PointTrackNet: An End-to-End Network For 3-D Object Detection and Tracking From Point Clouds

Sukai Wang, Yuxiang Sun, Chengju Liu, and Ming Liu

***Abstract*—Recent machine learning-based multi-object tracking (MOT) frameworks are becoming popular for 3-D point clouds. Most traditional tracking approaches use filters (e.g., Kalman filter or particle filter) to predict object locations in a time sequence, however, they are vulnerable to extreme motion conditions, such as sudden braking and turning. In this letter, we propose PointTrackNet, an end-to-end 3-D object detection and tracking network, to generate foreground masks, 3-D bounding boxes, and point-wise tracking association displacements for each detected object. The network merely takes as input two adjacent point-cloud frames. Experimental results on the KITTI tracking dataset show competitive results over the state-of-the-arts, especially in the irregularly and rapidly changing scenarios.**

***Index Terms*—Point cloud, multiple-object tracking, end-to-end, autonomous vehicles.**

I. Introduction

MULTIPLE-OBJECT Tracking (MOT) plays a very important role in autonomous vehicles and Advanced Driver Assistance Systems (ADAS) [1], [2]. The tracking trajectories can be used for path planning, collision avoidance, and precise pose estimation, etc. Most MOT approaches decompose the task into two stages: object detection and data association. During object detection, objects on roads can be categorized as cars, pedestrians, cyclists, and background, etc. During data association, the same objects at different time stamps are linked to form a trajectory. With the trajectory of each object, we can forecast their positions and predict potential accidents in the future.

Nowadays, various sensors, such as LiDAR, radar, camera, and GPS, have become available for driverless systems. Many methods use sensor fusion techniques for MOT, however, most of them face the challenge of how to keep the whole system working stable in the situation that any of the sensors encounters accidental damages. Besides, the system calibration error is also a major issue. So only using a single sensor could alleviate the integration issue.

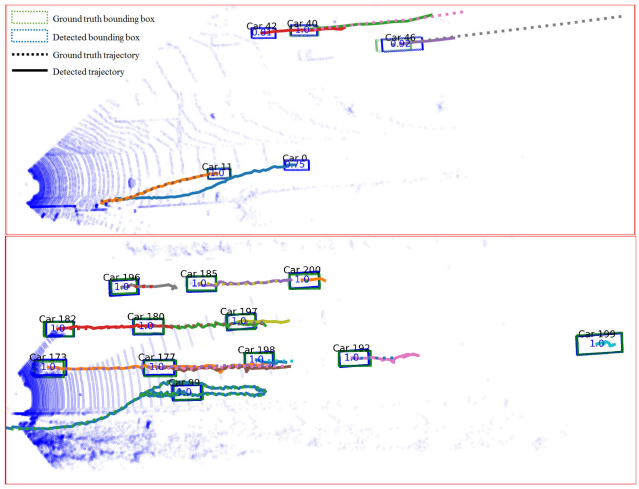

Fig. 1. Birds-eye-view of our detection and tracking results. The dotted lines and green bounding boxes represent the ground-truth, and the blue boxes and solid lines represent the predicted detection and tracking results. The figure is best viewed in color.

Object detection outputs bounding boxes of each object, which is the foundation of MOT. The tracking is to associate the bounding box of each object across different frames. The ID assigned to each bounding box is unique during tracking. Traditional tracking methods associate bounding boxes using motion prediction algorithms like Kalman filters, or particle filters, as well as bipartite graph matching methods like the Hungarian algorithm [3], [4]. They are vulnerable to extreme motion conditions, such as sudden braking and turning. In addition, they can be greatly degraded by unsatisfactory detection results. For example, if an object is lost in the detection, the bounding box of this object is likely to be assigned to other objects, hence leading to tracking errors.

In this letter, we propose an end-to-end 3-D object detection and tracking network using only a single Lidar sensor, which could alleviate the multi-sensor issue and the limitation of the traditional methods. Sample results of our network can be found in Fig. 1. Our network takes as input the adjacent last and current point-cloud frames, and outputs the bounding boxes as well as the tracking trajectories. We perform experiments on the KITTI benchmark dataset and the results confirm the effectiveness of our design. The contributions of this work are listed as follows:

Manuscript received September 11, 2019; accepted January 24, 2020. Date of publication February 17, 2020; date of current version March 5, 2020. This letter was recommended for publication by Associate Editor M. J. Kim and Editor Y. Choi upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grants U1713211 and 61673300, in part by the Basic Research Project of Shanghai Science and Technology Commission under Grant 18DZ1200804, and in part by HKUST ECE Start-up Grant from HKUST for Heterogeneous Navigation System. *(Corresponding author: Ming Liu.)*

Sukai Wang, Yuxiang Sun, and Ming Liu are with the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China (e-mail: swangcy@connect.ust.hk; eeyxsun@ust.hk; liu.ming.prc@gmail.com).

