CrossTracker: Robust Multi-Modal 3D Multi-Object Tracking via Cross Correction

Lipeng Gu, Xuefeng Yan, Weiming Wang, Honghua Chen, Dingkun Zhu, Liangliang Nan, and Mingqiang Wei, *Senior Member, IEEE*

***Abstract*—Inaccurate detections remain a critical bottleneck in 3D multi-object tracking (MOT). Recent detection fusion-based methods incorporate camera detections as supplementary to reduce false detections and compensate for missing ones in LiDAR. However, their unidirectional camera-LiDAR correction lacks a feedback mechanism, precluding iterative mutual refinement between modalities for more robust LiDAR-based tracking. Inspired by the coarse-to-fine strategy in two-stage object detection, we introduce *CrossTracker*, a novel two-stage framework for online multi-modal 3D MOT. CrossTracker first constructs coarse camera and LiDAR trajectories independently, then performs trajectory fusion using both current and historical frames, without requiring future data. This ensures more robust mutual refinement between modalities. Specifically, CrossTracker comprises three core modules: i) the multi-modal modeling (M^3) module, which fuses data from images, point clouds, and even planar geometry derived from images to establish a robust tracking constraint; ii) the coarse trajectory generation (C-TG) module, which independently generates coarse trajectories for both modalities using the M^3 constraint; and iii) the trajectory fusion (TF) module, which applies mutual refinement between coarse LiDAR and camera trajectories through cross correction to ensure robust LiDAR trajectories. Extensive experiments show that CrossTracker outperforms 19 state-of-the-art methods, highlighting its effectiveness in leveraging the synergistic strengths of camera and LiDAR sensors for robust multi-modal 3D MOT. The code is available at https://github.com/lipeng-gu/CrossTracker.**

Received 31 March 2025; revised 14 July 2025 and 2 August 2025; accepted 18 August 2025. Date of publication 22 August 2025; date of current version 5 February 2026. This work was supported in part by the National Defense Basic Scientific Research Program of China under Grant JCKY2020605C003; in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant UGC/FDS16/E03/24; in part by Hong Kong Metropolitan University under Grant RD/2024/1.16; and in part by Changzhou City Science and Technology Project Applied Basic Research under Grant CJ20241078. This article was recommended by Associate Editor Z. Tang. *(Corresponding authors: Xuefeng Yan; Weiming Wang.)*

Lipeng Gu and Mingqiang Wei are with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: glp1224@163.com; mingqiang.wei@gmail.com).

Xuefeng Yan is with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China, and also with the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 211106, China (e-mail: yxf@nuaa.edu.cn).

Weiming Wang is with the School of Science and Technology, Hong Kong Metropolitan University, Kowloon, Hong Kong SAR (e-mail: wmwang@hkmu.edu.hk).

Honghua Chen is with the School of Data Science, Lingnan University, Tuen Mun, Hong Kong SAR (e-mail: honghuachen@LN.edu.hk).

Dingkun Zhu is with the School of Computer Science, Jiangsu University of Technology, Changzhou 213001, China (e-mail: zhudingkun@jsut.edu.cn).

Liangliang Nan is with the Urban Data Science Section, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: liangliang.nan@tudelft.nl).

Digital Object Identifier 10.1109/TCSVT.2025.3601667

***Index Terms*—CrossTracker, multi-modal 3D MOT, two-stage solution, trajectory fusion, cross correction.**

I. Introduction

3D multi object tracking (MOT) is vital for estimating object trajectories in 3D space across applications such as autonomous driving, indoor navigation, and industrial automation [1], [2], [3], [4]. Existing methods are broadly categorized into single- and multi-modal methods. Single-modal methods [5], [6], [7], [8], [9], [10], [11], [12] typically use a single modality (e.g., LiDAR point clouds) for object detection, subsequently associating these detections across frames via carefully designed tracking constraints to construct 3D trajectories. However, their inherent robustness is limited by reliance on a single modality for both detections and constraints. Specifically, detections often suffer from misses or false positives, and tracking constraints frequently lack sufficient discriminability for similar objects. Together, these limitations significantly compromise 3D MOT performance.

To address the challenges outlined above, several multi-modal methods [13], [14], [15], [16], [17], [18], [19] have been proposed. Early methods, known as feature fusion-based methods [16], [17], [18], first perform 3D detection on point clouds and then extract complementary features from both point clouds and corresponding images. These multi-modal features provide robust tracking constraints that effectively reduce trajectory fragmentation, yet they fall short in mitigating tracking failures caused by inaccurate detections, particularly missed detections. Recent detection fusion-based methods [13], [14], [15] focus on mitigating inaccuracies in detection to enhance tracking robustness. These methods observe that the inherent limitations of LiDAR data (e.g., its sparsity and uneven distribution) often render LiDAR detection unreliable for partially occluded or distant objects. In contrast, cameras capture richer semantic cues, offering superior perception in these scenarios. Consequently, these methods incorporate an additional camera modality that acts as a teacher to guide the LiDAR modality (i.e., the student), as illustrated in Fig. 1 (a). Specifically, the process begins by associating reliable LiDAR-camera detections with historical LiDAR trajectories. Next, LiDAR-only detections are matched with remaining unassociated historical trajectories. This two-step procedure minimizes false LiDAR detections and reduces the interference from false camera detections on LiDAR trajectories (see Fig. 2 (a)). Furthermore, camera-only detections are

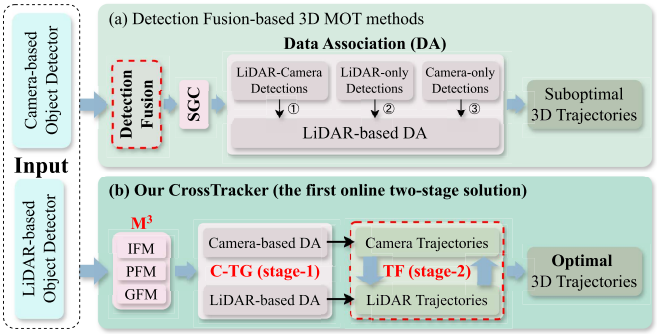

Fig. 1. **Main difference between CrossTracker and its competitors.** Existing methods [13], [14], [15] focus on *detection fusion*, classifying input detections into three groups: LiDAR-camera, LiDAR-only, and camera-only. These detections are then processed sequentially using spatial geometric constraints (SGC, e.g., 3D-IoU), prioritizing LiDAR-camera detections, followed by LiDAR-only, with camera-only detections later correcting LiDAR-induced tracking failures. Nevertheless, this sequential pipeline inherently adheres to a unidirectional, single-stage design and lacks a feedback mechanism whereby the LiDAR-corrected camera modality can iteratively refine LiDAR-based tracking. In contrast, our CrossTracker is the first online two-stage 3D MOT solution that emphasizes *trajectory fusion*, effectively correcting tracking failures across both modalities without depending on future information. It leverages multi-modal modeling (M^3), integrating image, geometric, and point cloud feature modeling (IFM, GFM, and PFM), within a two-stage pipeline comprising coarse trajectory generation (C-TG) and trajectory fusion (TF).

used to fill gaps caused by missed LiDAR detections, thereby preserving trajectory continuity (see Fig. 2 (b)). Compared to feature fusion, detection fusion provides a more explicit and fundamental enhancement in the robustness of 3D MOT.

Despite advancements, detection fusion-based methods are inherently limited by their unidirectional correction from the camera to the LiDAR. This limitation becomes particularly pronounced when the camera misses objects alone (Fig. 2 (c)) or when both modalities simultaneously miss detections (Fig. 2 (c) and (d)). This naturally raises an intriguing question: *Given that 3D trajectories are derived from the LiDAR modality, is it truly necessary to correct inaccuracies in the auxiliary camera detections?* The answer lies in the principle of mutual learning, in which a self-reinforcing guidance loop is formed as the teacher continuously refines its expertise by learning from the student. Within this framework, camera information corrected by LiDAR can, in turn, be more effectively leveraged to refine LiDAR trajectories in subsequent frames. For instance, if an object is detected by LiDAR but missed by the camera in Fig. 2 (c), once LiDAR compensates for it, the corrected camera information can be propagated to Fig. 2 (d), thereby helping to mitigating the case where both modalities fail. This raises another critical question: *Do existing methods [13], [14], [15] overlook the potential for correcting auxiliary camera information, or is their detection-fusion architecture inherently incapable of such correction?* Our analysis reveals that corrected information propagates solely through object trajectories, not through temporally isolated detections across frames. These methods [13], [14], [15] construct complete lifecycle trajectories for LiDAR, while camera detections are only used to generate short-lived tracklets that compensate for temporary LiDAR trajectory interruptions caused by missed detections. These short-lived tracklets behave more like temporally isolated detections rather than complete lifecycle trajectories, and are therefore insufficient for continuously propagating corrected information. Thus, no matter how finely they group LiDAR and camera detections or how sophisticated their tracking constraints are, these methods are fundamentally unable to address the challenges illustrated in Fig. 2 (c) and (d).

To address the challenges outlined in Fig. 2, we introduce *CrossTracker*, a novel two-stage framework for online multi-modal 3D MOT that leverages only current and historical data, without relying on future frames. To our knowledge, CrossTracker represents the first two-stage online 3D MOT solution, comprising three key modules: multi-modal modeling (M^3), coarse trajectory generation (C-TG), and trajectory fusion (TF). It adopts a coarse-to-fine strategy in which C-TG first constructs coarse camera and LiDAR trajectories independently, and subsequently, TF refines these trajectories through cross correction between modalities. This innovative design, featuring an explicit trajectory fusion phase, distinguishes CrossTracker from existing detection fusion-based methods [13], [14], [15]. Moreover, M^3 enforces robust tracking constraints by modeling discriminative multi-modal features, including joint image features, point cloud features, and even planar geometric features extracted from images, to accurately estimate object consistency across frames.

We evaluate CrossTracker on KITTI and six newly constructed adverse-scenario datasets, where it outperforms 19 state-of-the-art competitors. Our contributions are threefold:

- We propose CrossTracker, the first two-stage 3D MOT solution. It employs a coarse-to-fine tracking scheme to address tracking failures caused by detection inaccuracies, significantly enhancing tracking robustness.
- We introduce M^3, a multi-modal modeling module that leverages both camera and LiDAR data to estimate consistency probabilities between objects across frames, thereby providing a robust tracking constraint.
- We propose a distinct two-stage tracking scheme: i) C-TG independently generates coarse trajectories for the camera and LiDAR modalities, and ii) TF refines these trajectories through cross correction, fully leveraging the complementary strengths of both modalities.

II. Related Works

A. Tracking-by-Detection

Existing MOT methods fall into two paradigms: tracking-by-detection (TBD) and joint-detection-tracking (JDT). TBD-based methods [5], [13], [14], [15], [16], [20], [21], [22], [23], [24], [25], [26], [27] decouple detection and tracking into two tasks. They first utilize an off-the-shelf object detector to detect objects, and then generate trajectories based on the resulting detections. JDT-based methods [6], [7], [8], [28], [29], [30], [31], [32], [33], [34] combine both tasks within a unified end-to-end framework, jointly optimizing detection and tracking.

Owing to its flexibility, the TBD paradigm enables the seamless integration of any advanced object detectors [35], [36], [37], [38] to enhance MOT performance, thereby promoting its widespread adoption. A pioneering TBD-based

Fig. 2. **Limitations of existing detection fusion-based 3D MOT methods [13], [14], [15] and advantages of our method.** Existing methods effectively mitigate (a) false detections in either the camera or LiDAR and (b) missed detections solely in LiDAR. However, they struggle with (c) missed detections solely in the camera and (c and d) simultaneous missed detections in both modalities. In contrast, our CrossTracker adopts trajectory fusion, distinct from the detection fusion employed by existing methods. This enables it to effectively address all four challenges through an innovative two-stage tracking scheme.

method is SORT [20], which employs a Kalman filter for object state estimation and uses the Hungarian algorithm with 2D IoU as a tracking constraint to associate detections with trajectories. DeepSORT [21] extends SORT by incorporating appearance features based on deep neural networks [39], enhancing object discriminability. Subsequently, various TBD-based MOT methods [17], [18], [22], [40] further refine this framework.

Given the flexibility of the TBD paradigm, our CrossTracker is built upon this framework and further introduces a two-stage solution, significantly boosting 3D MOT performance.

B. Multi-Modal 3D Multi-Object Tracking

With advancements in 3D perception, TBD-based 3D MOT has achieved significant success [5], [9], [10], [11], [12], [13], [14], [15], [16], [22], [41], [42], [43], [44]. These methods are broadly categorized into single- and multi-modal methods. Single-modal methods [5], [9], [10], [11], [12], [22], [43], [44], like AB3DMOT [5], extend SORT [20] to 3D by incorporating 3D Kalman filtering for 3D motion modeling. However, they are inherently limited by their reliance on a single data source. In contrast, multi-modal methods [13], [14], [15], [16], [17], [18], [45], [46] harness the complementary strengths of LiDAR and camera sensors, significantly enhancing 3D MOT robustness. Furthermore, multi-modal methods are subdivided into feature-fusion and detection-fusion paradigms. Specifically, feature fusion-based methods [16], [17], [18], [45], [46] extract and fuse multi-modal features from camera and LiDAR data to establish robust tracking constraints, reducing the likelihood of tracking failures. Detection fusion-based methods [13], [14], [15], exemplified by EagerMOT [13], combine detections from camera and LiDAR detections to provide higher-quality inputs for 3D MOT, significantly enhancing its robustness.

