Exploring Simple 3D Multi-Object Tracking for Autonomous Driving

Chenxu Luo[1,2] Xiaodong Yang[1*] Alan Yuille[2]
[1]QCraft [2]Johns Hopkins University

Abstract

3D multi-object tracking in LiDAR point clouds is a key ingredient for self-driving vehicles. Existing methods are predominantly based on the tracking-by-detection pipeline and inevitably require a heuristic matching step for the detection association. In this paper, we present SimTrack to simplify the hand-crafted tracking paradigm by proposing an end-to-end trainable model for joint detection and tracking from raw point clouds. Our key design is to predict the first-appear location of each object in a given snippet to get the tracking identity and then update the location based on motion estimation. In the inference, the heuristic matching step can be completely waived by a simple read-off operation. SimTrack integrates the tracked object association, new-born object detection, and dead track killing in a single unified model. We conduct extensive evaluations on two large-scale datasets: nuScenes and Waymo Open Dataset. Experimental results reveal that our simple approach compares favorably with the state-of-the-art methods while ruling out the heuristic matching rules.

1. Introduction

3D multi-object tracking is a crucial component in an autonomous driving system as it provides pivotal information to facilitate various onboard modules ranging from perception, prediction to planning. LiDAR is the most commonly used sensor that a self-driving vehicle relies on to perceive its surroundings. Thus, tracking in LiDAR point clouds has been attracting increasing interests with the rapid development of self-driving vehicles in recent years.

Multi-object tracking is a long-standing task in computer vision and has been extensively studied in image sequence domain. Arguably, the tracking-by-detection is the most popular tracking paradigm, which first detects objects for each frame and then associates them across frames. These methods have shown promising results and benefited from huge progress in image object detection. They usually formulate the association step as a bipartite matching problem. Most existing works therefore focus on better defining the affinity matrix between tracked objects and new detections. In the matching criteria design, the motion [2] and appearance [32] are widely adopted as the association cues.

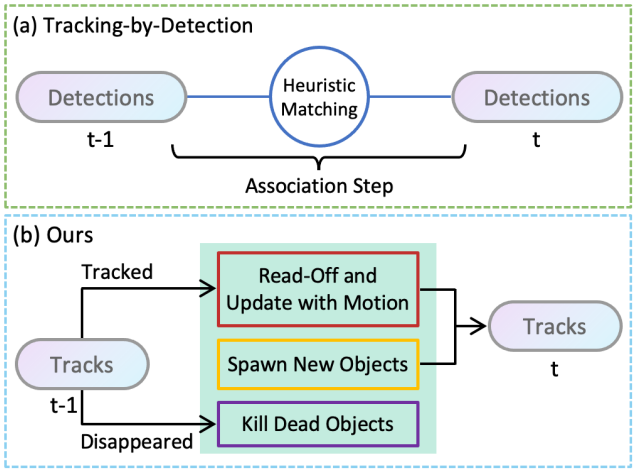


Figure 1: An overview of the tracking-by-detection pipeline and our approach. (a) performs 3D object detection in each point cloud and then match detected objects through an association step, which involves complex heuristic rules. (b) reads-off tracking identities and updates object locations using estimated motion, and at the same time manages newborn and dead tracks. Our model handles the three cases in a single forward pass without requiring heuristic matching.

For 3D multi-object tracking with LiDAR, the tracking-by-detection pipeline also plays the dominant role [6, 27]. Accordingly, in order to obtain the final tracking result, current methods inevitably require a heuristic matching step to link detected objects over time in a separate stage. There exists numerous hand-crafted rules when performing such a step. As compared in the supplementary material, different matching criteria and corresponding threshold for each specific object class substantially impact the final tracking performance. This also happens in the track life management, which is used to handle new-born objects and dead tracks. It is a common practice for these methods to initialize a track only when an object continuously presents for a certain number of frames in order to filter out false detections, and keep disappeared objects for several frames to

*Correspondence to xiaodong@qcraft.ai

tackle occlusion. Unfortunately, all these heuristic rules are not trainable and highly depend on their hyper-parameters that demand huge efforts to tune. What is worse, such rules and hyper-parameters are often data and model dependent, making it hard to generalize and laborious to re-tune when applying to new scenarios.

The main reason for the requirement of an additional heuristic matching step is the lack of connection between frames in conducting object detection. Recently, some methods [18, 31] estimate velocity or predict the location of an object in consecutive frames to provide such a connection across frames. However, they merely treat the forecasted detections as a bridge for object matching instead of using them as the final tracking output. Moreover, they only take into account the location relationship of objects between frames, without modeling the confidence for the association. So the confidence score only reflects the detection confidence in a single frame. As a result, these methods are often prone to spurious detections and have to manually decide the number of frames to keep for the purpose of dealing with occluded objects. Another issue lies in how to process newborn objects in an online tracking system. Existing methods [1, 18] re-detect all objects in the current frame so that matching is still in need to differentiate the newborn objects from the tracked ones.

In light of the above observation, we present **SimTrack**: a simple model for 3D multi-object tracking in point clouds. We simplify the existing hand-crafted tracking algorithms without requiring the heuristic matching step. Our approach is flexible to build upon the commonly used pillar or voxel based 3D object detection networks [9, 37]. We propose a novel hybrid-time centerness map, which represents objects by their first-appear locations within the given input period. Based on this map, we can directly link current detections to previous tracked objects without the need for additional matching. Since this map formulates the detection and association of an object at the same time, our model is able to inherently provide the association confidence between frames. Also, we introduce a motion updating branch to estimate the motion of tracked objects in order to update from their first-appear locations to the current positions. For the new-born objects and dead tracks, they can be determined simply by confidence thresholding as regular detections on the same map, thus removing the manual track life management too. As illustrated in Figure 1, our model eliminates the heurstic matching step, and unifies the tracked object linking, new-born object detection as well as dead track killing in a single forward pass.

To our knowledge, this work provides the first learning paradigm that is able to get rid of the heuristic matching step for 3D multi-object tracking in point clouds, and therefore, remarkably simplifies the whole tracking system. We introduce a novel end-to-end trainable model for joint detection and tracking through a hybrid-time centerness map and a motion updating branch. Experimental results reveal that this simple approach compares favorably to the existing methods. Our code and model will be made available at `https://github.com/qcraftai/simtrack`.

