3DMOTFormer: Graph Transformer for Online 3D Multi-Object Tracking

Shuxiao Ding[1,2], Eike Rehder[3], Lukas Schneider[1], Marius Cordts[1], Juergen Gall[2,4]
[1]Mercedes-Benz AG, Sindelfingen, Germany,
[2]University of Bonn, Bonn, Germany,
[3]Robert Bosch GmbH, Stuttgart, Germany
[4]Lamarr Institute for Machine Learning and Artificial Intelligence, Germany
{shuxiao.ding,lukas.schneider,marius.cordts}@mercedes-benz.com,
e.rehder@gmx.de, gall@iai.uni-bonn.de

Abstract

Tracking 3D objects accurately and consistently is crucial for autonomous vehicles, enabling more reliable downstream tasks such as trajectory prediction and motion planning. Based on the substantial progress in object detection in recent years, the tracking-by-detection paradigm has become a popular choice due to its simplicity and efficiency. State-of-the-art 3D multi-object tracking (MOT) approaches typically rely on non-learned model-based algorithms such as Kalman Filter but require many manually tuned parameters. On the other hand, learning-based approaches face the problem of adapting the training to the online setting, leading to inevitable distribution mismatch between training and inference as well as suboptimal performance. In this work, we propose 3DMOTFormer, a learned geometry-based 3D MOT framework building upon the transformer architecture. We use an Edge-Augmented Graph Transformer to reason on the track-detection bipartite graph frame-by-frame and conduct data association via edge classification. To reduce the distribution mismatch between training and inference, we propose a novel online training strategy with an autoregressive and recurrent forward pass as well as sequential batch optimization. Using CenterPoint detections, our approach achieves 71.2% and 68.2% AMOTA on the nuScenes validation and test split, respectively. In addition, a trained 3DMOTFormer model generalizes well across different object detectors. Code is available at: `https://github.com/dsx0511/3DMOTFormer`.

1. Introduction

3D multi-object tracking (MOT) is a fundamental task in many applications such as autonomous driving and mobile robots, aiming at localization, classification and persistent identification of surrounding objects over time. Especially accurate and consistent online tracking is of great importance for downstream tasks such as trajectory prediction, motion planning and robot navigation.

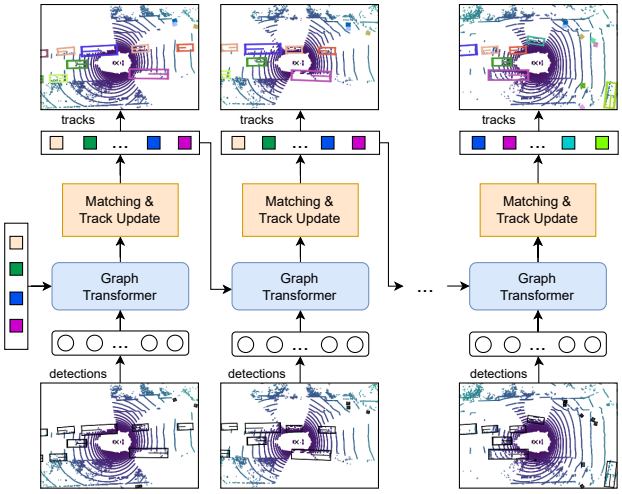

Figure 1. We propose 3DMOTFormer that reasons on the track-detection bipartite graph and estimates the data association using a Graph Transformer. Tracks for the next frame are generated by the matching and track update module autoregressively.

Due to recent advances in object detection performance, the tracking-by-detection paradigm has become a popular choice to accomplish MOT [5, 41, 45, 38, 42, 17]. Most tracking-by-detection approaches utilize detections from every frame generated with an off-the-shelf object detector and focus on associating the detection results across frames.

State-of-the-art tracking-by-detection methods typically use non-learned algorithms, *e.g.* Kalman Filters [38, 17, 3, 15], with a pre-defined motion assumption, *e.g.* constant turn rate and velocity (CTRV) model, followed by a geometric association metric, *e.g.* center distance or 3D

IoU, which requires a lot of handcrafting and heuristics. Learning-based approaches on the other hand aim at reducing heuristics but face the challenge of lifting the training to the online inference setting. Some approaches [8, 39, 14] adopt a teacher-forcing [40] training using ground truth trajectories with annotated instance IDs and/or annotated bounding boxes. However, during online inference, the network has to associate noisy detections to the tracked trajectories containing false associations caused by the network itself. This results in a strong distribution mismatch or overfitting despite applying plenty of data augmentations. Another line of works [19, 43, 14] regards detections as nodes in a spatiotemporal graph and applies Neural Message Passing (NMP) [11]. However, for online MOT, these methods require a graph with a fixed time window to evolve frame-by-frame, while the training is done on the static graph with the same time window but without dynamic evolving. OGR3MOT [43] worked on adapting this graph representation to the online setting but still uses a semi-online training and uses an additional heuristic track update for inference.

In this work, we present a novel transformer-based 3D MOT framework, which we call 3DMOTFormer, that is learnable and relies only on geometric cues, as shown in Figure 1. Our model iteratively reasons on the relationship between existing tracks as well as detections in a new frame and conducts association using edge classification. A greedy matching and a simple track update module generate tracks as input for the next frame, yielding an autoregressive loop. This results in a bipartite graph representation between tracks and detections. In contrast to existing approaches that process a spatiotemporal graph with a fixed time window [43, 14], we directly feed the processed track features into the new frame as initial track features to access temporal information, similar to the hidden states in RNNs. To tackle the different operation modes between training and test time, we propose a novel fully online training strategy which consists of an *autoregressive forward pass* and a *sequential batch backward pass*. Concretely, identical to the inference phase, our model evolves frame-by-frame autoregressively on sampled sequence clips during training, instead of modelling the sequence in a graph as a whole. We accumulate the loss at each frame and optimize the network after the whole training sequence was processed. The forward pass fully simulates the operation mode and the data distribution during the online inference phase, while the optimization method learns to recover from errors and considers the whole sequence. Considering the remarkable achievements of autoregressive models based on transformers in natural language processing [35, 28, 6], we use Edge-Augmented Graph Transformers [12], a variant of transformers that generalizes to sparse graphs and takes edge features into account for attention calculation. Also, structural information in the bipartite graphs between tracks and detections can be effectively captured using cross-attention, which justifies transformer-based models as a suitable choice for our MOT framework.

