Multi-Camera Multiple 3D Object Tracking on the Move for Autonomous Vehicles

Pha Nguyen[1], Kha Gia Quach[2], Chi Nhan Duong[2], Ngan Le[1], Xuan-Bac Nguyen[1], Khoa Luu[1]
[1] CVIU Lab, University of Arkansas, USA [2] Concordia University, CANADA
[1]{panguyen, thile, xnguyen, khoaluu}@uark.edu [2]{dcnhan, kquach}@ieee.org

Abstract

The development of autonomous vehicles provides an opportunity to have a complete set of camera sensors capturing the environment around the car. Thus, it is important for object detection and tracking to address new challenges, such as achieving consistent results across views of cameras. To address these challenges, this work presents a new Global Association Graph Model with Link Prediction approach to predict existing tracklets location and link detections with tracklets via cross-attention motion modeling and appearance re-identification. This approach aims at solving issues caused by inconsistent 3D object detection. Moreover, our model exploits to improve the detection accuracy of a standard 3D object detector in the nuScenes detection challenge. The experimental results on the nuScenes dataset demonstrate the benefits of the proposed method to produce SOTA performance on the existing vision-based tracking dataset.

1. Introduction

Object detection and tracking have become one of the most important tasks in autonomous vehicles (AV). Recent development of deep learning methods has dramatically boosted the performance of object understanding and tracking in autonomous driving applications thanks to the availability of public datasets. Far apart from prior video tracking datasets collected via single or stereo cameras, e.g., KITTI [13], recent public datasets and their defined tracking problems have become more realistic with multiple cameras in autonomous vehicles. They usually have a full set of camera sensors that aim to create a 360° surround view and provide more redundancy as backup, i.e. more overlapping field-of-views. There are some popular large-scale tracking datasets with multiple sensor setup, such as nuScenes [2], Waymo [32], Lyft [30], or Argoverse [6]. They have a lot more data than KITTI ranging from multiple surrounding cameras, LiDAR, radars and GPS.

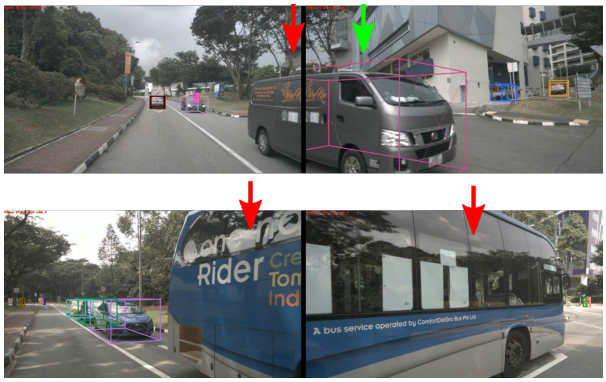

Figure 1. First row: the object detector and tracking method DEFT [5] fails to detect partial objects in one camera but can detect in another camera, Second row: The detector fails to detect objects in both cameras. Green arrow indicates true positive detection sample, red arrows indicate false negative detection samples.

Having enormous data as in recent public datasets helps to improve deep learning based 3D object detection. However, it also poses more challenging problems in practice, such as maintaining high accuracy and latency performance in variety points of views and environments. In addition, Multiple Object Tracking (MOT) is usually employed together with 3D object detection to track objects and maintain stability of prediction across video frames. In order to handle multiple views, a common approach to Multi-Camera Multiple Object Tracking (MC-MOT) [3, 7] is to firstly apply an MOT approach on each camera independently, i.e. single camera tracking (SCT), then link local tracklets across cameras together via global matching steps based on Re-ID features. However, this approach creates more errors, i.e. fragmented local tracklets, and more computation since the data association and the matching steps will perform multiple times both locally and globally. Therefore, using SCT multiple times is not the optimal option. In addition, it is unable to handle scenarios when the detector fails to detect objects from one of the cameras as shown in Fig. 1.

Therefore, this work proposes to formulate MC-MOT problem as a *global association graph* in a 360° view using an object detection as the inputs instead of SCT trajectories. Our proposed MC-MOT approach not only models object motion but also the appearance of each tracked object. We encode both location and appearance features in the node embeddings of the proposed graph where the nodes corresponding to each tracked object are updated and added to the graph over time. In addition, we adopt the new self-attention and cross-attention layers to decode motion and location, then propagate them across camera systems via 3D-to-2D transformation.

Contributions of this Work. The main contributions of this work can be summarized as follows. A new MC-MOT framework is firstly introduced where a global graph is constructed with *nodes* containing both appearance and motion features of the tracked objects and *the weighted edges* between tracked objects or nodes. The edge weights are computed based on the similarity in appearance and location between two tracked objects or nodes. Secondly, we present a new *Auto-regressive Graph Transformer network* including a self-attention layer to transform appearance features and cross-attention to predict the motion features of objects. This network can help to obtain a more robust node embedding to maintain accurate tracking when objects are on side views of cameras. Then, we further post-process the prediction results with motion propagation and node merging modules. Finally, the proposed framework will be evaluated with a comprehensive evaluation criterion to demonstrate its robustness compared against previous MC-MOT frameworks. The proposed method even helps to improve the detection accuracy of a standard 3D object detector on the nuScenes benchmark.

2. Related Work

MOT problem on AVs has recently received a lot of attention from the research community. There is an increasing amount of research work targeting trajectory estimation on moving sensors [8, 36] or combining appearance information to determine object IDs [14, 42, 43].

