DiTNet: End-to-End 3D Object Detection and Track ID Assignment in Spatio-temporal World

Sukai Wang, Peide Cai, Lujia Wang, Ming Liu

***Abstract*—End-to-end 3D object detection and tracking based on point clouds is receiving more and more attention in many robotics applications, such as autonomous driving. Compared with 2D images, 3D point clouds do not have enough texture information for data association. Thus, we propose an end-to-end point cloud-based network, DiTNet, to directly assign a track ID to each object across the whole sequence, without the data association step. DiTNet is made location-invariant by using relative location and embeddings to learn each object's spatial and temporal features in the Spatio-temporal world. The features from the detection module helps to improve the tracking performance, and the tracking module with final trajectories also helps to refine the detection results. We train and evaluate our network on the CARLA simulation environment and KITTI dataset. Our approach achieves competitive performance over the state-of-the-art methods on the KITTI benchmark.**

I. INTRODUCTION

3D multiple object detection and tracking is gradually receiving more and more attention in autonomous driving. For driving in a dynamic environment, the current state-of-the-art approaches have achieved competitive results to precisely detect moving objects and their past trajectories. However, most of them consider the detection and tracking as two separate problems, or weakly use the detection features in data association while tracking. The data association step is always classified out of the network because it is hard to be differentiable. This raises a question: Can we give unmanned vehicles the ability to efficiently detect moving objects with their track identity directly?

For the 3D object detection task, the state-of-the-art detection techniques which focus on 3D or 2D object detection obtain impressive results on various benchmarks. This is fundamental to high performance for all tracking-by-detection algorithms. However, the balance between false-negatives and false-positives remains crucial in tracking tasks. The detection score threshold (DST) setting for choosing the object bounding boxes at the final stage of a detector controls the detector's performance. A higher DST means more false-negative detections, and a lower DST means more false-positives. Too many false-positive detections could lead to a dramatic decrease in performance on the data association task due to the interference of redundant error information; however, more false-positives are more acceptable than more false-negatives because of the autopilot safety. This is the reason that the performance of the detection networks is the critical factor for tracking-by-detection trackers. In this paper, we propose a tracking-with-detection method that can examine all possible regions of proposal (ROIs) after detection and then directly assigns a unique track identity to them. The detection score threshold is discarded and replaced by the track identity score threshold.

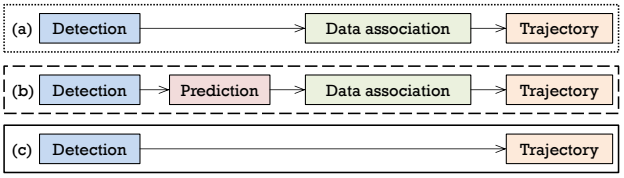

Fig. 1: Three paradigms for 3D object detection and tracking. Traditional approaches: (a) uses the detection boxes in data association by the Hungarian algorithm or IoU tracker; (b) predicts movement of each object in adjacent frames, which is helpful for distance association. Ours: (c) directly predicts the unordered track IDs from ROIs without data association.

Compared with 2D images, 3D point clouds data can provide explicit geometry information for generating accurate 3D locations and bounding box shapes. However, there are still differences between 2D and 3D input which lead to crucial problems in detection and tracking. One is that the texture information in a 3D point cloud is much less than in a 2D image because of the sparse points form. The other is that the object features of the same object at different times in a 3D point cloud are not location-invariant as in a 2D image. For example, the same car in adjacent 2D image frames will have the same color, shape, and paint patterns on the car body. But in the LiDAR view, all of the cars in the same relative position have almost the same type of point cloud shape and similar reflectivity, though the number of points will change significantly when the LiDAR's distance or perspective changes. This is the reason why it is hard to learn the appearance features from point clouds as well as from image-based methods. The solution to this problem of most existing 3D point cloud based-trackers [1] [2] is to use the distance matrix or intersection over union (IoU) matrix to associate the same object in adjacent frames. In this paper, we propose a novel feature grouping method to learn the

Manuscript received December 14, 2020; Revised January 12, 2021; Accepted February 7, 2021. This paper was recommended for publication by Editor Youngjin Choi upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China, under grant No. U1713211, Collaborative Research Fund by Research Grants Council Hong Kong, under Project No. C4063-18G, and HKUST-SJTU Joint Research Collaboration Fund, under project SJTU20EG03, awarded to Prof. Ming Liu. And it was supported by the Guangdong Science and Technology Plan Guangdong-Hong Kong Cooperation Innovation Platform (Grant Number 2018B050502009) awarded to Lujia Wang. *(Lujia Wang is corresponding author)*

Sukai Wang, Peide Cai, andp Ming Liu are with The Hong Kong University of Science and Technology, Hong Kong SAR, China (email: swangcy@connect.ust.hk; pcaiaa@connect.ust.hk; eelium@ust.hk)

Lujia Wang is with Cloud Computing Lab of Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China (email: lj.wang1@siat.ac.cn)

Digital Object Identifier (DOI): see top of this page.

association relationship of the proposals in a batch of point clouds as input, to replace the appearance features by merging the neighbor location and detection embeddings.

For multiple object tracking (MOT) tasks, data association, which is the process of associating the same object in different frames, is one of the key technologies. In contrast to the methods described in Fig. 1(a) [3], [4], [1], and (b) [2], we want to propose an end-to-end network (Fig. 1(c)), which can get straight to the track ID of each object without the data association step.