2. Related Work

2D Multi-Object Tracking. With continuous progress in image object detection [20, 21, 22, 25], most methods follow the tracking-by-detection pipeline that first detects objects for individual frames and then associates two sets of detections over time. We can categorize the association step into two primary groups: motion based and appearance based. The motion based methods utilize temporal modeling [30] to update detections and realize matching with a distance or intersection over union metric (IOU). Kalman filter [8] is widely adopted by the methods of this group for state estimation [2]. Some works predict location offset to facilitate motion modeling [7, 18, 34]. The appearance based methods instead consider visual appearance affinity for the same object in a sequence. Most of them apply re-identification [33, 39] to drive appearance feature learning to establish identity correspondence [13, 32].

Unlike the motion based tracking-by-detection methods that treat the predicted boxes only as a proxy for matching, the tracking-by-regression paradigm performs tracking by directly regressing previous locations to new positions in current frame. In [1] Trackor starts from the past location of each object and applies the region-pooled object detection features from current frame to obtain the updated location. It depends on an additional object detector to deal with newborn objects, and has to update multiple regions of interest and separate tracked objects from new detected objects via some heuristic rules. In contrast, our approach can directly produce the tracked and new-born objects in a single forward pass with no heuristic post processing.

Recently, inspired by the success of Transformer [26], several methods are developed to conduct joint detection and multi-object tracking through attention operations. TrackFormer [16] employs track query embeddings to follow object location changing over time in an auto-regressive way and adopts object queries as in [4] to deal with newborn objects. TransTrack [23] adopts the query-key mechanism to detect objects in current frames and associate them between frames by the learned object queries.

3D Multi-Object Tracking. In this field, the dominant methods exploit the tracking-by-detection pipeline. Due to the lack of appearance and texture cues in point clouds, the LiDAR based tracking models hinge on motion for association. AB3DMOT [27] extends Kalman filter to 3D for motion state estimation. CenterPoint [31] estimates the velocity of each object by adding a velocity regression head. FaF [15] associates objects through prediction. PnPNet [10]

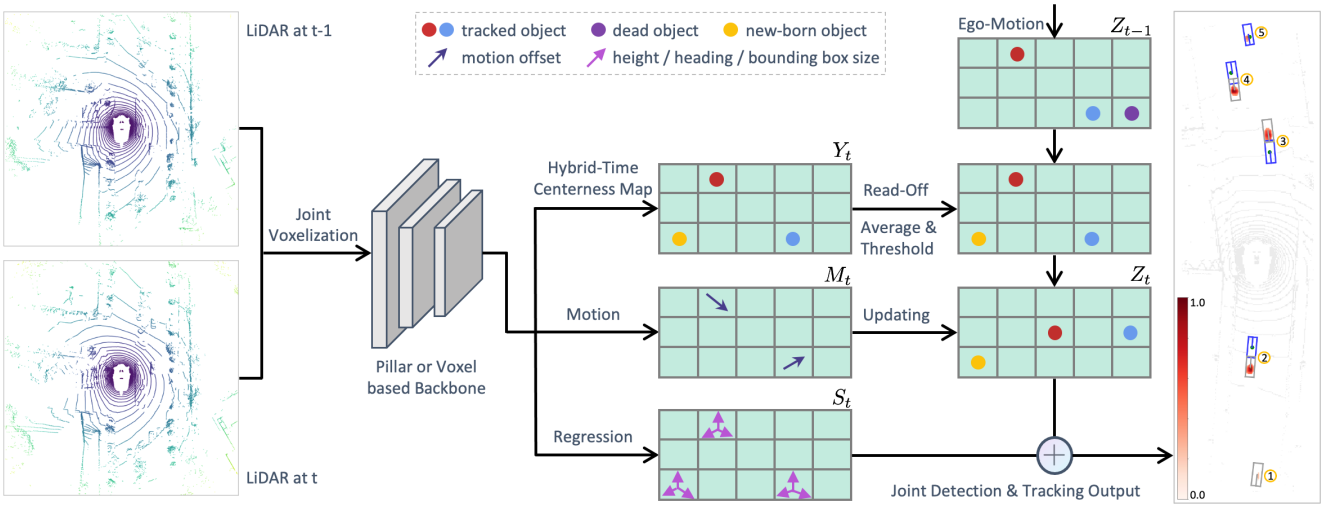


Figure 2: A schematic overview of SimTrack. Our model consists of a hybrid-time centerness map branch that detects the first-appear location of each object in the input snippet, a motion updating branch to predict the motion of an object within the period, and a regression branch to estimate other object properties. During the inference, we first transform the previous updated centerness map Z_{t-1} to the current coordinate system through ego-motion, and average it with the current hybrid-time centerness map Y_t which is next thresholded to remove the dead objects, and then read-off the tracking identities that share the same cells on Y_t and Z_{t-1}. After that, we update the tracked objects to their current locations using the predicted motion to obtain Z_t. We show a zoom-in area of the point cloud to illustrate the detection and tracking output, where the gray and blue boxes are detected objects at previous and current sweeps. ID (1) is a dead object with low confidence. ID (2-4) are tracked objects that are correctly localized from peaks in the confidence heatmap, and their current locations are accurately updated by the predicted motion. ID (5) is a new-born object.

learns affinity matrix between objects using 3D features and trajectories. Chiu et al. [5] incorporate appearance feature distance from cameras into the distance metric to enhance association. Most of them require a bipartite matching step, either using the Hungarian algorithm or a greedy matching algorithm, to acquire the final tracking output.

Detection and Motion. 3D object detection provides the fundamental tools for 3D multi-object tracking. In [37] VoxelNet applies 3D convolutions on the voxel features extracted by [19]. SECOND [29] enhances efficiency by using sparse 3D convolutions. CBGS [38] improves accuracy by class-balanced grouping and sampling. PointPillars [9] is developed to collapse height dimension and employ 2D convolutions to achieve better efficiency. Meanwhile, some recent methods [14, 28] demonstrate promising results for point cloud pillar motion estimation using self-supervision or proxy motion supervision derived from tracking.

3. Method

As illustrated in Figure 2, SimTrack unifies the tracked object linking, new-born object detection and dead object removal in a single end-to-end trainable model. Our key design to rule out the heuristic matching step and achieve the desired simplified tracking is based on the proposed hybrid-time centerness map and motion updating branch.

