GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with 2D-3D Multi-Feature Learning

Xinshuo Weng, Yongxin Wang, Yunze Man, Kris Kitani
Robotics Institute
Carnegie Mellon University
{xinshuow, yongxinw, yman, kkitani}@cs.cmu.edu

Abstract

3D Multi-object tracking (MOT) is crucial to autonomous systems. Recent work uses a standard tracking-by-detection pipeline, where feature extraction is first performed independently for each object in order to compute an affinity matrix. Then the affinity matrix is passed to the Hungarian algorithm for data association. A key process of this standard pipeline is to learn discriminative features for different objects in order to reduce confusion during data association. In this work, we propose two techniques to improve the discriminative feature learning for MOT: (1) instead of obtaining features for each object independently, we propose a novel feature interaction mechanism by introducing the Graph Neural Network. As a result, the feature of one object is informed of the features of other objects so that the object feature can lean towards the object with similar feature (i.e., object probably with a same ID) and deviate from objects with dissimilar features (i.e., object probably with different IDs), leading to a more discriminative feature for each object; (2) instead of obtaining the feature from either 2D or 3D space in prior work, we propose a novel joint feature extractor to learn appearance and motion features from 2D and 3D space simultaneously. As features from different modalities often have complementary information, the joint feature can be more discriminate than feature from each individual modality. To ensure that the joint feature extractor does not heavily rely on one modality, we also propose an ensemble training paradigm. Through extensive evaluation, our proposed method achieves state-of-the-art performance on KITTI and nuScenes 3D MOT benchmarks. Our code will be made available at https://github.com/xinshuoweng/GNN3DMOT

1. Introduction

Multi-object tracking (MOT) is an indispensable component of many applications such as autonomous driving

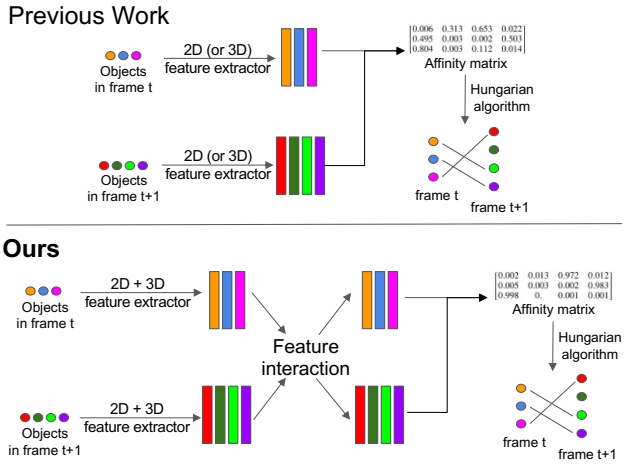


Figure 1. (**Top**): Prior work often employs a 2D or 3D feature extractor and obtain the feature independently from each object. (**Bottom**): Our work proposes a joint 2D and 3D feature extractor and a feature interaction mechanism to improve the discriminative feature learning for data association in MOT.

[21, 41, 49, 47]. Recent work approaches MOT in an online manner with a tracking-by-detection [4, 45] pipeline, where an object detector [32, 46, 28, 48] is applied to all frames and feature is extracted *independently* from each detected object. Then the pairwise feature similarity is computed between objects and used to solve the MOT with a Hungarian algorithm [40]. The key process of this pipeline is to learn discriminative features for objects with different identities.

Our observation is that the feature extraction in prior work is always independent for each object as shown in Figure 1 (Top) and there is no interaction. For example, an object's 2D appearance feature is computed only from its own image patch, not involving with features of other objects. We found that *this independent feature extraction is sub-optimal for discriminative feature learning*. This is reasonable as the feature similarity of different objects should be dependent in MOT, given the fact that an object in current frame can be matched to at most one object in previous

frame. In other words, if the pairwise feature similarity of two objects is increased, then the pairwise feature similarity of any one of these two objects with all other different objects should be decreased to avoid confusion for matching.

Based on the observation, we propose a novel *feature interaction mechanism* for MOT as shown in Figure 1 (Bottom). We achieve this by introducing the Graph Neural Networks (GNNs). To the best of our knowledge, our work is the first applying the GNNs to MOT. Specifically, we construct a graph with each node being the object feature. Then, at every layer of the GNNs, each node can update its feature by aggregating features from other nodes. This node feature aggregation process is useful because each object feature is now not isolated and can be adapted with respect to other object features. We observe that, after a few GNN layers, the computed affinity matrix becomes more and more discriminative than the affinity matrix obtained without feature interaction.

In addition to the feature interaction, another primary question for discriminative feature learning in MOT is about feature selection, *i.e.*, "what type of feature should we learn?". Among different features, motion and appearance are proved to be the most useful features. Although prior works [50, 20, 53, 2] have explored using both appearance and motion features, they only focus on either 2D or 3D space as shown in Figure 1 (top). That means, prior works use only 2D feature when approaching the 2D MOT or use only 3D feature when approaching the 3D MOT. However, this is not optimal as we know that 2D and 3D information are complementary. For example, two objects can be very close in the image but actually at a distance in 3D space because of depth discrepancy. As a result, the 3D motion feature is more discriminative in this case. On the other hand, 3D detection might not be very accurate for objects at a large distance to the camera and thus 3D motion can be very noisy. In this case, the 2D motion feature might be more discriminative.

To this end, we also propose a novel feature extractor that jointly learns motion and appearance features from both 2D and 3D space as shown in Figure 1 (bottom). Specifically, the joint feature extractor has four branches with each branch being responsible for 2D appearance, 2D motion, 3D appearance and 3D motion feature, respectively. Features from all four branches are fused before feeding into the GNNs for feature interaction. To ensure that the network does not heavily rely on one branch, we follow the concept of Dropout [34] and propose an ensemble training paradigm, allowing the network randomly turning off branches during training. As a result, our network can learn discriminative features on all branches.

Our entire network shown in Figure 2 is end-to-end trainable. We summarize our contributions as follows: (1) We propose a novel feature interaction mechanism for MOT by introducing the GNNs; (2) We propose a novel feature extractor along with an ensemble training paradigm to learns discriminative motion and appearance features from both 2D and 3D; (3) We achieve state-of-the-art performance on two standard 3D MOT benchmarks and also a competitive performance on the corresponding 2D MOT benchmarks.