Compared to feature fusion-based methods, detection fusion-based methods enhance tracking robustness by addressing the root causes of tracking failures. However, as shown in Fig. 1 (a), these methods rely on a unidirectional, single-stage correction from the camera to LiDAR. Lacking an effective trajectory interaction module, they struggle to address the challenges depicted in Fig. 2 (c) and (d). To overcome these limitations, our CrossTracker introduces an innovative two-stage tracking framework that shifts multi-modal fusion from detection fusion to trajectory fusion.

III. Method

A. Overview

We present CrossTracker, the first online, two-stage 3D multi-object tracking (MOT) framework that leverages multi-modal information from both current and historical frames. As shown in Fig. 3, CrossTracker consists of three core modules: i) multi-modal modeling (M^3), ii) coarse trajectory generation (C-TG, stage-1), and iii) trajectory fusion (TF, stage-2). For a given frame (e.g., frame $t-1$) after the initial one, trajectories within each modality can be categorized into three distinct sets: matched trajectories (T^k_{t-1}), unmatched detections (UD^k_{t-1}), and unmatched trajectories (UT^k_{t-1}), where $k \in \{c, l\}$ denotes the camera or LiDAR modality, respectively. Building upon this categorization, CrossTracker simplifies the complex two-stage 3D MOT problem by progressively refining and integrating these sets from both modalities.

Specifically, M^3 integrates image feature modeling (IFM), point cloud feature modeling (PFM), and planar geometric feature modeling (GFM) with a classifier to learn a robust tracking constraint. The first stage, C-TG, is dedicated to *intra-modal matching*, constructing coarse camera and LiDAR trajectories independently. Specifically, it associates detections (D^k_t) at frame t with trajectories (T^k_{t-1}, UD^k_{t-1}, and UT^k_{t-1}) from the preceding frame $t-1$. Consequently, this stage yields updated sets of matched trajectories (T^c_t and T^l_t), along with remaining unmatched detections (UD^c_t and UD^l_t), and remaining unmatched trajectories (UT^c_{t-1} and UT^l_{t-1}). Subsequently, the second stage, TF, is dedicated to *inter-modal*

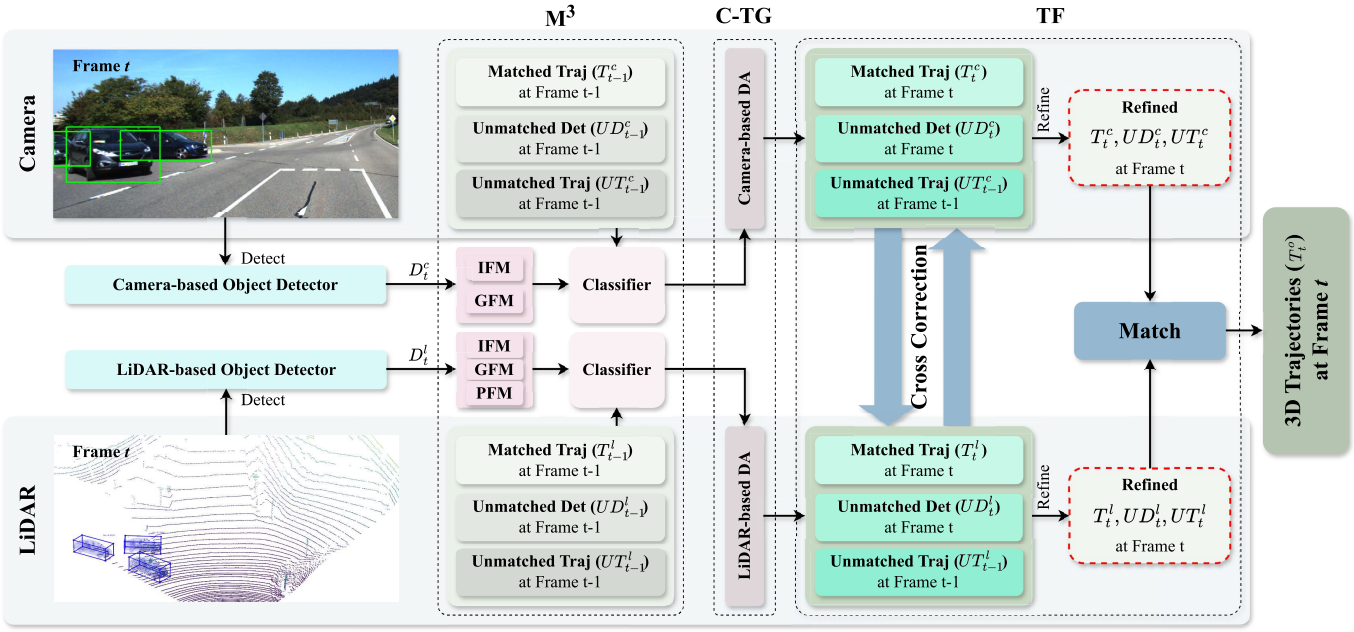

Fig. 3. **Overview of CrossTracker.** Our framework represents the first two-stage multi-modal 3D MOT method, which consists of three key components: i) multi-modal modeling (M^3), ii) coarse trajectory generation (C-TG, stage-1), and iii) trajectory fusion (TF, stage-2). Specifically, M^3 integrates image feature modeling (IFM), point cloud feature modeling (PFM), and planar geometric feature modeling (GFM) with a classifier to learn a robust tracking constraint. Following this, C-TG employs independent data association (DA) pipelines for the camera and LiDAR to construct coarse trajectories. Subsequently, TF refines coarse camera and LiDAR trajectories via cross correction, yielding the final optimized LiDAR trajectories.

matching, which performs trajectory fusion to ensure more robust mutual refinement between modalities, thereby yielding optimal LiDAR trajectories. Specifically, it addresses tracking failures that arise from potential false detections (in UD^c_t and UD^l_t) and missed detections (in UT^l_{t-1} and UT^c_{t-1}) from the first stage. Consequently, this stage generates refined sets for both LiDAR (T^l_t, UD^l_t, and UT^l_t) and camera (T^c_t, UD^c_t, and UT^c_t) modalities. Finally, a set of high-quality trajectories (T^o_t) is outputted by matching the refined LiDAR trajectories (T^l_t) against all refined camera information (T^c_t, UD^c_t, and UT^c_t).

B. Multi-Modal Modeling

Existing methods [13], [14], [15], [16], [17], [18], [20], [21], [22] primarily enhance robustness by modeling more discriminative features for each detected object. Specifically, single-modal methods [5], [6], [7], [12], [13], [14], [15], [20] often employ carefully crafted spatial geometric constraints (SGCs), such as 3D-IoU, for associating detections with trajectories. However, they are prone to association failures when dealing with objects that have similar positions but distinct appearances. To mitigate this limitation, multi-modal methods [13], [14], [15], [16], [17], [18], [22], [45], [46] incorporate deep neural networks (DNNs) to extract richer image and point cloud features that complement SGCs. Despite this, they often overlook integrating geometric information into the DNNs for end-to-end fusion with image and point cloud features, which could significantly simplify multi-modal feature modeling.

To address the aforementioned limitations, we propose M^3, a novel multi-modal modeling network. As illustrated in Fig. 4, M^3 is the first unified end-to-end network that integrates image, planar geometry, and point cloud modeling, along with consistency probabilistic estimation. Specifically, M^3 includes four components: an image feature modeling (IFM) module for image features (F_{img}), a planar geometric feature modeling (GFM) module for planar geometry features (F_{pg}), a point cloud feature modeling (PFM) for point cloud features (F_{pc}), a classifier module for estimating consistency probability, and a spatial geometric constraint (SGC) module for further refining this estimation during inference. Notably, the M^3 module can also be configured to exclusively use the camera sensor for modeling image features and planar geometric features.

1) Image Feature Modeling (IFM): IFM is instrumental in modeling image features. To maintain real-time performance, we employ a *ResNet-18* [39] with a *MaxPooling* layer. The module inputs an image patch and outputs a compact feature vector. Specifically, each image patch is cropped from a designated region based on the provided 2D bounding box, which may stem from a 2D detection or the projection of a 3D detection. The patch is resized to 80×80 pixels before the IFM module extracts a feature vector, denoted as $F_{img} \in \mathbb{R}^{1\times512}$.

2) Point Cloud Feature Modeling (PFM): PFM is a cornerstone for modeling point cloud features. Leveraging the lightweight and efficient *PointNet* [47] with a *MaxPooling* layer, it effectively captures the intricate characteristics of point clouds. Specifically, PFM inputs a point cloud patch $P \in \mathbb{R}^{N\times c}$, where N and c are the number of points and channels, respectively. This patch is generated from 3D bounding boxes and resampled to a fixed size of N points. The module then outputs a feature vector, denoted as $F_{pc} \in \mathbb{R}^{1\times512}$.

3) Planar Geometric Feature Modeling (GFM): Regarding geometric information, FANTrack [22] pioneers the extraction of such data by directly feeding bounding box parameters into

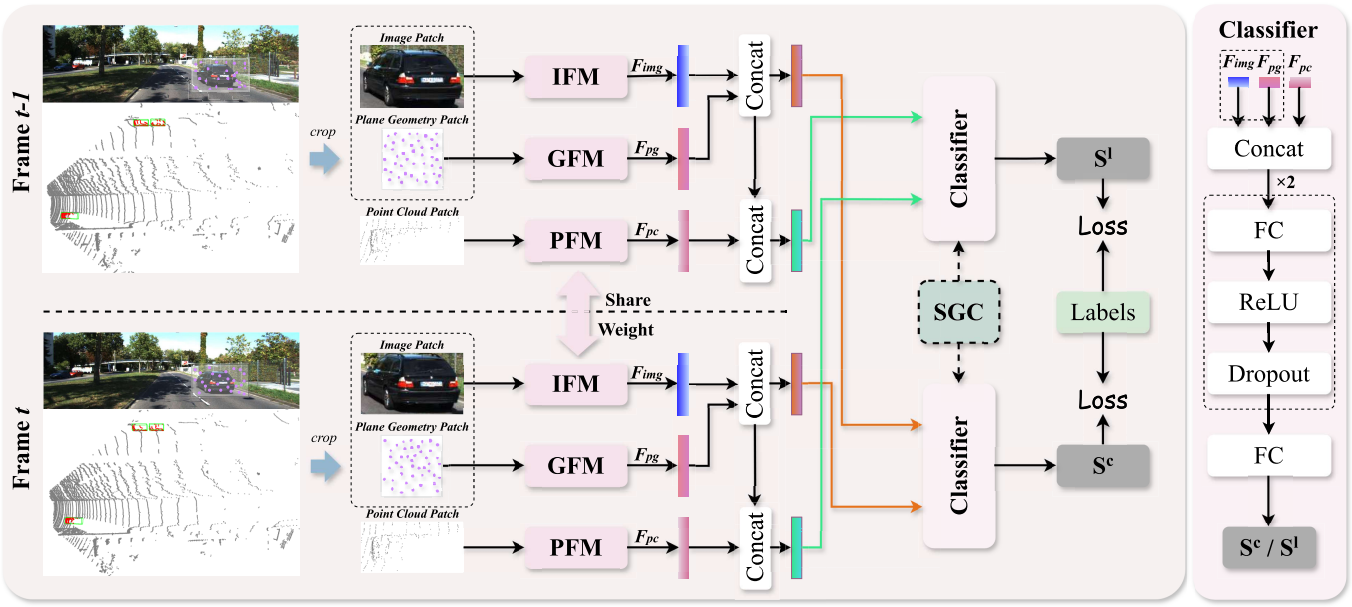

Fig. 4. **Overview of the M³ module.** It takes two consecutive frames of images and point clouds, along with their corresponding detections, as input. It then independently outputs the cross-frame consistency probability (i.e., similarity score) for objects in both the camera (S^c) and LiDAR (S^l) scenarios. M³ comprises four primary components: the image feature modeling (IFM) module for image features (F_{img}), the planar geometric feature modeling (GFM) module for planar geometry features (F_{pg}), and the point cloud feature modeling (PFM) module for point cloud features (F_{pc}), and the classifier for estimating similarity scores. Furthermore, SGC incorporates spatial geometric constraints during inference to further refine the similarity scores output by the classifier.