Chengju Liu is with the School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China (e-mail: liuchengju@tongji.edu.cn).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.2974392

- We propose an end-to-end 3-D object detection and tracking network, which takes as input two adjacent raw point clouds, and outputs the predicted bounding boxes as well as the point-wise association displacements.
- We propose a novel data association module in our network to merge the point features of the two frames and associate the corresponding features of the same object.
- We generate the predicted bounding boxes from the point-wise data association. The predicted bounding boxes can refine the detection results.

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

II. Related Work

A. 3-D Object Detection

The 3-D object detection is to localize and classify objects in 3-D scenes. The robust and accurate spatial information captured by a LiDAR is conducive to increasing the accuracy of detection results. In [5], the point clouds were projected into a bird's-eye-view where 2-D CNN could be applied on the 2-D images. In [6], point cloud was voxelized into grids, then high-level features could be extracted from the hand-crafted features encoded in the grids. In [7], voxel-wise features was extracted, instead of hand-crafted pre-prepared features. In [8], PointNet++[9] was utilized as a backbone network for 3-D object detection because of its simplicity and efficiency. PointNet++ can learn features directly from raw disordered point clouds, to solve the problem of high computational cost and information loss caused by dense convolution after quantization.

B. Data Association

Filter-based and batch-based methods are two choices for data association in multiple-object tracking.

Most batch-based approaches chose the 3-D objects' accurate trajectories as the output of the MOT problem [10]. Convolutional siamese networks [11] were broadly used for appearance feature extraction and similarity cost computation. After getting the localization results from the detection network, matching net [12] compared each object in adjacent frames and used learned metrics to find the corresponding pairs and produce discrete trajectories. In [10], a scoring net and matching net were proposed, and the tracking problem was changed to an end-to-end learning linear programming (LP) problem by designing a backpropagating subgradient descending equation through the linear program. These learning-based data-association methods were more robust and accurate than conventional methods, but the detected bounding box accuracy also limited the accuracy of the tracking or trajectory.

Filter-based approaches used Kalman filters [1], [13], [14] and Gaussian processes [15] to generate the association matrix between adjacent frames and used the Hungarian algorithm [13] to optimize it. PointIT [1] used spherical images generated from 3-D point clouds to get each object's location and extended the SORT [13] algorithm to finish the real-time tracking problem. The filter-based methods were efficient enough for real-time deployment. However, the error accumulation problem can scarcely be solved.

These existing MOT algorithms always need the objects' precise bounding boxes in each frame of a sequence. Several optical flow and scene flow have made some achievements, not only on 2-D images [16], but also on 3-D point clouds [17]. If we have the correct corresponding points pairs, and the displacement in adjacent frames, the movement of each object can be estimated exactly. Our proposed association module, which only estimates the objects and learns their point-wise displacement, can be added to any of the existing methods, and will improve the accuracy of data association.

III. The Proposed Network

Our proposed network only takes as input two adjacent raw disordered point clouds, and outputs object bounding boxes and the trajectory of each object. The detection and data association are the two main components of tracking methods. We design a point-wise data association method to reduce the possible negative impacts caused by degraded object detection.

Fig. 2 shows the overview of our network architecture. The network can be split into four modules: the point-wise feature extractor (3-D object detector), the data association module, the refinement module, and the trajectory generator module. In the following, we will introduce the four components as well as the loss function.

A. Point-Wise Feature Extractor

Given as input an $N \times 3$ point cloud, the object detector is proposed to generate an $N \times 2$ mask and M bounding boxes, where N means the number of the points and the mask is a binary 0-1 classification label to distinguish foreground and background. The point cloud feature is extracted from a backbone network. We leverage PointNet [18] and PointNet++[9] to learn point features instead of hand-crafted features. The backbone network processes the point cloud in the intuitive Euclidean space. With the disordered point cloud as input, it learns to overcome the non-uniformity and combines multi-scale neighborhood features to produce efficient and robust features.

In this letter, we propose a detector with four *Set Abstraction (SA) Modules* to downsample the point cloud, four *Feature Propagation (FP) Modules* to upsample from fewer points to more points and two *Fully Connected Layers* to finally output the mask and bounding boxes results. The SA modules downsample the lower level points $\{p_i^{l-1}\}_{i=1}^{m}$ to the higher level points $\{p_i^{l}\}_{i=1}^{n}$ where $m > n$, while a Set Upconv layer propagates them in the opposite direction. Same scale grouping (SSG) and multi-scale grouping (MSG) with farthest point sampling (FPS) [19] are applied in the SA module, where each layer's points are the subset of the upper layer's points. The grouping layer in the FP module constructs local region sets by finding the nearest points in the higher level points, which means increasing the number of points in the feature propagation.

