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

Towards Robust Reference System for Autonomous Driving: Rethinking 3D MOT

Leichen Wang[1], Jiadi Zhang[1 2], Pei Cai[1 3], Xinrun Li[1]

***Abstract*— With the rapid development of autonomous driving, the need for auto-labeling reference systems is becoming increasingly urgent. 3D multiple object tracking (MOT) is one of the most critical components of the reference system. In this work, we reviewed and rethought the common failure sources and limitations of the SOTA 3D MOT methods. We propose a set of innovative 3D MOT post-processing modules as a unified framework based on the observation. First, we design a self-learning-based detector to eliminate the outliers in each tracklet. Then a novel post-processing module, GGTrajRec, will recover the breakpoints and ID switches in the trajectories. Finally, a confidence-guided trajectory optimizer is implemented to ensure each trajectory's consistency.**

Extensive experiments on KITTI and nuScenes show that our method can improve the SOTA methods on most evaluation metrics by a remarkable margin. Currently, our results are second ranking on the KITTI tracking leaderboard. Specifically, our method offers the lowest FPs, highest DetRe, and AssRe values among all methods, which can significantly contribute to a stable and robust reference system for ADAS.

***Index Terms*— 3D MOT, Tracking Post-Processing**

I. INTRODUCTION

In recent years, autonomous driving (AD) and the Advanced Driver Assistance System (ADAS) have made significant progress using deep learning techniques. "*When it comes to autonomous driving, data is everything*", as Elon Musk said. The main bottleneck for improving ADAS is obtaining huge amounts of high-quality annotated data. On the one hand, the annotated data is incessantly needed for training supervised deep learning networks; on the other hand, all ADAS functions and algorithms must be evaluated and validated by "ground truth" before each release.

Most ground truth data is produced by differential global positioning systems(DGPS) or human annotation. However, the inflexibility of DGPS and the massive cost of manual annotation limit the acquisition of ground truth data. In order to reduce the cost of human operators and accelerate the annotation process, automatic and semi-automatic labeling tools have been developed in recent years as reference systems to cope with the shortage of ground-truth data. Several works investigated the problem of developing detection and tracking methods with high-performance but longer inference time. Based on the detection results, the objective of multiple object tracking (MOT) is to extract continuous dynamic information to obtain the total number of targets from the whole sequence of recordings [1]. Using consecutive frames,

[1]Robert Bosch CN firstname.lastname@cn.bosch.com
[2] TongJi University, China
[3] Nanyang Technological University, Singapore

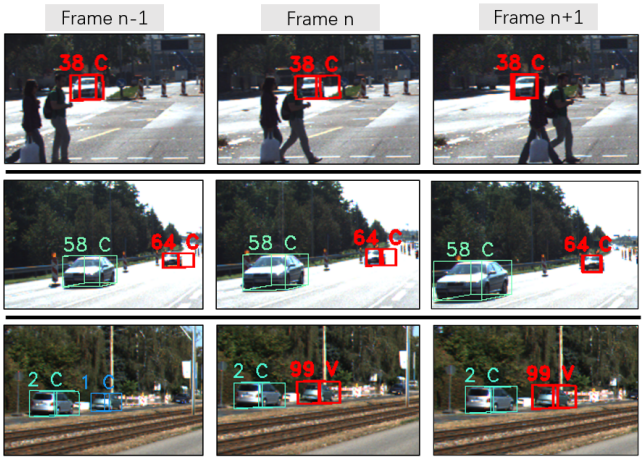

Fig. 1. Common issues of current SOTA 3D MOT method PC3T[2]. The number above the 2D BBox means ID, "C" means "Car", and "V" means "Van". In the *first* row: the red box (ID*38*) represents an outlier due to the part occlusion. In the *second* row: the target in the red box (ID *64*) shows the unstable and unreliable tracking performance in the early time steps of an approaching object. The first and second rows are mainly caused by the error angle estimation of the detection module. Such issues can not be solved by the existing filtering and smoothing algorithms because they rely significantly on state estimation. The *third* row indicates ID switch (ID changed from *1* to *99*) due to the short-term wrong classification.

MOT can further make the result more reliable than single-frame-based detection.

According to the trajectory generation methods, MOT algorithms are roughly classified into online and offline MOT. Online MOT receives current sensor data or detection results in each frame. Treating online MOT as a state estimation problem, the Kalman filter and the corresponding variations are widely used [3], [4], [5]. On the contrary, offline methods take overall detections in an entire sequence as input. Given global information, offline tracking can produce more reliable annotation than online methods. Currently, research about offline tracking mainly focuses on graph optimization [6], [7], [8], [9] and the probability-based forward-backward smoothing [10], [11], [12]. In addition, Rauch-Tung-Striebel smoother(RTS) is used in [10], which applies an extended RTS smoother to backward on measurements storing. Last, Labeled Multi-Bernoulli (LMB) Random Finite Set (RFS) is employed in [11] to design a track-before-detect (TBD) estimation.