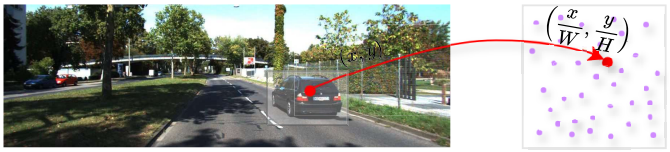

Fig. 5. **An example of planar geometry patch generation.** In the image above, the width and height are denoted as W and H, respectively. Within the detected bounding box for a object, N points are randomly sampled, and each sampled point (x, y) is normalized to $\left(\frac{x}{W}, \frac{y}{H}\right)$.

the DNNs. However, their approach of solely relying on center coordinates and side lengths limits both feature expressiveness and interpretability. To overcome this limitation, we propose a novel planar geometry patch generation method (see Fig. 5). This method transforms the bounding box into visually interpretable pseudo-point patches, represented as $P \in \mathbb{R}^{N\times 2}$, where N is the number of points randomly sampled within the 2D bounding box, and 2 represents the normalized horizontal and vertical coordinates of each point. The resulting patch is then directly fed into a *PointNet* with a *MaxPooling* layer, which outputs a feature vector denoted as $F_{pg} \in \mathbb{R}^{1\times 512}$.

4) Classifier: We reframe the complex task of estimating object consistency probability as a binary classification problem: determining whether two objects from different frames correspond to the same object, using *Cross Entropy Loss*. As shown in Fig. 4, to maintain a lightweight design, the *Classifier* is constructed with only three *FC* layers, along with two *Dropout* and two *ReLU* layers. Specifically, the classifier takes as input a concatenated feature vector, denoted as $concat(F_{img}, F_{pg}, F_{sg}) \in \mathbb{R}^{1\times 1536}$, and outputs the consistency probability S^l. Additionally, the classifier also supports the camera-only scenario. In this case, it processes a different concatenated feature vector, denoted as $concat(F_{img}, F_{pg}) \in \mathbb{R}^{1\times 1024}$, to output the consistency probability S^c. Given the output consistency probability from the classifier, the cost matrix of M³ between the M_{t-1} trajectories (including T_t, UD_{t-1} and UT_{t-1}) in the previous frame $t-1$ and the N_t detections (D_t) in the current frame t is calculated as

$$S^k = \begin{bmatrix} S^k_{1,1} & S^k_{1,2} & \cdots & S^k_{1,N^k_t} \\ S^k_{2,1} & S^k_{2,2} & \cdots & S^k_{2,N^k_t} \\ \vdots & \vdots & \ddots & \vdots \\ S^k_{M^k_{t-1},1} & S^k_{M^k_{t-1},2} & \cdots & S^k_{M^k_{t-1},N^k_t} \end{bmatrix} \tag{1}$$

where $k \in \{c, l\}$ denotes the camera or LiDAR, respectively.

5) Spatial Geometric Constraint (SGC): To further enhance the discriminativeness of features from M^3, we incorporate the spatial geometric constraint (SGC) during inference to refine the cost matrix S (see Eq. 1). Based on empirical findings, we leverage the 2D Intersection over Union (2D-IoU) and the 3D centroid distance (3D-CD) to measure the positional distance between objects for 2D and 3D detections, respectively. To compensate for inter-frame displacements, Kalman Filtering (KF) is adopted to predict historical 2D/3D trajectories, a widely popular method due to its effectiveness in various tracking applications. The KF used here is identical to that in SORT [20] and AB3DMOT [5], and thus will not be elaborated on further. Similar to Eq. 1, the cost matrix of SGC between adjacent frames is defined as

$$G^k = \begin{bmatrix} G^k_{1,1} & G^k_{1,2} & \cdots & G^k_{1,N^k_t} \\ G^k_{2,1} & G^k_{2,2} & \cdots & G^k_{2,N^k_t} \\ \vdots & \vdots & \ddots & \vdots \\ G^k_{M^k_{t-1},1} & G^k_{M^k_{t-1},2} & \cdots & G^k_{M^k_{t-1},N^k_t} \end{bmatrix} \tag{2}$$

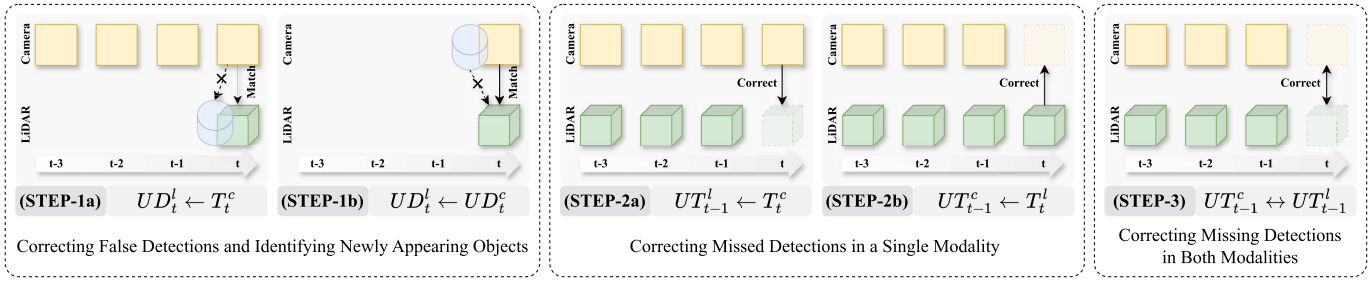

Fig. 6. **Five case studies illustrating *when* and *how* the TF module applies cross correction.** Detected objects are represented by solid shapes, missed objects by dashed lines, and false detections by cylinders. The cross-correction process within TF, which *sequentially executes from STEP-1 to STEP-3*, is applied based on *inter-modal matching combined with the following three empirical observations*: i) If a newly detected object is confirmed to simultaneously appear in both modalities through matching, the likelihood of it being a false detection is reduced, allowing it to be initialized as a new trajectory. ii) If a trajectory in one modality is lost due to a missed detection, it can be recovered by matching it with the corresponding updated trajectory in the other modality. iii) If a trajectory in both modalities is lost, it can be recovered by matching it with the corresponding historical trajectory from the other modality.

where $k \in \{c, l\}$. G^c is the 2D-IoU metric for 2D detections, and G^l is the 3D-CD metric for 3D detections. If G^k is the 2D-IoU metric, then set G^k to 1 minus G^k.

C. Stage-1: Coarse Trajectory Generation (C-TG)

Unlike prior methods [13], [14], [15] that typically rely on a single-stage framework (see Fig. 1 (a)), our CrossTracker marks the pioneering adoption of a two-stage framework (see Fig. 3). Analogous to two-stage object detectors [38], [48] which employ a region proposal network to generate coarse candidate boxes for subsequent refinement, CrossTracker first proposes a coarse trajectory generation (C-TG) module to generate coarse trajectories independently for each modality. Subsequently, a trajectory fusion (TF) module is designed to iteratively refine these coarse trajectories through cross correction between both modalities.

Within the C-TG module, at frame t, detections in D_t are initially ranked in descending order of their detection scores. Subsequently, a total cost matrix C^k is constructed to quantify the association costs between M_{t-1} trajectories (including T_{t-1}, UD_{t-1} and UT_{t-1}) at frame $t-1$ and N_t detections (D_t) at frame t, where $k \in \{c, l\}$. Specifically, the association cost between the $i-$th object from frame $t-1$ and the $j-$th object from frame t, denoted by $C_{i,j}^k$, is defined as

$$C_{i,j}^k = \begin{cases} G_{i,j}^k & , \text{ if } (S_{i,j}^k \geq \theta_S \text{ and } G_{i,j}^k < \theta_G^{loo}) \text{ or } G_{i,j}^k < \theta_G^{str} \\ \infty & , otherwise \end{cases} \quad (3)$$

where θ_S serves as the M^3 threshold for associating objects across consecutive frames. Similarly, θ_G^{loo} and θ_G^{str} are the SGC thresholds for this inter-frame association, where θ_G^{loo} is larger and provides a looser constraint than θ_G^{stri}. Subsequently, a greedy algorithm is employed to associate detections with trajectories, guided by Eq. 3. Accordingly, both modalities yield three sets each: coarse matched trajectories T_t^k, unmatched detections UD_t^k, and unmatched trajectories UT_{t-1}^k. These six sets from both modalities (T_t^c, T_t^l, UD_t^c, UD_t^l, UT_{t-1}^c, and UT_{t-1}^l) are subsequently processed by the TF module, which refines the coarse trajectories through cross correction.

D. Stage-2: Trajectory Fusion (TF)

Our TF module distinguishes itself from existing methods [13], [14], [15] in that it can effectively address all detection inaccuracies (see Fig. 2) that adversely impact trajectory robustness. As illustrated in Fig. 6, the intricate cross-correction mechanism within TF is streamlined into five cases through inter-modal matching, based on empirical observations. These cases are then sequentially addressed in three distinct steps: STEP-1, STEP-2, and STEP-3. Specifically, STEP-1 first confirms the simultaneous presence of new objects (from UD_t^c and UD_t^l) across both modalities via matching, enabling their initialization as new trajectories for integration into T_t^c and T_t^l, respectively. STEP-2 then recovers trajectories (from UT_{t-1}^c and UT_{t-1}^l) lost in one modality due to missed detections. This is achieved by matching them with corresponding updated trajectories from the other modality, facilitating their transfer to T_t^c and T_t^l, respectively. Finally, STEP-3 recovers trajectories (from UT_{t-1}^c and UT_{t-1}^l) lost in both modalities. This involves matching them with historical trajectories from the other modality, allowing their subsequent transfer to T_t^c and T_t^l, respectively. Building upon this three-step cross-correction mechanism, the significantly improved LiDAR and camera data facilitate the output of high-quality 3D trajectories (T_t^o).

INPUT: Prior to cross correction, T_t^l and T_t^c are matched using the 2D-IoU criterion with the threshold θ_{iou}. Subsequently, matched trajectories within T_t^l and T_t^c are excluded from the cross-correction process to streamline its operation.

STEP-1: *Unmatched detections in UD_t^l represent either false detections or newly appearing objects (see cases (STEP-1a) and (STEP-1b) in Fig. 6).* To identify the latter, a greedy algorithm matches detections in UD_t^l with those in T_t^c using 2D-IoU, discarding pairs below threshold θ_{iou}. Paired detections in UD_t^l are initialized as new trajectories, added to T_t^l, and then removed from UD_t^l. Similarly, remaining detections in UD_t^l and UD_t^c are paired, initialized as new trajectories, added to T_t^l and T_t^c, respectively, and then removed from their respective sets. *Note that any detections in UD_t^l and UD_t^c that remain unobserved and uncorrected for N consecutive frames are classified as false detections and discarded.*

STEP-2: *Unmatched trajectories in UT^c_{t-1} and UT^l_{t-1} can arise from trajectory termination or missed detections in one or both modalities.* Given that missed detections in one modality are more easily corrected, we prioritize cases (STEP-2a) and (STEP-2b) in Fig. 6. Initially, the KF model predicts the states of trajectories in UT^l_{t-1} at frame t, yielding $\hat{UT}^l_t$. Unmatched trajectories in $\hat{UT}^l_t$ are then matched with those in T^c, following the procedure in **STEP-1**. Beyond the 2D-IoU criterion, we further stipulate that both paired trajectories must have at least θ_{hits} consecutive observations (*hits*). For each pair $(\hat{\mathbf{ut}}^l_t, \mathbf{ut}^c_t)$ meeting these criteria, the corresponding $\mathbf{ut}^l_{t-1}$ is updated to frame t using $\hat{\mathbf{ut}}^l_t$, added to T^l_t, and removed from UT^l_{t-1}. Similarly, for each paired $(\hat{\mathbf{ut}}^c_t, \mathbf{t}^l_t)$, the corresponding $\mathbf{ut}^c_{t-1}$ is updated to frame t using $\hat{\mathbf{ut}}^l_t$, added to T^c_t, and removed from UT^c_{t-1}.

STEP-3: *Remaining unmatched trajectories in UT^c_{t-1} and UT^l_{t-1} may arise from simultaneous termination or missed detections in both modalities (see the case (STEP-3) in Fig. 6).* Based on predicted $\hat{UT}^c_t$ and $\hat{UT}^l_t$ at frame t from UT^c_{t-1} and UT^l_{t-1}, unmatched trajectories in both sets are matched using the procedure in **STEP-2**. Beyond the 2D-IoU and *hits* criteria, we further stipulate that both predicted trajectories must not be at the image plane boundary. This constraint effectively determines whether trajectories have terminated. For each pair $(\hat{\mathbf{ut}}^c_t, \hat{\mathbf{ut}}^l_t)$ meeting these criteria, the corresponding $\mathbf{ut}^c_{t-1}$ and $\mathbf{ut}^l_{t-1}$ are updated to frame t using $\hat{\mathbf{ut}}^c_t$ and $\hat{\mathbf{ut}}^l_t$, respectively, added to T^c_t and T^l_t, and removed from UT^c_{t-1} and UT^l_{t-1}. *Note that any trajectories in UT^c_{t-1} and UT^l_{t-1} that remain unobserved and uncorrected for N consecutive frames are deemed terminated and discarded.*

OUTPUT: After cross correction, detection inaccuracies at frame t are significantly mitigated. Since the camera provides finer-grained semantic information and thus yields more reliable detections, we associate the updated LiDAR trajectories T^l_t with the updated camera set $(T^c_t \cup UD^c_t \cup UT^c_t)$ using the 2D-IoU criterion with the threshold θ_{iou}. Notably, UD^c_t and UT^c_t are still included because stringent cross-correction constraints can prevent the correction of some newly detected or reliably unmatched trajectories. For each paired $(\mathbf{t}^l_t, \mathbf{t}^c_t)$, the LiDAR trajectory $\mathbf{t}^l_t$ is added to the high-quality 3D output set T^o_t. Additionally, to handle cases where camera detections degrade under adverse conditions, any LiDAR trajectory $\mathbf{t}^l_t$ that has been successfully tracked for at least three consecutive frames is also deemed reliable and included in T^o_t.