Inspired by F-Pointnet [20], we set shape of the detector's output as $N \times (2 + 3 + 2 \times NH + 4 \times NS + NC)$. The first $2 + 3$ means the probability and center of the bounding box,

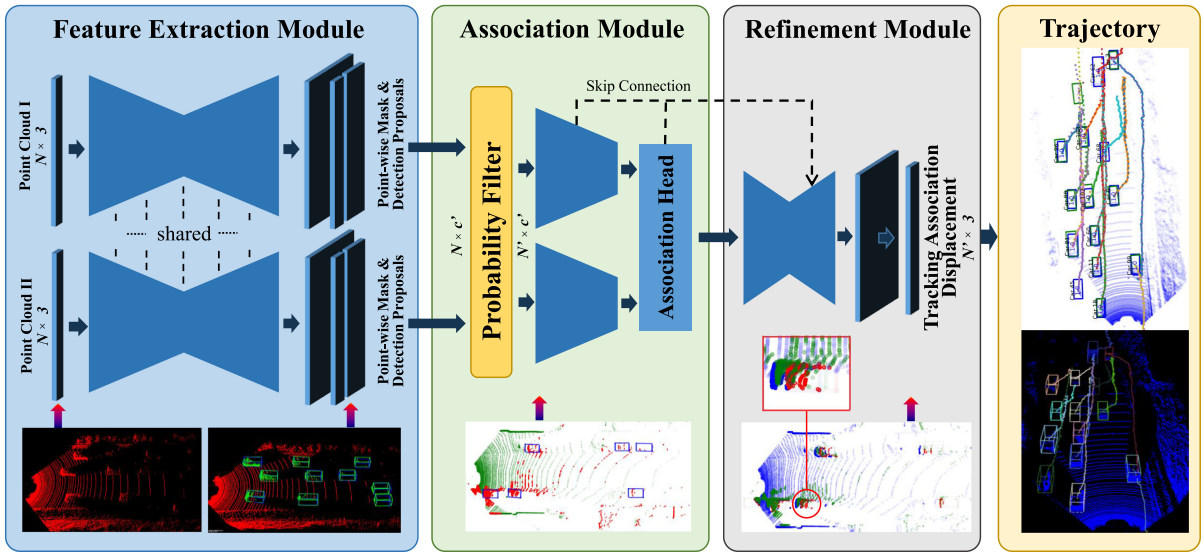

Fig. 2. PointTrackNet architecture. Feature Extraction Module outputs both point-wise mask and object bounding boxes. Association Module has a probability filter to reserve the high probability foreground points, and an association head to fuse the features of the two frames. Refinement Module outputs the point-wise tracking association displacements. And trajectory generator matches the same object and visualizes the bird's-eye-view and 3-D trajectories.

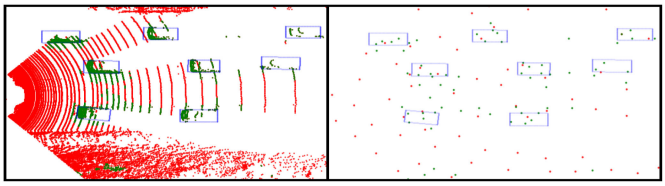

Fig. 3. Comparison between original FPS sampling and filter FPS sampling. The left is the result in TRA_SA1 and the right is in TRA_SA3 (detailed in Table I). The green points are the FPS sampling result after the probability filter. The red points are the original FPS sampling result. Blue boxes are the ground truth object bounding boxes. The figure is best viewed in color.

TABLE I
DETAILED NETWORK CONFIGURATIONS

	Layer	Radius	Point Num	MLP width
DET	SA1	1.0	2048	[32,32,64]
	SA2	2.0	512	[64,64,128]
	SA3	4.0	128	[128,128,256]
	SA4	8.0	32	[256,256,512]
	FP1	8.0	128	[256,256]
	FP2	4.0	512	[128,128,256]
	FP3	2.0	2048	[128,128,256]
	FP4	*	N	[256, 256]
	FC1	*	*	128
	FC2	*	*	C
TRA	Pr Filter (f1, f2)	*	N'	*
	SA1 (f1, f2)	0.5	2048	[32,32,64]
	SA2 (f1, f2)	1.0	512	[64,64,128]
	Assn.	5.0	512	[128,128]
	SA3	4.0	32	[256,256]
	FP1	4.0	512	[256,256]
	FP2	1.0	2048	[128,256,256]
	FP3	*	N'	[256, 256]
	FC1	*	*	128
	FC2	*	*	3

$2 \times NH$ means the number of classes with the residuals of the heading bins, $4 \times NS$ means the number of classes with the residuals of size (bounding box length, width, and height), and NC means its semantic class.

