Real-Time 3D Single Object Tracking With Transformer

Jiayao Shan, Sifan Zhou, Yubo Cui, and Zheng Fang, *Member, IEEE*

***Abstract*—LiDAR-based 3D single object tracking is a challenging issue in robotics and autonomous driving. Currently, existing approaches usually suffer from the problem that objects at long distance often have very sparse or partially-occluded point clouds, which makes the features extracted by the model ambiguous. Ambiguous features will make it hard to locate the target object and finally lead to bad tracking results. To solve this problem, we utilize the powerful Transformer architecture and propose a Point-Track-Transformer (PTT) module for point cloud-based 3D single object tracking task. Specifically, PTT module generates fine-tuned attention features by computing attention weights, which guides the tracker focusing on the important features of the target and improves the tracking ability in complex scenarios. To evaluate our PTT module, we embed PTT into the dominant method and construct a novel 3D SOT tracker named PTT-Net. In PTT-Net, we embed PTT into the voting stage and proposal generation stage, respectively. PTT module in the voting stage could model the interactions among point patches, which learns context-dependent features. Meanwhile, PTT module in the proposal generation stage could capture the contextual information between object and background. We evaluate our PTT-Net on KITTI and NuScenes datasets. Experimental results demonstrate the effectiveness of PTT module and the superiority of PTT-Net, which surpasses the baseline by a noticeable margin, ∼10% in the Car category. Meanwhile, our method also has a significant performance improvement in sparse scenarios. In general, the combination of transformer and tracking pipeline enables our PTT-Net to achieve state-of-the-art performance on both two datasets. Additionally, PTT-Net could run in real-time at 40FPS on NVIDIA 1080Ti GPU. Our code is open-sourced for the research community at https://github.com/shanjiayao/PTT.**

***Index Terms*—3D single object tracking, lidar point-cloud, siamese network, transformer, self attention.**

Manuscript received 19 August 2021; revised 29 November 2021; accepted 10 January 2022. Date of publication 27 January 2022; date of current version 7 June 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62073066 and U20A20197, in part by the Science and Technology on Near-Surface Detection Laboratory under Grant 6142414200208, in part by the Fundamental Research Funds for the Central Universities under Grant N182608003, in part by the Major Special Science and Technology Project of Liaoning Province under Grant 2019JH1/10100026, and in part by the Aeronautical Science Foundation of China under Grant 201941050001. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Vladan Velisavljevic. *(Jiayao Shan, Sifan Zhou, and Yubo Cui contributed equally to this work.) (Corresponding author: Zheng Fang.)*

Jiayao Shan is with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China, and also with the Science and Technology on Near-Surface Detection Laboratory, Wuxi 214000, China (e-mail: shanjiayao97@gmail.com).

Sifan Zhou, Yubo Cui, and Zheng Fang are with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: sifanjay@gmail.com; ybcui21@stumail.neu.edu.cn; fangzheng@mail.neu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMM.2022.3146714, provided by the authors.

Digital Object Identifier 10.1109/TMM.2022.3146714

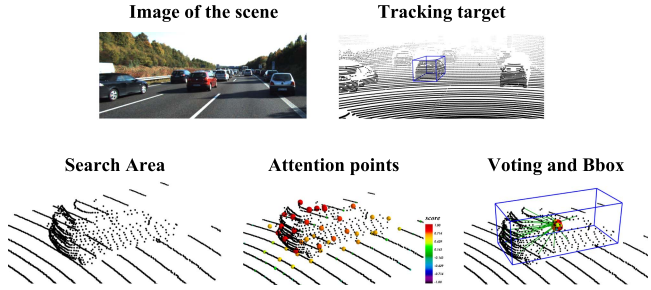

Fig. 1. Exemplified illustration to show the attention weights and voting result. Given raw point cloud, we specify search area and extract robust key-points first. Then, the transformer and attention mechanism focus on the key-points which carry rich and robust information. Finally, the voting module will generate votes according to the key-points and the predict bounding box.

I. Introduction

SINGLE object tracking (SOT) using LiDAR points has a wide range of applications in robotics and autonomous driving [1]–[3]. For example, the autonomous pedestrian following robot should robustly track its master and localize him/her accurately for efficient following control in the crowd. Another example is autonomous landing of unmanned aerial vehicles, where the drone needs to track the target and know the accurate distance and pose of the target for safe landing. However, most existing 3D SOT methods are usually using visual or RGB-D cameras [4]–[6], which may fail in visually degraded or illumination changing environments due to that they mainly depend on the dense images for target tracking.

In addition to visual or RGB-D sensors, 3D LiDAR sensors are also widely used in object tracking tasks [7]–[9] because they are less sensitive to illumination changes and could directly capture geometric and distance information more accurately. However, LiDAR-based 3D SOT has its own challenges. *First*, point data is sparse and disordered [10], which requires the network to be permutation-invariant to handle points well. *Second*, point cloud is spatially discrete, which is naturally different from dense image. *Third*, 3D object tracking needs to estimate higher space dimension information (e.g., x, y, z, w, h, l, ry) than 2D visual tracking, which brings more computational complexity. All these problems bring great challenges to realize a robust and real-time LiDAR-based tracking method.

Different from the existing LiDAR-based Multi-object Tracking (MOT) methods [9], [11]–[14], LiDAR-based 3D SOT methods need to model the similarity function between target template and search area to localize the target object. Although they

both need to compute the similarity, MOT methods compute the object-level similarity to association the detection results and tracklets, while SOT methods compute the intra-object-level similarity to localize the target object. Therefore, compared to 3D MOT, 3D SOT has its own challenges. SC3D [15] is the pioneer LiDAR-based 3D Siamese tracker which is based on the shape completion network. However, the method only uses an encoder consisting of 3 layers of 1D-convolutions to process the input point cloud, which makes it difficult to extract the robust point cloud feature representation. Besides, SC3D could not run in real-time and be trained end-to-end. Qi *et al.* [16] also proposed a point-to-box (P2B) network to estimate the target bounding box from the raw point cloud. However, their approach usually fails to track in sparse point cloud scenarios. Meanwhile, P2B gives no preference to non-accidental coincidences [17] which have more contribution to locating the target center. Recently, Fang *et al.* [18] combined Siamese network and LiDAR-based RPN network [19] to tackle 3D object tracking. Nevertheless, they directly use the classification scores to sort regression results, ignoring the inconsistency between the localization and classification. It is worth noting that points located in different geometric positions often have different importance in representing targets. However, these aforementioned methods do not weigh point cloud features based on this characteristic. Besides, the point cloud features extracted from the template and the search area contain less potential object information and more background noise due to the sparsity and occlusion of point clouds. Therefore, how to pay attention to the spatial clues is the key to improve the performance of the 3D object tracker.

Recently, Transformer has shown amazing performance in feature encoding due to its powerful self-attention module [20]–[22]. Transformer usually consists of three main modules, including input (word) embedding, position encoding and attention module. Compared with the convolution network, content-adaptive property and unique position module of transformer make it more suitable for processing 3D point clouds. In addition, the 3D SOT task only focuses on local areas, which makes the transformer suitable for this task, although the transformer is sensitive to time and space cost.

In this paper, we propose a Point-Track-Transformer (PTT) module for 3D single object tracking to learn features more effectively by leveraging the superiority of the transformer models on set-structured point clouds. The core idea is to focus on the important features of the target object by utilizing the self-attention and position encoding mechanism to weigh the point cloud features. PTT module contains three blocks for feature embedding, position encoding, and self-attention feature computation, respectively. Feature embedding aims to place features closer in the embedding space if they have similar semantic information. Position encoding is used to encode the coordinates of point cloud into high dimension distinguishable features. Self-attention generates refined attention features by computing attention weights. Furthermore, to evaluate the effectiveness of our PTT module, we embed the PTT module into the dominant P2B [16] to construct a novel 3D SOT tracker termed PTT-Net. In PTT-Net, we add PTT into the voting stage and proposal generation stage, respectively. PTT embedded in the voting stage could model interactions among point patches located in different geometric positions, which learns context-dependent features and helps the network focus on more representative features of objects. Meanwhile, PTT embedded in the proposal generation stage could capture the contextual information between object and background, and help the network to effectively suppress background noise. These modifications can efficiently improve the performance of 3D object tracker. The experimental results of our PTT-Net on KITTI tracking dataset [23] demonstrate the superiority of our method ($\sim$10%'s improvement compared to the baseline). We further evaluate our PTT-Net on NuScenes dataset [24], the results show that our method could achieve new state-of-the-art performance. Additionally, PTT-Net could run in real-time at 40FPS on a single NVIDIA 1080Ti GPU.

Overall, our main contributions are as follows:

- *PTT module:* we propose a Point-Track-Transformer (PTT) module for 3D single object tracking using only raw point clouds, which could weigh point cloud features to focus on deeper-level object clues during tracking.
- *PTT-Net:* we construct a 3D single object tracking network embedded with PTT modules which can be trained end-to-end. To the best of our knowledge, this is the first work to apply transformer to 3D object tracking task using point clouds.
- *Open-source:* extensive experiments on KITTI and NuScenes datasets show that our method outperforms the state-of-the-art methods with remarkable margins at the speed of 40 FPS. Besides, we open source of our method to the research community.

Apart from that, as an extended work of our conference paper [25], we add more detailed descriptions on the network architecture and dataset. Besides, we also carry out more qualitative experiments and visualizations of our PTT-Net to analyze the effectiveness of transformer in 3D single object tracking task. For the extended experiments on the more challenging NuScenes dataset, our method still achieves state-of-the-art performance. The results indicate that our method could be adapted to more complex scenes, further confirming the effectiveness of our method.

