MV-3DT: Multi-View 3D Object Tracking via Cross-Camera 3D Fusion

Anonymous Author(s)
Affiliation
Address
email

Abstract: Autonomous vehicles need to be aware of the 3D locations and trajectories of the other traffic participants at any given time to safely navigate in the real world. Yet, camera-based 3D object tracking methods prioritize optimizing the single-camera setup and resort to post-hoc fusion in a multi-camera setup. In this paper, we propose a cross-camera 3D fusion method, called *MV-3DT*, that associates objects both temporally and across cameras, and improves the overall tracking consistency. In particular, our method fuses 3D detection results from multiple cameras before association and reduces identity switches significantly. Our experiments on large-scale driving datasets show that fusion before association leads to a large margin of improvement on multiple base models. Accordingly, we set a new state of the art among all camera-based methods on the competitive NuScenes 3D tracking benchmark, outperforming previously published methods by 8.7% points in average multi-object tracking accuracy (AMOTA).

Keywords: Autonomous driving, 3D multi-object tracking, Multi-camera fusion

1 Introduction

Locating and tracking dynamic objects in 3D is vital to autonomous driving safety systems. Accordingly, advanced techniques with complex pipelines are often employed, which generally use multiple 3D sensor modalities ranging from stereo cameras to LiDAR and radar [1, 2, 3, 4, 5]. On the contrary, a high-quality monocular multi-camera system provides an excellent low-cost alternative to expensive 3D active sensors to capture the complete surrounding of a vehicle with no blind spots. Furthermore, many modern consumer vehicles are already equipped with such a multi-camera system; hence, no extra installation budget is required to use such a system for improving 3D multi-object tracking (MOT). Yet, for the mindful use of a multi-camera setup, it is important to have a method that can effectively utilize both temporal association and cross-camera association of moving objects in 3D. Such an approach can help build the object's long-term trajectory as cameras locate it from different viewpoints.

Existing camera-based 3D object tracking methods are designed for single-camera setups [6, 7, 8]. The only credible difference between such camera-based approaches which follow the classical tracking-by-detection [9] lies in the association of object tracks [10, 11, 12, 13]. For instance, recent work by Zhou et al. [7] relied on the 2D motion for object track association, while Chaabane et al. [8] utilized object appearance to associate detection over time. Hu et al. [6] combines the appearance and 3D motion information for object track association. Even though these recent methods perform well on the temporal association of object tracks, they fail to associate object tracks in 3D across multiple cameras installed on the vehicle.

To address this limitation, we put forward an effective solution to the multi-camera 3D tracking problem. We merge 3D detections and their appearance information from multiple cameras into a single 3D tracking component that performs both temporal and cross-camera association. Given the cross-camera association, we leverage a cross-camera motion model to improve the quality of single-frame 3D detections even when an object has just appeared in a particular camera's field of

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

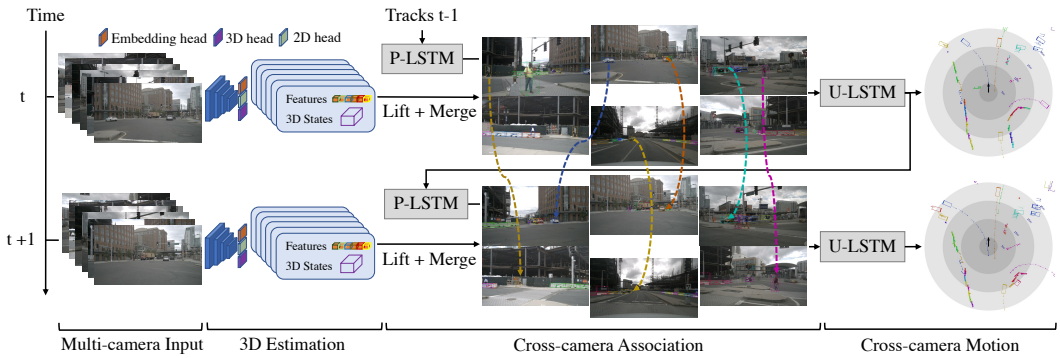

Figure 1: **Method overview**. We first utilize a 3D estimation network to generate 3D detections and corresponding appearance features for each multi-camera input. Then, we lift all the detections into world-space and merge them using 3D non-maximum suppression. Next, we perform cross-camera association and motion modeling and obtain the final tracking results that are consistent across all cameras.

view (FOV), leading to *highly accurate tracking* with much *smoother trajectory* of objects across cameras in 3D. We summarize our key contributions as follows:

- We propose a multi-camera 3D fusion method, called MV-3DT, that associates objects both temporally and across cameras resulting in a 17% relative improvement over the single-camera baseline.
- We show that with MV-3DT, we can benefit from cross-camera trajectories leading to significant improvements in motion model quality, which in turn benefits data association.
- The idea of leveraging the cross-camera information mindfully not only provides smoother object trajectories but also offers a new state-of-the-art for camera-based 3D tracking on the competitive NuScenes [1] benchmark, outperforming existing camera-based methods by a significant margin.