Tracking using Motion Model Weng et al. [36] propose a simple yet effective baseline that utilizes classic state estimator Kalman Filter for 3D bounding boxes. They can be obtained not only from a LiDAR point cloud object detector [20, 21, 28, 44, 45] but also from an image-based object detector [14, 26, 29, 43]. Chiu et al. [8] improves the Kalman Filter tracking system by measuring the Mahalanobis distance between the predicted states and observations. This method is promisingly reliable in filtering outliers and handling both partially and fully occluded objects.

Tracking using Appearance Model Zhou et al.'s approaches [42, 43] are widely used in single camera tracking problems. By treating objects as points, these approaches simplify the tracking procedure that is usually a combination of many expensive steps from detection to assigning object ID. Simonelli et al. [29] introduce a novel disentangling transformation for detection loss and a self-supervised term for bounding boxes confidence score. Hu et al. [14] try to estimate robust 3D box information from 2D images then adopt 3D box-reordering and LSTM as a motion module to link objects across frames.

Tracking using Hybrid Approaches Chaabane et al. [5] train the object detection and the object association task simultaneously by adding a feature extractor and a matching head after object detector. Besides, an LSTM is used as a motion prediction module as an alternative to Kalman Filter. Similarly, Yin et al. [39] follow the same process, but perform feature extraction on point cloud maps.

Tracking using Modern Approaches Graph Neural Network, Self-Attention, and Transformer [34] introduce a new learning-from-context paradigm. It recently has attracted considerable attention from the research community because of its promising performance in a wide range from Natural Language Processing [10, 17, 19, 24] to Computer Vision [4, 11, 25, 33, 35, 47] tasks. Currently, there are none of these methods applied in MC-MOT on autonomous vehicles but it is worthy to name a few SCT-MOT approaches [9, 12, 18, 31, 37, 38, 46]. Weng et al. [37] propose the first feature interaction method that leverages Graph Neural Network to individually adapt an object feature to another object features. Meinhardt et al. [18] propose a new tracking-by-attention paradigm besides existing tracking-by-regression, tracking-by-detection and tracking-by-segmentation to deal with occlusions and reason out tracker's spatio-temporal correspondences. Sun et al. [46] utilize Query-Key mechanism to perform joint-detection-and-tracking, disentangle complex components in previous tracking systems.

3. Our Proposed Method

In this section, we first overview our proposed 3D object tracking pipeline where we construct and maintain a Global Graph with the Graph Transformer Networks in Subsection 3.1. Then, Subsection 3.2 will detail the structure of Graph Transformer Networks and how it is used to model appearance and motion of tracked objects. Finally, Subsection 3.4 describes how we train the Graph Transformer Networks.

3.1. MC-MOT via Global Graph Constructing

Given C cameras, denoted by the set $\mathcal{C} = \{c_1, \ldots, c_C\}$, they are used to perceive surrounding environment of a vehicle. In MC-MOT, we assume each camera attached with an off-the-shelf 3D object detector to provide initial location of objects in real-world coordinates. In this work, KM3D [16] is used to provide 3D object location and features but it can be replaced by any other 3D object detectors.

In the previous MC-MOT approaches [3] [7], [40] [22], the methods depend on tracking results of an MOT algorithm on each camera independently. There is no mechanism to model the relationship between cameras while they have a strong relations. Instead, our proposed MC-MOT take detection results directly from the detectors and match with current tracked objects using an auto-regressive approach by taking the cameras relation into consideration. In our approach, a single graph is constructed and maintained across time by graph transformer networks (detailed in Sec. 3.2).

At time step t, our MC-MOT framework receives detection outcomes $\mathcal{O}_c^{(t)} = \{\mathbf{o}_{i,c}^{(t)}\}$ generated by a 3D object detector from all synchronized camera inputs. The detected i-th object $\mathbf{o}_{i,c}^{(t)}$ contains its location in 3D $\mathbf{l}_{i,c}^{(t)}$ and its features $\mathbf{f}_{i,c}^{(t)}$. Then, our MC-MOT framework will update and maintain a set of tracked objects, called tracklets $\mathcal{T}_c^{(t)} = \{\mathbf{tr}_{k,c}^{(t)}\}$, based on detected objects at time step t and previous tracklets at time step $t-1$. Each $\mathbf{tr}_{k,c}^{(t)}$ is a vector with 3D location and features of the corresponding tracked object. This set of tracklets are represented by a global graph $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$, where the vertex set $\mathcal{V}^{(t)}$ contains *all the tracklets* $\mathcal{T}_c^{(t)}$ tracked up to time t and the edge set $\mathcal{E}^{(t)}$ contains *geometry distance* between two tracklets. In this way, $\mathcal{G}^{(t)}$ can be obtained using graph transformer networks from a joint set of $N_{\mathcal{T}}$ nodes of the previous graph $\mathcal{G}^{(t-1)}$ and $N_{\mathcal{O}}$ new nodes formed by current detections $\mathcal{O}_c^{(t)}$s. The changes in the global graph from frame-to-frame are likely adding new nodes as new objects are detected or removing old nodes as tracklets are out of view. This step is done by graph link prediction using a Softmax classifier similar to [23]. Next, we will discuss how the transformer decoder can be employed to update the embedding features for each node with self-attention layer and how to predict tracked objects' motion via cross-attention layer.

3.2. Auto-Regressive Graph Transformer Networks

In this section, we introduce Graph Transformer Networks (GTN) to transform and update node embeddings by attending to other nodes for robust appearance and motion modeling. First, the building blocks of this GTN, i.e. graph self-attention layer and graph cross-attention layer, are presented in Sub-sec. 3.2.1 and 3.2.2, respectively. Then, we perform motion propagation and node merging operators that include the removing and the adding nodes in the graph via link prediction in Sub-sec. 3.2.3 and 3.2.4, respectively.

