2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

DFR-FastMOT: Detection Failure Resistant Tracker for Fast Multi-Object Tracking Based on Sensor Fusion

Mohamed Nagy, Majid Khonji, Jorge Dias and Sajid Javed[1]

***Abstract*— Persistent multi-object tracking (MOT) allows autonomous vehicles to navigate safely in highly dynamic environments. One of the well-known challenges in MOT is object occlusion when an object becomes unobservant for subsequent frames. The current MOT methods store objects information, such as trajectories, in internal memory to recover the objects after occlusions. However, they retain short-term memory to save computational time and avoid slowing down the MOT method. As a result, they lose track of objects in some occlusion scenarios, particularly long ones. In this paper, we propose DFR-FastMOT, a light MOT method that uses data from a camera and LiDAR sensors and relies on an algebraic formulation for object association and fusion. The formulation boosts the computational time and permits long-term memory that tackles more occlusion scenarios. Our method shows outstanding tracking performance over recent learning and non-learning benchmarks with about 3% and 4% margin in $MOTA$, respectively. Also, we conduct extensive experiments that simulate occlusion phenomena by employing detectors with various distortion levels. The proposed solution enables superior performance under various distortion levels in detection over current state-of-art methods. Our framework processes about $7,763$ frames in 1.48 seconds, which is seven times faster than recent benchmarks. The framework will be available at https://github.com/MohamedNagyMostafa/DFR-FastMOT.**

I. Introduction

Multi-object tracking (MOT) provides information about the surrounding objects, allowing autonomous vehicles (AVs) to avoid collisions with other cars by making proper navigation decisions. AVs rely on sensors such as cameras and LiDAR to obtain sufficient information for object tracking. We categorize the recent research work into two groups: the first group uses mono-sensor, and the other group employs multi-sensors to collect information for MOT.

As shown by Chaabane [1], Tokmako [2], and Wu [3], they rely on either a camera or LiDAR sensor to localize and track objects. However, sensors usually have limitations; for example, LiDAR performance decreases in fog and sandy weather. On the other hand, a camera sensor resolution drops in night scenes. Accordingly, Wang [4] and Kim [5] fuse data from a camera and LiDAR sensors as well as employ Kalman Filter (KF) [6] to track objects by trajectory estimation. Similarly, Wu [3] fuses LiDAR, IMU, and GPS sensors to track objects in 3D, considering object orientations. Even though Wu [3] achieves high tracking performance as a learning-based solution compared to other benchmarks, including non-learning based, the tracker skeleton is vast. It requires evolved hardware, like multiple GPUs, which makes the method not applicable to some applications, such as mobile robots. Besides, learning-based solutions fail under prolonged occlusion scenarios when an occluded object does not appear within the frame's stack fed to the deep learning model. Furthermore, the proposed non-learning solutions, like Wang [4] and Kim [5], sacrifice the tracking performance for less computational time by maintaining a short-term memory for objects during occlusions. Thus, these methods fail to capture some occlusions, as shown in Figure 1.

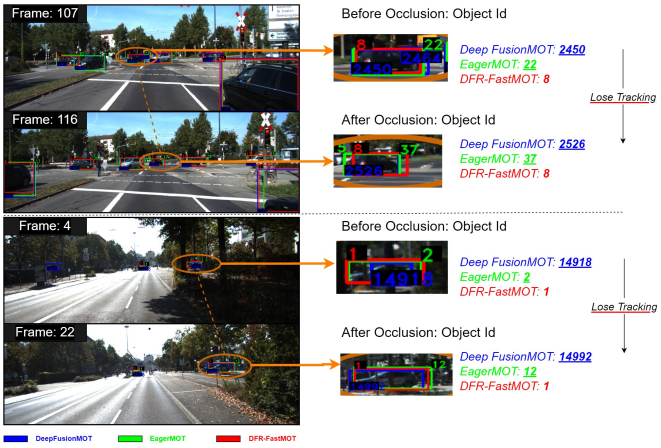

Fig. 1: Our approach recovered objects in two occlusion scenarios when DeepFusionMOT [4] and Eager-MOT [5] methods could not retrieve them.

In this work, we propose an MOT method that tackles the limitations of the previous work, non-learning (Low performance caused by object occlusion) and learning (High computational time) solutions. Our method utilizes data from LiDAR and camera sensors. It relies on an algebraic model to associate and fuse objects, enhancing the computational time and allowing memory expansion to capture more occlusion scenarios. The method can utilize mono or multi-detectors to perform MOT, which makes it applicable to several autonomous applications, like mobile robots.

We summarize our contributions as follows:

1) We propose a light MOT framework with a remarkable tracking performance over the current learning and non-learning MOT methods with less computational time, which is about seven times faster than recent non-learning methods.

