ST-TrackNet: A Multiple-Object Tracking Network Using Spatio-Temporal Information

Sukai Wang, *Member, IEEE*, Yuxiang Sun, *Member, IEEE*, Zheng Wang, *Senior Member, IEEE*, and Ming Liu, *Senior Member, IEEE*

***Abstract*— Multiple-object tracking (MOT) is a crucial component in autonomous driving systems. However, inaccurate object detection is always the bottleneck for MOT. Most detectors are not designed to take the temporal information across consecutive frames into consideration. To take advantage of such information, we design a novel data representation, the spatio-temporal (ST) map, which collects a batch of detection results spatio-temporally, and we train a novel network, ST-TrackNet, to assign predicted track IDs to each positive detection across a sequence. With our ST map detection fed into the tracker, the correlation of objects between adjacent frames becomes prominent, which improves the performance of the tracker in the data association step. Moreover, the long-term trajectory in a sequence also helps to refine the detection results. We train and evaluate our network on the KITTI dataset, a CARLA simulation dataset, and a dataset recorded in a factory environment. Our approach generally achieves superior performance over the state-of-the-art.**

***Note to Practitioners*—We investigate the MOT problem in this paper. A spatio-temporal pipeline is proposed to provide a solution to this problem. Object detection results produced by off-the-shelf object detectors are used to form the proposed ST maps. In low signal-to-noise ratio (SNR) situations, our proposed framework can achieve more accurate and robust tracking results with more false-positives. Due to the simplicity and modular design of our framework, it can be applied directly after the detection stage to achieve the online tracking task. The proposed method is evaluated on several datasets, and the experimental results demonstrate its effectiveness. Our method can also be used for other autonomous driving applications, such as path planning and trajectory prediction.**

***Index Terms*— Autonomous driving, multi-object tracking, deep learning, spatio-temporal map.**

I. Introduction

MULTIPLE-OBJECT tracking (MOT) is gradually receiving more and more attention in Advanced Driver Assistance Systems (ADAS). A fully ADAS contains data composing [1], calibration [2], localization and state estimation [3], self-trajectory prediction and planning [4], perception, etc. In the perception part, MOT task aims to associate the objects in a long-term sequence with their unique track ID. And as the prior result, the performance of object detection networks is the key factor for tracking-by-detection trackers in MOT. The motivation of object detection is to find the bounding boxes and heading angles for objects. The current state-of-the-art object detection techniques have achieved impressive results on different benchmarks. However, the balance between false-negatives and false-positives remains a crucial problem. The confidence threshold (CT) for choosing the object bounding boxes at the final stage of a detector influences the detection performance: the higher the values of the CT, the more false-negatives, and the lower the value, the more false-positives. Too many false-positive detections can lead to a dramatic decrease in the data association stage due to the interference of redundant error information. However, considering autopilot safety, more false-positives are more acceptable than more false-negatives. With this prerequisite, one of our motivations is to filter the false-positives and associate the same object at different times in one trajectory.

The data association in consecutive frames is the main step in MOT problems. The real world is four-dimensional (i.e., spatio-temporal): the 3-D length, width and height, and the 1-D time. Objects in the 4-D world usually exhibit spatio-temporal continuity. For example, objects in real road environments will not appear or disappear suddenly. Fig. 1 shows a plot of object detection and tracking results in the 4-D spatio-temporal world, and this kind of plot is called a spatio temporal (ST) map in this paper. The MOT task is to find the red trajectories in the map. From the ST map, we can see that the object trajectories become more distinguishable, which can benefit the data association in MOT.

Manuscript received 25 April 2022; accepted 9 September 2022. Date of publication 31 October 2022; date of current version 5 January 2024. This article was recommended for publication by Associate Editor Y. Pan and Editor C. Seatzu upon evaluation of the reviewers' comments. This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Project 2021B1515120032 and in part by the Zhongshan Science and Technology Bureau Fund under Project 2020AG002. The work of Ming Liu was supported by the Project of Hetao Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone under Grant HZQB-KCZYB-2020083. *(Corresponding author: Ming Liu.)*

Sukai Wang is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China, and also with the Clear Water Bay Institute of Autonomous Driving, Nanshan, Shenzhen 518000, China (e-mail: swangcy@connect.ust.hk).

Yuxiang Sun is with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: sun.yuxiang@outlook.com).

Zheng Wang is with the Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: wangz@sustech.edu.cn).

Ming Liu is with The Hong Kong University of Science and Technology (Guangzhou), Nansha, Guangzhou, Guangdong 511400, China, also with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, SAR, China, and also with the HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen 518057, China (e-mail: eelium@ust.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2022.3216450.

Digital Object Identifier 10.1109/TASE.2022.3216450

In this paper, we propose a point-wise neural network, ST-TrackNet, which treats detections in the ST map as points in a point cloud. Our network takes as input the spatio-temporal point cloud and outputs a track ID for each object. However, there are two critical issues in the network. One is that the number of trajectories is not constant, and the other is that the order of trajectories is not required to be fixed. One particular object only needs to be classified into one trajectory, no matter what the track ID of the trajectory is. And the number of trajectories depends on the number of real objects at that moment, which is uncertain. These two issues are especially prominent during network training. In our approach, we set a maximum number of trajectories to solve the first issue, and apply a track assignment module to efficiently assign ground-truth trajectories to the predicted trajectories for the second. The ST map also contains more regulations than an ordinary point cloud. We employ novel sampling and feature extraction methods to learn the association relations in the ST map. The contributions of this work are listed as follows:

1) We propose a new data format, the ST map, to rearrange the detection results and encode the spatio-temporal information in a point cloud type.
2) We propose novel ST-downsampling and ST-upsampling layers in our network to learn the correlation features for point-like objects in ST maps. And we first introduce the label assignment method to solve the random track ID prediction problem as a classification problem.
3) Our approach outperforms the state-of-the-art on the KITTI online benchmark in terms of MOTP, IDS, and FRAG. In addition, we demonstrate that the tracking results can be used to refine the detection results.

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

II. Related Works

A. Tracking-by-Detection MOT

Based on whether the tracking frameworks separate the detection from the trackers, they can be categorized as tracking-by-detection methods or tracking-before-detection methods. In this paper, we choose a tracking-by-detection framework, like most current algorithms. In data association, some approaches handled handcrafted object features [5], [6], while others extracted learned features from the detector network [7], [8], and some recent research has proved that only using bounding boxes can also achieve high tracking performance [9], [10]. In tracking-by-detection frameworks, the tracking performance is mainly based on the detection results. How to choose the detection threshold to balance the false-negative and false-positive detections is a key problem to be solved.

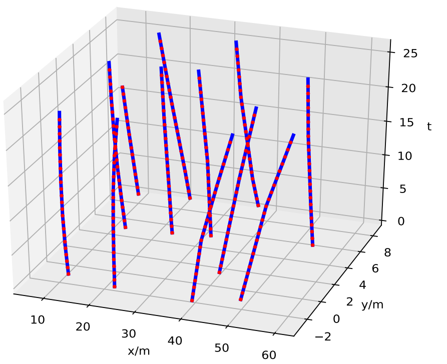

Fig. 1. **Visualization of ST map.** The points and lines represent the detected objects and their tracked trajectories, respectively. The t represents time, and the coordinates in the x-y plane represent the 2-D projected positions of the objects.

B. Single-Scan MOT

MOT problems can be classified based on their area of focus. According to the number of input frames, the algorithms can be single-scan or multi-scan. Multi-scan techniques [11], [12], [13] use an entire sequence or multiple frames of input to find the optimal trajectory assignment, while single-scan [14], [15], [16] methods only take two adjacent frames as input and associate the objects within them. In the data association of the single-scan MOT methods, the Kalman Filter [7], [8] and Gaussian [17] are the conventional methods applied for motion prediction. Convolutional Siamese networks [18], [19] are also used widely for association similarity computation, to predict the cost matrix of all object pairs and then the Hungarian algorithm [20] can be applied to obtain a solution.

SORT [7] and AB3DMOT [9] are two baseline methods for 2-D and 3-D tracking-by-detection single-scan MOT, which take as input the two adjacent frames, and finish the online and real-time tracking results. These two methods use off-the-shelf object detectors, with the Kalman Filter and Hungarian algorithm to obtain the trajectories. Meanwhile, PointTrackNet [15] is an end-to-end network for object detection and tracking, which takes as input single-scan point clouds and outputs the object movement for object tracking. Our proposed end-to-end network tries to learn the global and connectivity features in a batch of input detection data, rather than using Hungarian algorithm directly, to handle the tracklets autonomously.