We evaluate our method on the nuScenes [7] tracking benchmark using CenterPoint detections [42] as input. Our method achieves 71.2% and 68.2% AMOTA on the validation and test split, respectively, yielding state-of-the-art performance among all geometry-based approaches. We show the generalization of 3DMOTFormer where a frozen 3DMOTFormer model still achieves competitive performance when inferring on detections from another detector.

Our main contributions can be summarized as follows:

- We propose 3DMOTFormer, a novel online 3D MOT framework based on Edge Augmented Graph Transformers [12] for learning data association, which reduces the need for handcrafted components compared to previous state-of-the-art.
- 3DMOTFormer is tailored towards an online training strategy for MOT, which fully mimics the setup and hence the data distribution during online inference.
- Our method achieves state-of-the-art performance, in particular 71.2% and 68.2% AMOTA on the nuScenes [7] validation and test split, respectively, using CenterPoint detections [42] as input.
- 3DMOTFormer achieves competitive performance when inferring on detections from another detector. This allows flexible deployment of the same 3DMOTFormer model independent of the object detector.

2. Related Work

In this section, we discuss previous works based on the tracking-by-detection paradigm with a focus on 3D MOT. These works can be divided into model-based and learning-based approaches. Some approaches adopt neural networks as a complementary module in their pipeline but still heavily rely on the pre-defined motion model. Thus, we still categorized them into model-based approaches.

2.1. Model-based multi-object tracking

Using Kalman Filters (KF) with a 2D motion model achieved great success in 2D MOT, *e.g*. SORT [5] with its succeeding variants [41, 10, 1] and ByteTrack [45]. Inspired by this success, AB3DMOT [38] uses a 3D motion model for KF and an association using 3D IoU. Some works improve AB3DMOT using other association metrics, *e.g*. Mahalanobis distance in Chiu *et al*. [17] and 3D Generalized IoU (GIoU) [30] in SimpleTrack [27]. Others investigate the track life management module, *e.g*. the confidence-based track spawn and termination in CBMOT [3] and the

permanent preservation without termination in ImmortalTracker [37]. CenterTrack [46] and CenterPoint [42] replace the filter algorithm with a simpler constant velocity model where they use a temporal object detection model with accurate velocity regression. Many recent 3D object detection works [2, 13, 20, 24] achieve competitive tracking performance using the CenterPoint-based tracker. GNN-PMB [23] proposes a Poisson multi-Bernoulli filter using global nearest neighbor for data association and achieves state-of-the-art performance. Although model-based approaches for MOT so far performed better than learned ones, our learning-based method is able to outperform model-based methods while reducing the need for manual parameter tuning and designing heuristics. Another line of works incorporates additional data or multi-modal sensor fusion, including both geometric and appearance cues. EagerMOT [15] fuses 3D and 2D detections on a greedy basis in a two-stage association. Other works use a CNN-based image feature extractor and conduct appearance-based association complementary to geometric metrics, *e.g*. Chiu *et al*. [16] and CAMO-MOT [36].

2.2. Learning-based multi-object tracking

Learning-based methods usually use a Graph Neural Network (GNN) to address the association task. The first group of works [39, 8] treats object association as a bipartite graph between tracked trajectories and detections. They typically use a temporal encoding of the tracks, *e.g*. LSTM in GNN3DMOT [39] or spatiotemporal Transformer in TransMOT [8]. As a result, an offline teacher-forcing [40] training with ground truth object associations is needed, which leads to overfitting despite data augmentations. Another group of works [19, 29, 14, 43] uses a spatiotemporal graph with a temporal window size where the association is done for every consecutive frame. In 2D MOT, MPNTrack [19] uses a Message Passing Network (MPN) [11] to address the offline tracking as a min-cost network flow problem [44]. TrackMPNN [29] moves MPNTrack [19] towards the online setting by updating the graph dynamically as a rolling window and accumulating losses over the sequence during training. In 3D MOT, OGR3MOT [43] lifts MPNTrack [19] to the online setting by extending predictive track nodes based on KF and a semi-online training method. This training method consists of two stages: the first stage uses ground truth to generate track node data, whereas in the second stage, the track node data is inferred using a trained first-stage model. However, these works need a complex heuristic algorithm to decode multi-frame network outputs into associated trajectories [29, 43] which needs to resolve conflicts between different track hypotheses. In contrast, our method uses a dynamic bipartite graph, which requires a much simpler track update operation to decode the network output into hard association. This enables a fully autoregressive forward pass during training to further approximate the online inference. We use the accumulated loss over multiple frames to train the network, similar to [29], in order to extend the temporal receptive field of the bipartite graph and to optimize the online inference process.

3. Our Approach

An overview of our proposed 3DMOTFormer is shown in Figure 2. Given existing tracks and new detections at time stamp t, we first build track G_T, detection G_D and association graphs G_A. The initial track $h_T^{(0)}$, detection $h_D^{(0)}$, and edge features $h_A^{(0)}$ as well as the graph structure are processed by a transformer-based model consisting of graph self-attention and edge-augmented graph cross-attention [12]. Based on updated edge features $h_A^{(L_d)}$, we compute the affinity between tracks and detections which is further processed by the track update module to autoregressively generate inputs for the next frame. We also estimate velocities to predict track locations. Our training stage fully mimics this online inference schema: we run our model in an autoregressive manner, calculate the affinity and velocity loss for each frame, and finally propagate gradients after the entire training sequence has been processed.