2 Related Work

Monocular 3D Detection. Driven by the success of convolutional neural networks (CNNs) in object detection [14, 15, 16, 17, 18], many recent works on single image 3D object detection have been using CNNs to regress the 3D parameters of objects from an image [19, 20, 21, 22, 23, 24]. In contrast, another line of work [25, 26, 27] lifts the image to a 3D point cloud using single image depth prediction and detects objects in the 3D point cloud, similar to LiDAR-based 3D object detectors [28, 29, 30]. Lately, transformer networks have been explored for end-to-end object detection [31]. Inspired by this, Wang et al. [32] proposes to use a transformer network for 3D object detection from multiple cameras. Our work extends beyond single image 3D detection by performing temporal refinement of the 3D bounding box parameters, improving upon the initial detection result.

2D Multi-Object Tracking. Most recent works in MOT follow the tracking-by-detection paradigm [9]. First, an object detector is used to recognize the objects of interest in each frame to subsequently associate the detections using different cues, such as 2D motion [33, 10, 34, 35, 36, 7], visual appearance [37, 38, 11, 13, 39, 8, 40] or 3D motion [41, 42, 43, 44, 45]. Recently, researchers have explored transformer networks for MOT [46, 47, 48]. These works take inspiration from [31] and use the query-based detection process for temporal query propagation to track objects across time with transformer network. In addition, there is a broad range of work on multi-camera MOT [49, 50, 51, 52, 53]. Those works focus on scenarios with highly overlapping, stationary surveillance cameras in *e.g.* smart city applications [54]. Their aim is to overcome occlusion problems in monocular MOT systems. Our work on the other hand focuses on multi-camera tracking in the context of autonomous driving, where the cameras usually have little overlap and are mounted on a moving platform.

Monocular 3D Multi-Object Tracking. In the realm of 3D MOT, there are many methods that rely on LiDAR sensors since they provide both a very accurate 3D localization and 360-degree coverage of the environment [41, 55, 56, 57, 58, 59]. On the contrary, camera-based 3D MOT has

been approached by only a handful of works [6, 7, 8]. Those methods are conceptually similar to 2D MOT systems, employing tracking-by-detection and using motion and/or appearance cues for the association. While sharing the properties of 2D MOT systems, the camera-based approaches cannot benefit from the 360-degree coverage of the LiDAR-based 3D MOT methods. Nevertheless, none of the methods above consider cross-camera association explicitly. Concurrent to our work, Zhang et al. [60] propose a method that combines [32] with the ideas in [46, 47, 48] into a transformer-based multi-camera 3D tracking method. However, our work differs significantly from [60] in that we do not rely on a transformer network for joint detection and association, but fuse the detections from multiple cameras into a single tracking component. Moreover, our method outperforms MUTR3D significantly in experimental evaluation, even with a weaker 3D detector.

3 Method

We first define the problem setup and the foundation of the 3D tracking pipeline inspired by [6]. Afterward, we detail our multi-camera 3D tracking method. The whole pipeline is illustrated in Figure 1.

Problem setup. Assume we are given input images I_t^m from a set of cameras $m \in M$ and the corresponding camera locations in world-frame ξ_t in a sequence of time steps $t \in T$. We aim to finding the set of tracks $\mathcal{T} = \{\tau^1, \ldots, \tau^N\}$ corresponding to the objects in the scene. Each track $i \in [1, N]$ consists of a set of states $\langle \tau_t^i, \tau_{t+1}^i, ... \rangle$ depending on the time steps that object is visible in. Each state $\tau_t^i = \langle \mathbf{f}_t^{\tau^i}, \mathbf{b}_t^{\tau^i}, c_t^{\tau^i} \rangle$ consists of an appearance embedding $\mathbf{f}_t^{\tau^i} \in \mathbb{R}^{256}$, a 3D bounding box $\mathbf{b}_t^{\tau^i} \in \mathbb{R}^7$ and a confidence value $c_t^{\tau^i} \in [0, 1]$. The 3D bounding box is defined as $\mathbf{b}_t^{\tau^i} = [x, y, z, \theta, l, w, h]$ where (x, y, z, θ) define the 3D center of the object and its orientation around the vertical axis in 3D world-frame, and (l, w, h) define the object dimensions.

3D estimation network. We use a 3D estimation network that extracts object detecions $\mathcal{D}_t$ from a single input image I_t^m, consisting of corresponding 3D bounding boxes $\mathbf{b}_t$, confidence values c_t and appearance embeddings $\mathbf{f}_t$. The detections $d \in \mathcal{D}_t$ share the same state space with the tracks $\mathcal{T}$, *i.e.* $d_t = \langle \mathbf{f}_t^d, \mathbf{b}_t^d, c_t^d \rangle$. Note that $\mathbf{b}_t^d$ is transformed into world coordinate system with ξ_t.