2. Related Work

Online Multi-Object Tracking. Recent work approaches online MOT using a tracking-by-detection pipeline, where the performance is mostly affected by two factors: object detection quality and discriminative feature learning. After the affinity matrix is computed based on the pairwise similarity of learned discriminative feature, online MOT can be solved as a bipartite matching problem using the Hungarian algorithm [40]. For a fair comparison with others, prior work often uses the same detection results so that the factor of the object detection quality can be eliminated.

To obtain discriminative feature, prior work mostly focuses on the feature selection. Among different features, it turns out that motion and appearance are the most discriminative features. Early work employs hand-crafted features such as spatial distance [25] and Intersection of Union (IoU) [5, 20] as the motion feature, and use color histograms [55] as the appearance feature. Recent works [35, 2, 53, 9, 50] often employ the Convolutional Neural Networks (CNNs) to extract the appearance feature. For the motion feature, many filter-based methods [45, 4] and deep learning based methods [53, 2] have been proposed. Although prior works [50, 20, 53, 2] have explored using both motion and appearance features, they have been only focusing on either 2D or 3D space, which might lead to failure of tracking if the feature from 2D or 3D is not robust at certain frames. In contrast to prior work, we propose a novel feature extractor with four branches that jointly learns motion and appearance features from both 2D and 3D space. As a result, our method can compensate for the inaccuracy of the feature in one branch with features from other branches.

Perhaps [56] is the closest to our work in terms of the feature selection as [56] also proposes to jointly learn the 2D and 3D features. However, our work differs from [56] as follows: (1) [56] only uses the appearance feature without leveraging any motion cue. We observe that, when using both motion and appearance features, performance can be improved significantly; (2) With our proposed ensemble training paradigm, the network can be enhanced to extract high-quality features for all four branches. However, [56] simply learns 2D and 3D appearance features simultaneously, which might lead to one feature dominating the other, which violates the purpose of multi-feature learning; (3) The last but most important is that our work also proposes a feature interaction mechanism for discriminative feature learning by introducing the GNNs while [56] does not.

Graph Neural Networks. In addition to the feature selection, we also propose a novel feature interaction mechanism for discriminative feature learning in MOT, which is achieved by introducing the GNNs. GNNs was first proposed by [12] to directly process graph-structured data using neural networks. The major component of the GNNs is the node feature aggregation technique, with which node can update its feature by interacting with other nodes. With this technique, significant success has been achieved in many fields using GNNs such as semantic segmentation [7, 54], action recognition [19, 31, 57, 42], single object tracking [10], person re-identification [51], point cloud classification and segmentation [44].

Although GNNs have shown promising performance in many fields, there is no existing work that applies GNNs to MOT. To the best of our knowledge, our work is the first attempt using GNNs for online MOT. With the node aggregation technique of the GNNs, our proposed method can iteratively evolve the object features so that the feature of different objects can more discriminative. Our work is significantly different from prior work in which object features are isolated and independent of other objects. Perhaps the relation network proposed in [15] is the closest to our work in terms of modeling the feature interaction. However, the feature interaction in [15] only exists in the spatial domain to encode context information for object detection. Although a temporal relation network is proposed in the follow-up work [52], the feature of a tracked object is only aggregating from its past trajectory and no interaction with other object features exist. In contrast, our work proposes a generic feature interaction framework that can model any kind of interaction in both spatial and temporal domains and is applicable for features from different modalities.

3. Approach

The goal of online MOT is to associate existing tracked objects from previous frame with new detected objects in current frame. Given M tracked objects $o_i \in O$ at frame t where $i \in \{1, 2, \cdots, M\}$ and also N detected objects $d_j \in D$ in frame t+1 where $j \in \{1, 2, \cdots, N\}$, we want to learn discriminative feature from O and D and then find the correct matching based on the pairwise feature similarity.

In Figure 2, our entire network consists of: (a) a 3D appearance and motion feature extractor; (b) a 2D appearance and motion feature extractor. Both 2D and 3D feature extractors are applied to all objects in O and D and then the extracted features are fused together, (c) a graph neural network that takes the fused object feature as input and constructs a graph with node being the object feature in frame t and t+1. Then, the graph neural network iteratively aggregates the node feature from the neighborhood and computes the affinity matrix for matching using edge regression.

To apply the online MOT to an entire video at inference time, an object detector must be applied to all frames in advance. As our 2D and 3D feature extractors need object detection correspondences in 2D and 3D space, it is nontrivial to obtain the 2D detections and 3D detections separately and then obtain the detection correspondences. Instead, we only use a 3D object detector to obtain 3D detections and then 2D detections are projected from the 3D detections given the camera projection matrix. Following [32, 46], we parameterize the 3D detection as a tuple of $d^{3D}=\{x, y, z, l, w, h, \theta\}$ where (x, y, z) denotes the object center in 3D space, (l, w, h) denotes the object size and θ is the heading angle. For 2D detection, we parameterize it as a tuple of $d^{2D}=\{x_c, y_c, w, h\}$ where (x_c, y_c) is object center in 2D space and $(w,\ h)$ denotes width and height. For tracked objects O, we use the same parameterization except for having an additional assigned ID I, *i.e.*, $o^{3D}=\{x, y, z, l, w, h, \theta, I\}$ and $o^{2D}=\{x_c, y_c, w, h, I\}$.

3.1. Joint 2D and 3D Feature Extractor

To utilize the information for different modalities and learn discriminative feature, our proposed joint feature extractor with four branches leverages appearance and motion features from both 2D and 3D space, where two branches perform the 3D appearance and motion feature extraction and other two branches perform the 2D feature extraction.

3D Appearance/Motion Feature Extraction. As shown in Figure 2 (a), given a detected object d_j^{3D} in frame t+1 or a tracked object o_i^{3D} in frame t, we want to obtain the corresponding 3D feature $f_{3D_i}^{t}$ and $f_{3D_j}^{t+1}$ including both appearance and motion information. For appearance branch, we use the LiDAR point cloud as the appearance cue. We first extract the point cloud enclosed by the 3D detection box and then apply the PointNet [8, 26] to obtain the feature. For motion branch, we directly use the 3D detection box as the motion cue. Note that we use different 3D motion feature extractor for tracked and detected objects as tracked objects have associated trajectory in past frames while detected objects do not have. For tracked object o_i^{3D}, we apply the LSTMs that take into the object's 3D detections in past T frames to obtain the feature. For detected object d_j^{3D}, we use a 2-layer MLP (Multi-Layer Perceptron) that takes the detection in frame t+1 as input to extract the feature. The final 3D feature $f_{3D_i}^{t}$ and $f_{3D_j}^{t+1}$ for tracked and detected objects is obtained by concatenating the 3D motion and appearance features. To balance the contribution of the motion and appearance features, we force the final motion and appearance feature vectors to have the same dimensionality.