In this paper, we propose an end-to-end tracking with detection neural network, DiTNet, which directly outputs the 3D bounding box with the track ID of each object in the Spatio-temporal world. DiTNet takes as input the raw point cloud batch (numbers of adjacent point cloud frames), and outputs each object's detection bounding box in the frames and track ID in the batch. There are two critical issues in the network. One is that the number of trajectories is not constant, which means we could have 0 to L trajectories. The other is that the order of trajectories is not required to be fixed, which means the track identity is unordered. The same object only needs to be classified into one trajectory, no matter what the track ID is. These two issues are especially prominent during network training. In our approach, we set a max number of trajectories to solve the first issue, and apply the Hungarian algorithm [5] to efficiently assign the predicted trajectories with label trajectories for the second. Note that the Hungarian algorithm is only used while training, and it is not necessary in the inference processing. The contributions of this work are listed as follows:

- We propose a novel detection arrangement method in the Spatio-temporal map to learn the correlation relationship of objects in a batch of frames as input.
- We propose an instance-based network, which uses novel grouping and feature extraction layers to learn the correlation location information and instance features in the spatial and temporal feature modules.
- We propose a real end-to-end detection network to output the objects' boxes and track IDs directly, without the usage of non-maximum suppression (NMS) and the Hungarian algorithm.
- The sequence of the point cloud input can be used to improve the performance of detection and tracking. Our approach achieves competitive performance over the state-of-the-art methods on the KITTI benchmark.

The remainder of this letter is organized as follows. Section II reviews the related work. Section III describes the framework and components of DiTNet in detail. Section IV presents the experimental results and discussions. Conclusions and future work are given in the last section.

II. RELATED WORK

A. Time-Related Networks

In time series-related tasks, different kinds of network architectures focus on the importance of temporal relations in a batch of data, such as Recently Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM). Time-related networks are used widely in forecasting tasks like trajectory prediction [6] [7].

Social LSTM [7] was proposed to learn the movements of the general human and predict the trajectories of multiple pedestrians, and sequential data are used to provide short-term and long-term information. ReSeg [8] was proposed for semantic segmentation in different scenarios as an RNN-based network. This kind of usage is more like an alternative to Convolutional Neural Networks (CNNs), which shows that RNN models have the ability to learn the dependencies between time connection data. Wang et al. [9] proposed a Bayesian and conditional random field-based framework in the Spatio-temporal map to solve the data association problem between frames. A Spatio-temporal LSTM [10] which uses sequential data as input was introduced to save the long-term context information. Structural-RNN [11] combines the high-level Spatio-temporal graphs with RNNs, which reconstructs the factors in Spatio-temporal graph to achieve a rich RNN mixture.

Though the features of the same object are always changing over time in 3D point clouds, the related location information over the whole sequence can also help to rearrange the track IDs of the detections. Thus, in our network, the long-term temporal and spatial features are consolidated to finish the data association end-to-end.

B. 3D Object Detection and Tracking

In existing 3D object detection solutions, 3D point cloud-based networks can predict more accurate locations and the bounding boxes of objects compared to 2D images. Networks like those in [12] and [13] use PointNet++ [14] as the 3D backbone to learn the 3D features from dense to sparse. And other networks [15] still preserve the image-based 2D CNN to compose the large-scale features by scattering the learned features from pillars or voxels to the Bird's-eye-view map.

For multi-object tracking problems, most of the current approaches focus on the data association problem. The Kalman Filter [3] [4] and Gaussian [16] processes are all famous as the conventional methods for position prediction and association matrix generation, and convolutional Siamese networks [17] [18] are also used widely for association similarity computation. Once the cost matrix of all objects is obtained, the data association problem can also be seen as an optimization problem. Thus, the Hungarian algorithm [3] can be applied to solve it. Matching Net [19] [20] can find the corresponding pairs in adjacent frames and generate the trajectories. Multi-scan methods, which take as input a sequence of data rather than two single adjacent frames, have proposed to use a neural network to learn the optimization solutions to find the optimal trajectory assignment [21] [19]. Meanwhile single-scan methods only take as input two adjacent frames and associate the objects in these two frames [22] [2], by predicting the association matrix or the movement of each object to find the best association relationship. AB3DMOT [1] and DEEP SORT [4], which are two baseline tracking-by-detection approaches, use rich texture features of each object as attributes of the objects and the Kalman Filter and Hungarian algorithm