Data association. We associate the detections $\mathcal{D}_t$ with the active tracks $\mathcal{T}_t$ at time t by greedy assignment given an affinity matrix. The affinity matrix $\mathbf{A}(\mathcal{T}_t, \mathcal{D}_t) \in \mathbb{R}^{||\mathcal{T}_t|| \times ||\mathcal{D}_t||}$ consists of appearance, location, and velocity correlation. We compute the appearance correlation based on the appearance embedding similarity in [40] as

$$\mathbf{a}_{deep}(\tau_t, d_t) = \frac{1}{2}\left[\frac{\exp(\mathbf{f}_t^d \cdot \mathbf{f}_t^\tau)}{\sum_{d \in \mathcal{D}} \exp(\mathbf{f}_t^d \cdot \mathbf{f}_t^\tau)} + \frac{\exp(\mathbf{f}_t^d \cdot \mathbf{f}_t^\tau)}{\sum_{\tau \in \mathcal{T}} \exp(\mathbf{f}_t^d \cdot \mathbf{f}_t^\tau)}\right]. \tag{1}$$

The location correlation is based on the absolute difference between the predicted and observed 3D bounding box parameters as

$$\mathbf{a}_{loc}(\tau_t, d_t) = \exp(\frac{-|\mathbf{b}_t^\tau - \mathbf{b}_t^d|}{r}), \tag{2}$$

where r is a constant scalar. We take into account the absolute difference between the predicted motion $V_{\tau_t} = \overrightarrow{P_{\tau_{t-1}} P_{\tau_t}}$ and observed pseudo motion $V_{d_t} = \overrightarrow{P_{\tau_{t-1}} P_{d_t}}$ as well as the centroid distance $|P_{\tau_t} - P_{d_t}|$ to compute the motion correlation as follows:

$$\begin{aligned} &\mathbf{a}_{motion}(\tau_t, d_t) = w_{cos}\, \mathbf{a}_{centroid} + (1 - w_{cos})\, \mathbf{a}_{pseudo} \\ &\text{where, } \mathbf{a}_{centroid} = \exp(\frac{-|P_{\tau_t} - P_{d_t}|}{r}),\ \mathbf{a}_{pseudo} = \exp(\frac{-|V_{\tau_t} - V_{d_t}|}{r}),\ w_{cos} = \angle(V_{\tau_t}, V_{d_t}) \end{aligned} \tag{3}$$

and $w_{cos} \in [0, 1]$ is the normalized cosine similarity of two motion vectors. Note that we disentangle objects from ego-motion by transforming each object center P into a world-frame given ξ. The essence of w_{cos} is to favor centroid distance if two motion vectors are in the same hemisphere and pseudo motion otherwise. This ensures that we do not match a pair with conflicting motions. Finally, the affinity matrix $\mathbf{A}(\mathcal{T}_t, \mathcal{D}_t)$ is computed as a weighted sum of the individual affinities:

$$\mathbf{A}(\mathcal{T}_t, \mathcal{D}_t) = w_{deep}\mathbf{A}_{deep}(\mathcal{T}_t, \mathcal{D}_t) + (1 - w_{deep})\mathbf{A}_{motion}(\mathcal{T}_t, \mathcal{D}_t) \cdot \mathbf{A}_{loc}(\mathcal{T}_t, \mathcal{D}_t), \tag{4}$$

where w_{deep} is a constant that weighs appearance and a hybrid of motion and location correlation.

Motion model. We employ two LSTM networks Φ_{pred} and Φ_{update} to handle the extrapolation of trajectories given track histories and the update of trajectories given new observations, respectively. Before data association, we use Φ_{pred} to project the previous object state $\mathbf{b}_{t-1}$ one time step into the future to compute the location and motion correlation. We input the previous $n-1=5$ object velocities $\mathbf{v}^{\tau}_{t-n:t-1}$ and the previous hidden state of the LSTM $\mathbf{h}^{pred}_{t-1}$ into the network, where the object velocity at time t is defined as $\mathbf{v}^{\tau}_{t} = \mathbf{b}^{\tau}_{t} - \mathbf{b}^{\tau}_{t-1}$. The network outputs the predicted velocity $\hat{\mathbf{v}}^{\tau}_{t}, \mathbf{h}^{pred}_{t} = \Phi_{pred}(\mathbf{v}^{\tau}_{t-n:t-1}, \mathbf{h}^{pred}_{t-1})$. After data association, we refine the observed object state $\mathbf{b}^{\tau}_{t}$ using the velocity estimated by Φ_{update}. It takes as input the predicted velocity $\hat{\mathbf{v}}^{\tau}_{t}$, observed velocity $\mathbf{v'}^{\tau}_{t}$, the confidence of the associated detection c^{τ}_{t} and the previous hidden state $\mathbf{h}^{pred}_{t-1}$. The network estimates the refined velocity $\mathbf{v}^{\tau}_{t}, \mathbf{h}^{update}_{t} = \Phi_{update}([\hat{\mathbf{v}}^{\tau}_{t}, \mathbf{v'}^{\tau}_{t}, c^{\tau}_{t}], \mathbf{h}^{update}_{t-1})$, which in turn refines the object state $\mathbf{b}^{\tau}_{t} = \mathbf{b}^{\tau}_{t-1} + \mathbf{v}^{\tau}_{t-1}$. Note that for both networks, we first use a linear projection to encode each input into a 64-dimensional vector which we omit for ease of notation.

Track management. In order to handle newly appearing, disappearing, and reappearing objects during inference, we need to perform online track management. We keep a track memory where each track $\tau \in \mathcal{T}$ is assigned one of three possible states $\{\texttt{active}, \texttt{inactive}, \texttt{dead}\}$. At each time step t, all tracks that are currently marked as `active` or `inactive` are used to calculate their affinity with the current detections $\mathcal{D}_{\sqcup}$. If a detection $d_t \in \mathcal{D}_{\sqcup}$ has high affinity with a track $\tau \in \mathcal{T}$, d_t, it will be assigned to τ. If a track τ has no matching detection, it will transition into state `inactive`. If τ did not have a matching detection for L time steps, τ transitions into state `dead` and will no longer be considered for matching. Note that we propagate the position of inactive tracks forward in time with Φ_{pred}, so that they can be properly matched even if they disappeared for a longer time span. Each detection that has no matching track will spawn a new track in `active` state.