[1]This work was supported by Khalifa University under Award Ref. CIRA-2020-286, KKJRC-2019-Trans1, and KUCARS. Mohamed Nagy, Majid Khonji, Jorge Dias, and Sajid Javed are with the Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates (emails: mohamed.nagy@ieee.org, {majid.khonji, jorge.dias, sajid.javed}@ku.ac.ae).

2) We simulate occlusion phenomena by employing various detection distortion levels to evaluate the performance of our solution under object occlusion. The framework shows superior performance throughout the experiments over other benchmarks.
3) We show the framework's capability to track objects under different types of occlusion, such as off-scene objects, objects with single detection information 2D/3D, and multi-object occlusion.

II. Literature Review

Many research works propose outstanding performance in object tracking [7]–[11] that employ a tracking-by-detection paradigm. Bewley et al. [12] introduce SORT for online MOT where they use Kalman filter (KF) [6] for object's trajectory estimation using historical observations. They operate the Hungarian algorithm [13] to associate the objects in the subsequent frames. On the other hand, Bochinski et al. [14] take advantage of high-rate frames and sophisticated detectors to associate objects using Intersection Over Union (IoU). Hence, they achieve a competitive speed as an MOT approach.

Despite the outstanding performance achieved by these trackers, object occlusion remains challenging since they need to handle memory for the tracked objects. Moreover, they assume the observant is always static, which is not applicable in AV applications where the observant is usually in motion. In addition, they utilize mono-sensor for MOT when it is preferable to fuse multi-sensor data to overcome mono-sensor faults.

Other research works propose MOT solutions for AVs. We will discuss the research work that uses mono-sensor [1]–[3], [15], [16] and multi-sensor [4], [5], [17]–[19] and emphasize the pros and cons of each.

Chaabane et al. [1] propose a joint deep learning model that composes detection and tracking tasks relying on object appearance in captured camera frames by integrating an LSTM model [20] to capture motion constraints. However, This method needs to tackle the object occlusion problem. Hence, Tokmakov et al. [2] introduce an object permanence tracker supported by a spatio-temporal and recurrent memory module that identifies observed objects' location and identities using whole history. Although they have superior performance for tracking occluded objects, they maintain a permanent history for objects that may harm association efficiency in long-term running. Besides, utilizing one source of information reduces the robustness of the solution.

On the other hand, research works [4], [5], [17] consider object tracking using multi-sensor fusion. The methods obtain information using sophisticated detectors from 2D camera frames and 3D LiDAR point cloud. Kim et al. [5] use pre-trained deep learning models [21]–[23] for object detection. Next, they engage KF to estimate object trajectories by associating the detection outcomes with prior observations using IoU. The limitation of this work is that they use a naive KF model that assumes objects always have a constant velocity, which does not apply to objects like cars. Wang [18] and Kim [19] apply non-linear filters to estimate complex motion for tracked objects. Kim [19] uses object distance from LiDAR and Radar sensors to track objects utilizing the extended KF and shows the result by operating Prescan simulator [24] in different scenarios. In contrast, Wang [18] proposes a modified version of the unscented KF for state estimation that improves tracking accuracy. Meanwhile, Weng et al. [25] use baseline algorithms, Hungarian algorithm [13], and show the capability of baseline algorithms to achieve a comparable tracking accuracy to the proposed deep learning solutions.

The previous work employs a short-term memory for objects that prevents capturing some occlusion scenarios, which eventually influences the overall tracking performance. In this work, we tackle this problem by introducing an algebraic formulation for association and fusion steps that enhances the MOT computational time and allows the integration of long-term memory.

III. Methodology

The framework accepts $2D$ and $3D$ detection from camera D_t^{2d} and LiDAR D_t^{3d} at time t, and unifies the detected objects in a matching phase to prevent duplicated information of the same object. Next, we associate the detection with prior observed objects in the memory to obtain sets of unmatched objects $O_t^{unmatched}$ and matched objects $O_t^{matched}$. The memory module updates the history of the matched objects $O_t^{matched}$ and adds the unmatched objects $O_t^{unmatched}$ as new objects. Furthermore, the memory module discards aged objects that did not appear for a number of frames H_t that exceeds ϵ frames. We eventually employ KF with constant acceleration to update objects' trajectories $T_t^{\forall}$ stored in the memory and obtain state estimation for the subsequent frames based on changes in the trajectories. Figure 2 shows an overview of the framework.

In this section, we begin by exploring the detection inputs for the framework, Section III-A. Next, we will explain the association and fusion mechanism in Section III-B. Then, we will discuss the tracking module in Section III-C, followed by the memory management module of the framework in Section III-D.