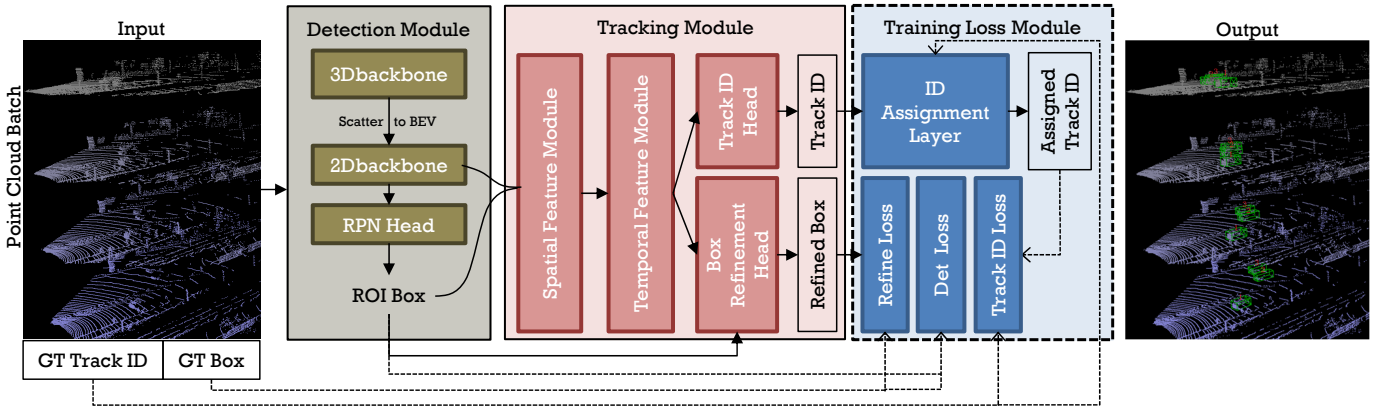

Fig. 2: The overall network architecture. The detection module takes the continuous point cloud batch as input and generates the ROIs with high-level features. Then the tracking module uses them to get the refined boxes with the track ID. The training loss module only works during the training to calculate the ground-truth ID assignment matrix and the training loss.

as post-process association methods to finish the real-time tracking.

III. OUR APPROACH

A. The Overall Architecture

The overall architecture of our DiTNet is shown in Fig. 2. The network is composed of three main parts: the detection module, Spatio-temporal tracking module, and training loss module. The detection module outputs the positions and bounding boxes of the ROIs from the raw point cloud data batch. The embeddings of the ROIs and the boxes are combined as the detections and passed to the tracking module (Sec III-B). The tracking module can output the unique track ID of each object in a batch of adjacent frames, and then an easy post-processing algorithm is used to connect the track ID in the whole sequence. The architecture of Spatio-temporal tracking module is inspired by the arrangement of detections in the Spatio-temporal world (Sec III-C). The training loss module in the blue block in Fig. 2 is time-consuming but only works while training, which means that, although the network needs a longer time for training, it is efficient enough for real-time inference (Sec III-D).

B. Detection Module

The detection module is proposed to generate the 3D ROIs of each possible object with its learnable features. As introduced in [23], there are four main modules in current state-of-the-art detectors: the 3D backbone, 2D backbone, ROI head, and dense detection head. Our detection module makes reference to PointPillars [15], which consists of three submodules: a 3D pillar feature net, 2D encoder-decoder backbone, and RPN detection head. Firstly, we discretize the raw 3D points into evenly spaced grids in the ground plane, which are called pillars. Each pillar is fed separately into the voxel feature encoding (VFE) network to extract the 3D embeddings inside every pillar, which can be scattered back to the 2D BEV pseudo-image using their position index. A 2D CNN backbone is then applied to generate multi-scale high-level features from the pseudo-image. We use C to denote the output features, which is concatenated from different upsampling layers in the 2D multi-scale backbone. Finally, the detection head uses two convolutional layers which have a 1×1 kernel to predicts the 3D bounding boxes with their class for objects in each pillar position. Axis-aligned NMS is performed at inference time.

The detection module can be replaced by any other 3D object detector, as long as its output includes the detection results with their high-level features. The critical consideration in choosing the best detector backbone is to balance speed and detection performance. Whether the input of the detector is a 3D point cloud or 2D image is not the key factor in our tracking task, because if the input is raw point clouds which have minor texture information of the objects, the Spatio-temporal module will learn the relative position information of all neighboring objects to find the correct corresponding relationship. How to choose the detection score threshold of NMS is one of the crucial problems of most tracking-by-detection methods, because the tracking task is more sensitive to false-positive detections than the detection task. Other methods choose to use a higher score threshold to filter the unreliable detections, but our proposed detection module uses a small threshold to ensure as many detections as possible will be included for the next tracking process. The high-level features of the detections then contribute to the following tracking module.

C. Tracking Module

From the detection module, we can get: the object detections' absolute global location $\{L_i^t = (x_i^t, y_i^t, z_i^t)\}_{i=1}^{N_t}$, where x, y, and z are the 3D global position of the objects at time t, and N_t means there are at most N_t objects at each time; the objects' detection score $\{S_i^t\}_{i=1}^{N_t}$, where $S \in [0,1]$, $S \in \mathbb{R}$; and the middle embeddings of each object from the 2D backbone in the detection module $\{C_i^t\}_{i=1}^{N_t}$, where each feature has K-dim channels. We use a multi-scan point cloud batch as input, which is obtained by using a sliding window to crop batches of data. Let T denote the window size, and we can obtain the object detections in this time window: $\{D_i^t\}_{i\in[1,N_t],t\in[1,T]}$. The detections are rearranged in the Spatio-temporal world with their global xyz location in a single frame, and also with their moment of appearance. A t axis is proposed in Spatio-temporal world to store the time information.

The detections from the detection module have four main properties: (1) The detections are unordered, which means they can be arranged in different orders by their positions and features. In other words, our network should be able to learn the order-invariant features of all detections. (2) All of the various detections in a single frame must have their own unique track IDs. (3) The detections are not isolated within the environment; in other words, the neighborhoods of each detection can provide feature information while learning. (4) The tracking results should have invariance under trajectory transformation. For example, theoretically, the tracking results should be the same when the trajectories appear in different places, without considering the environment's influence, as long as all detections have the same relative position of all pairs in the whole sequence. Another example is that the time interval Δt in a sequence is not the key factor because $v_i^t = \frac{|L_i^t - L_i^{t+1}|}{\Delta t}$, where L_i^t and $L_i^{t+1} \in \mathbb{R}^3$ are the absolute location, is the speed of objects. And the speed of different objects or the same objects at different times is not a constant in a real environment.