However, although offline tracking has achieved a good performance, it is still not robust enough for valid perception evaluation. Since most research primarily formulates tracking tasks based on reliable detection results, the tracking errors introduced by false detection results are not addressed. Furthermore, they did not consider characteristics of 3D MOT

exhaustively. As shown in Fig.1, some problems still occur in the SOTA MOT framework, such as tracklet outliers, unstable tracking in the far range, and ID switches.

We posit that handling those issues is essential in 3D MOT. In this paper, we identify that the consistency of the object's attributes and the persistence of the real-world trajectories provide strong cues to refine the 3D MOT results. To fully use such information, we design a set of post-processing modules to fine-tune the tracking results. Our approach aims to fit into any offline system to obtain the highest accuracy.

Contributions.

Common failure sources and limitations of 3D MOT. We reviewed and rethought the common failure sources and limitations of current SOTA 3D MOT methods. To the best of our knowledge, this is the first work that points out the issue of tracklet outliers in the field of 3D MOT.

3D MOT post-processing framework. We propose a novel post-processing framework to obtain the highest possible accuracy annotation. Given tracklets and unassociated detection candidates of the whole sequence, our approach recovers and refines the missing or error parts of the trajectories, then produces the results in a plug-and-play manner. It consists of three parts: self-learning-based outlier detection, graph-based global trajectory recovery module (GGTrajRec), and confidence-based trajectory optimizer.

Evaluation. Comprehensive experiments on both KITTI [13] and nuScenes [14] show that our method outperforms the different SOTA baseline method by a large margin. We also provide a thorough study to show the influence of each component on performance.

II. Characteristics, common issues, and Limitations of 3D MOT

The MOT methods can be divided into 2D and 3D by the type of output data. We found that 3D MOT has the following unique characteristics. First, 3D MOT should maintain geometric consistency of the targets during a continuous dynamic environment. Second, all trajectories of 3D MOT in the global coordinate system are smooth and continuous, which can be predictable by a suitable motion model. For 2D MOT, these two points can not be guaranteed in an image coordinate system.

Based on the observation, we showed the typical common issues, and limitations of existing 3D MOT methods in Fig.1; each row represents one type of common issue. In the following part, we further describe each issue in detail.

1) Tracklet outlier: Tracklet outlier represents the erroneous element in the tracklet, which is caused by inaccurate or false detection. Most of the existing works tend to solve the problem by Kalman filter [15] or RTS based backward smoothing[11]. However, the inaccurate measurement integration at the initial moments will lead to relatively poor state estimation. One critical example is that head-to-tail reversal occurs by pointcloud-based angle estimation, *e.g.* confusing $0^{\circ}$with 180°. The abrupt and unexpected change in the angle domain makes Kalman filter ineffective.

2) Unreliable and unstable performance in the far range: As the distance between the target and the sensor grows, the perception resolution decreases significantly, thus making the confidence and performance of measurements from the target less reliable. Similarly to the tracklet outlier, the track states may be missing or erroneous in the early time steps of the track of an approaching object.

3) Identity (ID) switch: ID switch of a target is a typical tracking error. Due to the missing detection and wrong classification, an entire trajectory of an object will be divided into multiple pieces of tracklets. The occurrence of ID switch is influenced by the complexity of the scene and the number of the pre-defined detection categories. In general, ID switch occurs more frequently as the scene becomes more complex or the number of the pre-defined detection object increases.

III. Methodology

In this section, we introduce a unified 3D MOT post-processing framework. As shown in Fig.2, given tracklets and unassociated detections, our framework consists of the following parts: outlier detector, GGTrajRec, and confidence-based trajectory optimizer. We start by clarifying the problem statement in III-A and showing how the self-learning-based track outlier detector works in III-B. Then we discuss the individual modules GGTrajRec in III-C. The last subsection III-D indicates the final trajectory optimizer.

Thanks to the plug-in design, our framework is highly flexible and can be compatible with any 3D MOT method.

A. Problem Statement

Given a set of tracklets candidates $\mathcal{T}$ and unassociated detections $\mathcal{D}$ generated by any existing tracking frameworks, our goal is to refine the tracking results of the whole input sequence and to output a new set of trajectories $\mathcal{T}^*$. The elements x of the tracklets are given as a list of 3D BBoxs at each frame. An element $x_i \in \mathbb{R}^{10}$ is parameterized by its 3D position, orientation (heading angle θ), size (w,h,l), velocity (v_x, v_y), and the corresponding confidence score γ.