2D Appearance/Motion Feature Extraction. As in Figure 2 (b), the structure of 2D feature extractor is very similar to the 3D feature extractor explained above except for two aspects: (1) objects o_i^{2D} or d_j^{2D} are parameterized as 2D box (x_c, y_c, w, h) in instead of the 3D box. Therefore, the input

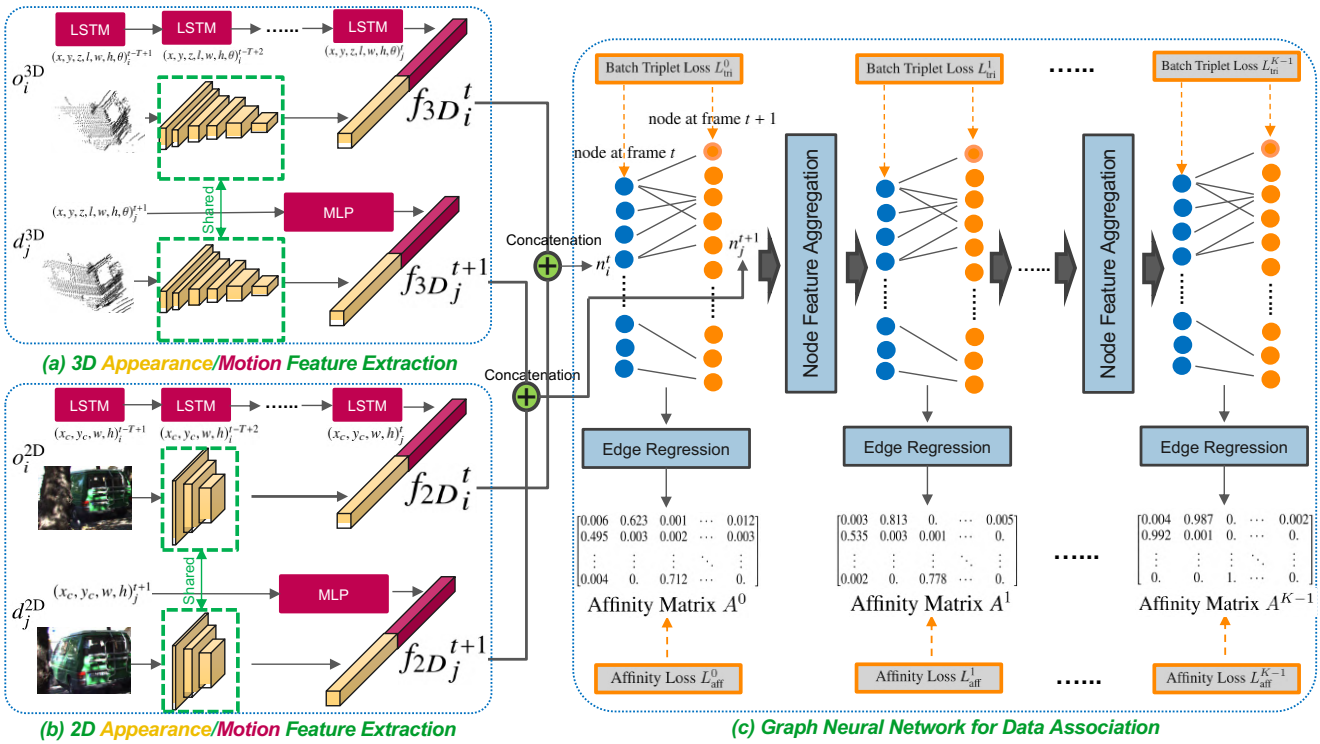


Figure 2. **Proposed Network.** (a)(b) Our proposed joint feature extractor obtains the feature for tracked objects o_i in frame t and detected objects d_j in frame t+1 by utilizing the appearance and motion information from both 2D and 3D space; (c) We fuse the object features from different branches and construct a graph with the node being the object feature. Then, in every layer of the GNN, the node features are iteratively updated with the node feature aggregation technique and the affinity matrix is computed via the edge regression module. To train the entire network, we employ batch triplet loss on the node feature and affinity loss on the predicted affinity matrix in all layers.

to motion branch is different; (2) for appearance branch, we use image patch as the appearance cue, which is cropped from the entire image based on the 2D detections. To process the image patch and obtain the 2D appearance feature, we use CNNs (*e.g.*, VGGNet [33] or ResNet [14]). The final 2D feature $f_{2\mathrm{D}_i}^t$ and $f_{2\mathrm{D}_j}^{t+1}$ is obtained by concatenating the 2D motion and appearance features.

Feature Fusion. Before feeding the object feature into the graph neural network, we need to fuse the feature obtained from the 2D and 3D feature extractors. We have tried two different fusion operators: (1) concatenate the 2D and 3D features; (2) add the 2D and 3D features together. Using the "add" fusion operator is feasible because we also force the 2D and 3D features (*e.g.*, $f_{2\mathrm{D}_i}^t$ and $f_{3\mathrm{D}_i}^t$) to have the same dimensionality. We will show how different fusion operator affects the performance in the experiments. We use the concatenation as the fusion operator in our final network.

Ensemble Training Paradigm. As our network has four branches of feature extractor and one branch may dominate the others, which violates the purpose of multi-feature learning. To avoid such cases, we propose an ensemble training paradigm. Similar to the concept of the Dropout [34], we randomly drop one to three branches (*i.e.*, keep at least one) during every iteration of the training. Specifically, we create two random generators. The first random generator produces 0 ("not drop") or 1 ("drop") with a ratio r of producing "drop", where r is a scalar between 0 and 1. In the case of "drop", the second random generator produces a random integer between 1 to 14, which controls which combination of branches should be dropped. For example, the dropped branches can be a combination of 2D motion and 3D appearance branches.