The rest of this paper is organized as follows. In Section II, we discuss the related work. Section III describes the proposed PTT module and PTT-Net. We validate the performance of our method on KITTI and NuScenes datasets in Section IV and we conclude in Section V.

II. Related Work

This section will briefly discuss the related works in 2D siamese trackers, 3D single object tracking, transformer, and self-attention mechanism.

A. 2D Siamese Tracking

Early 2D visual trackers mainly focused on correlation filtering [26]–[29]. However, these methods are based on the tracking template matching mechanism, so they cannot cope with the rapid deformation of the tracking target. Recently, the realization of 2D object tracking tasks based on Siamese networks has become the mainstream with the rapid development

of deep learning [30]–[39]. Luca *et al.* [30] proposed SiamFC which was the first pioneer work of Siamese trackers. The visual tracking task was handled as a similarity problem, and the cross-correlation module was introduced into the network structure. Subsequently, a large number variants of SiamFC [30] were proposed. Li *et al.* [31] introduced the Region Proposal Network (RPN) into the Siamese network. SiamRPN could regress more accurate 2D bounding box than SiamFC. Besides, Li *et al.* [32] explored the relationship between the number of network layers and tracker performance, optimized the network depth of the tracker to improve the tracking accuracy. Furthermore, Wang *et al.* [34] integrated the task framework of image segmentation and image tracking, and used the mask to improve the accuracy of the tracker. Paul *et al.* [35] proposed a two-stage network for visual tracking, which used the re-detection of the first frame template and the previous frame template to modify the tracking target. Their method surpassed all previous methods on six short-term tracking benchmarks and four long-term tracking benchmarks, and achieved amazing results. [36] introduced an informative enhanced loss, which can enable the network to capture information from an overall perspective. Han *et al.* [37] proposed an asymmetric convolution module, which could capture the semantic correlation information well. To address the problem of decisive samples missing during offline training, Dong *et al.* [38] proposed a compact latent network to make the model could quickly adapt to new scenes. And Dong *et al.* [39] introduced a novel hyper-parameter optimization method by using deep reinforcement learning. Chen *et al.* [40] proposed a tracking framework by fusing the template and search features with transformer. In summary, the 2D visual tracking method has made great progress in the past decade and has been applied to many practical scenarios. However, limited by the sensor, the 2D visual trackers are still very sensitive to illumination changes. In addition, most of the 2D visual tracking methods only obtain the pixel coordinates of the tracking target, but sometimes it is necessary to know the accurate three-dimensional pose of the tracking target.

B. 3D SOT Using Point Cloud

Giancola *et al.* [15] proposed the first pioneer LiDAR-based 3D single object tracker which utilized the Kalman Filter to generate massive target proposals. They exploited shape completion to learn the shape information of target, but their method has a poor generalization ability and could not run in real-time. Zarzar *et al.* [41] leveraged 2D Siamese network which works on Bird-Eye-View (BEV) representation to generate 3D proposals. However, this method may lose fine-grained geometry details which are important for tracking tiny objects. Cui *et al.* [42] also adopted a 3D Siamese tracker only using point cloud, but they could not estimate the orientation and size information of the target. Fang *et al.* [18] combined 3D Siamese network and 3D RPN network to track targets, while the performance is limited by the RPN network. Besides, Zou *et al.* [43] integrated 2D image and 3D point cloud information for 3D object tracking. However, this method relies more on 2D tracker and uses the ground truth to track objects, which is unreasonable for realistic application. Qi *et al.* [16] proposed P2B which used deep hough voting to obtain the potential centers (votes) and estimated the target center based on those votes. However, they ignore the fact of points in different positions have different contributions in tracking. Furthermore, its random sampling mechanism loses the location distribution information of the raw point cloud. To deal with these shortcomings, we propose a PTT module to capture the feature correlations among the neighbor point around the target object by weighing different point features. Moreover, we use farthest point sampling instead of random sampling to obtain more raw point cloud information.

C. Transformer and Self-Attention

Recently, transformer has revolutionized natural language processing and image analysis [20], [44]–[47]. Hu *et al.* [44] and Ramachandran *et al.* [45] applied scalar dot product self-attention to local pixel neighbors. Zhao *et al.* [48] applied vector self-attention operations to image tasks. These works combined or replaced CNNs with self-attention layers and confirmed the transformer's great potential in visual tasks.

Inspired by those works, Zhao *et al.* [49] used a Point Transformer layer by applying vector self-attention operations, which had a great performance improvement in point cloud classification and segmentation tasks. Because self-attention operator, which is the core of transformer networks, is intrinsically a set operator: positional information is provided as attributes of elements that are processed as a set [20], [48]. Therefore, transformer is suitable for point cloud processing due to its positional attributes. Besides, Nico *et al.* [50] proposed SortNet as a part of Point Transformer and achieved competitive performance on point cloud classification and part segmentation tasks. Meanwhile, Guo *et al.* [51] also introduced Point Cloud Transformer (PCT), which performed well on shape classification, part segmentation, and normal estimation tasks. Recently, Pan *et al.* [52] proposed a PointFormer as the drop-in replacement backbone for 3D object detection and gained state-of-the-art performance. Obviously, transformer has unique advantages for point cloud feature learning.

In addition, transformer and attention mechanism have recently been widely used in 2D tracking tasks [40], [53]–[57]. The tracker using the transformer or attention also shows superior performance with the help of the transformer's powerful attention mechanism for features. Therefore, we apply the transformer to the 3D point cloud tracking task to improve the performance of the tracker.

III. Methodology

In this section, we first analyze the challenges and bottlenecks of current LiDAR-based single object trackers and discuss feasible solutions. Then, we revisit the transformer and present our PTT module for LiDAR-based object tracking. Finally, we introduce our PTT-Net in detail.

A. Baseline

P2B [16] is the dominant 3D SOT method using point clouds. In this work, we use P2B as the baseline. The main idea of P2B is to localize the target center in 3D search area and execute the proposal generation and verification jointly. Hence we can divide P2B into two parts. The first part is feature enhancement. The input template and search point clouds are extracted through a shared weight backbone network [58], then the corresponding similarity could be obtained by calculating the point-wise cosine similarity in an implicitly embedded space. The second part is the region proposal network which generates the proposals by deep hough voting mechanism [59] from the semantic features. Besides, P2B also utilizes the proposal clustering network to leverage the ensemble power and obtain accurate target proposals.

However, P2B tends to suffer from the defects that it gives no preference to non-accidental coincidences [17] which have more contribution to locate the target center. Therefore, we would like to explore the differences among the augmented features by using the transformer architecture.

B. Challenges

In Section I, we had pointed out several challenges for processing point cloud data. For 3D single object tracking task, there are also several challenges as follows.

1) Error Accumulation and Propagation: In tasks involving point clouds and deep learning, tracking is naturally different from detection and segmentation due to its spatial-temporal continuity. Since the tracking target is a sequence, which makes two frames similar in spatial and temporal terms. Thus, the dominant tracking algorithms utilize the spatial and temporal prior information to initialize the search area. In spite of its effectiveness in reducing computational complexity, bad tracking results will lead to large tracking error in challenging scenarios. This is because the consecutive tracking predictions will accumulate over the historical error and propagate it to the next frame. And when error accumulates enough, the tracker will fails.

2) Sparsity Sensitive: The Siamese network from VOT is generally adopted in existing algorithms for point cloud object tracking. However, the image data is usually dense, while point cloud is naturally sparse. This causes gaps in the effect of applying Siamese network in image and point cloud. Sparse point cloud data makes it difficult for the backbone network to extract robust point cloud features. Hence, the existing LiDAR-based tracking methods are sensitive to the sparsity of point cloud data.

3) Feature Ambiguity: The sparsity of point clouds limits networks on modeling interactions among point patches located in different geometric positions and capturing contextual information, which makes the features extracted by the models ambiguous. The points at locations where the object surface have a low-dimensional structure, such as a plane, contribute ambiguous features [17]. The feature ambiguity makes the tracker hard to classify the fore-background points and regress the box center, and finally leads to bad tracking results.

These challenges motivate us to propose new algorithm to deal with those problems. To this end, we propose our PTT (Point-Track-Transformer) module and PTT-Net. PTT module can handle the sparsity of points with the help of the attention mechanism in Transformer, where the contribution of each point is automatically learned in the network training. Besides, the ambiguity of features will be suppressed through self-attention mechanism because of the role of attention is to refine features and guide the network to focus on more representative features of the tracking target. Finally, our PTT-Net embedded with PTT module could capture sparse dependencies even from a few points, thus reducing the accumulation of errors.

C. Revisiting Transformer

Transformer [20] is firstly introduced to aggregate information from the entire input sequence for machine translation. It can handle sequential tasks well due to its attention mechanism. The core architecture of transformer can be divided into three parts: input feature embedding, position encoding, and self-attention. Self-attention is the core module, which mainly focuses on the differences of input features and generates refined attention features based on global or local context. Given the input feature $G = \{g_i\}_{i=1}^{N}$ after feature embedding, the general formula of self-attention is:

$$\begin{aligned} Q, K, V &= \alpha(G), \beta(G), \gamma(G) \\ \mathcal{A} &= \rho(\sigma(Q, K) + P) \odot (V) \end{aligned} \tag{1}$$

where α, β and γ are point-wise feature transformations (e.g. linear layers or MLPs). Q, K, and V are the $query$, key and $value$ matrices, respectively. σ is the relation function between Q and K. P is the position encoding feature. ρ is a normalization function (e.g. *softmax*). $\odot$ means Hadamard product, which is used to obtain the output features from the attention weights and V. $\mathcal{A}$ is the attention feature produced by the self-attention module. For the relation function σ, the regular form in machine translation [20] is:

$$\sigma(Q, K) = QK^T \tag{2}$$

D. PTT Module

To further integrate the self-attention mechanism into the point cloud tracking task, we modify the transformer module proposed in [49] to capture point cloud features better. The point transformer layer in [49] is proposed to process the raw point cloud for classification and segmentation tasks. However, we utilize the point transformer to benefit the tracking task and enable the tracker to capture spatial relations and object geometry shape information. All these modifications construct the PTT module, which is used to refine the features from raw sparse point clouds and eliminate the ambiguity among features. The architecture of PTT module is shown in Fig. 2.