A. Detection Module

The framework requires input data from a camera and LiDAR sensors; however, it may rely solely on 2D or 3D detectors to perform tracking. The framework can employ a mono-detector, either 2D or 3D, and obtain the other detection information using calibration parameters and point cloud projection that allows transformation from camera to LiDAR system coordinates and vice versa, as shown in Figure 2. The essence of employing mono-detection is to make the framework applicable for real-time applications when employing two detectors will be costly in terms of power consumption and hardware requirements for some AVs, particularly for mobile robots.

In the case of multi-detectors, we involve an additional step, matching, where we match detection D_t^{2d} and D_t^{3d} from the detectors to prevent detection duplication for the same

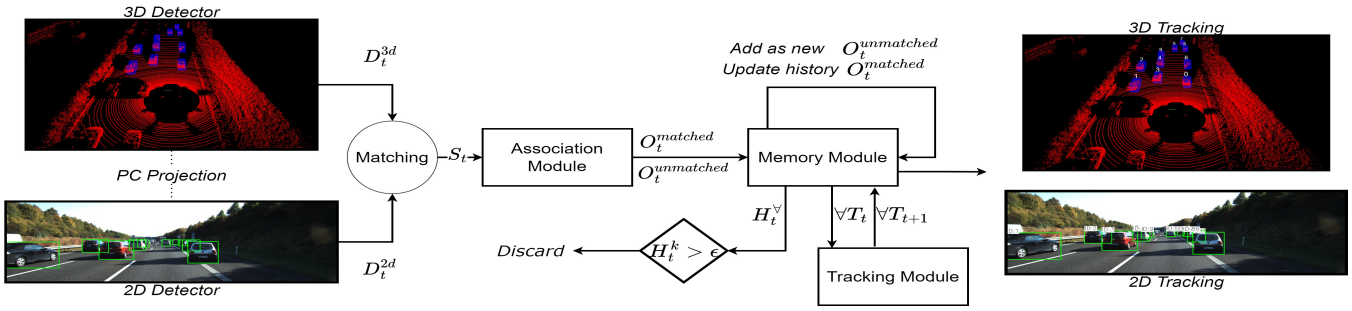

Fig. 2: A high-level overview of the proposed framework presents a pipeline of the vital modules in the framework. The framework accepts the detection of objects in sensor data and unifies the objects to prevent object duplication. Next, it associates the unified objects with objects stored in the memory. Eventually, the framework updates the trajectory estimation of all objects in the memory and estimates their state for the next frame.

object. To accomplish so, we use the same procedures of mono-detector by obtaining $2D$ detection for objects from D_t^3d. Then, we consider a detection duplication of an object when the transformed $2D$ bounding box from D_t^{3d} matches one of the objects in D_t^{2d}. In the case of matching, we assign the matched data in D_t^{2d} and D_t^{3d} to the object; otherwise, the detection will be classified as two different objects. Hence, the outcome is a set of objects S_t that contains objects with $2D$, $3D$, or a combination of the detection.

B. Association Module

The framework performs association and fusion steps in an algebraic formulation that makes it ***seven times faster*** than recently published benchmarks [4] [5]. We initially introduce an association matrix, Equation 1, for each sensor; association matrix M_c for the camera and M_l for the LiDAR sensor. The matrices have the same formulation as Equation 1, where the number of rows m is the number of current detected objects by the sensor, and the number of columns n is the number of objects stored in the memory. v_{ij} represents the association value between a recent detected object i and object j stored in the memory. Sections III-B.1 and III-B.2 describe how the association value is assigned based on the sensor's type.

$$M_{m\times n} = \underbrace{\left.\begin{pmatrix} v_{0,0} & v_{0,1} & v_{0,2} & \cdots \\ v_{1,0} & v_{1,1} & v_{1,2} & \cdots \\ v_{2,0} & v_{2,1} & v_{2,2} & \cdots \\ \vdots & & & \ddots \\ v_{m,0} & v_{m,1} & v_{m,2} & \cdots \end{pmatrix}\right\} m \text{ observed objects}}_{n \text{ prior observed objects}} \tag{1}$$

In the fusion part, we merge the association matrices using Equation 2 to obtain a fused association matrix M_f. α_c and α_l represent the significance of the matrices. For example, If LiDAR detection provides a robust association result rather than camera detection, we would trust M_l over M_c by increasing α_l and decreasing α_c.

$$\begin{aligned} M_f &= \alpha_c M_c + \alpha_l(1 - M_l) \\ \alpha_c + \alpha_l &= 1 \\ \alpha_c, \alpha_l &\leq 1 \end{aligned} \tag{2}$$

Based on the matching step explained in Section III-A, we can guarantee that both M_c and M_l will have the same dimension. In case detection for an object does not exist, we assign the object's row in the matrix by the matrix threshold based on the sensor type, introduced in Sections III-B.1 and III-B.2.