3.2. Graph Neural Network for Data Association

Graph Construction. After feature fusion, we should have M features for tracked objects in frame t and also N features for detected objects in frame t+1. We then construct a graph with each node being the object feature. In total, we have M+N nodes in the graph as shown in Figure 2 (c). We then define the neighborhood of the node (*i.e.*, edges in the graph). One simple way is to have an edge between each pair of nodes, which results in a fully-connected graph and can be computationally expensive. Instead of using this simple edge construction, we utilize prior knowledge about online MOT, where the matching should only happen across frames (*i.e.*, not within the same frame). Specifically, we construct the edge only between the pair of nodes in different frames. Also, for any tracked object o_i in frame t, the

possible matched detection d_j in frame t+1 is most likely located in the nearby location. Therefore, we construct the edge only if two nodes' detection centers have distance less than $\text{Dist}^{\text{3D}}_{\text{max}}$ meters in 3D space and $\text{Dist}^{\text{2D}}_{\text{max}}$ pixels in the image. As a result, we have a sparse edge connection across frames in our final network as shown in Figure 2 (c).

Edge Regression. To solve the online MOT, we need to compute the $M \times N$ affinity matrix A based on the pairwise similarity of the features extracted from M tracked objects in frame t and N detected objects in frame t+1. In the context of GNN, we call this process as edge regression. We have tried three metrics for measuring the feature similarity. The first two are cosine similarity and negative L2 distance, which are conventional metrics used in the MOT community. The third one is to employ a two-layer MLP that takes the difference of two node features as input and outputs a scalar value between 0 to 1 as the pairwise similarity score:

$$A_{ij} = \text{Sigmoid}(\sigma_2(\text{ReLU}(\sigma_1(n_i^t - n_j^{t+1})))), \quad (1)$$

where σ_1 and σ_2 are two different linear layers. In addition, n_i^t and n_j^{t+1} are two node features in different frames where $i \in \{1, 2, \cdots, M\}$, $j \in \{1, 2, \cdots, N\}$, In our final network, we use the MLP as the metric for edge regression and we will show how performance is affected by different metrics in the experiments.

Node Feature Aggregation. To model feature interaction in GNN, we iteratively update the node feature by aggregating features from the neighborhood (*i.e.*, nodes connected by the edge) in every layer of the GNN as shown in Figure 2 (c). To comprehensively analyze how different types of node aggregation rules affects the performance of the MOT, we study four rules used in modern GNNs (e.g., GraphConv [22], GATConv [37], EdgeConv [44], *etc*) as below:

$$\text{(Type 1)} \quad n_i^{t\prime} = \sum\nolimits_{j \in \mathcal{N}(i)} \sigma_3(n_j^{t+1}), \quad (2)$$

$$\text{(Type 2)} \quad n_i^{t\prime} = \sigma_4(n_i^t) + \sum\nolimits_{j \in \mathcal{N}(i)} \sigma_3(n_j^{t+1}), \quad (3)$$

$$\text{(Type 3)} \quad n_i^{t\prime} = \sigma_4(n_i^t) + \sum\nolimits_{j \in \mathcal{N}(i)} \sigma_3(n_j^{t+1} - n_i^t), \quad (4)$$

$$\text{(Type 4)} \quad n_i^{t\prime} = \sigma_4(n_i^t) + \sum\nolimits_{j \in \mathcal{N}(i)} \sigma_3(A_{ij}(n_j^{t+1} - n_i^t)), \quad (5)$$

where $\mathcal{N}(i)$ denotes a set of neighborhood nodes in frame t+1 with respect to the node i in frame t, given the fact that edge is only defined across frames in our sparse graph construction. Also, σ_3, σ_4 are linear layers which have different weights across layers of the GNN. The weight A_{ij} is obtained from the affinity matrix in the current layer. Note that before the node feature aggregation in each layer, a nonlinear ReLU operator is applied to the node features.

In type 1 rule of Eq. 2, node feature is updated by aggregating features from only the neighborhood nodes, which is limited for MOT because the feature of the node itself is forgotten after aggregation. In type 2 rule, we compensate for this limitation by adding feature of the node itself as shown in the first term of Eq. 3 in addition to the features aggregation from the neighborhood. In type 3 rule of Eq. 4, feature from the neighborhood node in the second term is replaced with the difference of the features between the node itself and the neighborhood node. In type 4 rule of Eq. 5, we add an attention weight obtained from the affinity matrix to the feature aggregation in the second term so that the network can focus on the neighborhood node with a higher affinity score, *i.e.*, possibly the object with the same ID. We will evaluate all four node feature aggregation rules and also the number of graph layers (*i.e.*, number of times performing the node feature aggregation) in the experiments.

3.3. Losses

Our proposed network employs two losses in all K layers during training: (1) batch triple loss L_{tri}; (2) affinity loss L_{aff}. We can summarize the entire loss function L as below:

$$L = \sum\nolimits_{k=0}^{K-1} (L_{\text{tri}}^k + L_{\text{aff}}^k). \quad (6)$$

Batch Triplet Loss. In order to learn discriminative features for matching, we first apply a batch triplet loss to node feature in every layer of the GNN. For node n_i^t that has a matched node n_j^{t+1} (*i.e.*, the object o_i has the same ID with d_j), the batch triplet loss in each layer is defined as:

$$\begin{aligned} L_{\text{tri}} = \max(\; & ||n_i^t - n_j^{t+1}|| - \min_{\substack{d_s \in D \\ id_i \neq id_s}} ||n_i^t - n_s^{t+1}|| \\ & - \min_{\substack{o_r \in O \\ id_r \neq id_j}} ||n_r^t - n_j^{t+1}|| + \alpha, \; 0), \end{aligned} \quad (7)$$

where α is the margin of the triplet loss. n_s^{t+1} is a node in frame t+1 that has a different ID from node n_j^{t+1} and n_i^t. Similarly, n_r^t is a node in frame t that has a different ID from node n_i^t and n_j^{t+1}. Note that the above batch triplet loss is slightly different from the original definition as in [39, 1]. First, we only have one positive pair of node that has the same ID as shown in the first term $||n_i^t - n_j^{t+1}||$ so that there is no need to apply the max operation over a batch. For the negative pair of node, we have two symmetric terms, where the first negative term forces the node feature n_i^t to be different from any node that has a different ID in frame t+1 and the second negative term forces the node feature n_j^{t+1} to be different from any node that has a different ID in frame t. In the case that n_i^t does not have a matched node in frame t+1 with the same ID, we delete the first term $||n_i^t - n_j^{t+1}||$ for the positive pair of node in Eq. 7 and only minimize the remaining two negative terms in the loss L_{tri}.