To fit these special properties in the tracking task, we discard all absolute global locations or features in the Spatio-temporal feature module, while using the relative location information and relative features to represent the relative relationship of objects to objects. d_k in Fig. 3 means the relative features: for target detection $D \in \mathbb{R}^K$, grouped neighbor detections $D_g = \{D_{g_i}\}, D_g \in \mathbb{R}^K, i \in [1, N_n]$, $d_k = D - D_g = \{D - D_{g_i}\}$, and $i \in [1, N_n]$.

The tracking module consists of three main parts: a spatial feature extraction module, temporal feature extraction module, and track head module.

1) Spatial feature extraction module: The spatial feature extraction module consists of a local feature extraction module and a frame global feature extraction module, as shown in Fig. 3 (c). The local feature module learns the relative features from the nearest S neighbor detections, and the frame global module learns the relative features from all detections in the same time frame.

Take $D_{i_0}^{t_0}$ as the target detection for example. In the local feature extraction module, the grouping layer chooses the nearest objects $G = \{D_i^{t_0}\}$, where $i \neq i_0$, the number of grouped objects is less than the maximum grouping number N_s, and $||L_i^{t_0} - L_{i_0}^{t_0}||_2 \leq dL_{max}$.

Given an unordered detection set $\{D_1, D_2, ..., D_{N_t}\}$ with their locations and embeddings $D_i = (L_i, C_i)$, where $L_i \in \mathbb{R}^d, C_i \in \mathbb{R}^K$, the local feature module defines a set function $f : \mathscr{X} \to \mathbb{R}$:

$$f(\{D_1, D_2, ..., D_{N_t}\}) = \underset{i=1,...,N_t}{\text{MAX}} (\gamma(W \cdot (G - D_i))), \quad (1)$$

where γ is multi-layer perceptron (MLP) networks, and W is the weight of each relative embeddings. W is calculated by the distance between each grouped detection and the target detection and the detection score of the target detection. For normalization, the distance weight is calculated by:

$$W_{ij} = W_s \cdot W_{dL} = S_{ij} \cdot exp(-\alpha \cdot ||L_i - G_j||),$$

where α is a parameter which is set to 10 in our experiment.

In the global feature module, only the grouping method is different from the local feature module. It will choose all detections in the same time frame, except the target detection itself, as the grouped detections, and the MLPs are used for the weighted relative embeddings learning.

2) Temporal feature extraction module: Compared with the spatial feature extraction module, the temporal feature module mainly learns the relative spatial features in the different time frames.

Take Fig. 3 (a) as an example. The object with track ID 1 is the most leftward detection in all time frames, object 2 is in the middle, and object 3 is the most rightward detection. Thus the spatial features from the spatial feature extraction module will be clustered into three groups, for objects 1, 2, and 3 respectively. We can use the temporal feature module to find the objects in different frames with the most similar embeddings. And in Fig. 3 (b), we can find that there are two false-positive detections in time frame 2 and time frame 3. The temporal feature module is proposed to learn the relative location displacement of each object from others in different time frames. Thus object 4 will be tracked successfully due to the smooth trajectory.

As shown in Fig. 3 (d), we predict the temporal features frame by frame. All detections learn the embeddings from the detections in different time frames. For each object in time t_0, $\{D_i^{t_0}\}, i \in [1, N_t]$, the objects $G_i^{t_j}, t_j \neq t_0, i \in [1, N_t]$ will be grouped. For grouped objects in each time frames ($\{t_j = 1, 2, ..., T, t_j \neq t_0\}$), the learning function is the same as the frame global feature extraction module in Eq. 1. And after obtaining each set ofgrouped features, the mean pooling method is used to merge all features in this batch of data. The temporal feature network can be described by:

$$f_t(D, G) = \underset{t=1,...,T}{\text{MEAN}}(f(W \cdot (pD_t, G_t))). \quad (2)$$

The MaxPool can be explained as finding the most similar object, and the MeanPool helps to merge all information of each trajectory in the whole time batch.

3) Track Head: As shown in Fig. 3 (e), the track head module consists of two heads: a track ID head and box refinement head. Two separate conv. layers with 1×1 kernels are used in these two heads for the predicted track ID and the refined box detection. The problem is how to assign the predicted bounding boxes to the ground-truth, and how to assign the unordered track IDs to the ground-truth IDs. The solution to these problems is described in the training and testing section as follows.

D. Training and Testing

The one-hot track ID generator is proposed to generate the one-hot track IDs, and in the training process of DiTNet, an ID assignment layer is used to reorder the predicted one-hot track IDs with the ground-truth IDs, for subsequent loss computation. In the process of testing, DiTNet takes as input the raw point cloud batch and directly outputs the track ID with the detection bounding box of each object without the Hungarian algorithm or other data association methods.