1) Camera Association Matrix (M_c): To associate two objects in 2D frames, we use IoU under a certain threshold a_c. The association value v_{ij} between two objects can be represented as follows:

$$v_{ij} = \begin{cases} v_{IoU} : v_{IoU} \geq a_c, \\ 0 \quad\quad : \text{Otherwise.} \end{cases}$$

If v_{IoU} is less than a_c, the corresponding v_{ij} will equal 0, which means the objects are not matched. Otherwise, we maintain v_{IoU} for v_{ij}. Accordingly, the M_c matrix shows matched objects with a corresponding v_{ij} close to 1. The time requires for matrix traversal is $\mathcal{O}(mn)$.

2) LiDAR Association Matrix (M_l): To handle more object occlusions, we do not utilize IoU for the 3D association matrix; we employ $3D$ centroid distance [26] instead. When a prolonged occlusion happens, it is less likely to have an intersection between the estimated bounding box and the currently detected bounding box. Hence, we compute the Euclidean distance between centroids of the $3D$ bounding boxes, which gives more flexibility to re-identify occluded objects. We perform a similar strategy as the M_c matrix, $\mathcal{O}(mn)$, by having a_l as a Euclidean distance threshold. v_{ij} will be assigned as follows:

$$v_{ij} = \begin{cases} v_{dist} : v_{dist} < a_l, \\ a_l \quad\quad : \text{Otherwise.} \end{cases}$$

We match the objects when v_{dist} converges to zero; otherwise, we assign the threshold a_l that refers to these two objects as not matched.

To equally fuse M_c and M_l, we need to ensure that both matrices have values between 0 and 1, which is not the case for M_l. Therefore, we normalize M_l by dividing by the maximum value, a_l, to obtain a normalized matrix. Eventually, we need to reverse this behavior to match M_c matrix. Thus, we perform $(1 - M_l)$ as shown in Equation 2

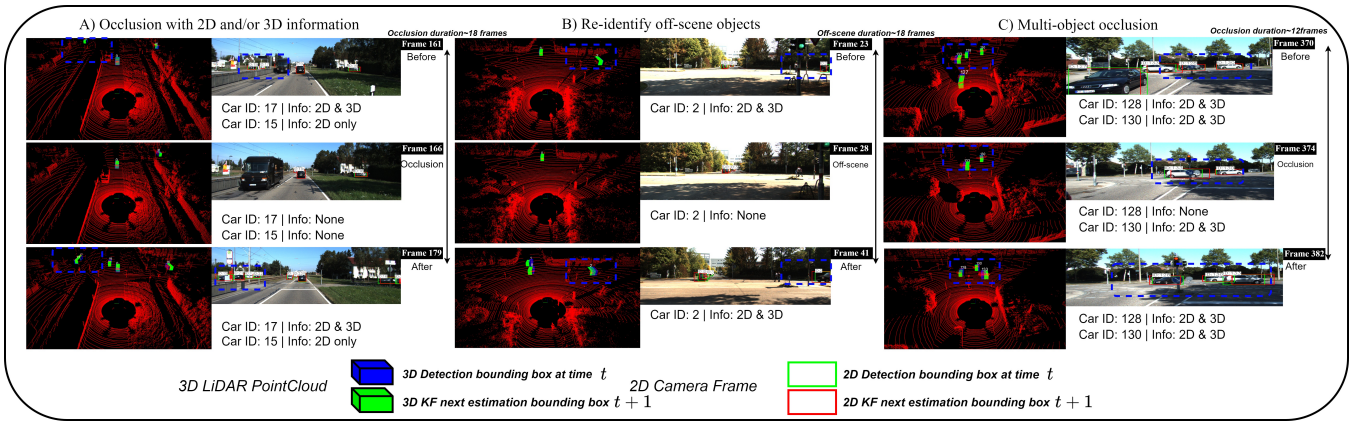

Fig. 3: The performance of the framework over various occlusion scenarios. A) Two occluded cars with either 2D or 3D information. B) Re-identify off-scene objects caused by ego motion. C) Two cars occluded by a third tracked car.

to obtain a reversed behavior. Thus, v_{ij} represents matched objects when it converges to one instead.

Using Equation 2, we obtain a fused matrix M_f that combines the association matrices from the sensors. We eventually perform an algorithm similar to the Hungarian algorithm [13] to obtain the final association of the objects from M_f under association threshold a_f. We pick the maximum value v_{ij} that exists in M_f. Then, we associate the corresponding objects and repeat the same algorithm till the maximum value v_{ij} is below the threshold a_f. In this case, the algorithm terminates. In the worst case, searching for all elements in M_f takes $\mathcal{O}(m^2n^2)$, which is the complexity of matching all objects in M_f. Thus, the overall computational time for the association step of each sensor (M_c, M_l) and the fused matrix (M_f) is $\mathcal{O}(2mn) + \mathcal{O}(m^2n^2) \rightarrow \mathcal{O}(m^2n^2)$.