Affinity Loss. In addition to the batch triplet loss applied to the node feature, we also employ an affinity loss L_{aff} to directly supervise the final output of the network, *i.e.*, the predicted affinity matrix A. Our affinity loss consists of two individual losses. First, as we know that the ground truth affinity matrix A^g can only have integer 0 or 1 on all the entries, we can formulate the prediction of the affinity matrix

as a binary classification problem. Therefore, our first loss is the binary cross entropy loss L_{bce} that is applied on each entry of our predicted affinity matrix A as shown below:

$$L_{\text{bce}} = \frac{-1}{MN} \sum_{i}^{M} \sum_{j}^{N} A_{ij}^{g} \log A_{ij} + (1 - A_{ij}^{g}) \log(1 - A_{ij}). \quad (8)$$

Also, we know that each tracked object o_i^t in frame t can only have either one matched detection d_j^{t+1} or no match at all. In other words, each row and column of the A^g can only be a one-hot vector (*i.e.*, a vector with 1 in a single entry and 0 in all other entries) or an all-zero vector. This motivates our second loss for the affinity matrix. For all rows and columns that have a one-hot vector in A^g, we apply the cross entropy loss L_{ce} to the corresponding rows and columns of A. As an example shown below, the column $A_{\cdot j}^{g}$ in ground truth affinity matrix is a one-hot vector and the loss L_{ce} for the jth column is defined as:

$$L_{\text{ce}} = \frac{-1}{M} \sum_{i}^{M} A_{ij}^{g} \log\left(\frac{\exp A_{ij}}{\sum_{i}^{M} \exp A_{ij}}\right). \quad (9)$$

We can now summarize the affinity loss L_{aff} as below:

$$L_{\text{aff}} = L_{\text{bce}} + L_{\text{ce}}. \quad (10)$$

3.4. Tracking Management

Although the discriminative feature learning can help resolve confusion for matching, it is still possible that a tracked object is matched to a false positive detection. Also, there might be the case where a tracked object still exists but cannot find a match due to missing detection (*i.e.*, false negative). To avoid such cases, a tracking management module that controls the birth and death of the objects is necessary in MOT to reduce the false positives and false negatives. We follow [4, 45] and maintain a death count and a birth count for each object. If a new object is able to find the match in Bir_{min} frames continuously, we will then assign an ID to this object and add it to the set of tracked objects O. However, if this object stops finding the match before being assigned an ID, we will reset the birth count to zero. On the other hand, if a tracked object cannot find the matched detection in Age_{max} frames, we believe that this object has disappeared and will delete it from the set of tracked objects O. However, if this tracked object can still find a match before being deleted, we believe that the object still exists and will reset the death count to zero. In the first frame of the video, we initialize the tracked objects O as an empty set.

4. Experiments

4.1. Settings

Dataset. To demonstrate the strength of our joint 2D-3D feature extractor, we evaluate our network on KITTI [11] and nuScenes [6] datasets, which provide both 2D (images and 2D boxes) and 3D data (LiDAR point cloud and 3D boxes). For KITTI, same as most prior works, we report results on the car subset for comparison. For nuScenes, we evaluate on all categories and the final performance is the mean over all categories. As the focus of this paper is 3D MOT, we report and compare 3D MOT performance on the KITTI and nuScenes datasets. Since KITTI has an official 2D MOT benchmark, we also report 2D MOT results on KITTI for reference, which is achieved by projecting our 3D MOT results to the image space.

Evaluation Metrics. We use standard CLEAR metrics [3] (including MOTA, MOTP, IDS, FRAG and FPS) and also the new sAMOTA, AMOTA and AMOTP metrics proposed in [45] for 3D MOT and 2D MOT evaluation. For 3D MOT evaluation, we use the evaluation tool proposed by [45]. As KITTI and nuScenes datasets do not release the ground truth of test set to users, we use the validation set for 3D MOT evaluation. For KITTI 2D MOT evaluation, we use the official KITTI 2D MOT evaluation tool [11]. In terms of the training, validation and testing split, we use the official one on nuScenes. As KITTI does not have an official split, we use the one proposed by [29].

Baselines. For 3D MOT, we compare with recent open-source 3D MOT systems such as FANTrack [2], mmMOT [56] and AB3DMOT [45], which also use the 3D LiDAR data (either directly used in 3D MOT or indirectly used in order to obtain the 3D detections for 3D MOT) for fair comparison with our 3D MOT method. For 2D MOT, we compare with state-of-the-art published 2D MOT systems on the KITTI MOT leaderboard.

4.2. Implementation Details

3D Object Detection. For fair comparison in KITTI, we use the same 3D detections from PointRCNN [32] for all 3D MOT methods (including our method and the baselines) that require 3D detections as inputs. For 3D MOT methods that also require 2D detections, *e.g.*, the 2D feature extraction branch in our method, we use the 2D projection of 3D detections from [32]. For nuScenes, the same rule also applies except that the 3D detections obtained by [32] is replaced with the 3D detections obtained by [58]. For data augmentation, we perturb the ground truth box during training with a ratio of 0.1 with respect to the size of the box.

Joint Feature Extractor. We use the feature with same dimensionality of 64 for all four branches. For 3D appearance branch, we use the PointNet with six 1D Convolutional layers that maps the input point cloud with size of P (number of points) $\times$ 4 (x, y, z, reflectance) to P $\times$ 64 (4$\Rightarrow$16$\Rightarrow$32 $\Rightarrow$64$\Rightarrow$128$\Rightarrow$256$\Rightarrow$64). Then, a max pooling operation is applied along the axis of P to obtain the 3D appearance feature with the dimensionality of 64. For 2D appearance branch, we resize the cropped image patch for each object to 56×84 and use the ResNet34 to extract the 2D appearance

Table 1. Quantitative comparison on KITTI-Car val set. The evaluation is conducted in **3D** space using [45] 3D MOT evaluation tool.

Method	Input Data	**sAMOTA** (%) ↑	AMOTA (%) ↑	AMOTP (%) ↑	MOTA (%) ↑	MOTP (%) ↑	IDS ↓	FRAG ↓
mmMOT [56] (ICCV'19)	2D + 3D	70.61	33.08	72.45	74.07	78.16	10	125
FANTrack [2] (IV'19)	2D + 3D	82.97	40.03	75.01	74.30	75.24	35	202
AB3DMOT[45] (arXiv'19)	3D	91.78	44.26	77.41	83.35	78.43	**0**	15
Ours	2D + 3D	**93.68**	**45.27**	**78.10**	**84.70**	**79.03**	**0**	**10**

Table 2. Quantitative comparison on KITTI-Car test set. The evaluation is conducted in **2D** space using KITTI 2D MOT evaluation tool.