B. Association Module

The association module contains a probability filter, two SA layers, and an association head. The probability filter is proposed to balance the fore-background points and decrease the computation cost. The point-wise mask in the feature extraction module can be used as the attention map to choose the salient points. We use the mask probability information to keep the top N' foreground points, which provide much more useful local geometry features than most of the background points. For the reserved N' points, the FPS in the sampling layers is used, the same as in the conventional methods. Fig. 3 presents the comparative results of the FPS used in the original raw point clouds with the filtered point clouds. We visualize the sampling results in TSA 1 and TSA 2 (details in Table I). We find that there are rare points belonging to the foreground if we use FPS sample after four times, especially in distant vehicles, in a wide range of environments, but our method can reserve more foreground points in the entire process. Two SA layers are connected to down-sample the filtered point cloud.

Then an association head module is proposed to merge the features of two frames. The inputs are two adjacent filtered point cloud points: $\{p_i^{t-1} = (x_i^{t-1}, y_i^{t-1}, z_i^{t-1}); p_i^t = (x_i^t, y_i^t, z_i^t)\}_{i=1}^{N'}$ and features $\{f_i^{t-1}; f_i^t\}_{i=1}^{N'}$, where t ranges from 0 to the end of a sequence, N' is the number of filtered points, the feature vector $f_i \in \mathbb{R}^c$. The module learns an embedded feature $f_i^{'} \in \mathbb{R}^{c^{'}}$ that associates with point p_i^{t-1} in the first frame.

The visualization of the association head module is shown in Fig. 4. For every point p_i^{t-1}, we find the nearest K points in the point set $\{p_j^t\}_{j=1}^K \subset \{p^t\}$, with the features $\{f_j^t\}_{j=1}^K$, and the Euclidean distance $\{d_j^t\}_{j=1}^K \subset \mathbb{R}^3$ between p_i^{t-1} with new

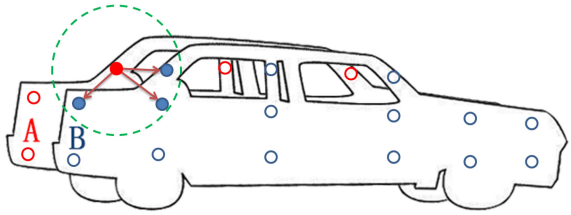

Fig. 4. Association Head visualization. Red points in frame **A** are in the past frame in time $t-1$, and blue points in frame **B** are in the current frame in time t. Solid blue points are the grouped K points closest to the single chosen solid red point. We fuse the features of red solid points and blue solid points, as the embedded features. The figure is best viewed in color.

points in the next frame. After concatenating f_i^{t-1}, $\{f_j^t\}_{j=1}^K$ and $\{d_j^t\}_{j=1}^K$ as embedded features, we try to use multi-layer perceptrons and element-wise max pooling to learn the point-wise association tracking displacement between the points of two adjacent frames.

C. Refinement Module

The refinement module is composed of one SA layer, three Set Upconv layers and two fully connected layers. For the second Set Upconv layer, we introduce a skip connection from TSA_L1$_{f1}$ to concatenate with it. The detailed layer parameters are shown in Table I.

The refinement module is proposed to refine the association features. The output of the last Set Upconv layer $\{f_i^{l=0}\}_{i=1}^{N'} \subset \mathbb{R}^{c'}$ has the same number of points N' as the points after the filter step.

Two FC layers are connected to calculate the final tracking results $\{t_i\}_{i=1}^{N'} \subset \mathbb{R}^3$, where t_i means point-wise association displacement in two frames.

D. Trajectory Generator

We use the predicted point-wise association tracking displacement to predict the movement of the bounding boxes of objects in the last frame. The complete pseudocode in the tracking management is described in Algorithm 1.

E. Loss Function

The loss can be split into two parts, detection loss and tracking loss, as (1):

$$\mathscr{L} = \mathscr{L}_{det} + \mathscr{L}_{tracking}. \tag{1}$$

Similar to [20], we set the detection loss as (2), where $\mathscr{L}_{mask}$ means the point-wise fore-background loss, $\mathscr{L}_{box}$ means bounding box loss in (3), and $\mathscr{L}_{sem\_cls}$ means the semantic class loss. $\mathscr{L}_{mask}$ and any *class*-relevant losses use the focal loss [21] to tackle the fore-background imbalance problem, as (4):

$$\mathscr{L}_{det} = \mathscr{L}_{mask} + \mathscr{L}_{box} + \mathscr{L}_{sem\_cls}, \tag{2}$$

$$\mathscr{L}_{box} = \mathscr{L}_{center} + \mathscr{L}_{h\_cls} + \mathscr{L}_{h\_reg} + \mathscr{L}_{s\_cls} + \mathscr{L}_{s\_reg}, \tag{3}$$

$$\mathscr{L}_{*\_cls} = -(1-p_t)^{\gamma} log(p_t). \tag{4}$$

Algorithm 1: Association to Trajectory.

