Measurement 256 (2025) 117864

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

A dynamic-confidence 3D multi-object tracking method based on spatio-temporal association

Ruihao Zeng, Mohsen Ramezani *

The University of Sydney, School of Civil Engineering, Sydney, NSW, 2006, Australia

ARTICLE INFO

Dataset link: KITTI official website, nuScenes official website

Keywords:
LiDAR
Data association
Spatiotemporal feature
Dynamic prediction confidence
Point cloud

ABSTRACT

3D Multi-object tracking (MOT) is a crucial challenge for decision-making in autonomous driving. In this article, we propose an online 3D MOT method based on point cloud data to enhance the accuracy of tracking objects in complex scenarios while the sensor is moving. The method employs a constant acceleration model and incorporates orientation angle variation. After undergoing Kalman filter smoothing, the method can effectively and accurately estimate the future states of target objects, mitigating the issue of direction oscillation in the object detection stage. To tackle the bidirectional pairing problem between predicted targets and detection candidates, we introduce a novel spatio-temporal feature-based data association model. This model leverages a dynamic confidence threshold to address tracking temporarily occluded objects. Moreover, this model can effectively formulate affinity even amidst the noisy and confusing data typically generated by mobile sensors. Through extensive evaluations of the KITTI dataset, our method surpasses state-of-the-art methods. The method performance is further validated on the nuScenes dataset, solidifying the robustness and effectiveness of the proposed approach.

1. Introduction

Multi-object tracking (MOT), also referred to as multi-target tracking, is a key technique for understanding the surrounding environment using sensors. The primary goals of MOT are to detect and localize multiple objects in a given scene, distinguish their unique identities, and generate precise tracking trajectories. The objects being tracked can vary widely, from vehicles and pedestrians to specific equipment, animal groups, or even large-scale environments. MOT plays a critical role in applications such as connected and autonomous vehicles (CAVs) [1] and intelligent transportation systems. As an intermediate task in these systems, MOT provides essential data for downstream processes such as environmental perception, behavior analysis, and intelligent decision-making.

MOT is generally divided into two main categories: 2D MOT and 3D MOT. 2D MOT primarily relies on information extracted from two-dimensional spatial images, such as the OC-SORT method proposed by [2]. In contrast, 3D MOT incorporates depth information to generate three-dimensional bounding boxes for target objects and assigns them distinct identities for differentiation. Certain sensors, such as stereo cameras and LiDAR, along with images captured from various perspectives, like bird's-eye views (BEV), provide the necessary data for this type of tracking. Of these, LiDAR has garnered considerable attention in both research and practical applications due to its high positioning accuracy and robust performance in challenging conditions, such as low-light environments or adverse weather like thunderstorms. Notable projects, including Waymo [3] and Argoverse [4], highlight the preference for LiDAR in these scenarios.

Mobile MOT in CAVs introduces significant challenges compared to static tracking in Autonomy through Infrastructure (ATI) [5], particularly due to the dynamic nature of both sensors and target objects in real-world applications. Such mobility often leads to variations in scale, perspective, and additional noise, making it difficult to transform coordinate measurements into a unified global system. This increased complexity also intensifies the challenge of identifying consistent features across different time frames, potentially leading to failed object associations, tracking interruptions, or frequent switching among tracked objects. The dynamic states of objects (e.g., velocity and acceleration) are typically derived from historical trajectories. As changes in the states of surrounding objects are key factors for decision-making, accurate tracking is essential in CAV systems [6]. Furthermore, moving sensors are more susceptible to occlusion, which can cause temporarily obscured objects to be re-identified as new entities. Such misidentifications can critically undermine the understanding or prediction of objects' behavior. Addressing these challenges is crucial for improving the reliability and robustness of mobile MOT solutions,

* Corresponding author.
E-mail addresses: ruihao.zeng@sydney.edu.au (R. Zeng), mohsen.ramezani@sydney.edu.au (M. Ramezani).

https://doi.org/10.1016/j.measurement.2025.117864
Received 7 October 2024; Received in revised form 15 March 2025; Accepted 14 May 2025
Available online 31 May 2025

ensuring their effectiveness in meeting the demands of real-world CAV applications [7].

Many studies have attempted to overcome the challenges of 3D MOT by exploring the potential information within limited point clouds. For instance, [8,9] employ various distance measurement metrics to determine the associations between objects. Similarly, [10–12] emphasize that the same object should exhibit consistent features across different time steps, leveraging deep neural networks to extract feature information from bounding boxes and their internal point clouds. Furthermore, some approaches [13–16] focus on trajectory smoothness, examining the short-term and long-term associations between an object's current state and its trajectory to achieve accurate tracking. However, this field still faces numerous challenges, not only due to the limitations of the data itself but also influenced by the complexity of the application scenarios. We can take nuScenes [17], one of the largest MOT datasets that includes both vision (2D) tracking and LiDAR (3D) tracking for the same scenarios, as an example to summarize the research gaps in 3D MOT.

1. **Limited object information in point cloud.** Point cloud data primarily captures the surface characteristics of reflected objects. While it enhances the spatial structure and positional information of object surfaces compared to 2D images, it lacks many semantic details such as color, material, and shape. In the nuScenes dataset, the average number of point clouds within each 3D bounding box is approximately 67, significantly fewer than the number of pixels in a 2D bounding box. Moreover, the precise reflection properties of point clouds can introduce additional noise, and changes in detection angles can lead to unpredictable spatial relationships in point cloud distributions. According to the nuScenes leaderboard, the average AMOTP (Average Multi-Object Tracking Precision), a key performance metric where higher values indicate better tracking accuracy, is around 68.04% for 3D MOT, which is substantially lower than the 98.99% achieved in 2D MOT.
2. **Complex and dynamic scenes.** In the nuScenes dataset, within an average tracking range of 45 m, approximately 35 bounding boxes are contained per frame. In such environments, the trajectories of tracked objects can become unpredictable due to the complexity of the surroundings. Furthermore, when multiple objects are in close proximity, accurately associating the same objects across different time steps becomes a significant challenge for the tracker.
3. **Maintaining continuous tracking of occluded and temporarily missing objects.** In the leaderboard, the average IDS (Identity Switches, where a smaller value indicates more stable tracking) for 3D MOT is 2074, which is larger than the average IDS of 1420 for 2D MOT. Addressing this challenge requires the integration of a detector with robust object recognition capabilities, especially for occluded objects. Additionally, it tests the tracker's ability to maintain consistent tracking, even when objects temporarily disappear from the field of view.
4. **Low tracking efficiency.** This challenge is often accompanied by specific requirements for input data. While sparse point cloud data may lack certain object details compared to 2D image data, it can, in some cases, provide more efficient tracking performance.

To address these problems, we propose a LiDAR-based dynamic-confidence 3D MOT method using point cloud data that relies on spatio-temporal association. Our proposed method achieves more accurate and stable tracking results compared to state-of-the-art methods, while maintaining a computational efficiency of nearly 300 FPS (where most methods operate at 100–200 FPS). Moreover, our method demonstrates outstanding performance across various tracking tasks involving different categories in multiple datasets. The proposed method offers the following key contributions:

1. The proposed method is designed to achieve multi-object tracking at high frame rates without the need for GPU resources, expanding its applicability across a wider range of scenarios.
2. The proposed method introduces a novel object association model, calculating spatio-temporal correlations between object pairs based on four factors: geometry, velocity, historical features, and spatial distribution. This enables accurate object association even in complex scenarios using only point cloud data.
3. The proposed method enhances continuous object tracking by accounting for angle variations and dynamically adjusting association confidence.
4. The proposed method is tested on two datasets across multiple object categories to validate its accuracy, efficiency, and robustness.

This paper is structured as follows. Section 2 examines the related work in three different categories of the MOT field from the perspective of input data. Section 3 clarifies the foundational concepts of the problem and defines the states essential for the method. Section 4 provides explanations for each process. Subsequently, Section 5 presents comprehensive descriptions of the experiments, while conclusions are drawn in Section 6. The nomenclature is presented in Appendix A.

2. Related work

According to the classification method proposed in [18], MOT approaches can be categorized based on several criteria: tracking dependency (detection-free or detection-based), processing mode (online or offline), output data type (stochastic or deterministic), and input data type (2D or 3D). Detection-free tracking eliminates reliance on detection results by manually specifying the objects to be tracked within each frame, as opposed to detection-based methods. This approach is commonly utilized in 2D object tracking applications, such as the TrackAnything framework [19]. The online method refers to a tracker that processes only the available historical data, whereas an offline method enables the tracker to utilize the entire data sequence for tracking purposes [18]. The stochastic method describes a tracker whose results may vary for the same data sequence due to the probabilistic nature of neural networks. Conversely, the deterministic method ensures that the tracker produces consistent and unique results for each execution.

As MOT is categorized into three distinct types based on the different input data types, Table 1 highlights some of the most notable advantages and disadvantages of each type. A major focus of most studies is on how to preserve the strengths of each category while addressing its weaknesses.

2.1. 2D MOT

2D MOT is predominantly applied in pedestrian tracking [20]. With advancements in deep learning, estimating the distance from the sensor to the target object using images from a monocular camera has become feasible. Due to the ubiquity of 2D images, many studies rely solely on video data for tracking. Scheidegger et al. [21] employed the Poisson multi-Bernoulli mixture filter (PMBM) [22] to correlate 3D detection results across frames. These 3D detection results are generated by the region-based convolutional neural network (R-CNN) [23], leading to robust tracking outcomes. Additionally, object trajectories exhibit a clear sequential relationship in the temporal dimension. Building on this concept, Hu et al. [24] introduced mono3DT, which applies depth sorting to consecutive frames, enabling strong correlations between tracking instances. Transformers can indeed be effective in this application, given the temporal continuity of tracked objects. For example, [25] divides images into patches and embeds into a Transformer for learning purposes. Keypoint-based image detection technology is also

Table 1
Comparison of the advantages and disadvantages of 2D, 3D, and 2D+3D MOT methods.

	2D MOT	3D MOT	2D+3D MOT
Advantages	Abundant texture, color, and shape information. Effective tracking in well-lit, textured scenes.	Rich spatial details (position, orientation, size). Enhanced robustness against occlusions and viewpoint changes.	Integration of complementary cues from 2D and 3D data. Improved scene understanding for advanced tasks.
Disadvantages	Limited depth perception and spatial understanding. Vulnerability to scale changes, occlusions, and inconsistent viewpoints.	Complex data acquisition and processing needs. Lack of object texture, color, and shape information.	High computational cost and increased complexity in data fusion. Necessity for sensor synchronization and calibration.

instrumental in the development of MOT. CenterTrack [26] leverages key-point features of objects. With the assistance of a long short-term memory network (LSTM) [27], it can achieve notable tracking performance. The OC-SORT tracker [2] enhances this stability by combining object observations with a Kalman filter to address direction estimation errors that can occur due to object occlusion in center-based tracking.

While these methods benefit from the robust generalization capabilities of deep learning, they also grapple with several issues. These include a lack of precision in the depth measurement of bounding boxes assigned to target objects, high computational demands, and challenges in achieving real-time tracking efficiency.

2.2. 3D MOT

3D MOT primarily relies on point cloud data and the attributes of 3D bounding boxes generated by detectors. These methods establish relationships by measuring the differences between features of various bounding boxes. For instance, [8] uses the Hungarian algorithm [28] to match objects based on their overlap rates across different frames. Chiu et al. [9] improved this approach by incorporating the Mahalanobis distance [29] as a novel basis for affinity correlation. However, tracking objects in intricate environments remains a significant challenge. This is largely because point cloud-based object detection algorithms [30] do not exhibit the same level of stability as 2D image detection methods, leading to frequent detection discontinuities. This issue has not been adequately mitigated until the introduction of tracking uncertainty, a concept that improves tracking performance for objects that intermittently reappear.

Moreover, object feature extraction [10] and filter selection [11] play a crucial role in determining the effectiveness of tracking outcomes. Considering the spatio-temporal relationships of the same tracked object, Transformers [31] can analyze the spatio-temporal interactions of different objects to identify those with strong spatio-temporal correlations. [12] also utilizes attention mechanisms to classify edge features in sparse graphs, thus identifying pairs that are most likely to correspond to the same object. In the case of the VirConv tracker [10], after implementing the stochastic discard and noise-resistance strategies, the model is capable of learning both shape specifics and deep semantic features of point cloud data in terms of their compositional patterns in a fast and effective manner.

Cho et al. [13] employed graph convolutional networks (GCNs) to additionally consider long-term association information between objects, thereby utilizing historical trajectories to reduce the matching difficulty of incomplete detections. In addition to focusing on the association of objects across consecutive frames, analyzing the global information is also a viable approach. Tracklets, as a representation of tracking results over a short period from a global perspective, reflect the temporal correlations of tracked objects. Wu et al. [14] extended their focus beyond individual tracked targets and instead investigated associations among tracklets within frame sequences, reorganizing them to obtain accurate object trajectories. RethinkMOT [15] similarly addresses tracklets, where it processes outliers within tracklets and restores breakpoints and ID switch points in trajectories, thereby achieving high-precision tracking results. From a global perspective, single-feature analysis also incorporates more information. For instance, PolarMOT [16] focuses on geometric relationships by employing graph networks to learn the spatiotemporal relationships of objects from detections, thereby transforming the association task into a classification problem.