3.1 Cross-camera association

In [6], a separate tracker for each camera $m \in M$ is employed, and the results are combined in an online fashion with 3D non-maximum suppression (NMS). In particular, the data association of each camera happens independently, as well as the track management. Since each camera has only a limited FOV, the association becomes less stable, since there are a lot of disappearing and reappearing tracks to handle. Moreover, if an object moves across cameras, the identity of the object will not be preserved across cameras. We explore how a multi-camera tracker can overcome those limitations with a few simple modifications to the tracking pipeline.

First, we investigate how performance is affected if we use 3D NMS to naively combine the existing tracks from the single-camera trackers. We call this approach detect $\rightarrow$ track $\rightarrow$ merge. Next, we propose to change this paradigm to detect $\rightarrow$ merge $\rightarrow$ track, *i.e.* instead of aggregating the tracking result of camera-dependent trackers, we *first* aggregate the *detection* results from all cameras $m \in M$ and subsequently apply global data association, motion modeling, and track management. The advantages of the detect $\rightarrow$ merge $\rightarrow$ track paradigm are threefold. ***(i)*** In the regions where the cameras have an overlapping FOV, the multi-camera tracker has access to more detection candidates, meaning that an object that is highly truncated in one camera can be recognized in a neighboring camera. ***(ii)*** Cross-camera data association results in long trajectories that span across multiple cameras, which allows for a richer track history and thus better motion modeling. ***(iii)*** Our data association scheme leverages appearance, location and velocity information to follow tracks more effectively across cameras compared to 3D NMS based merging of single-camera tracking results.

3.2 Cross-camera motion modeling

A crucial advantage of our multi-camera fusion is that we can exploit long, cross-camera trajectories for motion modeling. This leads to an improved multi-frame refinement via Φ_{update} and more accurate location prediction via Φ_{pred}, which in turn benefits the data association since we can calculate better estimates for the affinities $\mathbf{a}_{loc}(\tau_t, d_t)$ and $\mathbf{a}_{motion}(\tau_t, d_t)$.

In addition to leveraging cross-camera trajectories during inference, we propose to integrate them into the training process of the motion model. The intuition is that due to object truncation, the 3D localization at image borders is usually brittle. Thus, if we train the motion model on these trajectories, it can handle those situations better. In contrast to [6], who create the trajectory training

data per camera and train on the union of the single-camera trajectories, we create a cross-camera trajectory dataset that matches ground truth trajectories spanning multiple cameras to the aggregated set of detections from all cameras. To match detections to a ground truth trajectory, we use birds-eye-view (BEV) distance instead of the Intersection-over-Union (IoU) of 2D bounding boxes in [6].

To train the networks Φ_{pred} and Φ_{update}, we randomly sample a ground truth trajectory and extract the matched ground truth and detection pairs in a randomly chosen temporal window Δt of fixed size, possibly spanning across multiple camera views. In particular, we use the set of detections to simulate the process, *i.e.* we obtain the predicted velocities $\hat{\mathbf{v}}_{\Delta t}$ from Φ_{pred} given the track history and the refined velocities $\mathbf{v}_{\Delta t}$ from Φ_{update} given the predicted velocity, the observed velocity, and the detection confidence. Given the corresponding ground truth velocities $\tilde{\mathbf{v}}_{\Delta t}$ inferred from the associated ground truth 3D bounding boxes, we calculate the trajectory loss:

$$\mathcal{L}_{traj}(\mathbf{v}_{\Delta t}, \hat{\mathbf{v}}_{\Delta t}, \tilde{\mathbf{v}}_{\Delta t}) = \frac{1}{||\Delta t||} \sum_{t \in \Delta t} \delta(\hat{\mathbf{v}}_t, \tilde{\mathbf{v}}_t) + \delta(\mathbf{v}_t, \tilde{\mathbf{v}}_t), \tag{5}$$

where $\delta(\cdot, \cdot)$ is the Huber loss [61]. Moreover, we add a regularization term that encourages Φ_{pred} to predict a smooth trajectory over Δt:

$$\mathcal{L}_{linear}(\hat{\mathbf{v}}_{\Delta t}) = \frac{1}{||\Delta t||} \sum_{t \in \Delta t} ||(\hat{\mathbf{v}}_{t+1} - \hat{\mathbf{v}}_t) - (\hat{\mathbf{v}}_t - \hat{\mathbf{v}}_{t-1})||_1. \tag{6}$$

Finally, we calculate the overall loss $\mathcal{L}_{motion}$ as a weighted combination of the above terms:

$$\mathcal{L}_{motion}(\mathbf{v}_{\Delta t}, \hat{\mathbf{v}}_{\Delta t}, \tilde{\mathbf{v}}_{\Delta t}) = \mathcal{L}_{traj}(\mathbf{v}_{\Delta t}, \hat{\mathbf{v}}_{\Delta t}, \tilde{\mathbf{v}}_{\Delta t}) + w_{linear}\mathcal{L}_{linear}(\hat{\mathbf{v}}_{\Delta t}). \tag{7}$$

