

# PTT: Point-Track-Transformer Module for 3D Single Object Tracking in Point Clouds

Jiayao Shan<sup>†</sup>, Sifan Zhou<sup>†</sup>, Zheng Fang\*, Yubo Cui

**Abstract**— 3D single object tracking is a key issue for robotics. In this paper, we propose a transformer module called Point-Track-Transformer (PTT) for point cloud-based 3D single object tracking. PTT module contains three blocks for feature embedding, position encoding, and self-attention feature computation. Feature embedding aims to place features closer in the embedding space if they have similar semantic information. Position encoding is used to encode coordinates of point clouds into high dimension distinguishable features. Self-attention generates refined attention features by computing attention weights. Besides, we embed the PTT module into the open-source state-of-the-art method P2B to construct PTT-Net. Experiments on the KITTI dataset reveal that our PTT-Net surpasses the state-of-the-art by a noticeable margin ( $\sim 10\%$ ). Additionally, PTT-Net could achieve real-time performance ( $\sim 40\text{FPS}$ ) on NVIDIA 1080Ti GPU. Our code is open-sourced for the robotics community at <https://github.com/shanjiayao/PTT>.

## I. INTRODUCTION

3D single object tracking (SOT) has a wide range of applications in robotics and autonomous driving [1], [2]. However, most existing 3D SOT methods are equipped with RGB-D cameras [3], [4], which inherit the characteristics of 2D images and depend heavily on RGB-D information, thus trackers may fail in visually degraded or illumination changing environments.

In addition to RGB-D sensors, 3D LIDAR sensors are also widely used in object tracking tasks [5], [6] because they are less sensitive to illumination changes and could directly capture geometric information more accurately. However, using only point clouds for 3D SOT has its challenges. *First*, point cloud is sparse and disordered [7], which requires the network to be permutation-invariant. *Second*, 3D object tracking needs to estimate higher space dimension (e.g.  $x, y, z, w, h, l, ry$ ) than 2D visual tracking, which takes more computational complexity. *Third*, compared with rigid object tracking (e.g. car), it's more challenging to track non-rigid objects (e.g. pedestrian) since it is too hard to extract stable features.

Currently, most point cloud based 3D SOT methods follow 2D visual object tracking (VOT) approaches (e.g. Siamese networks [8], [9]) to track 3D target. The Siamese network formulates VOT task as learning a similarity function between the template branch and the search branch. SC3D [10] is the pioneer point cloud based 3D Siamese Tracker based on shape completion network. However, it could not

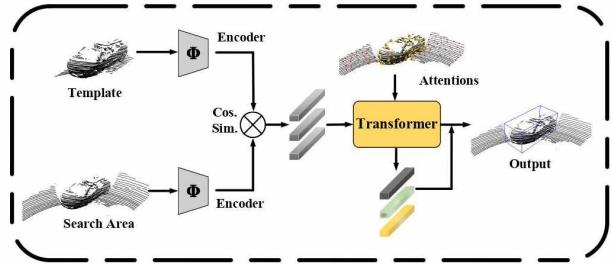


Fig. 1. Exemplified illustration to show how PTT module works. Compared with the existing 3D single object tracking method, our PTT module works after calculating the similarity features, and weighs the features based on their importance to improve tracking performance.

be trained end-to-end and run in real-time. Besides, Qi et al. [11] also proposed a point-to-box (P2B) network to estimate target bounding box from the raw point cloud. However, this approach does not cope well with sparse scenarios. Recently, Fang et al. [12] jointed Siamese network and point cloud based Region Proposal Network (RPN) [13] to tackle 3D SOT task. Nonetheless, the performance of their method is unsatisfying. 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 due to the sparsity of point clouds. Therefore, how to pay attention to the important clues is the key to improving the performance of the 3D object tracker.

Recently, transformer has revolutionized natural language processing and image analysis [14]–[16]. 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 [14], [17]. Therefore, transformer is suitable for point cloud due to its positional attributes.

In this paper, we explore the application of the transformer network for 3D SOT task, and propose a transformer module named PTT (Point-Track-Transformer). To pay more attention to the important feature of the object, we use the transformer's powerful self-attention and position encoding mechanism to weigh the point cloud features. To evaluate the effect of our PTT module, we embed our PTT module into the open-source state-of-the-art method P2B [11] to build a new network called PTT-Net. Finally, the experimental results of our PTT-Net on KITTI [18] dataset demonstrate

<sup>†</sup>Authors with equal contribution.