C. Tracking Module

The tracking module performs KF for all objects stored in the memory. The process involves two central states: Update state and Prediction state. The module updates objects' trajectory estimation computed at time $t-1$ in a $2D$ camera frame T_t^{2d} and $3D$ LiDAR point cloud T_t^{3d} using current observation of the objects by the sensors. The module updates objects' location, velocity, and acceleration in the update state. In the prediction state, the module computes the following estimation of the objects' location in both spaces, T_{t+1}^{2d} and T_{t+1}^{3d}, given the last updated parameters, velocity and acceleration, available for each object. In occlusion, the module only launches the prediction state, which means the estimation of objects' location and velocity will change; meanwhile, the acceleration will be the same since we use a constant acceleration model. Figure 3 shows three scenarios for object occlusion and how our method retrieves the occluded objects given either 2D, 3D, or both trajectories information.

To reduce the computational time, we present the KF system and object data into matrices and apply KF estimation on the least number of points required to draw 2D and 3D bounding boxes, which are two boundary points. For the 2D bounding box, we solely employ top-left and bottom-right points to perform KF operations, and the top-left and bottom-right opposite corner points in the case of a 3D bounding box. We approach the following steps to perform KF estimation for an object:

- ***First-time observation***: When the object is observed for the first time, we assign the detected bounding box location as the initial state for the object $T_t^{2d/3d}$, followed by the KF prediction state to obtain state estimation $T_{t+1}^{2d/3d}$.
- ***Mono-sensor observation***: For simplification, assume the LiDAR sensor solely observes the object. In this case, we perform both update and prediction states for KF on the 3D bounding box data and perform the prediction step on the last estimated 2D bounding box, if available.
- ***Multi-sensor observation***: If LiDAR and the camera observe the object, we perform the update and prediction states for KF on both 2D and 3D bounding box information.
- ***Not observed***: In the case of not observing an object, we still perform the prediction state for KF on both 2D and 3D bounding box information to keep track of the object for a future appearance.

D. Memory Module

The memory module is essential for object occlusion. Our framework enables long-term memory to capture high-range occlusion scenarios, as shown in Figure 3. We discard aged objects that do not appear for subsequent frames ϵ, as illustrated in Figure 2. To do so, we employ Equation 3 to select the minimum number of subsequent frames H_t^k where an object k is not observed by the camera for a number of frames $H_t^{k^{2d}}$ and LiDAR for a number of frames $H_t^{k^{3d}}$ at time t.

$$H_t^k = min(H_t^{k^{2d}}, H_t^{k^{3d}}) \tag{3}$$

As explained in section III-C, the framework consistently updates KF estimation for all remaining objects in the

Detector	Method	HOTA↑	MOTA↑	MOTP ↑	DetA↑	AssA↑	IDSW↓	IDF1↑	MT↑	ML↓	Frag↓
2D YOLOv3 [27]	EagerMOT [5]	36.5%	41.6%	76.4%	35.4%	38%	792	48.1%	105	124	855
+	DeepFusion-MOT [4]	30%	31.8%	**77.4%**	27.4%	33.1%	696	40.3%	28	241	**819**
3D PC Projection	**DFR-FastMOT(Our)**	**39.2%**	**44.5%**	76.3%	**36.3%**	**42.8%**	**386**	**53.4%**	**113**	**106**	907
2D RCC [28]	EagerMOT [5]	70.8%	82.2%	**90.8%**	79.5%	63.3%	1303	74%	413	26	213
+	DeepFusion-MOT [4]	42.6%	40.2%	90.7%	41.7%	43.7%	1203	44.9%	81	211	1063
3D PC Projection	**DFR-FastMOT(Our)**	**81.9%**	**91%**	90.7%	**83.6%**	**80.3%**	**215**	**90.1%**	**485**	**12**	**239**
2D RCC [28]	EagerMOT [5]	69.1%	67.2%	85.2%	62.6%	76.5%	112	80.9%	499	10	316
+	DeepFusion-MOT* [4]	77.5%	87.3%	86.6%	75.4%	79.9%	**83**	**91.9%**	450	14	301
3D PorintRCNN [29]	**DFR-FastMOT(Our)**	**82.8%**	**90.7%**	**90.6%**	**83.1%**	**82.6%**	177	91.4%	**503**	**8**	**238**
2D RCC [28]	EagerMOT [5]	78%	87.3%	87.6%	76.8%	79.5%	**91**	89.3%	509	7	246
+	DeepFusion-MOT [4]	77.2%	85.8%	86.8%	74.9%	79.9%	136	**90.9%**	426	22	280
3D PointGNN [30]	**DFR-FastMOT(Our)**	**82.2%**	**90.2%**	**90.5%**	**82.5%**	**82%**	189	90.5%	**516**	**6**	**224**
2D TrackRCNN [31]	EagerMOT [5]	68.6%	63.2%	85%	60.8%	**77.6%**	102	80.4%	**508**	**10**	**262**
+	DeepFusion-MOT [4]	62.2%	57.7%	**86.9%**	51.5%	75.3%	**65**	73.2%	307	129	286
3D PorintRCNN [29]	**DFR-FastMOT(Our)**	**70%**	**80.1%**	82.6%	**67.7%**	72.7%	193	**85.9%**	425	18	608
2D TrackRCNN [31]	EagerMOT [5]	**78.2%**	**86%**	**87.5%**	**75.8%**	**80.8%**	**83**	**90.2%**	**522**	**6**	**184**
+	DeepFusion-MOT [4]	66%	65.9%	86.2%	58.6%	74.6%	133	79 %	351	86	386
3D PointGNN [30]	**DFR-FastMOT(Our)**	69.7%	80%	82.6%	67.7%	72.1%	203	85.2%	429	13	561