4 Experiments

Datasets. We evaluate our method on two popular 3D MOT benchmarks. First, we experiment on the NuScenes benchmark [1] which consists of street scenes captured in varying scenery and geographic areas, namely Boston and Singapore. The scenes are captured from a moving vehicle with a variety of sensors attached. It provides six different camera views with a 360-degree coverage of the environment. In total, the dataset contains 1000 sequences sampled at 2 Hz. Second, we evaluate our method on the Waymo Open benchmark [5], which has similar characteristics to the NuScenes benchmark. However, the sampling rate is 10 Hz, and the five cameras mounted on the vehicle cover only about a $5/8$ azimuth angle of the horizon.

Evaluation metrics. We follow the evaluation protocol in NuScenes and use the metrics defined in [55]. The main metrics are average multi object tracking accuracy (AMOTA) and average multi object tracking precision (AMOTP) which are integrals over the MOTA / MOTP metrics from [62] computed via n-point interpolation. AMOTA is defined as $\mathrm{AMOTA} = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}} \mathrm{MOTA}_r$, where $\mathrm{MOTA}_r = \max(0, 1 - \frac{\mathrm{IDS}_r + \mathrm{FP}_r + \mathrm{FN}_r + -(1-r)\mathrm{P}}{r\mathrm{P}})$. Specifically, MOTA_r is a recall normalized MOTA and IDS_r, FP_r, FN_r are number of ID switches, false positives and false negatives at recall r. P is the total number of ground truth positives. AMOTP is defined as $\mathrm{AMOTP} = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}} \frac{\sum_{i,t} d_{i,t}}{\sum_t \mathrm{TP}_t}$ where $d_{i,t}$ is the BEV distance between track i and its ground truth track at time t and TP_t denotes the number of matched ground truth tracks at time t.

Implementation details. We train the 3D estimation network using ResNet-101 [63] as backbone with batch size of 32 for 24 epochs with random image flipping as data augmentation on 8 V100 GPUs. The initial learning rate is set to 0.01 with a linear learning rate warm-up in the first 1000 iterations. We decay the learning rate by a factor of 10 at epochs 16 and 22, respectively. We use SGD [64] with a momentum of 0.9 and a weight decay of 0.0001. For NuScenes, we use 1600×900 resolution and train on the training set only. On Waymo Open, we scale the images to 1280×854 resolution and use batch size of 16 during training. We decay the learning rate by a factor of 5 at epochs $8, 12, 16, 20, 22$, respectively. To merge 3D detections across cameras, we use NMS with a 3D IoU threshold of 0.1. To generate the cross-camera trajectory dataset, we use a matching threshold of 2 meters BEV distance. We train Φ_{pred} and Φ_{update} with batch size of 128 for 100 epochs. We use $w_{linear} = 0.001$ and Adam [65] with AMSGrad and a weight decay of 0.0001.

Table 1: Comparison between our single-camera baseline [6], the detect → track → merge paradigm and our method on the NuScenes validation split. We further compare the our cross-camera motion modeling, both with our motion model (VeloLSTM) and a Kalman Filter baseline (KF3D).

Pipeline	Motion Model	AMOTA ↑	AMOTP ↓	RECALL ↑	MOTA ↑	IDS ↓
QD-3DT [6]	-	0.221	1.529	0.371	0.193	5256
	KF3D	0.232	1.530	0.393	0.206	5574
	VeloLSTM	0.247	1.507	0.399	0.218	5919
Detect → Track → Merge	-	0.248	1.500	0.386	0.212	4451
	KF3D	0.256	1.506	0.408	0.225	4256
	VeloLSTM	0.264	**1.490**	0.413	0.235	4470
Detect → Merge → Track	-	0.249	**1.490**	0.405	0.212	2399
	KF3D	0.279	1.501	0.415	0.252	**1982**
	VeloLSTM	**0.283**	**1.490**	**0.444**	**0.262**	2131

Table 2: Comparison of different training schemes on the NuScenes validation set. † denotes using new training scheme for 3D estimation network.

Method	Trajectory data	matching	Total Trajectories	AMOTA ↑	AMOTP ↓	IDS ↓
MV-3DT	Single-camera	2D IoU	29527	0.283	1.490	2131
MV-3DT	Cross-camera	BEV Distance	36112	*0.289*	*1.483*	**2129**
MV-3DT†				**0.311**	**1.433**	2536

4.1 Ablation studies

Cross-camera association. In Table 1, we compare the original pipeline in [6], the naive merging of single camera results (detect → track → merge) and our method as described in section 3.1. In addition, we investigate the influence of the motion model on the tracking result. In particular, we compare our LSTM-based motion model (VeloLSTM) with a Kalman Filter baseline (KF3D). We observe that naive merging helps the overall performance and results in a gain of 1.7% in AMOTA using the VeloLSTM motion model. With our cross-camera association (detect → merge → track), we observe another 1.9% gain in AMOTA compared when using VeloLSTM motion model. We see that the ID switches decrease dramatically by 3788 and highlight the benefit of the cross-camera association. Interestingly, the gain of cross-camera association is very small when not using the motion model (0.1%), supporting our hypothesis that the motion model benefits greatly from the cross-camera association. Moreover, the influence of the motion model on the final tracking performance is the highest when using cross-camera association, *i.e.* 2.6% vs. 1.6% vs. 3.4% difference of using VeloLSTM versus using no motion model. This underlines the importance of cross-camera motion modeling to achieve consistent tracking results. Overall, the results show that our cross-camera association outperforms the baselines by a significant margin.