C. Multi-Scan MOT

Multi-scan MOT takes as input a batch of frames to achieve better tracking performance by fusing long-term data. Various kinds of networks focus on the importance of temporal relations in a batch of data, such as Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM) [21] and Gated Recurrent Unit (GRU) [22]. LSTM and GRU are variants of RNNs, with LSTM solving the short-term memory problem of RNNs, and GRU is a more efficient variant of LSTM. Time-related networks are used widely in forecasting tasks like trajectory prediction [23], [24], motion planning [25], etc.

Spatio-temporal-based methods are another approach to merge long-term spatial information. Wang et al. [26] proposed a Bayesian and conditional random field-based framework in an ST map to solve the data association problem between frames. Spatio-temporal LSTM [27], which uses sequential data as input, was introduced to save long-term contexture. End-to-end network DiTNet [28] used a PointPillar-based detector to extract multi-frames detection features, and merged batches of features to learn the associated track IDs. However, the huge consumption of time and resources makes the algorithms difficult to implement in real time.

Our proposed framework uses bounding boxes to generate the ST map representation, and consolidate the detection results at all times to help us finish the data association task. Experiments show that our ST-TrackNet can also achieve comparable tracking performance with much faster speed using our bounding box-based features only.

III. The Proposed Approach

A. Approach Overview

The overview of our approach is shown in Fig. 2. It consists of two main components: 1) an ST map generator, and 2) our proposed ST MOT network, which is named ST-TrackNet.

The input to the ST map generator can be 2-D camera images, 3-D point clouds, or both. Using an object detector, the position and bounding box information of objects can be obtained from the raw sensor measurements. Then, we convert the object detection results into a 3-D ST map. Our ST-TrackNet treats the ST map as a point cloud, taking it as input, and it outputs the track ID of each detected object. The network mainly consists of an ST-downsampling module, an ST-upsampling module and a track assignment module. Note that the track assignment module is only used during training to associate the predicted trajectories with the ground-truth trajectories so that the training loss can be calculated.

B. Spatio-Temporal Map Generator

The offline object detector can provide 1) the position coordinates $\{\mathsf{p}_i^t = (x_i^t, y_i^t, z_i^t)\}_{i=1}^{N_t}$, where N_t represents the maximum number of objects at the current time t; and 2) the bounding boxes $\{b_i^t = (l_i^t, w_i^t, h_i^t, a_i^t, s_i^t)\}_{i=1}^{N_t}$, where l, w, h, a, and s are the length, width, height, orientation angle, and confidence score of a bounding box. With the position results obtained from a whole sequence, a sliding window is used to crop a batch of data on the time axis. Let T denote the length of frames in the ST Map (time window size) and Δt denote the step size of the sliding window. The object positions in this time window are denoted as $\{\mathsf{p}_i^t = (x_i^t, y_i^t, z_i^t)\}_{i\in[1,N_T],t\in[0,T-1]}$, where $N_T = N \cdot T$. To construct the ST map, we first drop the height information z of the position coordinates, because road objects (e.g., cars, pedestrians and cyclists) move on the 2-D road plane in most cases. Then we plot the 2-D coordinates of objects (x, y) from the whole sequence with time t as the ST map. The ST map can be viewed as a 3-D point cloud.

Our proposed tracking framework can handle batch-based or online 3-D object tracking tasks. The batch-based tracking method means that we can achieve the whole data first; and the online method means the tracker will utilize every online

Algorithm 1 Creation of ST Map From Detection Results

```
Input :
  Detections in time $t$:
  $\{\mathsf{Det}_i = (x_i, y_i, z_i, l_i, w_i, h_i, a_i, s_i)\}_{i=1}^{N_t}$,
  Detections in sequence from time 1 to $T_{seq}$: $\{\mathsf{Det}^t\}_{t=1}^{T_{seq}}$
Output:
  ST maps in the whole sequence: $M = \{P^t, F^t\}_{t=1}^{T_{seq}}$,
  Points in each ST map: $P = \{p_i\}_{i=1}^{N}$,
  Features of each point: $F = \{f_i\}_{i=1}^{N}$
1  $M \leftarrow \varnothing$, $P \leftarrow \varnothing$, $F \leftarrow \varnothing$
2  while $t + T \leq T_{seq}$ do
3      // Start to create t-th ST map: $M^t = \{P^t, F^t\}$
4      for $t \leq t_j \leq t + T$ do
5          // Add detections to $P^t$ and $F^t$
6          foreach $p_i^{t_j} \in \mathsf{Det}^{t_j}$ do
7              Add $(t_j - t, x_i^{t_j}, y_i^{t_j})$ to $P^t$
8              Add $(l_i^{t_j}, w_i^{t_j}, h_i^{t_j}, a_i^{t_j}, s_i^{t_j})$ to $F^t$
9          end
10     end
11     Add $M^t$ to $M$
12     $t = t + \Delta_t$    # Move forward $\Delta_t$ frames
13 end
```

input in real-time and then associate the new objects with the past trajectories. When our method is applied as an online method, every ST map will include the frames from $t_{current} - T$ to the current frame $t_{current}$, and the step of the sliding window should be 1. If it is applied as a batch-based method by contrast, the step size can be larger than 1 to speed up the tracking process. The pseudo-code of the ST map generator in a batch-based situation is described in Algorithm 1.

There are three main reasons that we need to rearrange the detections into an ST map. Firstly, the ST grouping layer in our ST-TrackNet uses the spatial and temporal relationship of objects in different frames to group the neighborhoods. The ST map can provide it so that the subsequent network architecture can learn from other fundamental frameworks like PointNet++ [29]. Secondly, the state prediction step in Algorithm 3 needs to fit quadratic curves in the X-t, Y-t, and Z-t plane, and uses the curve function to predict the current location. The ST map can provide convenience for collecting essential data. Lastly, the ST map representation is better for visualization and trajectory analysis, so it is worthwhile to spend a trivial amount of time creating the ST map for further usage.

C. ST-TrackNet

The aim of our ST-TrackNet is to predict the object track ID with confidence scores between different time layers of the ST map. As previously mentioned, our network mainly consists of an ST-downsampling module, an ST-upsampling module, and a track ID prediction module. The downsampling and upsampling modules help to merge the time-related features, and the track ID prediction module applies the softmax function to predict the tracking confidence score and track ID of each

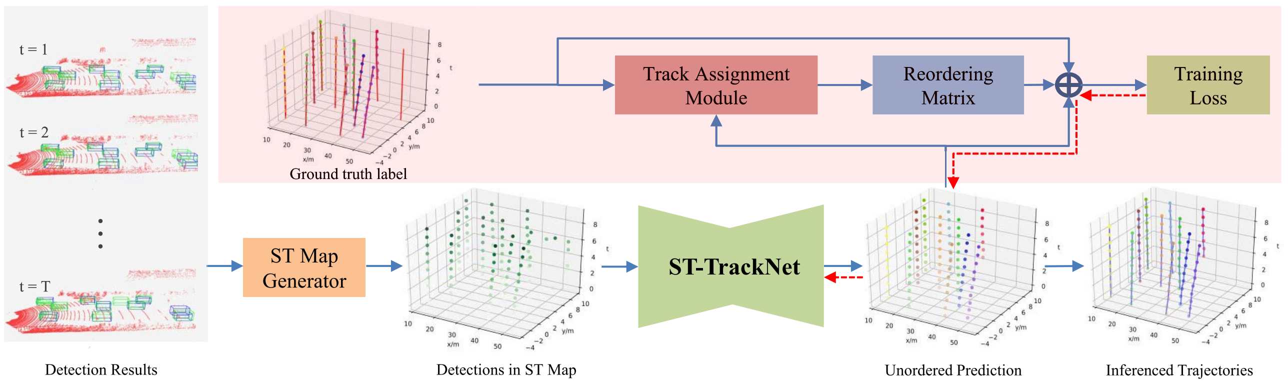

Fig. 2. **Overall architecture.** Given T frames' detection results in the grey shadow, which are cropped from the sequence by the time sliding window, the ST map generator generates the ST map, and the ST-TrackNet learns to predict each object's track ID. The modules in the red shadow are only used in the training process, to find the bipartite matchings between the predicted trajectories and the ground-truth trajectories. The red dotted lines show the backward data flow.