B. Outlier Detector

Based on the observed failure sources in II, we ask the following question: how to classify each element x_i of the given tracklets into in-distribution data x_i^{IN} and out of distribution data (OOD) x_i^{OOD}? To address this issue, we combined Self-Supervised outlier Detection (SSD)[16] and Contrastive Predictive Coding (CPC)[17] to design a novel self-supervised outlier detection framework. In our actual application, only unlabeled data is easily accessible since the auto-labeling system aims to provide the most accurate results without manual annotation. Thus, we propose an outlier detector based on the unlabeled input tracklets alone. As shown in part A of Fig.2, the key insight of our model is to classify OOD by maximizing mutual information between in-distribution detection x_i^{IN} and context feature of the corresponding tracklet c_i. Followed by CPC, we use a simple log-bilinear model to represent the mutual information $I(x_i, c_i)$, which is designed to indicate the probability of x_i to be

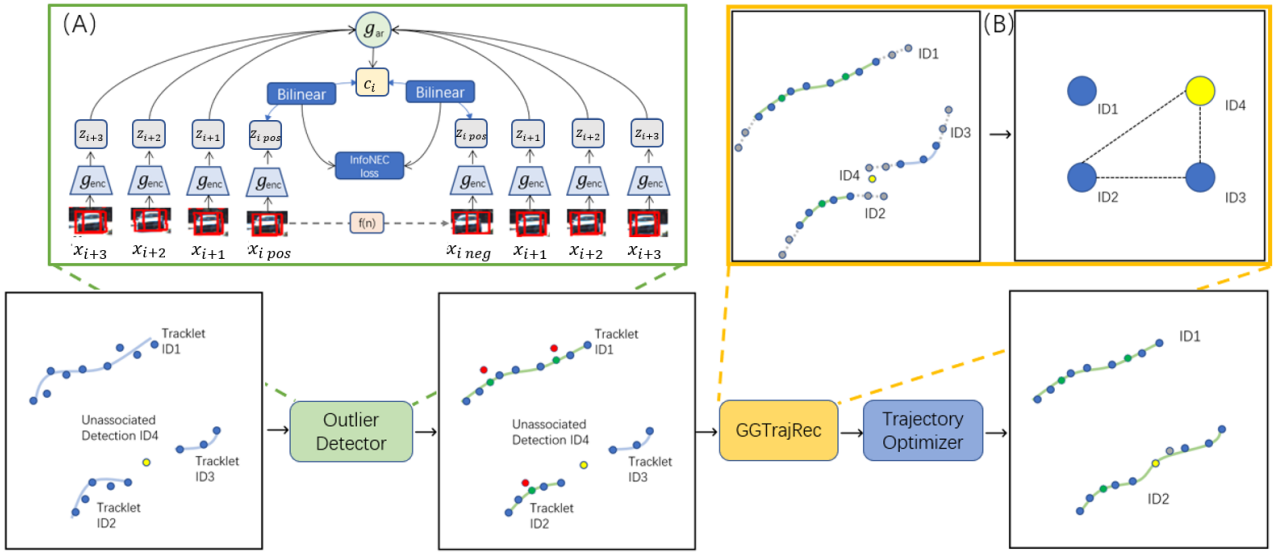

Fig. 2. **Overview of our proposed 3D MOT post-processing framework**, which is composed of three modules: *Outlier Detector* in III-B, *GGTrajRec* in III-C, and *confidence-guided tracklet optimizer* in III-D. The top left part (A) shows the structure of outlier detector. The top right part (B) shows the pipeline of GGTrajRec. In each sketch, blue points mean the elements in the given tracklets; yellow point represents the unassociated detection; red points mean the detected outliers; greens points indicate the smoothed elements to replace the outliers; gray points represent the prediction states of each tracklet by the motion model.

Algorithm 1: Self-supervised Tracklet Outlier Detector

Input: $\mathcal{T}_{input}$, $\mathcal{T}_{test}$, a non linear encoder g_{enc}, auto-regressive model g_{ar}, noise distribution function $f(n)$, mutual information model I
Output: classify x_i into x_i^{IN} or x_i^{OOD} $\forall x \in t, \forall t \in \mathcal{T}_{test}$
Filter $\mathcal{T}_{input}$ into the most confident set $\mathcal{T}_{training}$;
Encoding Phase
for t *in* $\mathcal{T}_{training}$ **do**
  for *each element* x_i *in* t **do**
    positive instance: $z_i = g_{enc}(x_i)$;
    negative instance: $\hat{z}_i = g_{enc}(f_n(x_i))$;
  end
  encode the context feature c_i of the fixed-length latent representation $[z_{i-n}, ..., z_{i+n}]$;
end
Training Phase
$\mathcal{L}_N = -\mathbb{E}[log \frac{I(x_i, c_i)}{\sum_{x_j \in X} I(f_n(x_j)), c_i)}]$;
Training by minimizing $\mathcal{L}_N$ over $\mathcal{T}_{training}$;
Adaptively adjust the threshold ζ of outlier score;
Test Phase
for t *in* $\mathcal{T}_{test}$ **do**
  for *each element* x_i *in* t **do**
    $x_i \in \mathcal{T}_{test}$ is x_i^{OOD} if $I(c_i, z_i) < \zeta$.
  end
end

positive. To achieve it, the input x_i will be encoded into z_i and compared with such latent representations c_i by predicting the in-distribution state at the current time stamp using powerful autoregressive models g_{ar}. In other word, g_{ar} is designed to summarize all neighboring z in the latent space and produces a predicted value at time i. Thus, the mutual information can be represented as the inner product between the predictive value and the given value z_i.