3.2.1 Graph Self-Attention Layer for Appearance Modeling

Each node $k \in \mathcal{V}^{(t)}$ in the graph $\mathcal{G}^{(t)}$ contains the object's 3D location $\mathbf{l}_{k,c}^{(t)}$ and its feature embedding $\mathbf{f}_{k,c}^{(t)}$, i.e. Re-ID features. The Re-ID features are provided by KM3D [16] as its outputs together with 3D box predictions. To consider the effects of cameras on appearance features, the self-attention layer takes the input node features as the concatenation of embedding features with camera and location encoding as $\mathbf{h}_k^l = \{\mathbf{f}_{k,c}^{(t)} | \mathbf{c} | \mathbf{l}_{k,c}^{(t)}\} \in \mathbb{R}^{D_E}$, where $l = 0$ only applied for the input of the first layer, $\mathbf{f}_{k,c}^{(t)} \in \mathbb{R}^{D_F}$, $\mathbf{c} \in \mathbb{R}^{D_C}$ and $\mathbf{l}_{k,c}^{(t)} \in \mathbb{R}^3$. We use pre-computed camera and location encoding to concat with the node features before the first layer, similar to how positional encodings are added in the original Transformer [34]. Then, the self-attention layer provides the output embeddings as $\mathbf{h}_k^{l+1}$ for layer l. This output can be used as the input for the next layer if there is more than one self-attention layer.

In order to further improve pairwise attention scores as in [34], we incorporate pairwise edge features by multiplying them together. In summary, the output of the self-attention layer is computed as follows,

$$\mathbf{h'}_k^{l+1} = \mathbf{O}_h^l \Big\|_{i=1}^{H} \left(\sum_{j \in \mathcal{V}^{(t)}} \mathbf{w}_{kj}^{i,l} \mathbf{V}^{i,l} \mathbf{h}_k^l \right) \tag{1}$$

$$\mathbf{e'}_{kj}^{l+1} = \mathbf{O}_e^l \Big\|_{i=1}^{H} \left(\sum_{j \in \mathcal{V}^{(t)}} \mathbf{w'}_{kj}^{i,l} \right) \tag{2}$$

$$\mathbf{w}_{kj}^{i,l} = \text{softmax}_j(\mathbf{w'}_{kj}^{i,l}) \tag{3}$$

$$\mathbf{w'}_{kj}^{i,l} = \left(\frac{\mathbf{Q}^{i,l} \mathbf{h}_k^l \cdot \mathbf{K}^{i,l} \mathbf{h}_j^l}{\sqrt{D_h}} \right) \cdot \mathbf{E}^{i,l} \mathbf{e}_{kj}^l \tag{4}$$

where $\mathbf{w}_{kj}^{i,l}$ are the attention coefficients for the i-th attention head, $\|$ is the feature vector concatenation operation, $\mathbf{Q}^{i,l}, \mathbf{K}^{i,l}, \mathbf{V}^{i,l}, \mathbf{E}^{i,l} \in \mathbb{R}^{D_Z \times D_E}$ denote the "queries", "keys", "values" linear projection matrices and node embedding, respectively, as defined in [34] and D_Z is the output feature dimension. H denotes number of attention head in multi-head attention setting.

The outputs $\mathbf{h}_k^{l+1}$ and $\mathbf{e}_{kj}^{l+1}$ are then passed through feed forward layers with residual connections and normalization

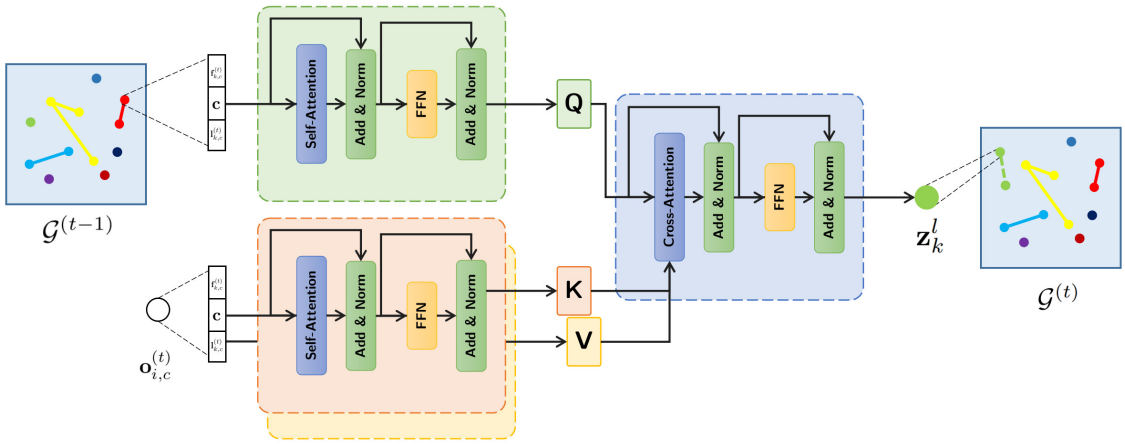

Figure 2. The proposed framework via Graph Transformer Networks. For every new detected object, we calculate new graph feature described in Sub-sec. 3.2.1 and 3.2.2. Then, we perform motion propagation and node merging operators that include the removing and the adding nodes in the graph via link prediction in Sub-sec. 3.2.3 and 3.2.4.