The authors are with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang, China; Corresponding author: Zheng Fang, e-mail: fangzheng@mail.neu.edu.cn

the superiority of our method ( $\sim 10\%$  improvement on both Success and Precision). Besides, PTT-Net could run at 40FPS.

Overall, there are three main contributions as follows:

- **PTT module:** a Point-Track-Transformer (PTT) module for 3D single object tracking using only point clouds, weighing point cloud features to focus on deeper-level object clues during tracking.

- **PTT-Net:** a 3D single object tracking network named PTT-Net embedded with PTT modules which can be end-to-end trained. To the best of our knowledge, this is the first work to apply transformer to 3D object tracking task based on point cloud.

- **Open-source:** Experiments on KITTI tracking dataset [18] show our method outperforms the state-of-the-art methods with remarkable margins. Besides, we open source our method to the research community.

## II. RELATED WORK

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

### A. 3D SOT Using Point Cloud

Giancola et al. [10] proposed the first pioneer point cloud based 3D single object tracker which utilized the Kalman Filter to generate massive target proposals. They introduced the shape completion module to enrich the feature learning on points. However, their method has a poor generalization ability and could not run in real-time. Zarzar et al. [19] leveraged 2D Siamese network which converted raw point clouds into Bird-Eye-View (BEV) representation to generate 3D proposals. This method may lose fine-grain geometry details which are important for tracking tiny objects. Cui et al. [20] also adopted a 3D Siamese tracker only using point cloud. However, they could not estimate the orientation and size information of the target. Fang et al. [12] jointed 3D Siamese network and 3D RPN network to track targets, but their performance is limited by the one-stage RPN network. Besides, Zou et al. [21] integrated 2D image and 3D point cloud information for 3D SOT. However, this method relies more on 2D trackers and the performance is not satisfactory when using previous results to initialize tracker. Qi et al. [11] proposed P2B which used deep hough voting to obtain the potential centers (votes) and estimated target center based on those votes. However, it ignores the fact that points in different positions have different contributions to tracking. Furthermore, its random sampling mechanism loses the location distribution information of the raw point cloud. Based on these shortcomings, we propose a PTT module to weigh different point features and use farthest point sampling instead of random sampling to obtain more raw point cloud information.

### B. Transformer and Self-attention

Recently, there have been many wonderful works based on Transformer [14], [15]. Hu et al. [15] and Ramachandran

et al. [16] applied scalar dot product self-attention in local pixel neighbors. Zhao et al. [17] utilized vector self-attention operations to image tasks. Inspired by these works, Zhao et al. [22] used a Point Transformer layer by applying vector self-attention operations, which had a great performance improvement in point cloud classification and segmentation tasks. Nico et al. [23] proposed SortNet as a part of Point Transformer, and achieved competitive performance on point cloud classification and partial segmentation tasks. Meanwhile, Guo et al. [24] also introduced Point Cloud Transformer (PCT), which performed well on shape classification, part segmentation, and normal estimation tasks. Obviously, transformer has unique advantages for point cloud feature learning. We hence aim to extend the transformer paradigm to our 3D SOT task with more attention to features.

## III. METHODOLOGY

Here, we focus on the application of transformer network in 3D SOT. Given an input of  $M$  points with XYZ coordinates, a backbone network modified on the basis of PointNet++ [25] is used to extract the point cloud and learn deep features. It outputs a subset of the input containing  $N$  interest points (seeds)  $S = \{s_i\}_{i=1}^N$ .  $s_i = (c_i, f_i)$  is composed by 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 the feature space of  $f_i$ , and output refined features  $f_i^*$  with the same dimension.

### A. Transformer

The 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:

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

where  $\alpha$ ,  $\beta$  and  $\gamma$  are point-wise feature transformations (e.g. linear layers or MLPs).  $Q$ ,  $K$ , and  $V$  are the *query*, *key* and *value* matrices, respectively.  $\sigma$  is the relation function between  $Q$  and  $K$ .  $P$  is the position encoding feature.  $\rho$  is a normalization function (e.g. *Softmax*).  $\odot$  means Hadamard product.  $\mathcal{A}$  is the attention feature produced by self-attention layer.

### B. PTT Module

We modify the transformer module proposed in Point Transformer [22] to weigh the point cloud features. In [22], the point transformer layer is proposed to process the raw point cloud for classification and segmentation tasks. Here, we set an importance-based transformer module to focus on the differences among the input point cloud features for 3D SOT task.