Method	Input Data	**MOTA** (%) ↑	MOTP (%) ↑	MT (%) ↑	ML (%) ↓	IDS ↓	FRAG ↓	FPS ↑
CIWT [24] (ICRA'17)	2D	75.39	79.25	49.85	10.31	165	660	2.8
FANTrack [2] (IV'19)	2D + 3D	77.72	82.32	62.61	8.76	150	812	25.0 (GPU)
AB3DMOT[45] (arXiv'19)	3D	83.84	85.24	66.92	11.38	**9**	**224**	**214.7**
BeyondPixels [30] (ICRA'18)	2D	84.24	85.73	73.23	2.77	468	944	3.33
3DT [16] (ICCV'19)	2D	84.52	**85.64**	73.38	**2.77**	377	847	33.3 (GPU)
mmMOT [56] (ICCV'19)	2D + 3D	84.77	85.21	73.23	**2.77**	284	753	4.8 (GPU)
MASS [17] (IEEE Access'19)	2D	**85.04**	85.53	**74.31**	**2.77**	301	744	100.0
Ours	2D + 3D	80.40	85.05	70.77	11.08	113	265	5.2 (GPU)
Ours + 2D detections from [27]	2D	82.24	84.05	64.92	6.00	142	416	5.1 (GPU)

Table 3. Quantitative comparison on nuScenes validation set. The evaluation is conducted in **3D** space with 3D MOT evaluation tool.

Method	**sAMOTA** (%) ↑	AMOTA (%) ↑	AMOTP (%) ↑	MOTA (%) ↑
FANTrack [2]	19.64	2.36	22.92	18.60
mmMOT [56]	23.93	2.11	21.28	19.82
AB3DMOT[45]	27.90	4.93	23.89	21.46
Ours	**29.84**	**6.21**	**24.02**	**23.53**

Table 4. **Effect of Joint Feature Extractor**. Results are evaluated on KITTI-Car val set using **3D** MOT evaluation tool. Appearance and motion features are denoted as "A" and "M" respectively.

Feature Extractor	**sAMOTA** (%) ↑	AMOTA (%) ↑	AMOTP (%) ↑	MOTA (%) ↑
2D A	88.31	41.62	76.22	79.42
2D M	64.24	23.95	61.13	54.88
3D A	88.27	41.55	76.29	77.38
3D M	88.57	41.62	76.22	81.84
2D+3D A	89.39	42.55	76.24	83.02
2D+3D M	91.75	44.75	78.05	84.54
2D M+A	90.56	44.39	**78.20**	83.15
3D M+A	91.30	44.31	78.16	84.06
2D+3D M+A (**Ours**)	**93.68**	**45.27**	78.10	**84.70**

feature. For 2D and 3D motion branches, we use a two-layer LSTMs with a hidden size of 64 and number of past frames T=5 for tracked objects. For tracked objects which only have associated detections in past R ($< T$) frames, we repeat the earliest detection T-R times so that the objects can have T frames of detections. For detected objects, we employ a two-layer MLP (4⇒16⇒64 in 2D motion branch, 7⇒32 ⇒64 in 3D motion branch).

Feature Fusion and Ensemble Training Paradigm. In feature fusion, if a branch is dropped, we fill in zeros into the feature corresponding to the dropped branch before fusion so that the feature fusion module is compatible with the ensemble training paradigm. For drop ratio, we use $r = 0.5$.

Graph Neural Network and Miscellaneous. We use the Dist^{3D}_{max}=5 and Dist^{2D}_{max}=200 in our sparse graph construction. We use three GNN layers (*i.e.*, K=4) with each layer having feature with same dimensionality. For example, when we use "concatenate" as the fusion operator, we will have node feature with dimensionality of 256 in all layers of GNN. For edge regression, we use a two-layer MLP with hidden feature dimension of 256⇒64⇒1. For batch triplet loss, we use the margin α=10. For the tracking management, we use Age_{max}=4 and Bir_{min}=10.

4.3. Experimental Results

Results on KITTI. We summarize the 3D MOT and 2D MOT results on KITTI-Car dataset in Table 1 and 2. For 3D MOT evaluation in Table 1, our proposed method consistently outperforms other modern 3D MOT systems in all metrics. For 2D MOT evaluation in Table 2, our network is behind prior work and only achieves 80.40 2D MOTA. One possible reason is that the 2D projection of 3D detection results we use has lower precision and recall than a state-of-the-art 2D detector [27] used in prior work. For fair comparison, we simply replace the input 2D detections with [27] while keeping all hyper-parameters fixed and show the results in the last row of Table 2. As a result, the MOTA of our proposed method is improved about 2% without bells and whistles. We argue that it is highly possible that our proposed method can achieve higher performance on 2D MOT after hyper-parameter searching based on 2D MOT evaluation. Currently, all ablation analysis is performed on 3D MOT evaluation, meaning that the hyper-parameters of our method are only tuned for 3D MOT and not for 2D MOT.

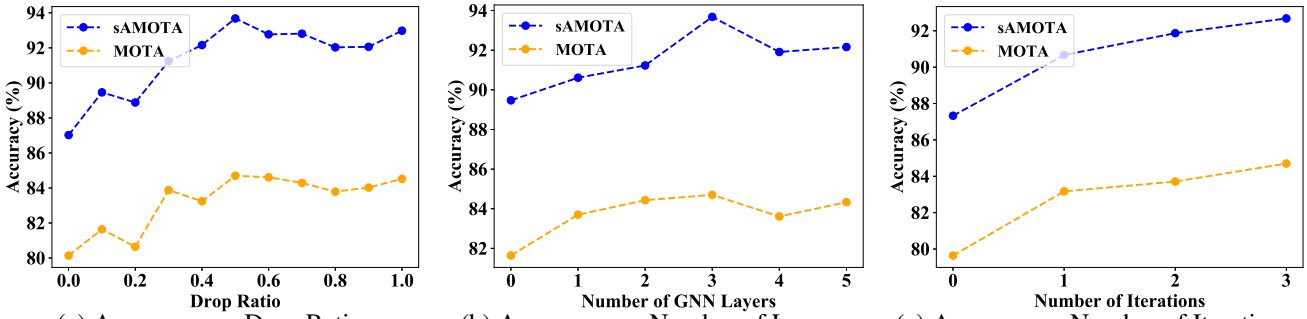
Results on nuScenes. In Table 3, our method achieves the state-of-the-art 3D MOT performance on nuScenes. As the 3D detection performance is not yet mature on nuScenes compared to KITTI, 3D MOT performance on nuScenes is consistently lower than on KITTI.