layers (see Fig. 2), defined as follows.

$$\mathbf{h}''^{l+1}_k = \text{norm}\left(\mathbf{h}'^{l+1}_k + \mathbf{h}^l_k\right) \tag{5}$$

$$\mathbf{h}'''^{l+1}_k = \text{FFN}^l_h\left(\mathbf{h}''^{l+1}_k\right) \tag{6}$$

$$\mathbf{h}^{l+1}_k = \text{norm}\left(\mathbf{h}''^{l+1}_k + \mathbf{h}'''^{l+1}_k\right) \tag{7}$$

where $\mathbf{h}''^{l+1}_k$ and $\mathbf{h}'''^{l+1}_k$ denote the outputs of intermediate layers. FFN is the feed forward layers.

$$\mathbf{e}''^{l+1}_{kj} = \text{norm}\left(\mathbf{e}'^{l+1}_{kj} + \mathbf{e}^l_{kj}\right) \tag{8}$$

$$\mathbf{e}'''^{l+1}_{kj} = \text{FFN}^l_e\left(\mathbf{e}''^{l+1}_{kj}\right) \tag{9}$$

$$\mathbf{e}^{l+1}_{kj} = \text{norm}\left(\mathbf{e}''^{l+1}_{kj} + \mathbf{e}'''^{l+1}_{kj}\right) \tag{10}$$

where $\mathbf{e}''^{l+1}_k$ and $\mathbf{e}'''^{l+1}_k$ denote the outputs of intermediate layers.

3.2.2 Graph Transformer Layer for Motion Modeling

In this section, we demonstrate how tracked objects in tracklet nodes are used as queries while newly detected objects are used as keys and values in our proposed transformer layer. This layer perform a cross-attention mechanism instead of self-attention mechanism where queries are different from keys. The input of this layer are the output node embedding from previous self-attention layers and the output of this layer are new tracklet nodes for the current frame t. It takes an object feature from previous frames as input query instead. This inherited object feature conveys the appearance and location information of previously seen objects, so this layer could well locate the position of the corresponding object on the current frame and output "tracking boxes". This design helps to capture the attention on current frame detection features and previous frame track queries, to continuously update the representation of object identity and location in each track query embedding.

We first put together all detected objects as $X_{\mathcal{O}} \in \mathbb{R}^{N_{\mathcal{O}} \times D_Z}$ and all tracked objects as $X_{\mathcal{T}} \in \mathbb{R}^{N_{\mathcal{T}} \times D_Z}$. Then the l-th output of the multi-head cross attention layer is defined as

$$\mathbf{z}^l_k = \mathbf{O}^l_z \Big\|_{i=1}^{H} \left(\sum_{j \in \mathbf{X}_{\mathcal{O}}} \mathbf{W}^{i,l}_{kj} \mathbf{V}^{i,l} \mathbf{X}^T_{\mathcal{T}}[k]\right) \tag{11}$$

$$\mathbf{W}^{i,l}_{kj} = \text{softmax}_j\left(\frac{\mathbf{Q}^{i,l}\mathbf{X}^T_{\mathcal{T}}[k] \cdot \mathbf{K}^{i,l}\mathbf{X}^T_{\mathcal{O}}[j]}{\sqrt{D_h}}\right) \tag{12}$$

where $\mathbf{Q}^{i,l}, \mathbf{K}^{i,l}, \mathbf{V}^{i,l} \in \mathbb{R}^{D_E \times D_Z}$, are the "queries", "keys" and "values" linear projection matrices, respectively, as defined in [34] and D_Z is the output feature dimension.

Similar to attention layer, we can stack multiple cross-attention layers together. Then we get the final output to pass through FFN to provide final set of new node embeddings including location and class predictions for frame t.

3.2.3 Cross-Camera Motion Propagation

In this section, we provide a more detailed formulation on how to obtain Re-ID features of the detected objects from camera c_k to camera c_j. First, we compute the transformation matrix to transform 3D object locations to 2D/image coordinates. This transformation which is composed of a transformation from camera-to-world for camera c_k, a transformation from world-to-camera for camera c_j, and a transformation from camera-to-image for camera c_j, is de-

fined as follows.

$$\mathbf{M}_{kj} = \mathbf{M}_{I_j} * \mathbf{M}_{E_j} * \mathbf{M}_{E_k}^{-1} \tag{13}$$

where $\mathbf{M}_{E_j}$ and $\mathbf{M}_{E_k}^{-1}$ are the extrinsic camera matrix for camera c_k to camera c_j, respectively. $\mathbf{M}_{I_j}$ is the intrinsic camera matrix for camera c_j. Note that we only consider two adjacent cameras c_k and c_j where they have a certain amount of overlapping views. Then, we use the transformed 2D/image location to extract the re-id features at the corresponding location on the image. Finally, we update the existing node or add a new node for all the tracked objects $\mathbf{tr}_{k,c_j}^{(t)}$.

3.2.4 Node Merging via Edge Scoring

After having transformed node and edge features, we train a fully connected layer and a softmax layer as a classifier to determine the similarity between two nodes as previously proposed in [23]. The classifier produces a probability score $s \in [0, 1]$. The higher the score is, the more likely the two nodes are linked. Then we remove detection nodes that have a low class score which indicates that the detection is matched with an existing tracklet. We also merge nodes that have high similarity scores that have the same camera encoding, i.e. detected within single camera and update edge weights as the similarities among tracklet nodes to indicate the same target ID from different cameras. These necessary steps are similar to a non-maximum suppression (NMS) applied to trajectory for post-processing although cross-attention layer help spatially discriminate almost identical track query embeddings merging to the same target ID.