Our outlier detection consists of the following steps: 1) For each element of the selected tracklets, a sequence of latent representations z_i can be encoded from a sequence of detections x_i as *positive* instance by a non-linear encoder g_{enc}. Then an autoregressive model g_{ar} maps a tracklet of fixed length, $[z_{i-n}, ..., z_{i-1}, z_{i+1}..., z_{i+n}]$ into the high-level representation (or slow context feature) c_i. 2) Using a predefined noise distribution function f_n, each element can be converted into an artificial outlier $f_n(x_i)$, commonly referred to as *negative* instance. 3) In the training phase, the function is optimized to minimize the InfoNEC loss $\mathcal{L}_N$ [17] 4) In the test phase, we can further calculate the mutual information between an element and the rest tracklet $I(x_i, c_i)$. If the mutual information value is less than a given threshold, the element will be classified as an outlier, and replaced by interpolating of the previous and subsequent frames.

However, strictly speaking, our method is an *inaccurate* self-supervision method because there is no pure in-distribution data. The input training set also contains negative instances due to false detections. To maximize the validity of positive instances, we add a filter before the encoding phase. It is supposed to find the most confident tracklets from the given ones according to a percentage confidence threshold. We set the default threshold as the top 30% most confident tracklets. For more details, please refer to Algo.1 and IV-B.

C. Graph-based Global Trajectory Recovery Module

In this subsection, we introduce a novel Graph-based Global Trajectory Recovery Module (GGTrajRec), which is designed to recover the breakpoints and ID switches in trajectories in an end-to-end manner. The initial intuitions are from TrackletNet[18] and lifted disjoint paths[8], which regard tracklets as the vertices in the graph with the edges measuring the similarities between tracklets. They aim to avoid being stuck at a bad local minimum solution by graph optimization. However, such approaches have two shortcomings. First, these works omit the information of the unassociated detections which could also be utilized to recover the missing part of the whole real trajectories. Thus, unlike [18] and [8], we use v to denote vertices that

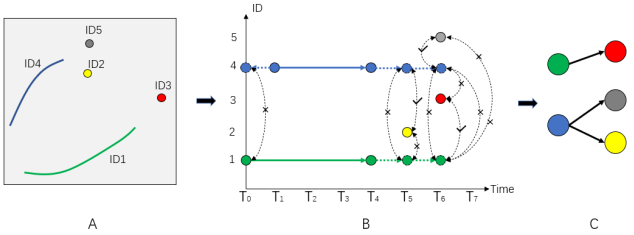

Fig. 3. The pipeline of graph construction. *Left (A):* Input tracklets and unassociated detections from the previous outlier detector module III-B. *Middle (B):* Schematic diagram for searching possible matching pairs. Check mark represents a possible matching pair, while cross mark means inpossible matching due to either overlapped in time domain or too large pairwise cost. *Right (C):* Graph model for tracklets clustering.

correspond to both tracklets $\mathcal{T}$ and unassociated detections $\mathcal{D}$. The second shortcoming is that they model the similarity between two tracklets without considering the cost of false classification. In this context, we design a simple but effective schematic diagram to find all possible matching pairs for 3D MOT problems based on forward-backward prediction and pre-defined pairwise cost.

1) Graph Construction: As shown in Fig.3, the graph construction consists of the following steps: First, we generate the schematic diagram to search for all possible matching pairs. As shown in Fig.3 part B, X axis is the timestamp, Y axis means the target ID. Each tracklet will be mapped into *start tracklet node* $\mathcal{S}_{id}^{ts}$ and *end tracklet node* $\mathcal{E}_{id}^{ts}$; each unassociated detection will be mapped as *detection node* $\mathcal{D}_{id}^{ts}$, where ts means the timestamp. Then, for each tracklet, we predict n steps of possible current states for each tracklet at the discrete-time by motion model *e.g.* constant velocity (CV). In other words, $\mathcal{E}_{id}^{ts}$ will be extended to n steps as the nodes $\hat{\mathcal{E}}_{id}^{ts+1}...\hat{\mathcal{E}}_{id}^{ts+n}$ by forwarding CV motion model; the historical state estimation $\hat{\mathcal{S}}_{id}^{ts-1}...\hat{\mathcal{S}}_{id}^{ts-n}$ will be recovered before the $\mathcal{S}_{id}^{ts}$ by backward CV motion model, refer to Fig.2 part B.