Input: Detected object bounding boxes B_{t-1} and B_t in time t; Point-wise association displacement d_i; LiDAR points P_t

```
if t = 0 then
    foreach B_i^0 do
        initialize new trajectory and save
else
    foreach B_i^{t-1} do
        find ∀p_m ∈ B_i^{t-1}
        d̄ = (1/N_pm) Σ_{p_m ∈ B_i^{t-1}} t_m
        Pos_{B_i^t} = Pos_{B_i^t} + d̄
        if max(IoU) for each B_i^t > τ then
            Add this pair to Match_Pairs
        else
            Add B_i^{t-1} to Unmatch_List_{t-1}
    foreach B_i^{t-1} do
        if B_i^{t-1} hasn't been chosen then
            Add B_i^t to Unmatch_List_t
    Use Match_Pairs, Unmatch_List_{t-1} and
     Unmatch_List_t to update trajectory and save
```

We use the mask information to split the total point cloud into positive part and negative part, where the positive means foreground and the negative means background. Then we apply the weighted L2 Loss in the tracking-net in (5):

$$\mathscr{L}_{tracking} = \alpha \frac{N}{N_{pos}} L2^{pos} + \beta \frac{N}{N_{neg}} L2^{neg}, \tag{5}$$

where $L2^{pos}$ and $L2^{neg}$ are the L2 loss functions of positive and negative point clouds, which are distances between predicted tracking displacement with the ground truth. N means the number of points in the total point cloud, N_{pos} and N_{neg} are number of points in each part of points.

IV. Experimental Results and Discussions

A. Dataset and Evaluation Metrics

We evaluate our model on the 3-D object tracking benchmark dataset from KITTI [22]. The dataset consists of 21 training and validation sequences with publicly given ground truth labels and 29 test sequences that need to evaluate on-line. For the training and validation set, there are 7987 point-cloud frames, 636 vehicle trajectories, and 30601 individual vehicles in total. We split the training and validation set into two parts, *Seq-0000* to *Seq-0015* for training and *Seq-0016* to *Seq-0020* for validation.

During the training process, we firstly extract the ground points, and label the points in the ground truth bounding boxes as foreground points and the others as background. Then, we compare two objects in adjacent frames which have the same object ID, and substitute the bounding boxes' movement for point-wise association tracking displacement. During the testing process, we only feed two adjacent raw disordered point clouds to the network.

TABLE II
ANALYSIS OF COMPUTATION TIME. THE UNIT IS SECOND

K=64	N'=5000	N'=1000	N'=500
Input N=15000	0.099	0.086	0.085
Input N=10000	0.074	0.060	0.059
Input N=5000	0.048	0.037	0.036

The CLEAR MOT metrics [23] are used in our method for evaluating detection and tracking accuracy. Mostly Tracked (MT), Mostly Lost (ML), ID Switches (IDS) and FRAGmentation (FRAG) can reflect tracking orientation characteristics. A higher MT and a lower ML, IDS, and FRAG mean the tracker has improved in continuous tracking and reduced the trajectory FRAG and IDS.

B. Network Details

The detailed parameters of each layer are shown in Table I. Each learnable layer adopts multi-layer perceptrons with a few Linear-BatchNorm-ReLU layers parameterized by its linear layer width.

We train the detection network and tracking network separately. The detection network is trained using a batch size of 8 and the tracking using a batch size of 6. We use an Adam [24] optimizer for training, in which beta1 $= 0.9$, beta2 $= 0.999$, epsilon $=$ 1e-08. The cyclical learning rates (CLR) [25] is used in our architecture to lead a higher training speed. The learning rate changes linearly, upper bound is 0.001, lower bound is 0.0001, and the cycle range is 8 epochs.

The proposed PointTrackNet is implemented on Tensorflow 1.9.0 with CUDA 9.0 and cuDNN 7.0 libraries. It is trained on an Intel 3.7 GHz i7 CPU and a single GeForce GTX 1080Ti graphics card.

The number of points in point-wise point cloud network is the key factor in computation cost. We compare the computation time of two raw point cloud inputs in the different sets of input size and the probability filter size in Table II. In our experiment, we set the number of input point cloud N to 15000, the number of points after probability filter N' to 5000, and the K in Association Module to 64.

C. Ablation Study

An ablation study is designed to evaluate various multi-feature fusion methods in the association head module. The association module uses the local features from two detectors to learn the weight related to the displacement of each point in the first frame. Which layer's features to choose and how to merge these features in different frames are studied in this section.