Remark: Within our framework, the decision to discard a new object or terminate a trajectory relies solely on two conditions: i) failure to track in its respective modality for θ_{age} consecutive frames, and ii) lack of correction by the other modality during that period. Consequently, in certain scenarios (e.g., low-light conditions where LiDAR more reliably captures objects than cameras), a legitimate new object detected by only one modality will never be discarded as long as it can be continuously tracked within its respective modality. Once the other modality detects it, the object is confirmed reliable through cross correction (see Fig. 6 (STEP-1a)) and outputted. Furthermore, in the extreme case where the camera fails, legitimate new objects cannot be confirmed through cross correction and thus cannot be output. Nevertheless, our method can directly output the LiDAR trajectories generated in the first stage and still deliver strong performance (see the first row of Table V).

IV. Experiments

A. Dataset

To evaluate the performance of CrossTracker, we leverage the widely recognized KITTI tracking benchmarks. The KITTI dataset comprises 21 training sequences and 29 test sequences, totaling 8008 and 11095 frames, respectively. Each frame includes a corresponding point cloud and an RGB image. Specifically, our CrossTracker is evaluated on the *Car* and *Pedestrian* categories of the KITTI dataset. Furthermore, following existing methods [13], [14], [15], the KITTI training set is partitioned into two subsets: a training set consisting of sequences *0000, 0002, 0003, 0004, 0005, 0007, 0009, 0011, 0017, 0020* and a validation set comprising sequences *0001, 0006, 0008, 0010, 0012, 0013, 0014, 0015, 0016, 0018, 0019*.

To further validate the generalizability of CrossTracker, we construct six variant datasets from KITTI to simulate various adverse environmental conditions: 1) KITTI-F, simulating fog for both LiDAR and camera; 2) KITTI-MB, simulating motion blur for both LiDAR and camera; 3) KITTI-L-CS, simulating cross-sensor interference for LiDAR; 4) KITTI-L-BM, simulating beam missing for LiDAR; 5) KITTI-C-IN, simulating impulse noise for the camera; and 6) KITTI-C-LI, simulating low illumination for the camera. Given that the KITTI test set lacks ground truth labels and necessitates submissions to its official website for evaluation, which has a restricted number of permissible attempts, we construct the aforementioned six datasets using the KITTI training and validation subsets. Our future work will focus on the collection and annotation of datasets for these challenging scenarios.

B. Camera- and LiDAR-Based Object Detectors

To ensure a fair and comprehensive evaluation, we conduct experiments with various combinations of camera- and LiDAR-based object detectors. Specifically, we utilize Point-GNN [37], PointRCNN [38], CasA [35], and VirConv [36] as LiDAR-based object detectors, and utilize RRC [49], Perma [51], Track-RCNN [50] as camera-based object detectors.

C. Evaluation Metrics and Experimental Setup

We evaluate CrossTracker using the official toolkit with both CLEAR MOT [56] and HOTA [57] metrics. For CLEAR, we report MOTA, sMOTA, MOTP, and IDSW, while HOTA, along with its DetA and AssA components, provides a unified measure of detection and association quality.

Implemented in Python with PyTorch, CrossTracker is tested on an Intel Core i9 3.70 GHz CPU, 128 GB RAM, and a GTX 4090 GPU. The M^3 module is trained for 15 epochs with a $1e^{-4}$ learning rate and $5e^{-5}$ weight decay, simultaneously optimizing three classifiers corresponding to the three scenarios presented in Table III.

TABLE I

COMPARISON OF 3D MOT METHODS ON THE KITTI TEST SET.METHODS WITH THE SUPERSCRIPT "L" DENOTE LIDAR-BASED DETECTORS, WHILE THOSE WITH "C" DENOTE CAMERA-BASED OBJECT DETECTORS.SPECIFICALLY, "L1", "L2", AND "L3" UTILIZE POINTGNN [37] FOR BOTH CATEGORIES, CASA [35] (FOR CARS), AND VIRCONV [36] FOR CARS, RESPECTIVELY.SIMILARLY, "C1" UTILIZES RRC [49] FOR CARS AND TRACK-RCNN [50] FOR PEDESTRIANS; "C2" EMPLOYS PERMA [51] FOR CARS; AND "C3" UTILIZES FASTER R-CNN [48] FOR CARS.CASTRACK IS EVALUATED IN ONLINE MODE FOR FAIR COMPARISON, WHILE OTHER RESULTS ARE OBTAINED FROM THE OFFICIAL KITTI BENCHMARK

Method	Input	Car						Pedestrian					
		HOTA	DetA	AssA	MOTA	IDWS	sMOTA	HOTA	DetA	AssA	MOTA	IDWS	sMOTA
FAMNet [9]	C	52.56	61.00	45.51	75.92	521	59.02	-	-	-	-	-	-
LGM [10]	C	73.14	74.61	72.31	87.60	448	72.76	-	-	-	-	-	-
DEFT [11]	C	74.23	75.33	73.79	88.38	344	74.04	-	-	-	-	-	-
TripletTrack [44]	C	73.58	73.18	74.66	84.32	522	72.26	42.77	39.54	46.54	50.08	323	**35.71**
PolarMOT [12]	L	75.16	73.94	76.95	85.08	462	71.82	43.59	39.88	**48.12**	46.98	270	24.61
FANTrack [22]	L	60.85	64.36	58.69	75.84	743	61.49	-	-	-	-	-	-
AB3DMOT [5]	L	69.99	71.13	69.33	83.61	113	70.80	37.81	32.37	44.33	38.13	**181**	20.80
CasTrack^{L2} [43]	L	77.32	75.08	80.32	86.33	184	74.27	-	-	-	-	-	-
CasTrack^{L3} [43]	L	79.97	77.94	82.67	89.19	201	77.07	-	-	-	-	-	-
BeyondPixels [52]	C+L	63.75	72.87	56.40	82.68	934	69.98	-	-	-	-	-	-
mmMOT [16]	C+L	62.05	72.29	54.02	83.23	733	70.12	-	-	-	-	-	-
JRMOT [18]	C+L	69.61	73.05	66.89	85.10	271	72.11	34.24	38.79	30.55	45.31	631	29.78
MOTSFusion [53]	C+L	68.74	72.19	66.16	84.24	415	71.14	-	-	-	-	-	-
JMODT [17]	C+L	70.73	73.45	68.76	85.35	350	72.19	-	-	-	-	-	-
DeepFusionMOT [14]	C+L	75.46	71.54	80.05	84.63	84	71.66	-	-	-	-	-	-
StrongFusionMOT [15]	C+L	75.65	72.08	79.84	85.53	58	72.62	43.42	38.86	48.83	39.04	316	16.54
PNAS-MOT [46]	C+L	67.32	77.69	58.99	89.59	751	-	-	-	-	-	-	-
BcMODT [54]	C+L	71.00	73.62	69.14	85.48	381	72.22	-	-	-	-	-	-
EagerMOT^{L1C1} [13]	C+L	74.39	75.27	74.16	87.82	239	74.97	39.38	40.60	38.72	49.82	496	28.01
MMF-JDT^{L2C3} [45]	C+L	79.52	75.83	84.01	88.06	**38**	-	-	-	-	-	-	-
CrossTracker^{L1C1}	C+L	78.02	75.85	80.88	89.52	103	75.80	**45.64**	**44.34**	47.41	**57.15**	289	33.57
CrossTracker^{L2C2}	C+L	80.64	77.91	84.10	90.17	135	77.38	-	-	-	-	-	-
CrossTracker^{L3C2}	C+L	**82.34**	**80.14**	**85.24**	**92.13**	92	**79.89**	-	-	-	-	-	-

D. Main Results

1) Quantitative Evaluation on KITTI: We employ two baseline methods, EagerMOT [13] and CasTrack [43], and conduct comparative experiments on the KITTI test set using three different combinations of camera- and LiDAR-based detectors to highlight the superiority of our CrossTracker.

Table I shows that CrossTracker consistently outperforms EagerMOT [13] across all metrics, particularly for challenging pedestrians. Quantitatively, for cars, CrossTracker improves HOTA, MOTA, IDWS, and sMOTA by 3.63%, 1.70%, 136, and 0.83%, respectively. Similarly, for pedestrians, CrossTracker improves HOTA, MOTA, IDWS, and sMOTA by 6.26%, 7.33%, 207, and 5.56%, respectively. Crucially, both CrossTracker and EagerMOT utilize the same 2D detections from RRC [49] and Track-RCNN [50] and the same 3D detections from Point-GNN [37] as inputs. However, EagerMOT incorporates additional 2D segmentation information, while CrossTracker does not. Despite this, CrossTracker remains significantly superior to EagerMOT. This compelling performance demonstrates the effectiveness of our two-stage framework in tackling all challenges shown in Fig. 2, particularly challenges (c) and (d), which EagerMOT fails to handle.

To further evaluate CrossTracker, we conduct a comparative analysis with the state-of-the-art CasTrack [43], employing the same LiDAR-based detectors (CasA [35] or VirConv [36]) for both methods. Experimental results reveal that CrossTracker surpasses CasTrack in online performance. Specifically, when utilizing CasA [35] and VirConv [36] as LiDAR-based detectors, CrossTracker achieves a 3.32% and 2.37% improvement in the HOTA metric, respectively. Notably, when utilizing VirConv [36] as the LiDAR-based detector and Perma [51] as the camera-based detector, our method achieves state-of-the-art performance on the KITTI leaderboard, securing first place in both DetA and MOTA metrics, with scores of 80.14% and 92.13%, respectively. These improvements underscore the significant enhancement of 3D trajectory output attributable to the cross-correction mechanism within the TF module. Furthermore, our method maintains strong competitiveness, particularly in terms of the HOTA metric, when compared to 17 other state-of-the-art methods.

2) Quantitative Evaluation Under Adverse Conditions: To further validate the generality and robustness of CrossTracker, we conduct comparative experiments on six newly constructed adverse-condition datasets, using 3DSSD [55] as the 3D detector and YOLOv5l as the 2D detector. As shown in Table II, CrossTracker is evaluated against the single-modal AB3DMOT and the multi-modal DeepFusionMOT [14].