3.1. Preliminary

Our approach exploits the center based representation for 3D objects. Thanks to the innate connection between detection and tracking under this representation, tracks can be described as paths formed by points in space and time. Here we briefly review the center based 3D object detection. Given a raw point cloud, we first voxelize it into regular grids using either pillars [9] or voxels [37]. We extract the feature of each pillar or voxel by a small PointNet [19]. After that the standard 2D or 3D convolutions are used to compute the features in bird's eye view (BEV). As for the detection head, we represent each object by its center location on the centerness map similar to [31]. For training, a 2D Gaussian heatmap around the center of each object is created to form the target centerness map. All detection output including the centerness map, local offset, object size and heading can be generated by the detection head.

3.2. Overview

Let $P^t = \{(x, y, z, r)_i\}$ indicate an orderless point cloud consisting of the measurements of coordinates (x, y, z) and reflectance r at time t. Our model takes a snippet of point clouds as input. For simplicity, we directly combine multiple point clouds by transforming the past sweeps to the cur-

rent coordinate system through ego-motion compensation. As a common practice, we also add a relative timestamp to each point such that a point is represented as (x, y, z, r, Δ_t), where Δ_t is the relative timestamp to the current sweep. After the voxelization and feature extraction, our detection head makes use of the centerness map to detect the first-appear location of an object in the input snippet and estimates the object motion within the period. In the inference, we simply read-off the tracking identity of an object from previous centerness map and then use the predicted motion to update the object to its current location.

3.3. Joint Detection and Tracking

As discussed above, in order to eliminate the heuristic matching and manual track life management, we propose to perform joint detection and tracking in a simplified model through the combination of a hybrid-time centerness map and a motion updating branch.

Hybrid-Time Centerness Map. With the intention to provide linkage with previous detections and detect new-born objects simultaneously, we propose a hybrid-time centerness map. Specifically, our model takes two consecutive LiDAR sweeps at $t-1$ and t as input. For the target centerness map, we represent each object by its center location, where this object first appears in the input sequence. Suppose that the ground-truth object locations at frame $t-1$ and t are respectively $\{d_i^{t-1}\}_{i=1,\cdots,n_{t-1}}$ and $\{d_i^t\}_{i=1,\cdots,n_t}$. Our target assigning strategy is defined as follows.

- For a tracked object that presents in both $t-1$ and t frames, denoted as d_i^{t-1} and d_j^t, we create its target heatmap at d_i^{t-1}, which is the first-appear location of this object in the input sequence.
- For a dead object that only shows up in the first frame $t-1$ but disappears in the second frame t, we treat it as a negative example and do not assign any target heatmap for this object.
- For a new-born object that only presents in the second frame t, we create its target heatmap at d_i^t.

In this manner, for a tracked object, we can directly link with its previous detection by reading-off the identity at same location of the updated centerness map (see details bellow) from the previous timestamp. For the dead objects, they can be simply removed by thresholding the confidence scores. And for the new-born objects, we perform regular detection on the same hybrid-time centerness map. Therefore, our hybrid-time centerness map builds the foundation to merge tracked object association, dead object removal, and new-born object detection in a single unified model. Furthermore, we can utilize the confidence scores obtained from this hybrid-time centerness map to imply both detection confidence (i.e., the probability of an object existing in current timestamp) and association confidence (i.e., the probability of an object linking to its previous location).

Motion Updating Branch. As aforementioned, we link each tracked object to its previous location on the hybrid-time centerness map to establish identity correspondence. However, to achieve an online tracking system, we need to further obtain the current location of the object. We thus introduce a motion updating branch to estimate the offset of an object between two sweeps. In practice, for each object, at its center location in the first frame, we regress the offset to its current location: $(\Delta_u, \Delta_v) = (u_t - u_{t-1}, v_t - v_{t-1})$, where (u, v) is the object center coordinates. We then take advantage of this motion field to transform a hybrid-time centerness map into an updated centerness map.

We note that some previous methods such as CenterPoint [31] also estimates object velocity. However, the main difference is that they only regard motion as an assistant to conduct matching. They use motion to propagate the detections of current frame and use them to match with the detections from previous frame. In other words, the propagated boxes only serve as a bridge for matching the detected objects across frames, instead of being used as the final tracking result. SimTrack however shows that our hybrid-time detection and motion estimation can be combined to produce the tracking output without the need of heuristic matching. Moreover, during the inference, CenterPoint requires to manually tune a class-specific distance threshold to determine whether a motion-derived box can be matched with a detected box. In comparison, our model remarkably simplifies tracking pipeline through a single forward pass to obtain both detections and correspondences.

Other Regression Branches. In addition to the motion, we regress other 3D object properties including height z, bounding box size (w, l, h), and heading in the format of $(\sin\theta, \cos\theta)$ where θ is the yaw angle of the bounding box.

Loss Functions. When training the hybrid-time centerness map, we adopt the focal loss similar to [31, 35]:

$$\mathcal{L}_{\text{cen}} = \frac{-1}{N} \sum_{c,d_i} \begin{cases} (1 - Y_{c,d_i})^{\alpha} \log(Y_{c,d_i}), & \text{if } \tilde{Y}_{c,d_i} = 1 \\ (1 - \tilde{Y}_{c,d_i})^{\beta} (Y_{c,d_i})^{\alpha} & \\ \log(1 - Y_{c,d_i}), & \text{otherwise} \end{cases} \tag{1}$$

where $\tilde{Y}$ and Y indicate the target and predicted hybrid-time centerness maps, N denotes the number of objects, α and β are the hyper-parameters of focal loss [11]. For the motion updating branch, we enforce the standard ℓ_1 loss:

$$\mathcal{L}_{\text{mot}} = \frac{1}{N} \sum_{i=1}^{N} |\tilde{M}_{d_i} - M_{d_i}|, \tag{2}$$

where $\tilde{M}$ denotes the ground truth motion map, and M is the predicted motion map. Similarly, we also make use of the standard ℓ_1 loss for other regression branches:

$$\mathcal{L}_{\text{reg}} = \frac{1}{N} \sum_{i=1}^{N} |\tilde{S}_{d_i} - S_{d_i}|, \tag{3}$$

where $\tilde{S}$ and S represent other ground truth and predicted regression maps for object height, size and heading. We only compute these losses at the center locations d_i on the corresponding maps. In summary, the total objective is a weighted sum of the three loss functions:

$$\mathcal{L}_{\text{total}} = \omega_{\text{cen}}\mathcal{L}_{\text{cen}} + \omega_{\text{mot}}\mathcal{L}_{\text{mot}} + \omega_{\text{reg}}\mathcal{L}_{\text{reg}}, \tag{4}$$

where ω_{cen}, ω_{mot} and ω_{reg} are the balancing coefficients to control the importance of three loss terms.