For 3D SOT task, given an input of M points with XYZ coordinates, a backbone network is used to extract the point cloud and learn deep features. It outputs a subset of the input containing N interest points (seeds) $S = \{s_i\}_{i=1}^{N}$. Here, $s_i = (c_i, f_i)$ is composed of a vector c_i of 3D coordinate and a D-dimensional descriptor f_i of the local object geometry. Our goal of using transformer is to perform an attention weighting operation on

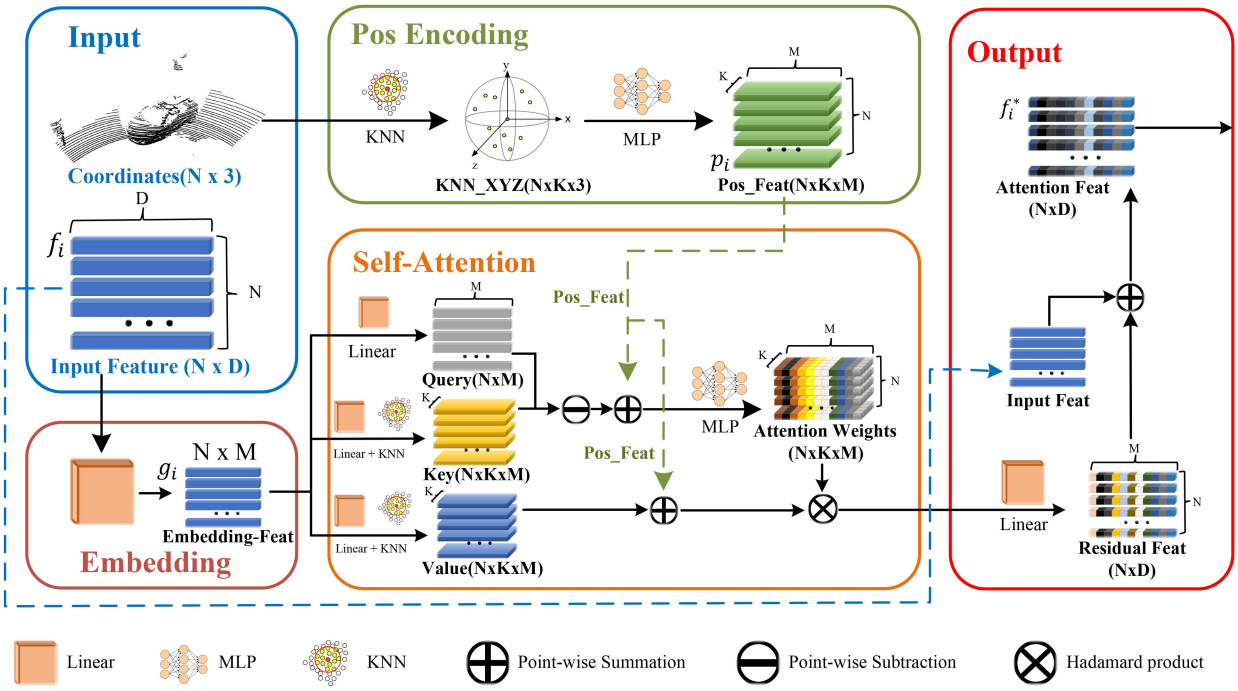

Fig. 2. PTT module architecture. It consists of three blocks: feature embedding, position encoding, and self-attention. The whole input are the coordinates and their corresponding features. Feature embedding module maps input features into embedding space. In position encoding module, the k-nearest neighbor algorithm is used to obtain local position information, then the encoded position features will be learned by an MLP layer. The self-attention module learns refined attention features for input features based on local context. The output features of PTT module are the sum of input and residual features.

the feature space of f_i, and output refined features $f_i{}^*$ with the same dimension.

PTT module processes features by utilizing shape and geometry information. Given a point set $S = \{s_i\}_{i=1}^{N}$, $s_i = (c_i, f_i)$, $c_i \in \mathbb{R}^3$ and $f_i \in \mathbb{R}^D$. c_i and f_i represent 3D coordinates and descriptor of point s_i. Feature embedding module maps input features into embedding space $\mathbb{R}^M$: $f_i \to g_i, g_i \in \mathbb{R}^M$. Position encoding module extracts higher-level M-dimensional features p_i from input coordinates c_i: $c_i \to p_i, p_i \in \mathbb{R}^{K \times M}$. Finally, the self-attention module calculates attention weights and attention features $f_i^*, f_i^* \in \mathbb{R}^D$ by taking embedding features and position features as inputs. To avoid the vanishing gradient problem in the training stage, we also adopt the residual architecture proposed in [60], and take the sum of the attention features and input features as output features.

1) Feature Embedding: The original feature embedding module in natural language processing is to map each word in the input sequence to a high-dimensional vector. In this work, we use the linear layer to complete the feature embedding operation, and map the input point cloud feature dimension from D To M: $\mathbb{R}^D \to \mathbb{R}^M$, which can place the feature closer in the embedding space when the semantics are more similar and make the network have a stronger fitting ability.

2) Position Encoding: Position encoding module plays a crucial role in transformer, which allows operators to adapt to the local structure of the input data [20]. 3D point coordinates are valuable features indicating the local structures. Therefore, we utilize the coordinates directly as the input of the position encoding module. Compared to the techniques used in natural language processing, we use a simple and yet efficient approach by mapping the coordinates of each point to the feature dimension and the resulting position encoding is added to the attention matrix. We adopt the relative coordinates to make the network better capture the spatial correlation between points and local geometric shape information. Since the feature f_i extracted by [58] can provide the local context information, we obtain the position encoding features $P = \{p_i\}_{i=1}^{N}$ with function η. For input point set S including N points, the position encoding feature for each point is:

$$p_i = \eta(c_i - c_j) \tag{3}$$

where c_i is the coordinate of the i-th point in S. c_j is the j-th coordinate in local neighborhood region of c_i. η is an MLP with two linear layers and one ReLU non-linearity. Here, we use KNN to capture the local context and set $k = 16$ by inheriting the experimental results in [49].

3) Self-Attention: As Fig. 2 shows, self-attention module computes three vectors for each point: Q, K, V through α, β, γ, where α, β, γ are all shared linear layers. It is worth noting that K and V are aggregated from the features of the k neighborhood points, which aims to encode more local context information. Here, $Q \in \mathbb{R}^M, K \in \mathbb{R}^{k \times M}$, and $V \in \mathbb{R}^{k \times M}$. For relation function σ, we use $\sigma(Q, K)$ to obtain point-wise attention weights, the detail implementation of $\sigma(Q, K)$ will be introduced in next

part. And an MLP layer γ is used to introduce additional trainable transformations and match the output dimension. Then, we add the position encoding features P to both the attention vector σ and the transformed features K. Finally, the residual features recorded as $\mathcal{A}$ are defined as the weighted sum of the attention weights with all V vectors. The formula is as follows:

$$\mathcal{A} = \rho\left(\gamma(\sigma(Q, K) + P)\right) \odot (V + P) \tag{4}$$

where ρ is a normalization function (softmax) and γ is a non-linear mapping function (MLP) that includes two linear layers and one ReLU non-linearity. σ is the relation function between Q and K. $\mathcal{A}$ is output attention features.

4) Relations Functions: The relation function $\sigma(Q, K)$ is the core of self-attention. Different ways of obtaining the relation between Query vector and Key vector could construct different types of attention modules. As mentioned in [49], self-attention operators can be classified into two types: scalar attention [20] and vector attention [48]. In scalar attention, the relation function $\sigma(Q, K)$ can be expressed in the form of (2), which computes the scalar product between Query vector and Key vector. The vector attention obtains vector attention weights by using channel-wise subtraction operation. Besides, it has been confirmed in [49] that vector attention is a natural fit for point cloud than scalar attention since it supports adaptive modulation of individual feature channels, not just whole feature vectors. Therefore, we use the vector attention structure. The relation function $\sigma(Q, K)$ is a subtraction operation. The formula is as follows:

$$\sigma(Q, K) = Q - K \tag{5}$$

E. PTT-Net

This section details our PTT-Net which is a more accurate and robust target tracking based on the existing open-source method P2B [16]. In the following, we first explore the effect of different sampling methods on point cloud-based tracking tasks. Then, the position where the PTT module is embedded and the loss function of the network training are described.

1) Sampling Strategies: Here, we first discuss the impact of different sampling strategies on tracking task. The purpose of sampling is to extract key points and ensure that the number of points in the template or search point cloud is aligned with the input dimension of network. However, different sampling methods will lead to different degrees of target information loss. At the same time, we find that in the existing tracking pipeline, the more foreground points left in the search area after sampling, the more accurate the regression results of the network. In contrast, due to the unbalanced distribution of foreground and background points, the classification accuracy of the network will decrease. Therefore, a suitable sampling strategy can not only achieve input alignment, but also improve the tracking performance of the tracking network.

