar} = -1$ m/s^2. No rain



$\mathbf{S_4}$: $v_{ego} = 50$ km/h, $v_{tar} = 50$ km/h, $a_{tar} = -2$ m/s^2. Rain



$\mathbf{S_5}$: $v_{ego} = 50$ km/h, $v_{tar} = 50$ km/h, $a_{tar} = -4$ m/s^2. Rain

Fig. 4: **Traffic Jam Chauffeur scenarios $\mathbf{S_3}$, $\mathbf{S_4}$ and $\mathbf{S_5}$.** In all cases, longitudinal position and velocity are considered on the left/right, respectively. **Top:** Position and velocity error evolution for every sensor we use, as well as our method and CBMOT [2]. **Bottom:** Evolution of the relative longitudinal position and velocity from our method and CBMOT [2], and the corresponding ground truth.

Fig. 5: **Traffic Jam Chauffer scenario $\mathbf{S_4}$: Image and detections.** We display information from the time $t = 2.74$ s. On the left, the effect of the rain on the camera is visible. On the two right images, we see the detections of the camera (green square), and radar (cyan squares); along with the ground truth (blue rectangle) and the sensor fusion (green edges), our method in the middle and CBMOT [2] on the right.

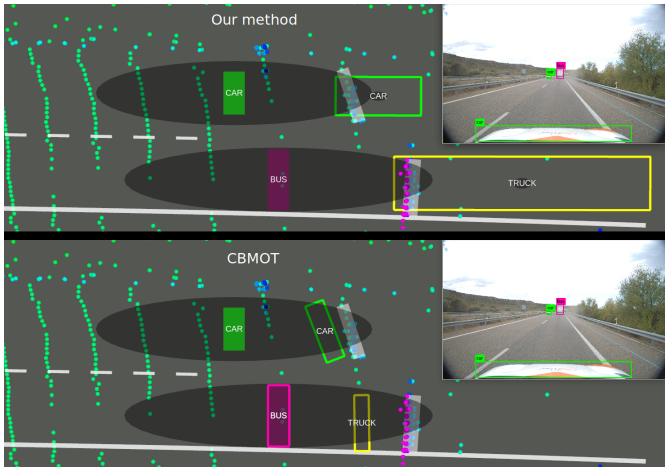


Fig. 6: **Qualitative results on a real traffic monitoring scenario.** The raw point cloud from the 360° lidar color-coded by height with its detections in grey, the camera detection in yellow for trucks, green for cars, and pink for buses, and the results of the MOT algorithm are the edges of the bounding boxes color-coded as the camera detections, with our method on top and CBMOT [2] on the bottom. The covariance of each detection is displayed as a shadow ellipse. For reference, the camera image with detections is shown in the top-right corner. Detected lanes by a camera with integrated detections are shown in white.

future work, these trajectories will be analyzed to assess the effect of activating the platoon on other highway users. For this experiment we do not have ground truth data available, so can we only report qualitative results. As we can see in Fig. 1, the radar is providing the velocity information but also many false positives, the cameras have a large longitudinal error, and the lidars provide the most accurate positioning and some false positives. Thus, we use camera detections to initialize tracklets. Then, the rest of the detections are associated to those tracklets and our MOT module calculates the position, size, class, velocity, and acceleration of every vehicle in the scene over time, obtaining estimations with

a lower covariance and error. None of the sensors provide complete bounding box estimations, but our MOT is capable of reconstructing the complete bounding box by using our proposed method, whereas CBMOT [2] keeps the partial bounding boxes from the sensors. Some camera detections contain misclassifications, especially confusing trucks and buses as in Fig 6, but this is handled in the MOT with our approach. We can see the effect of not handling misclassifications in the CBMOT [2] results, where an extra tracklet is erroneously created. The positional error is also higher in CBMOT [2] since they do not consider the error in the detections.

IV. CONCLUSION

We have presented a versatile and accurate MOT method that works with a wide variety of sensors and detectors without the need to synchronize them, and that can be used in different real-world scenarios. To improve the robustness in these scenarios, we proposed a method to handle misclassifications and partial bounding boxes from the object detector in the MOT module. We also proposed a simple yet effective method to estimate the error in the conversion from 2D to 3D camera detections using depth estimation, and a novel method to obtain the acceleration in the MOT module. An in depth discussion is included to justify the importance of this information when MOT systems are deployed in real applications, such as the traffic jam chauffeur function that we have used to test our system in proving grounds. Our system was tested for extensive sets of sensors by mounting cameras, lidars, and radars on two vehicles to monitor highway traffic. We compared our results against CBMOT [2] and highlighted how our contributions improve the results. We believe that our approach is generic and can be easily integrated with other MOT methods that can benefit from our contributions.

V. ACKNOWLEDGMENTS

This work has been supported by the project MoHuCo PID2020-120049RB-I00 funded by MCIN/AEI/10.13039/501100011033 and by the Government of Catalonia under 2020 DI 105.

REFERENCES

[1] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," in *IROS*, 2020.

[2] Nuri Benbarka, Jona Schröder, and Andreas Zell, "Score refinement for confidence-based 3D multi-object tracking," in *IROS*, 2021.

[3] Ziqi Pang, Zhichao Li, and Naiyan Wang, "Simpletrack: Understanding and rethinking 3D multi-object tracking," *arXiv preprint arXiv:2111.09621*, 2021.

[4] Li Zhang, Yuan Li, and Ramakant Nevatia, "Global data association for multi-object tracking using network flows," in *CVPR*, 2008.

[5] Samuel Schulter, Paul Vernaza, Wongun Choi, and Manmohan Chandraker, "Deep network flow for multi-object tracking," in *CVPR*, 2017.

[6] Hsu-kuang Chiu, Jie Li, Rareş Ambruş, and Jeannette Bohg, "Probabilistic 3D multi-modal, multi-object tracking for autonomous driving," in *ICRA*, 2021.

[7] Yihan Zeng, Chao Ma, Ming Zhu, Zhiming Fan, and Xiaokang Yang, "Cross-modal 3D object detection and tracking for auto-driving," in *IROS*, 2021.

[8] Harold W Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[9] David F Crouse, "On implementing 2D rectangular assignment algorithms," *TAESS*, vol. 52, no. 4, pp. 1679–1696, 2016.

[10] Aleksandr Kim, Aljoša Ošep, and Laura Leal-Taixé, "Eagermot: 3D multi-object tracking via sensor fusion," in *ICRA*, 2021.

[11] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[13] Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[14] Simon Julier and Jeffrey Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *AeroSense*, 1997.

[15] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl, "Center-based 3D object detection and tracking," in *CVPR*, 2021.

[16] Kaiqi Liu and Jianqiang Wang, "Fast dynamic vehicle detection in road scenarios based on pose estimation with convex-hull model," *Sensors*, vol. 19, no. 14, pp. 3136, 2019.

[17] Yang Liu, Bingbing Liu, and Hongbo Zhang, "Estimation of 2D bounding box orientation with convex-hull points-a quantitative evaluation on accuracy and efficiency," in *IV*, 2020.

[18] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy, "Robust multi-modality multi-object tracking," in *ICCV*, 2019.

[19] Apoorva Joglekar, Devika Joshi, Richa Khemani, Smita Nair, and Shashikant Sahare, "Depth estimation using monocular camera," *IJCSIT*, vol. 2, no. 4, pp. 1758–1763, 2011.

[20] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao, "Train in germany, test in the usa: Making 3D object detectors generalize," in *CVPR*, 2020.