Backbone Network. SimTrack is flexible to be built on various backbones. In the experiments, we mainly use PointPillars [9] as the pillar based backbone due to its computational efficiency for onboard deployment. To compare with other methods, we also evaluate with the more accurate and larger VoxelNet [38] as the voxel based backbone.

Online Inference. During the inference, the updated centerness map Z records the tracking identity, center location, and confidence score of each object. And the tracking identity is placed at the object center location. For the initial frame in a sequence, our approach takes only one sweep as input and performs detection to initialize the updated centereness map Z_0. For the later frames, the model takes current sweep and one previous sweep as input. All point clouds are transformed to current vehicle coordinate system using ego-motion.

In comparison to the existing methods that depend on a heuristic matching step, SimTrack uses a simple read-off to establish the association. As illustrated in Figure 2, at time t we first transform the updated centerness map Z_{t-1} of previous timestamp to the current coordinate system using ego-motion. We then average Z_{t-1} with the current hybird-time centerness map Y_t. For each object center, if there is an existing tracking identity at the same location on Z_{t-1}, the object is regarded as a tracked object and reads-off this tracking identity. We initialize a new track for each of the rest object centers. In our approach, there is no need to specifically handle the dead objects as they can be naturally discarded when thresholding Y_t. Afterwards, we update Y_t to Z_t by using the predicted motion map M_t to obtain the current locations of tracked objects. We summarize the inference outline of our approach in Algorithm 1.

Algorithm 1: Online Inference of SimTrack

Input: a sequence of point clouds $P_0, P_1, \cdots$
Output: joint detection and tracking output
for $t = 0, 1, \cdots$ **do**
  if $t == 0$ **then**
    $Y_0, M_0, S_0 \leftarrow \text{Network}(P_0)$
    threshold and NMS on Y_0
    initialize tracking identities on Y_0
    $Z_0 \leftarrow Y_0$
  else
    transform Z_{t-1} by ego-motion
    $Y_t, M_t, S_t \leftarrow \text{Network}(P_t, P_{t-1})$
    $Y_t \leftarrow (Y_t + Z_{t-1})/2$
    threshold and NMS on Y_t
    read-off tracking identities from Z_{t-1} to Y_t
    initialize new-born tracking identities on Y_t
    $Z_t \leftarrow \text{Update}(Y_t, M_t)$
  end
end

4. Experiments

In this section, we first describe our experimental setup including datasets, evaluation metrics and implementation details. A variety of ablation studies and related analysis are then provided for in-depth understanding of different design choices in our approach. We report extensive comparisons with the state-of-the-art methods on the two benchmarks.

4.1. Datasets

We extensively evaluate our proposed approach on the two large-scale autonomous driving datasets: nuScenes [3] and Waymo Open Dataset [24]. **nuScenes** contains 1000 scenes, each of which is around 20 seconds, and the point clouds are captured by a 32-beam LiDAR. This dataset is split to 700, 150 and 150 scenes for training, validation and testing, respectively. The frequency of LiDAR is 20Hz and the annotations are provided at 2Hz. There are 10 classes in total for detection, and among them, 7 moving classes are used for the tracking evaluation. Following the official evaluation protocol, we set the detection and tracking range to [-51.2m, 51.2m]×[-51.2m, 51.2m]. **Waymo** contains 798 training and 202 validation sequences, and the point clouds are captured by 5 LiDARs at 10 Hz. The official evaluation is carried out in the range [-75m, 75m]×[-75m, 75m], and breaks down the performance on two difficulty levels: LEVEL_1 and LEVEL_2, where the former evaluates on objects with more than five points and the latter includes objects with at least one point.

4.2. Evaluation Metrics

We follow the official evaluation metrics defined by the two benchmarks for comparison. nuScenes adopts the center distance with a threshold of 2m in BEV, i.e., an object within 2m of a ground truth is considered to be true positive. Waymo uses 3D IOU of 0.7 for the vehicle class. It applies MOTA as the main evaluation metric, which penalizes three error types: false alarms (FP), missing objects (FN) and identity switch (IDS) at each timestamp. Waymo evaluation system automatically picks up a best confidence threshold for MOTA. nuScense on the other hand employs AMOTA to compute the average MOTA under different recalls. We also report FRAGS that counts for the fragments of a track caused by missing detections.

Method	Detection	Car	Pedestrian	Bicycle	Motor	Bus	Trailer	Truck	Overall
CenterPoint	Pillar-Det	82.5	**69.6**	20.2	40.0	78.4	40.8	**63.9**	56.5
Kalman Filter	Pillar-Det	74.7	60.3	14.0	36.0	74.9	39.1	59.5	51.2
CenterPoint	Pillar-Track	79.4	64.0	24.7	53.5	**78.9**	46.3	59.1	58.0
Kalman Filter	Pillar-Track	76.6	68.3	25.6	54.5	74.6	45.0	57.3	57.4
Ours	Pillar-Track	**84.1**	68.3	**27.7**	**57.6**	76.1	**46.6**	59.2	**60.0**
CenterPoint	Voxel-Det	82.9	**73.6**	40.9	54.6	79.9	48.8	**65.2**	63.7
Kalman Filter	Voxel-Det	75.7	65.7	33.5	52.2	76.7	48.2	61.1	59.0
CenterPoint	Voxel-Track	81.0	70.2	**48.0**	60.6	79.7	50.9	61.1	64.5
Kalman Filter	Voxel-Track	77.5	57.3	41.5	52.4	77.2	49.4	59.1	59.2
Ours	Voxel-Track	**84.3**	71.8	45.3	**64.6**	**80.5**	**54.7**	61.8	**66.1**

Table 1: Comparison of the tracking results using different detection modes with pillar and voxel based backbones on the validation set of nuScenes. We report the overall and per class AMOTA.