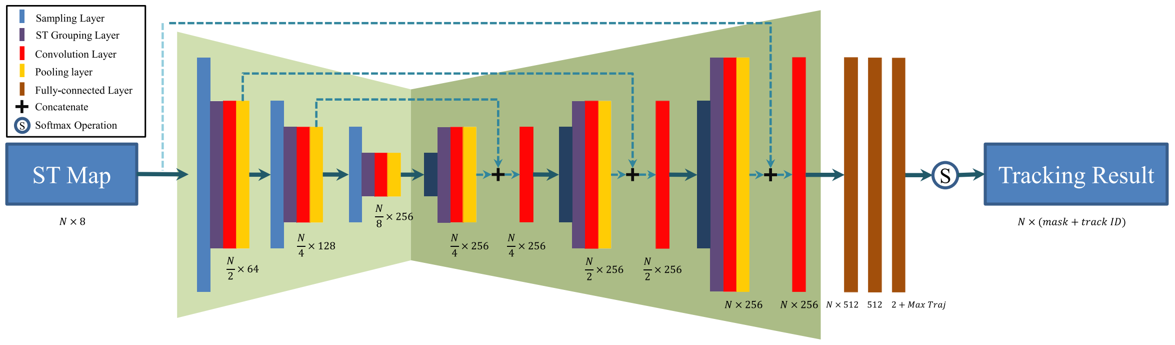

Fig. 3. **Architecture of ST-TrackNet.** In this figure, the blue dotted lines represent a skip connection, and $Max\ Traj$ means the maximum number of objects at each time frame. The tracking result includes each point's mask and track ID information.

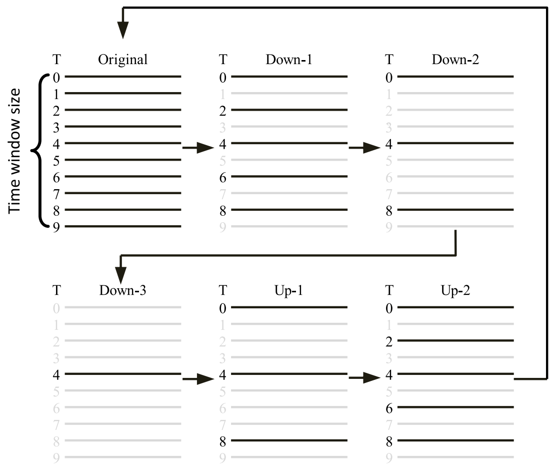

Fig. 4. **ST downsampling and upsampling layers in *ST Sampling* module.** In this example, we set the time window size to 10, and set three downsampling layers and three upsampling layers.

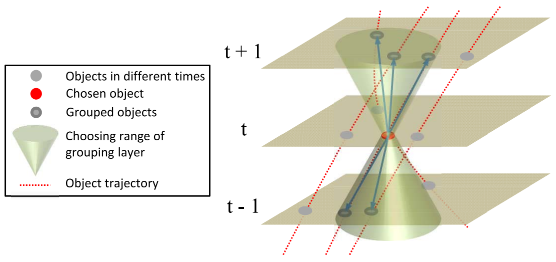

Fig. 5. ***ST Grouping* layer.** The chosen object groups the features of the objects inside the range of choice.

object. The inference architecture of ST-TrackNet is shown in Fig. 3. Our ST-TrackNet consists of three downsampling layers, three upsampling layers, and three fully-connected (FC) layers. The input of ST-TrackNet is a point cloud in the ST map which has size $N \times 8$, and the size of the output of the last FC layer is $N \times (2 + \textit{max trajectories number})$. The softmax operation is applied to generate the mask in [0, 1] and applied for each track ID in a time layer, because one object only has one track ID, and different points of the same object in one ST map should have the same track ID.

The ST-downsampling module consists of three sequential *Sampling-Grouping-Convolution* blocks. We adopt the point-cloud processing method proposed in PointNet [30], which directly takes as input the raw point clouds. The *Sampling* layer in each block aims to downsample the points from the ST map. The first *Sampling* layer downsamples the input point cloud $\{p_i^t\}_{t \in T_0}$ into $\{p_i^t\}_{t \in T_1}$, where T_0 and

T_1 are the set of different time layers, and $T_1 \subset T_0$. During the downsampling operation, we extract all the points in the interval time planes. For example, in this paper, we choose the time window size as 10. Then in the first downsampling operation, the points in time $\{1, 3, 5, 7, 9\}$ will be temporally dropped and saved, and points in time $\{0, 2, 4, 6, 8\}$ will be passed into the *ST Grouping* layer. The process of all *Sampling* layers in the three downsampling and upsampling operations is shown in Fig. 4.

Inspired by the K-Nearest Neighbors (KNN) algorithm, and considering the characteristics of our ST map and the tracking task, we propose an hourglass-like *ST Grouping* layer in Fig. 5. This layer aims to group the neighboring features of the downsampled or upsampled points. For one chosen point in the grouping layer, the other points in the same time plane are not grouped with this point. The reason is that there are not two or more objects that have the same track ID at the same time. In the local region grouped by the *ST Grouping* layer, we use the relative position as one of the inputs for distance computation. For each chosen point p, the relative position d is the Euclidean distance between its K neighbors: $\{d_i = p_i - p\}_{i\in[1,K]}$. Then the output of the first grouping layer has the size $N' \times K \times C$, where K is the number of grouped points, and C is the number of feature channels.

The output is next fed into the *Convolution* layer, which consists of several multi-layer perceptrons (MLPs), and aims to produce the features. A *pooling* layer is then set to merge the features from the gathered points to the chosen point. In this paper, we choose *max-pooling* method in this layer.

The ST-upsampling module likewise consists of three sequential blocks, *Interpolation-Grouping-Convolution*. The *Interpolation* layer in each block is designed to gradually restore the points which were dropped in the *Sampling* layers in the ST-downsampling modules. The reason for this restoration is that every original point needs a track ID. We adopt a similar hourglass-like grouping operation in the ST-upsampling module to that adopted in the ST-downsampling module, as shown in Fig. 5. In each *Grouping* layer, the features of the corresponding sampling layer are concatenated into the features after the *Convolution* layer in the ST-downsampling module through *skip connections*, with another shared MLP connected. Our proposed hourglass-like grouping operations help the network better extract the location-correlated features without interfering with other objects at the same time.

D. Training Process

1) Track Assignment Module: 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. Thus, we design a track assignment module to find the bipartite-matching between the predicted and ground-truth trajectories. The module takes as input the predicted trajectories as well as the ground-truth trajectories and assigns the most well-matched ground-truth trajectory ID to the predicted one.

We employ the Intersection over Union (IoU) as the cost metric to measure how well a pair of trajectories matches. It calculates the ratio for the number of overlapping points over 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 module is $B \times N \times (L+2)$, where B is the batch size, N is the number of points, and 2 is the size of the confidence score. Let $\{S_i, S_j\}$ denote a pair of trajectories, $C_{i,j} \in \mathbb{R}^{L\times\bar{L}}$ denote the cost for the matching, and $I \in \mathbb{R}^{L\times\bar{L}}$ denote the reordering matrix. The assignment problem can be formulated as a bipartite matching problem:

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

In this paper, we use the Hungarian algorithm [20] 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. The Hungarian algorithm is only applied in the training process, so our tracking network is end-to-end and the inference speed is much faster than the training speed.

2) Loss Functions: Our loss consists of a binary cross entropy loss $\mathcal{L}_{mask}$, a softmax IoU (SIoU) loss $\mathcal{L}_{SIoU}$, a triplet loss $\mathcal{L}_{tri}$, and a discriminative loss $\mathcal{L}_{dis}$:

$$\mathcal{L}_{all} = \mathcal{L}_{mask} + \mathcal{L}_{SIoU} + \mathcal{L}_{tri} + \mathcal{L}_{dis}. \tag{2}$$

$\mathcal{L}_{mask}$ is computed from the tracking confidence score of each point:

$$\mathcal{L}_{mask} = -s \cdot \log(\hat{s}) - (1-s) \cdot \log(1-\hat{s}), \tag{3}$$

where $s \in \{0, 1\}$ is the ground truth, 1 represents that the detection is true positive, 0 represents that it is false positive, and $\hat{s}$ is the prediction confidence score of each point.