2.3. Joint 2D and 3D MOT

In addition to methods that leverage a single type of data input, many researchers have adopted techniques that utilize multiple data types simultaneously. For example, [32] introduces the scale-rotation-translation score (SRTs), a novel approach enabling affinity correlations for 3D bounding boxes while also leveraging YOLO [33] to extract semantic information from 2D images, thereby improving tracking performance. Similarly, some similarity networks, combined with VGG modules [34], are used to simultaneously extract analogous features from both images and point clouds. MotionTrack [35] establishes an end-to-end multi-class MOT baseline using Transformers with multi-modality sensor inputs. Additionally, data fusion can be introduced during either the detection or association phase. For example, DeepFusion [36] uses different compositions of intersection over union (IoU) to merge detection results from multiple perspectives. In contrast, the multi-modality MOT framework (mmMOT) [37] simultaneously extracts features from two dimensions and fuses them during the data association phase. A similar approach is also employed in JRMOT [38]. Moreover, bird's-eye view (BEV) images [39] provide unique information that can be valuable for enhancing data association.

3. Preliminaries

In this section, we introduce the key definitions and assumptions that underpin the formulation of the LiDAR-based multi-object tracking problem. The relevant nomenclature is provided in Appendix A.

3.1. Problem formulation

The primary goal of multi-object tracking is to assign a unique ID to each object across different timestamps within a given scenario. In typical cases involving a single LiDAR sensor, the sensor platform may have various configurations, including mobile setups, such as a probe vehicle, or stationary installations, such as a traffic light. Additionally, the objects within the LiDAR's detection range may exhibit both stationary and dynamic behavior relative to the sensor platform. The detection algorithm used in these systems can classify target objects into various categories, such as different types of vehicles, pedestrians, and cyclists. For the purposes of our research, we focus on a complex scenario where a single LiDAR sensor is mounted on a probe vehicle (moving sensor).

In a sequence of frames with the total duration T, when the detection results generated by the detector and point cloud data are taken as input, we are able to obtain the list of tracking states at each timestamp and the list of uniquely determined tracking IDs by applying the point cloud data (pcd) at t-th frame denoted as pcd_t and the tracking process as $\mathcal{F}$. For the data sequence from $1 : T$ time step, we have: In a sequence of frames over the total duration T, when the detection results and point cloud data are used as input, we can obtain the list of tracking states at each timestamp and the list of uniquely assigned tracking IDs.

Table 2
General nomenclature. ▲ denotes a variable that can be substituted.

Notation	Description	Unit
▲	Any variable or state notation	Various
$\dot{\blacktriangle}$	Velocity in the specified direction	$m\ s^{-1}$
$\ddot{\blacktriangle}$	Acceleration in the specified direction	$m\ s^{-2}$
$\blacktriangle'$	Lived variable or a state that does not exceed survival time limitation	–
$\hat{\blacktriangle}$	Predicted variable or state	–
$\tilde{\blacktriangle}$	Updated variable or state	–
$\blacktriangle^*$	Sub-state of a specified state	–
$n(\blacktriangle)$	Number of a variable or state	–
$d^{\blacktriangle}$	Number of dimensions of a variable or state	–

This is achieved by applying the point cloud data (pcd) at the t-th frame, denoted as pcd_t, and the tracking process represented by $\mathcal{F}$. For the data sequence from time step 1 to T, we have:

$$\left(\mathcal{S}_{1:T}, \varkappa_{1:T}\right) \leftarrow \mathcal{F}\left(\mathcal{D}^*_{1:T}, pcd_{1:T}\right), \tag{1}$$

where $\mathcal{S}_t$ denotes the tracked set, and $\mathcal{D}^*_t$ refers to the subset of detected state, both of which are clarified in Section 3.2. $\varkappa_t$ represents the set of unique identities assigned to each object at the t-th frame.

Here, we introduce two assumptions aimed at optimizing the prediction process by incorporating specific constraints without significantly limiting overall tracking performance.

1. **Constant Acceleration Assumption:** We assume that each target object maintains constant acceleration in all directions over a short time interval of 100 ms (the unit time interval, denoted as τ). This assumption is reasonable in many scenarios [40–42], where objects move smoothly without sudden changes in a short time in their motion patterns.
2. **Linear Relationship Assumption:** We assume that the initial state of each object can be approximated by a Gaussian distribution. The state of the tracked object at the current time step exhibits a linear relationship with its state at the previous time step, under the superposition of Gaussian noise. Additionally, the states of different objects are considered independent.

The proposed method is primarily composed of variables, states, and sets. Variables represent the values of specific features of an object at a given timestamp. States are typically matrices that consist of multiple variables and are temporally unique. Sets, in contrast, encompass all states of a particular object throughout an entire time sequence. Basic naming conventions are outlined in Table 2.

For instance, if $\left[x^i_t, y^i_t, z^i_t\right]$ (denoted as p^i_t) represents the three-dimensional coordinates of object i at time t, then $\left[\dot{x}^i_t, \dot{y}^i_t, \dot{z}^i_t\right]$ (denoted as $\dot{p}^i_t$) and $\left[\ddot{x}^i_t, \ddot{y}^i_t, \ddot{z}^i_t\right]$ (denoted as $\ddot{p}^i_t$) represent the velocity and acceleration of the object, respectively. If θ^i_t denotes the object's orientation angle, then $\dot{\theta}^i_t$ and $\ddot{\theta}^i_t$ represent the object's angular velocity and angular acceleration. Additionally, $\left[w^i_t, h^i_t, l^i_t\right]$ represents the width, height, and length of the object's bounding box at time t, while f^i_t represents the object's feature values at time t. The comprehensive nomenclature and notations are provided in Appendix A.

3.2. State and sub-state

As a detection-based multi-object tracking method, four states are introduced to represent the information extracted and inferred from detection results at different stages of the process. These states capture the evolving nature of the tracked objects and facilitate the integration of detection data into the tracking workflow. The four states are introduced below.

3.2.1. Detected state

The set of detected states includes all the information of detected bounding boxes at each timestamp in the data sequence, with each element representing the detected information for a single bounding box. The detected state matrix at timestamp t is denoted as $\mathcal{D}_t = \{D^{1:n(D^*_t)}_t\} \in \mathbb{R}^{d^D \times n(D^*_t)}$. Here, $n(D^*_t)$ represents the number of objects in the detected sub-states D^* at time t, corresponding to the number of detected targets at that time. Each element D^i_t in $\mathcal{D}_t$ contains information about the position, size, orientation, and features of the bounding box, as well as the initialized velocities and accelerations:

$$D^i_t = \left[p^i_t, \dot{p}^i_t, \ddot{p}^i_t, w^i_t, h^i_t, l^i_t, \theta^i_t, \dot{\theta}^i_t, \ddot{\theta}^i_t, f^{i,1:\xi}_t\right]^T. \tag{2}$$

$f^{i,1:\xi}_t$ represents the selected features from 1 to ξ within the bounding box of the i-th object at time t. The feature selection process is detailed in Eq. (6). Given the dynamic nature of both the sensor and target objects, with variations in the sizes of detected bounding boxes at different times and orientations, the size variability is intrinsically linked to changes in object position and orientation. Therefore, we sample the sizes of bounding boxes at each timestamp t to capture the dynamic characteristics of the object features.

3.2.2. Predicted state

The set of predicted states includes all information for predicted bounding boxes at each timestamp, based on the tracked states from the previous timestamp in the data sequence. Each element represents the predicted information for a single bounding box. The predicted state matrix at time t is denoted as $\hat{\mathcal{D}}_t = \{\hat{D}^{1:n(S_{t-1})}_t\} \in \mathbb{R}^{d^{\hat{D}} \times n(S_{t-1})}$. Here, $n(S_{t-1})$ refers to the number of tracked states S that are predictable at the previous timestamp $t-1$. Each element $\hat{D}^i_t$ in $\hat{\mathcal{D}}_t$ includes the predicted position, orientation, velocities, features, inherited size, and accelerations of each bounding box:

$$\hat{D}^i_t = \left[\hat{p}^i_t, \hat{\dot{p}}^i_t, \hat{\ddot{p}}^i_t, \hat{w}^i_t, \hat{h}^i_t, \hat{l}^i_t, \hat{\theta}^i_t, \hat{\dot{\theta}}^i_t, \hat{\ddot{\theta}}^i_t, \hat{f}^{i,1:\xi}_t\right]^T. \tag{3}$$

3.2.3. Updated state

The set of updated states includes all the updated information for detected bounding boxes at each timestamp, based on data association in the sequence. Each element represents the updated information for a single bounding box. The updated state matrix at timestamp t is denoted as $\tilde{\mathcal{D}}_t = \{\tilde{D}^{1:n(D_t)}_t\} \in \mathbb{R}^{d^{\tilde{D}} \times n(D_t)}$. Here, $n(D_t)$ refers to the number of detected states D (including both matched and unmatched detected states) at time t. Each element $\tilde{D}^i_t$ in $\tilde{\mathcal{D}}_t$ includes the updated position, size, orientation, velocities, accelerations, and inherited features of each bounding box:

$$\tilde{D}^i_t = \left[\tilde{p}^i_t, \tilde{\dot{p}}^i_t, \tilde{\ddot{p}}^i_t, \tilde{w}^i_t, \tilde{h}^i_t, \tilde{l}^i_t, \tilde{\theta}^i_t, \tilde{\dot{\theta}}^i_t, \tilde{\ddot{\theta}}^i_t, \tilde{f}^{i,1:\xi}_t\right]^T. \tag{4}$$

3.2.4. Tracked state

The set of tracked states includes all information for both detected bounding boxes and unmatched predicted bounding boxes at each timestamp in the data sequence. Each element represents the complete information for a single bounding box. The tracked state matrix at

time t is denoted as $S_t = \{S_t^{1:[n(\tilde{D}_t)+n(\hat{D}'_t)]}\} \in \mathbb{R}^{d^S \times [n(\tilde{D}_t)+n(\hat{D}'_t)]}$. Here, $n(\tilde{D}_t) + n(\hat{D}'_t)$ represents the number of updated states and predicted states that have not exceeded their survival time. Only these states are considered tracked and further proceed to the next timestamp for updates. In other words, the set of tracked states includes both updated states and lived predicted states. The symbol † denotes the value of the tracked state after updating, distinguishing it from the detected state. Each element S_t^i in S_t contains the position, size, orientation, velocities, accelerations, and features of each bounding box:

$$S_t^i = \left[{}^{\dagger}p_t^i, {}^{\dagger}\dot{p}_t^i, {}^{\dagger}\ddot{p}_t^i, {}^{\dagger}w_t^i, {}^{\dagger}h_t^i, {}^{\dagger}l_t^i, {}^{\dagger}\theta_t^i, {}^{\dagger}\dot{\theta}_t^i, {}^{\dagger}\ddot{\theta}_t^i, {}^{\dagger}f_t^{i,1:\xi}\right]^T. \tag{5}$$

3.2.5. Sub-state

The concept of a sub-state is introduced to represent a partial or incomplete representation of a target's state, which arises when certain attributes or information remain undetermined during the initialization and computation phases of the model. In the analysis of object detection results, it becomes clear that detection methods treat objects at different timestamps as separate, unrelated entities. Consequently, the detection outcomes exclusively provide positional information about these objects, excluding velocity and acceleration, which are typically derived from temporal changes in position. As such, the detected state is considered incomplete. Hence, $D^*{}_t$ and D^{*i}_t denote the subset and sub-state of the detected state, comprising direct information from the detection results, excluding velocity and acceleration. To maintain consistent variable dimensions, velocity and acceleration are initialized subsequently, transforming the detected state from D^{*i}_t to D^i_t. $\hat{D}^*{}_t$ and $\hat{D}^{*i}_t$ denote the sub-state of the predicted state used during prediction and update, without feature information. $\tilde{D}^*{}_t$ and $\tilde{D}^{*i}_t$ refer to the sub-state of the updated state used for the update step, also without feature information. $S^*{}_t$ and S^{*i}_t denote the sub-state of the tracked state used for prediction, excluding feature information at the subsequent timestamp.

$$D^{*i}_t = \left[p_t^i, w_t^i, h_t^i, l_t^i, \theta_t^i\right]^T, \tag{6}$$

$$\hat{D}^{*i}_t = \left[\hat{p}_t^i, \hat{\dot{p}}_t^i, \hat{\ddot{p}}_t^i, \hat{w}_t^i, \hat{h}_t^i, \hat{l}_t^i, \hat{\theta}_t^i, \hat{\dot{\theta}}_t^i, \hat{\ddot{\theta}}_t^i\right]^T, \tag{7}$$

$$\tilde{D}^{*i}_t = \left[\tilde{p}_t^i, \tilde{\dot{p}}_t^i, \tilde{\ddot{p}}_t^i, \tilde{w}_t^i, \tilde{h}_t^i, \tilde{l}_t^i, \tilde{\theta}_t^i, \tilde{\dot{\theta}}_t^i, \tilde{\ddot{\theta}}_t^i\right]^T, \tag{8}$$

$$S^{*i}_t = \left[{}^{\dagger}p_t^i, {}^{\dagger}\dot{p}_t^i, {}^{\dagger}\ddot{p}_t^i, {}^{\dagger}w_t^i, {}^{\dagger}h_t^i, {}^{\dagger}l_t^i, {}^{\dagger}\theta_t^i, {}^{\dagger}\dot{\theta}_t^i, {}^{\dagger}\ddot{\theta}_t^i\right]^T. \tag{9}$$

4. Proposed MOT method

In this section, we provide a detailed description of the proposed 3D MOT method. The overall framework is shown in Fig. 1. Each module is discussed in detail in the subsequent subsections, which include the processing model, prediction model, data association model, and update model. By explaining the workflow of the entire method and the functionality of each module, this section offers a thorough understanding of our proposed approach.