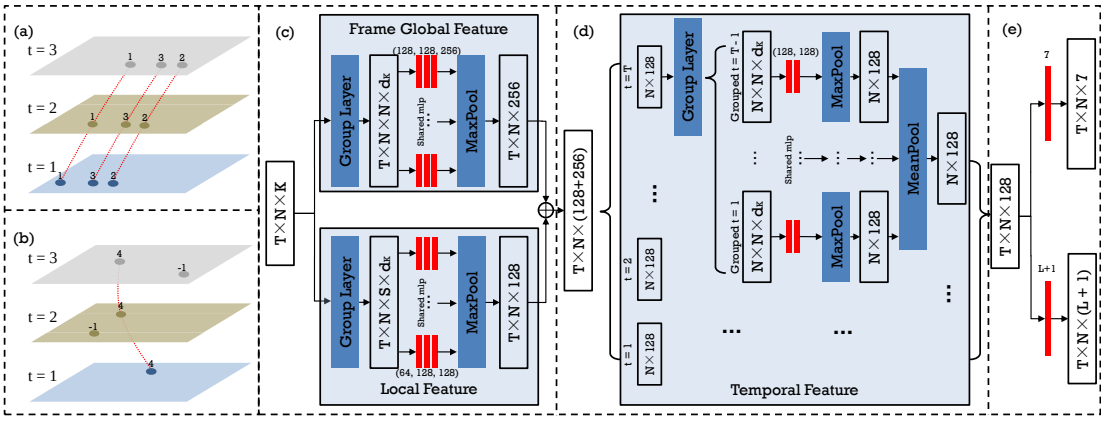

Fig. 3: Two schematics of the common detection distribution and details of the tracking module in DiTNet. (a) An example of spatial feature-based object detection distribution. (b) An example of temporal feature-based object detection distribution. (c) Spatial feature extraction module, which is concatenated by frame global feature extraction module and local feature extraction module. (d) Temporal feature extraction module. (e) Tracking head, which outputs the refined bounding boxes with their track ID. The dots in (a-b) are detections, and red dashed lines are ground-truth trajectories. The red blocks in (c–e) are MLPs with 1×1 kernels, and the number means the number of kernels.

One-hot track identity generator: Firstly, a softmax function is applied for each object's track ID. Then we find the largest track ID probabilities one by one, and make sure each object only has one track ID and each chosen ID is unique in a single time frame.

ID Assignment Layer: Due to the randomness of the track ID assigned to each trajectory, we cannot directly calculate the loss between the predicted tracking trajectory and the ground-truth trajectory. Therefore, for the training process, we design a track ID assignment layer to find the matchings between the predicted trajectories and the ground-truth trajectories. This layer takes as input the predicted trajectories as well as the ground-truth trajectories, and assigns the most matched ground-truth track ID to the predicted one.

We employ the IoU as the cost metric to measure how well a pair of trajectories matches. It calculates the ratio of the number of overlapping points to the total number of points in a pair. Since the number of predicted trajectories varies, it is reasonable to set $L \geq \bar{L}$, where L is the number of predicted trajectories and $\bar{L}$ is the number of ground-truth trajectories. The size of the output of the track ID module is $T \times N_t \times (L+1)$, where N_t is the maximum number of detections and 1 is the dustbin for the false track ID. Let $\{Tr_i, Tr_j\}$ denote a pair of trajectories, and $I \in \mathbb{R}^{L \times \bar{L}}$ denote the reordering matrix. After discarding the dustbin of the false track ID channel, we can get $C_{i,j} \in \mathbb{R}^{L \times \bar{L}}$ as the cost for the matching, where L is the predicted track ID and $\bar{L}$ is the ground-truth ID. The assignment problem can be formulated as a bipartite matching problem:

$$\begin{aligned} \underset{I}{\text{argmin}} \quad & \sum_{i=1}^{L}\sum_{j=1}^{\bar{L}} I_{i,j}C_{i,j} \\ \text{subject to:} \quad & 0 \leq i \leq L, 0 \leq j \leq \bar{L}, \\ & I_{i,j} \in \{0,1\}, \ \sum_{i=1}^{L} I_{i,j} = 1, \ \text{for } \forall j. \end{aligned} \tag{3}$$

In this paper, we use the Hungarian algorithm [5] to solve this problem. With the matching matrix I, we can associate the predicted unordered trajectories with the ground-truth trajectory to get the ordered trajectories. Then the network loss can be calculated. Note again that the Hungarian algorithm is only proposed to assign the predicted track identity with the ground-truth identity for network training. This step does not exist in the inference process.

Post-processing: During the inference process, the network predicts the track IDs in each batch. And by moving the aforementioned sliding window step by step, for each predicted track ID, the algorithm will check whether the objects with this ID have existed in the past trajectories. If yes, then the fresh detection will be added to the existing trajectory, and if the brand new track ID has not appeared before, then a new trajectory will be added to the full trajectory list.

Loss Function: Our loss consists of detection loss and tracking loss, and tracking loss consists of a binary cross entropy loss $\mathscr{L}_{mask}$, a softmax IoU (SIoU) loss $\mathscr{L}_{SIoU}$, a softmax cross entropy loss for each trajectory $\mathscr{L}_{uniqe}$ for track uniqueness, and a triplet loss $\mathscr{L}_{tri}$:

$$\mathscr{L} = \mathscr{L}_{det} + \mathscr{L}_{tra} + \mathscr{L}_{refine} \tag{4}$$

$$\mathscr{L}_{tra} = \mathscr{L}_{mask} + \alpha \cdot \mathscr{L}_{sIoU} + \beta \cdot \mathscr{L}_{uniqe} + \gamma \cdot \mathscr{L}_{tri}, \tag{5}$$

where α, β, and γ are weighting parameters, which are set to $0.1, 1$, and 1.