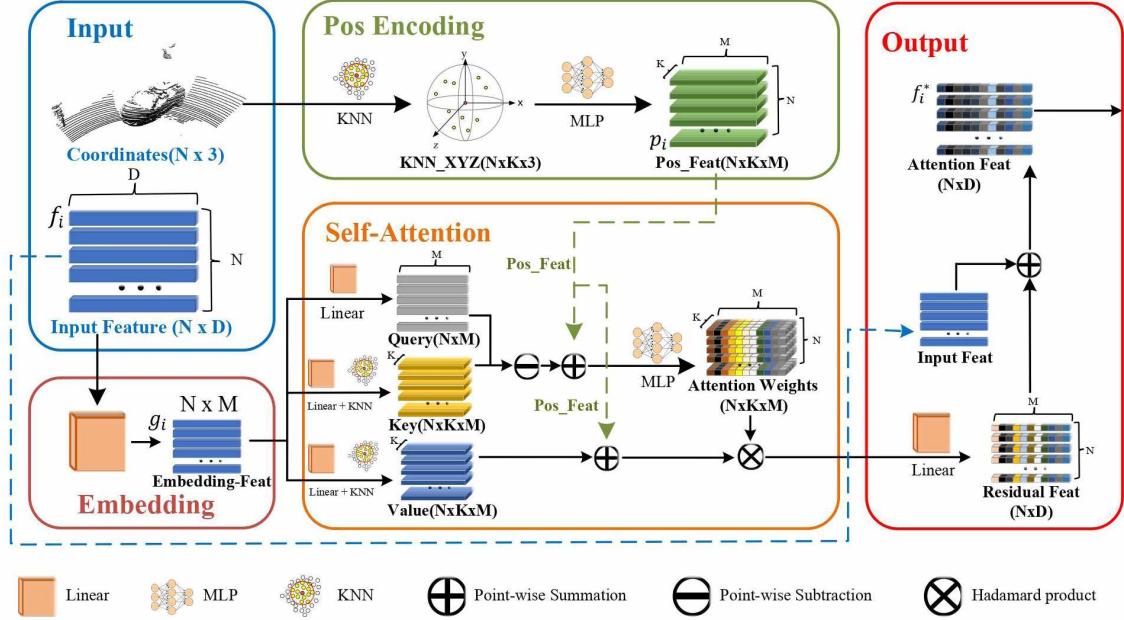


Fig. 2. **PTT module architecture.** It consists of three blocks: feature embedding, position encoding, and self-attention. The whole inputs are the coordinates and their corresponding features. Feature embedding module maps input features into embedding space. In position encoding module, the k-nearest neighbor algorithm is used to obtain local position information, then the encoded position features will be learned by a Multi Layer Perceptron (MLP). 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.

Therefore, we explore how to embed transformer modules into 3D SOT task and propose our Point-Track-Transformer (PTT) module. 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  $\mathbb{R}^M$ :  $f_i \rightarrow g_i$ ,  $g_i \in \mathbb{R}^M$ . Position encoding module extracts higher-level  $M$ -dimensional features  $p_i$  from input coordinates  $c_i$ :  $c_i \rightarrow p_i$ ,  $p_i \in \mathbb{R}^{K \times M}$ . 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 training stage, we also adopt the residual architecture in [22], 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 (NLP) 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  $M$ :  $\mathbb{R}^D \rightarrow \mathbb{R}^M$ , which can place the feature closer in the embedding space if 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 [14]. And 3D point coordinates themselves are the natural input for position

encoding. Therefore, we utilize the coordinates directly as the input of the position encoding module. Besides, we use the relative coordinates to make the network better capture the spatial correlation between points and local geometric shape information. Since the feature  $f_i$  is extracted by [25] which can provide the local context information, we obtain the position encoding features  $P = \{p_i\}_{i=1}^N$  with function  $\eta$ . For input point set  $S$  including  $N$  points, the position encoding feature for each point is:

$$p_i = \eta(c_i - c_j) \quad (2)$$

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  $c_i$  by using K-nearest-neighbor (KNN). And  $\eta$  is an MLP with two linear layers and one ReLU layer.

3) *Self-Attention:* As Fig. 2 shows, self-attention module computes three vectors for each point:  $Q$ ,  $K$ ,  $V$  through  $\alpha$ ,  $\beta$ ,  $\gamma$ , where  $\alpha$ ,  $\beta$ ,  $\gamma$  are all linear layers. It is worth noting that  $K$  and  $V$  are aggregated from the features of the  $k$  neighborhood points, which aim to encode more local context information. Here,  $Q \in \mathbb{R}^M$ ,  $K \in \mathbb{R}^{K \times M}$ , and  $V \in \mathbb{R}^{K \times M}$ .