After that, we iterate through each predicted node, match each predicted node to the other if the id are different and ts are the same. The reason is that one object can not appear in different tracklets simultaneously. In other word, the matching pair can not be overlapped in the time domain. Besides, there are two strict restrictions for the possible matching. First, the matching is not considered if the pairwise cost of two nodes is too large since their relationship is too weak to link. Second, if two IDs have multiple matching pairs, we only take the pair with the lowest pair cost. It is because any two vertices are allowed to have only one edge in the graph model. Based on the previous steps, the problem can be naturally formalized by a graph $G = (V, E)$, see part C of Fig.3. Its vertices set V represents a finite set of $\mathcal{T}$ as tracklets and $\mathcal{D}$ as unassociated detections; its edge set E represents a series of edges of possible matching pairs.

2) Pairwise Cost: Our proposed pairwise cost represents the correlations between different tracklets and unassociated detections, which is formulated between i-th element and j-th element as $\mathcal{PC}^{i,j} = \mathcal{G}^{i,j} + \mathcal{M}^{i,j} + \mathcal{C}^{i,j}$, $\mathcal{G}$ represents geometry cost, $\mathcal{C}$ represents classification cost, and $\mathcal{M}$ for motion cost.

Inspired by [2], the geometry cost $\mathcal{G}$ is defined as:

$$\mathcal{G}^{i,j} = \omega_1 \cdot \mathcal{N}\Big(\sum_{k\in\{h,w,l\}} \frac{\big|k^i - k^j\big| \cdot (k^i + k^j)}{k^i \cdot k^j}\Big) + w_2 \cdot \mathcal{N}\big(\big|d^i - d^j\big|\big) + w_3 \cdot \mathcal{N}(1 - cos(\theta^i - \theta^j)) \quad (1)$$

, where w_1, w_2, w_3 are the importance weights, $\mathcal{N}$ denotes the normalization function, d is the target distance from the global origin, h w l θ denote the height, width, length, and orientation angle, respectively. The motion cost $\mathcal{M}$ consists of velocity orientation similarity and velocity measurement similarity, as follows:

$$\mathcal{M}^{i,j} = \omega_4 \cdot \mathcal{N}\Big(1 - \frac{v^i \cdot v^j}{\|v^i\| \cdot \|v^j\|}\Big) + w_5 \cdot \mathcal{N}\big(\big|v^i - v^j\big|\big) \quad (2)$$

, where w_4 and w_5 are the importance weights. Finally, the classification cost $C^{i,j}$ is formulated by a pre-defined cost matrix of size $n \times n$, which indicates the similarity between n different categories. For example, the cost between "*Pedestrian*" and "*Cyclist*" is much lower than "*Pedestrian*" and "*Truck*".

3) Graph Optimization: After the graph is built, the complete trajectories will be generated by clustering into different sub-graphs. The optimization process can be described as follows: given a graph $\mathcal{G}(V, E)$, let u and w be the possible matching vertices connected by the edge e, whose cost is defined as $-log(\mathcal{PC}_e)$. Let $\pi_e = 1$ when u and w are clustered together; $\pi_e = 0$ when u and w are clustered in the distinct tracks. The clustering cost of edge e can be further represented as $\pi_e \cdot (-log(\mathcal{PC}_e))$. We define the graph optimization problem as to minimize the total clustering cost on all edges as: $min \sum_{e\in E} \pi_e \cdot (-log(\mathcal{PC}_e))$. Thus, the graph optimization is formulated as a clustering problem, which is APX-hard. Followed by [18], we can achieve the global minimum solution by five clustering operations, i.e. assignment, merging, splitting, switching, and breaking.

D. Confidence-guided Trajectory Optimizer

Based on the characteristic of 3D MOT, we hypothesize that the size of the 3D object in one trajectory is changeless. Thus we propose a confidence-guided method to assign the most confident attribute to all elements in one trajectory, i.e. some attributes will be replaced by a confidence-score-based weighted average. The equation is as follows: $a^* = \frac{\sum_{j=1}^{n} a_i \cdot \gamma_i}{n}$, where a means the attributes of objects that will be optimized, for example height h, width w and length l. We set the confidence score of each element as $\gamma_i = \gamma_i^{det} \cdot \eta_{det} \cdot \gamma_i^{dis} \cdot \eta_{dis}$, where γ_i^{det} is confidence score from detection result, γ_i^{dis} is function of distance between the ego vehicle and target object, η_{det} and η_{dis} are corresponding coefficients.

For each element x_i of a trajectory, the track size will be corrected by all the other element x_j of the whole track t. As an end result, 3D dimensions of each trajectory will be constant in every step. Furthermore, the optimization attributes can be further extended for some special cases. As an example, for parked vehicles, not only 3D dimensions

but also 3D location and angle can be fixed. Note that, it is not suitable for the extra short trajectories since the data in short trajectories is not robust.