The detection ROI loss $\mathscr{L}_{det}$ and the box refinement loss $\mathscr{L}_{refine}$ are calculated the same as in the setting in [15]. $\mathscr{L}_{mask}$ is computed from the score tracking score of each point:

$$\mathscr{L}_{mask} = -s \cdot \log(\hat{s}) - (1-s) \cdot \log(1-\hat{s}), \tag{6}$$

where $s \in \{0,1\}$ is the ground-truth, 1 represents that the object detection is negative and 0 otherwise, and $\hat{s}$ is the predicted dustbin score of each object detection. With the associated tracking trajectories, we calculate the SIoU with:

$$\mathscr{L}_{SIoU} = 1 - \frac{1}{L}\sum_{l=1}^{L} \frac{\sum_{n=1}^{N} f_n^l \cdot g_n^l}{\sum_{n=1}^{N} f_n^l + \sum_{n=1}^{N} g_n^l - \sum_{n=1}^{N} f_n^l \cdot g_n^l}, \tag{7}$$

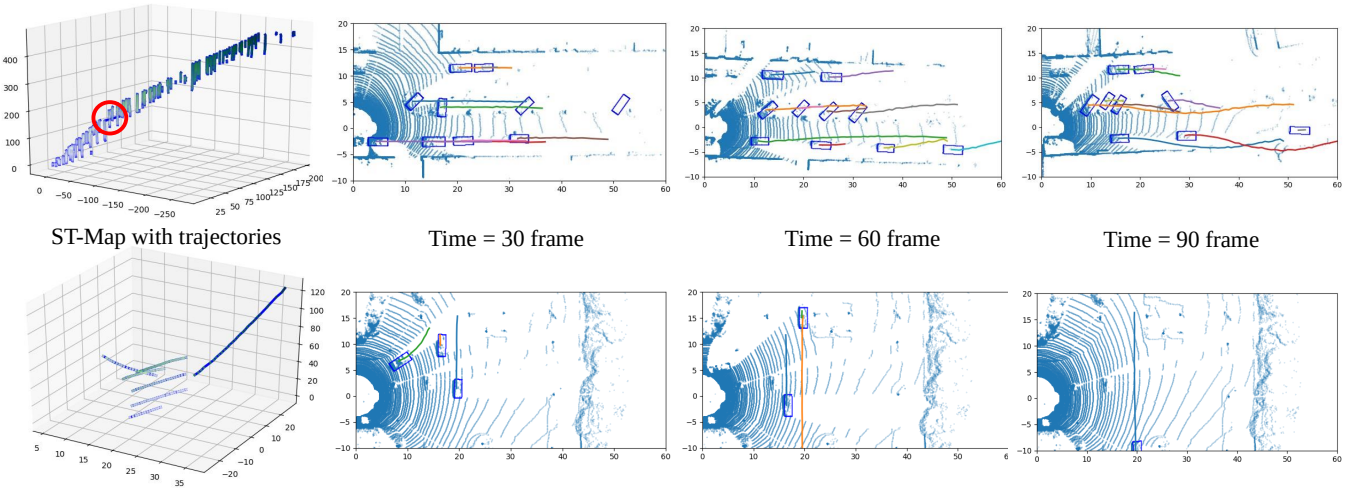

Fig. 4: The qualitative results. Top row is the visualization of tracking results of sequence 0 in the KITTI test dataset, and bottom row is tracking results of sequence 1. The leftmost column is the detection results and trajectories in Spatio-temporal world, the detections are green points and trajectories are blue lines. The three rightmost columns are the trajectories on the BEV of the LiDAR data at different time stamps.

where $f_n^l \in [0,1]$ is the probability for point n to have the trajectory l, which is obtained by applying a softmax function on the trajectory class prediction for point n, $g_n^l \in \{0,1\}$ represents the ground-truth, and N is the number of points. Focal loss [24] is applied in the trajectories' uniqueness loss:

$$\mathscr{L}_{unique} = mask \cdot (-\alpha(1-p)^{\gamma} log(p)), \tag{8}$$

where p is the object's positive probability in each trajectory, mask $\in \{0,1\}$ is the ground-truth of each trajectory, and $\alpha = 0.25$, $\gamma = 2$. The mask will be 0 if the trajectory does not exist in this point cloud batch. This loss only considers the positive trajectories, and it helps that each trajectory will only appear once in each time frame. As our task resembles the instance segmentation and re-ID, we borrow the triplet loss [25] for our task. It encourages the embeddings from the same track to stay as close as possible, while different tracks are separated as far as possible. Specifically, we choose the batch hard triplet loss in [25].

IV. EXPERIMENTS

A. Dataset and Evaluation Metrics

We train and evaluate our model on two datasets. One is the object tracking benchmark dataset from KITTI [34]. The other is the simulation dataset generated from CARLA [35].

Our approach is trained on the CARLA dataset and the first half of 21 sequences from KITTI which have ground-truth labels, and is evaluated on CARLA and the last half of the 21 sequences from the KITTI dataset. We uploaded our test results of the 29 test sequences on KITTI that need to be evaluated online. Since the KITTI dataset mainly uses vehicles as the validation type, we track the Car and Van categories only in that dataset. And in the CARLA dataset, we set the number of pedestrians to 30 and the number of vehicles to 20. Considering misdetections like false-positives, we set the number of detections in each time frame to $N_t = 90$ to include all detection ROIs in the detection module of DiTNet. We set the maximum number of trajectories in each time stamp to $M_{max} = 60$. The size of the track ID label of each point is $N \times (M_{max}+1)$. Its first channel belongs to negative detections and redundancies. The window size is set to 4 and the step size is set to 1 in our experiments for on-line inference.

Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), Mostly Tracked (MT), Mostly Lost (ML), ID Switches (IDS) and fragmentation (FRAG) from the CLEAR MOT metrics[46] are used for evaluating the detection and tracking accuracy.

B. Ablation Study and Comparative Results

We conduct ablation studies on two key factors: the length of the input point cloud batch, and the score threshold of the detection module for tracking. We also present two comparative experiments in this section. First of all, we compare our DiTNet to AB3DMOT [1], RANSAC200, RANSAC1000, and PointTrackNet [2]. RANSAC [26], which can be seen as one of the tracking-by-detection methods, is used here to generate the trajectories in the Spatio-temporal map. It connects the detections as a conventional trajectory fitting method. In our comparative experiments, we split our pre-trained end-to-end network into a detection module and a tracking module. Only the detection results are fed into RANSAC as the input. Thus it makes sense to compare the tracking-by-detection RANSAC with our end-to-end DiTNet when the detection results are the same. RANSAC200 means it iterates 200 times in the fitting process in each Spatio-temporal map, while RANSAC1000 means it iterates 1000 times. In our experiments, the trajectories are all assumed to be conics in the x-t, y-t, and z-t planes. When we have an estimated conic trajectory, "inliers" mean the objects which have less than the threshold distance from the trajectory at that time. The inlier threshold is set

TABLE I: COMPARATIVE RESULTS OF EVALUATION METRICS WITH DIFFERENT DETECTION SCORE THRESHOLDS FOR DIFFERENT TRACKERS IN THE KITTI VALIDATION DATASET. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS.

Car												
Method	Det Score Threshold = 0.2						Det Score Threshold = 0.6					
	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓
RANSAC200 [26]	0.5077	0.867	0.2836	0.3085	78	441	0.578	0.8714	0.3351	0.2251	121	436
RANSAC1000 [26]	0.7404	0.8677	0.6099	0.0762	212	626	0.7806	0.8712	0.6667	0.062	287	647
AB3DMOT [1]	0.6977	0.8653	0.7553	0.0319	**8**	168	0.7431	0.8678	0.7322	**0.0319**	**6**	264
DiTNet (length=2)	0.6743	0.8682	0.7438	0.0306	82	134	0.7203	0.8762	0.732	0.0502	47	131
DiTNet (length=3)	0.7855	0.8654	0.7942	0.0354	57	153	0.7928	0.877	0.7462	0.0543	34	142
DiTNet (length=4)	**0.7865**	0.8680	**0.8103**	0.0393	22	**105**	**0.8108**	0.8783	**0.7935**	0.0421	20	**120**
DiTNet (length=5)	0.786	**0.8683**	0.8081	**0.0397**	23	112	0.8012	**0.8791**	0.7813	0.0421	20	123
Pedestrian												
Method	Det Score Threshold = 0.2						Det Score Threshold = 0.6					
	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓
RANSAC200 [26]	0.4101	0.6709	0.2178	0.1188	551	1041	0.4521	0.67	0.2375	0.099	631	1087
RANSAC1000 [26]	0.4155	0.6713	0.3168	0.099	659	1134	0.4833	0.6705	0.3168	0.0891	637	1114
AB3DMOT [1]	0.4799	0.6699	0.4356	0.0693	**83**	435	0.5446	0.6699	0.4059	0.0693	**72**	451
DiTNet (length=2)	0.4236	**0.6823**	0.4105	0.0553	153	403	0.5674	0.6834	0.4246	0.0653	124	587
DiTNet (length=3)	0.4879	0.6822	0.4253	0.0521	148	412	0.5982	0.6823	0.4272	0.0615	122	564
DiTNet (length=4)	**0.5841**	**0.6823**	**0.4554**	**0.0495**	104	**384**	0.6104	0.6836	**0.4455**	**0.0594**	102	**401**
DiTNet (length=5)	0.5786	0.6816	0.4521	**0.0495**	102	386	**0.6124**	**0.6842**	0.4416	**0.0594**	97	412

TABLE II: COMPARATIVE RESULTS OF EVALUATION METRICS ON THE KITTI TEST DATASET.

Methods	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	Runtime↓	Environment
SRK-ODESA [27]	90.03%	84.32%	82.62%	2.31%	90	501	0.4s	GPU
JRMOT [28]	85.70%	85.48%	71.85%	4.00%	98	372	0.07s	4 cores @ 2.5 Ghz
MOTSFusion [29]	84.83%	85.21%	73.08%	2.77%	275	759	0.44s	GPU
mmMOT [30]	84.77%	85.21%	73.23%	2.77%	284	753	0.02s	GPU @ 2.5 Ghz
mono3DT [31]	84.52%	85.64%	73.38%	2.77%	377	847	0.03s	GPU @ 2.5 Ghz
AB3DMOT [1]	83.84%	85.24%	66.92%	11.38%	9	224	0.0047s	1 core @ 2.5 Ghz
3D-CNN/PMBM [32]	80.39%	81.26%	62.77%	6.15%	121	613	0.01s	1 core @ 3.0 Ghz
FANTrack [19]	77.72%	82.33%	62.62%	8.77%	150	812	0.04s	8 cores @ >3.5 Ghz
Complexer-YOLO [22]	75.70%	78.46%	58.00%	5.08%	1186	2092	0.01s	GPU @ 3.5 Ghz
mbodSSP* [33]	72.69%	78.75%	48.77%	8.77%	114	858	0.01s	1 core @ 2.7 Ghz
Point3DT [2]	68.24%	76.57%	60.62%	12.31%	111	725	0.05s	1 core @ >3.5 Ghz
DP-MCF [21]	38.33%	78.41%	18.00%	36.15%	2716	3225	0.01s	1 core @ 2.5 Ghz
DiTNet (Ours)	84.62%	84.18%	74.15%	12.92%	19	196	0.01s	1 core @ >3.5 Ghz