The relation function  $\sigma$  can be classified into two types: scalar [14] and vector [17]. And it has been proved in [22] that vector attention is more suitable for point cloud than scalar attention since it supports adaptive modulation of individual feature channels, not just whole feature vectors. Thus we set  $\sigma(Q, K) = Q - K$  to obtain point-wise attention weights. And an MLP layer  $\gamma$  is used to introduce additional trainable transformations and match the output

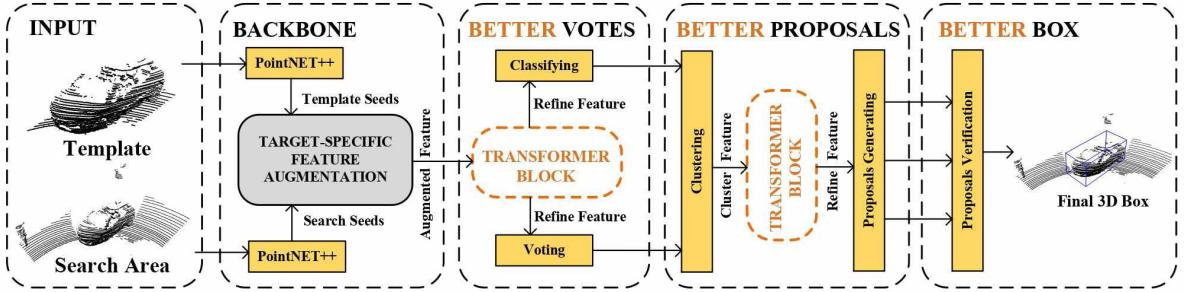


Fig. 3. **The pipeline of PTT-Net.** In order to verify the effect of our PTT module, we embedded two PTT modules into seeds voting stage and proposals generation stage of [11]. The first transformer block takes the augmented feature as input, and outputs the refine features for voting. The second transformer block refines the cluster features to help PTT-Net obtain better proposals.

dimension. Then, we add the position encoding features  $P$  to both the attention vector  $\sigma$  and the transformed features  $K$ . Finally, the residual features recorded as  $A$  are defined as the weighted sum of the attention weights with all  $V$  vectors. The formula is as follows:

$$A = \rho(\gamma(Q - K + P)) \odot (V + P) \quad (3)$$

where  $\rho$  is a normalization function (Softmax) and  $\gamma$  is a non-linear mapping function (MLP) that includes two linear layers and one ReLU layer.  $A$  is attention features.

### C. PTT-Net

The ability of the transformer to learn self-attention weights inspires us to try it on 3D SOT task. We formulate the problem of focusing on the differences in features as self-attention weighting. In order to verify the effect of our method, we embed our PTT module into P2B [11]. More specifically, the PTT modules are inserted in seeds voting stage and proposals generation stage of P2B.

In seeds voting stage, P2B generates votes based on the augmented features, which are from backbone in Fig. 3. We notice that [11] 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 and suppress the background noise. Therefore, we apply PTT module to weigh the augmented features and obtain the weighted features focused on foreground points (in Fig. 4(a)(b)).

In proposals generation stage, P2B generates proposals based on local context features. However, their method ignores the global semantic features of targets, so that they could not distinguish similar objects (e.g. two pedestrians, in Fig. 4(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.

As shown in Fig. 3, we embed our PTT module in the P2B [11] 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 [11] with remarkable margins.

1) *Loss Function:* The PTT module is trained with the other subnetworks in [11]. So we follow [11] to design our loss function. The overall loss consists of two parts as follows:

$$L_{all} = L_{cv} + \lambda_1 L_{cb} + \lambda_2 L_{rv} + \lambda_3 L_{rb} \quad (4)$$

where  $\lambda_1, \lambda_2, \lambda_3$  represent the weighting coefficient of each loss. Classification loss includes voting classification loss  $L_{cv}$  and 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 [18] as the benchmark, and set up more detailed experiments to show the superiority of our PTT-Net by comparing the different performance of the dominant methods [10]–[12], [21] on rigid (Car, Van) and non-rigid (Pedestrian, Cyclist) objects. The experiments show that PTT-Net outperforms the previous SOTA method with remarkable margins at 40fps.