Cross-camera motion modeling. Table 1 also shows that the quality of motion modeling is improved both through cross-camera trajectories in inference, *i.e.* we observe a higher gain in AMOTA of 3.4% (VeloLSTM) and 2.0% (KF3D) versus using no motion model. In contrast, motion modeling improved the results in the original pipeline by only 2.6% (VeloLSTM) and 1.1% (KF3D). The effect of training the motion model with cross-camera trajectories (see section 3.2) is illustrated in Table 2. We observe a further improvement of 0.6% in AMOTA compared to utilizing cross-camera trajectories only during inference. Moreover, Table 2 shows that the total number of trajectories for training increases, which can be attributed to the fact that for a trajectory to be considered for motion model training, it needs to be at least $||\Delta t||$ time steps long. Thus, considering cross-camera trajectories helps to increase training data diversity. In sum, we achieve a gain of 4.2% in AMOTA, equating to a 17% relative improvement compared to the single-camera baseline.

Qualitative comparison. Figure 2 illustrates MV-3DT results compared to the baselines in Table 1. Among all methods, only our proposed MV-3DT keeps coherent ID for the object moving across cameras. Moreover, it is apparent that the object orientation and 3D location of the objects are

Table 3: Comparison with MUTR3D [60] with different motion models. We report tracking performance on the NuScenes validation split. ‡We use DETR3D [32] as detector for fair comparison.

Method	Motion model	AMOTA ↑	AMOTP ↓	RECALL ↑	MOTA ↑	IDS ↓
MUTR3D [60]‡	SimpleTrack	0.293	**1.307**	0.418	0.263	**1695**
	FFN	0.294	1.498	0.427	0.267	3822
MV-3DT‡	-	0.307	1.394	0.434	0.263	2419
	KF3D	0.353	1.379	0.478	0.316	2050
	VeloLSTM	**0.359**	1.361	**0.498**	**0.326**	2152

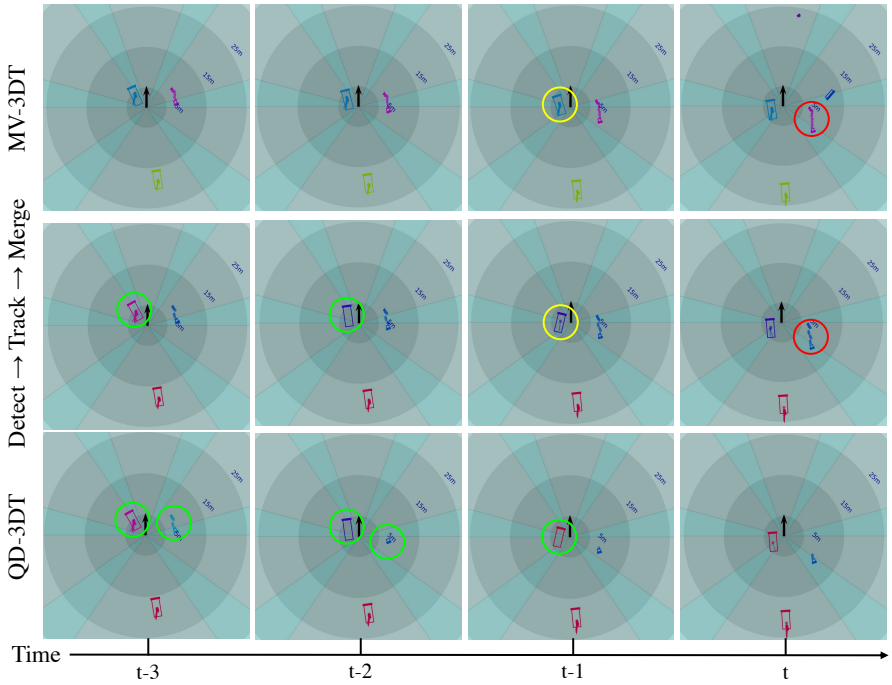

Figure 2: Qualitative comparison between different types of cross-camera aggregation. We plot the overlapping regions across cameras in light blue, and depict object identity as box color. Both baselines fail to track an object moving across cameras (green circles) and exhibit poor localization (yellow circles). In contrast, MV-3DT generates a smooth and precise object trajectory (red circles).

much more consistent in MV-3DT (yellow and red circles), which shows the benefit of cross-camera motion modeling.

3D estimation network training scheme. Following [32] and different from [6], we optimize the 3D estimation network training scheme by increasing the learning rate of the 3D detection head by a factor of 10. As shown in Table 2, this improves the overall performance by 2.2% in AMOTA.