Under clean input conditions, CrossTracker surpasses both DeepFusionMOT and AB3DMOT, with DeepFusionMOT outperforming AB3DMOT. This demonstrates the superiority of the detection fusion-based framework over the single-modality LiDAR-based framework, and highlights the advantage of our novel two-stage 3D MOT framework over the detection fusion-based framework. Moreover, CrossTracker consistently demonstrates significant superiority over both AB3DMOT and DeepFusionMOT across all six adverse-condition datasets, for both cars and pedestrians. While motion blur (in both LiDAR and camera) and cross-sensor interference (in LiDAR) cause the most significant performance degradation for CrossTracker, it remains robust under the other four adverse conditions. For instance, under low-illumination scenarios, the HOTA scores for cars and pedestrians drop by only 2.02% and

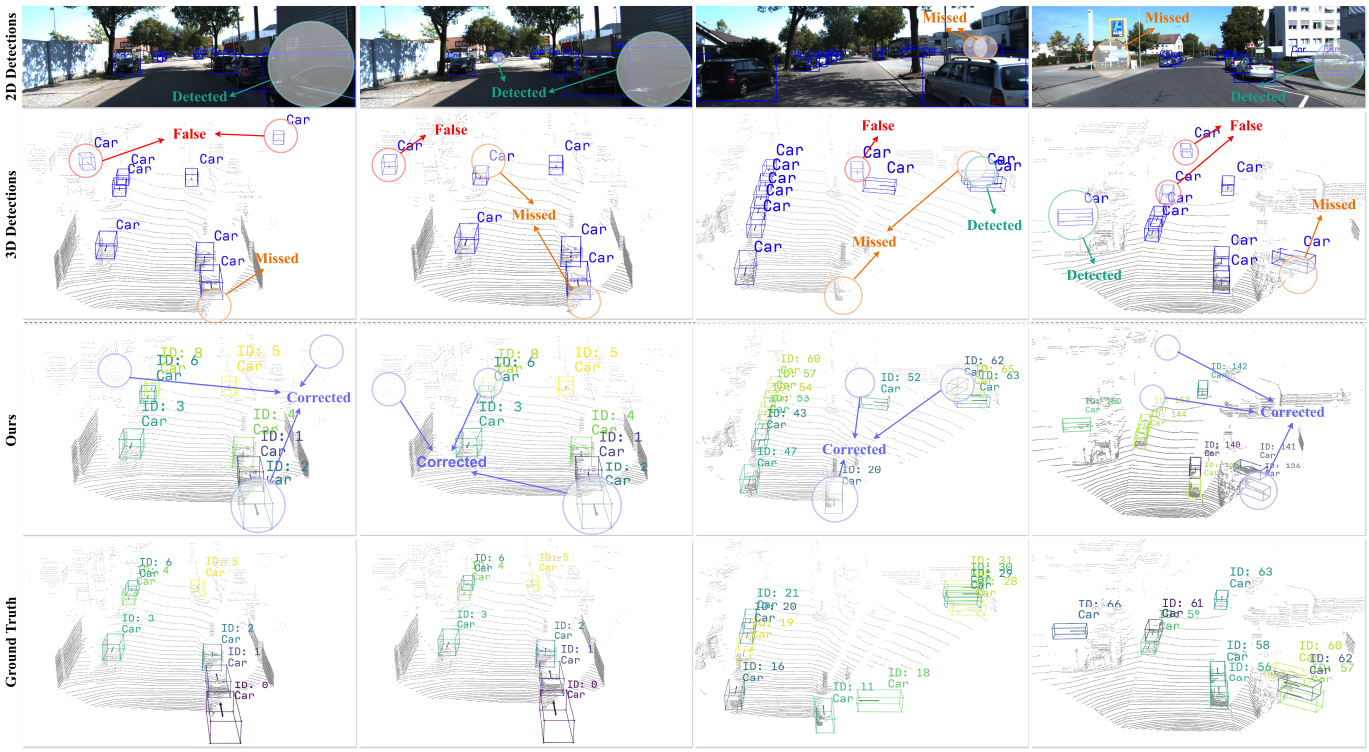

Fig. 7. **Qualitative evaluation of our CrossTracker on the sequence *0001* of the KITTI validation dataset.** The first and second rows show the detections from RRC [49] and Point-GNN [37], respectively. The third and fourth rows show tracking results from our CrossTracker and the ground truth, respectively.

TABLE II

COMPARISON OF 3D MOT METHODS UNDER ADVERSE CONDITIONS ON KITTI-F, KITTI-MB, KITTI-L-CS, KITTI-L-BM, KITTI-C-IN, AND KITTI-C-LI DATASETS. ALL EXPERIMENTS ADOPT 3DSSD [55] AS THE 3D DETECTOR AND YOLOV5L AS THE 2D DETECTOR. COMPARED METHODS INCLUDE AB3DMOT [5] (DENOTED AS ABM) AND DEEPFUSIONMOT [14] (DENOTED AS DFM)

Method	LiDAR					Camera					Car			Pedestrian		
	Clean	Fog	Motion Blur	Cross Sensor	Beam Missing	Clean	Fog	Motion Blur	Impulse Noise	Low Illumination	HOTA	DetA	AssA	HOTA	DetA	AssA
ABM [5]											72.24	68.35	76.68	41.19	35.78	**47.59**
DFM [14]	✓					✓					75.37	72.87	78.29	42.99	40.46	45.86
Ours											**76.87**	**73.92**	**80.30**	**44.29**	**41.99**	46.90
ABM [5]											67.75	67.21	72.73	32.12	26.19	39.63
DFM [14]		✓					✓				69.83	64.64	75.69	34.03	29.10	40.01
Ours											**72.96**	**69.41**	**77.03**	**35.02**	**29.52**	**41.74**
ABM [5]											64.64	62.42	67.04	33.77	30.42	37.56
DFM [14]			✓					✓			61.64	57.31	66.38	26.28	24.34	28.48
Ours											**66.51**	**62.68**	**70.69**	**35.13**	**32.29**	**38.30**
ABM [5]											55.40	50.21	61.56	39.03	34.92	43.83
DFM [14]				✓		✓					63.22	60.46	66.51	40.57	38.48	43.00
Ours											**66.44**	**64.21**	**69.23**	**42.28**	**40.34**	**44.49**
ABM [5]											60.66	61.77	60.07	37.17	35.42	39.28
DFM [14]					✓	✓					67.89	66.16	70.03	39.38	37.65	41.42
Ours											**72.08**	**69.84**	**74.77**	**40.93**	**39.83**	**42.26**
ABM [5]											72.24	68.35	76.68	41.19	35.78	**47.59**
DFM [14]	✓								✓		67.43	61.79	73.79	30.30	24.13	38.16
Ours											**73.36**	**67.39**	**80.15**	**42.08**	**38.33**	46.39
ABM [5]											72.24	68.35	76.68	41.19	35.78	**47.59**
DFM [14]	✓									✓	69.00	63.55	75.16	40.38	36.90	44.37
Ours											**74.88**	**70.00**	**80.43**	**43.26**	**39.82**	47.21

0.96%, respectively. In comparison, DeepFusionMOT shows consistently limited robustness across all six scenarios, with the HOTA score for cars dropping by at least 5.5%.

Another important observation is that under motion blur (in both LiDAR and camera), impulse noise (in camera data), and low illumination (in camera data), DeepFusionMOT performs even worse than AB3DMOT. This suggests that although DeepFusionMOT incorporates camera detections to supplement LiDAR, its detection fusion-based framework does not decouple the two modalities. As a result, it is effective only when camera inputs are clean; otherwise, noisy detections from degraded images can negatively impact overall performance. In contrast, our method exhibits significantly higher robustness under both clean and adverse conditions.

3) Qualitative Evaluation: Fig. 7 further illustrates CrossTracker's capability to correct numerous false and missed detections across both modalities. For instance, camera data effectively corrects false LiDAR detections (columns 1-4) and recovers missed LiDAR detections (columns 1, 2, and 4). Conversely, LiDAR information compensates for missed camera detections (column 4). Remarkably, CrossTracker even

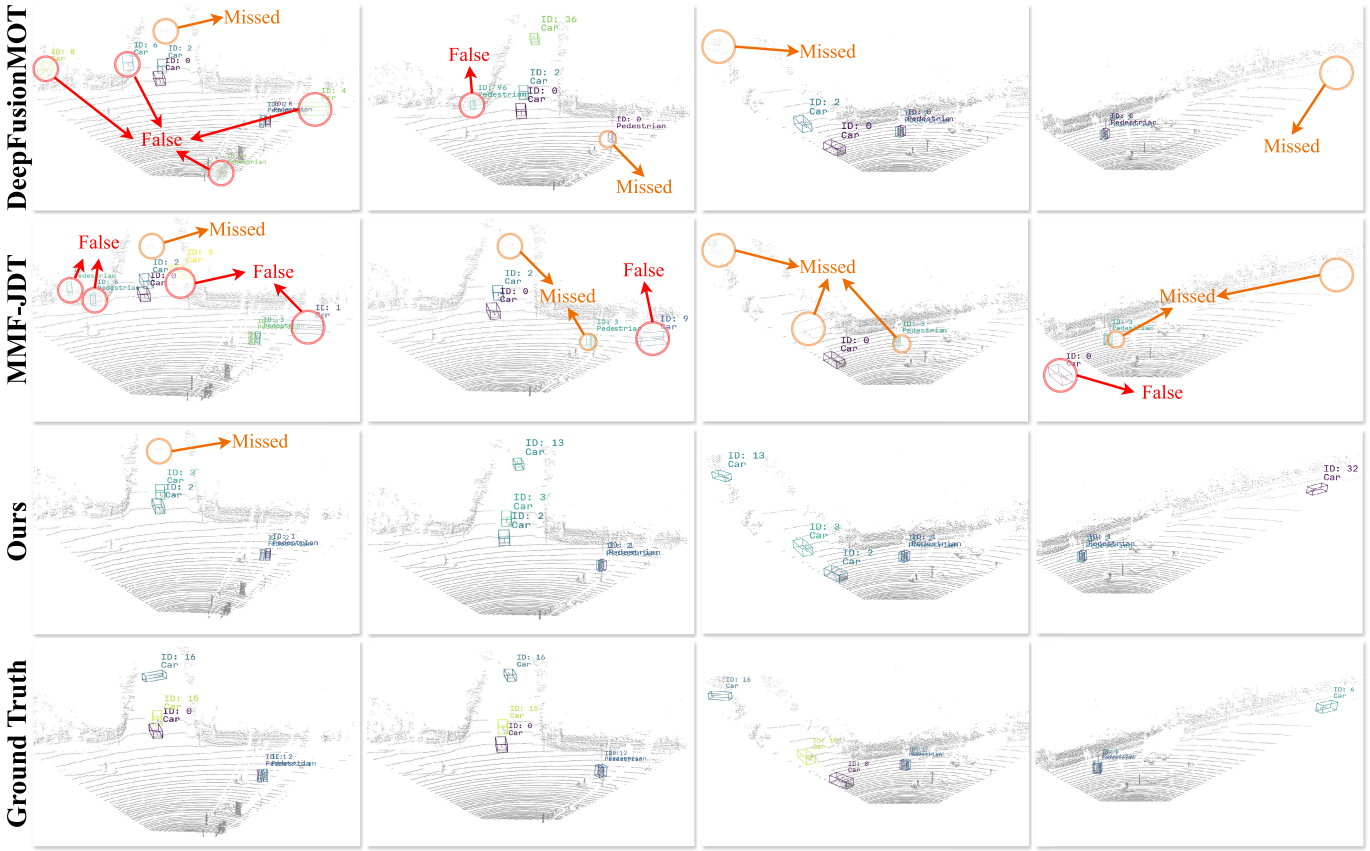

Fig. 8. **Qualitative comparison of our CrossTracker with other state-of-the-art multi-modal methods on the sequence *0014* of the KITTI validation set.** For a fair comparison, CrossTracker, DeepFusionMOT [14], and MMF-JDT [45] use PointRCNN [38] as the 3D detector. Additionally, CrossTracker and DeepFusionMOT use RRC [49] as the 2D detector, while MMF-JDT adheres to its default configuration, which uses Faster R-CNN [48].

TABLE III

ABLATION STUDY OF THE CLASSIFIER WITHIN THE M^3 MODULE FOR ESTIMATING INTER-OBJECT CONSISTENCY ACROSS DIFFERENT MODALITY FEATURES ON THE KITTI VALIDATION SET FOR CARS AND PEDESTRIANS

M^3			F1_score	Precision	Recall
IFM	GFM	PFM			
✓			91.91	88.98	95.58
✓	✓		93.20	89.74	97.70
✓	✓	✓	**93.35**	**89.93**	**97.77**

achieves accurate tracking in challenging cases where both modalities fail (column 3, the car on the far right). Collectively, these results underscore CrossTracker's exceptional ability to overcome complex challenges, ensuring robust 3D MOT.

Furthermore, Fig. 8 qualitatively compares our CrossTracker with state-of-the-art multi-modal methods, such as DeepFusionMOT [14] and MMF-JDT [45]. The four illustrated columns consistently demonstrate CrossTracker's significant advantages in mitigating tracking failures caused by both missed and false detections. For instance, the first column highlights CrossTracker's robust ability to overcome false detections, while DeepFusionMOT and MMF-JDT introduce numerous erroneous trajectories. Simultaneously, columns two, three, and four collectively illustrate CrossTracker's capacity to maintain continuous tracking even for long-distance objects, where DeepFusionMOT and MMF-JDT fail.

E. Ablation Study

To systematically analyze the contributions of the proposed M^3 and TF modules, and the influence of various object detectors on the performance of CrossTracker, we conduct ablation experiments on the KITTI validation set.

1) Effects of M^3: Table III demonstrates that each feature (IFM, GFM, and PFM) substantially enhances inter-object consistency estimation. More specifically, with only image data, the classifier achieves the lowest performance, with an F1-score of 91.91%, Precision of 88.98%, and Recall of 95.58%. Adding planar geometric features improves these metrics by 1.29%, 0.76%, and 2.12%, respectively. Incorporating point cloud features achieves the highest scores of 93.35% for F1-score, 89.93% for Precision, and 97.77% for Recall.