3.3. Processing Flow

In this section, we briefly summarize the pipeline of our proposed graph transformer networks to predict tracklet motion, motion propagation and node merging in Algorithm 1.

3.4. Model Training

In this section, we present how to train our proposed graph transformer networks, including self-attention and cross-attention layers.

Training Data. We train our proposed method on a large-scale dataset, i.e. nuScenes, training set with 750 scenes of 20s each and use its validation set for our ablation study. The ground truth 3D bounding boxes and the extracted ReID features from the pre-trained models in [22, 41] were used together as the inputs for training GTN. Each training sample contains a chunk size of two consecutive frames from a training sequence.

Algorithm 1 The process pipeline for global graph constructing, motion prediction, propagation & node merging

Init $t \leftarrow 0$ /* Time */, $V \leftarrow \emptyset$
while $t < t_{\max}$ **do**
  Obtain the set of detected objects $\mathcal{O}_c^{(t)}$ from 3D object detector [16] in all cameras.
  for $\mathbf{o}_{k,c}^{(t)} \in \mathcal{O}_c^{(t)}$ **do**
    $\mathcal{V}^{(t)} \leftarrow \mathcal{V}^{(t-1)} \cup \mathbf{o}_{k,c}^{(t)}$ /* Add new nodes to graph */
    /* Use the vector $\{\mathbf{f}_{k,c}^{(t)} | \mathbf{c} | \mathbf{l}_{k,c}^{(t)}\}$ as node features. */
  end for
  for $k \in \mathcal{V}^{(t)}$ **do**
    Obtain new node embedding $\mathbf{h'}_k$ /* Section 3.2.1 */
  end for
  Obtain new set of nodes $\mathcal{V}'^{(t)}$ with location and classification of tracked objects $\mathbf{tr}_{k,c}^{(t)}$ via motion modeling /* Section 3.2.2 */
  for $c \in C$ **do**
    Propagate the location of $\mathbf{tr}_c^{(t)}$ to adjacent cameras /* Section 3.2.3 */
  end for
  for $v_i \in \mathcal{V}'^{(t)}$ **do**
    Obtain edge scoring to the remaining nodes and node merging /* Section 3.2.4 */
    Assign ID based on edge scores.
  end for
  $t \leftarrow t + 1$
end while

Training Loss. Our framework can be trained with two adjacent frames by optimizing for detections and tracklets prediction at frame t, given previous frame tracklets. Our joint objective function include *learning node embedding* capturing both structural information from the graph, *computing weighted linking score* between two nodes in the graph and *learning to predict tracklets motion*.

For *learning node embedding*, we measure binary cross-entropy loss $\mathcal{L}_{emb}$ between nodes that belong to the same objects for the model to output similar feature embeddings.

$$\begin{aligned}\mathcal{L}_{emb}(v_k) = &\sum_{v_j \in \mathcal{N}_b^{(t)}(v_k)} -\log\left(\sigma\left(< e'_{v_k}, e'_{v_j} >\right)\right) \\ &- w_g \sum_{v_i \in \mathcal{N}_g^{(t)}(v_k)} \log\left(1 - \sigma\left(< e'_{v_k}, e'_{v_i} >\right)\right)\end{aligned} \tag{14}$$

where $< \cdot >$ is the inner production between two vectors, σ is Sigmoid activation function, $\mathcal{N}_b^{(t)}(v_k)$ is the set of fixed-length random walk neighbor nodes of v_k at time step t, $\mathcal{N}_g^{(t)}(v_k)$ is a negative samples of v_i for time step t, $\mathcal{N}_a^{(t)}(v_k) = \mathcal{N}_b^{(t)}(v_k) \cup \mathcal{N}_g^{(t)}(v_k)$ and w_g, negative sam-

Method	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓
3D KF [36]	0.8153	0.5155	0.7382	1.6186
LSTM [5]	0.8041	0.4548	0.6744	1.6139
Ours	**0.5132**	**0.4388**	**0.3677**	**1.2189**

Table 1. Motion Errors comparison for different motion modeling

pling ratio, is an adjustable hyper-parameter to balance the positive and negative samples.

For edge scoring, we use a cross-entropy loss function $\mathcal{L}_c(e_{kj})$ based on measurement features to ensure the score between two nodes that are connected is higher than other nodes.

For *learning to predict tracklets motion*, we set prediction loss to measure the set of predictions for $N_{\mathcal{O}}$ detections and $N_{\mathcal{T}}$ tracklets comparing with ground truth objects in terms of classification and location (bounding boxes). Set-based loss produces an optimal bipartite matching between $N_{\mathcal{O}}$ detections and ground truth objects while $N_{\mathcal{T}}$ tracklets will be matched with boxes from previous frames. The matching cost is defined as follows.

$$\mathcal{L}_{set} = \sum_{i=1}^{N_{\mathcal{O}}+N_{\mathcal{T}}} (\lambda_{cls}\mathcal{L}_{cls} + \lambda_{box}\mathcal{L}_{box} + \lambda_{iou}\mathcal{L}_{iou}) \quad (15)$$

where $\lambda_{cls}, \lambda_{box}$ and λ_{iou} are combination weighting parameters for each component losses. $\mathcal{L}_{cls}$ is the cross-entropy loss between prediction classification and ground truth category labels. $\mathcal{L}_{box}$ and $\mathcal{L}_{iou}$ are the ℓ_1 loss and the generalized intersection over union (IoU) [27] for 3D bounding boxes. Finally, we have the total loss defined as

$$\mathcal{L}_{total} = \mathcal{L}_{emb} + \mathcal{L}_c + \mathcal{L}_{set} \quad (16)$$

4. Experimental Results

In this Section, we detail the benchmark dataset and metrics in Subsection 4.1. Then, the setups for all experiments and the ablation study will be presented in Subsections 4.2 and 4.3 respectively. The comparisons with the State-of-the-Art (SOTA) methods will be detailed in Subsection 4.4 on a large-scale Tracking Challenge, i.e. nuScenes Vision Track.