IV. Experimental setup

A. Dataset and Evaluation Metrics

We extensively evaluate the proposed method on the two widely used large-scale datasets: KITTI[13] and nuScenes[14]. The KITTI tracking challenge consists of recordings for 21 training sequences and 29 test sequences. It is *allowed* to implement the technique of post-processing in the KITTI tracking challenge, *e.g.* PC3T[2] used the global feature to fine-tune the trajectories. To compare with the other SOTA methods on the KITTI tracking leaderboard, we follow the official tracking metrics of the KITTI tracking benchmark: Higher Order Tracking Accuracy (HOTA). Considering the motivation of a robust reference system, we mainly focus on the evaluation metrics DetRe, AssRe, and FP. Note that *it is forbidden* to implement post-processing in the nuScenes tracking challenge, so the evaluation of nuScenes is only limited to the validation set. For the nuScenes tracking challenge, the primary metrics for evaluating MOT are Average Multi-Object Tracking Accuracy (MOTA) and Average Multi-Object Tracking Precision (AMOTP). For more details, please refer to [14].

B. Implementation Details

Our outlier detection method is implemented in Pytorch for all benchmarks and it is trained and evaluated on an Ubuntu PC. g_{enc} is a simple encoder that runs directly on each element of tracklets. We use 1D-convolution layers and ReLU activations. Next, we use a GRU[27] based autoregressive model g_{ar} to learn the latent representation c with the contrastive loss. To generate artificial negative instances, we leverage the noise distribution function $f(n)$ from data augmentation techniques [28]. It consists of translation factor $\Delta x, \Delta y, \Delta z$, rotation factor $\Delta\theta$, and scaling factor s. $\Delta x, \Delta y, \Delta z$ are from a normal distribution $n(1, \sigma^2)$, where $\sigma^2 \in \{1, 2, 3\}$; $\Delta\theta$ is from a uniform distribution $U(-\beta, +\beta)$, where $\beta \in \{\pi/4, \pi/2, \pi\}$; s is from a uniform distribution $U(1-\omega, 1+\omega)$, where $\omega \in \{0.1, 0.2, 0.3\}$. We used Adam optimizer with a batch size of 8 and a learning rate of 1e-4. The model is trained until convergence at about 400,000 updates. For GGTrajRec, we predict *two* steps in both forward and backward directions for each tracklet.

Since we aim to propose a post-processing framework to fit any existing 3D MOT work, we choose the top open-source methods on each leaderboard as the baseline. For KITTI tracking challenge, we compare our results to the top open-source work PC3T. For a fair comparison, our implementation is based on the officially released code of PC3T[1]. Since data sequences of KITTI test set are not fixed, we first divide the extra long sequence into short subsequences of each 60 frames, then merge the results of each subsequence. The selection of hyperparameters is based on the highest HOTA scores found in the KITTI validation set by grid searching. As a result, we set $\{0.45, 0.45, 0.1, 0.7, 0.3\}$ for $\{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}$ and $\{0.6, 0.4\}$ for $\{\eta_{det}, \eta_{dis}\}$, respectively. For nuScenes, we select the top two open-source tracking methods, SimpleTrack[29] and CBMOT[30] as baselines to perform the evaluation. The selection of hyperparameters is based on the highest AMOTA scores in the nuScenes validation set. We set $\{0.5, 0.2, 0.3, 0.7, 0.3\}$ for $\{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}$ and $\{0.7, 0.3\}$ for $\{\eta_{det}, \eta_{dis}\}$.

V. Results

A. Main Results

Result on KITTI Given the tracking results from baseline method PC3T, our method ranks second on KITTI's official leaderboard[2] as shown in Table I. Compared with the baseline method, our method surpassed 2.59% as HOTA, 5.04% as DetRe, and 2.94% as AssRe. Specifically, our method offers the lowest FPs, highest DetRe and AssRe values among all methods. In terms of the critical evaluation metrics of a reference system, our method shows considerable performance superiority.

Result on nuScenes We evaluate our approach on the nuScenes validation set. The results are presented in Table II. Again, due to the challenging regulation of *online* tracking approach and to make a fair comparison, we only aim to show the benefit of our framework instead of achieving the top performance on the nuScenes tracking leaderboard. Compared with SimpleTrack and CBMOT, the AMOTA scores are improved to 2.0 and 3.2, respectively.

B. Qualitative Analysis

As a qualitative analysis, Fig. 4 shows an example of representative results. The middle column shows the original input tracking result from PC3T; the tracked object deviated from its ground truth trajectory due to occlusions in the movement process. However, our method allows for more robust and stable tracking without occlusion effects.

C. Ablation Studies

1) Effectiveness of sub-component: To investigate the effectiveness of each component, we turn off the components step by step in Table III. When the outlier detector turns on, we observe that the system gives a high HOTA result (+1.7). Conversely, the improvement caused by GGTrajRec (+0.0) and trajectory optimizer (+0.1) is relatively insignificant when the outlier detector turns off. From this observation, we can conclude that using GGTrajRec or trajectory optimizer alone does not work. The reason is that when tracklets still contain many outliers, the other two modules can not work effectively. Thus, eliminating the outliers affects the subsequent post-processing modules.