Common 3D point cloud sampling methods include *Random Sampling (RS)*, *Farthest Point Sampling (FPS)*, and *Farthest Point Sampling in Feature dimension (Feat-FPS)*. RS can achieve high sampling efficiency, but the sampling result depends on the distribution density of point cloud. FPS can better retain the geometric information of the original point cloud, and is a more balanced sampling method. In addition, Feat-FPS proposed in 3D-SSD [61], which can sample in feature space, has better sampling results for targets of different semantic categories. The visualization of the three different sampling methods is shown in Fig. 4. We find that FPS has more uniform sampling results, while Feat-FPS pays more attention to points on the target. P2B uses RS, although it has a higher computational efficiency, it leads to the loss of some key information, which limits their tracking accuracy.

In summary, for the sampling strategy of tracking task, we adopted FPS which can retain the geometric information of the original point cloud and make the foreground and background points balanced. A demonstration of its good tracking performance will be shown in Section IV-D.

2) Embedding Position of PTT: The ability of transformer to learn self-attention weights inspires us to try it on 3D SOT task. We formulate the problem of focusing on the differences of features as self-attention weighting. In order to verify the effect of our method, we embed our PTT module into the previous open-source *state-of-the-art* (SOTA) LiDAR-based 3D SOT work P2B [16]. More specifically, the PTT modules are inserted in seeds voting stage and proposals generation stage of P2B respectively. In the seeds voting stage, P2B generates votes by utilizing the augmented features, which are obtained from the backbone network (in Fig. 3). We notice that [16] ignores the differences among different point cloud features in the search area, and gives no preference to the points in different locations when generating votes. However, it is important to focus on the points which contain more geometric information. Therefore, we apply PTT module to weigh the augmented features and model interactions among point patches to learn the context-dependent feature (in Fig. 5(a), (b)).

In proposals generation stage, P2B generates proposals based on local context features. However, their method ignores the global semantic features of the targets, so that they could not distinguish similar objects (e.g., two pedestrians, in Fig. 5(c), (d)). Therefore, we use the PTT module to further weigh the target-wise context features obtained by the aggregation network in P2B for tracking deeper-level target clues to capture the contextual information between object and background.

As shown in Fig. 3, we embed our PTT module in the open-source SOTA method P2B [16] to build PTT-Net. We add PTT module to the seeds voting and proposal generation stages, and weigh the augmented features and cluster features respectively. Experiments show that our PTT-Net outperforms the SOTA method with remarkable margins.

3) Loss Function: The PTT module is trained with the other sub-networks in [16]. Therefore, we follow [16] to design our loss function. The overall loss consists of two parts as follows:

$$L_{all} = L_{cv} + \lambda_1 L_{cb} + \lambda_2 L_{rv} + \lambda_3 L_{rb} \tag{6}$$

where $\lambda_1, \lambda_2, \lambda_3$ represent the weighting coefficient of each loss. Classification loss includes voting classification loss L_{cv} and

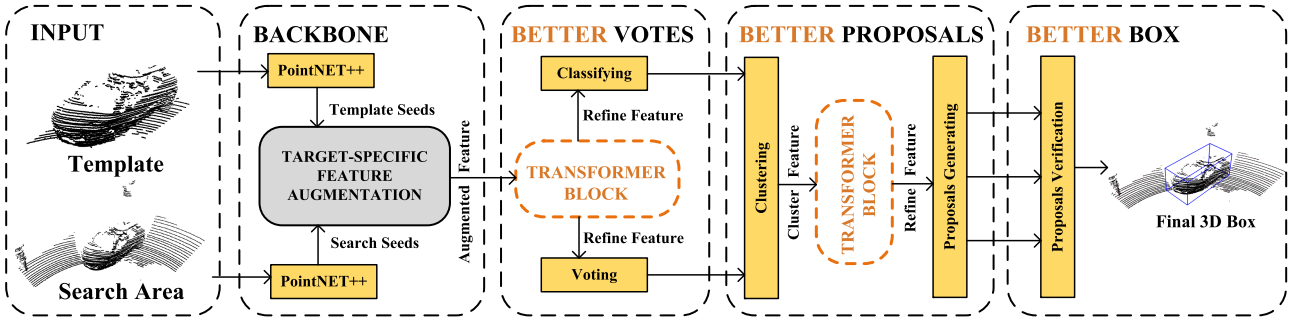

Fig. 3. The pipeline of PTT-Net. In order to verify the effect of our PTT module, we embed two PTT modules into seeds voting and proposals generation stage of the deep hough voting framework.

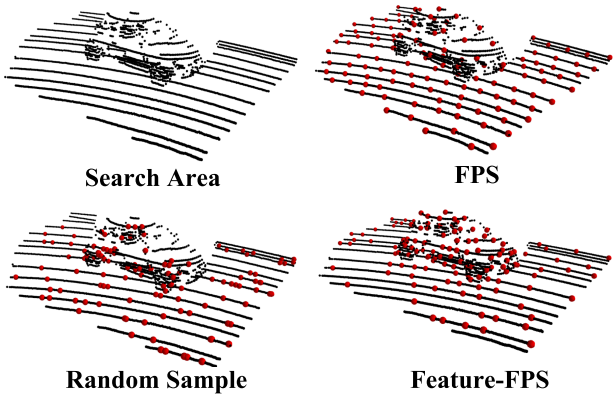

Fig. 4. Visualization of different sampling methods. The input points and sampled points are labeled as black and red respectively. As shown in the figure, the result of RS depends on the distribution density. And FPS can better retain the geometric information. However, the Feat-FPS mainly focuses on the foreground points, which will cause the unbalanced distribution between foreground and background points.

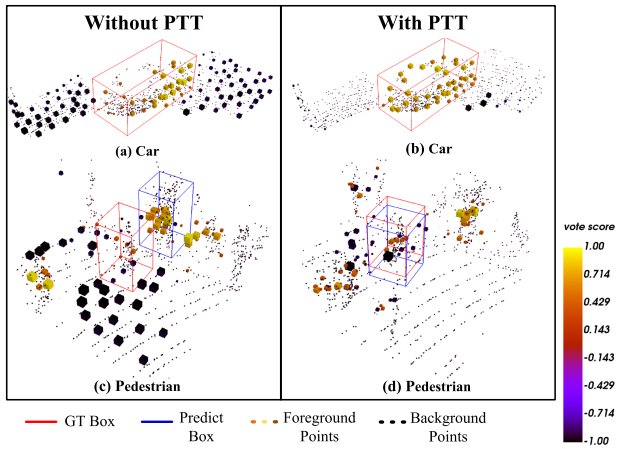

Fig. 5. Visualization of classification (a)–(b) and tracking (c)–(d) results with or without PTT module. The point will be paid more attention if it has a higher score. Compared (a) with (b), PTT module pays more attention to the foreground points. Compared (c) with (d), PTT module could still track targets robustly in crowded scenes(with multiple pedestrians).

proposal box classification loss L_{cb}. The regression loss includes the voting loss L_{rv} and the proposal box regression loss L_{rb}.

IV. Experiments

We used KITTI tracking dataset [23] and NuScenes [24] dataset as the benchmark. Similar to [15], [16], [18], [43], we mainly focused on rigid car tracking and performed ablation studies on KITTI. We also conducted extended experiments with other three target categories (Pedestrian, Van, Cyclist) to comprehensively evaluate the performance of our method for non-rigid objects (Pedestrian) tracking on KITTI. Besides, we also follow BAT [63] to evaluate our PTT-Net on NuScenes. The experiments show that PTT-Net outperforms previous SOTA methods with remarkable margins and can run at 40 fps.[1]

A. Experimental Protocols

1) Datasets: We used the training set of KITTI and NuScenes dataset to train and evaluate our method. For KITTI dataset, there are more than 20,000 manually labeled 3D objects using Velodyne HDL-64E 3D lidar (10HZ). Following [15], [16], [18], [43], we split the dataset as follows: 0-16 for training, 17-18 for validation and 19-20 for testing. Specially, we first extract each frame label from every scene label of KITTI tracking benchmark. Then, we further extract each ID label from every frame label, and finally concatenate the labels of the same ID to obtain each tracklet label from its first frame to its final frame. By this way, we convert the KITTI MOT label to SOT label. Furthermore, for each tracklet, only the first frame includes the 3D ground truth bounding box (bbox) in the testing phase. Therefore, we initialize our tracker with the first frame 3D ground truth bbox during tracking, and then track the object in the sequence frames by our tracker. In order to better illustrate that the point cloud in KITTI is challenging for tracking task, we choose the Car category and count the number of foreground points in each frame, then calculate the percentage referred to all frames from 00-20 sequence in Fig. 6. The total number of frames is 27292. The number of frames that are less than 20 points is 7123, accounting for 26.10% of the total. The number of frames between 20-100 is 8650, accounting for 31.69% of the total. In addition, the number of 100-500 and more than 500 frames are 6810 and 4709, respectively, accounting for 24.95% and 17.25% of the total. This fully shows that about half of the frames in KITTI

[1]Our experiment video is available at https://youtu.be/z5Vkm8r9Wus

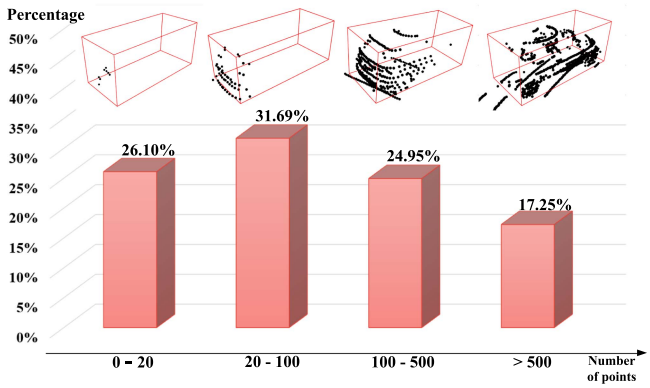

Fig. 6. The percentages and visualizations of the different number intervals of point cloud in the car category. The number of frames containing less than 20 points accounts for 26.10% of the total. The number of frames containing 20–100 points accounts for 31.69% of the total. In addition, the number of 100–500 and more than 500 frames are accounting for 24.95 and 17.25% of the total, respectively. For four different intervals, we visualize four point clouds with 11, 52, 293, and 883 points, respectively.

are sparse scenes. Hence the KITTI dataset is challenging for tracking task. This will be the bottleneck of limiting the tracking accuracy as mentioned in Section III-B. Furthermore, we exemplify the visualization results of point clouds at various number intervals in Fig. 6.