Method	Overall					Car			
	AMOTA↑	FP↓	FN↓	IDS↓	FRAGS↓	AMOTA↑	AMOTP↓	IDS↓	FRAGS↓
AB3DMOT [27]	15.1	15088	75730	9027	2557	27.8	1.325	7654	1957
Probabilistic-3D [6]	55.0	17533	33216	950	776	71.9	0.580	541	449
CenterPoint* (Voxel-1440) [31]	63.8	18612	**22928**	**760**	529	82.9	0.384	315	296
Ours (Voxel-1024)	**64.5**	**17443**	26430	1042	**472**	**83.6**	**0.343**	**214**	**186**

Table 2: Comparison of the tracking results on the test set of nuScenes. * denotes using deformable convolution and test-time augmentation. Voxel means voxel based backbone, 1024 and 1440 indicate feature map sizes.

4.3. Implementation Details

We implement our approach in PyTorch [17] based on the codebase of CenterPoint [31] and Det3D [38]. We train our models on 8 TITAN RTX GPUs with a batch size of 8 and 4 per GPU for nuScenes and Waymo, respectively. Each model is trained for 20 epochs on nuScenes and 12 epochs on Waymo. We utilize AdamW [12] as the optimizer and the one-cycle learning rate scheduling. We apply the standard data augmentation including global rotation and scaling, flipping along X and Y axes, as well as 3D object cut-and-paste from other point clouds.

We set the balancing coefficients $(\omega_{\text{cen}}, \omega_{\text{mot}}, \omega_{\text{reg}})$ in Eq. (4) to (1, 1, 0.25) for nuScenes and (1, 1, 1) for Waymo. We extensively compare our approach with Kalman filter and CenterPoint, the two methods that are dominantly used for 3D multi-object tracking. We experiment with two different backbones including pillar based and voxel based to validate the generalizability of our approach. We set pillar size as [0.2m, 0.2m] and voxel size as [0.1m, 0.1m, 0.2m]. Note we do not use higher voxelization resolution for the consideration of computational efficiency during inference. We follow [31] to set the Gaussian heatmap radius when creating the target hybrid-time centerness map. For thresholding the hybrid-time centerness map, we take the default value 0.1 as the one used for detection in [31]. Note this threshold is not further tuned for removing dead tracks so that no additional hyper-parameter is introduced.

4.4. Results on nuScenes

Validation Set. Table 1 shows the tracking comparisons on the validation set. We report the results in the overall and per class AMOTA. Since a tracking approach is largely affected by the detection performance, for better understanding of SimTrack, we provide two types of tracking performance based on the detection results from (i) Pillar/Voxel-Det: the regularly trained detection models that are used by the original tracking methods; and (ii) Pillar/Voxel-Track: our joint detection and tracking models.

As demonstrated in Table 1, our approach significantly outperforms the original CenterPoint and Kalman filter by 3.5% and 8.8% with the pillar based backbone, and 2.4% and 7.1% with the voxel based backbone. By using the detection results of our approach, the tracking performance of CenterPoint and Kalman filter can be both improved. This is due to our better detection and motion estimation (see details in ablation studies) that benefit from the end-to-end coupled detection and tracking training. Nevertheless, our approach still outperforms both of them by around 2% when they use our detection results, suggesting that our better tracking performance is not only due to the better detection but also the proposed tracking design.

It is worth noting that one group of important hyper-parameters for CenterPoint is the maximum distance threshold allowed to be considered matched for each of the different classes. CenterPoint carefully picks the thresholds using

	AMOTA↑	AMOTP↓	IDS↓	FRAGS↓
Unified	**60.0**	**0.774**	**1406**	**412**
Separated	44.6	0.956	4097	761

(a) Comparison of the unified and separated maps on the validation set of nuScenes.

	mAVE↓	Car	Pedestrian	Bicycle	Motor	Bus
Baseline (Pillar)	0.300	0.306	0.231	0.229	0.659	0.600
Ours (Pillar)	**0.201**	**0.207**	**0.206**	**0.166**	**0.254**	**0.348**
Baseline (Voxel)	0.272	0.300	0.227	0.201	0.421	0.469
Ours (Voxel)	**0.191**	**0.209**	**0.208**	**0.132**	**0.234**	**0.304**

(b) Comparison of the velocity estimation on the validation set of nuScenes.

Combining Maps	Overall			Per Class AMOTA↑						
	AMOTA↑	IDS↓	FRAGS↓	Car	Pedestrian	Bicycle	Motor	Bus	Trailer	Truck
✗	50.0	4761	1315	79.2	51.0	14.6	33.7	71.6	45.3	54.9
✓	**60.0**	**1406**	**412**	**84.1**	**68.3**	**27.7**	**57.6**	**76.1**	**46.6**	**59.2**

(c) Evaluation of combining the current hybrid-centerness map with the previous updated centerness map.

Resolution	Overall				Pedestrian				Motor			
	AMOTA↑	AMOTP↓	IDS↓	FRAGS↓	AMOTA↑	AMOTP↓	IDS↓	FRAGS↓	AMOTA↑	AMOTP↓	IDS↓	FRAGS↓
1/4 (0.8m)	60.0	0.774	1406	412	68.3	0.625	1216	246	57.6	0.839	45	9
1/2 (0.4m)	**61.1**	**0.680**	**646**	**320**	**75.0**	**0.504**	**490**	**172**	**65.9**	**0.604**	**7**	**5**

(d) Comparison of the tracking results using pillar based backbone with different centerness map resolutions.

Table 3: A set of ablation studies on the validation set of nuScenes.

the velocity error statistics based on the validation set. Its tracking performance is sensitive to the selected thresholds. For instance, if the threshold for car is changed from 4m to 1m, its AMOTA drops from 82.5% to 81.0%, and if the threshold constraint is relaxed to 10m, its AMOTA further drops to 72.1%. In comparison, our approach fully gets rid of such manually tuned thresholds and is therefore more robust and handy for deployment to new scenarios.

Test Set. We submit the result of our voxel based model to the testing server of the tracking benchmark of nuScenes. For this submission, we do not use any test-time augmentation. As demonstrated in Table 2, our approach without bells and whistles outperforms the enhanced CenterPoint which is equipped with deformable convolutions and test-time augmentation. Especially for the most important car class in autonomous driving, our model reduces IDS and FRAGS from 315 and 296 to 214 and 186.