4.1. Process flow

Fig. 1 presents an overview of the proposed detection-based online tracking framework along a continuous timeline, illustrating the two complementary data streams and their interactions. The detection stream (the upper part of Fig. 1) introduces new observations obtained from a selected object detector at each frame, while the tracking stream (the lower part of Fig. 1) leverages historical state information to select the correct target in the current frame for matching and determining object trajectories over time. Box A in Fig. 1 represents the main processing steps of our proposed method.

By utilizing instantaneous global positioning system (GPS) and inertial measurement unit (IMU) data, along with the ego-vehicle's built-in sensor parameters, all measurements from the detection stream are anchored to a common reference point established at the initial frame for spatial consistency. Once the detected states are initialized in the global coordinate system, a data association procedure is executed using the integrated dynamically adjusted predicted confidence information from the previous frame. These confidence measures guide the binary matching rules, ensuring that each new detection is either matched with an existing prediction if it aligns well, or flagged as unmatched if no suitable counterpart is found. Similarly, some predicted states may remain unmatched if no current detections correspond to them. Matched state pairs and unmatched detected states undergo a state update process and are refined by a Kalman filter into tracked states. Unmatched predictions, meanwhile, are subjected to Survival Time Management to determine whether they will be retained. As the cycle repeats for each frame, these iterative refinements accumulate and the system incrementally improves its confidence and accuracy in tracked object states. This process flow yields stable and reliable long-term tracking performance, even in the presence of noisy measurements, target occlusions, and variations in object behavior.

4.2. Detected state initialization

As the detection results only provide the detected sub-state D^{*i}_t, containing details about position, size, and orientation, it is necessary to initialize the missing information to maintain a consistent data calculation format. Therefore, for the i-th target in the t-th frame, we have:

$$\text{Init}\left(D^{*i}_t\right) \rightarrow \begin{cases} \dot{x}_t^i, \dot{y}_t^i, \dot{z}_t^i = 0 \\ \ddot{x}_t^i, \ddot{y}_t^i, \ddot{z}_t^i = 0 \\ \dot{\theta}_t^i, \ddot{\theta}_t^i = 0 \end{cases}, \tag{10}$$

$$f_t^{i,1:\xi} = \begin{cases} \mathcal{N}\left[x_t^i, y_t^i, z_t^i, w_t^i, h_t^i, l_t^i, \theta_t^i\right] & \xi = 7 \\ \mathcal{N}\left[x_t^i, y_t^i, z_t^i, w_t^i, h_t^i, l_t^i\right] & \xi = 6 \\ \mathcal{N}\left[x_t^i, y_t^i, z_t^i, \theta_t^i\right] & \xi = 4 \\ \mathcal{N}\left[x_t^i, y_t^i, z_t^i\right] & \xi = 3 \end{cases}, \tag{11}$$

where $\mathcal{N}[\cdot]$ indicates that the matrix is processed using min–max normalization. The parameter ξ specifies the number of selected features to be retained. The effect of varying ξ on the tracking performance will be analyzed in detail in Section 5. Following initialization, the detected state D_t^i is obtained through concatenation.

$$D_t^i = \left[\text{Init}\left(D^{*i}_t\right), f_t^{i,1:\xi}\right]. \tag{12}$$

4.3. State prediction

Assumption 1 (constant acceleration) enables us to forecast the dynamic motion of targets using a basic physical model. For any non-initial frame t, we have:

$$\begin{aligned} \hat{x}_t^i &= x_{t-1}^i + \dot{x}_{t-1}^i\tau + \tfrac{1}{2}\ddot{x}_{t-1}^i\tau^2, \quad \hat{\dot{x}}_t^i = \dot{x}_{t-1}^i + \ddot{x}_{t-1}^i\tau, \\ \hat{y}_t^i &= y_{t-1}^i + \dot{y}_{t-1}^i\tau + \tfrac{1}{2}\ddot{y}_{t-1}^i\tau^2, \quad \hat{\dot{y}}_t^i = \dot{y}_{t-1}^i + \ddot{y}_{t-1}^i\tau, \\ \hat{z}_t^i &= z_{t-1}^i + \dot{z}_{t-1}^i\tau + \tfrac{1}{2}\ddot{z}_{t-1}^i\tau^2, \quad \hat{\dot{z}}_t^i = \dot{z}_{t-1}^i + \ddot{z}_{t-1}^i\tau, \\ \hat{\theta}_t^i &= \theta_{t-1}^i + \dot{\theta}_{t-1}^i\tau + \tfrac{1}{2}\ddot{\theta}_{t-1}^i\tau^2, \quad \hat{\dot{\theta}}_t^i = \dot{\theta}_{t-1}^i + \ddot{\theta}_{t-1}^i\tau, \end{aligned} \tag{13}$$

where τ represents the unit time interval of the data sequence. Additionally, the predicted accelerations $\hat{\ddot{p}}_t^i$ and the bounding box dimensions $\hat{w}_t^i, \hat{h}_t^i, \hat{l}_t^i$ at frame t remain unchanged from frame $t-1$. The predicted features $\hat{f}_t^{i,1:\xi}$ at frame t are directly inherited from frame $t-1$ and are not involved in the prediction process. As a result, the predicted sub-state $\hat{D}^{*i}_t$ and the predicted state covariance $\hat{P}_t^i$ can be derived as follows:

$$\hat{D}^{*i}_t = \mathbf{A}D^{*i}_{t-1}, \tag{14}$$

$$\hat{P}_t^i = \mathbf{A}P_{t-1}^i\mathbf{A} + \mathbf{Q} \quad \left(P_0^i = \mathbf{I}\right), \tag{15}$$

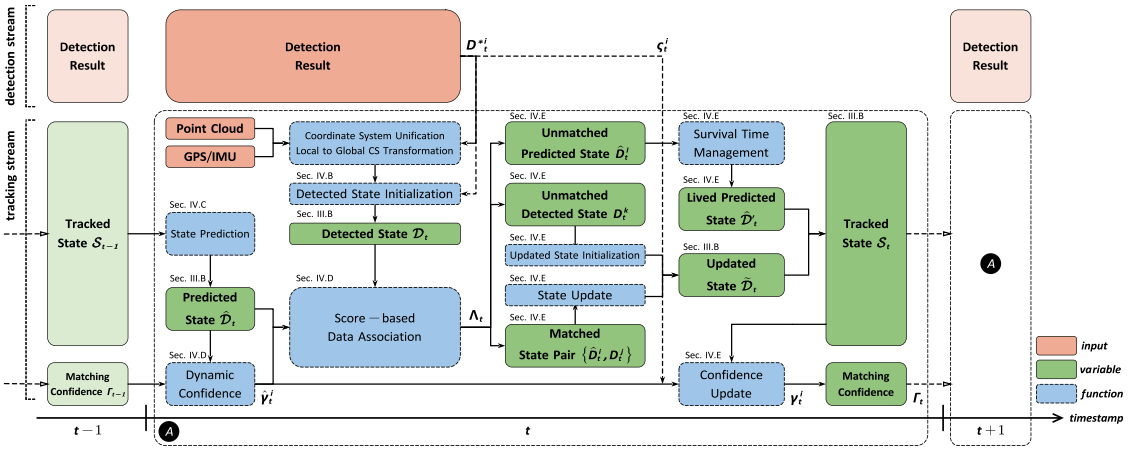


Fig. 1. Spatio-temporal correlation-based 3D multi-object tracking method. The solid orange boxes represent the external inputs to the method, while the solid green boxes denote the variables. The dashed blue boxes indicate the functions within this method. The light-colored boxes correspond to the upstream and downstream modules in relation to the current time.

where $\mathbf{A} \in \mathbb{R}^{15\times18}$ represents the transformation matrix derived from Eq. (13). $\mathbf{Q}$ refers to the noise generated, which follows a Gaussian distribution with zero mean and zero covariance. $\mathbf{I}$ denotes the identity matrix.

4.4. Data association model

The data association mechanism seeks to match the predicted states with candidate detected states in the current frame. By taking into account both spatial and temporal information to link objects across successive frames, we introduce a joint score function that leverages geometry, velocity, features, and the spatial distribution of point clouds to identify potential matching pairs. The score matrix is provided in Eq. (25).

4.4.1. Geometry score

The geometry score is a standard reference measure for evaluating the appearance discrepancy between predicted and detected states. The geometry score $SC_{\text{geo}}(\hat{D}_t^i, D_t^j)$, which compares the i-th predicted state with the j-th detected state at frame t, is formulated as:

$$SC_{\text{geo}}\left(\hat{D}_t^i, D_t^j\right) = \lambda_1 \cdot \mathcal{N}\left(\sum_{s\in\{w,h,l\}} \frac{\left|\hat{s}_t^i - s_t^j\right|}{\hat{s}_t^i + s_t^j}\right) + \lambda_2 \cdot \mathcal{N}\left(\| \hat{p}_t^i - p_t^j \|_2^2\right) + \lambda_3 \cdot \mathcal{N}\left(\sin\left|\hat{\theta}_t^i - \theta_t^j\right|\right), \tag{16}$$

where $\mathcal{N}(\cdot)$ denotes the min–max normalization function, and $\| \cdot \|_2$ represents the Euclidean norm (L2 norm). λ_1, λ_2, and λ_3 correspond to the respective weights for size, position, and orientation. Overall, the right-hand side of this equation individually computes the differences in size, position, and orientation between the bounding boxes of the predicted and detected states.

4.4.2. Velocity score

The velocity score is introduced to quantify the difference in the average velocity change of an object along its historical trajectory, taking into account the velocity at the current frame. Since all velocities are initially set to zero when the detected state is initialized, it is necessary to reassign the detected state's velocity based on the historical average velocity of the predicted state when calculating the velocity score for each state pair. Thus, the average velocity change of the i-th predicted state, from the time the object first appears until frame t, is denoted as $\{\bar{\Delta}\dot{\hat{p}}_t^i | \hat{D}_t^i\}$. Given the average velocity change of the i-th predicted state from the time the object first appears until frame $t-1$ as $\bar{\Delta}\dot{\hat{p}}_{t-1}^i$, the velocity of the j-th detected state at frame t can be estimated as:

$$\left\{\dot{p}_t^j | D_t^j\right\} = \left\{{}^{\dagger}\dot{p}_{t-1}^i | S_{t-1}^i\right\} + \left\{\bar{\Delta}\dot{\hat{p}}_{t-1}^i | \hat{D}_{t-1}^i\right\}. \tag{17}$$

Therefore, the velocity score can be calculated as:

$$SC_{\text{vel}}\left(\hat{D}_t^i, D_t^j\right) = \lambda_4 \cdot \mathcal{N}\left(\frac{\left|\bar{\dot{p}}_t^i - \dot{p}_t^j\right|}{\bar{\dot{p}}_t^i + \dot{p}_t^j}\right) + \lambda_5 \cdot \exp\left(-\frac{1}{\| \dot{\hat{p}}_t^i - \dot{p}_t^j \|_2^2}\right), \tag{18}$$

where λ_4 and λ_5 represent different weights. $\bar{\dot{p}}_t^i$ denotes the average velocity across t frames if the velocity of the predicted state $\hat{D}_t^i$ is considered as the velocity at frame t. Similarly, $\dot{p}_t^j$ is derived when accounting for the detected state D_t^j. Therefore, the two terms on the right-hand side of the equation compute the velocity differences between the predicted and detected states from different perspectives.

4.4.3. Historical feature score

Since feature values are not included in linear prediction or Kalman filtering (the method for updating them is detailed in Section 4.5.3), the original historical information of each trajectory is preserved over time. Retaining the historical features of objects offers a clear approach to handling changes in object scale. This is particularly useful in sparse point cloud data, where the distance between the LiDAR and the target object significantly affects the object's scale. Thus, the deviation between the detected state and historical features can be computed as follows:

$$SC_{\text{fea}}\left(\hat{D}_t^i, D_t^j\right) = \mathcal{N}\left(\| \hat{f}_t^{i,1:\xi} - f_t^{j,1:\xi} \|_2^2\right). \tag{19}$$

4.4.4. Distribution score

The distribution score is designed to measure the similarity in the spatial distribution of point clouds within two separate bounding boxes. By projecting the 3D point cloud data onto three axes, the similarity can be calculated using the spatial probability distribution of the points. To ensure computational efficiency, when calculating the point clouds within the bounding box, we only calculate the distribution probability of point clouds projected onto the coordinate axes. In the Cartesian coordinate system where the direction of travel of the ego vehicle is the y-axis, we take the calculation of the distribution probability in the x-axis direction as an example (as shown in Fig. 2). We divide the bounding box into several segments (we use 10 segments in our experiment), and calculate the distribution probability of all points projections on the x-axis within these 10 segments.

