Exploiting More Information in Sparse Point Cloud for 3D Single Object Tracking

Yubo Cui, Jiayao Shan, Zuoxu Gu, Zhiheng Li, and Zheng Fang, *Member, IEEE*

***Abstract*—3D single object tracking is a key task in 3D computer vision. However, the sparsity of point clouds makes it difficult to compute the similarity and locate the object, posing big challenges to the 3D tracker. Previous works tried to solve the problem and improved the tracking performance in some common scenarios, but they usually failed in some extreme sparse scenarios, such as for tracking objects at long distances or partially occluded. To address the above problems, in this letter, we propose a sparse-to-dense and transformer-based framework for 3D single object tracking. First, we transform the 3D sparse points into 3D pillars and then compress them into 2D bird's eye view (BEV) features to have a dense representation. Then, we propose an attention-based encoder to achieve global similarity computation between template and search branches, which could alleviate the influence of sparsity. Meanwhile, the encoder applies the attention on multi-scale features to compensate for the lack of information caused by the sparsity of point cloud and the single scale of features. Finally, we use set-prediction to track the object through a two-stage decoder which also utilizes attention. Extensive experiments show that our method achieves very promising results on the KITTI and NuScenes datasets.**

***Index Terms*—Point cloud, 3D object tracking, deep learning.**

I. Introduction

GIVEN the initial target object in the first frame, 3D single object tracking aims to estimate the 3D state of the target object in subsequent frames. Therefore, 3D single object tracking has a wide range of applications, such as autonomous driving and robotics. Meanwhile, with the development of 3D computer vision [1], [2], [3], [4], [5], [6], 3D single object tracking with point clouds received increasing attention. Similar to visual tracking, most point-cloud-based 3D single object tracking methods [7], [8], [9], [10] also adopt the Siamese pipeline, that is, by cropping the previous and current points based on the previous predicted box to get the template and search point clouds, and then inputting template and search point clouds to predict the state of the object based on their similarity. However, compared to dense images, point clouds are usually sparse, which is not only unfavorable for similarity computation, but also for target localization for 3D single object tracking.

To address the sparsity problem, previous works usually focus on enhancing the feature representation. For example, BAT [9] and V2B [11] enhance the features with box information and shape information respectively, to improve the robustness against sparseness. However, since they both adopt a point-based pipeline, they usually need to randomly downsample the points to a fixed number, such as 1024 for the search point cloud and 512 for the template point cloud, to pass through the network, which may lose the geometric information. LTTR [12] takes voxel-based pipeline and transformer to enhance the features. Nevertheless, they only consider the top-level extracted features but ignore the others, leading to information loss. Meanwhile, their transformer finally outputs region-level features, which are too coarse to address the sparsity problem of point cloud.

Therefore, we believe that a good 3D tracker should have following abilities: *First*, the input representation should retain as much original information as possible. *Second*, because of the sparsity of points, the fusion module should utilize as many features from backbone as possible to have a better similarity. *Third*, the similarity computation should fully exploit the correlation information to have a better measurement.

Based on the above analyses, in this letter, we propose a sparse-to-dense and transformer-based 3D tracking framework, which is named SMAT (**S**parse-to-dense and **M**ulti-scale **A**ttention **T**racker) and shown in Fig. 1. Specially, we first transform the input points to pillars and apply a 2D backbone to extract features. This transformation could avoid information loss due to downsampling and keep the geometric structure information of the original points. Meanwhile, the extracted feature could have a dense 2D representation, which could alleviate the sparsity problem of the point cloud. We then propose a multi-scale attention-based encoder to fully exploit the

Manuscript received 25 May 2022; accepted 12 September 2022. Date of publication 22 September 2022; date of current version 30 September 2022. This letter was recommended for publication by Associate Editor G. Costante and Editor E. Marchand upon evaluation of the reviewers' comments. 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 N2226001, and in part by 111 Project under Grant B16009. *(Corresponding author: Zheng Fang.)*

Yubo Cui is with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China, also with the Science and Technology on Near-Surface Detection Laboratory Wuxi 214000, China, and also with the National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Shenyang 110819, China (e-mail: ybcui21@stumail.neu.edu.cn).

Jiayao Shan, Zuoxu Gu, and Zhiheng Li are with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: shanjiayao97@gmail.com; guzuoxu@gmail.com; zhihengli@stumail.neu.edu.cn).

Zheng Fang is with the Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China, also with the National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Shenyang 110819, China, and also with the The Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education Shenyang 110819, China (e-mail: fangzheng@mail.neu.edu.cn).

Code is available at https://github.com/3bobo/smat.