### A. Experimental protocols

1) *Dataset:* We used the training set of KITTI which includes more than 20,000 manually labeled 3D objects using Velodyne HDL-64E 3D lidar (10HZ). For the splits of dataset, we follow [10]–[12], [21], which divide 20 sequences into three parts 00-16, 17-18, 19-20, corresponding to training set, validation set, and test set respectively.

2) *Evaluation Metric:* Following previous work [10]–[12], [21], we report Success and Precision metrics defined by One Pass Evaluation (OPE) [26], which represent overlap and error Area Under the Curve (AUC) respectively.

3) *Implementation Details:* We use the farthest point sampling (FPS) instead of random sampling (RS) in origin P2B [11]. In 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 training epoch is 60. Besides, we extend the offset from  $(x, y, \theta)$  to  $(x, y, z, \theta)$  when generating more template samples during data augmentation in [11]. In testing stage, we also add  $Z$

TABLE I

PERFORMANCE COMPARISON ON THE KITTI DATASET FOR THE CAR CATEGORY.

| Module               | Modality | 3D Success        | 3D Precision | FPS  |
|----------------------|----------|-------------------|--------------|------|
| AVOD-Tracking [27]   | R+L      | 63.1              | 69.7         | -    |
| F-Siamese [21]       | R+L      | 37.1              | 50.6         | -    |
| SC3D [10]            | L        | 41.3              | 57.9         | 1.8  |
| P2B [11]             | L        | 56.2              | 72.8         | 45.5 |
| 3DSiamRPN [12]       | L        | 58.2 <sup>2</sup> | 76.2         | 20.8 |
| <b>PTT-Net(Ours)</b> | L        | 67.8 <sup>1</sup> | 81.8         | 40.0 |

R and L mean RGB and LiDAR respectively. <sup>1</sup> <sup>2</sup>Red and blue mean the performance score is ranked first and second respectively.

TABLE II

EXTENSIVE COMPARISONS WITH DIFFERENT CATEGORIES. "PED" DENOTES "PEDESTRIAN".

|           | Category             | Car  | Ped  | Van  | Cyclist | Mean  |
|-----------|----------------------|------|------|------|---------|-------|
|           | Frame Number         | 6424 | 6088 | 1248 | 308     | 14068 |
| Success   | SC3D [10]            | 41.3 | 18.2 | 40.4 | 41.5    | 31.2  |
|           | P2B [11]             | 56.2 | 28.7 | 40.8 | 32.1    | 42.4  |
|           | FSiamese [21]        | 37.1 | 16.2 | -    | 47.0    | -     |
|           | 3DSiamRPN [12]       | 58.2 | 35.2 | 45.6 | 36.1    | 46.6  |
|           | <b>PTT-Net(Ours)</b> | 67.8 | 44.9 | 43.6 | 37.2    | 55.1  |
| Precision | SC3D [10]            | 57.9 | 37.8 | 47.0 | 70.4    | 48.5  |
|           | P2B [11]             | 72.8 | 49.6 | 48.4 | 44.7    | 60.0  |
|           | FSiamese [21]        | 50.6 | 32.2 | -    | 77.2    | -     |
|           | 3DSiamRPN [12]       | 76.2 | 56.2 | 52.8 | 49.0    | 64.9  |
|           | <b>PTT-Net(Ours)</b> | 81.8 | 72.0 | 52.5 | 47.3    | 74.2  |

axis offset to generate predicted box. Other parameters are consistent with settings of [11].

### B. Quantitative Experiments

To better evaluate our method, we designed two quantitative experiments on KITTI dataset. 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 on Pedestrian, Van, and Cyclist.

1) *Comparisons on car category*: We compared the performance of our PTT-Net with the existing methods on the KITTI dataset and reported results for 3D car tracking in Tab. I. In order to fit the requirement of real scenarios, we generate the search area centered on the previous result. The results show our PTT-Net has achieved SOTA performance in all evaluation metrics. Compared with the baseline algorithm P2B [11], our performance has been greatly improved by  $\sim 11\%$ . Additionally, compared with [27] and [21] 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.