3.1. Graph Representation

We model the multi-object tracking using a graph representation, where a detection or a tracklet is regarded as a node. We employ a sparse graph representation to remove redundant connections and provide more structured data for a better interaction modelling and feature extraction. We build three graphs: (1) a detection graph G_D which enables a message passing between detections to update detection features with a scene embedding, (2) a track graph G_T which models the interactions between existing unassociated tracks from previous frames, and (3) an association graph G_A which reasons on the relationship between tracks and detections that will be potentially associated.

Detection graph The detection graph $G_D = (V_D, E_D)$ with detection nodes V_D and detection edges E_D is built to model the interaction between newly detected objects, where each object represents a node $v_{D,i} \in V_D$. Similar to OGR3MOT [43], the initial feature embedding of a detection node $x_{D,i}$ is a concatenation of box center position, size, yaw, velocity, one-hot encoded class and detection score. All these values are readily available in typical 3D detectors, *e.g*. CenterPoint [42]. Next, $x_{D,i}$ is embedded using a Multi-Layer Perceptron (MLP) into $h_{D,i}^{(0)}$ as the input of the model. The detection graph is truncated using a fixed class-agnostic distance threshold.

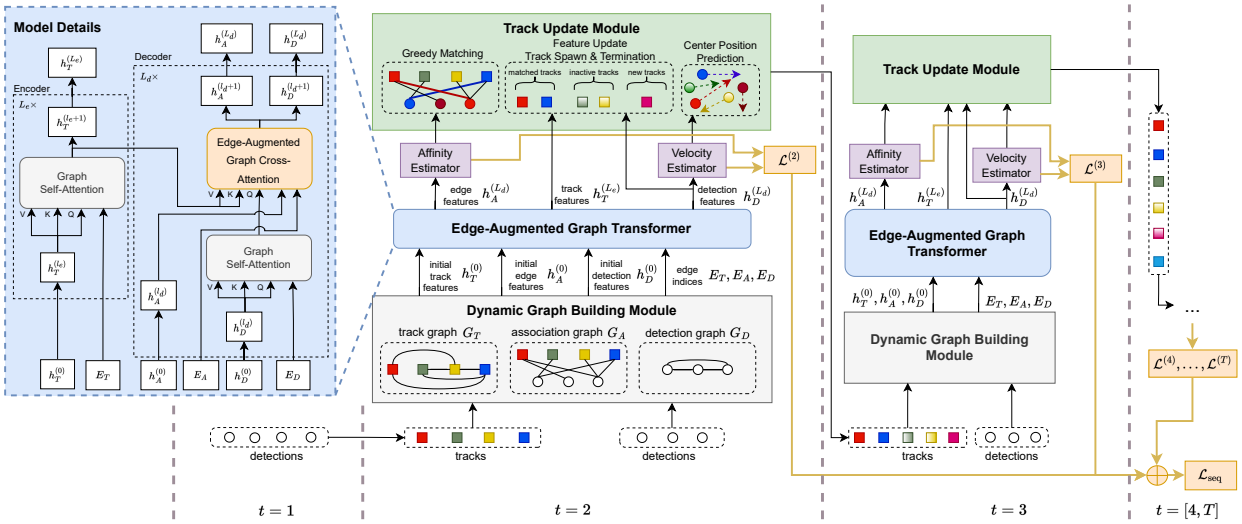

Figure 2. Overview of our 3DMOTFormer framework. Our model processes graph-structured data consisting of tracks and detections. We use an encoder with graph self-attention to encode existing tracks, and a decoder with both graph self-attention and edge-augmented graph cross-attention [12] that processes detection and edge features. The decoder outputs are used to estimate affinity and velocity, which are further used to update tracks. The network runs autoregressively during training and we optimize the network using the losses of all frames.

Track graph The track graph $G_T = (V_T, E_T)$ with track nodes V_T and track edges E_T is similar to the detection graph G_D. As our model runs recurrently, for each track in the memory, we use the processed track feature from the previous frame as the initial feature $h_{T,i}^{(0)}$ in a new frame. This initial feature is similar to the hidden state in an RNN, which enables the model to access the information from the history. The track edges E_T are established with the same truncation mechanism as in the detection graph.

Association graph The association graph $G_A = (V_D, V_T, E_A)$ is a bipartite graph that connects detections V_D and tracks V_T using association edges E_A. We first predict the position of all tracks in the new frame assuming constant velocity, where the velocity is estimated by our network. In contrast to the other two graphs, an association edge is only established between two nodes with the same category and it is truncated using class-specific distance thresholds. The thresholds are calculated based on the dataset statistic of the maximal velocity of a certain class, following [42]. In addition, every association edge $e_{A,ij} \in E_A$ contains a corresponding initial edge feature $h_{A,ij}^{(0)}$. We follow OGR3MOT [43] to use original position difference, size difference, yaw difference, frame difference and the center distance after prediction. Using edge features, we enable a more comprehensive interaction modelling between tracks as well as detections.

3.2. Graph Transformer for MOT

Transformers [35] are widely used as an autoregressive model in natural language processing. Also, they have shown promising performance on soft association, *e.g.* for feature matching [33, 34] and for multi-modal sensor fusion [2]. Our work also runs autoregressively and the soft association with attention-weighted feature aggregation can help to implicitly acquire multiple hypotheses from past frames, which makes transformers a suitable choice for MOT. The following sections present the adaptation of transformers to our graph representation.