With the associated tracking trajectories, we can calculate the SIoU with the following:

$$\mathcal{L}_{SIoU} = 1 - \frac{1}{L}\sum_{l=1}^{L} \frac{\sum_{n=1}^{N} f_n^l \cdot g_n^l}{\sum_{n=1}^{N} f_n^l + \sum_{n=1}^{N} g_n^l - \sum_{n=1}^{N} f_n^l \cdot g_n^l}, \tag{4}$$

where N is the number of points, $g_n^l \in \{0, 1\}$ represents the ground truth, and $f_n^l \in [0, 1]$ is the probability that point n is on the trajectory l, which is obtained by applying the softmax function on the trajectory class prediction for point n.

As our task resembles the instance segmentation for point clouds (i.e., point-wise trajectory ID labelling), we borrow the triplet loss and discriminative loss [31] that are widely used in instance segmentation for our task. These two losses encourage the features from the same tracklets to stay as close as possible to each other, while the features on different tracklets are separated as far as possible. Specifically, we choose the batch hard triplet loss:

$$\mathcal{L}_{tri} = \frac{1}{L}\sum_{l=1}^{L}\left\{\max_{p_a,p_b\in l} D[F_\theta(p_a), F_\theta(p_b)] - \min_{p_a\in l, p_c\notin l} D[F_\theta(p_a), F_\theta(p_c)] + \delta_{tri}\right\}_+, \quad (5)$$

where δ_{tri} is a margin parameter, $D(\cdot)$ is the distance function, $F_\theta(p)$ means the features of point p from the output of the last upsampling layer, p_a and p_b are points from the same tracking trajectory, p_a and p_c are points from different tracking trajectories, and the symbol $\{\}_+$ represents the ReLu function. The discriminative loss $\mathcal{L}_{dis}$ is calculated by:

$$\mathcal{L}_{dis} = \omega_{var}\cdot\mathcal{L}_{var} + \omega_{distant}\cdot\mathcal{L}_{distant} + \omega_{reg}\cdot\mathcal{L}_{reg}, \quad (6)$$

where different ω represent different weighting parameters, $\mathcal{L}_{var}, \mathcal{L}_{distant}$ and $\mathcal{L}_{reg}$ are respectively calculated by

$$\mathcal{L}_{var} = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{N_l}\sum_{n=1}^{N_l}\left[\|\mu_l - F_\theta(p_n)\| - \delta_v\right], \quad (7)$$

$$\mathcal{L}_{distant} = \frac{1}{L(L-1)}\sum_{l_m,l_n\in L}\left[2\delta_d - \left\|\mu_{l_m} - \mu_{l_n}\right\|\right]_+^2, \quad (8)$$

$$\mathcal{L}_{reg} = \frac{1}{L}\sum_{l\in L}\|\mu_l\|, \quad (9)$$

where N_l is the number of points in the trajectory l, μ_l is the average embedding in the trajectory l, μ_{l_m} and μ_{l_n} represent the average embeddings for trajectories l_m and l_n, and δ_v and δ_d are margin parameters. In our experiments, we set $\alpha = 1.0$, $\beta = \gamma = 0.1$, $\omega_{var} = \omega_{reg} = 1.0$, $\omega_{distant} = 0.001$, and the margin parameters as 1.0.

E. Inference Process

During the inference process, our network takes as input the ST map and outputs the track ID for each point in a batch. By moving the aforementioned sliding window step by step, we first find the track ID in each batch, then use a post-processing module in Algorithm 2 to make the network more robust, and finally connect all track IDs and refine the detection results for the whole sequence in Algorithm 3.

Our ST-TrackNet may make some wrong predictions, for example, two tracklets being predicted to have the same ID or one track having a small number of points assigned the wrong ID. Thus, in the post-processing in Algorithm 2, we use the small RANSAC [32] algorithm to find the best fitting tracklets, and apply rules (maximum acceleration, maximum speed and breaking times) to ensure the tracklet is valid. In this way, the network will be more robust to different situations. In this paper, we use RANSAC{$\#NUM$} to name the RANSAC algorithm with a particular number ($\#NUM$) of iterations. Since the number of points fed into the RANSAC algorithm and the number of iterations are small, the time cost of the algorithm can be ignored. The details of the Kalman filter used in Algorithm 3 are the same as in [9]. The objects P in the two algorithms have all detection and tracking features, $\{t, x, y, z, w, h, l, \theta, detection\ score, tracking\ score\}$.

Algorithm 2 Post-Processing Algorithm to Refine the ST-TrackNet Prediction Results

```
Input  : ST map location $P = \{p_i^t\}_{i\in[1,N_T],t\in[1,T]}$,
         predicted track ID and confidence score of each
         point $ID = \{[ID_i^t, score_i^t]\}_{i\in[1,N_T],t\in[1,T]}$,
         maximum number of tracklets $L$.
Output: Tracklets set $S = \{P, ID\}$.
1  $S \leftarrow \varnothing$
2  for $i <= L$ do
3      Find the largest tracklet $S_i = \{P_i, ID_i\}, ID_i == i$
4      $P \leftarrow P_i$
5      while $S_i \neq \varnothing$ do
6          RANSAC20 get valid tracklet
           $S_v = \{P_v, ID_v\}, P_v \subset P$
7          if Success then
8              $S_i \leftarrow S_i - S_v,\ P \leftarrow P - P_v,\ S \leftarrow S + S_v$
9          else
10             break
11         end
12     end
13 end
14 Output $S$.
```

IV. Experimental Results and Discussions

A. Datasets

The first dataset we use is the object tracking benchmark dataset from KITTI [33], which has 21 sequences with ground truth. The second is the simulation dataset generated from CARLA [34]. Our approach is trained on the CARLA dataset, and quantitatively evaluated on CARLA and the KITTI dataset. Moreover, we qualitatively demonstrate our method on sequences which are recorded in crowded environments. We also upload our test results on the 29 test sequences from KITTI to the online benchmark. Since the KITTI dataset mainly uses the vehicle class for validation, we only track the Car and Van categories on KITTI. In the CARLA dataset, we set the number of pedestrians to 30 and the number of vehicles to 20 in the settings.

B. Evaluation Metrics

The CLEAR MOT metrics are used widely for evaluating detection and tracking accuracy. The first metric is Multiple Object Tracking Accuracy (MOTA):

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}, \quad (10)$$

where m_t, fp_t and mme_t are the numbers of misses, false positives and mismatched objects at time t respectively. The second metric is Multiple Object Tracking Precision (MOTP):

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}, \quad (11)$$

where d is the metric distance between the detection and ground truth bounding boxes for matched pairs and c is the total number of matches. These two metrics help us to estimate

Algorithm 3 Online Update Tracklets in the Sequence Using Predicted ST Map Result

```
Input  : ST map tracklets $\{S_t = \{[P_i, ID_i]\}_{i\in[1,L]}\}_{t>0}$.
Output: Tracklets set $Trk = \{[P_j, ID_j]\}_{j>0}$
1  $Trk \leftarrow \varnothing$
2  for Online update $t$ do
3      for valid $i <= L$ do
4          # Compose tracklet $s_i = [P_i, ID_i] \in S_t$
5          Search $P_i$ in $P_j \in Trk$
6          if have same point in $[P_j, ID_j] \in Trk$ then
7              # refine current detection using previous
               3 detections
8              Find last 3 points in this tracklet.
9              State Prediction: Fitting quadratic curves in
               XYZ plane and predict current values.
10             State Update: Kalman Filter to update current
               detection.
11             Add current point into $[P_j, ID_j] \in Trk$ with
               $ID_i = ID_j$.
12         else
13             # newborn tracklet
14             Assign new track ID to $s_i$.
15             $Trk \leftarrow Trk + s_i$.
16         end
17     end
18 end
19 Output $Trk$.
```

the overall performance of the detection and tracking pipeline. Besides these metrics, Mostly Tracked (MT), Mostly Lost (ML), ID Switches (IDS) and FRAGmentation (FRAG) can provide more 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 fragmentations and ID_switches.