Let the spatial probability distributions along the three axes be represented as ${}^{\{x,y,z\}}\hat{D}_t^i$ and ${}^{\{x,y,z\}}D_t^j$, then we have:

$$SC_{\text{dis}}\left(\hat{D}_t^i, D_t^j\right) = \eta_1 \cdot S_{\text{c}}\left({}^{x}\hat{D}_t^i, {}^{x}D_t^j\right) + \eta_2 \cdot S_{\text{c}}\left({}^{y}\hat{D}_t^i, {}^{y}D_t^j\right) + \eta_3 \cdot S_{\text{c}}\left({}^{z}\hat{D}_t^i, {}^{z}D_t^j\right), \tag{20}$$

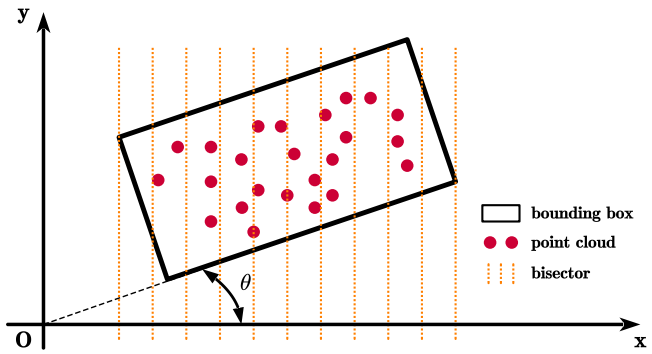


Fig. 2. The calculation method of the distribution score stated by Eqs. (20) and (21). The rectangular shape represents the bounding box, with the point clouds represented by red-colored points. θ denotes the orientation of this box. The orange dashed lines indicate dividing the bounding box into several equal parts along the horizontal direction.

where η_1, η_2, and η_3 represent the proportions of distribution similarity in the three respective directions. S_c denotes the cosine similarity. Given two m-dimensional attribute vectors, **A** and **B**, the cosine similarity $S_c(\mathbf{A}, \mathbf{B})$ can be computed using the dot product and magnitudes, as shown in Eq. (21):

$$S_c(\boldsymbol{A}, \boldsymbol{B}) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\| \boldsymbol{A} \| \| \boldsymbol{B} \|} = \frac{\sum_{n=1}^{m} A_n B_n}{\sqrt{\sum_{n=1}^{m} A_n^2} \sqrt{\sum_{n=1}^{m} B_n^2}}, \tag{21}$$

where A_n and B_n represent the nth components of the vectors **A** and **B**, respectively.

4.4.5. Dynamic tuned association score

The joint score matrix $\Lambda_t^{i,j}$ in our proposed method is derived by computing the weighted sum of the geometry, velocity, feature, and distribution scores, as shown in the following equation, where $\varphi_{\{1,2,3,4\}}$ represents the weight assigned to each score.

$$\Lambda_t^{i,j} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ -\varphi_4 \end{bmatrix}^T \times \begin{bmatrix} SC_{\text{geo}}(\hat{D}_t^i, D_t^j) \\ SC_{\text{vel}}(\hat{D}_t^i, D_t^j) \\ SC_{\text{fea}}(\hat{D}_t^i, D_t^j) \\ SC_{\text{dis}}(\hat{D}_t^i, D_t^j) \end{bmatrix}. \tag{22}$$

Additionally, considering that prediction reliability decreases as the number of prediction steps increases, a dynamic prediction confidence is introduced. This confidence reduction strategy is applied based on the 2D intersection over union (IoU) metric, leading to an adjustment of the joint score. Let M and N represent two bounding boxes, and the $IoU(M, N)$ is determined by the following rule.

$$IoU(M, N) = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{M \cap N}{M \cup N}. \tag{23}$$

Thus, the prediction confidence is introduced here to represent the level of trustworthiness associated with the current prediction. In practical situations, if a suitable detection can be found to pair with each prediction at every time step, the prediction confidence will remain at its initial value. However, if there are predictions for which no suitable detection can be found, then the prediction confidence for the next time step will decrease according to the following rules. The prediction confidence $\hat{\gamma}_t^i$ of i-th predicted state can be obtained as:

$$\hat{\gamma}_t^i = \begin{cases} 1 & t = 1 \text{ or } \nexists\ S_t^i \\ IoU(\hat{D}_t^i, S_{t-1}^i) \cdot \gamma_{t-1}^i & IoU(\hat{D}_t^i, S_{t-1}^i) \geqslant \mu \\ \mu \cdot \gamma_{t-1}^i & \text{otherwise} \end{cases}, \tag{24}$$

where μ is set as the static threshold for the confidence deduction purpose. The value of $\hat{\gamma}_t^i$ at current time step is influenced by the value of γ_{t-1}^i from previous time step. Additionally, the detection confidence also becomes a factor during the state update stage. For further details, please refer to the explanation of Eq. (27) and Eq. (28) in Section 4.5.3.

For all predicted states $\hat{D}_t$, the corresponding joint score graph Λ_t is:

$$\Lambda_t = \begin{bmatrix} \hat{\gamma}_t^1 \cdot \Lambda_t^{1,1} & \cdots & \hat{\gamma}_t^i \cdot \Lambda_t^{i,1} \\ \vdots & \ddots & \vdots \\ \hat{\gamma}_t^1 \cdot \Lambda_t^{1,j} & \cdots & \hat{\gamma}_t^i \cdot \Lambda_t^{i,j} \end{bmatrix}. \tag{25}$$

Here, i corresponds to the sequence of predicted states, while j corresponds to the sequence of detected states. Thus, when a predicted state has low confidence, we dynamically adjust its score downward to simplify matching, as shown in Fig. 3. These low-confidence states have a wider matching range, making it easier to find a suitable match.

The pairs with the lowest joint scores, determined using a greedy algorithm as outlined in Algorithm 1, are considered matched pairs, unless all joint scores for a predicted state exceed a given threshold ϵ. In these instances, the matching process fails, either due to the temporary absence of objects or the appearance of new objects in the scene, as shown in Fig. 3.

Algorithm 1: Pseudo-code of greedy algorithm for association.

Input: Joint score graph Λ_t, Predicted state $\hat{D}_t^i$, Number of predicted state m, Detected state D_t^j, Number of detected state n, Associated threshold ϵ

Output: Set of matched Pairs M, Set of unmatched predicted states U_P, Set of unmatched detected states U_D

```
Initialization
M ← ∅, idx ← 1, List ← ∅
U_P ← ∅, U_D ← ∅
for i ← 1 to m do
    for j ← 1 to n do
        if Λ_t[n][m] ⩽ ϵ then
            if Λ_t[n][m] ⩽ List[idx] then
                List[idx] ← Λ_t[n][m]
                M[idx] ← (D̂_t^m, D_t^n)
            end
        end
    end
    idx = idx + 1
end
for ∀ D̂_t^i ∉ M or ∀ D_t^j ∉ M do
    U_P ← U_P ∪ (D̂_t^i)
    U_D ← U_D ∪ (D_t^j)
end
```

Based on joint score graph Λ_t, this dynamic association strategy can be understood as follows: Suppose there are n predicted states and m detected states in the current scene. Let us designate i-th state among predicted states and j-th state among detected states. In pairs where the prediction $\hat{D}_t^i$ is the same but different detections $D_t^{1:m}$ are involved, each pair has the same prediction confidence $\hat{\gamma}_t^i$. In this case, finding the best match is solely determined by the association score. However, in pairs where the detection D_t^j is the same but different predictions $\hat{D}_t^{1:n}$ are considered, some predictions may have higher pairing scores with the detection due to a long disappearance period. In such cases, lower prediction confidence can play a regulating role, allowing even those predictions with higher scores to have the opportunity to be matched with the detection. Indeed, from this perspective, low confidence expands the potential pairing range.

Consider a scenario (as depicted in Fig. 4), where an object is traveling on a road from $t-4$ to $t+4$. At time t, it disappears from the detector's field of view due to being occluded, only to reappear at $t+3$. Under the strategy of dynamic confidence, the pairing range fluctuates only within a small range before step t because predicted states can find correct matches. However, starting from time t, predicted states fail to find correct matches. Assuming the object's disappearance time does not exceed the survival time, the pairing range will gradually increase at each time step. Therefore, we can observe that at time

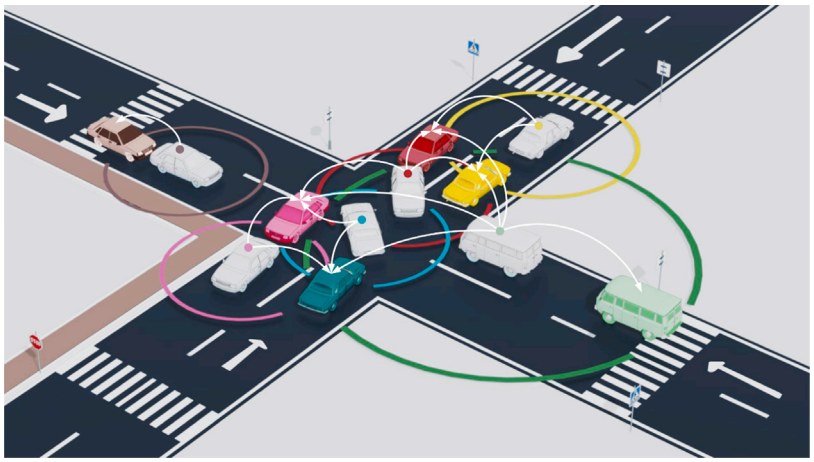


Fig. 3. Dynamic association strategy. Grey-colored car models represent predicted targets, while colored models denote detected targets. Vehicles move in the direction indicated by the gray arrows. The colored rings centered on each prediction represent the association range for the corresponding prediction, where detections covered by the ring are considered potential associations, indicated by white arrows. The color of the dot above each prediction corresponds to the color of its actual associated detection. The proximity between predicted and detected objects reflects the extent of their overlap.

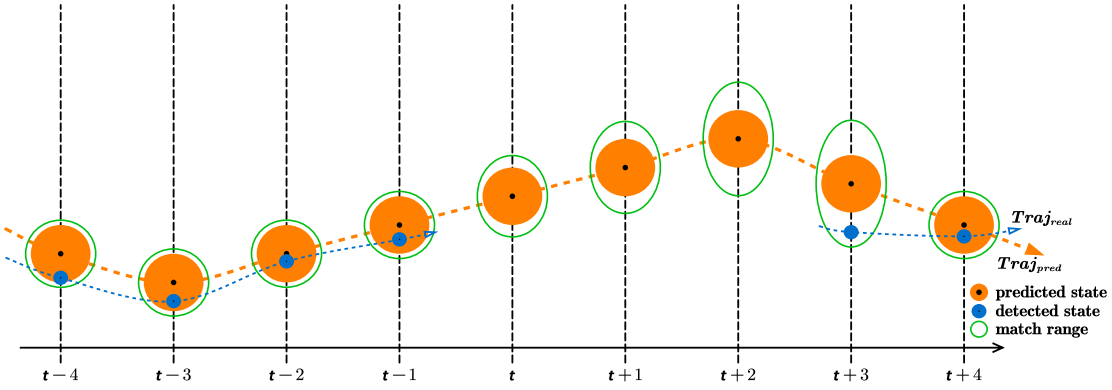


Fig. 4. Case analysis of temporarily occluded objects. On the timeline, the positions of the same object at different time steps are depicted. The blue circles represent the detected states of the object, the yellow circles represent the predicted states, and the green circles represent the pairing range at each timestamp. Lines connecting circles of the same color at different time points represent the trajectory of that state.

$t+3$, the detected state happens to appear at the edge of the pairing range, enabling predicted state to find a correct match again. Without this mechanism, the pairing range would remain the same as before the object disappeared, and the reappearing detected state might be identified as a new object.

4.5. State update

The update module incorporates new measurements into existing object tracks and refines their state estimates. By merging predictions with detections, it produces updated object states. The pseudo-code for the state update is presented in Appendix C.

4.5.1. Kalman filter update

The Kalman filter is an optimal recursive estimator that combines measurements and predictions to estimate the state within a dynamic system over time. Based on the assumption that measurement noise is linear and Gaussian distributed, the Kalman filter can efficiently estimate the updated sub-state $\tilde{D}^{*i}_t$ based on predicted sub-state $\hat{D}^{*i}_t$ and detected sub-state D^{*j}_t. Specific updates is elaborated in detail in Appendix B.

4.5.2. Unmatched state update

To initialize the unmatched detected sub-state D^{*k}_t, the same method used for detection initialization is applied. This involves setting all velocities and accelerations in the sub-state to 0. Additionally, the features $\tilde{f}^{k,1:\xi}_t$ of the updated state are directly inherited from the detected state. The predicted state covariance P^k_t is also set to the identity matrix $\mathbf{I}$, ensuring that the initial uncertainty is assumed to be equal across all dimensions. To address cases where no matching pair can be found for a predicted sub-state $\hat{D}^{*l}_t$—due to it being temporarily out of view or absent from the scene—a maximum survival time is defined, implementing the "max-age" concept proposed by Benbarka et al. [43]. If the number of prediction steps exceeds the maximum survival time, the predicted state is removed from the trajectory list.