Table III illustrates the ablation study result with various feature fusion methods. We consider the coordinates difference between p_i^{t-1} and $\{p_j^t\}_1^K$ as the "fundamental" association displacement. So if we use a feature-correlated fusion method, like the cosine distance or dot product, the closer the characteristics of the two points are, the greater the value after the fusion will be. That is the reason why these two methods show advantages over the other two methods in Table III.

TABLE III
ABLATION STUDY RESULTS OF EVALUATION METRICS WITH VARIOUS FEATURE FUSION METHODS. BOLD FONT HIGHLIGHTS THE BEST RESULTS

Fusion method	MOTA↑	MOTP↑	MT↑	ML↓
elementwise poduct	0.704	0.792	0.539	0.080
concat	0.717	0.790	0.65	0.044
cosine distance	**0.747**	**0.811**	0.68	0.047
dot product	0.734	0.810	**0.74**	**0.036**

TABLE IV
ABLATION STUDY RESULTS OF EVALUATION METRICS WITH VARIOUS LAYERS' FEATURES TO FUSE. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS

Fusion method	MOTA↑	MOTP↑	MT↑	ML↓
DSA2 + DFP2	0.602	0.800	0.639	0.062
DSA3 + DFP3	0.634	0.795	0.612	0.075
Filter + FC2	0.707	**0.801**	0.685	0.040
Filter + FC1	**0.739**	0.800	**0.710**	**0.034**

Table IV illustrates the ablation study results with different layers' features to fuse. We choose the features from $DSA2 + DFP2$, $DSA3 + DFP3$, $filter + FC1$, $filter + FC2$ to evaluate. We can find that using the probability filter described in Sec. III.(2) is better than using the original farthest point sampling result. So using more features in FC1 is better than FC2, though it will slightly increase the computational load.

D. Comparative Results

Kalman filter is used in SORT [13] and DEEP SORT [14] to update the past frame's object position. However, when the movement of the objects or the collecting vehicle is irregular, the predicted update result will lead to a severe increase of the mismatches and switches of output trajectories. We design an experiment in which we assign a displacement to individual object cars, which can also be seen as the movement of the vehicle that collected the dataset. We augment the dataset, that is to say, apply a point-wise displacement into the points inside the bounding boxes.

Fig. 5 shows the comparison results on the KITTI validation dataset. In the figure, we can see that with increasing displacement, evaluation metric MOTA of the traditional trackers SORT and DEEP SORT decreases significantly. For our method, however, using the predicted tracking to compensate for the shift distance between two frames, the evaluation results are relatively stable at a high level. The results show that our method presents great superiorities in relatively larger changing environments, as well as in the cases that the data-collecting vehicle regularly moves. The ground truth tracking assists our method to have performance which is at its upper bound. The tiny gap between our predicted tracking with the upper bound shows that even the tracking value are not so precise, the object IoU corresponding process can provide the correct results. The detailed comparison results of all evaluation metrics with various displacements are described in Table V, which show overall superior performance than the baseline methods.

TABLE V
COMPARATIVE RESULTS OF EVALUATION METRICS WITH VARIOUS DISPLACEMENTS FOR DIFFERENT TRACKERS. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS

	Methods	*Displacement* = 0.0						*Displacement* = 2.0					
		MOTA↑	MOTP↑	MT↑	ML↓	ID_sw↓	FRAG↓	MOTA↑	MOTP↑	MT↑	ML↓	ID_sw↓	FRAG↓
pred bboxes	SORT	0.611	0.797	0.728	0.032	99	162	0.357	0.807	0.032	0.184	87	205
	DEEP_SORT	0.648	**0.805**	0.652	0.054	87	151	0.259	0.811	0.000	0.293	80	**137**
	PointTrackNet	**0.738**	0.804	**0.750**	**0.021**	**68**	**135**	**0.647**	**0.812**	**0.445**	**0.130**	**78**	144
gt bboxes	SORT	0.927	0.933	0.976	0.000	43	53	0.601	1.000	0.228	0.032	107	228
	DEEP_SORT	0.926	0.933	0.976	0.000	42	56	0.698	1.000	0.423	0.021	126	239
	PointTrackNet	**0.978**	**0.933**	**1.000**	**0.000**	**34**	**39**	**0.957**	**1.000**	**0.989**	**0.000**	**29**	**44**

TABLE VI
COMPARATIVE RESULTS OF EVALUATION METRICS ON KITTI TEST DATASET. * MEANS THIS METHOD IS MERELY LIDAR-BASED. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS

Method	MOTA↑	MOTP↑	MT↑	ML↓	ID_sw↓	FRAG↓
CEM [26]	51.94%	77.11%	20.00%	31.54%	125	396
ODAMOT [27]	59.23%	75.45%	27.08%	15.54%	389	1274
mbodSSP [28]	72.69%	78.75%	48.77%	8.77%	114	858
SSP [28]	72.72%	78.55%	53.85%	8.00%	185	932
RMOT [29]	65.83%	75.42%	40.15%	9.69%	209	727
MDP [30]	76.59%	82.10%	52.15%	13.38%	130	387
MCMOT-CPD [31]	78.90%	82.13%	52.31%	11.69%	228	536
aUToTrack* [32]	**82.25%**	80.52%	**72.62%**	**3.54%**	1025	1402
FANTrack* [12]	77.72%	**82.33%**	62.62%	8.77%	150	812
Complexer-YOLO* [33]	75.70%	78.46%	58.00%	5.08%	1186	2092
DSM* [10]	76.15%	83.42%	60.00%	8.31%	296	868
PointTrackNet* (Ours)	68.23%	76.57%	60.62%	12.31%	**111**	**725**

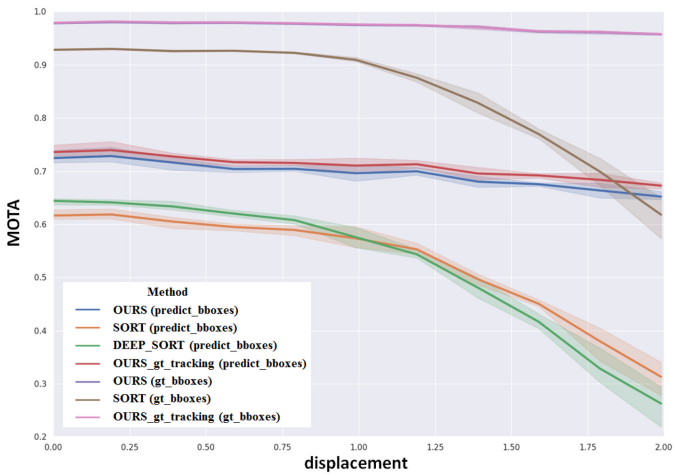

Fig. 5. Comparison results with baseline trackers on KITTI validation dataset. The MOTA results of three different trackers in various manual displacement ranges from 0.0 to 2.0 (meters). '(gt_bboxes)' means that the method uses ground truth detection result, and 'gt_tracking' means that the method uses ground truth point-wise association tracking displacement. The figure is best viewed in color.

In Table VI we compare our results with the publicly available LiDAR-based methods and some well-known multi-sensor tracking methods from the KITTI tracking benchmark. Our approach shows the competitive results with the state-of-the-art. We can see that in terms of some of the metrics, like IDS and FRAG, ours outperforms all the other methods. These two metrics are more related to the tracking performance, which means that despite low accuracy in the object detection, our network can still obtain better results on the tracking. Furthermore, the detection in our end-to-end framework can be replaced by other point-wise feature extractors, to improve the tracking accuracy.

E. Qualitative Evaluation

Fig. 6 displays sample qualitative results by running our tracker on the KITTI tracking validation sequences. We compare the tracking results on a multi-car scenario (sequence 20) with a multi-obstacle scenario (sequence 19), and different manually added displacements of each object to imitate the movement of the LiDAR. We visualize the trajectories with predicted bounding boxes from 3-D perspective. The bounding box and trajectory of each object share the same color. When displacement is a random value less than 2.0, the first car, in the navy color, is lost after 10 frames. Because our detector lost the car in frames 7 to frame 9, the huge displacement of that car confuses the tracker. However, the result using the original point cloud as input shows that our tracker performs fairly well, even when the detector lost the target car. The tracking association information helps the tracker to reduce the impact of the short term disappearance of objects. Sequence 19 with displacement equal to 2.0 can be seen as a failure sample. The reason could be that the object car is far away from the LiDAR so that the feature is not integrated enough. In addition, the large number of obstacles in front of the car could result in serious occlusions. Increasing the training sample and improving the detector performance could be conducive to improving results.