Table IV further demonstrates that the integration of all three modalities consistently boosts 3D MOT. Specifically, for cars, image features contribute most, leading to a HOTA improvement of up to 2.50%. For pedestrians, planar geometric features are the most significant contributor, improving HOTA by 0.31%. Furthermore, the joint utilization of all three features leads to HOTA score improvements of 2.52% for cars and 0.51% for pedestrians. Additionally, following AB3DMOT [5], we incorporate ego-motion compensation

TABLE IV

ABLATION STUDY OF THE M^3 MODULE AND EGO-MOTION COMPENSATION (EMC) ON THE KITTI VALIDATION SET.EXPERIMENTS EMPLOY RRC [49] FOR CARS, TRACK-RCNN [50] FOR PEDESTRIANS, AND POINT-GNN [37] FOR BOTH.EMC IS EXCLUSIVELY APPLIED TO 3D TRAJECTORIES

M^3			EMC	Car			Pedestrian		
IFM	GFM	PFM		HOTA	DetA	AssA	HOTA	DetA	AssA
				81.79	79.96	83.93	51.73	50.55	53.25
✓				84.29	82.82	86.03	51.83	50.61	53.38
✓	✓			84.29	82.75	86.10	52.14	50.69	53.92
✓	✓	✓		84.31	82.81	86.08	52.24	50.69	54.14
✓	✓	✓	✓	**84.49**	**82.90**	**86.36**	**52.84**	**50.71**	**55.34**

TABLE V

ABLATION STUDY OF THE TF MODULE ON THE KITTI VALIDATION SET.THE TF MODULE EXECUTES A THREE-STEP PIPELINE SHOWN IN FIG. 6.EXPERIMENTS EMPLOY RRC [49] FOR CARS, TRACK-RCNN [50] FOR PEDESTRIANS, AND POINT-GNN [37] FOR BOTH

TF					Car			Pedestrian		
STEP-1a	STEP-1b	STEP-2a	STEP-2b	STEP-3	HOTA	DetA	AssA	HOTA	DetA	AssA
					77.78	77.94	77.88	49.74	48.23	51.63
✓					79.52	80.49	78.82	50.94	50.45	51.78
✓	✓				79.77	80.79	79.03	50.99	50.47	51.84
✓	✓	✓			82.20	82.39	82.25	**52.84**	**50.77**	55.29
✓	✓	✓	✓		82.39	82.57	82.45	52.83	50.75	55.28
✓	✓	✓	✓	✓	**84.49**	**82.90**	**86.36**	**52.84**	50.71	**55.34**

TABLE VI

ABLATION STUDY OF DIFFERENT CAMERA- AND LIDAR-BASED DETECTORS ON THE KITTI VALIDATION SET FOR CARS

LiDAR-based Detector	Camera-based Detector	HOTA	DetA	AssA	MOTA	sMOTA	MT	ML	FRAG	IDSW
PointRCNN [38]	RRC [49]	80.30	80.09	80.80	93.33	80.10	**169**	2	41	22
Point-GNN [37]	RRC [49]	84.49	82.90	86.36	95.11	83.16	170	3	33	16
CasA [35]	RRC [49]	86.08	85.18	87.19	95.01	85.09	173	2	20	7
VirConv [36]	RRC [49]	87.67	**88.19**	87.28	**95.69**	**87.65**	174	2	15	11
PointRCNN [38]	Perma [51]	80.83	80.21	81.74	92.85	79.47	174	**0**	34	69
Point-GNN [37]	Perma [51]	81.16	82.58	86.02	94.69	82.48	175	**0**	31	19
CasA [35]	Perma [51]	85.61	84.56	86.85	94.09	84.15	178	**0**	16	**5**
VirConv [36]	Perma [51]	**88.19**	87.54	**88.98**	94.92	86.88	176	**0**	**13**	**5**

(EMC) for 3D trajectories, which further boosts the HOTA scores by 0.18% for cars and 0.6% for pedestrians.

2) Effects of TF: Table V clearly demonstrates how the TF module significantly enhances the quality of the output 3D trajectories by sequentially addressing the five challenging cases (as illustrated in Fig. 6). Specifically, when using only the LiDAR stream for single-stage 3D MOT, the tracking performance for both cars and pedestrians is the lowest, with HOTA metrics of 77.78% and 49.74%, respectively. Sequentially addressing these five cases leads to significant improvements across all metrics for cars. Notably, the HOTA metric increases by 1.74%, 0.25%, 2.43%, 0.19%, and 2.10%, respectively. This highlights the substantial impact of addressing cases (STEP-1a), (STEP-2a), and (STEP-3) on overall performance. While pedestrians also benefit from the proposed TF module, the most significant enhancements are observed after addressing cases (STEP-1a) and (STEP-2a), resulting in HOTA metric improvements of 1.20% and 1.85%, respectively.

3) Effects of Object Detectors: CrossTracker, rooted in the tracking-by-detection paradigm, can be compatible with various camera- and LiDAR-based detectors. To comprehensively analyze the effect of different camera- and LiDAR-based detectors on CrossTracker, we conduct a comparative experiment, as shown in Table VI. Specifically, PointRCNN [38], Point-GNN [37], CasA [35], and VirConv [36] are employed as LiDAR-based detectors, while RCC [49] and Perma [51] serve as camera-based detectors. Note that for all six detectors, the detection results provided by the corresponding authors are directly utilized.

Table VI demonstrates that while the choice of camera- and LiDAR-based detectors can influence the overall performance of CrossTracker, our proposed two-stage paradigm consistently yields more robust and smoother 3D trajectories as indicated by lower IDWS scores. Our findings also highlight the significant impact of the quality of input 3D detections on the final 3D tracking outcomes. Specifically, VirConv, when combined with either RRC or Perma, consistently outperformed other detector combinations, achieving the highest HOTA metrics of 87.67% and 88.19%, respectively. CasA followed closely, yielding HOTA metrics of 86.08% and 85.61% when combined with RRC and Perma, respectively.

F. Computational Efficiency Discussion

Efficient computation is crucial for 3D MOT. We conduct a detailed ablation study to analyze the performance of our method across various metrics, including FPS, CPU/GPU memory footprint, network parameters, and network FLOPs.

Since the M^3 module is the sole neural network-based component in CrossTracker, we first evaluate its efficiency via parameters, FLOPs, and GPU memory footprint. As shown in Table VII, even with three modalities, it incurs just 16.43 M parameters, 8.45 GFLOPs, and 76.71 MB GPU per sample.

TABLE VII
ABLATION STUDY ON THE IMPACT OF MODALITY FEATURES ON PARAMETERS, FLOPS, AND GPU MEMORY FOOTPRINT OF M^3 ON THE KITTI VALIDATION SET FOR CARS

M^3			Params (M)	FLOPs (GFLOPs)	GPU (MB)
IFM	GFM	PFM			
✓			11.46	6.98	54.88
✓	✓		13.36	7.71	62.95
✓	✓	✓	16.43	8.45	76.71

Table VIII further analyzes the computational efficiency of CrossTracker, detailing its FPS and CPU/GPU memory footprint. With the M^3 and TF components disabled, CrossTracker runs at 600 FPS, using 293.74 MB CPU and 0 MB GPU memory. When both M^3 and TF are enabled, the FPS remains high at 78, substantially exceeding the real-time requirement of 25 FPS. In this configuration, the CPU memory footprint modestly increases by 400 MB, and the GPU memory footprint is 4458.26 MB.

Furthermore, CrossTracker delivers highly competitive inference speeds compared to existing multi-modal fusion methods. For example, mmMOT [16], JRMOT [18], MOTS-Fusion [53], MMF-JDT [45], and PNAS-MOT [46] operate at just 4 FPS, 14 FPS, 3 FPS, 5 FPS, and 13 FPS, respectively, because their heavy neural architectures fall well below the 25 FPS real-time threshold. In contrast, owing to the lightweight design of its neural component M^3, CrossTracker achieves 78 FPS. Although EagerMOT [13] (90 FPS), DeepFusionMOT [14] (110 FPS), and StrongFusionMOT [15] (67 FPS) meet real-time requirements without using neural networks during inference, our method still delivers comparable performance.

G. Sensitivity Analysis

1) Sensitivity Analysis of Kalman Filter Initialization: For the initial state, CrossTracker follows prior methods (e.g., SORT [20] and AB3DMOT [5]) to directly initialize the state with the bounding box from the trajectory's initial frame.

Furthermore, given the acknowledged significant impact of the covariance matrix configuration (the state covariance matrix P, the measurement noise covariance matrix R, and the process noise covariance matrix Q) on tracking performance, we conduct a comprehensive sensitivity analysis, as detailed in Table IX. Specifically, for 3D objects, the matrices $P \in \mathbb{R}^{10\times10}$, $R \in \mathbb{R}^{7\times7}$, and $Q \in \mathbb{R}^{10\times10}$ are defined. Conversely, for 2D objects, the matrices $P \in \mathbb{R}^{7\times7}$, $R \in \mathbb{R}^{4\times4}$, and $Q \in \mathbb{R}^{7\times7}$ are defined. Each of these covariance matrices is initialized as an identity matrix and then scaled by its respective coefficient. Following established practices [5], [20], the bottom-right block of P, denoted $P_{[-3:,-3:]}$, is scaled by a coefficient P_{coe1}, while the top-left block, $P_{[:-3,:-3]}$, is scaled by P_{coe2}. Similarly, the measurement noise covariance $R_{[:,:]}$ is scaled by R_{coe}, and motion noise block $Q_{[-3:,-3:]}$ by Q_{coe}. As summarized in Table IX, tracking performance is least sensitive to the choice of P and most sensitive to that of Q. Balancing accuracy for both pedestrians and cars, we select $P_{coe1} = 10000$, $P_{coe2} = 10$, $R_{coe} = 1$, and $Q_{coe} = 0.01$.

2) Sensitivity Analysis of Thresholds: We construct a detailed sensitivity analysis, varying the M^3 threshold (θ_S), the SGC thresholds (θ_G^{loo} and θ_G^{str}), the TF thresholds (θ_{iou} and θ_{hits}), as detailed in Tables X and XI.

M^3 is designed to be modality-agnostic, allowing a unified threshold θ_S across modalities and object categories. In contrast, SGC is modality-specific: the camera uses 2D-IoU, while LiDAR uses 3D-CD. Accordingly, θ_G^{loo} and θ_G^{str} are defined per modality, and further adapted per object category in the LiDAR modality to account for varying displacement speeds. As shown in Table X, θ_S exhibits insensitivity to both cars and pedestrians; for instance, tracking performance remains unchanged when it is set to 0.4 or 0.5. This observation suggests that the multi-modal features extracted by M^3 are highly discriminative. For the camera, both θ_G^{str} and θ_G^{loo} prove to be robust for both categories, exhibiting only minor fluctuations, specifically within 0.3%, in the HOTA metric. For the LiDAR, θ_G^{str} demonstrates robustness for both categories, with HOTA metrics fluctuating within 0.1%. In contrast, θ_G^{loo} remains insensitive to cars, and its sensitivity to pedestrians falls within an acceptable range, as evidenced by a 0.35% change in HOTA when set to 2 and 3, respectively.

Within TF, the threshold θ_{iou} is applied five times: three times during STEP-1/2/3 of cross correction, and once each for the input and output. For finer-grained sensitivity analysis, we separate it into θ_{iou}^{s1}, θ_{iou}^{s2}, θ_{iou}^{s3}, θ_{iou}^{in}, and θ_{iou}^{out}, and analyze cars and pedestrians independently. As shown in Table XI, for cars, all thresholds are insensitive for cars. For instance, varying θ_{iou}^{s1}, θ_{iou}^{s2}, θ_{iou}^{s3}, and θ_{iou}^{in} results in HOTA fluctuations within approximately 0.2%, and similarly, changes in θ_{iou}^{out} and θ_{hits} lead to variations within approximately 0.3%. For pedestrians, the HOTA metric remains stable (within 0.1%) when adjusting θ_{iou}^{s1}, θ_{iou}^{s3}, and θ_{iou}^{out}, but shows more notable sensitivity to θ_{iou}^{s2} and θ_{hits}, with variations of approximately 2% and 1%, respectively. Furthermore, we also evaluate a shared threshold setting across both categories: θ_{iou}^{s1}, θ_{iou}^{s2}, and θ_{iou}^{s3} are all set to 0.5, while θ_{iou}^{in} and θ_{iou}^{out} to 0.7 and 0.3; and θ_{hits} to 3. Under this configuration, the HOTA score for cars decreases by only 0.79% from the optimal, and for pedestrians, by just 0.48%, indicating strong robustness.

We also conduct a sensitivity analysis on the threshold θ_{age} used to discard a new object or terminate a trajectory. Table XII shows that varying θ_{age} from 1 to 2 has a significant impact on tracking performance, with the HOTA for pedestrians increasing by 1.74%. In contrast, adjusting it from 2 to 3 results in a relatively minor effect, where the HOTA for cars only improves by 0.02%. Accordingly, we set $\theta_{age} = 3$.

Detailed sensitivity analysis shows that, although tracking performance varies with threshold adjustments, the fluctuations are minor, typically within 0.3% Exceptions are noted for θ_{iou}, θ_{hits}, and θ_{age} when applied to pedestrians. Nevertheless, even in these specific cases, identifying optimal values does not entail complex or laborious efforts.

H. Deployment Feasibility Discussion

Our CrossTracker exhibits a high inference speed of 78 FPS, which is essential for facilitating low-latency and high-safety operations (well exceeding the 25 FPS requirement).