Digital Object Identifier 10.1109/LRA.2022.3208687

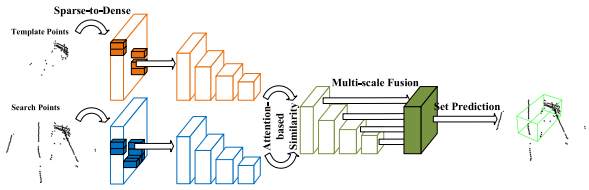

Fig. 1. The overall architecture of our proposed SMAT. We first transform the sparse point cloud to dense BEV features, and use a shared 2D backbone to extract 2D dense features. We then propose an encoder MAE to compute the two branches similarity at each scale and fuse the multi-scale features to fully exploit the information. Finally, we predict the box of object by set-prediction.

correlation information and compute the similarity. Different from previous works that usually ignore the shallow features, we retain all scale features to compute the similarity and then fuse them together. Thus, the similarity could include texture-rich information from shallow features and semantic-rich information from deep features. Meanwhile, the encoder uses attention to compute the similarity, which could have more consideration of global correlation information. The multi-head mechanism and global dependence modeling ensure the similarity could have a more comprehensive measurement. Moreover, for a simple but efficient framework, we adopt an attention-based two-stage decoder to track object by set-prediction. Comprehensive evaluation results show that our SMAT achieves the state-of-the-art results on KITTI [13] and NuScenes [14] datasets. Overall, our contributions are as follows:

- We propose a sparse-to-dense and transformer-based framework, which adopts an encoder-decoder paradigm, to handle the sparsity challenge in 3D single object tracking.
- We propose a novel encoder to replace the fusion module in the previous similarity-based tracker [7], [8], [9], which utilizes the attention at multi-scale features to fully exploit the information from the original sparse point cloud.
- Our method achieves promising performance on KITTI and NuScenes datasets. Extensive ablation studies also verify the effectiveness of our improvements.

The rest of this letter is organized as follows. Section II reviews related work on object tracking and transformer. Section III described the overall algorithm framework and details of the network. The experimental results of the proposed method on KITTI and NuScenes datasets are shown in Section IV. Finally, Section V concludes the letter.

II. Related Work

A. 2D Siamese Tracking

Recently, the Siamese-like networks [15], [16], [17], [18], [19], [20], [21], [22], [23] have been widely used in visual object tracking. The Siamese-like networks usually have two branches for template and search, and extract their features with a shared backbone at first. Then, they fuse the features from two branches by computing their similarity, such as cross-correlation, and use the fused features to regress boxes. However, because of the difference between point clouds and images, these methods are inapplicable to 3D object tracking with point clouds.

B. 3D Single Object Tracking

SC3D [7] is the pioneering work in 3D single object tracking with point clouds. They generate a set of candidate point clouds by Kalman Filter and select the tracked one based on a cosine similarity score. However, SC3D only takes a one-dimensional feature to compute the similarity, thus losing much local information, especially for sparse point cloud. Meanwhile, the KF also makes it could not be trained end-to-end. P2B [8] proposes a feature augmentation to augment the point-wise cosine similarity with target cues and takes VoteNet [24] to regress the box. Lately, based on P2B [8], PTT [10] utilizes the transformer to enhance the fused feature of sparse point cloud, BAT [9] exploits the 3D box information by introducing the box cues into comparison to have an accurate similarity compassion for sparse point cloud. However, their random downsampling for the input point cloud may lose the geometric information and makes it hard to output high-quality proposals in sparse scenarios. V2B [11] introduces the shape information into features and converts the augmented feature to a dense feature map by voxelization and max-pooling to have dense predictions. Nevertheless, they just convert the feature but ignore the input, thus they also suffer from information loss due to the random downsampling. Additionally, LTTR [12] adopts a voxel-based pipeline to avoid sampling and uses a region-level transformer to enhance the points features. Nonetheless, the region-level enhancement is also too coarse to alleviate the sparsity of the point cloud. Similar as PTT [10], PTTR [25] also utilizes self-attention to enhance the point features, they further uses cross-attention to compute the similarity between template and search points. Recently, M^2-Tracker [26] introduces a motion-based paradigm to track the 3D object, they predicts the relative target motion rather than computing the similarity and achieve the state-of-the-art performance.

C. Visual Transformer

Transformer [27] is first proposed in natural language processing. With the help of the attention mechanism, the transformer shows a strong ability in modeling the global dependencies of input. Therefore, transformer has also become popular in many tasks of computer vision. ViT [28] applies a pure transformer architecture in image classification. They split the image into many patches and input them into an encoder to classify. Swin [29] proposes a shifted window-based attention and a pure hierarchical backbone. PVT [30] designs a progressive shrinking pyramid and spatial-reduction attention to build a pure transformer backbone. They further update PVT to PVTv2 [31] by overlapping patch embedding and convolutional feed-forward networks. Additionally, LocalViT [32] introduces convolution into the transformer architecture to improve the local relation modeling. ConvSteam [33] finds that convolutional stem could help optimization stability and improve performance for vision transformer. In the dense prediction, DETR [34] first applies the transformer to object detection and deals with the detection problem as a set of predictions to match object queries based on the attention module. Deformable-DETR [35] further proposes multi-scale deformable attention module to speed up the