[1]https://github.com/hailanyi/3D-Multi-Object-Tracker

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

Method	Sensor	Setting	HOTA↑	DetRe↑	DetPr↑	AssRe↑	AssPr↑	MOTA↑	FP↓
TripleTrack[19]	C/L	3D	73.58	76.18%	86.81%	77.31%	89.95%	84.32	4642
DEFT[20]	L	3D	74.32	79.96%	83.97%	78.30%	85.19%	88.38	2647
EagerMOT[15]	C/L	3D	74.39	78.77%	86.42%	76.24%	91.05%	87.82	3497
DF-MOT[21]	C/L	3D	75.56	75.34%	85.25%	78.30%	85.19%	84.63	4601
Mono-3D[22]	C/L	3D	75.47	78.86%	82.98%	80.23%	88.88%	88.48	2754
OC-SORT[23]	C	2D	76.54	80.64%	86.36%	80.33%	87.17%	90.28	2685
PermaTrack[24]	C	3D	78.03	81.71%	**86.54%**	81.14%	89.49%	91.33	2320
PC3T*[2]	L	3D	77.80	79.19%	84.07%	84.77%	88.75%	88.81	2810
RAM[25]	L	3D	79.53	82.54%	86.33%	84.21%	88.77%	90.61	2094
PC-TCNN[26]	L	3D	**80.90**	84.22%	84.58%	87.46%	**90.47%**	**91.70**	1482
Ours*	L	3D	80.39	**84.23%**	83.57%	**87.63%**	88.90%	91.53	**1298**
Delta	L	3D	+2.59	+5.04%	-0.50%	+2.86%	+0.15%	+2.72	-1602

TABLE I

A comparison of existing algorithms on the KITTI tracking benchmark test set. Methods using the same tracking framework are labeled by "∗", and "Delta" is the comparison between our method and original PC3T. The ↑ symbol indicates that bigger value means better performance.

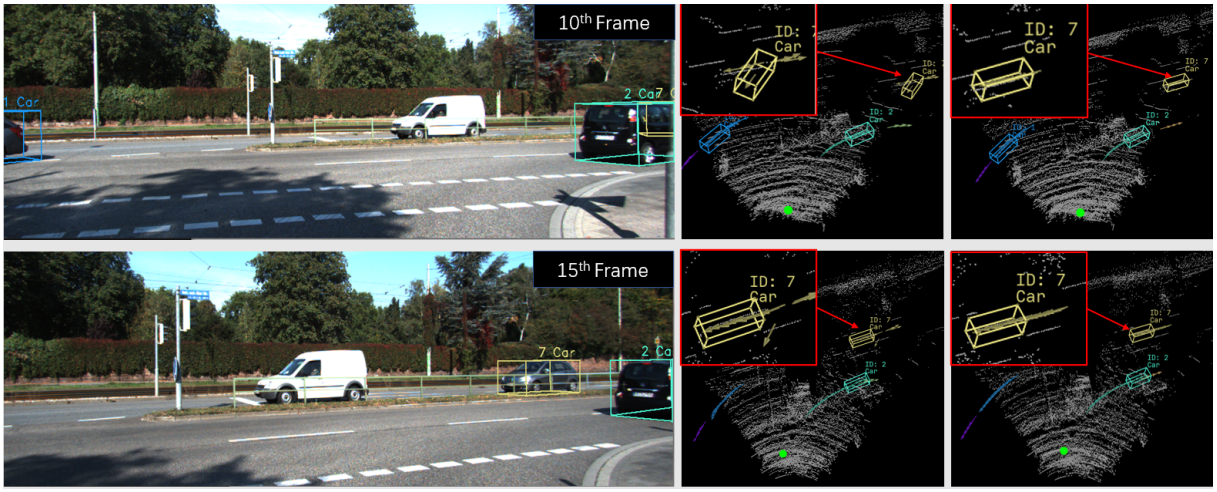

Fig. 4. Qualitative Analysis on KITTI val dataset, scene 0004. The upper row indicates the 10^{th} frame while the bottom shows the 15^{th}. The first column shows the tracked objects on the image plane. The second column shows the original tracking result of the baseline tracking framework PC3T; the third column shows our post-processed result. The 3D BBoxs of the current frame are coloured according to the different object IDs. All historical trajectories are denoted by arrows. We manually highlight the original outlier due to the wrong detection by occlusion ID *7* in the red box. Please zoom in for details.

Methods	AMOTA↑	AMOTP(m)↓
SimpleT[29]	69.6	0.547
+Ours	71.6	0.461
Delta	+2.0	-0.086
CBMOT [30]	69.2	0.563
+Ours	72.4	0.430
Delta	+3.2	-0.133

TABLE II

Results on the val set of nuScenes tracking dataset.