TABLE I: Results using the KITTI evaluation tool on the training and evaluation dataset employing three detector performances *(Poor/Moderate/High)* with two of the recent benchmarks, EagerMot [5] and DeepFusion-MOT [4]. **Black bold** highlights are the highest achieved value per metric for each detector. **Red bold** highlights are the highest achieved value per metric over all detectors. We provide this experiment results to the public so they can access the results on individual streams of the dataset throughout this link: https://github.com/MohamedNagyMostafa/KITTI-MOT.Bench-Evals.git. We highly encourage the following research to use the results to evaluate their trackers and share the tracking results.

memory and stores the new state estimation to be integrated into the association module later.

IV. Results and Discussion

We split this section into four subsections. We initially introduce the utilized dataset and setup details of the conducted experiment in Section IV-A. Then, we evaluate the performance of our method with benchmark trackers under low/moderate/high detection distortion to simulate occlusion phenomena in Section IV-B. In the comparison, we run two of the recent benchmark tracker frameworks. In Section IV-C, we evaluate our method against recent learning and non-learning state-of-art methods. Finally, we conduct a running time comparison in Section IV-D.

A. Dataset and experimental setup

We conduct our experiment on the KITTI [32] [33] dataset that involves 21 streams with about 8,000 consecutive frames since it contains long stream duration of various urban scenarios while driving in Karlsruhe, Germany. We use LiDAR and camera information and calibration parameters provided in the dataset. We use the KITTI evaluation tool [34] for evaluation. To provide a transparent comparison with benchmark models, we pick two recent benchmarks, EagerMOT [5] and DeepFusion-MOT [4], that provide modifiable source code to integrate and run various detectors. We conduct the experiments on the following hardware scheme: *Processor: 11th Gen Intel Core i7-11370H 3.30GHz, GPU: NVIDIA Geforce RTX3070 Laptop GPU*. However, our framework only uses the CPU to perform tracking.

B. Evaluation under detection distortion

A high detection distortion means a detector frequently loses detection for objects due to poor detectors, so the distortion resembles object occlusion when we lose detection for objects in subsequent frames. In contrast, suitable detectors have minor detection distortions. In this experiment, we simulate occlusion phenomena by conducting detectors with different detection distortions: low, moderate, and high. We run the source code of two recent benchmarks, EagerMot [5] and DeepFusion-MOT [4], and use the KITTI evaluation toolkit [34] [33]. We involve 20 streams from the training and validation dataset. We exclude stream *0017* since we do the evaluation only for cars, and *0017* does not contain any car objects. Table I shows an overview results of the experiment. We can summarize it as follows:

Poor detector ***(High distortion)***: The **1st-row** in Table I shows an experiment using poor detection performance. We conduct YOLOv3 [27] for 2D detection and use calibration parameters to project point cloud to obtain 3D detection. Our tracker achieves 39.2% and 44.5% for $HOTA$ and $MOTA$, respectively, which is 3% higher than EagerMOT [5] and 12% higher than DeepFusion-MOT [4]. Our approach has a high association accuracy of 42.8%, with less id switching than other trackers. Since poor detection means inconsistent detection for objects, the results reflect tracker stability under inconsistent or disconnected detection. Our tracker outperforms the two benchmarks with inconsistent detection.