4.5. Ablation Studies

Occlusion Analysis. Being able to handle occlusion is one of the challenges in 3D multi-object tracking as objects can be partially or fully occluded in point clouds for a while. One common practice is to keep dead tracks for a certain number of frames and update their locations by assuming a constant velocity mode. We observe that this heuristic rule substantially impacts IDS, which for example deteriorates from 238 to 500 if dead objects are not retained for certain pre-defined time in CenterPoint. In contrast, SimTrack implicitly tackles occlusion by combining with the confidence scores on previous updated centerness map and updating with the estimated motion. If an object is occluded in current frame but has strong cues in previous frame, our approach is able to keep the object and conjecture its current location. Figure 3 shows one example where the orange-colored car is heavily occluded for a few frames. Our model can successfully keep tracking of this car, while CenterPoint fails to link with the original identity. The other example demonstrates that inaccurate velocity estimated by CenterPoint results in identity switch, as velocity estimation for small objects like pedestrian is difficult. Our approach can better handle such cases too.

Unified or Separated Maps. Here we demonstrate that conducting tracked object association and new-born object detection on a unified map achieves better performance. We also implement a model using separated maps for tracking and detection. Specifically, we modify the hybrid-time centerness map by providing two channels for each class, one for tracked objects and the other for new-born objects. The target assigning strategy remains the same. Table 3a compares the two designs with the pillar based backbone. It is found that using separated maps performs much worse than using a unified map. We hypothesis that this is due to the extreme unbalancing between the two maps. For a normal scene, the new-born objects only take up a small portion of all objects, thus making training hard.

Combining Maps. As described in Algorithm 1, the current hybrid-time centerness map is averaged with the previous updated centerness map. In Table 3c, we compare this combinatorial design with an alternative that only uses the current hybrid-time centerness map. The overall and per class results are consistently and considerably improved by combining the two maps. This simple averaging operation provides effective temporal fusion, and in particular, is vital to tackle occlusion as analyzed above.

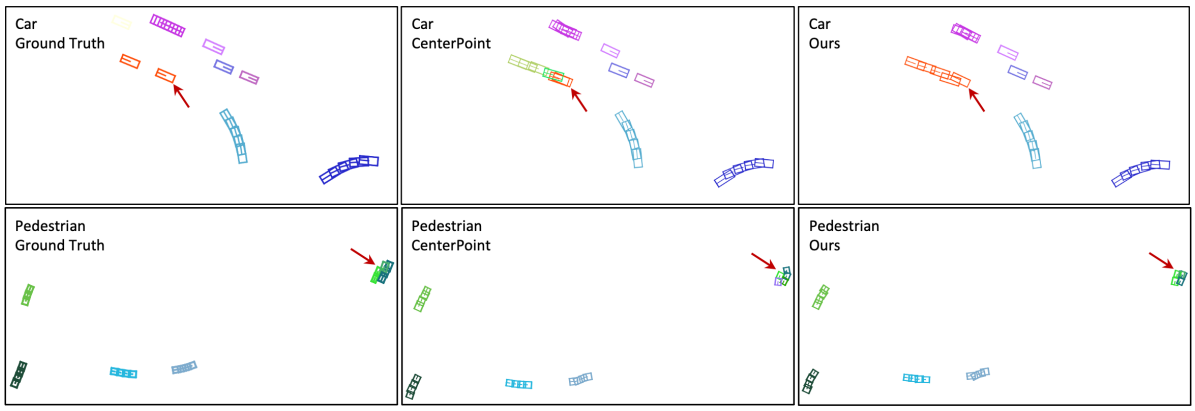


Figure 3: Comparison of the qualitative tracking results on the validation set of nuScenes. Each color encodes an object identity over time. Note due to occlusion the ground truth does not provide the annotation of the orange-colored car.

Method	MOTA↑	MOTP↓	Miss↓	Miss Match↓	FP↓
Baseline [24]	42.5 / 40.1	18.6 / 18.6	40.0 / 43.4	**0.14** / **0.13**	17.3 / 16.4
CenterPoint [31]	51.4 / 47.9	17.6 / 17.6	47.7 / 41.4	0.19 / 0.18	**10.7** / 10.6
Ours	**53.1** / **49.6**	**17.4** / **17.4**	**35.5** / **39.8**	0.20 / 0.19	11.2 / **10.5**

Table 4: Comparison of the vehicle tracking performance on the validation set of Waymo. We report the tracking results with the pillar based backbone, and the numbers are in the format LEVEL_1 / LEVEL_2.

Resolution. Next we show that the performance of our approach can be greatly improved by simply increasing the centerness map resolution, which is in particular effective for small objects such as pedestrian and motorcycle. In the original backbone network, we set pillar size to [0.2m, 0.2m] and downsampling rate to 4, meaning on the centerness map, the size of each cell is [0.8m, 0.8m]. To increase the resolution, we keep the encoder intact but only modify the upsampling layer in decoder to change the downsampling rate to 2. Table 3d shows the overall tracking performance and the specific results of two small object classes: pedestrian and motorcycle. Compared with the lower resolution, using higher resolution improves the tracking performance of the two classes substantially.

Velocity Estimation. In addition to tracking, we show that our approach can produce more accurate velocity for moving objects. We adopt mAVE, the metric officially defined by nuScenes to measure the error of velocity estimation for true positives under different recall rates. Table 3b reports mAVE of all classes with both pillar based and voxel based backbones. We compare our model with CenterPoint based baseline. For the pillar based backbone, our approach reduces the velocity error by 33%, in particular for motorcycle, the velocity error is largely reduced by 61%. It clearly validates that our jointly end-to-end trained detection and tracking model can better exploit the dynamics of moving objects. This improved velocity estimation is potentially beneficial to various downstream tasks such as trajectory prediction and motion planning.

4.6. Results on Waymo

Here we compare the tracking results of the vehicle class on the validation set. In this experiment, we also use the pillar based backbone for the consideration of low latency. As shown in Table 4, our model delivers clear performance gains over the baseline method provided by Waymo. Compared to CenterPoint, our approach obtains better or on-par results under different metrics. Since nuScenes and Waymo have distinct LiDARs and evaluation metrics, our consistent improvements on the two datasets collectively validate the generalizability of SimTrack. More importantly, we achieve the superior results without requiring the heuristic matching and complex tracking life management as commonly used by the competing algorithms.