Moreover, CrossTracker boasts a memory footprint well-suited for modern automotive-grade SoCs. Its GPU memory

TABLE VIII

ABLATION STUDY ON THE IMPACT OF M^3 AND TF ON SYSTEM CPU/GPU MEMORY FOOTPRINT AND FPS ON THE KITTI VALIDATION SET FOR CARS

M^3			TF					Memory (MB)		FPS
IFM	GFM	PFM	STEP-1a	STEP-1b	STEP-2a	STEP-2	STEP-3	CPU	GPU	
								293.74	0	600
✓								590.95	2837.13	103
✓	✓							639.24	3809.63	98
✓	✓	✓						704.06	4467.70	80
✓	✓	✓	✓					716.95	4467.70	79
✓	✓	✓	✓	✓				714.87	4467.70	79
✓	✓	✓	✓	✓	✓			716.58	4454.03	79
✓	✓	✓	✓	✓	✓	✓		717.04	4456.88	78
✓	✓	✓	✓	✓	✓	✓	✓	714.89	4458.26	78

TABLE IX

ABLATION STUDY ON THE COVARIANCE MATRIX COEFFICIENTS (P_{coe1}, P_{coe2}, R_{coe}, AND Q_{coe}) OF THE KALMAN FILTER ON THE KITTI VALIDATION SET.OUR EXPERIMENTS EMPLOY CAMERA-BASED DETECTORS (RRC [49] FOR CARS AND TRACK-RCNN [50] FOR PEDESTRIANS) ALONGSIDE A LIDAR-BASED DETECTOR (POINT-GNN [37] FOR BOTH CATEGORIES).SELECTED CATEGORY-SPECIFIC SETTINGS ARE HIGHLIGHTED IN GREEN

P_{coe1}			P_{coe2}			R_{coe}			Q_{coe}			Car			Pedestrian		
100	1000	10000	0.1	1	10	0.01	0.1	1	0.0001	0.001	0.01	HOTA	DetA	AssA	HOTA	DetA	AssA
✓			✓			✓			✓			84.46	82.88	86.31	52.09	50.60	53.93
	✓		✓			✓			✓			84.45	82.87	86.31	52.09	50.60	53.93
		✓	✓			✓			✓			84.45	82.87	86.31	52.09	50.60	53.93
		✓		✓		✓			✓			84.44	82.86	86.30	51.88	50.62	53.48
		✓			✓	✓			✓			84.44	82.85	86.30	52.24	50.50	54.36
		✓			✓		✓		✓			84.46	82.90	86.30	52.55	50.25	55.26
		✓			✓			✓	✓			84.18	82.55	86.10	51.62	50.37	53.21
		✓			✓			✓		✓		84.29	82.68	86.19	52.70	50.37	**55.43**
		✓			✓			✓			✓	**84.49**	**82.90**	**86.36**	**52.84**	**50.71**	55.34

TABLE X

ABLATION STUDY OF THE THRESHOLD (θ_S) IN M^3 AND THRESHOLDS (θ_G^{loo} AND θ_G^{str}) IN SGC ON THE KITTI VALIDATION SET.HERE, θ_S IS SHARED FOR BOTH MODALITIES.CONVERSELY, θ_G^{loo} AND θ_G^{str} ARE TUNED FOR CAMERA ($\theta_{G,c}^{loo}$ AND $\theta_{G,c}^{str}$) AND LIDAR ($\theta_{G,l}^{loo}$ AND $\theta_{G,l}^{str}$), OWING TO THE DISTINCT METRICS EMPLOYED.FURTHERMORE, $\theta_{G,l}^{loo}$ AND $\theta_{G,l}^{str}$ ARE SET DIFFERENTLY FOR CARS AND PEDESTRIANS, OWING TO THEIR DIFFERING DISPLACEMENT SPEEDS.OUR EXPERIMENTS EMPLOY CAMERA-BASED DETECTORS (RRC [49] FOR CARS AND TRACK-RCNN [50] FOR PEDESTRIANS), ALONGSIDE A LIDAR-BASED DETECTOR (POINT-GNN [37] FOR BOTH CATEGORIES).SELECTED CATEGORY-SPECIFIC SETTINGS ARE HIGHLIGHTED IN GREEN

θ_S			$\theta_G^{str,c}$			$\theta_G^{loo,c}$		Car								Pedestrian							
								$\theta_G^{str,l}$			$\theta_G^{loo,l}$					$\theta_G^{str,l}$			$\theta_G^{loo,l}$				
0.3	0.4	0.5	0.5	0.6	0.7	0.9	1	1	2	3	7	8	HOTA	DetA	AssA	0.5	0.75	1	2	3	HOTA	DetA	AssA
✓			✓			✓		✓			✓		83.76	82.20	85.59	✓			✓		51.68	49.95	53.77
	✓		✓			✓		✓			✓		84.02	82.23	86.10	✓			✓		52.02	50.19	54.22
		✓	✓			✓		✓			✓		84.02	82.22	86.11	✓			✓		52.02	50.19	54.22
		✓		✓		✓		✓			✓		84.30	82.70	86.19	✓			✓		51.92	50.01	54.21
		✓			✓	✓		✓			✓		84.34	82.75	86.22	✓			✓		52.13	50.20	54.45
		✓			✓		✓	✓			✓		84.35	82.74	86.24	✓			✓		52.42	50.52	54.68
		✓			✓		✓		✓		✓		84.42	82.85	86.27		✓		✓		52.52	50.61	54.80
		✓			✓		✓			✓	✓		84.45	82.89	86.28			✓	✓		52.49	50.62	54.73
		✓			✓		✓			✓		✓	**84.49**	**82.90**	**86.36**			✓		✓	**52.84**	**50.71**	**55.34**

footprint is approximately 4.46 GB, which is only a small portion of the up to 64 GB LPDDR5 found on advanced platforms (e.g., NVIDIA DRIVE AGX Orin). Additionally, its 714 MB CPU memory footprint can be readily accommodated by typical multi-core processors in these systems.

Furthermore, CrossTracker requires only 8.45 GFLOPs and a compact 16.43 M parameters. This lightweight design is critical for in-vehicle deployment, ensuring minimal latency and low power consumption under strict thermal and power budgets. Such efficiency is especially compelling when contrasted with the vast processing capabilities of advanced platforms (e.g., NVIDIA DRIVE AGX Orin), which provides peak FP32 performance of up to 5.325 TFLOPS.

In summary, the combined high inference speed, low memory footprint, and minimal computational requirements make our method an excellent candidate for practical, safety-critical autonomous driving applications on state-of-the-art hardware.

V. LIMILATION

Our CrossTracker, while effective, has some limitations. Firstly, tracking objects missed by both camera and LiDAR modalities at the field of view boundary is difficult. As illustrated in Fig. 9, both modalities failed to detect a car located at the view boundary, leading to tracking failure. Secondly, the TF module may introduce noise during the cross correction of camera and LiDAR trajectories, which can adversely affect the output 3D trajectories. As shown in Table V, the handling of case (STEP-2b) slightly decreases the HOA metric for pedestrians. Thirdly, our method is limited to single-camera (monocular) input and does not yet support multi-camera setups. Fourthly, since our method is based on the tracking-by-detection paradigm, its performance is inherently tied to that of the underlying object detector. Consequently, if the detector fails to identify an object class not present in its training data, our method will also fail. Fifthly, even under extremely adverse

TABLE XI

ABLATION STUDY OF THRESHOLDS (θ_{iou} AND θ_{hits}) IN TF ON THE KITTI VALIDATION SET.SPECIFICALLY, WE VARY θ_{iou} ACROSS FIVE VALUES: θ^{s1}_{iou}, θ^{s2}_{iou}, θ^{s3}_{iou} (FOR STEP-1/2/3 OF CORRECTION), AND θ^{in}_{iou} AND θ^{out}_{iou} (FOR MATCHING THE INPUT AND OUTPUT CAMERA-LIDAR TRAJECTORIES).EXPERIMENTS EMPLOY RRC [49] FOR CARS, TRACK-RCNN [50] FOR PEDESTRIANS, AND POINT-GNN [37] FOR BOTH.SELECTED CATEGORY-SPECIFIC SETTINGS ARE HIGHLIGHTED IN GREEN, WHILE OPTIMAL SHARED SETTINGS ARE SHOWN IN GRAY

Car

θ^{s1}_{iou}			θ^{s2}_{iou}			θ^{s3}_{iou}			θ^{in}_{iou}			θ^{out}_{iou}			θ_{hits}			HOTA	DetA	AssA
0.3	0.4	0.5	0	0.02	0.05	0.3	0.4	0.5	0.3	0.4	0.5	0.2	0.3	0.4	1	2	3			
✓			✓			✓			✓			✓			✓			83.83	81.92	86.03
	✓		✓			✓			✓			✓			✓			83.84	81.94	86.02
		✓	✓			✓			✓			✓			✓			83.85	81.95	86.04
		✓		✓		✓			✓			✓			✓			83.93	82.07	86.08
		✓			✓	✓			✓			✓			✓			83.95	82.11	86.08
		✓			✓		✓		✓			✓			✓			84.01	82.25	86.04
		✓			✓			✓	✓			✓			✓			84.15	82.58	86.00
		✓			✓			✓		✓		✓			✓			84.22	82.56	86.15
		✓			✓			✓			✓	✓			✓			84.24	82.58	86.17
		✓			✓			✓			✓		✓		✓			84.41	82.81	86.29
		✓			✓			✓			✓			✓	✓			**84.52**	**82.96**	86.35
		✓			✓			✓			✓			✓		✓		84.49	82.90	**86.36**
		✓			✓			✓			✓			✓			✓	84.24	82.70	86.06
	0.5			0.5			0.5			0.7			0.3			3		83.73	82.48	85.27

Pedestrian

θ^{s1}_{iou}			θ^{s2}_{iou}			θ^{s3}_{iou}			θ^{in}_{iou}			θ^{out}_{iou}			θ_{hits}			HOTA	DetA	AssA
0.2	0.3	0.4	0.3	0.4	0.5	0.6	0.7	0.8	0.5	0.6	0.7	0	0.05	0.1	1	3	5			
✓			✓			✓			✓			✓			✓			49.94	45.79	54.75
	✓		✓			✓			✓			✓			✓			49.91	45.92	54.53
		✓	✓			✓			✓			✓			✓			49.90	45.91	54.52
		✓		✓		✓			✓			✓			✓			51.85	48.14	**56.13**
		✓			✓	✓			✓			✓			✓			51.56	49.18	54.35
		✓			✓		✓		✓			✓			✓			51.47	49.32	54.02
		✓			✓			✓	✓			✓			✓			51.62	49.64	53.99
		✓			✓			✓		✓		✓			✓			51.83	49.45	54.64
		✓			✓			✓			✓	✓			✓			51.88	49.15	55.04
		✓			✓			✓			✓		✓		✓			51.92	49.25	55.03
		✓			✓			✓			✓			✓	✓			51.97	49.35	55.02
		✓			✓			✓			✓			✓		✓		52.65	50.43	55.25
		✓			✓			✓			✓			✓			✓	**52.84**	**50.71**	55.34
	0.5			0.5			0.5			0.7			0.3			3		52.36	50.24	54.86

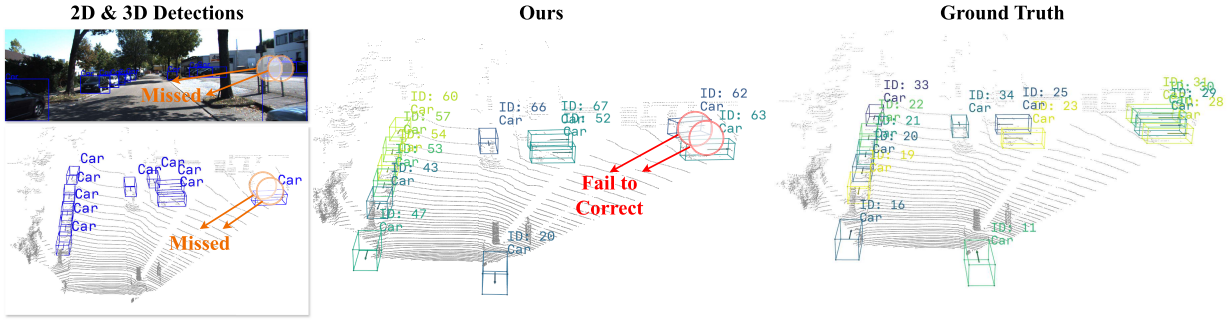

Fig. 9. **The case of failure.** This is a failure case where two modalities simultaneously miss detection at the boundary of the field of view, and the failure has not been successfully corrected by our CrossTracker.