Graph Transformer encoder As shown in the top left side of Figure 2, the Graph Transformer encoder generates updated feature encodings of existing tracks by modelling interaction between them using self-attention. For each layer l, the multi-head attention uses three different linear layers to project the track node features $h_{T,i}^{(l)}$ into value $v_{ic}^{(l)}$, key $k_{ic}^{(l)}$ and query $q_{ic}^{(l)}$, where c is an index for attention heads with $c \in [1, C]$. For each head, the attention from j-th to i-th node is calculated using inner product between key and query $\langle q, k \rangle = \frac{q^T k}{\sqrt{d}}$, divided by $\sqrt{d}$ where d denotes the model feature dimension. A normalized attention $\alpha_{ijc}^{(l)}$ is calculated by

$$\alpha_{ijc}^{(l)} = \frac{\exp\left(\langle q_{ic}^{(l)}, k_{jc}^{(l)} \rangle\right)}{\sum_{m \in \mathcal{N}(i)} \exp\left(\langle q_{ic}^{(l)}, k_{mc}^{(l)} \rangle\right)}, \tag{1}$$

where $\mathcal{N}(i)$ denotes the neighbors of node i and it is defined by the graph connectivity (see Section 3.1). Equation (1) corresponds to a sparse version of softmax that normalizes over neighbors of the node i rather than all nodes. The track feature is then updated by an attention-weighted aggregation over the value vectors of all neighboring nodes,

followed by a concatenation of all attention heads:

$$\hat{h}_{T,i}^{(l+1)} = W_{\mathrm{O}}^{(l)} \Big(\Big\|_{c=1}^{C} \big(\sum_{j \in \mathcal{N}(i)} \alpha_{ijc}^{(l)} v_{jc}^{(l)} \big) \Big), \tag{2}$$

where $\|$ is the concatenation operation and $W_{\mathrm{O}}^{(l)}$ denotes learnable weights of a linear layer. This self-attention uses a graph structure from edge indices E to derive the neighborhood $\mathcal{N}(i)$ for each node i, making it different to a standard transformer that considers all other nodes. $\hat{h}_{T,i}^{(l+1)}$ is processed by a Feed Forward Network (FFN) to generate the output of an encoder layer: $h_{T,i}^{(l+1)}$=FFN($\hat{h}_{T,i}^{(l+1)}$). The final track features $h_{T,i}^{(L_e)}$ are calculated by stacking L_e layers.

Edge-Augmented Graph Transformer decoder Our decoder layer consists of a graph self-attention and a graph edge-augmented cross-attention [12]. The self-attention is implemented in the same way as in the Graph Transformer encoder for new detections, which takes detection features $h_{D,i}^{(l)}$ and detection edges E_D as input and produces intermediate detection features $\tilde{h}_{D,i}^{(l)}$. At every decoder layer l, we project detection feature $\tilde{h}_{D,i}^{(l)}$ for each node i into the query, while key and value are converted from the encoder output $h_{T,i}^{(L_e)}$ for all decoder layers. Following [12], besides the inner product of key and query, edge features $h_{A,ij}^{(l)}$ also contribute to the attention calculation, *i.e.*

$$\alpha_{ijc}^{(l)} = \frac{\exp\left(\langle q_{ic}^{(l)}, k_{jc}^{(l)} \rangle + W_{A,ijc}^{(l)} h_{A,ij}^{(l)} \right)}{\sum_{m \in \mathcal{N}(i)} \exp\left(\langle q_{ic}^{(l)}, k_{mc}^{(l)} \rangle + W_{A,imc}^{(l)} h_{A,im}^{(l)} \right)}, \tag{3}$$

where $W_{A,ijc}^{(l)} \in \mathbb{R}^{d \times 1}$ projects the edge feature into a scalar which represents a part of the attention. Same for the encoder, the output detection feature $h_{D,i}^{(l+1)}$ of every node i is calculated using Equation (2) and an FFN. As for the edge features, we concatenate the cross-attentions before the softmax normalization of all heads and then project it back to the latent dimension d using learnable weights $W_O^{(l)} \in \mathbb{R}^{c \times d}$, *i.e.*

$$\hat{h}_{A,ij}^{(l+1)} = W_O^{(l)} \Big(\Big\|_{c} \left(\langle q_{ic}^{(l)}, k_{jc}^{(l)} \rangle + W_{A,ijc}^{(l)} h_{A,ij}^{(l)} \right) \Big). \tag{4}$$

The final output edge feature $h_{A,ij}^{(l+1)}$ of layer l is generated using another FFN: $h_{A,ij}^{(l+1)}$ = FFN($\hat{h}_{A,ij}^{(l+1)}$). We also stack multiple decoder layers, resulting in the final detection features $h_{D,i}^{(L_d)}$ and final association edge features $h_{A,ij}^{(L_d)}$.

3.3. Learning Targets

Affinity estimation Based on the final edge feature $h_{A,ij}^{(L_d)}$, we use an MLP to estimate an affinity score a_{ij} = MLP($h_{A,ij}^{(L_d)}$), which represents the probability that detection i and track j represent the same object. As the detections are estimated from a 3D detector, we run a Hungarian Matching [18] between annotated and detection boxes using 3D Intersection-over-Union (IoU) as matching cost in order to assign a ground truth ID for detection boxes. Unmatched detection boxes are marked as false positive. The classification target of the edge of $e_{A,ij}$ is positive, only if its connected detection i and track j share the same ID.

Velocity estimation Although many state-of-the-art 3D detectors are able to estimate velocities, the estimation can be more accurate when the objects are tracked for a longer time. In our framework, detections extract features from the hidden states of all tracks using cross-attention, thus capturing abundant historical information. For each detection i, we use an MLP to regress its velocity of the box center v_i = MLP($h_{D,i}^{(L)}$). The ground truth velocity is generated for true positive detections using their ground truth annotations. False positive detections are ignored during training. After the track update module, we predict the positions of tracked objects in the next frames using their corresponding velocity v_i. The predicted position is used in the association graph building as described in Section 3.1.