For network validation on each window, the IoU for each class is chosen as one of the evaluation metrics. After we reorder the predicted track IDs to pair with the ground truth track IDs, a higher $mIoU$ means better performance of the network. For point m and track class n,

$$mIoU = \frac{1}{L}\sum_{n=1}^{L}\frac{TP_n}{TP_n + FP_n + FN_n},$$

where TP_n, FP_n and FN_n mean the number of true-positive, false-positive and false-negative points inside the track class n. This matrix shows the most intuitive evaluation quality of the network.

C. Network Settings and Details

Considering misdetections like false-positive results, we set the number of points in each time stamp to $N = 90$ to include all detection results in the ST map. If the number of detection results is less than the number of points, we fill the points with $\{t, 0, 0\}$. We set the maximum number of trajectories in each time stamp as $L = 60$. The time window size T is set to 10 and the sliding window step size Δt is set to 4 in our experiments, since the all detection results of the datasets can be achieved before tracking. If our method is applied in an online situation, the step size Δt should be set to 1. During the training process, we get the labels of the objects in the sequence by fitting the label trajectories to conics in the x-y planes using the least-squares method. The detection results that are close to these curves are chosen as positive detections and assigned the curve label. The number of grouped points K is set to 6. The S_1 downsampling layer setting in Tab. I is used in the *ST Sampling* layers.

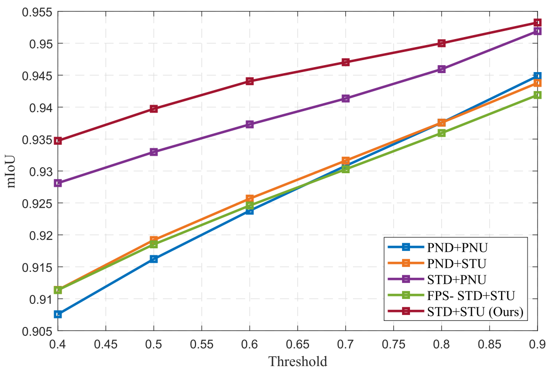

Fig. 6. **Comparison of sampling methods with detection confidence thresholds.** Our *ST Sampling* method helps networks stay relatively steady as the threshold changes, and shows better performance than other baseline methods.

D. Ablation Study

1) Sampling Methods With Detection Threshold: In our proposed framework, the ST map is considered to be a point cloud, and is fed into the point cloud-based network. In this section, we compare our proposed ST downsampling layers and ST upsampling layers with the PointNet++ based Set Abstract layers (downsampling layers) and Set Upconv layers (upsampling layers), to distinguish the tracking problem in the ST map from the common 3-D point cloud environment. Specifically, our *ST Sampling* layer is compared with the FPS method in all downsampling and upsampling layers.

The five networks in the comparison, which have different downsampling and upsampling layers, are described as follows: “PN” means PointNet++ based, “ST” means our ST based, “D” and “U” mean down- and up-sampling respectively. “FPS-” means using FPS in the downsampling layers.

Fig. 6 shows the comparison of the downsampling and upsampling methods with different detection confidence thresholds. Since our proposed network based on the ST map has the ability to refine the detection results by eliminating redundant detections, all networks stay relatively steady as the threshold changes. Our proposed ST-downsampling and ST-upsampling layers show better performance than the PointNet++ layers in feature extraction and interpolation, which means the points in the same time stamp do interfere with the track feature extraction. Comparing our *ST Sampling* layer with FPS, we can find that the performance becomes worse when the FPS layers replace the *ST Sampling* layers.

TABLE I
DIFFERENT DOWNSAMPLING LAYER SETTINGS

Name	Stride	Down-1	Down-2	Down-3
S_0	0	{0, 1, 2, ..., 9}	{0, 1, 2, ..., 9}	{0, 1, 2, ..., 9}
S_1	1	{0, 2, 4, 6, 8}	{0, 4, 8}	{4}
S_2	2	{0, 3, 6, 9}	{0, 6}	{6}
S_3	3	{0, 4, 8}	{0, 8}	{8}
S_4	1	{0, 2, 4, 6, 8}	{0, 8}	-
S_5	1	{0, 2, 4, 6, 8}	-	-

This comparison reveals that in such a time-related ST map environment, sampling from the time dimension is better than sampling from the spatial dimension.

2) Other Network Settings: After demonstrating that the *ST Sampling* method is better than the other point-based sampling methods, we conduct several experiments to study the performance of ST-TrackNet with different network settings. The main factors of influence of the proposed network include the time window size, downsampling layer setting, and the utility of each item in the loss function. Uniform experiments are conducted to illustrate the impact of each factors, and the comparison is shown in Tab. III. The mIoU metric represents the network's performance, and the MOTA, IDS, and FRAG metrics represent the tracker's performance. The detection confidence threshold in this experiment is set to 0.4.

Our multi-scan-based network uses the detection results in a continuous time period. The size of the sliding window, which is the input sequence length, decides the ability to compose the short-term occlusions or misdetections. In the experimental results #1–6 in Tab. III, the sampling layer stride is set as 1, all losses in Eq. 2 are used, and the time window size is varied from 3 to 12. The results of the metrics show that, when the mIoU is maintained at similar levels, the longer the time window size, the better the performance of the tracker. And when the time window size is longer, the mIoU performance will be lower. As we mentioned before, a longer time window size will cover more misdetections in a short time duration, and our tracker can track objects in a long time period. However, the mIoU metric, which directly represents the performance of the network, will gradually decrease as the window size increases. The overall results show that when the time window size is 4, the network accuracy is highest; and when the time window size is 10, the tracker is the best. This is the reason why we choose the time window size to be 10 in our base experiment.

Another possible key factor is the stride and number of layers in *ST Sampling*. The relationship of the downsampling layer stride and number of layers with the other layers while downsampling is shown in Tab. I, with the mean of Down-1 to 4 illustrated in Fig. 4. In experimental results #1 and #7–11 in Tab. III, the time window size is set to 10 and all losses in Eq. 2 are used. From these results, we can find that result #1 is better than result #7, and similar to results #8–9. Because in experiment #1, all layers are calculated in the four layers all along the network, the comparison of result #1 with result #7 reveals that the *ST Sampling* process can boost the mIoU from 0.9131 to 0.9295, while the comparison of results #10 and #11 (which have two and one downsampling layer respectively) with result #1 shows that the "deeper" the sampling process, the better the network performance. On this basis, our proposed network framework chooses to

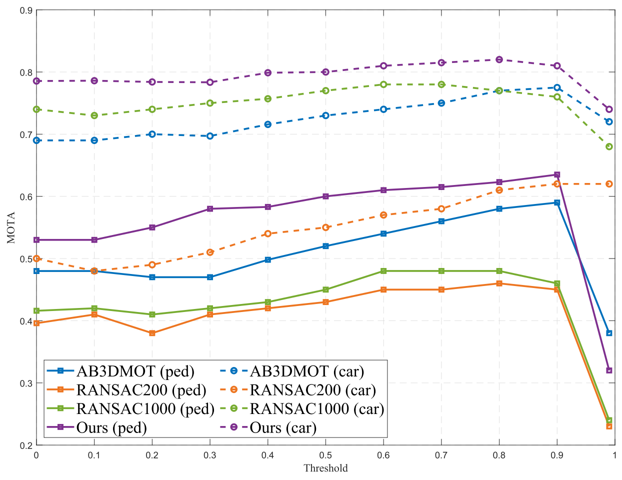

Fig. 7. **Comparison of trackers under different recall confidence thresholds.** Our method's performance stays stable in all recall thresholds, and it can achieve more accurate tracking results in a low SNR situation.