4.1. Benchmark Dataset and Metrics

4.1.1 Dataset

nuScenes [2] is one of the large-scale datasets for Autonomous Driving with 3D object annotations. It contains 1,000 videos of 20-second shots in a setup of 6 cameras, i.e. 3 front and 3 rear ones, with a total of 1.4M images. It also provides 1.4M manually annotated 3D bounding boxes of 23 object classes based on LiDAR data. This dataset is an official split of 700, 150 and 150 videos for training, validation and testing, respectively.

4.1.2 Metrics

The proposed method is evaluated using both detection and tracking metrics described in [2].

Detection Metrics. A commonly used metric, i.e. *Mean Average Precision (mAP)*, is defined as a match using a 2D center distance on the ground plane instead of intersection over union cost for nuScenes detection challenges.

Similarly, other motion-related metrics are also defined in nuScenes, such as *Average Translation Error (ATE)* measuring Euclidean center distance in 2D in meters, *Average Scale Error (ASE)* computing as $1 - IOU$ after aligning centers and orientation, *Average Orientation Error (AOE)* measuring by the smallest yaw angle difference between prediction and ground-truth in radians, *Average Velocity Error (AVE)* measuring the absolute velocity error in m/s and *Average Attribute Error (AAE)* computing as $1 - acc$, where acc is the attribute classification accuracy.

Last but not least, we also use the *nuScenes Detection Score (NDS)* that is based on a simple additive weighting of the mean of all other metrics above, including *mAP*, *mATE*, *mASE*, *mAOE*, *mAVE* and *mAAE*.

Tracking Metrics. The tracking performance is measured using the popular *CLEAR MOT* metrics [1] including *MOTA*, *MOTP*, ID switch (*IDS*), mostly tracked (*MT*), mostly lost (*ML*), fragmented (*FRAG*). Similar to nuScenes, we use two accumulated metrics introduced in [36] as the main metrics, including the average over the MOTA metric (*Average MOTA (AMOTA)*) and the average over the MOTP metric (*Average MOTP (AMOTP)*).

4.2. Experiments Setup

The proposed graph transformer networks module is trained with two consecutive frames where the graph $\{\mathcal{G}^{(t-1)}\}$ in the previous time step is used to predict new graph $\mathcal{G}^{(t)}$ at time step t. Then, Mini-batch (chunk of two) gradient descent is employed with Adam optimizer to learn all the parameters in the attention layers.

4.3. Ablation Study

In this section, we present some experiments to ablate the effect of each component of the proposed framework. Particularly, this section aims to demonstrate the followings: 1. better motion modeling with cross-attention layer in GTN; 2. the role of architecture choice of graph transformer networks.

The Role of Motion Model In this experiment, we evaluate the effectiveness of different motion modeling methods on detection performance. We use the locations predicted by motion models to compare with ground truth locations in

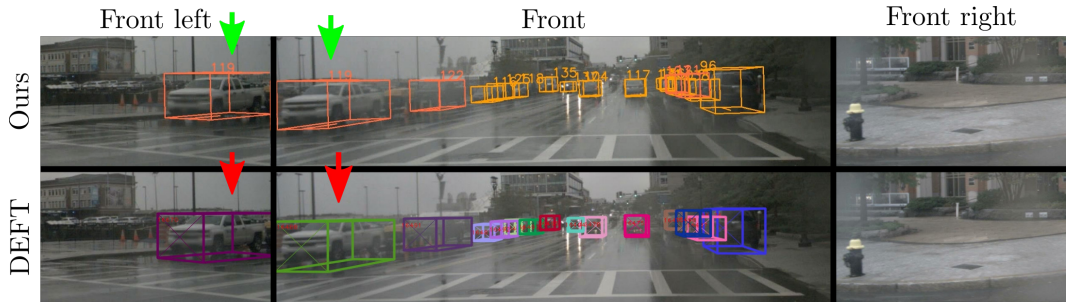

Figure 3. Our proposed method (top) can recognize a positive tracking case compare with a MC-MOT system which has no object's correlations linking module (i.e. DEFT) for all cameras (bottom). Green arrows indicate true positive tracking samples, red arrows indicate false negative tracking samples. Best viewed in color and zoom in.