For NuScenes dataset, there are 1000 driving scenes and 23 object categories, which make it is more challenging because of more complex scenes. We follow the same settings with the BAT [63] to obtain a fair comparison, and directly refer to the results reported in BAT [63] for comparison.

2) Evaluation Metric: Following previous work [15], [16], [18], [43], we report Success and Precision metrics defined by One Pass Evaluation (OPE) [64], which represent the overlap and error Area Under the Curve (AUC) respectively.

3) Implementation Details: We use FPS in our implementation instead of RS used in original P2B [16] for point cloud sampling. In the training stage, we use the Adam optimizer and set the initial learning rate to 0.001 and decrease by 5 times after 12 epochs. The batch size is 48 and the training epoch is 60. Besides, we extend the offset from (x, y, θ) to (x, y, z, θ) when generating more template samples during data augmentation in [16]. In the testing stage, we also add Z axis offset to generate the predicted box. Other parameters are consistent with settings of [16]. Meanwhile, we also follow the tracking setting of P2B [16]. Specially, we initialize the template point cloud with the point cloud of the first frame ground truth and update the template point cloud by fusing the point cloud of first frame ground truth with previous result. The search point cloud is updated based on the point cloud of the previous result, which can better meet the requirement of real scenes.

B. Quantitative Experiment

To better evaluate our method, we designed two quantitative experiments. In the first experiment, we quantitatively evaluated our method for 3D car tracking. In the second experiment, we further compared PTT-Net with the previous methods among different categories on both KITTI and NuScenes datasets.

1) Comparisons on Car Category: We compared the performance of our PTT-Net with the existing methods on the KITTI dataset and reported the results for 3D car tracking in Table I. To meet the requirement of real scenarios, we generate the search area centered on the previous tracking result. The results show that our PTT-Net has achieved SOTA performance in all evaluation metrics with remarkable margins. Compared with the baseline algorithm P2B [16], our performance has been greatly improved by ∼11% in 3D Success. Besides, compared with the previous SOTA method 3D-SiamRPN [18], our method performs better by a margin of ∼9% and ∼5% in 3D Success and 3D Precision respectively. It verifies the superiority of our method, and shows that PTT-Net could work better in challenging scenarios like sparse or occlusion scenarios, while other trackers often fail to track in these scenes. Additionally, compared with [62] and [43] which both use RGB+LIDAR fusion information, the Success/Precision results of PTT-Net outperform them 4.7%/12.1% and 30.7%/31.2% respectively. We think that this is because our method can capture important feature representations on tracking targets even if it is only based on raw point cloud data. More importantly, compared with other methods, our proposed method not only has a good performance, but also could run in real time with 40 FPS.

2) Comparisons on Other Categories: For four categories in KITTI tracking dataset, the Car category is the rigid object. To evaluate the tracking performance in more scenes and especially containing non-rigid objects, we further compared our method with previous methods on Pedestrian, Van, and Cyclist (Table II). Except for F-Siamese [43], which fuses image and point cloud information, all other methods adopt the same experimental settings.

As shown in Table II, the average performance of PTT-Net outperforms SC3D [15], P2B [16] and 3D-SiamRPN [18] by ∼24%, ∼13% and ∼9% respectively. It is worth noting that the Success/Precision results of PTT-Net show an improvement (9.7%/15.8%) on non-rigid object (Pedestrian) tracking. The result also verifies that our PTT module can help the network understand and learn the important features of the target better. Additionally, we notice that there are performance gaps between our method and the best method F-Siamese [43] in Van and Cyclist categories. We believe there are two reasons. First, the cyclist has the least training samples (only 1529 samples for training), which may limit the performance of the transformer. And we did not do any extra data augmentation for cyclist because we would like to use a fair setting among all categories. Second, F-Siamese firstly utilizes a 2D siamese tracker in the front end only using dense image data, and generates a search area based on the results of the 2D siamese tracker, which provides prior information for subsequent 3D SOT. We believe that the data fusion in F-Siamese may result in the better performance. Additionally, we also notice that the Success results of P2B achieves 28.7%/32.1% in pedestrian and cyclist respectively, and our method brings +16.2%/+5.2% gains in the two categories. Because our method is based on P2B, we believe the

TABLE I
PERFORMANCE COMPARISON ON **KITTI** FOR THE **CAR** CATEGORY

Module	Modality	3D Success	3D Precision	FPS
AVOD-Tracking [63]	RGB+LiDAR	63.1	69.7	-
F-Siamese [44]	RGB+LIDAR	37.1	50.6	-
SC3D [16]	LiDAR only	41.3	57.9	1.8
ETP2D-3D [42]	LiDAR only	36.3	51.0	-
P2B [17]	LiDAR only	56.2	72.8	45.5
3D-SiamRPN [19]	LiDAR only	58.2	76.2	20.8
PTT-Net(Ours)	LiDAR only	67.8	81.8	40.0

Red and blue mean the performance score is ranked first and second respectively.

TABLE II
EXTENSIVE COMPARISONS WITH DIFFERENT CATEGORIES ON **KITTI**(LEFT) AND **NUSCENES**(RIGHT) DATASET

	Dataset	KITTI					NuScenes				
	Category	Car	Pedestrian	Van	Cyclist	Mean	Car	Truck	Trailer	Bus	Mean
	Frame Number	6424	6088	1248	308	14068	64159	13587	3352	2953	84051
3D Success	SC3D [16]	41.3	18.2	40.4	41.5	31.2	22.31	30.67	35.28	29.35	24.43
	P2B [17]	56.2	28.7	40.8	32.1	42.4	38.81	42.95	48.96	32.95	39.68
	F-Siamese [44]	37.1	16.2	-	47.0	-	-	-	-	-	-
	3D-SiamRPN [19]	58.2	35.2	45.6	36.1	46.6	-	-	-	-	-
	BAT [64]	65.4	45.7	52.4	33.7	55.0	40.73	45.34	52.59	35.44	41.76
	PTT-Net(Ours)	67.8	44.9	43.6	37.2	55.1	41.22	50.23	61.66	43.86	43.58
3D Precision	SC3D [16]	57.9	37.8	47.0	70.4	48.5	21.93	27.73	28.12	24.08	23.19
	P2B [17]	72.8	49.6	48.4	44.7	60.0	43.18	41.59	40.05	27.41	42.24
	F-Siamese [44]	50.6	32.2	-	77.2	-	-	-	-	-	-
	3D-SiamRPN [19]	76.2	56.2	52.8	49.0	64.9	-	-	-	-	-
	BAT [64]	78.9	74.5	67.0	45.4	75.2	43.29	42.58	44.89	28.01	42.70
	PTT-Net(Ours)	81.8	72.0	52.5	47.3	74.2	45.26	48.56	56.05	39.96	46.04

Red and blue mean the performance score is ranked first and second respectively. And frame number indicates the instance number of each category.

results also verify the effectiveness of our method. Besides, our performance is much higher than that of F-Siamese method in the categories with more abundant data, such as vehicles and pedestrians. The Success/Precision results of PTT-Net outperform F-siamese by 30.7%/31.2% and 28.7%/39.8% in vehicles and pedestrian category respectively.

Besides, we also evaluated our PTT-Net on NuScenes dataset to further confirm the effectiveness of our method. Although KITTI dataset is commonly used by previous 3D SOT methods, its scale is too small and this may limit the performance of proposed network. Recently, BAT [63] reports the results of the dominant methods on NuScenes dataset. Hence we follow the settings of BAT and evaluate our PTT-Net on NuScenes, and report the results in Table II. As shown in Table II, our PTT-Net outperforms BAT in all categories. This indicates that our PTT-Net could address the challenging scenes in NuScenes more effectively.

3) Comparison of Speed and Performance: We further show the comparisons with previous trackers in terms of speed, 3D Success and 3D Precision on Car category. As shown in Fig. 7, our PTT-Net achieves SOTA performance on both 3D Success and 3D Precision. Meanwhile, our method has less computational burden. In other words, our method has both high accuracy and fast running speed.