2) *Comparisons on other categories*: We also compared with the dominant methods on Pedestrian, Van, and Cyclist (Tab. II). The average performance of PTT-Net outperforms P2B [11]  $\sim 13\%$ . 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. This also proves that

TABLE III

DIFFERENT WAYS FOR TEMPLATE GENERATION. "GT" DENOTES "GROUND TRUTH". "FIRST & PREVIOUS" DENOTES "THE FIRST GROUND TRUTH AND PREVIOUS RESULT".

|           | Method               | The First GT | Previous Result | First & Previous | All Previous |
|-----------|----------------------|--------------|-----------------|------------------|--------------|
| Success   | SC3D [10]            | 31.6         | 25.7            | 34.9             | 41.3         |
|           | P2B [11]             | 46.7         | 53.1            | 56.2             | 51.4         |
|           | 3DSiamRPN [12]       | 57.2         | -               | 58.2             | -            |
|           | <b>PTT-Net(Ours)</b> | <b>62.9</b>  | <b>64.9</b>     | <b>67.8</b>      | <b>59.8</b>  |
| Precision | SC3D [10]            | 44.4         | 35.1            | 49.8             | 57.9         |
|           | P2B [11]             | 59.7         | 68.9            | 72.8             | 66.8         |
|           | 3DSiamRPN [12]       | 75.0         | -               | 76.2             | -            |
|           | <b>PTT-Net(Ours)</b> | <b>76.5</b>  | <b>77.5</b>     | <b>81.8</b>      | <b>74.5</b>  |

TABLE IV  
DIFFERENT EMBEDDED POSITIONS OF PTT MODULE.

| Ablation            | 3D Success  | 3D Precision |
|---------------------|-------------|--------------|
| baseline [11]       | 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) | <b>67.8</b> | <b>81.8</b>  |

our PTT module can help the network understand and learn the important characteristics of the target better.

### C. Ablation Study

1) *Template Area Generation*: We explored four different settings of template point cloud generation, including the first ground truth, previous result, the fusion of the first ground truth and previous result, and all previous results. We reported results in Tab. IV-C. Obviously, our method achieves SOTA performances in all settings.

2) *Embedding location of PTT module*: To verify our design in Sec. III-C of positions where PTT modules are embedded, we tried different schemes (Tab. IV-C). The results show that embedding PTT module in both two stages of [11] can obtain the best improvement. Besides, as shown in Fig. 4, compared with (a) and (b), PTT-Net has better point cloud classification results which focus on foreground points. Comparing (c) with (d), PTT-Net could still track target pedestrian robustly when more proposal centers are generated from another pedestrian. This result effectively shows that transformer can learn more target-wise information.

### D. Qualitative Experiment

Attention can be understood as the place where the network focuses on. It is obvious that PTT module guides tracker in focusing on foreground points even if they are few.

1) *Advantageous cases*: We first exemplified the attention score in the voting stage in PTT-Net in Fig. 5. To further demonstrate the performance of our method, we selected four scenarios according to the type and difficulty of the tracking target. It is obvious that PTT module guides tracker

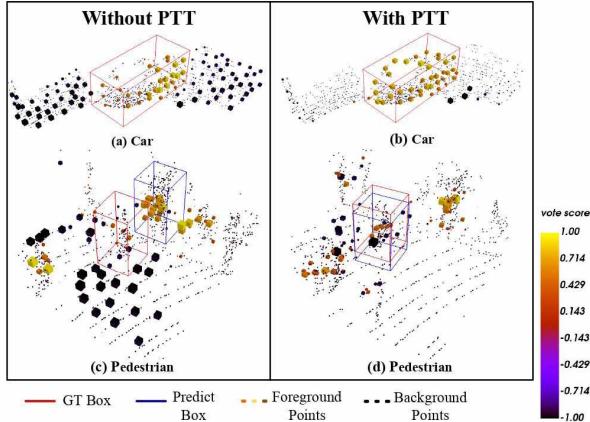


Fig. 4. **Visualization of classification(a-b) and tracking(c-d) results with or without PTT module.** The point will be paid more attention if it has a higher score. Compared (a) with (b), PTT module pays more attention to the foreground points. Compared (c) with (d), PTT module could still track targets robustly in crowded scenes (with multiple pedestrians).