4.2 Comparison to state of the art

Comparison to MUTR3D. To fairly compare against MUTR3D, we use the same detector, namely DETR3D [32]. In Table 3, we show the comparison to MUTR3D with different motion models. MUTR3D reports results with two different motion models, a Kalman Filter based one from [66] and their own motion model based on a feed-forward network (FFN). While their motion model can only improve marginally upon the Kalman Filter baseline, our motion model significantly outperforms the Kalman Filter baseline (KF3D). Moreover, our cross-camera association scheme is much more effective than the transformer-based one in MUTR3D, as can be seen from the ID switches which are much lower for our method. Note that while MUTR3D with motion model of [66] has the lowest ID switches, the detection recall is also much lower.

Comparison on the NuScenes benchmark. We evaluate our method on both the validation and testing splits of NuScenes in Table 4. On the validation split, we outperform the state-of-the-art

Table 4: Comparison with existing camera-based methods on the NuScenes benchmark. † denotes using new training scheme, ‡ denotes using DETR3D [32] as detector.

Method	Split	AMOTA ↑	AMOTP ↓	RECALL ↑	MOTA ↑	IDS ↓
DEFT [8]	Validation	0.201	N/A	N/A	0.171	N/A
QD-3DT [6]		0.247	1.507	0.405	0.221	5919
MUTR3D [60]‡		0.294	1.498	0.427	0.267	3822
MV-3DT†	Validation	0.311	1.433	0.472	0.278	2536
MV-3DT‡		**0.359**	**1.361**	**0.498**	**0.326**	**2152**
CenterTrack [7]	Testing	0.046	1.543	0.233	0.043	3807
DEFT [8]		0.177	1.564	0.338	0.156	6901
QD-3DT [6]		0.217	1.550	0.375	0.198	6856
MUTR3D [60]‡		0.270	1.494	0.411	0.245	6018
MV-3DT†	Testing	0.284	1.470	0.427	0.251	**2945**
MV-3DT‡		**0.357**	**1.343**	**0.472**	**0.317**	3070

method MUTR3D by 6.5% in AMOTA with the same detector, and still by 1.7% with a weaker detector. Moreover, our AMOTP is significantly lower (1.361 vs. 1.498) meaning that our trajectories are more precise than those of existing works. In addition, we also have the lowest number of ID switches, meaning that our association works the best. The results on the testing set corroborate those findings, *i.e.* we outperform the previous state-of-the-art method MUTR3D by 8.7% in AMOTA and achieve the lowest AMOTP and ID switches.

Comparison on the Waymo Open benchmark. We compare against the vehicle tracking results reported in [6] on the validation split of Waymo Open. The comparison in Table 5 shows that our method performs better in MOTA than QD-3DT, while also having a lower Mismatch ratio, meaning that we have both better overall performance and better association. In particular, we outperform QD-3DT significantly on each of the 3D IoU thresholds. Note that we do not evaluate the result with the IoU threshold of 0.7, as it is too restrictive for camera-based methods to achieve a meaningful score.

Table 5: Comparison of 3D Vehicle Tracking at Level 2 difficulty on the validation set of the Waymo Open benchmark.

Method	3D IoU	MOTA ↑	Mismatch ↓
QD-3DT [6]	0.3	0.1867	0.0134
	0.5	0.0308	0.0055
MV-QDT	0.3	**0.2032**	**0.0069**
	0.5	**0.0480**	**0.0018**

4.3 Limitations

Occlusion and truncation. Our method could fail if the predicted 3D detection is amiss. Specifically, in cases of high occlusion or truncation, the object's 3D properties are often ambiguous to the observer, *e.g.* an object's 3D center could even be out of the camera FOV. This is especially pronounced for big or close objects that are often only partially visible. While combining information from multiple time steps in our motion model helps alleviate those issues, integrating this information during the detection stage could further improve the results.

Classification. The challenges mentioned above can also influence the classification performance of the 3D detector. If an object is highly occluded or truncated, it is often challenging to infer its category from a single image, especially for semantically similar classes (*e.g.* bus and truck). However, if the category is estimated incorrectly, object identity may not be preserved.

5 Conclusion

We present MV-3DT, an effective solution to multi-camera 3D tracking. The key idea behind our work is that we merge 3D detections of multiple cameras into a single 3D tracking component that handles both temporal and cross-camera association. By leveraging cross-camera association, we can improve the quality of our motion model, which in turn further improves object association and leads to smoother trajectories across cameras. We obtain state-of-the-art results on the widely-used NuScenes [1] benchmark and outperform existing methods by a large margin.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[2] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. 2021.

[3] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Level 5 perception dataset 2020. https://level-5.global/level5/data/, 2019.

[4] Y. Liao, J. Xie, and A. Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[5] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[6] H.-N. Hu, Y.-H. Yang, T. Fischer, T. Darrell, F. Yu, and M. Sun. Monocular quasi-dense 3d object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[7] X. Zhou, V. Koltun, and P. Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, 2020.

[8] M. Chaabane, P. Zhang, J. R. Beveridge, and S. O'Hara. Deft: Detection embeddings for tracking. *arXiv preprint arXiv:2102.02267*, 2021.

[9] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[10] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. POI: multiple object tracking with high performance detection and appearance feature. In *European Conference on Computer Vision Workshop*, 2016.

[11] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *International Conference on Image Processing*, 2017.

[12] P. Bergmann, T. Meinhardt, and L. Leal-Taixé. Tracking without bells and whistles. *arXiv preprint arXiv:1903.05625*, 2019.