Inference Time. Our network runs at a rate of 5.2 FPS on the KITTI test set with a single 1080Ti GPU.

Qualitative Comparison. We show our qualitative results on two sequences of the KITTI test set in Figure 4.

4.4. Ablation Study

We conduct the ablation study on KITTI-Car validation set using 3D MOT evaluation tool proposed by [45].



(a) Accuracy v.s. Drop Ratio (b) Accuracy v.s. Number of Layers (c) Accuracy vs Number of Iterations

Figure 3. (a) **Effect of Ensemble Training Paradigm.** We vary the drop ratio r from 0 to 1 with an interval of 0.1. Results suggest that r=0.5 is the best. (b) **Effect of Number of GNN Layers.** We increase the number of layers from 0 (*i.e.*, deactivate the GNN) to 5 and use the output from the last layer of GNN for evaluation. The highest accuracy is obtained when using three layers. (c) **Effect of Feature Interaction.** For our final network with three GNN layers, we evaluate the output of layer 0 (*i.e.*, deactivate the GNN) to layer 3. Results suggest that the output from the last layer of the GNN achieve the highest performance.

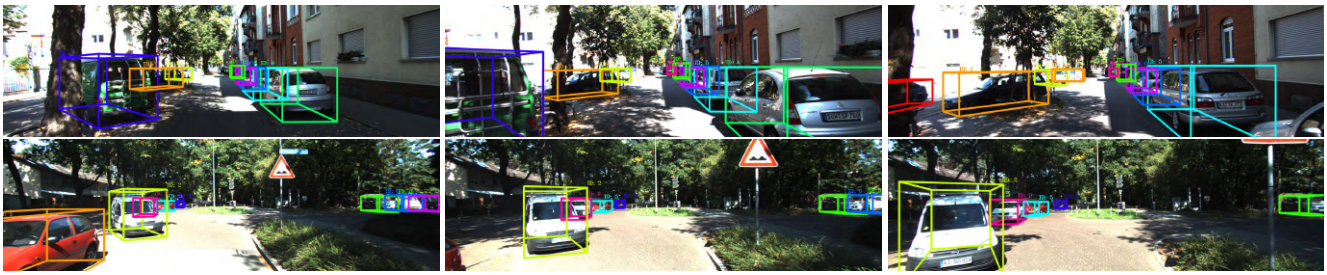


Figure 4. Qualitative results of our method on sequence 0 (top row) and 3 (bottom row) of the KITTI test set.

Table 5. **Effect of Feature Fusion Operators.** Results are evaluated on KITTI-Car val set using **3D** MOT evaluation tool.

Fusion	**sAMOTA** (%) ↑	AMOTA (%) ↑	AMOTP (%) ↑	MOTA (%) ↑
Add	89.98	42.97	75.96	82.55
Concatenate (**Ours**)	**93.68**	**45.27**	**78.10**	**84.70**

Table 6. **Effect of Edge Regression Modules**. Results are evaluated on KITTI-Car val set using **3D** MOT evaluation tool.

Edge Regression	**sAMOTA** (%) ↑	AMOTA (%) ↑	AMOTP (%) ↑	MOTA (%) ↑
Negative L2 Distance	82.26	41.38	72.42	70.71
Cosine Similarity	87.07	43.18	72.17	75.46
MLP (**Ours**)	**93.68**	**45.27**	**78.10**	**84.70**

Table 7. **Effect of Node Aggregation Rules**. Results are evaluated on KITTI-Car val set using **3D** MOT evaluation tool.

Node Aggregation	**sAMOTA** (%) ↑	AMOTA (%) ↑	AMOTP (%) ↑	MOTA (%) ↑
Type 1	75.61	32.84	65.81	67.43
Type 2 (SAGEConv [13])	87.81	41.06	76.29	77.22
Type 2 (GCN [18])	89.78	43.37	78.06	80.67
Type 2 (GraphConv [23])	91.15	44.78	77.93	82.31
Type 2 (GATConv [38])	91.66	44.57	77.99	82.37
Type 2 (AGNNConv [36])	91.88	44.95	78.00	84.32
Type 3 (EdgeConv [43])	92.17	44.65	77.98	83.73
Type 4 (**Ours**)	**93.68**	**45.27**	**78.10**	**84.70**

Effect of Joint Feature Extractor. In Table 4, we evaluate the effect of each individual feature extractor and the combination of them. We show that combining features from different modalities improves performance, suggesting that different features are complementary to others.

Effect of Feature Fusion Operators. In Table 5, we show that using "concatenate" is better than "add" for fusion.

Effect of Edge Regression Modules. In Table 6, the two-layer MLP used in our final network achieves better performance than the conventional similarity metrics.

Effect of Node Aggregation Rules. In Table 7, we show that type 4 rule performs the best. Also, for different GNNs with type 2 rule, performance varies significantly.

Effect of Ensemble Training Paradigm. In Figure 3 (a), we observe that using ensemble training paradigm significantly improves the performance with r=0.5 being the best.

Effect of Number of GNN Layers. In Figure 3 (b), increasing the number of GNN layers improves the performance with three GNN layers being the best. We did not experiment with GNN larger than five layers as the GNN tends to overfit when it becomes very deep.

Effect of Feature Interaction. In Figure 3 (c), we show that feature interaction in GNNs is effective as the performance increases when we use the output from a later layer.

5. Conclusion

We propose a 3D MOT method with a novel joint 2D-3D feature extractor and a novel feature interaction mechanism achieved by GNNs in order to improve the discriminative feature learning in MOT. Through extensive experiments, we demonstrate the effectiveness of each individual module in our proposed method, establishing state-of-the-art 3D MOT performance on the KITTI and nuScenes datasets.