4.5.3. Tracked state update

As shown in Fig. 1, the set of updated states, $\tilde{D}_t$, is formed by merging the states updated from matched pairs with the initialized states. Importantly, the features of the updated state $\tilde{D}^i_t$ must be updated to reflect the historical features as follows:

$$\tilde{f}^{i,1:\xi}_t = \begin{cases} \frac{1}{2}\left(\tilde{f}^{i,1:\xi}_{t-1} + \hat{f}^{i,1:\xi}_t\right) & \exists\ \tilde{f}^{i,1:\xi}_{t-1} \\ \hat{f}^{i,1:\xi}_t & \nexists\ \tilde{f}^{i,1:\xi}_{t-1} \end{cases}. \tag{26}$$

As a result, the set of tracked states $\mathcal{S}_t$ is formed by combining the updated states $\tilde{D}_t$ with the surviving predicted states $\hat{D}'_t$. Following this, all tracked states shift from prediction confidence to matching confidence, depending on the matching conditions. The specific rules

governing this transition are as follows:

$$\gamma_t^i = \begin{cases} 1 & \nexists\ \hat{\gamma}_{t-1}^i \text{ and sigmoid}\left(\varsigma_t^i\right) > 0.5 \\ \hat{\gamma}_t^i & \text{unmatched predicted state} \\ \min\left(\hat{\gamma}_t^i + \sigma_t^i, 1\right) & \text{otherwise} \end{cases}, \tag{27}$$

$$\sigma_t^i = 1 - IoU\left(\tilde{D}_t^i, D_t^i\right) \cdot \text{sigmoid}\left(\varsigma_t^i\right), \tag{28}$$

where sigmoid(ς_t^i) denotes the detected confidence ς_t^i of detection result activated by *sigmoid* function. γ_t^i is capped at 1 as the maximum value if it exceeds this value. $\nexists\ \hat{\gamma}_{t-1}^i$ indicates that the object appears for the first time, and sigmoid $\left(\varsigma t^i\right) > 0.5$ ensures that it can be confidently identified as the corresponding object. The updated confidence γ_t^i at each time step is incorporated into the matching confidence set Γ_t, which is subsequently carried forward to the next time step for recalculation. These two equations indicate that objects with higher detection confidence correspond to smaller values of γ_t^i, while objects with higher uncertainty correspond to larger values of γ_t^i. When an object first appears in the scene or reappears after being undetected, its confidence is updated to 1. If no suitable match is found, the predicted confidence from the predicted state is retained. Except for these cases, γ_t^i is influenced by the detection confidence. According to Eq. (28), when the detection result is more confident, the increment in the update of γ_t^i is smaller. Consequently, it becomes easier to find matching pairs in the next time step according to the smaller-is-better rule. Conversely, when the detector has a low level of confidence, the increment in the update of γ_t^i is larger, making it more difficult to correctly match objects in the next time step.

5. Experiment result

We performed a series of experiments and ablation tests[1] using the publicly available datasets KITTI [44] and nuScenes [17].

5.1. Experiment setup and evaluation metric

5.1.1. Experimental setup

All experiments were implemented in *Python* on Ubuntu 18.04 LTS, using an RTX 3070 graphics card and an AMD Ryzen 5-5600X processor with 16 GB of RAM, though our tracking method can operate without a GPU. The models requiring training in parallel experiments, along with all detectors, were trained on four NVIDIA V100 GPUs. For consistency, the training and testing sets consisted of 21 and 29 scenarios (each scenario processed as one data sequence), as officially divided by the datasets [44].

5.1.2. Implementation detail

The parameters in our proposed method, which are optimized with the goal of maximizing the MOTA on the KITTI dataset, are set as follows: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$ are set to $\{0.4, 0.4, 0.2, 0.7, 0.3\}$, $\{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ are set to $\{0.5, 0.125, 0.1875, 0.1875\}$, $\{\eta_1, \eta_2, \eta_3\}$ are set to $\{0.4, 0.4, 0.2\}$, μ is set to 0.7, and the associated threshold ϵ is set to 2. The maximum survival time is set to 12 (steps). The selected baseline detectors are Point-RCNN [45] and Megvii [46] to respectively generate tracking results on the testing data of KITTI and nuScenes.

5.1.3. Dataset

The proposed method was validated on two datasets. The primary dataset used is the KITTI tracking dataset,[2] a specialized subset of the KITTI dataset [44],[3] designed specifically for object tracking in autonomous driving scenarios in real-world environments. It provides extensive data and annotations to support the development and evaluation of tracking algorithms. The dataset includes an official leaderboard for easy performance validation and ranking of algorithms. It is divided into training and testing parts, with the training set used for parallel experiments and the testing set reserved for validation on the official website. Additionally, the nuScenes tracking dataset [17][4] was also used in this research to further validate the method's performance and robustness. The nuScenes dataset is partitioned similarly to the KITTI dataset but features a wider range of scenarios and object categories. It includes over 1000 sequences and 1.4 million bounding boxes. Unlike the KITTI dataset, which focuses primarily on cars and pedestrians, the nuScenes dataset introduces five additional categories. However, sequences in nuScenes are generally shorter in duration, and its leaderboard offers fewer methods for comparison compared to the more extensive leaderboard of the KITTI dataset.

5.1.4. Evaluation metric

We evaluated our proposed method and benchmarks using the *higher-order tracking accuracy* (HOTA) metrics, the primary evaluation criteria in the KITTI challenge. The widely-used CLEAR-MOT metrics in the object tracking field were also implemented for further evaluation. The first category of metrics includes HOTA, *localization accuracy* (LocA) score, *association accuracy* (AssA) score, and *detection accuracy* (DetA) score. The second category comprises *multiple object tracking accuracy* (MOTA), *multiple object tracking precision* (MOTP), and their related indices: *mostly tracked* (MT), *mostly lost* (ML), *ID switches* (IDS), *fragmentation* (Frag), *recall*, and *precision*. Additionally, we introduced *frames processed per second* (FPS) to evaluate the algorithm's efficiency. The metrics are as follows:

$$\begin{aligned} &\text{HOTA}_\alpha = \sqrt{\frac{\sum_{c\in\{\text{TP}_\alpha\}} \mathcal{A}_\alpha(c)}{\text{TP}_\alpha + \text{FN}_\alpha + \text{FP}_\alpha}}, \quad \text{HOTA} = \int_0^1 \text{HOTA}_\alpha d\alpha, \\ &\text{MOTA} = 1 - \frac{\text{FN} + \text{FP} + \text{IDS}}{\text{gtDet}}, \quad \text{MOTP} = \frac{1}{\text{TP}} \sum_{\text{TP}} C, \\ &\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ &\text{LocA} = \int_0^1 \frac{1}{\text{TP}_\alpha} \sum_{c\in\{\text{TP}_\alpha\}} \mathcal{S}(c) d\alpha, \quad \text{AssA} = \frac{1}{\text{TP}} \sum_{c\in\text{TP}} \mathcal{A}(c), \\ &\text{DetA} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}. \end{aligned} \tag{29}$$

Here, gtDet represents the ground-truth targets. TP (true positive) refers to tracking results that match with gtDet. Unmatched gtDets are labeled as FN (false negatives), while unmatched tracking results are labeled as FP (false positives), with both FN and FP indicating incorrect tracking outcomes. IDS represents the number of times IDs change inconsistently between consecutive frames, and Frag measures the number of times a tracker is interrupted. C denotes the average similarity score based on localization IoU (IoU_{Loc}), while $\mathcal{A}$ is an adjustable association score used to assess the degree of association. MOTA primarily evaluates tracking error rates, MOTP assesses localization errors, and HOTA provides an integrated evaluation metric. MT (rate $\geqslant$ 80%) and ML (rate $\leqslant$ 20%) measure the overlap rate between the tracked and actual trajectories. LocA separately evaluates the accuracy of localization under multiple localization thresholds α, where $\mathcal{S}(c)$ represents the spatial similarity score between predicted detections and ground-truth. AssA primarily evaluates the allocation accuracy for detected targets, assessing the model's handling of details for each detected object. DetA measures the overall detection capability of the model, focusing on missed detections and false positive results.

[1] The source codes and data storage guidelines are available on the project page on GitHub.

[2] KITTI Object Tracking Evaluation

[3] KITTI Vision Benchmark Suite

[4] nuScenes Tracking Task

Table 3
Car tracking results on KITTI testing dataset extracted from the official leaderboard. The methods are listed in descending order by year. * indicates methods that use the same detector, Point-RCNN. 2D refers to image data, while 3D refers to point cloud data. ↑ indicates that higher values correspond to better performance, while ↓ indicates that lower values are better. The best results are highlighted in **bold**, and the second-best results are marked with underlines.

Method	Modality	Source	HOTA↑	MOTA↑	MOTP↑	AssA↑	DetA↑	Recall↑	Precision↑	LocA↑	MT↑	ML↓	Frag↓	IDS↓
Proposed method*	3D	CPU	**82.28%**	90.44%	**87.02%**	84.82%	**78.44%**	92.78%	**98.94%**	**88.14%**	84.15%	5.85%	**52**	17
VirConvTrack* [10]	3D	GPU	81.87%	90.28%	86.93%	**85.19%**	78.14%	93.18%	98.12%	88.04%	83.23%	5.08%	77	**8**
RethinkMOT [15]	3D	GPU	80.39%	**91.47%**	85.63%	83.64%	77.88%	**96.63%**	95.95%	87.07%	**89.38%**	4.31%	134	46
LEGO [47]	3D	GPU	80.75%	90.80%	86.75%	83.27%	78.19%	96.06%	96.36%	87.92%	87.69%	**1.54%**	109	214
UG3DMOT [48]	3D	CPU	78.60%	88.10%	86.58%	82.28%	76.01%	92.15%	96.95%	87.84%	79.23%	5.38%	360	30
PolarMOT [16]	3D	GPU	75.16%	85.31%	85.52%	76.95%	73.94%	92.67%	94.40%	87.12%	81.38%	2.31%	599	462
MSA-MOT [49]	3D	GPU	78.52%	88.19%	85.47%	82.56%	75.19%	94.75%	94.53%	87.00%	87.23%	11.54%	428	91
PC3T* [50]	3D	CPU	77.80%	88.88%	84.37%	81.59%	74.57%	92.62%	97.75%	86.07%	80.00%	8.31%	201	225
AB3DMOT* [8]	3D	CPU	69.99%	83.92%	85.30%	69.33%	71.13%	88.17%	97.19%	71.31%	66.77%	9.08%	206	113
DeepFusionMOT* [36]	2D+3D	GPU	75.46%	84.80%	85.10%	80.05%	71.54%	87.94%	98.25%	86.70%	68.46%	9.08%	472	84
YONTD-MOT [51]	2D+3D	GPU	78.08%	85.19%	86.10%	84.01%	75.83%	89.76%	96.66%	87.23%	67.54%	7.08%	371	42
TripletTrack [52]	2D+3D	GPU	73.58%	84.77%	86.16%	74.66%	73.18%	88.18%	98.85%	87.37%	69.54%	3.38%	522	322
EagerMOT* [53]	2D+3D	GPU	74.39%	88.21%	85.73%	74.16%	75.27%	90.60%	98.69%	71.25%	76.62%	2.47%	390	239
JRMOT [38]	2D+3D	GPU	69.61%	85.70%	85.48%	66.89%	73.05%	89.51%	97.81%	76.64%	71.85%	4.00%	273	271
mmMOT [37]	2D+3D	GPU	62.05%	84.77%	85.21%	54.02%	72.29%	88.81%	97.93%	86.58%	73.23%	2.77%	570	733
aUToTrack [39]	2D+3D	GPU	59.83%	82.25%	80.52%	53.68%	67.82%	89.36%	97.03%	83.10%	72.62%	3.54%	484	1424
FANTrack [54]	2D+3D	GPU	60.85%	77.72%	82.33%	58.69%	64.36%	83.66%	96.15%	84.72%	62.62%	8.77%	701	743

5.2. Performance evaluation

5.2.1. Benchmark comparison

The official leaderboard shows that our method outperforms state-of-the-art models in most evaluation metrics, see Table 3. As an online method that only requires point cloud data and does not rely on GPU resources, our proposed approach outperforms most state-of-the-art methods in both the primary evaluation metric, HOTA, and the legacy metrics, MOTA and MOTP. This indicates fewer tracking errors and higher overlap rates with ground truth for predicting and updating object information. Specifically, our proposed method excels in the LocA metric compared to other benchmarks, indicating higher precision in localizing tracked object bounding boxes. This superiority is attributed to the incorporation of angle changes in the constant acceleration prediction model, resulting in smoother predictions. Similarly, the notable performance of our proposed method in the AssA and DetA metrics further demonstrates that our dynamic confidence mechanism helps stabilize the association strength between predictions and targets. Additionally, the proposed method exhibits high precision due to our aggregated data association model, which enables it to maintain correct pairing results even in complex scenarios.

However, the method lags behind some algorithms that utilize depth graph networks (such as LEGO) or perform tracklet cleaning in terms of the recall metric. This reflects the conservative nature of our proposed method, which tends to miss some true positives while ensuring that the majority of predicted positives are true positives. Moreover, the lower threshold ϵ in our proposed method somewhat restricts aggressive matching behavior. Furthermore, our proposed method achieves low values in the Frag and IDS metrics, indicating a strong long-term and stable tracking capability and the effectiveness of dynamic confidence in maintaining the uniqueness of tracked targets, even when they temporarily disappear from the detector's field of view. Moreover, compared with methods (VirConvTrack, PC3T, AB3DMOT, DeepFusionMOT, and EagerMOT) using the same detector Point-RCNN [45], our method has significant advantages in all evaluation indexes. This also rules out the possibility that the improved tracking results are solely due to the advanced capabilities of the detector.

In real-time applications, the efficiency of object tracking is just as crucial as tracking accuracy. To evaluate this, we selected seven open-source tracking methods, covering two different data input types (3D only, 2D+3D) and two data processing requirements (GPU, CPU-only). Each method was tested five times in the local environment, with the average efficiency presented in Fig. 5. The results indicate that methods requiring a GPU are significantly less efficient than those relying on CPU alone. While 60 FPS generally meets the standard for real-time tracking, it also demands higher computational performance from the device. Compared to methods that rely solely on the CPU, our approach balances competitive tracking accuracy with high data processing efficiency.