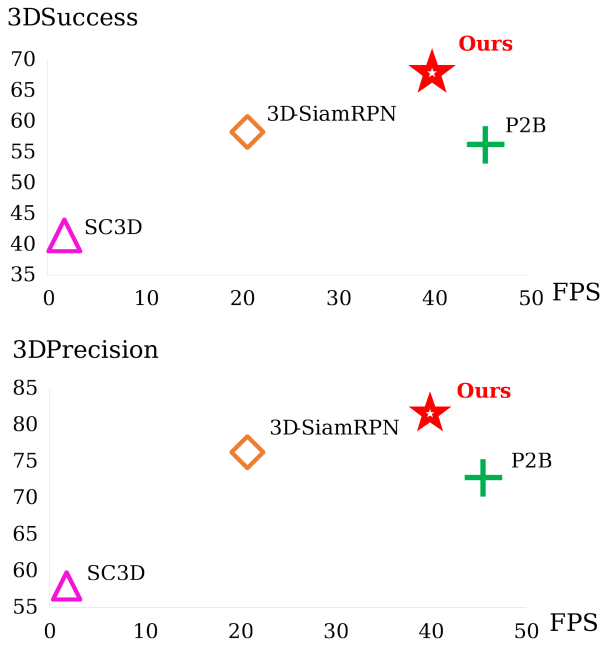

Fig. 7. Tracking Success and Precision vs. speed of the dominant trackers on Car category. The proposed PTT-Net is superior than SC3D [15], P2B [16], 3D SiamRPN [18] at 3D Success and 3D Precision, and could maintain a good inference speed.

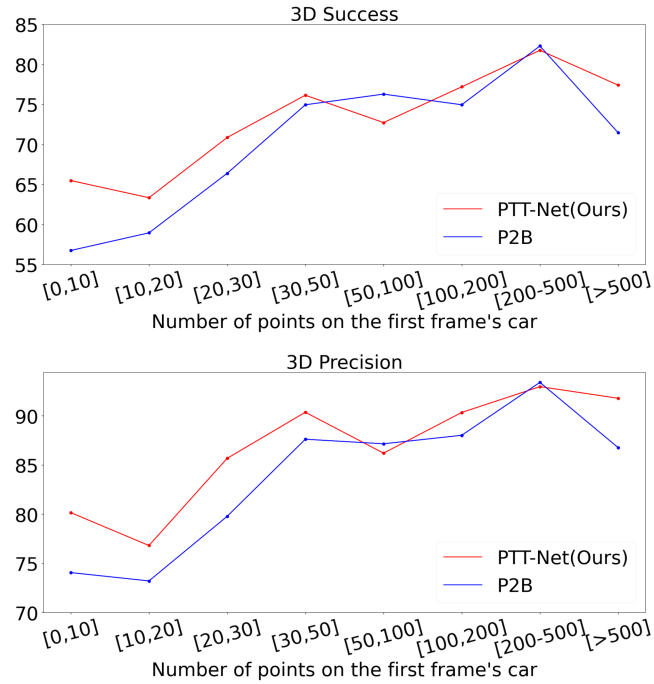

Fig. 8. The comparison of different points number intervals in the first frame between PTT and P2B. The Success and Precision results are shown in two line charts separately. Only one case that the performance of PTT is lower than P2B when the points number interval is in [50, 100]. And the average accuracy of PTT is much higher than that of P2B.

4) Comparisons in Different Points Intervals: Here, we choose the Car category and divide the tracking sequence according to the number of points in the first frame, which is used to initialize the template point cloud and plot the corresponding tracking performance curve (Fig. 8). As mentioned in Section III-B, the sparsity of point cloud limits the performance of 3D SOT trackers. Generally, in extremely sparse scenes (the number of points on the target is less than 50), most existing trackers often track off or fail. However, as shown in Fig. 8, the performance of our method outperforms P2B in both Success and Precision with a large margin when points number is less than 50. This shows that our method could deal with sparse scenes better and has more robust performance.

C. Qualitative Experiment

1) PTT Module Show: To show the effects of PTT modules in PTT-Net, we exemplified the attention points and scores when PTT-Net utilizes PTT Module or not in Fig. 9. For attention points and scores, we set the voting scores in the voting stage as attention scores. The higher the score, the more attention the corresponding points have been focused on. As shown in Fig. 9, from the first row to the fourth row, the point cloud of tracking target becomes more sparse, which makes the tracking difficulty higher. However, the results show that our PTT module can achieve robust tracking in all these different scenarios. And the attention scores tend to be higher in the location where the target features are rich. Meanwhile, we also observe that our PTT module helps network filter the background noise better and focus on the tracking target. Especially, from the visualization results in the fourth row of Fig. 9, even if there are few points, our PTT module could still help the network focus more on the foreground points, which once again proves the power of our PTT module.

2) Advantageous Cases: We visualized our advantageous cases over P2B and SC3D in Fig. 10. We can observe from Fig. 10(a) that in the sparse scenarios (less than 50 points) where both SC3D and P2B tracked off course or even failed, our PTT-Net still tracks the target robustly. In Fig. 10(b), even though P2B can track the target, their position estimation still has a large deviation in z axis. This also shows the effectiveness of our method. We could not only track the target robustly in sparse scenes, but also estimate the location information of the target more accurately.

3) Failure Cases: The Fig. 10 shows that PTT-Net can work well in most of scenes compared to SC3D and P2B. However, the sparsity of points still influences the performance of PTT-Net. Our method tends to fail when the points are extremely sparse. To show the failure case of our method in more detail, we visualized the failed tracking result when the points are less than 20. As shown in Fig. 11, our PTT-Net could not learn effective object features since there are almost no points in the initial search area. In addition, due to the sparse point cloud, the feature ambiguity also leads to the inaccurate estimation of the bounding box, which causes the propagation of errors and failure case finally.

D. Ablation Study

Here, we ablate the network architecture on KITTI dataset. First, we discuss different ways for seeds sampling, template generation, and search area generation. Then we ablate the different embedded position of our PTT module. Finally, different parameters selection of PTT module are also discussed.

1) Ways for Seeds Sampling: The first ablation study presented is designed to support our claim that FPS could benefit the classification task. As mentioned in Section III-E1, different sampling methods will lead to different degrees of target information loss. Here, we compared the effects of three different down-sampling methods: Random Sampling (RS), Farthest Point Sampling (FPS), and Feat-FPS proposed by [61] (Table V) on the performance of our method. We also showed the visualizations of three sampling methods (Fig. 4). We found that RS had the worst performance, and FPS could obtain the best performance, which was $\sim$7% higher than RS. Besides, Feat-FPS also had good tracking performance, which was only $\sim$1% lower than the FPS. We attribute this result to the fact that FPS can obtain seeds which belong to the foreground and background points uniformly. Meanwhile, FPS could keep the distribution probability of the original input point cloud to the greatest extent while reducing the dimension of the input, which will be beneficial to the classification and regression tasks of the tracking network.

2) Ways for Template Generation: Our method is consistent with P2B when generating template point clouds. Therefore, we explored the different way of template point cloud generation, including the first ground truth, previous result, the fusion of

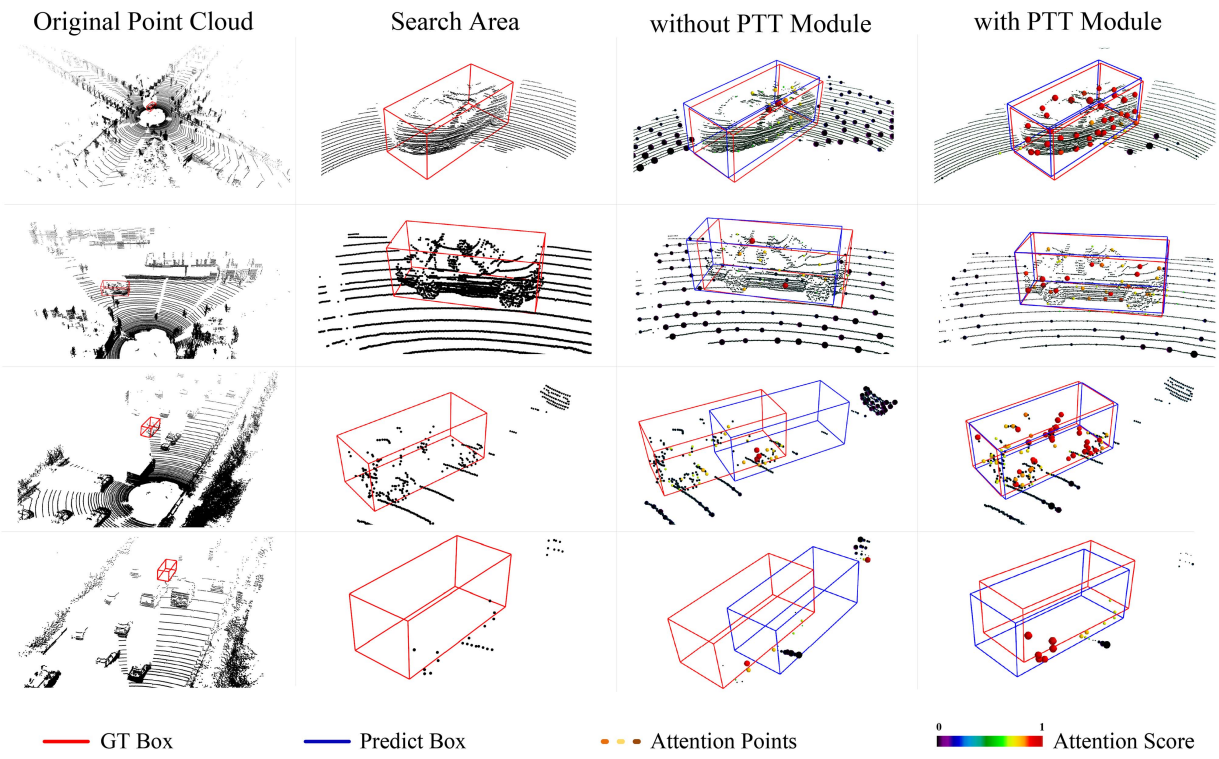

Fig. 9. Comparison of visualization results with or without PTT module. We compared the attention points and corresponding scores in the third column and the fourth column when the network had the PTT module or not. Furthermore, the tracking difficulty is increasing as the number of initial points in the search area decreases from top to bottom.