5. Conclusion

In this paper, we have presented SimTrack, an end-to-end trainable model for 3D multi-object tracking in LiDAR point clouds. Our approach takes a first step to simplify the existing hand-crafted tracking pipelines that involve complex heuristic matching and manual track life management. By combining the proposed hybrid-time centerness map and motion updating branch, our design seamlessly integrates tracked object association, new-born object detection and dead object removal in a single unified model. Extensive experimental results demonstrate the efficacy of our approach. We hope this work can inspire more research toward simple and robust tracking systems for autonomous driving.

Appendix

In this appendix, Section A exemplifies how the representative matching heuristics and related hyper-parameters impact the tracking performance. Section B presents more comparisons between SimTrack and CenterPoint. Section C reports the inference latency of our model. Section D provides more results on nuScenes and Waymo.

A. Heuristic Matching and Hyper-Parameters

Existing tracking methods involve a number of hyper-parameters in heuristic matching. Some widely used ones include matching threshold, maximum number of frames to keep for a dead track, minimum number of frames before initializing a new track, to name a few.

It is known that the tracking performance is sensitive to the hyper-parameter setting in heuristic matching. For the Kalman filter based tracking, the setting of covariance matrix greatly affects the tracking result. For instance, in [6] the AMOTA on the validation set of nuScenes is 37.1 when using the default covariance matrix, but the performance boosts to 51.2 after carefully tuning the covariance matrix based on the statistics of prediction errors.

To highlight the critical role of setting hyper-parameters for the heuristic matching step, we compare the tracking results of CenterPoint [31] with different hyper-parameters in Table 5. Specifically, we exemplify with two representative hyper-parameters: maximum age and maximum distance. The former is used for a dead track to be retained for a certain number of frames before it is removed. This helps when an object is occasionally occluded in a few frames and shows up again. The latter determines the distance threshold that allows to be matched. CenterPoint tunes this threshold based on the distribution of velocity errors on the validation set. As demonstrated in Table 5, the two factors significantly impacts the tracking performance. To obtain a reasonably good result, great efforts are in need to tune these hyper-parameters. As a comparison, our approach is heuristic-free but achieves better performance.

Model	Age	Distance	AMOTA↑	AMOTP↓	IDS↓	FRAGS↓
Center-Point	3	1.0	81.0	43.3	856	247
	3	2.0	83.1	39.5	256	184
	3	4.0	82.5	48.0	238	240
	3	∞	59.6	49.6	318	299
	0	4.0	80.0	48.0	365	352
Ours	-	-	**84.1**	**34.5**	**148**	**122**

Table 5: Impact of the representative heuristics and the setting of related hyper-parameters in the matching step of CenterPoint. We report the tracking results on the validation set (car category) of nuScenes. All results are produced by the pillar based backbone.

B. More Comparisons with CenterPoint

Here we provide more detailed comparisons on MOTA and IDS between SimTrack and CenterPoint under different recall rates. As shown in Figure 4a, our model has much less identity switch under high recall rates. This is because the heuristic matching based tracking methods like CenterPoint suffer from the large amount of false positive detections, while SimTrack is less vulnerable to false positives thanks to our joint detection and tracking design. This advantage makes our approach more robust and stable in particular for the scenarios where a high recall rate is desired. Figures 4b-4d respectively plot the curves of MOTA-Recall for car, pedestrian and motorcycle. Overall, our approach achieves superior MOTA at high recall rates.

C. Inference Latency

Our joint detection and tracking design is flexible to incorporate in a 3D object detection network and only introduces a small computational overhead to to the backbone network. Table 6 compares the inference latency between a detection-only model and our joint detection and tracking model using different centerness map resolutions with the pillar and voxel based backbones. As shown in this table, our approach only slightly increases the inference latency of the detection-only model by 1-2ms. We report the inference time on a single TITAN RTX GPU.

D. More Results on nuScenes and Waymo

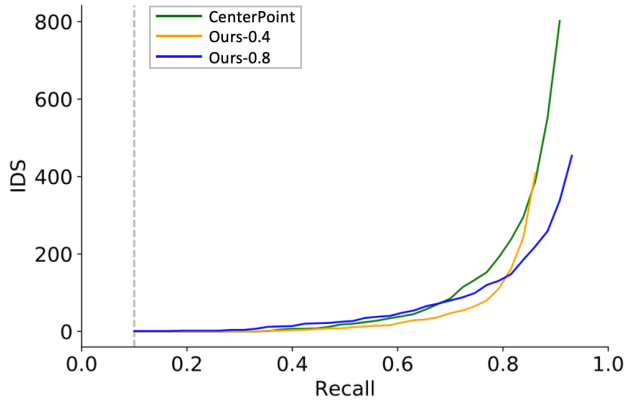
In addition to simplify and improve tracking, SimTrack can also boost the detection accuracy. Table 7 compares the detection results of SimTrack and CenterPoint. We report mAP and NDS of all classes on nuScenes. Note the result on the test set of CenterPoint is also based on its enhanced version as described in the paper. Our joint detection and tracking model can significantly improve the detection per-

Resolution	Pillar Backbone	Voxel Backbone
0.4m×0.4m	36ms / 38ms	65ms / 67ms
0.8m×0.8m	33ms / 34ms	63ms / 64ms

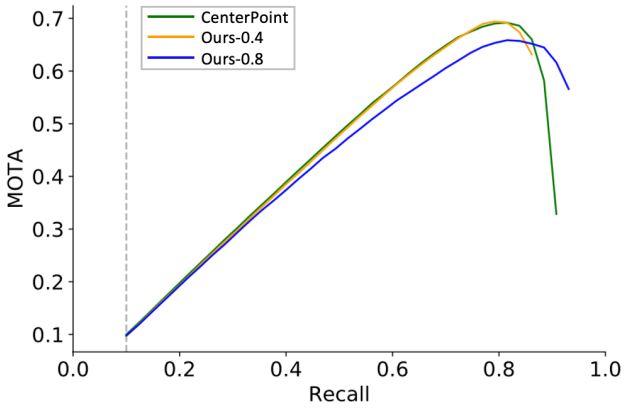
Table 6: Comparison of inference latency of the detection-only model vs. our joint detection and tracking model using different centerness map resolutions and backbones.