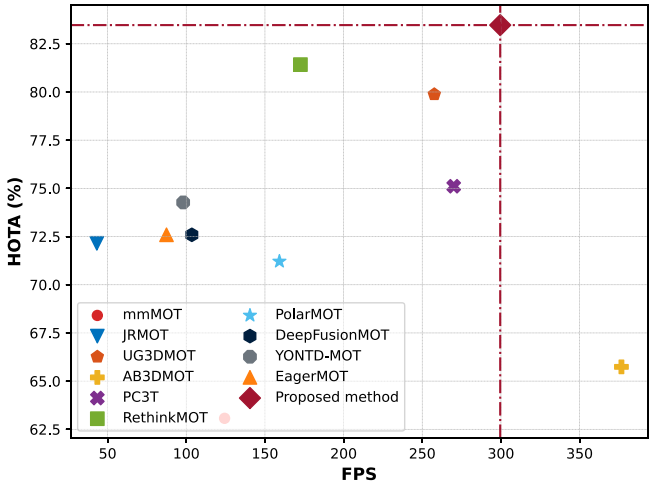
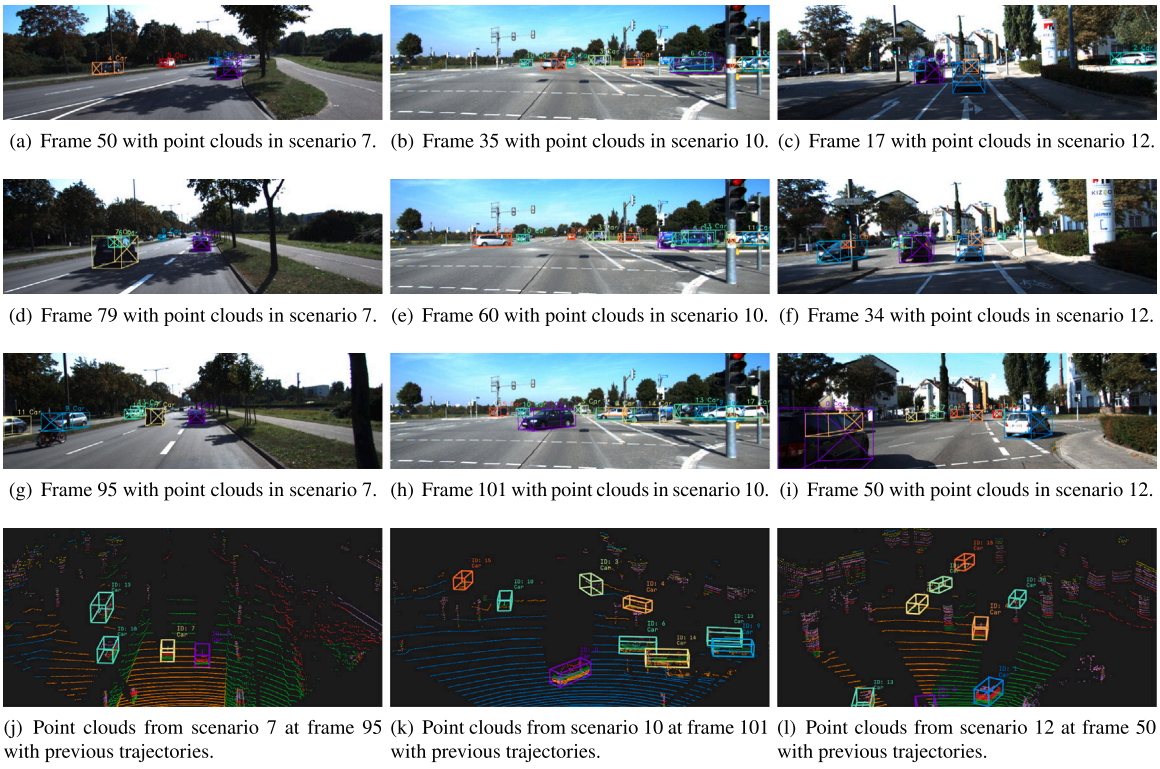


Fig. 5. FPS tests on the local computer environment. All comparison methods are either downloadable or trainable models available through their official GitHub repositories. The training procedures exclusively use the KITTI dataset. Some methods may perform below their optimal levels, as only default parameters were applied.

Furthermore, we analyzed the computational complexity of our proposed method to verify its efficiency. As a non-neural network algorithm running linearly along the timeline, our method does not involve training or inference processes, offering a significant advantage in complexity compared to neural network-based methods. In terms of time complexity, the algorithm is primarily dominated by the $O(m \cdot n)$ of the association model, where m and n respectively represent the number of detections and predictions. This complexity is manageably low due to the limited number of objects in real-world scenarios.

5.2.2. Result visualization

We selected three challenging scenarios (sequence 7, 10, and 12) from the testing set and visualized the tracking results, as shown in Fig. 6. For each scenario, we projected the point cloud data and the corresponding tracking results of each frame onto the depth images using the official coordinate transformation parameters, producing three sets of 2D visualizations. These provide a clearer representation of the tracking performance compared to sparse 3D point cloud data.



(a) Frame 50 with point clouds in scenario 7. (b) Frame 35 with point clouds in scenario 10. (c) Frame 17 with point clouds in scenario 12.

(d) Frame 79 with point clouds in scenario 7. (e) Frame 60 with point clouds in scenario 10. (f) Frame 34 with point clouds in scenario 12.

(g) Frame 95 with point clouds in scenario 7. (h) Frame 101 with point clouds in scenario 10. (i) Frame 50 with point clouds in scenario 12.

(j) Point clouds from scenario 7 at frame 95 with previous trajectories. (k) Point clouds from scenario 10 at frame 101 with previous trajectories. (l) Point clouds from scenario 12 at frame 50 with previous trajectories.

Fig. 6. Visualization of tracking results on the KITTI tracking dataset (testing set) across three different scenarios. The image results are generated by projecting point cloud data onto the stereo images.

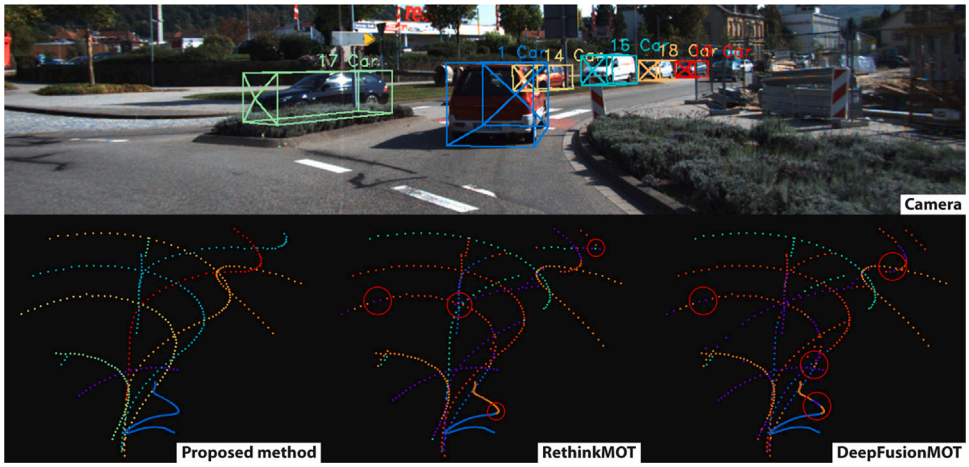
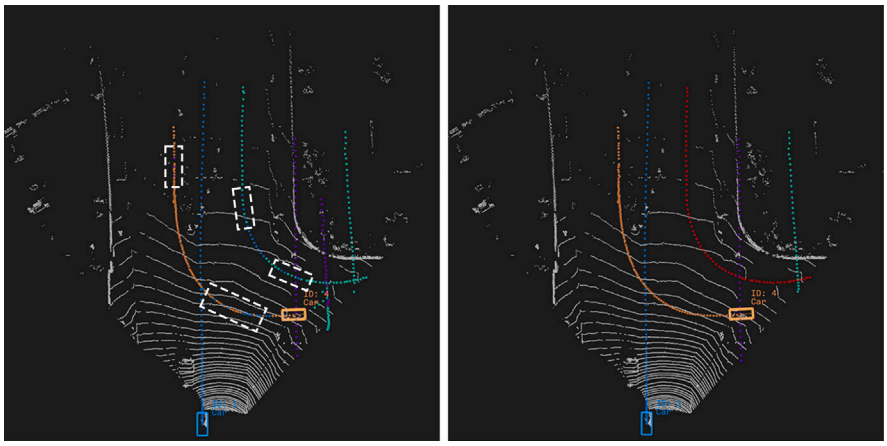


Fig. 7. The tracking results of three methods for the first 130 frames of scenario 28 in the testing dataset. The upper part shows a scene screenshot captured from an onboard camera. The lower part presents the tracking results of our proposed method, RethinkMOT, and DeepFusionMOT. Each dot represents the instantaneous position of a tracked object, with colors used to distinguish different objects. Red circles highlight potential ID switch occurrences.

To graphically demonstrate the stability of our proposed method in continuous tracking, we selected a complex roundabout scenario from the testing dataset, as illustrated in Fig. 7. In this roundabout, vehicles are intermittently occluded due to consecutive turns caused by preceding vehicles, leading to opposing-direction vehicles being partially obscured. We analyzed the first 130 frames of this scenario, comparing the tracking results of our proposed method with two other categories of methods (RethinkMOT [15] for 3D and DeepFusionMOT [36] for 2D+3D). Since the red circles highlight the main areas of ID changes, the results demonstrate that the proposed method exhibits more stable continuous tracking performance with fewer ID switches, particularly for temporarily occluded objects.

In addition, the effectiveness of the dynamic confidence strategy can be demonstrated through visualization. We selected two scenarios from the training dataset conducted in the experiments, as shown in Fig. 8. In both scenarios, there are moments when some vehicles are partially or completely obscured, disappearing from view. The figures on the left show the results without applying the dynamic confidence strategy, while the right ones display the outcomes with the proposed strategy implemented.

As shown in Fig. 8, when dynamic confidence is not applied, objects that reappear after being occluded are sometimes mistakenly identified as new objects, leading to the creation of new trajectory points and ID switches, especially in dense traffic or during turns. However, with



(a) Scenario 15 by frame 280 (without dynamic confidence).

(b) Scenario 15 by frame 280 (with dynamic confidence).

(c) Scenario 20 by frame 40 (without dynamic confidence).

(d) Scenario 20 by frame 40 (with dynamic confidence).

Fig. 8. Trajectories of tracked objects. The colored bounding boxes represent different tracked objects. White points depict the point clouds, while colored points show the historical trajectories of each object. The color of the trajectory corresponds to the color of the object's bounding box. The changing trajectory points within the white dashed box indicate ID switches that occurred during the tracking process.

the proposed strategy, the object trajectories become smoother, and ID switches are less frequent.

5.2.3. Parameter fine-tuning and sensitivity

In our method, affinity association is critical, and the weight assigned to each score significantly impacts the quality of the association. To determine the optimal weight combination, we used the delta tuning method [55], where one weight was optimized while the others were kept fixed.

5.3. Ablation study

To further assess the role and importance of each module in the method, we conducted several ablation experiments on the training dataset.

5.3.1. Score

Using the optimal weight combination, we conducted ablation tests on the weights, as shown in Table 4. Notably, when the geometry score is excluded, all metrics show significant degradation, with IDS reaching a high value of 746, indicating highly unstable tracking. This suggests that, in most cases, the spatial relationships between objects across time steps are the key to determining their association. Additionally, the results in Table 4 demonstrate that the other scores primarily function as refinements in complex scenarios during the data association process, without substantially impacting the overall performance. This observation is consistent with our findings during fine-tuning.

Table 4
Score ablation study. - means the corresponding score is dropped. The best results are highlighted in **bold**.

Geo	Vel	Fea	Dis	HOTA↑	MOTA↑	MOTP↑	IDS↓
✓	✓	✓	✓	**83.47%**	**90.74%**	**89.46%**	**10**
–	✓	✓	✓	62.01%	77.73%	89.03%	746
✓	–	✓	✓	81.70%	87.67%	89.21%	12
✓	✓	–	✓	81.64%	87.63%	89.21%	13
✓	✓	✓	–	81.50%	87.45%	89.24%	17

5.3.2. Detector

As an optional component, the detector naturally influences detection-based tracking methods like ours. To evaluate this impact, we tested four detectors with varying performance levels: SECOND [56], Point-RCNN [45], PV-RCNN [57], and VirConv [10]. Table 5 ranks these detectors based on their detection performance. The results confirm that a high-performing detector yields more accurate tracking results. However, with fine-tuning, even a detector with average performance can achieve tracking results comparable to those from top-performing detectors.

Table 5
Performance with different detectors. Note that the parameters of the overall method were fine-tuned based on the results from the Point-RCNN* detector (the original detector). The best results are highlighted in **bold**.

Detector	HOTA↑	MOTA↑	MOTP↑	IDS↓	Frag↓
SECOND [56]	79.60%	87.20%	89.04%	18	73
Point-RCNN* [45]	83.47%	**90.74%**	89.46%	10	34
PV-RCNN [57]	82.81%	89.98%	89.96%	23	43
VirConv [10]	**84.74%**	90.00%	**91.90%**	**9**	**22**

Table 6
Different predictors. The LSTM and GRU structures used are the original implementations from TensorFlow. The best results are highlighted in **bold**, and the second-best results are marked with underlines.