Loss function We use the Focal Loss [22] with $\alpha = 0.5$ and $\gamma = 1.0$ as association loss $\mathcal{L}_{\mathrm{a}}$ and smooth-ℓ_1 loss as the velocity loss $\mathcal{L}_{\mathrm{v}}$. The overall loss is $\mathcal{L} = \mathcal{L}_{\mathrm{a}} + \lambda_{\mathrm{v}} \mathcal{L}_{\mathrm{v}}$ where we set $\lambda_{\mathrm{v}} = 1.0$ as default in our experiments. We evaluate the impact of λ_{v} is in the supplementary material.

3.4. Fully Online Training

Unlike existing learned trackers [39, 43, 14] that use ground truth trajectories including annotated IDs as input, we use those IDs only for calculating the loss. However, an accurate association heavily relies on an observation of the motion in the past few frames. We tackle this problem by generating trajectories autoregressively during training and optimize the network using sequential outputs as a whole.

The affinity score reflects a soft association but not a hard decision as required for the track update. Hence, we use greedy bipartite matching, where we greedily match the detections starting with the highest detection score to the track with the highest affinity score, while a track cannot be matched twice. For a matched detection-track pair, the track feature will be replaced by the matched detection feature. We refer to the supplementary material for a detailed illustration of the track feature update. Based on the matching results, a simple heuristic track life management determines track spawning and termination. To achieve high recall, we initialize all unmatched detections as new tracks. We use a count-based track management where all tracks that are

unmatched for T_d frames are permanently deleted. The unmatched tracks with an age smaller than T_d are temporally inactive but still used in the next frame.

With the track update module in the loop, we accomplish a frame-by-frame autoregressive forward pass during training, identical to online inference. While we generate trajectories using the network itself, it can introduce errors which could significantly affect the training of subsequent frames. To solve this problem, our training strategy aims at optimizing the network using the whole sequence instead of each frame. Concretely, we store the losses for each time stamp $\mathcal{L}^{(t)}$ while processing the training sequence frame-by-frame. When the whole training sequence of length T has been processed by the network, we accumulate the losses of each time stamp to get the sequence loss: $\mathcal{L}_{\text{seq}} = \sum_{t=2}^{T} \mathcal{L}^{(t)}$. We then execute the back-propagation through time (BPTT) [31] to optimize the network using losses from all time stamps. Using this sequential batch optimization method, the network is trained to capture and correct the errors in the previous frames, which subsequently brings a better online performance.

4. Experiments

4.1. Experimental Setup

Dataset NuScenes [7] is a large-scale dataset focusing on perception and prediction for autonomous vehicles that we use for training and testing. It contains 1000 scenes of 20 second length, which are spilt into 700, 150 and 150 scenes as training, validation and test set, respectively. The driving data is collected using multiple sensors, including multi-view cameras, a 32-beam LiDAR, RADARs etc. Despite a higher capture frequency of these sensors, the dataset is annotated at 2Hz.

Metrics For evaluation, we follow the nuScenes tracking benchmark protocol. The primary metrics are the AMOTA and AMOTP that are proposed in [38], where AMOTA is used for ranking. The AMOTA (Average Multi Object Tracking Accuracy) improves the MOTA metric [4] by averaging over the recall-normalized MOTA (MOTAR) over different recall thresholds. AMOTP (Average Multi Object Tracking Precision) reflects the average of position errors over different recall thresholds. In addition, nuScenes uses a variety of secondary metrics, *e.g*. MOTA, MOTP, IDS and FRAG from CLEAR MOT [4] and MT/ML from MOT Challenges [26, 9]. These metrics are computed after applying an independent threshold for each class where the highest MOTA is reached.

Detector As most state-of-the-art methods for MOT report their results on nuScenes using the CenterPoint [42] detector, we use the same detections for a fair comparison. In addition, we use the detectors from MEGVII [47] and BEVFusion [24] to validate the generalization of our method.

Baselines As 3DMOTFormer only uses 3D detections to accomplish data association, we compare our method with state-of-the-art tracking-by-detection approaches that rely on 3D geometric cues from CenterPoint detections. Hence, we use the learning-based OGR3MOT [43] and PolarMOT [14] as primary baselines and additionally five model-based methods: CenterPoint [42], CBMOT [3], SimpleTrack [27], ImmortalTracker [37], and GNN-PMB [23].

Implementation details Following SimpleTrack [27], we first use Non-Maximum Suppression (NMS) with a 3D IoU threshold of 0.1 to filter duplicates. During training, we sample mini-sequences of length T=6 frames from training scenes as our training samples, which corresponds to 2.5 s at a frequency of 2Hz. A track is deleted if it is unmatched for T_d=3 frames. All models are trained using AdamW [25] for 12 epochs with a batch size of 8. We use a learning rate of 0.001 and a weight decay of 0.01.

4.2. Benchmark Results

Test set Table 1 shows results on the nuScenes test set using CenterPoint detections. We first compare 3DMOTFormer with the baselines that are listed in the upper part of Table 1. Our approach outperforms other learning-based approaches significantly, yielding 2.6%P and 1.8%P AMOTA improvements over OGR3MOT and PolarMOT, respectively. Furthermore, 3DMOTFormer surpasses the highest ranking model-based approach GNN-PMB [23] by 0.4%P in AMOTA. In addition, we achieve a notably better AMOTP than the baselines using the same detection boxes, which verifies that our approach can associate more precisely. Compared to the averaged metrics AMOTA and AMOTP, our best-achieved MOTA at single recall threshold as well as corresponding secondary metrics are relatively lower. This on the other hand shows a more balanced performance over different recall thresholds of our approach. Second, we compare to approaches that use additional information besides geometric cues from 3D object detections, *c.f*. the lower half of Table 1. Our approach still outperforms Chiu *et al*. [16] and EagerMOT [15] with 2D data and achieves on par performance to NEBP [21]. Only the concurrent work ShaSTA [32] that heavily relies on LiDAR backbone features performs better than our approach.