Moderate detector ***(Medium distortion)***: The **2nd-row** of Table I shows the tracking performance under a moderate performance detector. We conduct a similar setup as the previous experiment with little modifications by replacing YOLOv3 [27] with a more sophisticated detector, RCC [28]. We still obtain 3D detection by point cloud projection. The results show that DeepFusion-MOT [4] cannot perform well with poor 3D detection. In contrast, EagerMOT [5] perfor-

mance has improved by achieving 70.8% for $HOTA$ and 82.2% for $MOTA$. However, our tracker still outperforms EagerMOT [5] by nearly 11% higher in both $HOTA$ and $MOTA$ metrics. Our model also has less id switching, even with poor 3D detection, which is the opposite for both EagerMOT [5] and DeepFusion [4]. Finally, our model can achieve the highest $MOTA$ value, 91%, by conducting only one detector with 5% higher than other benchmarks.

Good detector ***(Low distortion)***: The **last four rows** in Table I present tracking results utilizing sophisticated 2D and 3D detectors. In this experiment, we switch between RCC [28] and TrackRCNN [31] 2D detectors, and PointGNN [30] and PorintRCNN [29] 3D detectors. Hence, we have four rows to represent all possible combinations. Our tracker generally has a better overall performance of 82.8% and 90.7% for $HOTA$ and $MOTA$, respectively. The tracker has superior performance under different combinations of 2D detectors. However, we notice that EagerMOT [5] has superior performance with TrackRCNN [31] for 2D detection and PointGNN [30] for 3D detection. The performance of EagerMOT [5] with PointGNN [30] is better than PointRCNN [29], which explains having an almost similar performance for EagerMOT [5] in the sixth and fourth rows where PointGNN [30] is placed.

In conclusion, our tracker maintains superior performance regardless of the performance of the assigned detectors. Besides, our model has stable id switching over the experiments, which is not applied for other benchmarks when the IDSW explodes with poor and moderate detector performance. However, our method has a higher IDSW than benchmarks with good detectors for 2D and 3D, which might be caused by noise obtained by holding a long-term memory in our framework to handle prolonged occlusion.

C. Evaluation with learning and non-learning benchmarks

Method	Detector	HOTA↑	MOTA↑	AMOTA↑
PC-TCNN [3]	3D	-	89.44%	-
FANTrack [35]	2D+3D	-	74.30%	40.03%
AB3DMOT [25]	3D	-	83.35%	44.26%
mmMOT [36]	2D+3D	-	74.07%	33.08%
GNN3DMOT [37]	2D+3D	-	84.70%	-
DeepFusion-MOT [4]	2D+3D	77.65%	88.33%	82.52%
EagerMOT [5]	2D+3D	78.53%	87.67%	45.62%
DFR-FastMOT (Our)	2D	83.4%	**93.06%**	**90.79%**
DFR-FastMOT (Our)	2D+3D	**84.28%**	91.96%	85.36%

TABLE II: A comparison with other state-of-art methods includes non-learning and learning-based trackers using the KITTI evaluation dataset.

We perform another evaluation with other benchmarks referring to claimed results on the evaluation dataset of KITTI and obtain tracking results of EagerMOT [5] and DeepFusion-MOT [4] frameworks by running them on our machine. The comparison involves recent learning and non-learning tracking approaches. Although the PC-TCNN [3] tracker achieves higher performance than other benchmarks as a learning-based approach, our tracker outperforms the benchmark models, including learning-based approaches, with a 3% margin in $MOTA$. As shown in Table II, the tracker achieves 93.06% $MOTA$ accuracy employing solely RCC 2D detector with point cloud projection. We notice that the tracker has consistent performance over the evaluation streams, allowing high $AMOTA$, 90.79%, dramatically higher than recent benchmarks. We achieve the highest $HOTA$, 84.28%, utilizing 2D RCC [28] and 3D PointRCNN [29] detectors.

D. Running-time comparison

To evaluate the running time, we fix 2D RCC [28] and 3D PointRCNN [29] detectors for all trackers, run each tracker individually on the same machine, and record time printed from the source code. We apply this experiment to all KITTI training and validation datasets to have 20 streams with 7,763 frames. We still exclude *0017* since it contains no car objects. As shown in Table III, Our tracker processes the dataset in 1.48 seconds, which is about seven times faster than other benchmarks. The main reason for having high speed is that the method uses an algebraic model, as explained in Section III-B, instead of traditional algorithmic approaches.

Detector	Method	Time(s.)
2D RCC [28]	**DFR-FastMOT (Our)**	**1.48**
+	EagerMOT [5]	11.47
3D PorintRCNN [29]	DeepFusion-MOT [4]	37.38

TABLE III: A running-time comparison for our tracker with recent benchmarks using KITTI training and evaluation datasets with 7,763 frames in total.