Predictor	HOTA↑	MOTA↑	MOTP↑	IDS↓	Frag↓	FPS↑
KF	81.52%	87.91%	88.60%	19	40	**336.8**
LSTM	81.76%	88.41%	89.73%	**10**	42	88.2
GRU	81.98%	88.32%	90.12%	11	37	63.3
Social-GAN [59]	80.47%	87.01%	89.33%	18	49	29.3
TPNet [60]	**83.65%**	89.28%	**90.48%**	12	**26**	43.8
Transformer-TF [61]	83.34%	90.29%	89.82%	**10**	33	49.6
Proposed method	83.47%	**90.74%**	89.46%	**10**	34	311.2

5.3.3. Predictor

The predictor, a key component of the method, also deserves attention. Our predictor is based on physical laws and incorporates certain assumptions. However, since we are dealing with time-series data, some neural network models can bypass these assumptions and directly predict future object states using historical data. Therefore, in addition to the Kalman filter (KF) and classic time-series prediction models such as the LSTM network and gated recurrent unit (GRU) [58], we also tested several state-of-the-art motion prediction models, including Social-GAN [59], TPNet [60], and Transformer-TF [61]. The results are presented in Table 6. It can be observed that our predictor generally achieves lower scores than neural network-based models in terms of the MOTP metric. This indicates that our predictor is less precise in predicting the exact locations of objects compared to neural networks, consistent with the strengths demonstrated by neural models in recent years. Moreover, under the same experimental conditions, accurate predictions indeed enhance tracking performance. However, our predictor performs comparably to, or even surpasses, motion predictors on comprehensive metrics such as HOTA and MOTA. This is largely because our approach focuses on the position at immediate next time step rather than long-term trajectories, which diminishes the advantages of motion predictors. In addition, the FPS results reveal that motion predictors often require additional external processing, such as transforming front-view information into bird's-eye view coordinates. These coordinate transformations, combined with the computational demands of neural network inference, significantly reduce tracking efficiency. Finally, the consistent tracking performance across different predictors further validates the robustness of our proposed method.

5.3.4. Dynamic confidence

As an innovative module in our method, we performed an ablation analysis on the effectiveness of dynamic confidence. Table 7 shows that when the confidence module is not applied, the model achieves the best performance in terms of MOTP. This is primarily because confidence adjustments forcefully alter the state of the targets, causing slight deviations from the ground truth. However, this correction enables our tracker to perform more accurately in complex scenes, as reflected by higher HOTA and MOTA scores, along with fewer IDS and Frag occurrences. On the other hand, while linear confidence improves tracking performance, it results in more ID switches and tracking fragments due to discrepancies between the fixed proportionate change in confidence and the actual situation.

Table 7
Performance under different confidence settings. "Linear" indicates that confidence is reduced and updated at a fixed rate. The best results are highlighted in **bold**.

Component	HOTA↑	MOTA↑	MOTP↑	IDS↓	Frag↓
Dynamic confidence	**83.47%**	**90.74%**	89.46%	**10**	**34**
Linear confidence	81.52%	88.70%	89.53%	205	222
w/o confidence	79.65%	88.28%	**91.62%**	35	50

Table 8
Instantaneous deflection analysis. A transient change of more than 45 degrees is classified as an instantaneous deflection. "Frequency" represents the instantaneous deflection frequency. The best results are highlighted in **bold**.

Component	Frequency↓	MOTA↑	MOTP↑	IDS↓	Frag↓
Angular V/A	**94**	**90.74%**	**89.46%**	**10**	**34**
w/o angular V/A	203	89.24%	87.13%	22	51

5.3.5. Angular velocity and acceleration

We also examined the effect of incorporating angular velocity (abbreviated as angular V) and angular acceleration (abbreviated as angular A) in reducing direction oscillation and its impact on the results. Here, we introduced the instantaneous deflection frequency to represent direction oscillation. This metric measures how often an object's orientation deviates by more than 45 degrees from its orientation in the previous and subsequent frames. As shown in Table 8, while the inclusion of angular V and angular A only slightly enhances overall tracking performance, it significantly improves the localization accuracy of correctly tracked objects. This improvement is also reflected in the notable reduction of the instantaneous direction oscillation.

5.4. Robustness evaluation

To assess the generalization capability of our method across different datasets, we conducted additional tracking tests on the nuScenes dataset, which includes a wider variety of object categories such as various types of vehicles, non-motorized vehicles, and pedestrians. We used detection outputs provided by the official Megvii [46] model and introduced four open-sourced, CPU-based detection frameworks (AB3DMOT [8], UG3DMOT [48], IPRL-TRI [9], and SimpleTrack [62]) as baselines. The evaluation metric used is MOTA, and the local test results are shown in Table 9. The findings clearly demonstrate the superior performance of our method across multiple object categories, with a particularly strong advantage for vehicles and pedestrians. The tracking performance for trailers is slightly worse, which may be attributed to the excessive length of the vehicles or the gap between the vehicle and the rear container, making it challenging to accurately compute the spatial distribution.

Furthermore, we evaluated the pedestrian tracking performance on the KITTI dataset. The comparison between our proposed method and the 2D, 2D+3D, and 3D benchmarks is presented in Table 10. Pedestrians have relatively small bounding boxes, which makes traditional 3D detection methods less effective in detecting them compared to 2D detection methods. Additionally, pedestrians exhibit greater distinguishability due to variations in clothing styles and colors. As a result, 2D methods demonstrate a significant advantage over the other two types of methods on the DetA metric. However, our proposed method achieves superior tracking accuracy and localization compared to similar methods and performs comparably to 2D methods on the HOTA metric. This further demonstrates the generalization capability of our proposed approach.

6. Summary and future research

To improve the perception capabilities of moving autonomous agents in complex environments, this paper presents a novel online 3D multi-object tracking (MOT) method for moving sensors. By introducing a new affinity model, the method enhances the sensor's ability

Table 9
Tracking results evaluated by MOTA in nuScenes tracking dataset with the same detector Megvii. Overall* results represent the average performance across all categories (including bicycle, bus, car, motorcycle, pedestrian, trailer, and truck). The best results are highlighted in **bold**.

Method	Overall*	Bicycle	Bus	Car	Motorcycle	Pedestrian	Trailer	Truck
AB3DMOT [8]	17.9%	0.9%	48.9%	36.0%	5.1%	9.1%	11.1%	14.2%
UG3DMOT [48]	55.7%	25.5%	62.8%	70.6%	54.9%	66.2%	**58.6%**	51.4%
IPRL-TRI [9]	56.1%	27.2%	74.1%	73.5%	50.6%	75.5%	33.7%	58.0%
SimpleTrack [62]	55.7%	**29.6%**	60.4%	71.2%	**56.3%**	68.4%	56.2%	47.6%
Proposed method	**57.1%**	28.9%	**75.3%**	**75.6%**	49.8%	**76.4%**	32.4%	**61.5%**

Table 10
Pedestrian tracking results on KITTI testing dataset extracted from the official leaderboard. The methods are listed in descending order by year. * indicates methods that use the same detector, Point-RCNN. 2D refers to image data, while 3D refers to point cloud data. ↑ indicates that higher values correspond to better performance, while ↓ indicates that lower values are better. The best results are highlighted in **bold**.

Method	Modality	Source	HOTA↑	AssA↑	DetA↑	LocA↑
Proposed method*	3D	CPU	51.78%	**59.22%**	42.90%	**79.01%**
PolarMOT [16]	3D	GPU	43.59%	48.12%	39.88%	71.34%
AB3DMOT* [8]	3D	CPU	37.81%	44.33%	32.37%	71.31%
TripletTrack [63]	2D+3D	GPU	42.77%	46.54%	39.54%	77.93%
EagerMOT* [64]	2D+3D	GPU	39.38%	38.72%	40.60%	71.25%
JHIT [65]	2D	GPU	54.07%	56.88%	**51.63%**	78.38%
OC-SORT [66]	2D	GPU	**54.69%**	59.08%	50.82%	78.52%
MO-YOLO [67]	2D	GPU	51.46%	58.39%	45.59%	77.86%

to accurately associate target objects, while incorporating object angle measurements to manage unstable angle estimations caused by relative motion. The proposed method not only improves prediction performance but also implements a new data association mechanism. Experimental results on real-world datasets, including KITTI and nuScenes, show that the method achieves state-of-the-art accuracy while maintaining high processing speed. The dynamic score adjustment strategy, based on detection confidence and predictive step length, effectively complements the newly proposed feature extraction function. With each update phase, this strategy significantly improves the tracker's ability to follow the same object and handle occluded objects in complex scenes.

However, as a method that relies solely on point cloud data, the proposed approach is limited in utilizing more visually intuitive features present in images, such as contours, colors, and textures, for object tracking. Additionally, our assumptions regarding noise distribution have certain limitations, as no scenario exhibits noise that strictly follows a Gaussian distribution. The dataset constraints also prevented us from performing long-term tracking tests and conducting a more comprehensive performance analysis, which can be explored in future work. Given the efficiency of this method, it can be effectively applied in autonomous transport infrastructure (ATI) fields and heterogeneous data fusion scenarios, offering enriched information features for tracking.

CRediT authorship contribution statement

Ruihao Zeng: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Mohsen Ramezani:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of AI assistance

Artificial Intelligence (AI) tools were employed in the preparation of this manuscript for proofreading and language enhancement purposes. No portion of the core contents of the article was generated by AI.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Nomenclature

All individual variables and variable sets are presented in Table 11.

Appendix B. Kalman filter update

The Kalman filter is widely used for object tracking in fields such as computer vision and robotics. For the predicted sub-state $\hat{D}^{*i}_t$ and detected sub-state D^{*j}_t (which corresponds to the measurement in the Kalman filter) in well-matched pairs, the updated sub-state $\tilde{D}^{*i}_t$ can be computed using the following steps:

$$\hat{\mathbf{o}}^{\mathbf{i}}_{\mathbf{t}} = \mathbf{H}\hat{P}^{i}_{t}\mathbf{H}^{\mathbf{T}} + \mathbf{R}, \tag{30}$$

$$\mathbf{K}^{\mathbf{i}}_{\mathbf{t}} = \hat{P}^{i}_{t}\mathbf{H}^{\mathbf{T}}\left(\hat{\mathbf{o}}^{\mathbf{i}}_{\mathbf{t}}\right)^{-1}, \tag{31}$$

$$\tilde{D}^{*i}_{t} = \hat{D}^{*i}_{t} + \mathbf{K}^{\mathbf{i}}_{\mathbf{t}}\left(D^{*j}_{t} - \mathbf{H}\hat{D}^{*i}_{t}\right), \tag{32}$$

$$P^{i}_{t} = \left(\mathbf{I} - \mathbf{K}^{\mathbf{i}}_{\mathbf{t}}\mathbf{H}\right)\hat{P}^{i}_{t}, \tag{33}$$

where $\mathbf{H}$ is the linear measurement matrix, defined as $\mathbf{H} = [\mathbf{I}\ \mathbf{O}] \in \mathbb{R}^{d^{D^*} \times d^{\hat{D}^*}}$. $\hat{\mathbf{o}}^{\mathbf{i}}_{\mathbf{t}}$ represents the uncertainty of the predicted state, calculated with the help of Gaussian noise $\mathbf{R}$, which has zero mean and covariance. $\mathbf{K}^{\mathbf{i}}_{\mathbf{t}}$ is the Kalman gain, determining the relative weight between the predicted state estimate and the measurement update. The predicted state covariance $\hat{P}^{i}_{t}$ is updated as shown in Eq. (15).

Appendix C. Pseudo-code of state update

The operations of concatenation, addition, and removal are all represented using the set notation.

Data availability

The KITTI dataset provides general access via KITTI official website. nuScenes dataset is also a public dataset available on the nuScenes official website.

Table 11
Nomenclature list of the proposed method.

Notation	Description	Unit
x_t^i	Global coordinate in x direction of center point of i-th bounding box at t-th frame	m
y_t^i	Global coordinate in y direction of center point of i-th bounding box at t-th frame	m
z_t^i	Global coordinate in z direction of center point of i-th bounding box at t-th frame	m
θ_t^i	Orientation (yaw with x-axis) of i-th bounding box at t-th frame	m
h_t^i	Height of i-th bounding box at t-th frame	m
l_t^i	Length of i-th bounding box at t-th frame	m
$f_t^{i,1:\xi}$	The selected ξ original features of i-th bounding box at t-th frame	–
κ_t^i	The unique identity of i-th bounding box at t-th frame	–
ς_t^i	The detection confidence of i-th detected state at t-th frame	–
$\hat{\gamma}_t^i$	The predicted confidence of i-th predicted state at t-th frame	–
γ_t^i	The updated confidence of i-th tracked state at t-th frame	–
$\Lambda_t^{i,j}$	The association score of i-th predicted object and j-th detected object at t-th frame	–
D_t^i	Detected state of i-th bounding box at t-th frame	–
$\hat{D}_t^i$	Predicted state of i-th bounding box at t-th frame	–
$\tilde{D}_t^i$	Updated state of i-th bounding box at t-th frame	–
S_t^i	Tracked state of i-th bounding box at t-th frame	–
p_t^i	Global coordinates of center point of i-th bounding box at t-th frame	–
D_t	The set of detected states at t-th frame	–
$\hat{D}_t$	The set of predicted states at t-th frame	–
$\tilde{D}_t$	The set of updated states at t-th frame	–
S_t	The set of tracked states at t-th frame	–
$\varkappa_t$	The set of unique identities at t-th frame	–
$\hat{\Gamma}_t$	The set of predicted confidences at t-th frame	–
Γ_t	The set of matching confidences at t-th frame	–
Λ_t	The set of association score at t-th frame	–

Algorithm 2: Pseudo-code of state update.