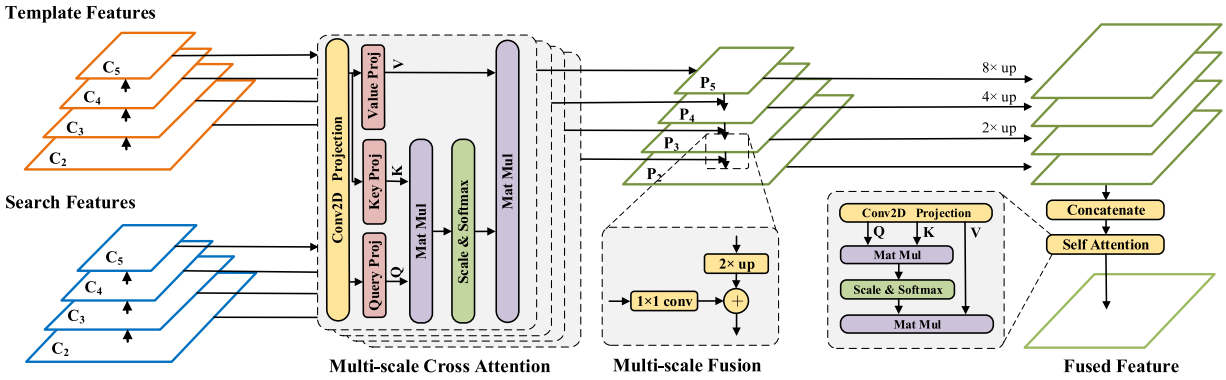

Fig. 2. Illustration of the proposed MAE. Given two sets of multi-scale features from template and search respectively, MAE first computes similarity by attention at each feature scale, and then fuses the multi-scale similarity feature to exploit the information of point cloud.

convergence of DETR and achieve better performance. Meanwhile, transformer has also been introduced into multi-object tracking [36], [37], [38] and segmentation [39], [40], [41].

III. Methodology

A. Overall Architecture

In the 3D scene, the object box could be represented as $(x, y, z, w, l, h, \theta)$, where (x, y, z) is the center, (w, l, h) is the size and θ is the orientation of the box respectively. For the tracking problem, our goal is to localize the target object with 3D point cloud frame by frame. Meanwhile, following the assumption [8] that the size of the target object is known through the first frame, we only need to estimate (x, y, z, θ).

In this letter, we aim at solving the sparsity problem in the point representation, similarity computation and feature utilization, thus we propose a sparse-to-dense transformer-based framework consisting of three components: sparse-to-dense feature encoding, multi-scale attention-based encoder and two-stage decoder. We will introduce each module in the following subsections.

B. Sparse-to-Dense Feature Encoding

To have a dense representation from sparse points, following [4], we project the points in an area with the size of $W \times L \times H$ from both branches into 3D pillars and apply a simplified PointNet [1] and maxpooling to generate dense BEV features. Following a shared 2D backbone, we extract 2D dense features from the BEV features. Specially, we use an attention-based backbone PVTv2 [31] to better capture the information of the BEV features. By this way, we convert the 3D sparse point clouds into dense 3D pillars and further obtain dense and compact 2D features.

C. Multi-Scale Attention-Based Encoder

After obtaining the two branch features C^s, C^t, where C^s, C^t represent the search and template feature respectively, we aim at fully exploiting the information of them to better capture the global similarity. Therefore, we propose the **M**ulti-scale **A**ttention-based **E**ncoder (MAE), as shown in Fig. 2.

Similarity Computation. Different from previous works using geometric similarity [8], [9], [10], [11] or cross-correlation [12], we use the multi-head attention mechanism to fuse features from two branches for its better global dependence modeling. The attention is proposed in [27], which projects the input features into Q, K, V embeddings to fuse them together based on their similarity. Specially, we first project the feature C^s, C^t to E^s, E^t by a shared Conv2D layer. Then, we use an attention-based block to fuse the template feature E^t and search feature E^s. Specially, the attention function is formulated as:

$$\mathrm{Attention(Q, K, V)} = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{1}$$

where Q, K, V are the query, key and value embedding respectively and d is the feature dimension of the K. Meanwhile, the attention-based block [27] consists of a multi-head attention (MHA) and a feedforward-network (FFN), thus it could be formulated as:

$$\mathrm{MHA(Q, K, V)} = \mathrm{Concat}(H_1, \ldots, H_h)W^o \tag{2}$$

$$\mathrm{FFN(X)} = \mathrm{Max}(0, W_1X + b_1)W_2 + b_2 \tag{3}$$

where H_j is computed by (1), representing attention function for j-th head and W^o is the head linear projection, h is the head number. W_1, W_2 and b_1, b_2 are weight matrices and basis respectively. X is the output of the MHA thus the FFN is after the MHA. Specially, we use a cross-attention block for different inputs to the attention. We project the search feature E^s to query embedding Q and project the template feature E^t to key embedding K and value embedding V as:

$$Q = E^sW^q, K = E^tW^k, V = E^tW^v \tag{4}$$

where W^q, W^k, W^v are the linear projection of query, key and value respectively. Through the cross-attention block, we obtain the fused similarity feature P.

Multi-scale Features Fusion. For multi-scale two-branch features fusion, there are two scale fusion strategies: early fusion and late fusion. The early fusion first fuses the multi-scale extracted features for each branch and then computes the similarity between two branches and fuses them, while the late fusion first computes two-branch features similarity and fuses them

together at each scale and then fuses the multi-scale similarity features. Compared to the early fusion, the late fusion could exploit multi-scale feature similarity. For example, the top fusion could explore more semantic similarity while the bottom fusion could explore more texture similarity. Therefore, we adopt the late fusion and will compare the two strategies later.