Validation set We further compare to the baselines on the validation set in Table 2, where 3DMOTFormer again achieves best AMOTA and AMOTP among all approaches. We observe that the AMOTA of the learning-based baselines (OGR3MOT and PolarMOT) differs more between

Method	Additional Cues	AMOTA↑	AMOTP↓	MOTA↑	MT↑	ML↓	TP↑	FP↓	FN↓	IDS↓	FRAG↓
OGR3MOT [43]	–	0.656	0.620	0.554	5278	2094	95264	17877	24013	288	371
PolarMOT-offline† [14]	–	0.664	0.566	0.561	**5701**	1686	**97909**	17856	**21414**	**242**	**332**
CenterPoint [42]	–	0.638	0.555	0.537	5584	1681	95877	18612	22928	760	529
CBMOT [3]	–	0.649	0.592	0.545	5319	1966	94916	**16469**	24092	557	450
SimpleTrack‡ [27]	–	0.668	0.550	0.566	5476	1780	95539	17514	23451	575	591
ImmortalTracker [37]	–	0.677	0.599	**0.572**	5565	1669	97584	18012	21661	320	477
GNN-PMB [23]	–	0.678	0.560	0.563	5698	**1622**	97274	17071	21521	770	431
Chiu et al. [16]	2D appearance	0.655	0.617	0.555	5494	1557	95199	18061	23323	1043	717
EagerMOT [15]	2D geometry	0.677	0.550	0.568	5303	1842	93484	17705	24925	1156	601
NEBP [21]	3D appearance	0.683	0.624	0.584	5428	1993	97367	16773	21971	227	299
ShaSTA [32]	3D appearance	0.696	0.540	0.578	5596	1813	97799	16746	21293	473	356
3DMOTFormer (ours)	–	**0.682**	**0.496**	0.556	5466	1896	95790	18322	23337	438	529

Table 1. Results on nuScenes test set using CenterPoint detections. AMOTA and AMOTP are the primary metrics on the benchmark. †denotes offline methods, ‡denotes using 10Hz data. We mark best performance in the comparison with baselines in bold text and underline where an even better performance is achieved by methods with additional information besides geometric cues from 3D detections.

Method	AMOTA↑	AMOTP↓	MOTA↑	IDS↓	FRAG↓
OGR3MOT [43]	0.693 (+0.037)	0.627	0.602	262	332
PolarMOT-offline† [14]	0.711 (+0.047)	–	–	**213**	332
PolarMOT-online [14]	0.673	–	–	439	**285**
CenterPoint [42]	0.665 (+0.027)	0.567	0.562	562	424
CBMOT [3]	0.675 (+0.026)	–	–	494	–
SimpleTrack‡ [27]	0.696 (+0.028)	0.547	0.602	–	403
ImmortalTracker [37]	0.702 (+0.025)	–	0.601	–	385
GNN-PMB [23]	0.707 (+0.029)	0.560	–	650	345
3DMOTFormer (ours)	**0.712** (+0.030)	**0.515**	**0.607**	341	436

Table 2. Results on nuScenes validation set using CenterPoint detections. † denotes offline methods, ‡ denotes using 10Hz data. Changes of AMOTA to the test set are shown in the brackets.

Source	Target	AMOTA↑	AMOTP↓	MOTA↑	IDS↓	FRAG↓
CenterPoint	CenterPoint	0.712	0.515	0.607	341	436
BEVFusion	BEVFusion	0.749	0.550	0.652	447	443
MEGVII	MEGVII	0.641	0.639	0.535	328	497
BEVFusion	CenterPoint	0.699	0.524	0.595	421	434
MEGVII	CenterPoint	0.697	0.544	0.591	418	445
CenterPoint	BEVFusion	0.747	0.553	0.652	526	475
MEGVII	BEVFusion	0.744	0.526	0.640	445	479
CenterPoint	MEGVII	0.632	0.643	0.529	409	518
BEVFusion	MEGVII	0.626	0.663	0.521	415	494

Table 3. Detector domain generalization experiment. Performance is reported using models that are trained on detections from source detector and tested on detections from target detector.

validation and test split compared to model-based approaches. This shows some overfitting of learned models, *e.g.* due to hyperparameter tuning using validation performance. In comparison, the performance difference for 3DMOTFormer is smaller and close to the one from model-based approaches. We attribute this observation to our online training strategy that effectively reduces the distribution mismatch between training and inference and hence leads to better generalization, which will be discussed next.

Generalization across detectors In contrast to typical learning-based approaches, model-based approaches [42, 17, 3, 23] generalize well across different 3D detectors, as the pre-defined motion models are derived from real world physics and thus independent of the detector. However, we show that a trained and frozen model of 3DMOTFormer also generalizes well to different detectors at test time than trained with. The upper part of Table 3 shows results, where we use the three different detectors CenterPoint [42], BEVFusion [24], and MEGVII [47] for both training and testing, where 3DMOTFormer consistently produces high performance for all detectors. The bottom part shows the results where we train on one source detector and run inference using this trained model on detections from another detector. For example, when using CenterPoint as target detector, the AMOTA of models trained on BEVFusion and MEGVII detections are only 1.3%P and 1.5%P worse than the standard setting. The same phenomenon can be observed when other detector combinations are used. Considering that our model can accurately estimate velocities in order to predict the track positions, we account this generalization capability to a learned underlying detector-agnostic motion model using sequential batch optimization.

Runtime Our approach runs at 54.7 Hz on an Nividia GeForce 2080Ti GPU. It is therefore well suited for real-time applications such as autonomous vehicles.

4.3. Ablation Studies

We conduct extensive ablation studies of 3DMOTFormer to highlight how the proposed components work. All exper-

T	AMOTA↑	AMOTP↓	MOTA↑	IDS↓	FRAG↓
2	0.6741	0.5738	0.5733	898	520
3	0.7105	0.5294	0.6095	384	428
4	0.7096	**0.5135**	0.6065	380	**418**
5	0.7111	0.5317	**0.6115**	373	430
6	0.7121	0.5149	0.6071	**341**	436
7	0.7120	**0.5135**	0.6105	343	432
8	**0.7124**	0.5203	0.6086	363	434

Table 4. Ablation study on the length of training sample T.