Input: Set of unmatched predicted states U_P, Number of unmatched predicted states l, Set of unmatched detected states U_D, Number of unmatched detected states o, Set of matched pairs M, Number of matched pairs q, Survival time Set S, Max survival time m_s

Output: Set of lived predicted states L_P, Set of updated states U, Set of tracked states T

Initialization

$L_P \leftarrow \emptyset,\ U \leftarrow \emptyset,\ T \leftarrow \emptyset$

for $i \leftarrow 1$ ***to*** q **do**
  $U_i \leftarrow$ Kalman Filter(M_i)
  $U \leftarrow U \cup U_i$
end
for $j \leftarrow 1$ ***to*** o **do**
  $U_j \leftarrow$ State Initialization(U_{Dj})
  $U \leftarrow U \cup U_j$
end
for $k \leftarrow 1$ ***to*** l **do**
  if $\exists S_k$ **then**
    $S_k = S_k - 1$
    if $S_k < 0$ **then**
      $U_P \leftarrow U_P \setminus U_{Pk}$
      $L_P \leftarrow L_P \setminus L_{Pk}$
    end
  end
  else
    $S_k \leftarrow m_s$
    $S \leftarrow S \cup S_k$
  end
end
$T \leftarrow U \cup L_P$

References

[1] Z. Xie, M. Ramezani, D. Levinson, Reduced-scale mobile robots for autonomous driving research, IEEE Trans. Intell. Transp. Syst. (2024).

[2] J. Cao, X. Weng, R. Khirodkar, J. Pang, K. Kitani, Observation-centric sort: Rethinking sort for robust multi-object tracking, 2022, arXiv preprint arXiv:2203.14360.

[3] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y.N. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z.F. Chen, D. Anguelov, Scalability in perception for autonomous driving: Waymo open dataset, in: 2020 Ieee/Cvf Conference on Computer Vision and Pattern Recognition, Cvpr, 2020, pp. 2443–2451.

[4] M.F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, J. Hays, Argoverse: 3D tracking and forecasting with rich maps, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Cvpr 2019, 2019, pp. 8740–8749.

[5] Z. Bai, G. Wu, X. Qi, Y. Liu, K. Oguchi, M.J. Barth, Infrastructure-based object detection and tracking for cooperative driving automation: A survey, in: 2022 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2022, pp. 1366–1373.

[6] A.D. Beza, Z. Xie, M. Ramezani, D. Levinson, From lane-less to lane-free: Implications in the era of automated vehicles, Transp. Res. Part C: Emerg. Technol. 170 (2025) 104898.

[7] L. Chen, A.H. Valadkhani, M. Ramezani, Decentralised cooperative cruising of autonomous ride-sourcing fleets, Transp. Res. Part C: Emerg. Technol. 131 (2021) 103336.

[8] X.S. Weng, J.R. Wang, D. Held, K. Kitani, 3D multi-object tracking: A baseline and new evaluation metrics, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, Iros, 2020, pp. 10359–10366.

[9] H.K. Chiu, J. Lie, R. Ambrus, J. Bohg, Probabilistic 3D multi-modal, multi-object tracking for autonomous driving, in: 2021 Ieee International Conference on Robotics and Automation, Icra 2021, 2021, pp. 14227–14233.

[10] H. Wu, C. Wen, S. Shi, X. Li, C. Wang, Virtual sparse convolution for multimodal 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 21653–21662.

[11] G. Guo, S.J. Zhao, 3D multi-object tracking with adaptive Cubature Kalman filter for autonomous driving, Ieee Trans. Intell. Veh. 8 (1) (2023) 512–519.

[12] S. Ding, E. Rehder, L. Schneider, M. Cordts, J. Gall, 3Dmotformer: Graph transformer for online 3D multi-object tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 9784–9794.

[13] M. Cho, E. Kim, 3D LiDAR multi-object tracking with short-term and long-term multi-level associations, Remote. Sens. 15 (23) (2023) 5486.

[14] H. Wu, Q. Li, C. Wen, X. Li, X. Fan, C. Wang, Tracklet proposal network for multi-object tracking on point clouds, in: IJCAI, 2021, pp. 1165–1171.

[15] L. Wang, J. Zhang, P. Cai, X. Lil, Towards robust reference system for autonomous driving: Rethinking 3D MOT, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 8319–8325.

[16] A. Kim, G. Brasó, A. Ošep, L. Leal-Taixé, Polarmot: How far can geometric relations take us in 3d multi-object tracking? in: European Conference on Computer Vision, Springer, 2022, pp. 41–58.

[17] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, 2019, arXiv preprint arXiv:1903.11027.

[18] W.H. Luo, J.L. Xing, A. Milan, X.Q. Zhang, W. Liu, T.K. Kim, Multiple object tracking: A literature review, Artificial Intelligence 293 (2021).

[19] J. Yang, M. Gao, Z. Li, S. Gao, F. Wang, F. Zheng, Track anything: Segment anything meets videos, 2023, arXiv preprint arXiv:2304.11968.

[20] Y.A. Li, C. Huang, R. Nevatia, Learning to associate: HybridBoosted multi-target tracker for crowded scene, in: Cvpr: 2009 IEEE Conference on Computer Vision and Pattern Recognition, vols. 1-4, 2009, pp. 2945–2952.

[21] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, K. Granstrom, Mono-camera 3D multi-object tracking using deep learning detections and PMBM filtering, in: 2018 IEEE Intelligent Vehicles Symposium, Iv, 2018, pp. 433–440.

[22] A.F. Garcia-Fernandez, J.L. Williams, K. Granstrom, L. Svensson, Poisson multi-Bernoulli mixture filter: Direct derivation and implementation, Ieee Trans. Aerosp. Electron. Syst. 54 (4) (2018) 1883–1901.

[23] R. Girshick, J. Donahue, T. Darrell, J. Malik, Region-based convolutional networks for accurate object detection and segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 38 (1) (2015) 142–158.

[24] H.N. Hu, Q.Z. Cai, D.Q. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, F. Yu, Joint monocular 3D vehicle detection and tracking, in: 2019 IEEE/CVF International Conference on Computer Vision, Iccv 2019, 2019, pp. 5389–5398.

[25] X. Zhu, Y. Jia, S. Jian, L. Gu, Z. Pu, ViTT: vision transformer tracker, Sensors 21 (16) (2021) 5608.

[26] X. Zhou, V. Koltun, P. Krähenbühl, Tracking objects as points, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV, Springer, 2020, pp. 474–490.

[27] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[28] G.A. Mills-Tettey, A. Stentz, M.B. Dias, The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs, Tech. Rep. CMU-RI-TR-07-27, Robotics Institute, Pittsburgh, PA, 2007.

[29] G.J. McLachlan, Mahalanobis distance, Resonance 4 (6) (1999) 20–26.

[30] D. Tsai, J.S. Berrio, M. Shan, E. Nebot, S. Worrall, MS3D++: Ensemble of experts for multi-source unsupervised domain adaptation in 3D object detection, IEEE Trans. Intell. Veh. (2024).

[31] J. Willes, C. Reading, S.L. Waslander, Intertrack: Interaction transformer for 3d multi-object tracking, in: 2023 20th Conference on Robots and Vision, CRV, IEEE, 2023, pp. 73–80.

[32] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, H.M. Gross, Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds, in: 2019 Ieee/Cvf Conference on Computer Vision and Pattern Recognition, Cvprw 2019, 2019, pp. 1190–1199.

[33] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[34] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[35] C. Zhang, C. Zhang, Y. Guo, L. Chen, M. Happold, Motiontrack: end-to-end transformer-based multi-object tracking with lidar-camera fusion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 151–160.

[36] X.Y. Wang, C.Y. Fu, Z.K. Li, Y. Lai, J.W. He, DeepFusionMOT: A 3D multi-object tracking framework based on camera-LiDAR fusion with deep association, Ieee Robot. Autom. Lett. 7 (3) (2022) 8260–8267.

[37] W.W. Zhang, H. Zhou, S.Y. Sun, Z. Wang, J.P. Shi, C.C. Loy, Robust multi-modality multi-object tracking, in: 2019 IEEE/CVF International Conference on Computer Vision, Iccv 2019, 2019, pp. 2365–2374.

[38] A. Shenoi, M. Patel, J. Gwak, P. Goebel, A. Sadeghian, H. Rezatofighi, R. Martin-Martin, S. Savarese, JRMOT: A real-time 3D multi-object tracker and a new large-scale dataset, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, Iros, 2020, pp. 10335–10342.

[39] K. Burnett, S. Samavi, S.L. Waslander, T.D. Barfoot, A.P. Schoellig, aUToTrack: A lightweight object detection and tracking system for the SAE AutoDrive challenge, in: 2019 16th Conference on Computer and Robot Vision, Crv 2019, 2019, pp. 209–216.

[40] N.L. Baisa, Derivation of a constant velocity motion model for visual tracking, 2020, arXiv preprint arXiv:2005.00844.

[41] S.R. Jondhale, R.S. Deshpande, Tracking target with constant acceleration motion using Kalman filtering, in: 2018 International Conference on Advances in Communication and Computing Technology, ICACCT, IEEE, 2018, pp. 581–587.

[42] G. Zhai, H. Meng, X. Wang, A constant speed changing rate and constant turn rate model for maneuvering target tracking, Sensors 14 (3) (2014) 5239–5253.

[43] N. Benbarka, J. Schröder, A. Zell, Score refinement for confidence-based 3D multi-object tracking, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2021, pp. 8083–8090.

[44] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The KITTI dataset, Int. J. Robot. Res. 32 (11) (2013) 1231–1237.

[45] S. Shi, X. Wang, H. Li, Pointrcnn: 3d object proposal generation and detection from point cloud, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 770–779.

[46] B. Zhu, Z. Jiang, X. Zhou, Z. Li, G. Yu, Class-balanced grouping and sampling for point cloud 3d object detection, 2019, arXiv preprint arXiv:1908.09492.

[47] Z. Zhang, J. Liu, Y. Xia, T. Huang, Q.-L. Han, H. Liu, LEGO: Learning and graph-optimized modular tracker for online multi-object tracking with point clouds, 2023, arXiv preprint arXiv:2308.09908.

[48] J. He, C. Fu, X. Wang, 3D multi-object tracking based on uncertainty-guided data association, 2023, arXiv preprint arXiv:2303.01786.

[49] Z. Zhu, J. Nie, H. Wu, Z. He, M. Gao, MSA-MOT: Multi-stage association for 3D multimodality multi-object tracking, Sensors 22 (22) (2022) 8650.

[50] H. Wu, W.K. Han, C.L. Wen, X. Li, C. Wang, 3D multi-object tracking in point clouds based on prediction confidence-guided data association, Ieee Trans. Intell. Transp. Syst. 23 (6) (2022) 5668–5677.

[51] X. Wang, J. He, C. Fu, T. Meng, M. Huang, You only need two detectors to achieve multi-modal 3d multi-object tracking, 2023, arXiv preprint arXiv:2304.08709.

[52] N. Marinello, M. Proesmans, L. Van Gool, TripletTrack: 3D object tracking using triplet embeddings and LSTM, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4500–4510.

[53] A. Kim, A. Ošep, L. Leal-Taixé, Eagermot: 3d multi-object tracking via sensor fusion, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 11315–11321.

[54] E. Baser, V. Balasubramanian, P. Bhattacharyya, K. Czarnecki, FANTrack: 3D multi-object tracking with feature association network, Iv19, 2019, pp. 1426–1433.

[55] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, et al., Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models, 2022, arXiv preprint arXiv:2203.06904.

[56] Y. Yan, Y. Mao, B. Li, Second: Sparsely embedded convolutional detection, Sensors 18 (10) (2018) 3337.

[57] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, H. Li, Pv-rcnn: Point-voxel feature set abstraction for 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10529–10538.

[58] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Gated feedback recurrent neural networks, in: International Conference on Machine Learning, PMLR, 2015, pp. 2067–2075.

[59] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, A. Alahi, Social gan: Socially acceptable trajectories with generative adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2255–2264.

[60] L. Fang, Q. Jiang, J. Shi, B. Zhou, Tpnet: Trajectory proposal network for motion prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6797–6806.

[61] F. Giuliari, I. Hasan, M. Cristani, F. Galasso, Transformer networks for trajectory forecasting, in: 2020 25th International Conference on Pattern Recognition, ICPR, IEEE, 2021, pp. 10335–10342.

[62] J.X. Li, Y. Ding, H.L. Wei, Y.T. Zhang, W.X. Lin, SimpleTrack: Rethinking and improving the JDE approach for multi-object tracking, Sensors 22 (15) (2022).

[63] N. Marinello, M. Proesmans, L. Van Gool, Triplettrack: 3d object tracking using triplet embeddings and lstm, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4500–4510.

[64] A. Kim, A. Ošep, L. Leal-Taixé, Eagermot: 3d multi-object tracking via sensor fusion, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 11315–11321.

[65] P.J. Claasen, J.P. de Villiers, One homography is all you need: IMM-based joint homography and multiple object state estimation, 2024, arXiv preprint arXiv:2409.02562.

[66] J. Cao, J. Pang, X. Weng, R. Khirodkar, K. Kitani, Observation-centric sort: Rethinking sort for robust multi-object tracking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 9686–9696.

[67] L. Pan, Y. Feng, W. Di, L. Bo, Z. Xingle, MO-YOLO: End-to-end multiple-object tracking method with YOLO and MOTR, 2023, arXiv preprint arXiv:2310.17170.