For each scale features C_i^s and C_i^t from the backbone with downsample rates of 2^i, where $i \in \{2, 3, 4, 5\}$, we first use the above similarity computation to obtain the fused feature P_i at each scale. Then, to enhance the bottom features, we use the top-down path and lateral connections following FPN [42] to propagate the information from the top feature to the bottom feature, which could be formulated as:

$$P_{i-1} = \mathrm{Conv}_{3\times3}\{\mathrm{Conv}_{1\times1}(P_{i-1}) + \mathrm{Upsample}(P_i)\}, i \in \{3, 4, 5\} \tag{5}$$

Then, to fuse the multi-scale features, we upsample P_3, P_4, P_5 to the size of P_2 and concatenate them together, and apply a Conv2D layer with 1×1 kernel to fuse the concatenated features to generate scale-fused feature U. Finally, a self-attention block is followed to update the fused feature U. The process could be formulated as:

$$\hat{P}_i = \mathrm{Upsample}(P_i), i \in \{3, 4, 5\} \tag{6}$$

$$U = \mathrm{Conv}_{1\times1}\{\mathrm{Concate}(\hat{P}_i)\}, \forall i \tag{7}$$

$$\bar{U} = \mathrm{FFN}(\mathrm{MHA}(U, U, U)) \tag{8}$$

Therefore, given two sets of multi-scale features representing template and search features respectively, the proposed MAE could output a fused feature. Benefiting from the MHA, the fused feature has a global similarity measurement in different feature spaces. Meanwhile, the multi-scale fusion also makes the fused feature have a cross-scale perception, exploiting more information of sparse point cloud.

D. Two-Stage Decoder

Inspired by DETR [34] and Deformable-DETR [35], we also adopt a two-stage decoder to predict the box with the fused multi-scale features $\bar{U}$. First, we use the fused feature to generates box pairs $\{b_p, c_p\}$ by two linear layers in regression and classification branches, thus b_p and c_p is the predicted boxes and scores in the first stage respectively. Second, we select top k output pairs based on their scores and project the selected box value (x, y, z) to an embedding feature and concatenate them with their corresponding feature in the feature $\bar{U}$. The concatenated features are further projected by a linear layer to generate a set of target queries T, where each query represents the feature embedding of one potential box. The process can be represented as:

$$c_p = \mathrm{Linear}_1(\bar{U}) \tag{9}$$

$$b_p = \mathrm{Linear}_2(\bar{U}) \tag{10}$$

$$\hat{b} = \{b_i \| c_i \in \mathrm{TopK}(c_p)\} \tag{11}$$

$$\hat{U} = \{\bar{U}_i \| c_i \in \mathrm{TopK}(c_p)\} \tag{12}$$

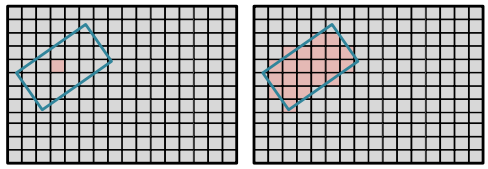

Fig. 3. (a) The previous label assign; (b) Our augmented label assign. Our assign strategy could have more positive samples.

$$T = \mathrm{Linear}_3(\mathrm{Concate}(\hat{U}, \mathrm{Proj}(\hat{b}))) \tag{13}$$

where $Proj$ means the concatenation of cosine and sine representation of the proposals. Meanwhile, the target queries could be created and initialized randomly for a one-stage prediction. Third, the generated target queries T are fed into a cross-attention block together with $\hat{U}$ as:

$$Q = TW^q, K = \hat{U}W^k, V = \hat{U}W^v, \tag{14}$$

Through the cross-attention block, the feature $\hat{T}$ is generated for prediction. Following two parallel linear layers to predict the box and score respectively, the decoder outputs a set of $\{b, c\}_{i=1}^k$, where $b \in \{x, y, z, \sin\theta, \cos\theta\}$, and we select the box with the highest score to track.

Different from Deformable-DETR which uses deformable attention and multi-scale features to predict, we only use single scale feature $\bar{U}$ to predict since it already has included multi-scale information in our encoder. Meanwhile, considering the sparsity of 3D points, it would be better to keep more reference points in attention, thus we use the vanilla attention rather than deformable attention.

E. Training

Through the proposed SMAT, we get a set of predictions $y = \{b, c\}_{i=1}^k$. Different from detection which has many labels in one batch data and considers one object as one label of data, there is only one label for every batch data in single object tracking. Therefore, suffering from the insufficient ground-truth label data, the convergence could be slow in training and the performance may be affected.

To solve this problem, we augment the label data based on the foreground pixels. Specially, we downsample the input point cloud into the same scale with C_2 and count the number of foreground pixels N_{fg} in the downsampled BEV point cloud to generate the ground truth set $\tilde{y} = \{\tilde{b}, 1\}_{i=1}^{N_{fg}}$. The augmentation can be interpreted as every foreground pixel is treated as the object. Fig. 3 shows the difference between previous label assign and our assign. Additionally, we follow the set prediction loss of [34], [35] to train the model. The matching cost is defined as follows:

$$\mathscr{L} = \lambda_{cls}\mathscr{L}_{cls} + \lambda_{L1}\mathscr{L}_{L1} \tag{15}$$

where $\mathscr{L}_{cls}$ is cross-entropy loss between the predicted and ground-truth classes, $\mathscr{L}_{L1}$ is L1 loss between the states of predicted and ground-truth boxes. λ_{cls} and λ_{L1} are weights for the two losses respectively.