to 1 meter, considering the detection error. The window size and step size in the Spatio-temporal map generation process are also set to be same as our method. The number of iterations depends on the complexity of the detection result. More iterations are needed to find the suitable trajectories in much denser environments. In the KITTI validation dataset, when the number of iterations is larger than 1000, the tracking results show very limited changes. The ablation studies' results are also compared with these methods.

Tab. I shows the comparative results and ablation studies of the evaluation metrics with a different detection score threshold for different trackers. The *length* means the length of the point cloud batch. We can find that our proposed DitNet outperforms others on MOTA by remarkable margins, which means the overall tracking performance is much better than that of the others. Besides this, we choose the ROI detection results as the detection for other tracking-by-detection methods, and our DiTNet can refine the detections, which makes the performance on MOTP better than that of other methods.

Next we compare our tracking results with the publicly available state-of-the-art approaches from the KITTI tracking benchmark. Tab. II shows the comparative results of the evaluation metrics on the KITTI test dataset. It reveals DiTNet's competitive performance over the other state-of-the-art methods. The running time of 0.01 seconds makes it possible for real-time tracking tasks.

C. Qualitative Results

Fig. 4 shows the qualitative results on the KITTI test dataset. This part of the dataset has no publicly available labels and our network still has good results. We choose to visualize the tracking results of sequence 0 and 1. The trajectories on the BEV of the LiDAR data at three different time stamps are shown in the three rightmost columns in the figure. We find only one ID switch, which is circled in red in sequence 0. In the 29 sequences of the KITTI test dataset, our network only has 19 ID switches and 196 trajectory fragmentations in total.

V. CONCLUSIONS

We proposed here a real end-to-end tracking with detection network, which can directly output the detection box and track identity of each object without data association. Our approach outperforms others on the KITTI benchmark on both the MOTP and IDS. In the future, we will try to fuse the

camera with LiDAR to achieve more complex textures of each object to improve the tracking performance.

REFERENCES

[1] X. Weng, J. Wang, D. Held, and K. Kitani, "3d multi-object tracking: A baseline and new evaluation metrics," *arXiv preprint arXiv:1907.03961*, 2020.

[2] S. Wang, Y. Sun, C. Liu, and M. Liu, "Pointtracknet: An end-to-end network for 3-d object detection and tracking from point clouds," 2020.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[5] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[6] K. Saleh, M. Hossny, and S. Nahavandi, "Long-term recurrent predictive model for intent prediction of pedestrians via inverse reinforcement learning," in *2018 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2018, pp. 1–8.

[7] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[8] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville, "Reseg: A recurrent neural network-based model for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 41–48.

[9] S. Wang, H. Huang, and M. Liu, "Simultaneous clustering classification and tracking on point clouds using bayesian filter," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2017, pp. 2521–2526.

[10] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *European conference on computer vision*. Springer, 2016, pp. 816–833.

[11] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 5308–5317.

[12] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 770–779.

[13] Z. Ding, X. Han, and M. Niethammer, "Votenet: a deep learning label fusion method for multi-atlas segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 202–210.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[16] T. Hirscher, A. Scheel, S. Reuter, and K. Dietmayer, "Multiple extended object tracking using gaussian processes," in *2016 19th International Conference on Information Fusion (FUSION)*. IEEE, 2016, pp. 868–875.

[17] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1592–1599.

[18] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5695–5703.

[19] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "Fantrack: 3d multi-object tracking with feature association network," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1426–1433.

[20] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*, 2016, pp. 850–865.

[21] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *CVPR 2011*. IEEE, 2011, pp. 1201–1208.

[22] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, and H. Michael Gross, "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[23] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.

[24] P. Yun, L. Tai, Y. Wang, C. Liu, and M. Liu, "Focal loss in 3d object detection," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1263–1270, 2019.

[25] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[26] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[27] D. Mykheievskyi, D. Borysenko, and V. Porokhonskyy, "Learning local feature descriptors for multiple object tracking," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[28] A. Shenoi, M. Patel, J. Gwak, P. Goebel, A. Sadeghian, H. Rezatofighi, R. Martin-Martin, and S. Savarese, "Jrmot: A real-time 3d multi-object tracker and a new large-scale dataset," *arXiv preprint arXiv:2002.08397*, 2020.

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

[30] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust multi-modality multi-object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2365–2374.

[31] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 5390–5399.

[32] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granström, "Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 433–440.

[33] P. Lenz, A. Geiger, and R. Urtasun, "Followme: Efficient online min-cost flow tracking with bounded memory and computation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4364–4372.

[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[35] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.