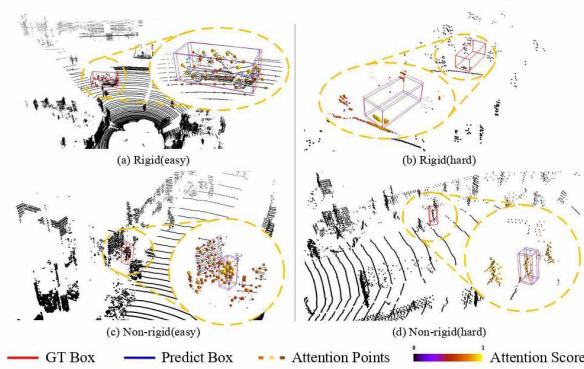


Fig. 5. **Exemplified illustration to show the attention scores of rigid cases(a-b) and non-rigid cases(c-d).** And we also divide the easy and hard cases according to the number of foreground points. It is obvious that PTT module guides tracker in focusing on foreground points even if they are few or extremely similar to others.

in focusing on foreground points even if they are few by comparing (a) and (b). Besides, (c) and (d) also show good performance in non-rigid object (pedestrian) tracking, even if the target and background are extremely similar (d), PTT module can distinguish them.

We then visualized our advantageous cases over P2B and SC3D in Fig. 7. We can observe that both SC3D and P2B tracked eventually failed in the sparse scenarios (less than 50 points), but our PTT-Net tracks the target tightly.

**2) Failure cases:** To show the performance of our method in more detail, we exemplified the impact of varying density of points on the first frame's car to PTT-Net in Fig. 6. The orange points indicate failure cases that the number of points from the initial frame are mostly less than 20. And even if four tracklets are initialized with 0 points. In these cases, our PTT-Net could not learn effective object characteristics, so it fails to track. As shown in Fig. 8, our PTT-Net could not learn effective object characteristics since no points in the

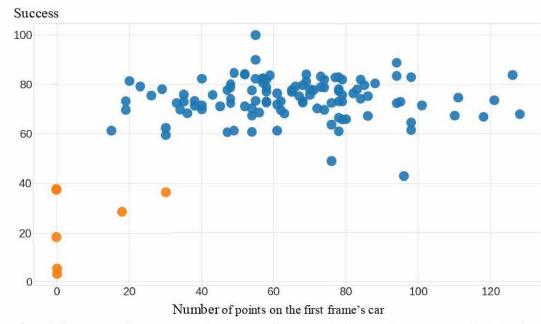


Fig. 6. **The number of points on the first frame's car and 3D Success.** There are 120 points in this scatter figure corresponding to 120 tracklets in testing sequences (19-20). Orange points indicate tracking off course or even failed.

initial search area.

#### E. Timing Breakdown

We calculated the average running time of all test frames in the Car category to evaluate running 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 [10], P2B [11] and 3D-SiamRPN [12] on the same platform are 1.8FPS, 45.5FPS and 20.8FPS, respectively.

## V. CONCLUSIONS

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 modules into the open-source state-of-the-art method [11] and construct a novel 3D SOT tracker named PTT-Net. 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.

## ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (62073066, U20A20197), Science and Technology on Near-Surface Detection Laboratory (6142414200208), the Fundamental Research Funds for the Central Universities (N182608003), Major Special Science and Technology Project of Liaoning Province (No.2019JH1/10100026), and Aeronautical Science Foundation of China (No. 201941050001).

## REFERENCES

- [1] U. Kart, A. Lukei, M. Kristan, J. Kämäräinen, and J. Matas, "Object tracking by reconstruction with view-specific discriminative correlation filters," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1339–1348, 2019.
- [2] E. Machida, M. Cao, T. Murao, and H. Hashimoto, "Human motion tracking of mobile robot with kinect 3d sensor," *2012 Proceedings of SICE Annual Conference (SICE)*, pp. 2207–2211, 2012.
- [3] Y. Liu, X.-Y. Jing, J. Nie, H. Gao, J. Liu, and G.-P. Jiang, "Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in rgb-d videos," *IEEE Transactions on Multimedia*, pp. 664–677, 2018.

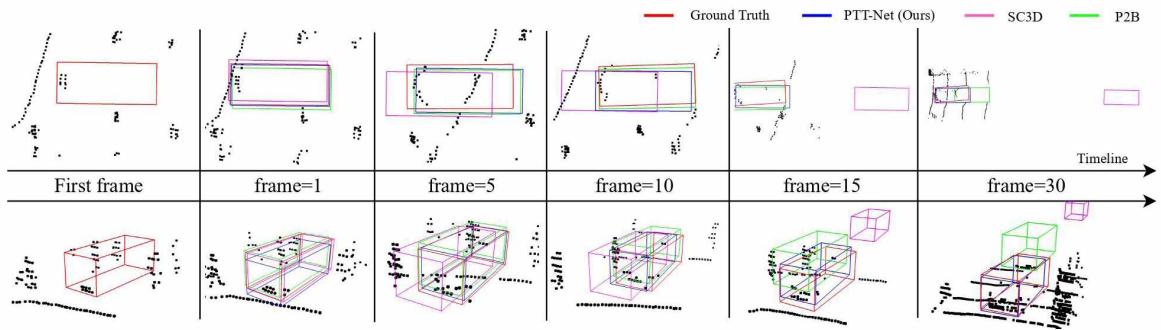


Fig. 7. **Advantageous cases of PTT-Net (ours) compared with SC3D and P2B.** In sparse point cloud scenes, our method can track the target well. Both P2B and SC3D failed to track.