Variant	AMOTA↑	AMOTP↓	MOTA↑	IDS↓	FRAG↓
Hungarian Matching	0.7006	0.5245	0.5924	786	492
GT identity guided	0.7073	0.5200	0.6065	360	**405**
w/o hidden state	0.7013	0.5326	0.6015	371	432
3DMOTFormer	**0.7121**	**0.5149**	**0.6071**	**341**	436

Table 5. Ablation study of other training variants.

iments are evaluated on the NuScenes validation set.

Training sample length We first evaluate the training sample length T which is an important factor for lifting our training to an online setting. By setting $T = 2$, the training degenerates, the autoregressive loop is dropped, and the network becomes an affinity estimator between detections of two frames. As shown in Table 4, the performance of $T = 2$ is surpassed by larger values with a large margin, *e.g.* 3.8%P AMOTA for the default setup with $T = 6$. Starting at $T = 3$, the training follows the online procedure described in Section 3.4 and we observe smaller deltas with increasing T. The results verify the necessity of our proposed training strategy as learning from the whole sequence is critical for the online inference. The overall performance peaks and converges at $T \in [6, 8]$ so we use $T = 6$ for all other experiments for training efficiency.

Other training-related factors We evaluate three different variants that are highly related to our training method: (1) *Hungarian Matching:* we replace the greedy matching by Hungarian Matching; (2) *GT identity guided:* we directly use the annotated association instead of greedy matching to accomplish the track update; (3) *w/o hidden state:* we reuse the box embedding as initial track feature instead of using the hidden state from the previous frame $h^{(L)}_{T,i,t-1}$. As shown in Table 5, using Hungarian Matching decreases AMOTA by 1.15%P and introduces a considerable amount of ID switches. This observation shows that high quality detections should have higher priority in data association, as realized by our greedy strategy that starts with the highest scoring detections. Guided by GT identity, AMOTA slightly decreases to 0.7073 due to a higher data distribution mismatch between training and inference. In contrast, our autoregressive forward pass avoids GT information leaking into training data, thus reducing the distribution mismatch. Without passing track features as hidden state, AMOTA decreases by about 1%P and the AMOTP is also significantly worse. This indicates the importance of the hidden state in capturing motion dynamics and the association history.

Variant	AMOTA↑	AMOTP↓	MOTA↑	IDS↓	FRAG↓
zero feature	0.6469	0.6245	0.5497	3715	871
w/o pred.	0.7055	0.5315	0.5994	411	445
w/o time diff.	0.7113	0.5249	0.6108	350	446
diff. affinity	0.6509	0.6185	0.5587	3547	838
concat affinity	0.6494	0.6170	0.5541	3308	1558
cosine affinity	0.6346	0.6345	0.5417	4489	922
3DMOTFormer	**0.7121**	**0.5149**	**0.6071**	**341**	**436**

Table 6. Ablation study on different options of edge features.

	Max dist.	AMOTA↑	AMOTP↓	MOTA↑	Recall↑	IDS↓	FRAG↓
with prediction	0.5×	0.6952	0.5224	0.5948	0.6952	666	**313**
	1.0×	**0.7121**	**0.5149**	**0.6071**	0.7387	**341**	436
	1.5×	0.7058	0.5372	0.6026	0.7427	416	517
	2.0×	0.7061	0.5384	0.6019	**0.7509**	427	539
w/o prediction	0.5×	0.5822	0.6367	0.4990	0.6423	2368	942
	1.0×	0.6862	0.5505	0.5850	0.7229	1138	635
	1.5×	0.7034	0.5386	0.5975	0.7315	640	500
	2.0×	0.7068	0.5362	0.5999	0.7351	447	509
Fully connected		0.7015	0.5316	0.5937	0.7474	476	493

Table 7. Ablation study on dynamic association graph building.

Edge features Next, we show the effectivity of incorporating edge features into our architecture using the Edge-Augmented Graph Transformer [12]. In the upper part of Table 6, we analyze the impact of different edge feature embeddings $h^{(0)}_A$: (1) *zero:* edge features are set to zero; (2) *w/o prediction:* with zeroed value for the center distance after prediction; (3) *w/o frame difference:* with zeroed value for the frame difference. Variant *zero* causes a very significant performance drop, which confirms the importance of the edge features. Variant *w/o prediction* leads to a decrease of AMOTA by 0.66%P, while *w/o frame difference* has no significant impact on AMOTA. This observation shows the ability of our model to accurately estimate velocities and to capture the underlying motion model for learning-based data association. The bottom part shows another variant, where we replace the Edge-Augmented Graph Transformer with a normal Graph Transformer decoder and conduct edge classification using the features of two nodes connected by an edge. We used the difference, concatenation and the cosine affinity of two node features to estimate the association score. All three models show a strong performance degradation which again verifies our design choices.

Dynamic association graph building Besides computational efficiency, the sparsity of the association graph also provides useful structure information for feature interaction and reduces the amount of negative association edges as well as the class imbalance, as can be seen in Table 7 by comparing our default choice in the second row and a variant with a fully-connected graph in the last row. In the first part of Table 7, we vary the class-specific distance threshold for every node in the graph by applying a multiplicative factor. With $0.5\times$ distance threshold, many potential connections are missing and this further leads to 1.69%P AMOTA decrease. Higher thresholds ($1.5\times$ and $2.0\times$) increase the recall but on the other hand introduce class imbalance, resulting in about 0.6%P AMOTA decrease. In the second part, we use original boxes of tracks without predicting them while constructing the graph. In this case, AMOTA performance increases with the distance threshold. Without prediction, tracks have a higher distance to reach the correct association and hence higher thresholds preserve the recall. However, this setting with prediction performs better and results in a more sparse graph structure. This again verifies the ability of our approach in estimating velocity accurately to achieve an effective graph building.