TABLE I
PERFORMANCE COMPARISON ON THE NUSCENES DATASET. THE BEST TWO RESULTS ARE HIGHLIGHTED IN RED, BLUE

Method	Paradigm	Car-64159		Pedestrian-33227		Truck-13587		Trailer-3352		Bus-2953		Mean-117278	
		Success	Precision	Success	Precision	Success	Precision	Success	Precision	Success	Precision	Success	Precision
M^2-Tracker [26]	Motion	55.85	65.09	32.10	60.92	57.36	59.54	57.61	58.26	51.39	51.44	49.23	62.73
SC3D [7]	Similarity	22.31	21.93	11.29	12.65	30.67	27.73	35.28	28.12	29.35	24.08	20.70	20.20
P2B [8]		38.81	43.18	28.39	52.24	42.95	41.59	48.96	40.05	32.95	27.41	36.48	45.08
BAT [9]		40.73	43.29	28.83	53.32	45.34	42.58	52.59	44.89	35.44	28.01	38.10	45.71
SMAT (Ours)		43.51	49.04	32.27	60.28	44.78	44.69	37.45	34.10	39.42	34.32	40.20	50.92

TABLE II
PERFORMANCE COMPARISON ON THE KITTI DATASET. THE BEST TWO RESULTS ARE HIGHLIGHTED IN RED, BLUE

Method	Paradigm	Car-6424		Pedestrian-6088		Van-1248		Cyclist-308		Mean-14068	
		Success	Precision	Success	Precision	Success	Precision	Success	Precision	Success	Precision
M^2-Tracker [26]	Motion	65.5	80.8	61.5	88.2	53.8	70.7	73.2	93.5	62.9	83.4
SC3D [7]	Similarity	41.3	57.9	18.2	37.8	40.4	47.0	41.5	70.4	31.2	48.5
SC3D-RPN [43]		36.3	51.0	17.9	37.8	-	-	43.2	81.2	-	-
P2B [8]		56.2	72.8	28.7	49.6	40.8	48.4	32.1	44.7	42.4	60.0
PTT [10]		67.8	81.8	44.9	72.0	43.6	52.5	37.2	47.3	55.1	74.2
BAT [9]		60.5	77.7	42.1	70.1	52.4	67.0	33.7	45.4	51.2	72.8
LTTR [12]		65.0	77.1	33.2	56.8	35.8	45.6	66.2	89.9	48.7	65.8
V2B [11]		70.5	81.3	48.3	73.5	50.1	58.0	40.8	49.7	58.4	75.2
PTTR [25]		65.2	77.4	50.9	81.6	52.5	61.8	65.1	90.5	57.9	78.1
SMAT (Ours)		71.9	82.4	52.1	81.5	41.4	53.2	61.2	87.3	60.4	79.5

IV. EXPERIMENTS

In this section, we evaluate the proposed SMAT on NuScenes [14] and KITTI [13] datasets. We first introduce the experimental setting and then compare our method with previous state-of-the-art methods on the two datasets. Finally, we conduct extensive ablation studies to investigate each component of SMAT to validate our improvement.

A. Experimental Setting

NuScenes Dataset. The NuScenes dataset has a total of 1,000 scenes, contains about 300,000 points every frame and has 360-degree view annotations. Meanwhile, it has been officially divided into training, validation and testing scenes. However, since it does not directly support single object tracking task, we follow the dataset setting of BAT [9] to train and test our method, where we use the training set for training and the validation set for testing. We refer to the published results in [9] for comparison.

KITTI Dataset. We use the training sequences of KITTI tracking dataset which contains 21 sequences. We follow the same setting as [8] to divide the sequences into training, validation, and testing splits, where 0–16 for training, 17–18 for validation and 19-20 for testing.

Implementation Details. We use PVTv2-b2 [31] as our backbone and follow their original settings. Specially, the head number is set to [1, 2, 5, 8], the depth is set to [3, 4, 6, 3], the expansion ratio of the feed-forward layer is set to [8, 8, 4, 8] and the channel number is set to [64, 128, 320, 512]. In the Cross-FPN, we set the output channel and feed-forward channel both to 256 and set the cross-attention heads and attention layers to 8. In the decoder, we set the head number and layer number to 8 and the feed-forward channel to 2048. We set $[0.1, 0.1, 4]$m as pillar size and $[-3.2, -3.2, -3, 3.2, 3.2, 1]$m as search area for Car.

Training Details. For KITTI dataset, we train the SMAT with 72 epochs with Adamw [44] optimizer with the initial learning rate of 0.0001, weight decay of 0.05 and batch size of 16 on NVIDIA 3090 GPU. The learning rate decayed by $10\times$ at epochs 63 and 69. For NuScenes dataset, the epochs are 36 with the learning rate decayed by $10\times$ at epochs 27 and 33 and batch size of 32. The other settings are the same as those for KITTI.

Evaluation metric. We use the One Pass Evaluation (OPE) [45] to measure Success and Precision. The Success measures the 3D IoU between the predicted box and the ground-truth box, the Precision measures the AUC of distance between the center of two boxes from 0 to 2 m.

B. Comparison With State-of-The-Arts

Results on NuScenes. As shown in Table I, our SMAT achieves the second performance in NuScenes dataset. Specially, our SMAT lags behind M^2-Tracker by 9.03% in Success and 11.81% in Precision respectively, and outperforms the third BAT [9] by 2.10% in Success and 5.21% in Precision. Compared to our SMAT and other methods, M^2-Tracker takes a different motion-based paradigm to track the target, which shows better results than similarity-based paradigm. Meanwhile, our SMAT achieves the best results among the similarity-based methods, showing that our MAE has a better similarity fusion. Moreover, for the truck class, we have a better prediction on center than BAT [9] but worse prediction on orientation, thus we have achieve better in precision but worse in success. Additionally, we notice that our trailer's performance is much lower than other methods. We believe that because of the long and thin shape of the trailer, our method needs a larger search area to input compared to the other methods, thus includes more noise.