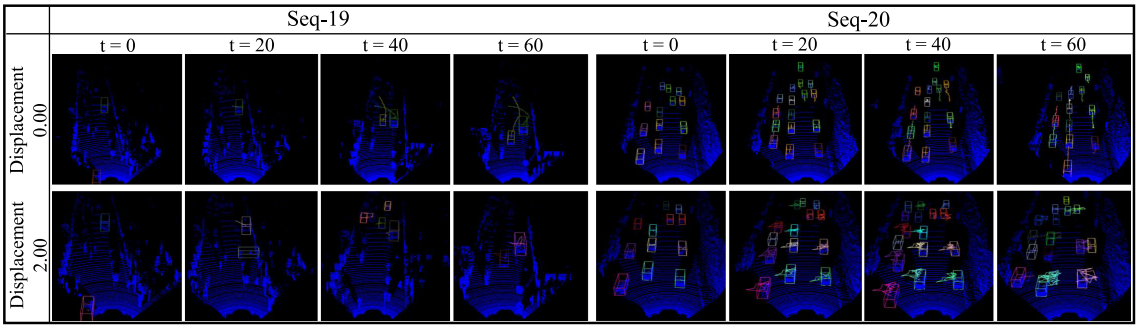

Fig. 6. The qualitative results of sequences 19 and 20. The top two rows show the original point cloud inputs and the bottom two rows show the augmented point cloud with random displacement less than 2.0. From left to right, the same object at different time stamps is represented by the same color and the lines are trajectories. The figure is best viewed in color.

V. Conclusions

We proposed here a novel end-to-end LiDAR-based network for 3-D object detection and tracking. The experimental results demonstrate that our network shows competitive results over the state-of-the-arts. Due to the diversity of detection approaches, we believe that the point-wise feature extractor can be replaced with more powerful one in the future to improve the overall tracking performance. We will also verify the tracking network's feasibility based on the voxel-based detector or 3-D to 2-D detector.

References

[1] Y. Wang, Y. Yu, and M. Liu, "PointIT: A fast tracking framework based on 3D instance segmentation," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1902.06379

[2] S. Wang, H. Huang, and M. Liu, "Simultaneous clustering classification and tracking on point clouds using bayesian filter," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2017, pp. 2521–2526.

[3] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Res. Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[4] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *Proc. IEEE 12th Int. Conf. Comput. Vision*, 2009, pp. 1515–1522.

[5] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "Birdnet: A 3D object detection framework from lidar information," in *Proc. IEEE 21st Int. Conf. Intell. Transp. Syst.*, 2018, pp. 3517–3523.

[6] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1513–1518.

[7] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 4490–4499.

[8] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3D object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 770–779.

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

[10] D. Frossard and R. Urtasun, "End-to-end learning of multi-sensor 3D tracking by detection," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 635–642.

[11] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proc. IEEE Conf. Comput. vision Pattern Recognit.*, 2015, pp. 1592–1599.

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

[13] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 3464–3468.

[14] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 3645–3649.

[15] T. Hirscher, A. Scheel, S. Reuter, and K. Dietmayer, "Multiple extended object tracking using gaussian processes," in *Proc. IEEE 19th Int. Conf. Inf. Fusion*, 2016, pp. 868–875.

[16] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 2758–2766.

[17] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 529–537.

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 652–660.

[19] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Trans. Image Process.*, vol. 6, no. 9, pp. 1305–1315, Sep. 1997.

[20] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 918–927.

[21] P. Yun, L. Tai, Y. Wang, C. Liu, and M. Liu, "Focal loss in 3D object detection," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1263–1270, Jan. 2019.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit*, 2012, pp. 3354–3361.

[23] K. Bernardin, A. Elbs, and R. Stiefelhagen, "Multiple object tracking performance metrics and evaluation in a smart room environment," in *Proc. 6th IEEE Int. Workshop Visual Surveillance, Conjunction ECCV*, vol. 90, Citeseer, 2006, p. 91.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, May 7-9, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[25] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," *Artif. Intell. Mach. Learn. Multi-Domain Operations Appl.*, vol. 11006, p. 1100612, 2019.

[26] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 58–72, Jan. 2014.

[27] A. Gaidon and E. Vig, "Online domain adaptation for multi-object tracking," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1508.00776

[28] P. Lenz, A. Geiger, and R. Urtasun, "Followme: Efficient online min-cost flow tracking with bounded memory and computation," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 4364–4372.

[29] J. H. Yoon, M.-H. Yang, J. Lim, and K.-J. Yoon, "Bayesian multi-object tracking using motion context from multiple objects," in *Proc. IEEE Winter Conf. Appl. Comput. Vision*, 2015, pp. 33–40.

[30] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 4705–4713.

[31] B. Lee, E. Erdenee, S. Jin, M. Y. Nam, Y. G. Jung, and P. K. Rhee, "Multi-class multi-object tracking using changing point detection," in *European Conference on Computer Vision*, Berlin, Germany: Springer, 2016, pp. 68–83.

[32] K. Burnett, S. Samavi, S. L. Waslander, T. D. Barfoot, and A. P. Schoellig, "aUToTrack: A lightweight object detection and tracking system for the SAE autodrive challenge," in *Proc. 16th Conf. Comput. Robot. Vision*, 2019, pp. 209–216.

[33] M. Simon *et al.*, "Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds," in *Proc. IEEE 19th Conf. Comput. Vision Pattern Recognit. Workshops*, 2019, pp. 0–0.