5. Conclusion

In this paper, we presented a novel 3D online multi-object tracking (MOT) framework using Graph Transformer which only relies on geometric cues, termed 3DMOT-Former. We formulate the association using a bipartite graph representation and exploit the Edge-Augmented Graph Transformer to reason on the graph structure and conduct data association. Our network runs recurrently and autoregressively during training and we use a sequential batch optimization to train the network, yielding a fully online training which is closely coupled with the online inference process of MOT. Our approach achieves state-of-the-art performance on the nuScenes dataset, outperforms all other geometric-based data association approaches, and shows good generalization across different detectors.

Acknowledgement

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Climate Action within the project "KI Delta Learning" (Förderkennzeichen 19A19013A). The authors would like to thank the consortium for the successful cooperation. Juergen Gall has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) GA 1927/5-2 (FOR 2535 Anticipating Human Behavior) and the ERC Consolidator Grant FORHUE (101044724).

References

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *ArXiv*, abs/2206.14651, 2022.

[2] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1080–1089, 2022.

[3] Nuri Benbarka, Jona Schröder, and Andreas Zell. Score refinement for confidence-based 3d multi-object tracking. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8083–8090, 2021.

[4] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[5] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Tozeto Ramos, and Ben Upcroft. Simple online and realtime tracking. *IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] 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. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[8] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4859–4869, 2023.

[9] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Qinfeng Shi, Daniel Cremers, Ian D. Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *ArXiv*, abs/2003.09003, 2020.

[10] Yunhao Du, Yang Song, Bo Yang, and Yanyun Zhao. Strongsort: Make deepsort great again. *ArXiv*, abs/2202.13514, 2022.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[12] Md Shamim Hussain, Mohammed J. Zaki, and D. Subramanian. Edge-augmented graph transformers: Global self-attention is enough for graphs. *ArXiv*, abs/2108.03348, 2021.

[13] Yang Jiao, Zequn Jie, Shaoxiang Chen, Jingjing Chen, Lin Ma, and Yu-Gang Jiang. Msmdfusion: A gated multi-scale lidar-camera fusion framework with multi-depth seeds for 3d object detection. *ArXiv*, abs/2209.03102, 2022.

[14] Aleksandr Kim, Guillem Brasó, Aljosa Osep, and Laura Leal-Taixé. Polarmot: How far can geometric relations take us in 3d multi-object tracking? In *European Conference on Computer Vision*, 2022.

[15] Aleksandr Kim, Aljosa Osep, and Laura Leal-Taixé. Eagermot: 3d multi-object tracking via sensor fusion. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 11315–11321, 2021.

[16] Hsu kuang Chiu, Jie Li, Rares Ambrus, and Jeannette Bohg. Probabilistic 3d multi-modal, multi-object tracking for autonomous driving. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 14227–14233, 2021.

[17] Hsu kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *ArXiv*, abs/2001.05673, 2020.

[18] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52, 1955.

[19] Guillem Brasóand Laura Leal-Taixé. Learning a neural solver for multiple object tracking. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6246–6256, 2019.

[20] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. *ArXiv*, abs/2206.00630, 2022.

[21] Mingchao Liang and Florian Meyer. Neural enhanced belief propagation for data association in multiobject tracking. *25th International Conference on Information Fusion (FUSION)*, pages 1–7, 2022.

[22] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:318–327, 2017.

[23] Jianan Liu, Liping Bai, Yuxuan Xia, Tao Huang, and Bing Zhu. Gnn-pmb: A simple but effective online 3d multi-object tracker without bells and whistles. *ArXiv*, abs/2206.10255, 2022.

[24] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *ArXiv*, abs/2205.13542, 2022.

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

[26] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *ArXiv*, abs/1603.00831, 2016.

[27] Ziqi Pang, Zhichao Li, and Naiyan Wang. Simpletrack: Understanding and rethinking 3d multi-object tracking. *ArXiv*, abs/2111.09621, 2021.

[28] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[29] Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Reza Ramezani, and Mohan Manubhai Trivedi. Trackmpnn: A message passing graph neural architecture for multi-object tracking. *ArXiv*, abs/2101.04206, 2021.

[30] Seyed Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[31] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[32] Tara Sadjadpour, Jie Li, Rares Ambrus, and Jeannette Bohg. Shasta: Modeling shape and spatio-temporal affinities for 3d multi-object tracking. *ArXiv*, abs/2211.03919, 2022.

[33] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4937–4946, 2020.

[34] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8918–8927, 2021.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] Li Wang, Xinyu Newman Zhang, Wenyuan Qin, Xiaoyu Li, Lei Yang, Zhiwei Li, Lei Zhu, Hong Wang, Jun Li, and Hua Liu. Camo-mot: Combined appearance-motion optimization for 3d multi-object tracking with camera-lidar fusion. *ArXiv*, abs/2209.02540, 2022.

[37] Qitai Wang, Yuntao Chen, Ziqi Pang, Naiyan Wang, and Zhaoxiang Zhang. Immortal tracker: Tracklet never dies. *ArXiv*, abs/2111.13672, 2021.

[38] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366, 2020.

[39] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris Kitani. Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6498–6507, 2020.

[40] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[41] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017.

[42] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11779–11788, 2021.

[43] Jan-Nico Zaech, Alexander Liniger, Dengxin Dai, Martin Danelljan, and Luc Van Gool. Learnable online graph representations for 3d multi-object tracking. *IEEE Robotics and Automation Letters*, 7(2):5103–5110, 2022.

[44] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[45] Yifu Zhang, Pei Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*, 2021.

[46] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *ArXiv*, abs/2004.01177, 2020.

[47] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *ArXiv*, abs/1908.09492, 2019.