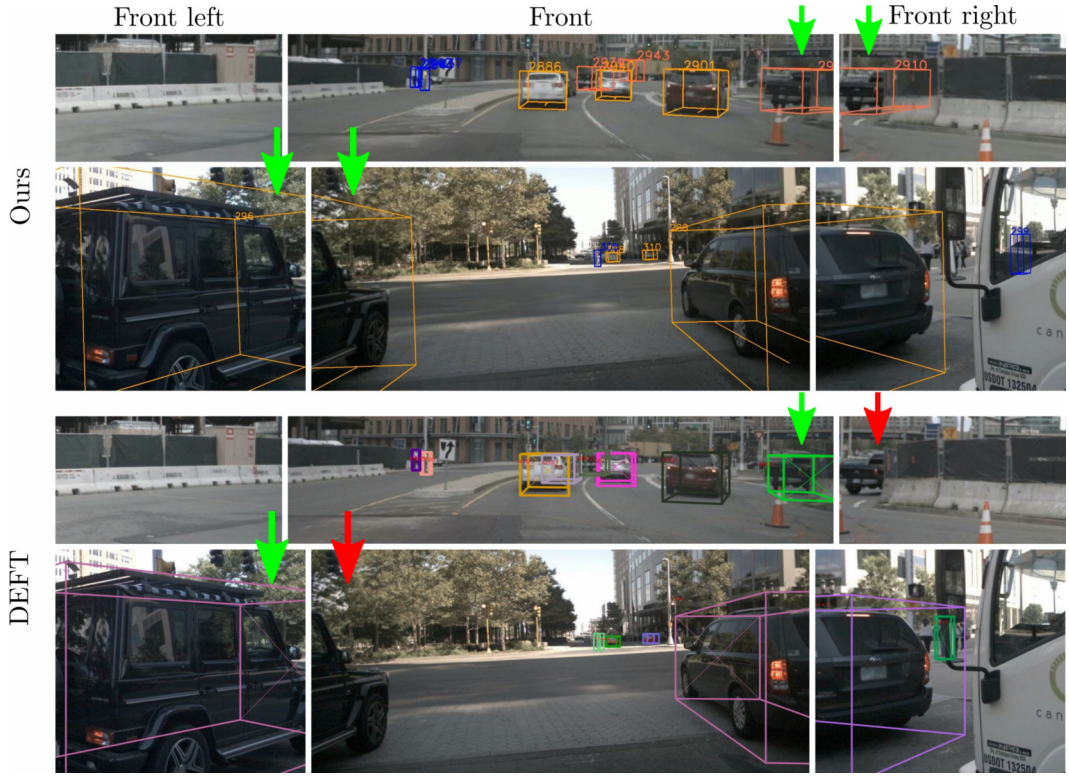

Figure 4. Our proposed method (top) can recover a false negative detection case compared with a MC-MOT system which runs independently on each camera (i.e. DEFT) (bottom). Green arrows indicate true positive detection samples, red arrows indicate false negative detection samples. Best viewed in color and zoom in.

Structures	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓
Self-attn 1-layer	0.812	0.298	0.820	**1.187**
Self-attn 2-layer	0.785	**0.286**	0.703	1.284
Self-attn 3-layer	**0.750**	0.293	**0.485**	1.432
Cross-attn 1-layer	0.824	0.293	0.866	1.281
Cross-attn 2-layer	0.772	**0.279**	0.670	1.287
Cross-attn 3-layer	**0.513**	0.439	**0.368**	**1.219**

Table 2. Ablation study on different configuration for self-attention and cross-attention layers.

terms of motion-related metrics. In such way, we can evaluate how good the motion model capturing and predicting the motion of tracked objects. We compare with two other commonly used motion models, i.e. 3D Kalman Filter [36] and LSTM [5]. As shown in Table 1, our GTN gives better results than a classical object state prediction technique, i.e. 3D Kalman Filter used in [36] and a deep learning based technique, i.e. LSTM module, used in [5].

Method	Glo. Assoc.	AMOTA	AMOTP	MOTAR	MOTA ↑	MOTP ↓	RECALL ↑	MT ↑	ML ↓	IDS ↓	FRAG ↓
MonoDIS [29]	✗	0.045	1.793	0.202	0.047	0.927	0.293	395	3961	6872	3229
CenterTrack [42]	✗	0.068	1.543	0.349	0.061	0.778	0.222	524	4378	2673	1882
DEFT [5]	✗	0.213	1.532	0.49	0.183	0.805	0.4	1591	2552	5560	2721
QD-3DT [15]	✗	**0.242**	1.518	**0.58**	**0.218**	0.81	0.399	1600	2307	5646	2592
Ours	✓	0.24	**1.52**	0.568	0.197	**0.832**	**0.453**	**1643**	**2162**	**1362**	**1462**

Table 3. Comparison of 3D tracking performance on the nuScenes validation set for Vision Track challenge. **Glo. Assoc.** indicates method linking object IDs across all cameras

Method	mAP ↑	NDS ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
MonoDIS [29]	0.2976	0.3685	0.7661	0.2695	**0.5839**	1.3619	0.184
MonoDIS [29] + **Our MP + NM**	**0.3019**	**0.3893**	**0.6558**	**0.2410**	0.6787	**1.3209**	**0.184**
CenterNet [43]	0.3027	0.3262	0.7152	0.2635	**0.6158**	1.4254	0.6567
CenterNet [43] + **Our MP + NM**	**0.3487**	**0.4016**	**0.5417**	**0.2023**	0.6317	**1.3094**	**0.6567**
KM3D [16]	0.2763	0.3201	0.7495	0.2927	0.4851	**1.4322**	0.6535
KM3D [16] + **Our MP + NM**	**0.3503**	**0.4117**	**0.6998**	**0.2323**	**0.1861**	1.8341	**0.5166**

Table 4. Comparison of 3D object detectors with and without using our motion propagation (MP) and node merging (NM) modules in terms of detection metrics on the nuScenes validation set for Vision Detection challenge

The Configuration for Graph Transformer Networks We conduct additional ablation studies to evaluate the effects on configuration of the attention modules in GTN, including the number of attention layers. Table 2 shows the performance of our proposed framework in terms of detection metrics using various configuration of the attention modules. We change the number of layer for self-attention and the cross-attention layers independently. We use a fixed number of layers, i.e. 2, for self-attention and the cross-attention layers while changing the other, respectively.