Results on KITTI. As shown in Table II, SMAT also performs the second on the mean of four categories. Meanwhile, our method achieves the best performance of 71.9% and 52.1% in Car and Pedestrian categories. Moreover, SMAT surpasses

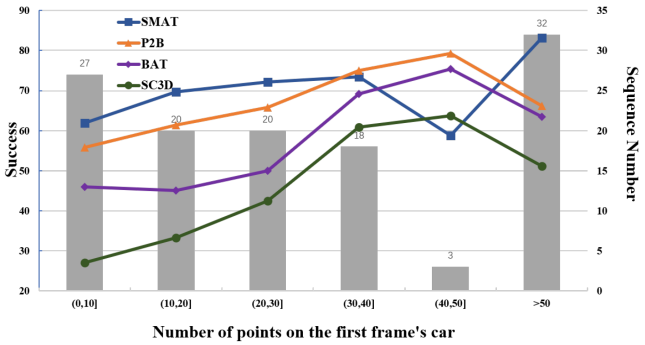

Fig. 4. The influence of the number of points on the first frame's car.

TABLE III
EFFECTS OF MULTI-SCALE FEATURE FUSION IN THE ENCODER

Multi-Scale	Early Fusion	Late Fusion	Success	Precision
C_2			59.2	69.3
C_5			61.7	74.3
√	√		63.8	75.7
√		√	**65.2**	**76.2**

V2B [11], which also have a dense prediction, in the Car, Pedestrian and Cyclist categories. The comparison verifies the validity of our sparse-to-dense transformation, showing a dense representation could retain more information from 3D point clouds for single object tracking. Additionally, compared to LTTR [12] which also adopts a voxel pipeline and transformer, our method exceeds by a large margin (11.7% in Mean), showing the efficiency of the proposed encoder MAE. Meanwhile, our SMAT achieves 17.6 FPS in the inference phase.

Robustness to Sparsity. To further explore the effectiveness of our method, especially for the sparse point cloud, we classify the car tracking sequences in KITTI according to the number of point clouds in the first frame and then evaluate our method on these sequences of different intervals. As shown in Fig. 4, SMAT shows better robustness to sparsity, especially for the targets holding less than 30 points. The figure verifies our improvement in the three aspects. Moreover, only three tracking sequences are in the interval of (40,50], thus we believe that the performance drop in this interval is mainly because of the insufficient samples. Additionally, we also visualize the tracking results in Fig. 5.

C. Ablation Study

In this section, we conduct comprehensive experiments to validate the design of SMAT. All experiments are conducted on the Car category of the KITTI dataset.

Multi-scale Fusion in MAE. We first analyze the influence of multi-scale feature fusion in our framework. Here we first introduce a baseline network that directly computes single-scale similarity. Meanwhile, the baseline network directly regresses one target box without set-prediction, and we term this regression manner as "Direct". Notice, for the experiment in Table III, all compared networks use the "Direct" regression. We believe this experimental setting can more clearly show the effectiveness of our method.

TABLE IV
COMPARISON OF DIFFERENT COMPUTATIONS IN THE ENCODER

Prediction	Similarity	Success	Precision
Direct	Cosine	62.1	73.2
	Euclidean	64.6	75.0
	Cross-correlation	64.1	75.6
	Ours	**65.2**	**76.2**
Set	Cosine	66.5	76.5
	Euclidean	65.3	75.4
	Cross-correlation	66.4	77.4
	Ours	**71.9**	**82.4**

We compared C_2, C_5, and multi-scale feature fusion with early and late fusion strategies, where C_2 and C_5 are bottom and top features which have downsample rate 2^2 and 2^5 respectively. As shown in Table III, compared to using C_2, using C_5 performs better by 2.5% and 5.0% gains in Success and Precision respectively. Meanwhile, the early multi-scale fusion further improves the performance, surpassing the network with fusing C_5 by +2.1% in Success and +1.4% in Precision. The results show that compared to use single-scale features, fusing multi-scale features could bring better tracking performance in our framework. Moreover, the late fusion achieves 65.2% and 76.2% in Success and Precision respectively, higher than early fusion by 1.4% and 0.5%. Compared to early fusion, the multi-scale similarity in late fusion may have a stronger correlation because they only need to focus on the information at their own scale and then fuse the information. In contrast, the early fusion has fused the multi-scale features before the similarity computation, thus the similarity feature only has one scale and may lose some correlation information.