[13] Z. Lu, V. Rathod, R. Votel, and J. Huang. Retinatrack: Online single stage joint detection and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[14] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[17] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, 2017.

[19] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3D object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, 2015.

[20] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3D object detection for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[21] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. 3D bounding box estimation using deep learning and geometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[22] G. Brazil and X. Liu. M3D-RPN: Monocular 3D region proposal network for object detection. In *IEEE International Conference on Computer Vision*, 2019.

[23] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder. Disentangling monocular 3D object detection. In *IEEE International Conference on Computer Vision*, 2019.

[24] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon. Is pseudo-lidar needed for monocular 3d object detection? In *IEEE International Conference on Computer Vision*, 2021.

[25] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[26] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan. Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving. In *IEEE International Conference on Computer Vision*, 2019.

[27] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander. Categorical depth distribution network for monocular 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[28] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[29] S. Shi, X. Wang, and H. Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[30] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[31] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 2020.

[32] Y. Wang, V. Guizilini, T. Zhang, Y. Wang, H. Zhao, , and J. M. Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *The Conference on Robot Learning (CoRL)*, 2021.

[33] A. Bewley, Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft. Simple online and realtime tracking. In *International Conference on Image Processing*, 2016.

[34] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking. In *European Conference on Computer Vision*, 2018.

[35] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 FPS with deep regression networks. In *European Conference on Computer Vision*, 2016.

[36] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *IEEE International Conference on Computer Vision*, 2017.

[37] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *IEEE International Conference on Computer Vision*, 2015.

[38] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese CNN for robust target association. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2016.

[39] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020.

[40] J. Pang, L. Qiu, X. Li, H. Chen, Q. Li, T. Darrell, and F. Yu. Quasi-dense similarity learning for multiple object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[41] D. Held, J. Levinson, and S. Thrun. Precision tracking with sparse 3d and dense color 2d data. In *IEEE International Conference on Robotics and Automation*, 2013.

[42] D. Mitzel and B. Leibe. Taking mobile multi-object tracking to the next level: People, unknown objects, and carried items. In *European Conference on Computer Vision*, 2012.

[43] A. Osep, W. Mehner, M. Mathias, and B. Leibe. Combined image-and world-space tracking in traffic scenes. In *IEEE International Conference on Robotics and Automation*, 2017.

[44] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *IEEE International Conference on Robotics and Automation*, 2018.

[45] J. Luiten, T. Fischer, and B. Leibe. Track to reconstruct and reconstruct to track. *IEEE Robotics and Automation Letters*, 2020.

[46] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702*, 2021.

[47] P. Sun, J. Cao, Y. Jiang, R. Zhang, E. Xie, Z. Yuan, C. Wang, and P. Luo. Transtrack: Multiple-object tracking with transformer. *arXiv preprint arXiv: 2012.15460*, 2020.

[48] F. Zeng, B. Dong, T. Wang, X. Zhang, and Y. Wei. Motr: End-to-end multiple-object tracking with transformer. *arXiv preprint arXiv:2105.03247*, 2021.

[49] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[50] X. Han, Q. You, C. Wang, Z. Zhang, P. Chu, H. Hu, J. Wang, and Z. Liu. Mmptrack: Large-scale densely annotated multi-camera multiple people tracking benchmark. *arXiv preprint arXiv:2111.15157*, 2021.

[51] E. Ristani and C. Tomasi. Features for multi-target multi-camera tracking and re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[52] Y. He, X. Wei, X. Hong, W. Shi, and Y. Gong. Multi-target multi-camera tracking by tracklet-to-target assignment. *IEEE Transactions on Image Processing*, 2020.

[53] K. Shim, S. Yoon, K. Ko, and C. Kim. Multi-target multi-camera vehicle tracking for city-scale traffic management. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2021.

[54] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, X. Yang, Y. Yao, L. Zheng, P. Chakraborty, C. E. Lopez, A. Sharma, Q. Feng, V. Ablavsky, and S. Sclaroff. The 5th ai city challenge. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2021.

[55] X. Weng, J. Wang, D. Held, and K. Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[56] T. Yin, X. Zhou, and P. Krähenbühl. Center-based 3d object detection and tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[57] C. Luo, X. Yang, and A. Yuille. Exploring simple 3d multi-object tracking for autonomous driving. In *IEEE International Conference on Computer Vision*, 2021.

[58] Y. Wang, K. Kitani, and X. Weng. Joint object detection and multi-object tracking with graph neural networks. In *IEEE International Conference on Robotics and Automation*, 2021.

[59] A. Kim, A. Ošep, and L. Leal-Taixé. Eagermot: 3d multi-object tracking via sensor fusion. In *IEEE International Conference on Robotics and Automation*, 2021.

[60] T. Zhang, X. Chen, Y. Wang, Y. Wang, and H. Zhao. Mutr3d: A multi-camera tracking framework via 3d-to-2d queries. *arXiv preprint arXiv:2205.00613*, 2022.

[61] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 1964.

[62] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008, 01 2008. doi:10.1155/2008/246309.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[64] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] Z. Pang, Z. Li, and N. Wang. Simpletrack: Understanding and rethinking 3d multi-object tracking. *arXiv preprint arXiv:2111.09621*, 2021.