4.4. Comparison against The State-of-the-Art Methods

In this section, we first compare our proposed framework with other vision-based (without using LiDAR or RADAR information) tracking approaches, which are the top in nuScenes vision only tracking challenge leaderboard. Then we conduct an experiment to demonstrate that using tracked 3D bounding boxes from our tracking framework can actually improve the detection metrics.

Comparison against Tracking Methods on Tracking Metrics This experiment compares our proposed method with other vision-based methods, including MonoDIS [29], CenterTrack [42] and DEFT [5], QD-3DT [15] which are the top/winner of nuScenes vision only tracking challenge. As we can see in Table 3, our method decreases error rates compared to top approaches, i.e. DEFT, in most of the metrics. Fig. 3 illustrates the key factor that help improve the tracking performance is that we perform appearance matching across cameras in addition to motion modeling. It shows that our proposed method (top) can assign object ID globally between cameras compared with DEFT [5] (bottom).

Comparison against Detection Methods on Detection Metrics Table 4 demonstrates that the combination of object detector and our motion propagation (MP) and node merging (NM) modules achieves the better results than original object detector. In this experiment, we compare three different 3D object detectors, including KM3D [16], MonoDIS [29] and CenterNet [43]. The best result achieves with the combination of KM3D object detector [16] and our MP+NM modules since it is guided by global decoded locations from our transformation procedure as described in 3.2.3. Fig. 4 illustrates the improvement on detector fail cases with the help from our tracking framework.

5. Conclusions

This paper has introduced a new global association graph model to solve the MC-MOT problem for AV. The proposed framework can learn to perform tracking frame-by-frame in an end-to-end manner starting from detections to motion prediction and global association tracklets with detections. These tasks are enhanced with self-attention and cross-attention layers so that the proposed graph can capture both structural and motion across cameras. The experiments show performance improvements in a large-scale dataset in AV in terms of vision-based detection and tracking accuracy.

Acknowledgment

This material is based upon work supported in part by the US NSF Data Science, Data Analytics that are Robust and Trusted (DART) and NSF WVAR-CRESH Grant.

References

[1] 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. 6

[2] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1, 6

[3] Yinghao Cai and Gerard Medioni. Exploring context information for inter-camera multiple target tracking. In *IEEE Winter Conference on Applications of Computer Vision*, pages 761–768. IEEE, 2014. 1, 3

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. 2

[5] Mohamed Chaabane, Peter Zhang, Ross Beveridge, and Stephen O'Hara. Deft: Detection embeddings for tracking. *arXiv preprint arXiv:2102.02267*, 2021. 1, 2, 6, 7, 8

[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 1

[7] Weihua Chen, Lijun Cao, Xiaotang Chen, and Kaiqi Huang. An equalized global graph model-based approach for multi-camera object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(11):2367–2381, 2016. 1, 3

[8] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *arXiv preprint arXiv:2001.05673*, 2020. 2

[9] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[12] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. Graph convolutional tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1

[14] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krähenbühl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *ICCV*, 2019. 2

[15] Hou-Ning Hu, Yung-Hsu Yang, Tobias Fischer, Fisher Yu, Trevor Darrell, and Min Sun. Monocular quasi-dense 3d object tracking. *ArXiv:2103.07351*, 2021. 8

[16] Peixuan Li. Monocular 3d detection with geometric constraints embedding and semi-supervised training, 2020. 3, 5, 8

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. 2

[18] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers, 2021. 2

[19] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation, 2018. 2

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 2

[21] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 2

[22] Yijun Qian, Lijun Yu, Wenhe Liu, and Alexander G. Hauptmann. Electricity: An efficient multi-camera vehicle tracking system for intelligent city. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020. 3, 5

[23] Kha Gia Quach, Pha Nguyen, Huu Le, Thanh-Dat Truong, Chi Nhan Duong, Minh-Triet Tran, and Khoa Luu. Dyglip: A dynamic graph model with link prediction for accurate multi-camera multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13784–13793, 2021. 3, 5

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018. 2

[25] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models, 2019. 2

[26] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *CVPR*, 2017. 2

[27] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019. 6

[28] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[29] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel Lopez-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 8

[30] Jean-Paul Skeete. Level 5 autonomy: The new face of disruption in road transport. *Technological Forecasting and Social Change*, 134:22–34, 2018. 1

[31] 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, 2020. 2

[32] 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: An open dataset benchmark. *arXiv preprint arXiv:1912.04838*, 3, 2019. 1

[33] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021. 2

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 3, 4

[35] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers, 2020. 2

[36] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics, 2020. 2, 6, 7

[37] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris Kitani. GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with 2D-3D Multi-Feature Learning. *CVPR*, 2020. 2

[38] Xinshuo Weng, Ye Yuan, and Kris Kitani. Parallelized 3D Tracking and Forecasting with Graph Neural Networks and Diversity Sampling. *arXiv:2003.07847*, 2020. 2

[39] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021. 2

[40] Zhimeng Zhang, J. Wu, Xuan Zhang, and C. Zhang. Multi-target, multi-camera tracking by hierarchical clustering: Recent progress on dukemtmc project. *ArXiv*, abs/1712.09531, 2017. 3

[41] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *ICCV*, 2019. 5

[42] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *ECCV*, 2020. 2, 8

[43] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019. 2, 8

[44] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection, 2017. 2

[45] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection. *arXiv e-prints*, page arXiv:1908.09492, Aug 2019. 2

[46] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2

[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2020. 2