References

[1] Hermans Alexander, Lucas Beyer, and Bastian Leibe. In Defense of the Triplet Loss for Person Re-Identification. *CVPRW*, 2019. 5

[2] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. FANTrack: 3D Multi-Object Tracking with Feature Association Network. *IV*, 2019. 2, 6, 7

[3] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *Journal on Image and Video Processing*, 2008. 6

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple Online and Realtime Tracking. *ICIP*, 2016. 1, 2, 6

[5] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-Speed Tracking-by-Detection without Using Image Information. *ICAVSS*, 2017. 2

[6] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, and Qiang Xu. nuScenes: A Multimodal Dataset for Autonomous Driving. *CVPR*, 2020. 6

[7] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Shuicheng Yan, Jiashi Feng, and Yannis Kalantidis. Graph-Based Global Reasoning Networks. *CVPR*, 2019. 3

[8] Ian Cherabier, Christian Hane, Martin R Oswald, and Marc Pollefeys. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, 2017. 3

[9] Davi Frossard and Raquel Urtasun. End-to-End Learning of Multi-Sensor 3D Tracking by Detection. *ICRA*, 2018. 2

[10] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. Graph Convolutional Tracking. *CVPR*, 2019. 3

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite. *CVPR*, 2012. 6

[12] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A New Model for Learning in Graph Domains. *IJCNN*, 2005. 3

[13] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *NIPS*, 2017. 8

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016. 4

[15] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation Networks for Object Detection. *CVPR*, 2018. 3

[16] 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. *ICCV*, 2019. 7

[17] Hasith Karunasekera, Han Wang, and Handuo Zhang. Multiple Object Tracking with Attention to Appearance, Structure, Motion and Size. *IEEE Access*, 2019. 7

[18] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *ICLR*, 2017. 8

[19] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition. *CVPR*, 2019. 3

[20] Weiqiang Li, Jiatong Mu, and Guizhong Liu. Multiple Object Tracking with Motion and Appearance Cues. *arXiv:1909.00318*, 2019. 2

[21] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. *CVPR*, 2018. 1

[22] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. *AAAI*, 2019. 5

[23] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. *AAAI*, 2019. 8

[24] Aljosa Osep, Wolfgang Mehner, Markus Mathias, and Bastian Leibe. Combined Image- and World-Space Tracking in Traffic Scenes. *ICRA*, 2017. 7

[25] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects. *CVPR*, 2011. 2

[26] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NIPS*, 2017. 3

[27] 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. *CVPR*, 2017. 7

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *NIPS*, 2015. 1

[29] Samuel Scheidegger, Joachim Benjaminsson, Emil Rosenberg, Amrit Krishnan, and Karl Granstr. Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering. *IV*, 2018. 6

[30] Sarthak Sharma, Junaid Ahmed Ansari, J. Krishna Murthy, and K. Madhava Krishna. Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking. *ICRA*, 2018. 7

[31] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-Based Action Recognition with Directed Graph Neural Networks. *CVPR*, 2019. 3

[32] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *CVPR*, 2019. 1, 3, 6

[33] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*, 2015. 4

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, 2014. 2, 4

[35] Shijie Sun, Naveed Akhtar, Huansheng Song, Ajmal Mian, and Mubarak Shah. Deep Affinity Network for Multiple Object Tracking. *TPAMI*, 2017. 2

[36] Kiran K. Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-Based Graph Neural Network for Semi-Supervised Learning. *arXiv:1803.03735*, 2018. 8

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018. 5

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018. 8

[39] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. MOTS: Multi-Object Tracking and Segmentation. *CVPR*, 2019. 5

[40] H W Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 1955. 1, 2

[41] Sen Wang, Daoyuan Jia, and Xinshuo Weng. Deep Reinforcement Learning for Autonomous Driving. *arXiv:1811.11329*, 2018. 1

[42] Xiaolong Wang and Abhinav Gupta. Videos as Space-Time Region Graphs. *ECCV*, 2018. 3

[43] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 2018. 8

[44] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 2019. 3, 5

[45] Xinshuo Weng and Kris Kitani. A Baseline for 3D Multi-Object Tracking. *arXiv:1907.03961*, 2019. 1, 2, 6, 7

[46] Xinshuo Weng and Kris Kitani. Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud. *ICCVW*, 2019. 1, 3

[47] Xinshuo Weng, Jianren Wang, Sergey Levine, Kris Kitani, and Nicholas Rhinehart. Unsupervised Sequence Forecasting of 100,000 Points for Unsupervised Trajectory Forecasting. *arXiv:2003.08376*, 2020. 1

[48] Xinshuo Weng, Shangxuan Wu, Fares Beainy, and Kris Kitani. Rotational Rectification Network: Enabling Pedestrian Detection for Mobile Vision. *WACV*, 2018. 1

[49] Xinshuo Weng, Ye Yuan, and Kris Kitani. Joint 3D Tracking and Forecasting with Graph Neural Network and Diversity Sampling. *arXiv:2003.07847*, 2020. 1

[50] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple Online and Realtime Tracking with a Deep Association Metric. *ICIP*, 2017. 2

[51] Jinlin Wu, Yang Yang, Hao Liu, Shengciao Liao, Zhen Lei, and Stan Z Li. Unsupervised Graph Association for Person Re-identification. *ICCV*, 2019. 3

[52] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-Temporal Relation Networks for Multi-Object Tracking. *ICCV*, 2019. 3

[53] Jimuyang Zhang, Sanping Zhou, Jinjun Wang, and Dong Huang. Frame-Wise Motion and Appearance for Real-time Multiple Object Tracking. *BMVC*, 2019. 2

[54] Li Zhang, Xiangtai Li, Anurag Arnab, Kuiyuan Yang, Yunhai Tong, and Philip H. S. Torr. Dual Graph Convolutional Network for Semantic Segmentation. *BMVC*, 2019. 3

[55] Li Zhang, Yuan Li, and Ramakant Nevatia. Global Data Association for Multi-Object Tracking Using Network Flows. *CVPR*, 2008. 2

[56] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust Multi-Modality Multi-Object Tracking. *ICCV*, 2019. 2, 6, 7

[57] Rui Zhao, Kang Wang, Hui Su, and Qiang Ji. Bayesian Graph Convolution LSTM for Skeleton Based Action Recognition. *ICCV*, 2019. 3

[58] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-Balanced Grouping and Sampling for Point Cloud 3D Object Detection. *CVPR*, 2019. 6