Method	PointPillars (v)		VoxelNet (v)		VoxelNet (t)	
	mAP	NDS	mAP	NDS	mAP	NDS
CenterPoint	50.3	60.2	56.4	64.8	58.0	65.5
Ours	**55.5**	**64.9**	**60.1**	**67.6**	**61.3**	**67.6**

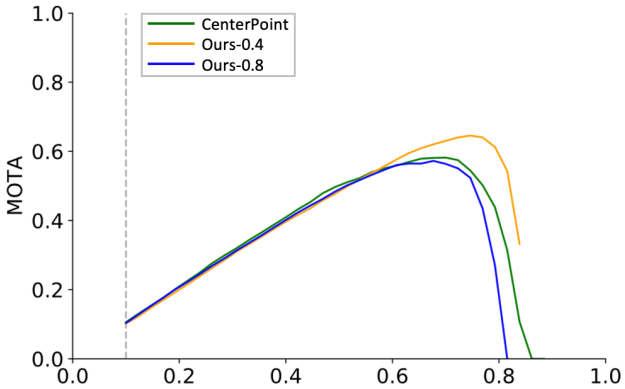
Table 7: Comparison of the 3D object detection results on the validation (v) and test (t) sets of nuScenes.



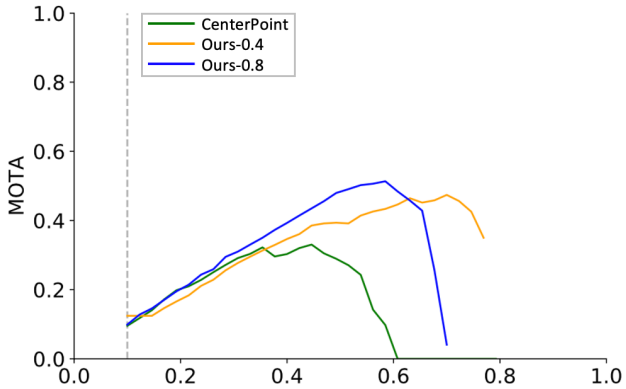
(a) IDS-Recall curve of the car category.



(b) MOTA-Recall curve of the car category.



(c) MOTA-Recall curve of the pedestrian category.



(d) MOTA-Recall curve of the motorcycle category.

Figure 4: Comparisons on IDS and MOTA between SimTrack and CenterPoint under different recall rates. All results are produced by the pillar based backbone. Ours-0.4 (0.8) denote the resolution of centerness map: 0.4m×0.4m (0.8m×0.8m). Ours-0.8 is the default resolution. See Section 4.5 in the paper for more details about the resolution.

Class	MOTA↑	Miss↓	Miss Match↓	FP↓
Vehicle	54.3 / 50.7	34.6 / 38.8	0.20 / 0.19	10.9 / 10.4
Pedestrian	58.3 / 53.9	31.5 / 35.2	0.60 / 0.57	10.5 / 10.3

Table 8: We report the tracking performance using dynamic voxelization on the validation set of Waymo, and the numbers are in the format LEVEL_1 / LEVEL_2.

formance. In Table 8, we provide more results of SimTrack on Waymo. We employ the pillar based backbone and adopt the dynamic voxelization proposed in [36] to replace the hard voxelization as used in all other experiments.

References

[1] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *ICCV*, 2019.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, 2016.

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with Transformers. In *ECCV*, 2020.

[5] Hsu-kuang Chiu, Jie Li, Rares Ambrus, and Jeannette Bohg. Probabilistic 3D multi-modal, multi-object tracking for autonomous driving. In *ICRA*, 2021.

[6] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3D multi-object tracking for autonomous driving. *arXiv:2001.05673*, 2020.

[7] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *ICCV*, 2017.

[8] Rudolph Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1960.

[9] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[10] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. PnPNet: End-to-end per-

ception and prediction with tracking in the loop. In *CVPR*, 2020.

[11] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[13] Zhichao Lu, Vivek Rathod, Ronny Votel, and Jonathan Huang. RetinaTrack: Online single stage joint detection and tracking. In *CVPR*, 2020.

[14] Chenxu Luo, Xiaodong Yang, and Alan Yuille. Self-supervised pillar motion learning for autonomous driving. In *CVPR*, 2021.

[15] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018.

[16] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. TrackFormer: Multi-object tracking with Transformers. *arXiv:2101.02702*, 2021.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[18] Jinlong Peng, Changan Wang, Fangbin Wan, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Chained-Tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. In *ECCV*, 2020.

[19] Charles Qi, Hao Su, Kaichun Mo, and Leonidas Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[21] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Yong Jae Lee, Alexander Schwing, and Jan Kautz. Instance-aware, context-focused, and memory-efficient weakly supervised object detection. In *CVPR*, 2020.

[22] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Alexander Schwing, and Jan Kautz. UFO^2: A unified framework towards omni-supervised object detection. In *ECCV*, 2020.

[23] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. TransTrack: Multiple-object tracking with Transformer. *arXiv:2012.15460*, 2020.

[24] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020.

[25] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *ICCV*, 2019.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[27] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D multi-object tracking: A baseline and new evaluation metrics. In *IROS*, 2020.

[28] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *CVPR*, 2020.

[29] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 2018.

[30] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. Making convolutional networks recurrent for visual sequence learning. In *CVPR*, 2018.

[31] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D object detection and tracking. In *CVPR*, 2021.

[32] Yifu Zhan, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the fairness of detection and re-identification in multiple object tracking. *arXiv:2004.01888*, 2020.

[33] Zhedong Zheng, Xiaodong Yang, Zhiding Yu, Liang Zheng, Yi Yang, and Jan Kautz. Joint discriminative and generative learning for person re-identification. In *CVPR*, 2019.

[34] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *ECCV*, 2020.

[35] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019.

[36] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3D object detection in LiDAR point clouds. In *CoRL*, 2020.

[37] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *CVPR*, 2018.

[38] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3D object detection. *arXiv:1908.09492*, 2019.

[39] Yang Zou, Xiaodong Yang, Zhiding Yu, Vijaya Kumar, and Jan Kautz. Joint disentangling and adaptation for cross-domain person re-identification. In *ECCV*, 2020.