TABLE XII

ABLATION STUDY OF THE THRESHOLD (θ_{age}) FOR DISCARDING NEW OBJECTS OR TERMINATING TRAJECTORIES ON THE KITTI VALIDATION SET.EXPERIMENTS EMPLOY RRC [49] AND POINT-GNN [37] FOR CARS AND PEDESTRIANS

θ_{age}			Car			Pedestrian		
1	2	3	HOTA	DetA	AssA	HOTA	DetA	AssA
✓			83.55	81.98	85.43	51.86	**52.56**	51.50
	✓		84.47	82.89	86.34	53.60	52.28	55.25
		✓	**84.49**	**82.90**	**86.36**	**54.29**	52.14	**56.81**

conditions (e.g., motion blur in both LiDAR and camera, and cross-sensor interference in LiDAR, as shown in Table II), while our method exhibits stronger robustness compared to other multi-modal methods, its performance still significantly lags behind that observed in clean scenarios. In light of these limitations, we are working toward further enhancing the robustness and generality of CrossTracker, such as by exploring open-set category-aware perception modules and visual enhancement modules tailored for adverse conditions.

VI. CONCLUSION

We present CrossTracker, a novel two-stage multi-modal 3D MOT method. Following a coarse-to-fine paradigm,

CrossTracker effectively addresses tracking failures stemming from inaccurate detections in complex scenarios. A key component is the M^3 module, which integrates multi-modal feature modeling (images, point clouds, and even image-derived planar geometry) and consistency probability estimation into a unified end-to-end network, eliminating manual calculations and advancing the state-of-the-art. Building upon M^3, the C-TG module generates coarse camera and LiDAR trajectories independently, which the TF module subsequently refines through cross correction. This trajectory fusion process yields highly robust 3D trajectories, surpassing existing methods.

Our work highlights the effectiveness of a two-stage, coarse-to-fine paradigm for 3D MOT, opening up new avenues for robust 3D MOT solutions. Moreover, our proposed CrossTracker framework not only sets a new benchmark in this field but also serves as a source of inspiration for future research. We believe this work will spark a wave of follow-up studies and drive the advancement of 3D MOT research.

REFERENCES

[1] J. Liu, W. Sun, C. Liu, X. Zhang, S. Fan, and W. Wu, "HFF6D: Hierarchical feature fusion network for robust 6D object pose tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7719–7731, Nov. 2022.

[2] G. L. Foresti, "Object recognition and tracking for remote video surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 1045–1062, Jul. 1999.

[3] C. Kim and J.-N. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 2, pp. 122–129, Feb. 2002.

[4] X. Hu et al., "Transformer tracking via frequency fusion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 2, pp. 1020–1031, Feb. 2024.

[5] X. Weng, J. Wang, D. Held, and K. Kitani, "AB3DMOT: A baseline for 3D multi-object tracking and new evaluation metrics," 2020, *arXiv:2008.08063*.

[6] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 474–490.

[7] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11784–11793.

[8] S. Wang, Y. Liu, T. Wang, Y. Li, and X. Zhang, "Exploring object-centric temporal modeling for efficient multi-view 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 3598–3608.

[9] P. Chu and H. Ling, "FAMNet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6171–6180.

[10] G. Wang, R. Gu, Z. Liu, W. Hu, M. Song, and J. Hwang, "Track without appearance: Learn box and tracklet embedding with local and global motion patterns for vehicle tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9856–9866.

[11] M. Chaabane, P. Zhang, J. Ross Beveridge, and S. O'Hara, "DEFT: Detection embeddings for tracking," 2021, *arXiv:2102.02267*.

[12] A. Kim, G. Brasó, A. Ošep, and L. Leal-Taixé, "PolarMOT: How far can geometric relations take us in 3D multi-object tracking?," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 41–58.

[13] A. Kim, A. Ošep, and L. Leal-Taixé, "EagerMOT: 3D multi-object tracking via sensor fusion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 11315–11321.

[14] X. Wang, C. Fu, Z. Li, Y. Lai, and J. He, "DeepFusionMOT: A 3D multi-object tracking framework based on camera-LiDAR fusion with deep association," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8260–8267, Jul. 2022.

[15] X. Wang, C. Fu, J. He, S. Wang, and J. Wang, "StrongFusionMOT: A multi-object tracking method based on LiDAR-camera fusion," *IEEE Sensors J.*, vol. 23, no. 11, pp. 11241–11252, Jun. 2023.

[16] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust multi-modality multi-object tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pp. 2365–2374, Oct. 2019.

[17] K. Huang and Q. Hao, "Joint multi-object detection and tracking with camera-LiDAR fusion for autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 6983–6989.

[18] A. Shenoi et al., "JRMOT: A real-time 3D multi-object tracker and a new large-scale dataset," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10335–10342.

[19] P. Sun et al., "TransTrack: Multiple object tracking with transformer," 2020, *arXiv:2012.15460*.

[20] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3464–3468.

[21] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3645–3649.

[22] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "FANTrack: 3D multi-object tracking with feature association network," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1426–1433.

[23] M. Hu, X. Zhu, H. Wang, S. Cao, C. Liu, and Q. Song, "STDFormer: Spatial–temporal motion transformer for multiple object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 11, pp. 6571–6594, Nov. 2023.

[24] H. Wu, H. Sun, K. Ji, and G. Kuang, "Temporal-spatial feature interaction network for multi-drone multi-object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 35, no. 2, pp. 1165–1179, Feb. 2025.

[25] X. Zhang, H. Yu, Y. Qin, X. Zhou, and S. Chan, "Video-based multi-camera vehicle tracking via appearance-parsing spatio-temporal trajectory matching network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 10, pp. 10077–10091, Oct. 2024.

[26] K. Shim, J. Byun, K. Ko, J. Hwang, and C. Kim, "Enhancing robustness of multi-object trackers with temporal feature mix," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 10, pp. 9822–9835, Oct. 2024.

[27] J. Lin, G. Liang, and R. Zhang, "LTTrack: Rethinking the tracking framework for long-term multi-object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 10, pp. 9866–9881, Oct. 2024.

[28] J. Cheng, S.-Y. Kuan, H. Latapie, G. Liu, and J. Hwang, "CenterRadarNet: Joint 3D object detection and tracking framework using 4D FMCW radar," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2023, pp. 998–1004.

[29] C. Tsai, G.-Y. Shen, and H. Nisar, "Swin-JDE: Joint detection and embedding multi-object tracking in crowded scenes based on Swin-transformer," *Proc. Eng. Appl. Artif. Intell.*, vol. 119, Mar. 2023, Art. no. 105770.

[30] K. Deng, C. Zhang, Z. Chen, W. Hu, B. Li, and F. Lu, "Jointing recurrent across-channel and spatial attention for multi-object tracking with block-erasing data augmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 8, pp. 4054–4069, Aug. 2023.

[31] W. Lv, N. Zhang, J. Zhang, and D. Zeng, "One-shot multiple object tracking with robust ID preservation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 6, pp. 4473–4488, Jun. 2024.

[32] J. Kong, E. Mo, M. Jiang, and T. Liu, "MOTFR: Multiple object tracking based on feature recoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7746–7757, Nov. 2022.

[33] S. You, H. Yao, and C. Xu, "Multi-object tracking with spatial–temporal topology-based detector," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 5, pp. 3023–3035, May 2022.

[34] Y. Jin, F. Gao, J. Yu, J. Wang, and F. Shuang, "Multi-object tracking: Decoupling features to solve the contradictory dilemma of feature requirements," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 9, pp. 5117–5132, Sep. 2023.

[35] H. Wu et al., "CasA: A cascade attention network for 3-D object detection from LiDAR point clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5704511.

[36] H. Wu, C. Wen, S. Shi, X. Li, and C. Wang, "Virtual sparse convolution for multimodal 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 21653–21662.

[37] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1708–1716.

[38] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[40] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "FairMOT: On the fairness of detection and re-identification in multiple object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 11, pp. 3069–3087, Nov. 2021.

[41] X. Li, D. Liu, Y. Wu, X. Wu, L. Zhao, and J. Gao, "Fast-poly: A fast polyhedral algorithm for 3D multi-object tracking," *IEEE Robot. Autom. Lett.*, vol. 9, no. 11, pp. 10519–10526, Nov. 2024.
[42] S. Tian, M. Duan, J. Deng, H. Luo, and Y. Hu, "MF-Net: A multimodal fusion model for fast multi-object tracking," *IEEE Trans. Veh. Technol.*, vol. 73, no. 8, pp. 10948–10962, Aug. 2024.
[43] H. Wu, W. Han, C. Wen, X. Li, and C. Wang, "3D multi-object tracking in point clouds based on prediction confidence-guided data association," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5668–5677, Jun. 2022.
[44] N. Marinello, M. Proesmans, and L. Van Gool, "TripletTrack: 3D object tracking using triplet embeddings and LSTM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 4499–4509.
[45] X. Wang et al., "A multi-modal fusion-based 3D multi-object tracking framework with joint detection," *IEEE Robot. Autom. Lett.*, vol. 10, no. 1, pp. 532–539, Jan. 2025.
[46] C. Peng et al., "PNAS-MOT: Multi-modal object tracking with Pareto neural architecture search," *IEEE Robot. Autom. Lett.*, vol. 9, no. 5, pp. 4377–4384, May 2024.
[47] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.
[48] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[49] J. Ren et al., "Accurate single stage detector using recurrent rolling convolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 752–760.
[50] P. Voigtlaender et al., "MOTS: Multi-object tracking and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Feb. 2019, pp. 7942–7951.
[51] P. Tokmakov, J. Li, W. Burgard, and A. Gaidon, "Learning to track with object permanence," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10840–10849.
[52] S. Sharma, J. Ansari, J. K. Murthy, and K. M. Krishna, "Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 3508–3515.
[53] J. Luiten, T. Fischer, and B. Leibe, "Track to reconstruct and reconstruct to track," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1803–1810, Apr. 2020.
[54] K. Zhang, Y. Liu, F. Mei, J. Jin, and Y. Wang, "Boost correlation features with 3D-MiIoU-Based camera-LiDAR fusion for MODT in autonomous driving," *Remote Sens.*, vol. 15, no. 4, p. 874, Feb. 2023.
[55] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 11037–11045.
[56] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *EURASIP J. Image Video Process.*, vol. 2008, pp. 1–10, May 2008.
[57] J. Luiten et al., "HOTA: A higher order metric for evaluating multi-object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 548–578, Feb. 2021.

Lipeng Gu received the M.Sc. degree from Donghua University in 2021. He is currently pursuing the Ph.D. degree in computer science and technology with Nanjing University of Aeronautics and Astronautics. His research interests include deep learning, computer vision, and computer graphics.

Xuefeng Yan received the Ph.D. degree from Beijing Institute of Technology in 2005. He was a Visiting Scholar at Georgia State University in 2008 and 2012. He is currently a Professor with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), China. His research interests include intelligent computing, MBSE/complex system modeling, simulation, and evaluation.

Weiming Wang received the Ph.D. degree from The Chinese University of Hong Kong in 2014. He was an Assistant Researcher at Shenzhen Institutes of Advanced Technology from 2014 to 2015. After that, he worked in high-tech companies for a few years. He is currently an Assistant Professor with Hong Kong Metropolitan University. His research interests include image processing, point cloud processing, and deep learning. He is an Associate Editor of *The Visual Computer*.

Honghua Chen received the Ph.D. degree from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2022. He is currently a Research Assistant Professor with the Division of Artificial Intelligence, School of Data Science, Lingnan University, Hong Kong. Previously, he spent two and a half years as a Research Fellow at the College of Computing and Data Science, Nanyang Technological University, Singapore, affiliated with the S-Lab. His research interests include 3D vision, 3D measurement, computer graphics, and geometric deep learning.

Dingkun Zhu received the Ph.D. degree in computer science and engineering from Hong Kong Metropolitan University. He is currently an Assistant Professor with the School of Computer Science, Jiangsu University of Technology, and a Post-Doctoral Fellow with The Hong Kong Polytechnic University. His research interests include computer graphics, 3D vision, image processing, and deep learning.

Liangliang Nan received the Ph.D. degree in mechatronics engineering from the Graduate University of the Chinese Academy of Sciences (CAS) in 2009. Then, he was an Assistant Professor and later an Associate Professor at Shenzhen Institute of Advanced Technology, CAS. Since 2013, he has been a Research Scientist at the Visual Computing Center, KAUST. In 2018, he joined Delft University of Technology, where he is currently an Associate Professor and the Director of the AI Laboratory on 3D Urban Understanding (3DUU). His research interests include the intersection of computer vision, computer graphics, and 3D geoinformation, with a focus on understanding and modeling real-world scenes.

Mingqiang Wei (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong (CUHK) in 2014. He is currently a Professor with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA). He has published more than 140 research papers in IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, SIGGRAPH, IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, IEEE TRANSACTIONS ON IMAGE PROCESSING, CVPR, and ICCV. His research interests include 3D vision and computer graphics. He was a recipient of the CUHK Young Scholar Thesis Awards in 2014. He is an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING, ACM TOMM, and *The Visual Computer* journal; and a Guest Editor of IEEE TRANSACTIONS ON MULTIMEDIA.