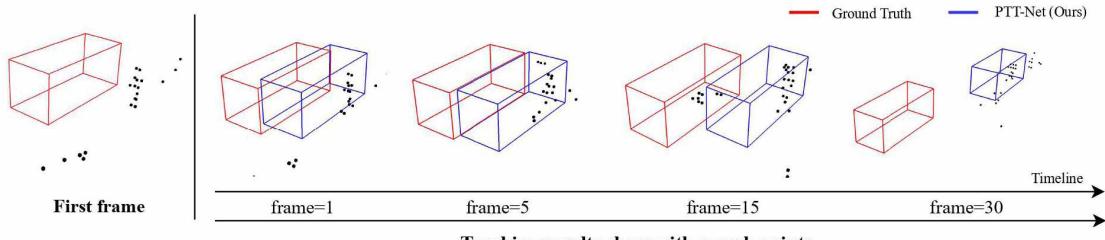


Fig. 8. **Failure cases.** no points in the initial search area

- [4] U. Kart, A. Lukei, M. Kristan, J. Kämäriäinen, and J. Matas, "How to make an rgbd tracker?", *Proc. European Conference on Computer Vision (ECCV)*, pp. 148–161, 01 2018.
- [5] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3569–3577, 2018.
- [6] M. Simon, K. Amende, A. Kraus, J. Honer, T. Sämann, H. Kaulbersch, S. Milz, and H. Groß, "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds," *2019 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1190–1199, 2019.
- [7] R. Q. Charles, S. Hao, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017.
- [8] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *European Conference on Computer Vision (ECCV)*, pp. 850–865, 2016.
- [9] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8971–8980, 2018.
- [10] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3d siamese tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1359–1368, 2019.
- [11] H. Qi, C. Feng, Z. Cao, F. Zhao, and Y. Xiao, "P2b: Point-to-box network for 3d object tracking in point clouds," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6328–6337, 2020.
- [12] 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 Journal*, vol. 21, no. 4, pp. 4995–5011, 2021.
- [13] S. Shi, X. Wang, and H. Li, "Pointrnn: 3d object proposal generation and detection from point cloud," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–779, 2019.
- [14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.
- [15] H. Hu, Z. Zhang, Z. Xie, and S. Lin, "Local relation networks for image recognition," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3463–3472, 2019.
- [16] P. Ramachandran, I. Bello, N. Parmar, A. Levskaya, A. Vaswani, and J. Shlens, "Stand-alone self-attention in vision models," in *Advances in Neural Information Processing Systems*, 2019.
- [17] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10073–10082, 2020.
- [18] A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, 2012.
- [19] J. Zarzar, S. Giancola, and B. Ghanem, "Efficient tracking proposals using 2d-3d siamese networks on lidar," *arXiv preprint arXiv:1903.10168*, 2019.
- [20] Y. Cui, Z. Fang, and S. Zhou, "Point Siamese Network for Person Tracking Using 3D Point Clouds," *Sensors*, vol. 20, p. 143, dec 2019.
- [21] H. Zou, J. Cui, X. Kong, C. Zhang, Y. Liu, F. Wen, and W. Li, "F-siamese tracker: A frustum-based double siamese network for 3d single object tracking," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8133–8139, 2020.
- [22] H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun, "Point transformer," *arXiv preprint arXiv:2012.09164*, 2020.
- [23] V. B. Nico Engel and K. Dietmayer, "Point transformer," *arXiv preprint arXiv:2011.00931*, 2020.
- [24] M. H. Guo, J. X. Cai, Z. N. Liu, T. J. Mu, R. R. Martin, and S. M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, vol. 7, no. 2, pp. 187–199, 2021.
- [25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, pp. 5099–5108, 2017.
- [26] M. Kristian, J. Matas, A. Leonardis, and F. Porikli, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pp. 2137–2155, 2016.
- [27] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2018.