Similarity Computation in MAE. We also compare different similarity computation methods in MAE with two prediction manners. We compare the commonly used cosine similarity, Euclidean distance, cross-correlation, and our attention-based computation. Specially, we replace the attention block in MAE with the compared three similarity methods. Meanwhile, the three similarity methods usually need a feature augmentation module in previous works [8], [9], [12], thus we further multiply the three similarity maps with the search feature to serve as a simple augmentation. As shown in Table IV, our attention-based method achieves the best performance in both prediction manners, surpassing the second by 0.6% and 5.4% in Success of the two prediction manners respectively. The results verify our view that a global similarity computation could have better correlation information for sparse point cloud. Specially, the geometric similarity only considers one space, such as the feature angle in cosine similarity or the feature magnitude in Euclidean distance. Therefore, the similarity is not sufficient and could be considered as a local measurement in feature space. Meanwhile, we improve the performance with 1.1% and 5.5% in the two prediction manners by replacing the cross-correlation with the attention-based computation. We believe that the cross-correlation is a spatial local linear matching operation, leading to information loss, especially for the sparse point cloud. Differently, the attention-based computation explores the global similarity spatially in different feature spaces. Therefore, it fully

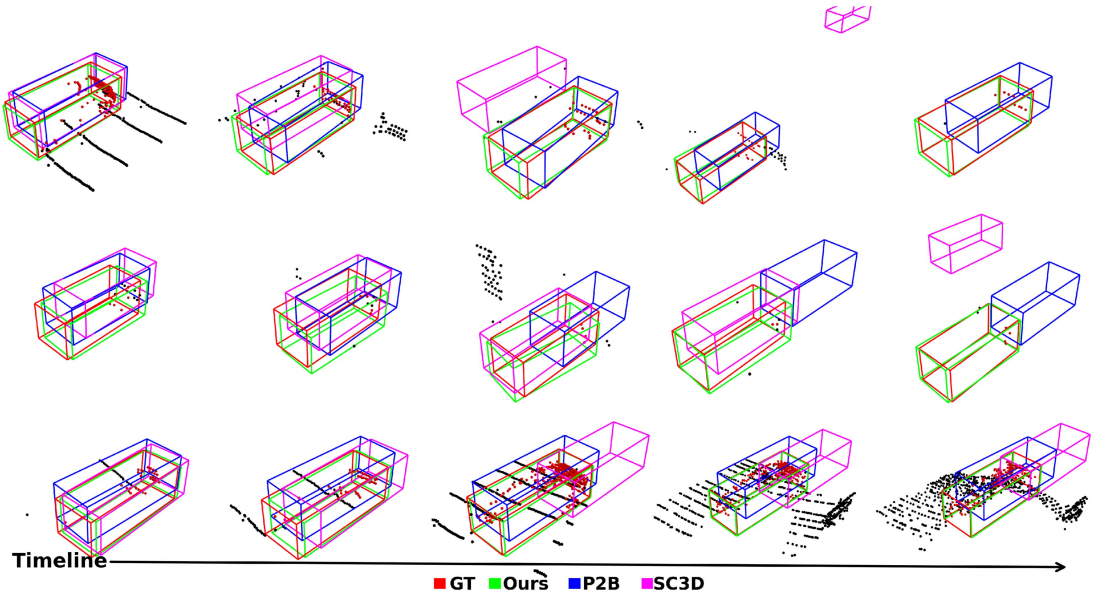

Fig. 5. Advantageous cases of our SMAT compared with P2B, SC3D on the Car category of KITTI Dataset. The top two rows are sparse while the bottom one is denser.

TABLE V
EFFECTS OF DIFFERENT COMPONENTS IN THE DECODER

Set-Prediction	Augmentation	Two-Stage	Success	Precision
			65.2	76.2
		√	66.4	78.4
√			53.2	65.5
√		√	67.0	79.1
√	√		68.8	81.4
√	√	√	**71.9**	**82.4**

TABLE VI
COMPARISON WITH THE DECODER OF DEFORMABLE-DETR

Method	Params	FLOPs	Success	Precision
DDETR [35]	**9.4M**	31.7G	70.9	81.4
Ours	10.1M	**28.2G**	**71.9**	**82.4**

TABLE VII
DIFFERENT TEMPLATE GENERATIONS. "F," "P" AND "AP" DENOTES THE FIRST GROUND-TRUTH, THE PREVIOUS RESULTS AND ALL PREVIOUS RESULTS RESPECTIVELY. THE DEFAULT SETTING IS "F&P"

	Method	Source of template			
		F	P	F&P	AP
Success	SC3D [7]	31.6	25.7	34.9	41.3
	P2B [8]	46.7	53.1	56.2	51.4
	BAT [9]	51.8	59.2	60.5	55.8
	PTT [10]	62.9	64.9	67.8	59.8
	V2B [11]	67.8	**70.0**	70.5	69.8
	SMAT (Ours)	**68.1**	66.7	**71.9**	**69.9**
Precision	SC3D [7]	44.4	35.1	49.8	57.9
	P2B [8]	59.7	68.9	72.8	66.8
	BAT [9]	65.5	75.6	77.7	71.4
	PTT [10]	76.5	77.5	81.8	74.5
	V2B [11]	**79.3**	**81.3**	81.3	**81.2**
	SMAT (Ours)	77.5	76.1	**82.4**	79.8

exploits the correlation information to have a better similarity measurement and bring better tracking performance.