V. Conclusion

We present a light MOT method that relies on an algebraic model to fuse and associate objects detected by a camera and LiDAR sensors. Our experiments show that the algebraic formulation for association and fusion steps dramatically reduces the computational time of the MOT method. This advantage allows long-term memory expansion in MOT methods that eventually captures complex object occlusion scenarios and boosts the overall tracking performance. We simulate object occlusion phenomena using different detection distortion levels and show that our method outperforms two of the recent benchmarks under inconsistent detection. We also evaluate our solution against learning-based and non-learning methods and show that our method outperforms recent learning-based research work by a high margin, 3% in $MOTA$, and other methods by 4% utilizing a mono-detector that makes it applicable for mobile robots. The work will be available at https://github.com/MohamedNagyMostafa/DFR-FastMOT.

References

[1] Mohamed Chaabane, Peter Zhang, J Ross Beveridge, and Stephen O'Hara. Deft: Detection embeddings for tracking. *arXiv preprint arXiv:2102.02267*, 2021.

[2] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10860–10869, 2021.

[3] Hai Wu, Qing Li, Chenglu Wen, Xin Li, Xiaoliang Fan, and Cheng Wang. Tracklet proposal network for multi-object tracking on point clouds. In *IJCAI*, pages 1165–1171, 2021.

[4] Xiyang Wang, Chunyun Fu, Zhankun Li, Ying Lai, and Jiawei He. Deepfusionmot: A 3d multi-object tracking framework based on camera-lidar fusion with deep association. *arXiv preprint arXiv:2202.12100*, 2022.

[5] Aleksandr Kim, Aljoša Ošep, and Laura Leal-Taixé. Eagermot: 3d multi-object tracking via sensor fusion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11315–11321. IEEE, 2021.

[6] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[7] Bing Shuai, Andrew Berneshawi, Xinyu Li, Davide Modolo, and Joseph Tighe. Siammot: Siamese multi-object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12372–12382, 2021.

[8] Linyu Zheng, Ming Tang, Yingying Chen, Guibo Zhu, Jinqiao Wang, and Hanqing Lu. Improving multiple object tracking with single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2453–2462, 2021.

[9] Daniel Stadler and Jurgen Beyerer. Improving multiple pedestrian tracking by track management and occlusion handling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10958–10967, 2021.

[10] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[11] Yang Zhang, Hao Sheng, Yubin Wu, Shuai Wang, Wei Ke, and Zhang Xiong. Multiplex labeling graph for near-online tracking in crowded scenes. *IEEE Internet of Things Journal*, 7(9):7892–7902, 2020.

[12] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[13] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[14] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[15] Andreas Reich and Hans-Joachim Wuensche. Monocular 3d multi-object tracking with an ekf approach for long-term stable tracks. In *2021 IEEE 24th International Conference on Information Fusion (FUSION)*, pages 1–7. IEEE, 2021.

[16] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[17] Hai Wu, Wenkai Han, Chenglu Wen, Xin Li, and Cheng Wang. 3d multi-object tracking in point clouds based on prediction confidence-guided data association. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[18] Muyuan Wang and Xiaodong Wu. Multi-object tracking strategy of autonomous vehicle using modified unscented kalman filter and reference point switching. *Journal of Shanghai Jiaotong University (Science)*, 26:607–614, 10 2021.

[19] Taeklim Kim and Tae-Hyoung Park. Extended kalman filter (ekf) design for vehicle position tracking using reliability function of radar and lidar. *Sensors*, 20(15), 2020.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[22] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark, 2019.

[23] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[24] Yan-Jyun Ou, Xiang-Li Wang, Chien-Lung Huang, Jinn-Feng Jiang, Hung-Yuan Wei, and Kuei-Shu Hsu. Application and simulation of cooperative driving sense systems using prescan software. In *2017 International Conference on Information, Communication and Engineering (ICICE)*, pages 173–176, 2017.

[25] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366. IEEE, 2020.

[26] Wentao Bao, Qi Yu, and Yu Kong. Object-aware centroid voting for monocular 3d object detection. *CoRR*, abs/2007.09836, 2020.

[27] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[28] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 752–760, 2017.

[29] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[30] Weijing Shi and Ragunathan (Raj) Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[31] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. MOTS: Multi-object tracking and segmentation. In *CVPR*, 2019.

[32] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[33] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixe, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision (IJCV)*, 2020.

[34] Jonathon Luiten and Arne Hoffhues. Trackeval. https://github.com/JonathonLuiten/TrackEval, 2020.

[35] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1426–1433. IEEE, 2019.

[36] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2365–2374, 2019.

[37] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris Kitani. Gnn3dmot: Graph neural network for 3d multi-object tracking with multi-feature learning, 2020.