TABLE III
DIFFERENT WAYS FOR TEMPLATE GENERATION

Source of template points	Success				Precision			
	PTT-Net	3D-SiamRPN [19]	P2B [17]	SC3D [16]	PTT-Net	3D-SiamRPN [19]	P2B [17]	SC3D [16]
The First GT	**62.9**	57.2	46.7	31.6	**76.5**	75.0	59.7	44.4
Previous result	**64.9**	-	53.1	25.7	**77.5**	-	68.9	35.1
First & Preivous	**67.8**	58.2	56.2	34.9	**81.8**	76.2	72.8	49.8
All previous	**59.8**	-	51.4	41.3	**74.5**	-	66.8	57.9

"First & Previous" denotes "the first ground truth (GT) and previous result".

TABLE IV
DIFFERENT WAYS FOR SEARCH AREA GENERATION

	Success			Precesion		
	SC3D [16]	P2B [17]	PTT-Net(Ours)	SC3D [16]	P2B [17]	PTT-Net(Ours)
Previous Result	41.3	56.2	**67.8**	57.9	72.8	**81.8**
Previous GT	64.6	**82.4**	75.9	74.5	**90.1**	88.9
Current GT	76.9	**84.0**	76.1	81.3	**90.3**	89.1

TABLE V
PERFORMANCE OF DIFFERENT SAMPLING METHODS

	Random Sample	Feat-Fps	Fps
3D Success	60.4	66.1	**67.8**
3D Precision	73.7	80.0	**81.8**

the first ground truth and previous result, and all previous results. Specially, the fusion between first frame ground truth and previous results means the fusion of two point clouds within the two 3D bounding boxes respectively. First, we extract the points in the box from the point cloud according to the first frame ground-truth box and the predicted box in the previous frame. Second, according to the angle of the box and the coordinates of the center point, the two frames of point clouds are normalized to the same coordinate system by rotating and translating respectively. Finally, the updated template point clouds could be obtained by concatenating two point clouds directly. We reported the results in Table III. We found that our PTT-Net outperformed 3D-SiamRPN, P2B and SC3D in all settings.

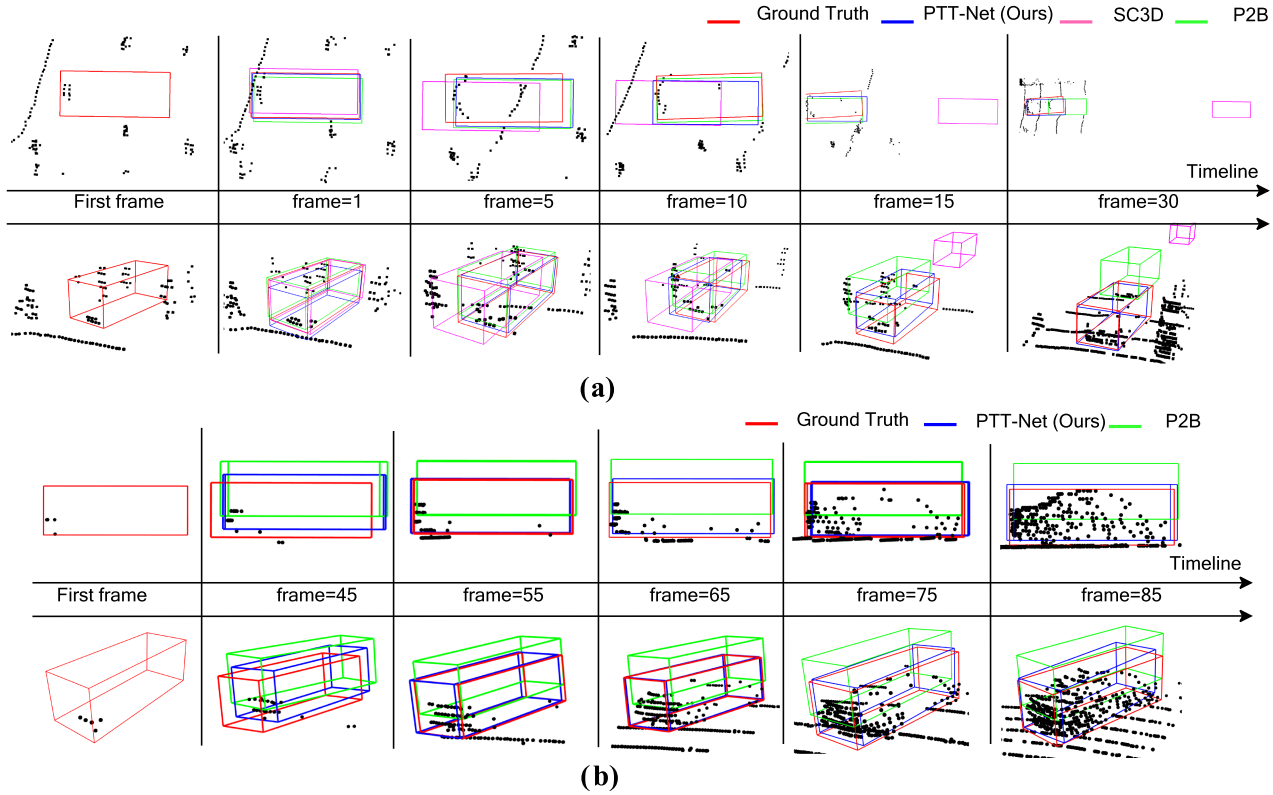

Fig. 10. Advantageous cases of PTT-Net compared with SC3D and P2B (a)–(b). In (a) and (b), the number of point clouds in the first frame is less than 50. Our method can track the target accurately. However, in scenario (a), both P2B and SC3D failed to track. In scenario (b), even though P2B could track the target, it still has an inaccurate z-axis estimation for the target. Meanwhile, SC3D has failed to track. These results show the robustness of our method in sparse point cloud scenarios. Please see our experiment video for more details.

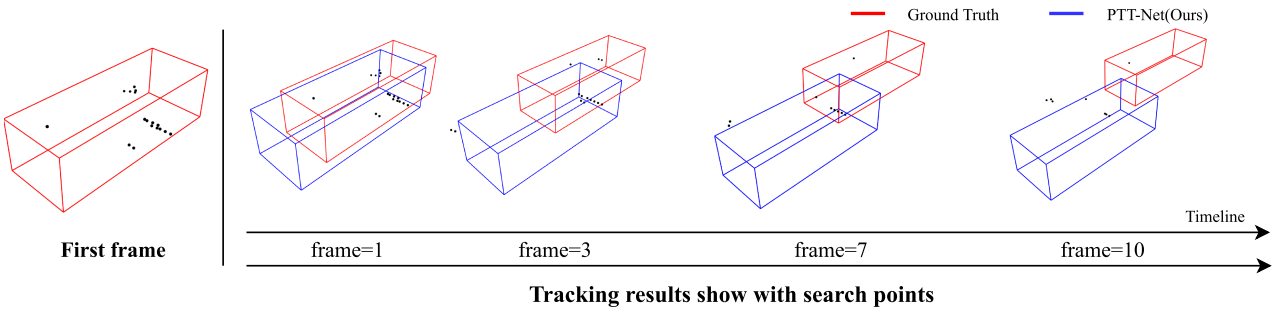

Fig. 11. Failure cases. No points in the initial search area.

Besides, the proposed method has the best performance by fusing the first ground truth and the previous result. We believe that this is because our method has more robust tracking results in sparse scenes. Therefore, if we fuse the first ground truth and the previous result to update the template points, it could further enhance the target information and improve the algorithm performance.

3) Ways for Search Area Generation: The generation strategy of search area in object tracking task directly determines the feature scale and quality that the network can learn. In previous work SC3D [15], P2B [16], there are performance comparisons in different search area situations. To further explore the performance of our method, we also conducted experiments on the search area generation strategies, and compared them with SC3D [15] and P2B [16]. The experimental results are shown in Table IV. Specifically, we compared three different search area generation methods: 1) centered on previous result; 2) centered on previous ground truth; 3) centered on current ground truth. The results show that the performance of the three methods has been greatly improved with search area generated by ground-truth. The reason is that each frame uses the ground-truth result, which can effectively avoid the accumulation of errors caused over time. However, it is worth noting that our method PTT is slightly lower than P2B after using the ground-truth, but the performance is still at a similar level.

4) Positions for PTT Module Embedding: To verify our design in Section III-E of positions where PTT modules are embedded, we tried different schemes (Table VI). The results show

TABLE VI
DIFFERENT EMBEDDED POSITIONS OF PTT MODULE

Ablation	3D Success	3D Precision
baseline [17]	56.2	72.8
Only PTT in Vote	62.1	76.9
Only PTT in Prop	65.7	78.9
PTT in all (PTT-Net)	**67.8**	**81.8**

TABLE VII
ABLATION STUDY OF HYPER-PARAMETERS IN PTT MODULE

		3D Success	3D Precision
Head Number	1	**67.8**	**81.8**
	2	67.2	80.9
	4	65.6	79.1
	8	65.5	78.1
Layer Number	1	**67.8**	**81.8**
	2	64.7	78.5
	4	64.0	76.8
	8	62.6	76.5

that embedding PTT module in both stages of [16] can obtain the best improvement. Comparing (a) with (b) in Fig. 5, PTT-Net has better point cloud classification results which focus on foreground points. Comparing (c) with (d), PTT-Net could still track the target pedestrian robustly when more proposal centers are generated from another pedestrian. Besides, as shown in Fig. 9, PTT-Net can focus on foreground points with the help of PTT module. This result effectively shows that the transformer can learn more target-wise information.