Decoder. Additionally, we compare the direct regression with set-prediction decoder, analyze the effect of label augmentation and two-stage prediction, and further compare the two-stage decoder with the decoder of Deformable-DETR [35]. As shown in Table V, the performance drops significantly (↓12.0% in Success and ↓10.7% in Precision) if we directly adopt one-stage set prediction. We believe that because the pure one-stage set prediction only outputs k pair boxes and is trained with only one ground-truth label, thus it is a sparse prediction which is hard to generate high-quality proposals for sparse point cloud. Meanwhile, the two-stage prediction improves the performance significantly to 67.0% and 79.1% in Success and Precision respectively. Additionally, when only using one-stage prediction, the proposed label augmentation also improves the performance to 68.8% and 81.4%, showing its simplicity and efficiency. Finally, by adopting the two-stage set-prediction decoder and label augmentation training together, the proposed network achieves the best performance. Furthermore, the performance of direct regression could be improved to 66.4% and 78.4% if we apply two-stage refinement, which is lower than our final result. We believe that direct regression lacks discrimination and forces all features to regress the target value no matter they are from foreground or background, thus limit the performance. Additionally, as shown in Table VI, compared to the decoder in Deformable-DETR [35], our decoder achieves better performance (↑1.0% in both Success and Precision). The results show that it is better to keep more reference points rather than reduce them, which is important for 3D object tracking with sparse point cloud.

Template Generation Strategy. We further compare our method with previous works under different template generation strategies on Car category of KITTI dataset. As shown in Table VII, our method achieves the best performance in three generation strategies including the first frame ground-truth. The results show that our method could achieve better results when given high confidence prior. Meanwhile, compared to most previous works, such as PTT and BAT, our SMAT has a smaller

gap 5.2% between different strategies, showing more robustness to the source of the template point cloud.

V. Conclusion

In this letter, we analyze the limitation of 3D object tracking with point cloud and present SMAT, a sparse-to-dense transformer-based framework. We also propose a multi-scale attention-based encoder MAE to fully exploit the information from the sparse point cloud. By solving the sparsity problem in the point representation, similarity computation and feature utilization, our method could achieve better performance. The comprehensive experiments demonstrate the effectiveness of our design of the proposed framework and encoder. For future work, we plan to explore more attention-based networks on 3D single object tracking.

References

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

[2] 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. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.

[3] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337.

[4] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12697–12705.

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

[6] S. Shi et al., "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10529–10538.

[7] 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.

[8] 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 Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6329–6338.

[9] 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.

[10] 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.

[11] L. Hui, L. Wang, M. Cheng, J. Xie, and J. Yang, "3D siamese voxel-to-bev tracker for sparse point clouds," in *Proc. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 28714–28727.

[12] Y. Cui, Z. Fang, J. Shan, Z. Gu, and S. Zhou, "3D object tracking with transformer," in *Proc. 32nd BMVC*, 2021, p. 317.

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

[14] H. Caesar et al., "Nuscenes: A multimodal dataset for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11621–11631.

[15] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2016, pp. 749–765.

[16] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2016, pp. 850–865.

[17] 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.

[18] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Atom: Accurate tracking by overlap maximization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4660–4669.

[19] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6182–6191.

[20] M. Danelljan, L. V. Gool, and R. Timofte, "Probabilistic regression for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7183–7192.

[21] Y. Xu, Z. Wang, Z. Li, Y. Ye, and G. Yu, "Siamfc++ : Towards robust and accurate visual tracking with target estimation guidelines," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 12549–12556.

[22] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8126–8135.

[23] F. Xie, C. Wang, G. Wang, W. Yang, and W. Zeng, "Learning tracking representations via dual-branch fully transformer networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 2688–2697.

[24] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9277–9286.

[25] C. Zhou et al., "PTTR: Relational 3D point cloud object tracking with transformer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8531–8540.

[26] C. Zheng et al., "Beyond 3D siamese tracking: A motion-centric paradigm for 3D single object tracking in point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8111–8120.

[27] A. Vaswani et al., "Attention is all you need," in *Proc. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 5998–6008.

[28] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. 9th Int. Conf. Learn. Representations*, 2021, pp. 1–21.

[29] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10012–10022.

[30] W. Wang et al., "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 568–578.

[31] W. Wang et al., "PVT v2: Improved baselines with pyramid vision transformer," *Comput. Vis. Media*, vol. 8, no. 3, pp. 415–424, 2022.

[32] Y. Li, K. Zhang, J. Cao, R. Timofte, and L. V. Gool, "LocalVit: Bringing locality to vision transformers," 2021, *arXiv:2104.05707*.

[33] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, and R. Girshick, "Early convolutions help transformers see better," in *Proc. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 30392–30400.

[34] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.

[35] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," in *Proc. Int. Conf. Learn. Representations*, 2021.

[36] P. Sun et al., "Transtrack: Multiple-object tracking with transformer," 2021, *arXiv:2012.15460.*

[37] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8844–8854.

[38] T. Zhu, M. Hiller, M. Ehsanpour, R. Ma, T. Drummond, and H. Rezatofighi, "Looking beyond two frames: End-to-end multi-object tracking using spatial and temporal transformers," 2021, *arXiv:2103.14829*.

[39] B. Dong, F. Zeng, T. Wang, X. Zhang, and Y. Wei, "SOLQ: Segmenting objects by learning queries," in *Proc. Neural Inf. Process. Syst.*, 2021, vol. 34, 2021, pp. 21898–21909.

[40] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," in *Proc. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 12077–12090.

[41] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 7262–7272.

[42] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944.

[43] J. Zarzar, S. Giancola, and B. Ghanem, "Efficient bird eye view proposals for 3D siamese tracking," 2019, *arXiv:1903.10168*.

[44] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[45] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2411–2418.