TABLE II
COMPARATIVE RESULTS OF DIFFERENT DETECTORS WITH REFINEMENT MODULE

	MOTA↑	MOTP↑	MT↑	ML↓
PointPillar w. refinement	0.7471	0.7866	0.7510	0.133
PointPillar w/o. refinement	0.7420	0.7842	0.7510	0.133
SECOND w. refinement	0.7763	0.8304	0.7841	0.127
SECOND w/o. refinement	0.7709	0.8259	0.7841	0.127
PointRCNN w. refinement	0.7895	0.8449	0.7801	0.108
PointRCNN w/o. refinement	0.7862	0.8393	0.7801	0.108

TABLE III
COMPARATIVE RESULTS OF DIFFERENT NETWORK SETTING IN ABLATION STUDY. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS

#	Network Setting	mIoU↑	MOTA↑	IDS↓	FRAG↓
1	$T = 10, S = S_1, \mathscr{L}_{all}$	0.9295	**0.8043**	**26**	**79**
2	$T = 12, S = S_1, \mathscr{L}_{all}$	0.9047	0.7754	**26**	81
3	$T = 8, S = S_1, \mathscr{L}_{all}$	0.9391	0.7986	**26**	**79**
4	$T = 6, S = S_1, \mathscr{L}_{all}$	0.9456	0.7617	31	86
5	$T = 4, S = S_1, \mathscr{L}_{all}$	**0.9587**	0.7448	48	92
6	$T = 3, S = S_1, \mathscr{L}_{all}$	0.9440	0.6120	72	146
7	$T = 10, S = S_0, \mathscr{L}_{all}$	0.9131	0.7774	**26**	83
8	$T = 10, S = S_2, \mathscr{L}_{all}$	0.9264	0.8031	**26**	84
9	$T = 10, S = S_3, \mathscr{L}_{all}$	0.9247	0.7953	27	83
10	$T = 10, S = S_4, \mathscr{L}_{all}$	0.9196	0.7708	27	86
11	$T = 10, S = S_5, \mathscr{L}_{all}$	0.9071	0.7693	**26**	83
12	$T = 10, S = S_1, \mathscr{L}^{-}_{SIoU}$	0.8106	0.7084	58	97
13	$T = 10, S = S_1, \mathscr{L}^{-}_{tri}$	0.8749	0.7645	35	93
14	$T = 10, S = S_1, \mathscr{L}^{-}_{dis}$	0.8925	0.7753	32	92
15	$T = 10, S = S_1, \mathscr{L}^{-}_{tri,dis}$	0.7766	0.6950	75	121

TABLE IV

COMPARATIVE RESULTS OF EVALUATION METRICS WITH DIFFERENT CTS FOR DIFFERENT TRACKERS ON KITTI **CAR** VALIDATION DATASET. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS

Methods	$CT = 0.0$						$CT = 0.3$					
	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓
RANSAC200 [32]	0.5077	0.8470	0.2836	0.3085	78	441	0.5103	0.8464	0.2836	0.3014	72	416
RANSAC1000 [32]	0.7404	0.8447	0.6099	0.0762	212	626	0.746	0.8421	0.6241	0.0762	224	622
AB3DMOT [9]	0.6977	0.8472	0.7553	**0.0319**	**8**	168	0.6977	0.8426	0.7553	**0.0319**	**8**	168
Ours	**0.7855**	**0.8583**	**0.8103**	0.039	22	**112**	**0.7834**	**0.8542**	**0.8103**	0.0425	14	**117**
Methods	$CT = 0.6$						$CT = 0.9$					
	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓
RANSAC200 [32]	0.578	0.8414	0.3351	0.2251	121	436	0.6278	0.8473	0.3812	0.1631	138	518
RANSAC1000 [32]	0.7806	0.8412	0.6666	0.062	287	647	0.7664	0.8491	0.5993	0.078	314	662
AB3DMOT [9]	0.7431	0.8478	0.7322	**0.0319**	**6**	264	0.7751	0.8434	0.6702	**0.0407**	**5**	384
Ours	**0.8108**	**0.8491**	**0.8085**	0.0443	20	**118**	**0.8138**	**0.8535**	**0.7287**	0.0567	11	**92**

TABLE V

COMPARATIVE RESULTS OF EVALUATION METRICS WITH DIFFERENT CTS FOR DIFFERENT TRACKERS ON KITTI **PEDESTRIAN** VALIDATION DATASET. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS

Methods	$CT = 0.0$						$CT = 0.3$					
	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓
RANSAC200 [32]	0.3961	0.6702	0.1782	0.099	648	1127	0.4101	0.6709	0.2178	0.1188	551	1041
RANSAC1000 [32]	0.4164	**0.6714**	0.2772	0.0792	661	1139	0.4155	0.6713	0.3168	0.099	659	1134
AB3DMOT [9]	0.4799	0.6699	0.4356	0.0693	**72**	435	0.4799	0.6699	0.4356	0.0693	**72**	435
Ours	**0.5362**	0.6701	**0.4358**	**0.0495**	99	**417**	**0.5841**	**0.6823**	**0.4554**	**0.0495**	104	**384**
Methods	$CT = 0.6$						$CT = 0.9$					
	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓
RANSAC200 [32]	0.4521	0.67	0.2375	0.099	631	1087	0.4566	0.6703	0.1782	0.1584	575	1010
RANSAC1000 [32]	0.4833	0.6705	0.3168	0.0891	637	1114	0.4686	0.6696	0.2079	0.1386	608	1065
AB3DMOT [9]	0.5446	0.6699	0.4059	0.0693	**72**	451	0.5935	0.67	0.3465	0.1188	**64**	484
Ours	**0.6104**	**0.6836**	**0.4455**	**0.0594**	102	**401**	**0.6257**	**0.6824**	**0.3666**	**0.0891**	92	**396**

TABLE VI

COMPARATIVE RESULTS OF EVALUATION METRICS ON KITTI **CAR** TEST DATASET. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS

Methods	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓	FRAG↓	Runtime↓	Environment
TuSimple [16]	86.62%	83.97%	72.46%	6.77%	293	501	0.6s	1 core @ 2.5 Ghz
PC3T [36]	**88.81%**	84.26%	**80.00%**	8.46%	225	201	**0.01s**	/
EagerMOT [37]	87.82 %	85.69 %	76.15 %	**2.46 %**	239	390	**0.01s**	4 cores @ 3.0 Ghz
AB3DMOT [9]	83.49 %	85.17 %	67.08 %	11.38 %	126	254	**0.01s**	1 core @ 2.5 Ghz
mono3DT [38]	84.28 %	85.45 %	73.08 %	2.92 %	379	573	0.03s	GPU @ 2.5 Ghz
DiTMOT [28]	84.53 %	84.36 %	74.77 %	12.77 %	101	210	0.08s	1 core @ >3.5 Ghz
MOTSFusion [39]	84.83%	85.21%	73.08%	**2.77%**	275	759	0.44s	GPU
mmMOT [40]	84.77%	85.21%	73.23%	**2.77%**	284	753	0.02s	GPU @ 2.5 Ghz
aUToTrack [41]	82.25%	80.52%	72.62%	3.54%	1025	1402	**0.01s**	1 core @ >3.5 Ghz
FANTrack [13]	77.72%	82.33%	62.62%	8.77%	150	812	0.04s	8 cores @ >3.5 Ghz
Complexer-YOLO [14]	75.70%	78.46%	58.00%	5.08%	1186	2092	**0.01s**	GPU @ 3.5 Ghz
Point3DT [15]	68.24%	76.57%	60.62%	12.31%	111	725	0.05s	1 core @ >3.5 Ghz
mbodSSP* [42]	72.69%	78.75%	48.77%	8.77%	114	858	**0.01s**	1 core @ 2.7 Ghz
ST-TrackNet (Ours)	82.24%	**85.78%**	74.31%	9.54%	**13**	**236**	**0.01s**	1 core @ >3.5 Ghz

use three downsampling and upsampling layers and sets the downsampling layer stride as 1 in the *ST Sampling* process.

In experiments #12–15, we respectively remove the loss items $\mathcal{L}_{SIoU}$, $\mathcal{L}_{tri}$, $\mathcal{L}_{dis}$, and $\mathcal{L}_{tri}$ with $\mathcal{L}_{dis}$ in $\mathcal{L}_{all}$ in Eq. 2. For example, $\mathcal{L}^{-}_{SIoU}$ means the SIoU loss is removed. Our ST-TrackNet decreases the mIoU performances to 0.8106 for $\mathcal{L}_{all} - \mathcal{L}_{SIoU}$, 0.8749 for $\mathcal{L}_{all} - \mathcal{L}_{tri}$, 0.8925 for $\mathcal{L}_{all} - \mathcal{L}_{dis}$, and 0.7766 for $\mathcal{L}_{all} - \mathcal{L}_{tri} - \mathcal{L}_{dis}$. Since the triplet loss and discriminative loss are used for clustering the features of the same object at different times, the spatial and temporal features of the same object over a period of time should be similar. From experiments #13–15, we can find that these two loss items are similar for the network training, but they are necessary to cluster the features without track ID limitation. Meanwhile, the SIoU loss helps the network to output the track ID directly, which is unique at one moment and ranges from 0 to a maximum track ID in that time period.