2) Analysis of Impact on different offline Trackers: Since our method leverage the global information, we raise a question: Does the post-processing still work for offline 3D MOT? We followed the official repo of PC3T and the TBD alogrithms from [11]. Benefiting from our framework, both approaches produce HOTA values of +2.4% and +1.9%, respectively, as shown in Table IV. The results are clear that 3D offline MOT frameworks can benefit from our method.

TABLE III

Result on KITTI val set to show the effectiveness of components. *OD* means outlier detector, *GG* means GGTrajRec, *TO* means trajectory optimizer.

OD	GG	TO	HOTA
×	×	×	80.1
√	×	×	81.8
×	√	×	80.1
×	×	√	80.2
√	√	×	82.1
√	×	√	81.9
×	√	√	80.3
√	√	√	82.5

TABLE IV

Ablation Studies on different 3D offline MOT methods on KITTI val set. *PC3T off* means the offline tracking result of PC3T; *PC3T+TBD* means adopting the TBD algorithm [11] on PC3T framework.

Methods	HOTA
PC3T off	80.1
+Ours	82.5
Delta	+2.4
PC3T+TBD	79.9
+Ours	81.8
Delta	+1.9

VI. CONCLUSIONS

We reviewed and rethought the current 3D MOT's limitations and common failure sources. To address these summarized problems, we propose a set of innovative and effective tracking post-processing modules as a unified framework.

Extensive experiments on KITTI and nuScenes show that our method can improve the SOTA methods by a large margin. Currently, our results are second ranking on the KITTI tracking leaderboard. Specifically, our method offers the lowest FPs, highest DetRe, and AssRe values among all methods, which can significantly contribute to a stable and robust reference system for ADAS. In the future, the proposed framework will be integrated into a semi-automatic annotation platform that can produce refined, accurate reference data.

REFERENCES

[1] Imran Ahmed, Sadia Din, Gwanggil Jeon, Francesco Piccialli, and Giancarlo Fortino. Towards collaborative robotics in top view surveillance: A framework for multiple object tracking by detection using deep learning. *IEEE/CAA Journal of Automatica Sinica*, 8(7):1253–1270, 2021.

[2] 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.

[3] Guojun Wang, Jian Wu, Rui He, and Shun Yang. A point cloud-based robust road curb detection and tracking method. *Ieee Access*, 7:24611–24625, 2019.

[4] Jacopo Pegoraro and Michele Rossi. Real-time people tracking and identification from sparse mm-wave radar point-clouds. *IEEE Access*, 9:78504–78520, 2021.

[5] Weijie Mei, Guangming Xiong, Jianwei Gong, Zhai Yong, Huiyan Chen, and Huijun Di. Multiple moving target tracking with hypothesis trajectory model for autonomous vehicles. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[6] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[7] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

[8] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking. In *International conference on machine learning*, pages 4364–4375. PMLR, 2020.

[9] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6247–6257, 2020.

[10] Egon Ye and Matthias Althoff. Model-based offline vehicle tracking in automotive applications using a precise 3d model. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1128–1135. IEEE, 2019.

[11] Boqian Yu and Egon Ye. Track-before-detect labeled multi-bernoulli smoothing for multiple extended objects. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2020.

[12] Egon Ye, Gerald Würsching, Sascha Steyer, and Matthias Althoff. Offline dynamic grid generation for automotive environment perception using temporal inference methods. *IEEE Robotics and Automation Letters*, 6(3):5501–5508, 2021.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[14] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[15] Aleksandr Kim, Aljoša Osep, and Laura Leal-Taix'e. Eagermot: 3d multi-object tracking via sensor fusion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[16] Vikash Sehwag, Mung Chiang, and Prateek Mittal. Ssd: A unified framework for self-supervised outlier detection. *arXiv preprint arXiv:2103.12051*, 2021.

[17] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[18] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 482–490, 2019.

[19] Nicola Marinello, Marc Proesmans, and Luc Van Gool. Triplettrack: 3d object tracking using triplet embeddings and lstm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4500–4510, June 2022.

[20] Mohamed Chaabane, Peter Zhang, Ross Beveridge, and Stephen O'Hara. Deft: Detection embeddings for tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021.

[21] 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. *IEEE Robotics and Automation Letters*, pages 1–8, 2022.

[22] 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, 2021.

[23] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking, 2022.

[24] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. In *ICCV*, 2021.

[25] Pavel Tokmakov, Allan Jabri, Jie Li, and Adrien Gaidon. Object permanence emerges in a random walk along memory. In *ICML*, 2022.

[26] 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*, 2021.

[27] Rui Fu, Zuo Zhang, and Li Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328. IEEE, 2016.

[28] Martin Hahner, Dengxin Dai, Alexander Liniger, and Luc Van Gool. Quantifying data augmentation for lidar based 3d object detection. *arXiv preprint arXiv:2004.01643*, 2020.

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

[30] Nuri Benbarka, Jona Schröder, and Andreas Zell. Score refinement for confidence-based 3d multi-object tracking. *arXiv preprint arXiv:2107.04327*, 2021.