5) Parameters Selection for PTT Module: Here we discuss the details of our PTT module, including the number of heads and the number of attention layers, as shown in Table VII. For the number of heads, we observe that $head = 1$ and $layer = 1$ achieves the best performance, and stacking more heads or layers could not bring in performance improvement but more parameters and lower speed. We believe that since our PTT module is directly applied on the fusion feature, which already has the similarity representations, more heads or layers in PTT may make the feature focus on other unimportant features, thus distracting the already fused similarity features. Therefore, we set both the heads and layers to 1.

E. Timing Breakdown

We calculated the average running time of all test frames for car to measure PTT-Net's speed. PTT-Net achieved 40 FPS on a single NVIDIA 1080Ti GPU, including 8.3 ms for preparing point cloud, 16.2 ms for model forward propagation, and 0.5 ms for post-processing. The running time of SC3D [15], P2B [16] and 3D-SiamRPN [18] on the same platform are 1.8FPS, 45.5FPS and 20.8FPS, respectively.

V. CONCLUSION

In this work, we explored the application of transformer network in 3D SOT task and proposed PTT module. The PTT module aims at weighing point cloud features to focus on the important features of objects. We also embedded the PTT module into the open-source state-of-the-art method [16] to build a PTT-Net 3D tracker. Experiments show that PTT-Net outperforms previous state-of-the-art methods with remarkable margins. We hope that our work will inspire further investigation of the application of transformers to 3D object tracking.

REFERENCES

[1] A. I. Comport, É. Marchand, and F. Chaumette, "Robust model-based tracking for robot vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., (IEEE Cat. No.04CH37566)*, 2004, vol. 1, pp. 692–697.
[2] U. Kart, A. Lukezic, M. Kristan, J. K. Kamarainen, and J. Matas, "Object tracking by reconstruction with view-specific discriminative correlation filters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1339–1348.
[3] E. Machida, M. Cao, T. Murao, and H. Hashimoto, "Human motion tracking of mobile robot with Kinect 3D sensor," in *Proc. SICE Annu. Conf.*, 2012, pp. 2207–2211.
[4] A. Bibi, T. Zhang, and B. Ghanem, "3d part-based sparse tracker with automatic synchronization and registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1439–1448.
[5] Y. Liu *et al.*, "Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in RGB-D videos," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 664–677, Mar. 2019.
[6] U. Kart, J. K. Kamarainen, and J. Matas, "How to make an RGBD tracker?," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 148–161.
[7] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3569–3577.
[8] 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*, 2019, pp. 1190–1199.
[9] 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. Automat. Lett.*, vol. 5, no. 2, pp. 3206–3212, Apr. 2020.
[10] R. Q. Charles, S. Hao, 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.*, 2017, pp. 652–660.
[11] 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.*, 2020, pp. 10359–10366.
[12] A. Shenoi *et al.*, "JRMOT: A real-time 3d multi-object tracker and a new large-scale dataset," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10335–10342.
[13] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "Fantrack: 3d multi-object tracking with feature association network," in *Proc. IEEE Intell. Veh. Symp.*, 2019, pp. 1426–1433.
[14] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3569–3577.
[15] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3d siamese tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1359–1368.
[16] H. Qi, C. Feng, Z. Cao, F. Zhao, and Y. Xiao, "P2B: Point-to-box network for 3D object tracking in point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6328–6337.
[17] H. Du, L. Li, B. Liu, and N. Vasconcelos, "SPOT: Selective point cloud voting for better proposal in point cloud object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 230–247.
[18] Z. Fang, S. Zhou, Y. Cui, and S. Scherer, "3d-SiamRPN: An end-to-end learning method for real-time 3D single object tracking using raw point cloud," *IEEE Sensors J.*, vol. 21, no. 4, pp. 4995–5011, 2021.

[19] S. Shi, X. Wang, and H. Li, "PointRCNN: 3d object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.
[20] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[21] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[22] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol. - Proc. Conf.*, 2019, pp. 4171–4186.
[23] A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[24] H. Caesar *et al.*, "nuscenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11 618–11 628.
[25] J. Shan, S. Zhou, Z. Fang, and Y. Cui, "PTT: Point-track-transformer module for 3D single object tracking in point clouds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1310–1316.
[26] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2544–2550. [Online]. Available: https://ieeexplore.ieee.org/document/5539960/
[27] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.
[28] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6638–6646.
[29] H. K. Galoogahi, T. Sim, and S. Lucey, "Correlation filters with limited boundaries," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4630–4638.
[30] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 850–865.
[31] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8971–8980.
[32] B. Li, W. Wu, Q. Wang, F. Zhang, and J. Yan, "SiamRPN++: Evolution of siamese visual tracking with very deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4282–4291.
[33] X. Dong and J. Shen, "Triplet loss in siamese network for object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 472–488.
[34] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: An unifying approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1328–1338.
[35] P. Voigtlaender, J. Luiten, P. H. S. Torr, and B. Leibe, "Siam R-CNN: Visual tracking by re-detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6577–6587.
[36] S. Tian, X. Liu, M. Liu, S. Li, and B. Yin, "Siamese tracking network with informative enhanced loss," *IEEE Trans. Multimedia*, vol. 23, pp. 120–132, 2021.
[37] W. Han, X. Dong, F. S. Khan, L. Shao, and J. Shen, "Learning to fuse asymmetric feature maps in siamese trackers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16565–16575.
[38] X. Dong, J. Shen, L. Shao, and F. M. Porikli, "CLNet: A. compact latent network for fast adjusting siamese trackers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 378–395.
[39] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, and F. M. Porikli, "Dynamical hyperparameter optimization via deep reinforcement learning in tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1515–1529, May 2021.
[40] X. Chen *et al.*, "Transformer tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8122–8131.
[41] J. Zarzar, S. Giancola, and B. Ghanem, "Efficient tracking proposals using 2d-3d siamese networks on lidar," 2019, *arXiv:1903.10168*.
[42] Y. Cui, Z. Fang, and S. Zhou, "Point siamese network for person tracking using 3D point clouds," *Sensors*, vol. 20, no. 1, p. 143, Dec. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/20/1/143
[43] H. Zou *et al.*, "F-siamese tracker: A frustum-based double siamese network for 3d single object tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 8133–8139.
[44] H. Hu, Z. Zhang, Z. Xie, and S. Lin, "Local relation networks for image recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3463–3472.
[45] P. Ramachandran *et al.*, "Stand-alone self-attention in vision models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 68–80.
[46] N. Carion *et al.*, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, vol. 12346, pp. 213–229.
[47] H. Zhong *et al.*, "Self-adaptive neural module transformer for visual question answering," *IEEE Trans. Multimedia*, vol. 23, pp. 1264–1273, 2021.
[48] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 073–10 082.
[49] H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun, "Point transformer," 2020, *arXiv:2012.09164*.
[50] V. B. Nico Engel and K. Dietmayer, "Point transformer," 2020, *arXiv:2011.00931*.
[51] M. H. Guo *et al.*, "PCT: Point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, 2021.
[52] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, "3D object detection with pointformer," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2021, pp. 7463–7472.
[53] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer, "TrackFormer: Multi-object tracking with transformers," 2021, *arXiv:2101.02702*.
[54] P. Sun *et al.*, "TransTrack: Multiple-object tracking with transformer," 2020, *arXiv:2012.15460*.
[55] P. Chu, J. Wang, Q. You, H. Ling, and Z. Liu, "TransMOT: Spatial-temporal graph transformer for multiple object tracking," 2021, *arXiv:2104.00194*.
[56] N. Wang, W. gang Zhou, J. Wang, and H. Li, "Transformer meets tracker: Exploiting temporal context for robust visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1571–1580.
[57] J. Shen, X. Tang, X. Dong, and L. Shao, "Visual object tracking by hierarchical attention siamese network," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3068–3080, Jul. 2020.
[58] 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.
[59] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3 d object detection in point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9277–9286.
[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[61] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3d single stage object detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11037–11045.
[62] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–8.
[63] C. Zheng *et al.*, "Box-aware feature enhancement for single object tracking on point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 13199–13208.
[64] M. Kristian, J. Matas, A. Leonardis, and F. Porikli, "A novel performance evaluation methodology for single-target trackers," in *Proc. IEEE Trans. Pattern Anal. Mach. Intell.*, 2016, pp. 2137–2155.

Jiayao Shan received the B.S. degree from Dalian Jiaotong University, Dalian, China. He is currently working toward the M.S. degree under the supervision of Prof. Zheng Fang in Northeastern University, Shenyang, China. His research interests include deep learning, lidar perception, and 3D object detection and tracking.

Sifan Zhou received the B.S. degree from Civil Aviation University of China, Tianjin, China, in 2018, and the M.S. degree from Northeastern University, Shenyang, China, in 2021. His research interests include deep learning, visual perception, and 3D object tracking.

Yubo Cui received the B.S. degree in automation and the M.S. degree in robot science and engineering from Northeastern University, Shenyang, China, in 2017 and 2020, respectively. He is currently working toward the Ph.D. degree under the supervision of Prof. Zheng Fang in Northeastern University. His research interests include 3D computer vision, deep learning and machine learning.

Zheng Fang (Member, IEEE) received the B.S. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from Northeastern University, Shenyang, China, in 2002 and 2006, respectively. From 2013 to 2015, he was a Postdoctoral Research Fellow with Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Full Professor with the Faculty of Robot Science and Engineering, Northeastern University. His research interests include visual/laser SLAM, perception and autonomous navigation of various mobile robots.