In the dataset preparation, we augment the CARLA dataset to simulate the different detection results with different CTs and effects. All networks are trained using the same training dataset and are tested on the detection results from PointRCNN [35] with different network settings.

3) Different Detectors With Refinement Module: In the inference process of our framework, the Kalman filter and quadratic curve fitting in the refinement module helps refine the current detection results using the last three objects in

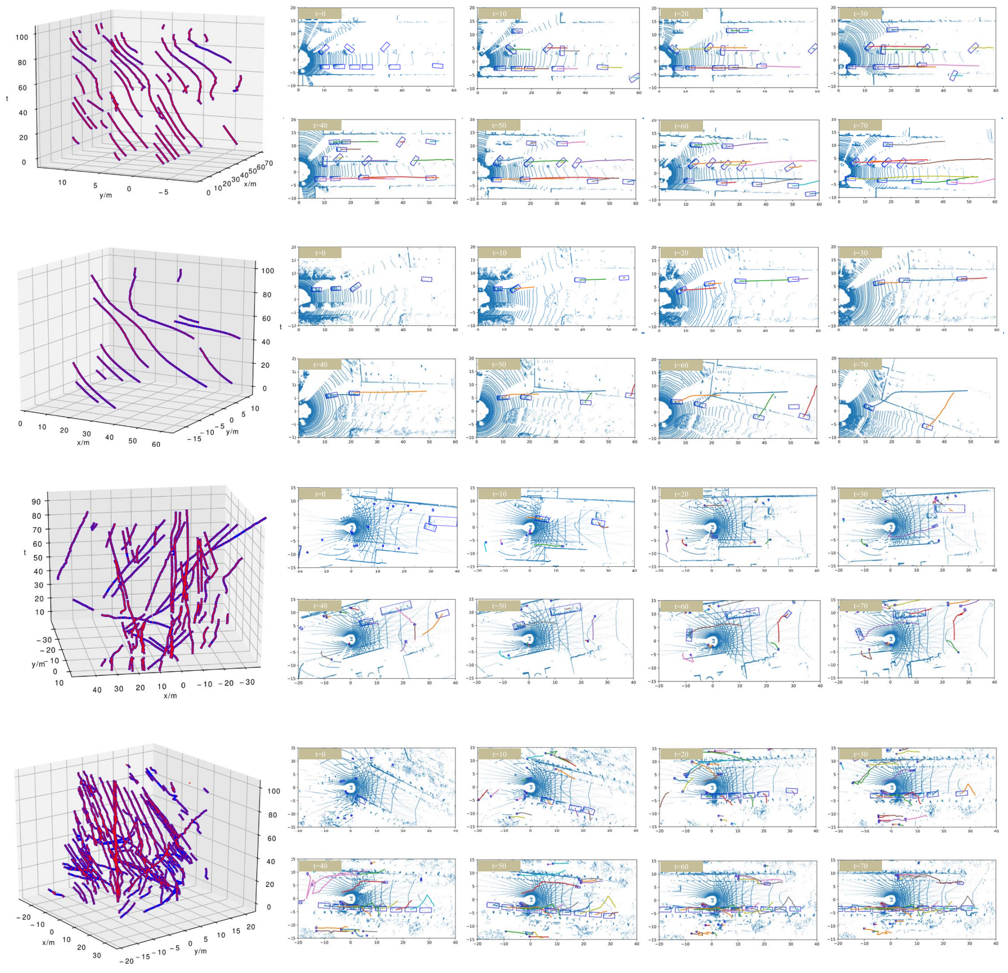

Fig. 8. **Qualitative results.** Top two rows are the visualization of tracking results of sequence 0 and sequence 3 on KITTI test dataset. And bottom two rows show two different scenes in Foxconn factory. The leftmost column is the detection results and trajectories in ST map, the detections are red points and trajectories are blue lines. The right most four columns are the bird's eye view of trajectories of Lidar data at different time stamps.

one tracklet. Tab. II shows the comparison results of our tracking-by-detection framework with different detectors with or without the refinement module. The experiments are using the whole 21 sequences on KITTI training dataset. When the framework does not the refinement module, the Algorithm 3 changes line 9-11 to use the original detections rather than the predicted and updated detections. When the last three objects has a fragmentation, which means missing one or more detections in the past, the complementary detection will use the quadratic curve fitting results directly.

From Eq. 10 and 11 we can find that the detection accuracy mainly inferences the MOTP metric, and the MT and ML metrics are more robust when the detection results are modified a little. The MOTP metric in Tab. II illustrates that the precision of the detections in the whole sequence will improve with our refinement module. The relatively similar MOTA results between different detectors show that our framework is robust in terms of detection results.

E. Comparative Results

In this section, we compare our method with the other three baselines RANSAC200 [32], RANSAC1000 [32], and AB3DMOT [9] on the KITTI validation dataset. Also, we show the comparative results of all evaluation metrics on the KITTI test dataset with the state-of-the-art methods.

For tracking-by-detection methods, the performance of the detector is one of the key factors in the tracking performance. Thus we compare our method with other tracking-by-detection methods which have the same detection results. AB3DMOT [9] uses off-the-shelf 3-D bounding box detection results, with a 3-D Kalman filter in the motion prediction step and the Hungarian algorithm in the data association step, to finish accurate and real-time tracking. We choose AB3DMOT as the single-scan tracking-by-detection baseline and use PointRCNN as the detection method for both AB3DMOT and our method. Using the same detector guarantees that the comparison focuses on the data association and tracking. RANSAC [32] is used here to generate the trajectories directly in the ST map. It makes sense to use RANSAC as a conventional trajectory fitting method for comparison. RANSAC200 means we iterate 200 times in the fitting process in each time window, while RANSAC1000 means iterating 1000 times. The number of iterations depends on the complexity of the detection results. More iterations are needed to find the suitable trajectories in a denser object environment. In the KITTI validation dataset, when the number of iterations is larger than 1000, the tracking results have very limited changes.

Tab. IV and Tab. V show the comparative results of the evaluation metrics with different CTs for the different trackers. Tab. IV evaluates the tracking performance on cars and vans, and Tab. V evaluates the pedestrians. From the results we can find that, firstly, our proposed network outperforms the others on MOTA by a remarkable margin, which means the overall tracking performance is much better than that of the others. Secondly, since we choose the same detection results before tracking, the performance on MOTP is close between the methods. But our network still improves on the others by above 1%, which means the tracking can not only choose the true-positive detections but also helps refine the detection results. Considering the identity switches and the trajectory fragmentation, our multi-scan method helps to improve the completeness of the tracker. We further draw the comparative results of MOTA in Fig. 7, from which we can find that our performance stays stable in all recall thresholds, and we can achieve more accurate tracking results in a low SNR situation.

Tab. VI shows the comparative results of the evaluation metrics on the KITTI test dataset (Car). It reveals our method's competitive results on MOTA, while we can also see that, on the MOTP, IDS, and FRAG metrics, our method outperforms all the others on the KITTI benchmark. The running time of 0.01 seconds shows that our tracking framework is real-time, which means that if we first have the detection results, we can track them in real-time.

F. Qualitative Results

Fig. 8 demonstrates the qualitative results on the KITTI test dataset and our own LiDAR dataset from an unmanned vehicle in a factory environment. This part of the dataset has no public labels, and our network still shows good results. We choose to visualize the tracking results of sequences 0 and 3 on KITTI, and two scenes from our dataset. The bird's-eye-view of the trajectories from Lidar data at three different time stamps are shown in the right-most four columns in the figure. We find only one IDS in sequence 0 on KITTI and fairly good continuity on our dataset. These qualitative results demonstrate the practicality and robustness of our approach. In a total 29 sequences of the KITTI test dataset, our network only has 13 IDSs in all.

V. Conclusion and Future Work

In this paper, we proposed a novel tracking-by-detection tracker framework, which includes an ST map generator and an ST MOT network, ST-TrackNet. The ST map generator takes the detection results in a sequence as input, and outputs the point cloud in the ST map. The 3-D detection results can be obtained by any object detector. Our proposed ST-TrackNet has the ability to output the track ID of each object directly, without the data association step required in other trackers. It consists of three ST downsampling layers, three ST upsampling layers, and three FC layers. In the training process, we designed a track assignment module to find the matchings between the predicted trajectories and the ground-truth trajectories, to solve the problem that the track IDs are random and unordered. From the experiment results, we found that our *ST Sampling* method is better than the other point-based sampling methods. Our method's tracking performance stays stable in all recall thresholds, and it can achieve more accurate tracking results in a low SNR situation. On the open KITTI car test dataset, our approach outperforms the others on the KITTI benchmark in terms of both MOTP and IDS. However, a number of issues still need to be solved in the future. For example, the features fed into the network are insufficient, because we only choose the position and bounding box information as the input features. In the future, we will collect the features of each object inside the detector, and provide them to our neural network, which could improve our method's performance.

References

[1] S. Wang, J. Jiao, P. Cai, and L. Wang, "R-PCC: A baseline for range image-based point cloud compression," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 10055–10061.

[2] J. Jiao, H. Ye, Y. Zhu, and M. Liu, "Robust odometry and mapping for multi-LiDAR systems with online extrinsic calibration," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 351–371, Feb. 2022.

[3] L. Xiong et al., "G-VIDO: A vehicle dynamics and intermittent GNSS-aided visual-inertial state estimator for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 63, no. 8, pp. 11845–11861, Aug. 2022.

[4] Z. Fu, L. Xiong, Z. Qian, B. Leng, D. Zeng, and Y. Huang, "Model predictive trajectory optimization and tracking in highly constrained environments," *Int. J. Automot. Technol.*, vol. 23, no. 4, pp. 927–938, Aug. 2022.

[5] J. Choi, S. Ulbrich, B. Lichte, and M. Maurer, "Multi-target tracking using a 3D-LiDAR sensor for autonomous vehicles," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 881–886.

[6] S. Song, Z. Xiang, and J. Liu, "Object tracking with 3D LiDAR via multi-task sparse learning," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, Aug. 2015, pp. 2603–2608.

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

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

[9] X. Weng, J. Wang, D. Held, and K. Kitani, "3D multi-object tracking: A baseline and new evaluation metrics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10359–10366.

[10] H. Wu, W. Han, C. Wen, X. Li, and C. Wang, "3D multi-object tracking in point clouds based on prediction confidence-guided data association," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5668–5677, Jun. 2022.

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

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

[13] 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)*, Jun. 2019, pp. 1426–1433.

[14] M. Simon et al., "Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2019, pp. 1–10.

[15] 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," *IEEE Robot. Autom.*, vol. 5, no. 2, pp. 3206–3212, Apr. 2020.

[16] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3029–3037.

[17] T. Hirscher, A. Scheel, S. Reuter, and K. Dietmayer, "Multiple extended object tracking using Gaussian processes," in *Proc. 19th Int. Conf. Inf. Fusion (FUSION)*, Mar. 2016, pp. 868–875.

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

[19] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5695–5703.

[20] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, Mar. 1955.

[21] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[23] K. Saleh, M. Hossny, and S. Nahavandi, "Long-term recurrent predictive model for intent prediction of pedestrians via inverse reinforcement learning," in *Proc. Digit. Image Comput., Techn. Appl. (DICTA)*, Dec. 2018, pp. 1–8.

[24] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 961–971.

[25] P. Cai, Y. Sun, Y. Chen, and M. Liu, "Vision-based trajectory planning via imitation learning for autonomous vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 2736–2742.

[26] 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 (ROBIO)*, Dec. 2017, pp. 2521–2526.

[27] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio–temporal LSTM with trust gates for 3D human action recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 816–833.

[28] S. Wang, P. Cai, L. Wang, and M. Liu, "DiTNet: End-to-end 3D object detection and track ID assignment in spatio–temporal world," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3397–3404, Apr. 2021.

[29] 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. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[30] 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. Vis. Pattern Recognit.*, Jun. 2017, pp. 652–660.

[31] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," 2017, *arXiv:1708.02551*.

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

[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[34] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, 2017, pp. 1–16.

[35] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.

[36] A. Kim, A. Ošep, and L. Leal-Taixé, "EagerMOT: 3D multi-object tracking via sensor fusion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2021, pp. 11315–11321.

[37] H.-N. Hu et al., "Joint monocular 3D vehicle detection and tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 5390–5399.

[38] J. Luiten, T. Fischer, and B. Leibe, "Track to reconstruct and reconstruct to track," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1803–1810, Apr. 2020.

[39] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust multi-modality multi-object tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2365–2374.

[40] K. Burnett, S. Samavi, S. Waslander, T. Barfoot, and A. Schoellig, "AUToTrack: A lightweight object detection and tracking system for the SAE AutoDrive challenge," in *Proc. 16th Conf. Comput. Robot Vis. (CRV)*, May 2019, pp. 209–216.

[41] 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. Vis. (ICCV)*, Dec. 2015, pp. 4364–4372.

Sukai Wang (Member, IEEE) received the bachelor's degree in measurement control and instrument science from Zhejiang University in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Robotics Institute, The Hong Kong University of Science and Technology (HKUST), Hong Kong, China. His research interests focus on multiple object detection and tracking, point cloud compression, and deep learning on autonomous driving.

Yuxiang Sun (Member, IEEE) received the bachelor's degree from the Hefei University of Technology, Hefei, China, in 2009, the master's degree from the University of Science and Technology of China, Hefei, in 2012, and the Ph.D. degree from The Chinese University of Hong Kong, Shatin, Hong Kong, in 2017. He is currently a Research Assistant Professor with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong. Prior to that, he was a Research Associate at the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. His current research interests include autonomous driving, deep learning, robotics and autonomous systems, and semantic scene understanding. He serves as an Associate Editor of IEEE ROBOTICS AND AUTOMATION LETTERS, IEEE International Conference on Robotics and Automation, and IEEE/RSJ International Conference on Intelligent Robots and Systems.

Zheng Wang (Senior Member, IEEE) received the B.Eng. degree from Tsinghua University, China, the M.Sc. degree from the Imperial College, U.K., and the D.-Ing. degree from the Technical University of Munich, Germany. He served as a Postdoctoral Fellow at Nanyang Technological University and Harvard University, respectively. From 2014 to 2019, he served as an Assistant Professor at the Department of Mechanical Engineering, The University of Hong Kong, before joining SUSTECH in 2019 as Professor and the Principal Investigator. His research interests include: robotic design and control, soft bionic robotics, underwater robotics, medical robotics, and power line and specialized robotics. He had successfully led and collaborated in over 20 research grants across EU, Singapore, USA, Hong Kong, and China, receiving sponsorship from HK RGC, ITC, Shenzhen Technological and Innovation Committee, and major leading companies from Microsoft, Delta, Lenovo, MBrain, Hytera, and Tencent. He is currently a Senior Member IEEE RAS and a member of ASME.

Ming Liu (Senior Member, IEEE) received the Ph.D. degree in robotics from ETH Zurich. He is currently the Director of the Intelligent Autonomous Driving Center of HKUST. He published over 300 articles. His research interests include dynamic environment modeling, 3-D mapping, machine learning, and visual control. He received various IEEE awards, including the IEEE IROS Young Professional Award in 2018 and 15 best paper awards or finalist awards, such as the Best Paper Award in Information for IEEE International Conference on Information and Automation 2013, the Best RoboCup Paper Award for IEEE International Conference on Intelligent Robots and System 2013, the Best Conference Paper Award for IEEE-CYBER 2015, the Best Paper in Automation Award for IEEE-ICIA 2017, and the Best Paper Awards for IEEE-ROBIO 2019 and IPIN2019. He was the Program Chair of IEEE-RCAR 2016 and the Conference General Chair of ICVS 2017 and ICVS2019. He is particularly interested in the investigation of novel, real-time online approaches in solving autonomous driving, mobile robot mapping, and navigation. His team developed the first autonomous vehicle in Hong Kong in 2017. During COVID-19, his autonomous vehicles served several cities and were reported by more than 100 media, including IEEE Spectrum and Xinhua.