Exploring Point-BEV Fusion for 3D Point Cloud Object Tracking With Transformer

Zhipeng Luo ©, Changqing Zhou ©, Liang Pan ©, Gongjie Zhang ©, Tianrui Liu ©, Yueru Luo ©, Haiyu Zhao ©, Ziwei Liu ©, and Shijian Lu ©

Abstract—With the prevalent use of LiDAR sensors in autonomous driving, 3D point cloud object tracking has received increasing attention. In a point cloud sequence, 3D object tracking aims to predict the location and orientation of an object in consecutive frames. Motivated by the success of transformers, we propose Point Tracking TRansformer (PTTR), which efficiently predicts high-quality 3D tracking results in a coarse-to-fine manner with the help of transformer operations. PTTR consists of three novel designs. 1) Instead of random sampling, we design Relation-Aware Sampling to preserve relevant points to the given template during subsampling. 2) We propose a Point Relation Transformer for effective feature aggregation and feature matching between the template and search region. 3) Based on the coarse tracking results, we employ a novel Prediction Refinement Module to obtain the final refined prediction through local feature pooling. In addition, motivated by the favorable properties of the Bird’s-Eye View (BEV) of point clouds in capturing object motion, we further design a more advanced framework named PTTR++, which incorporates both the point-wise view and BEV representation to exploit their complementary effect in generating high-quality tracking results. PTTR++ substantially boosts the tracking performance on top of PTTR with low computational overhead. Extensive experiments on multiple datasets show that our proposed approaches achieve superior 3D tracking accuracy and efficiency.

Index Terms—3D object tracking, autonomous driving, computer vision, point cloud, vision transformer.

I. INTRODUCTION

With the rapid development of 3D sensors in the past decade, solving various vision problems [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] with point clouds has attracted increasing attention due to the huge potential in applications such as autonomous driving, motion planning, and robotics. As a long-standing research problem in computer vision, object tracking with point clouds has also drawn wide research interests. 3D object tracking aims to determine the object pose and position of the tracked object in consecutive frames in a point cloud sequence. However, 3D tracking still faces a number of open and challenging problems such as LiDAR point cloud sparsity, random shape incompleteness, texture feature absence, etc.

Existing 3D object tracking approaches can be largely categorized into two groups: multi-object tracking (MOT) and single-object tracking (SOT). MOT methods [11], [12], [13], [14] generally adopt a detect-to-track strategy by first detecting objects in each frame and then matching the detections across consecutive frames based on the estimated location or speed. In contrast, SOT methods only process a subset of the point cloud scene, which usually come with much lower computational costs and higher throughput. We study SOT in this work, and our objective is to estimate the location and orientation of a single object in subsequent frames given an object template in the first frame.

The pioneering deep-learning-based 3D SOT method SC3D [15] first generates a series of candidates given the last location of a specific object, and then makes predictions by selecting the best-matched candidate in the latent space. However, it is not end-to-end trainable and suffers from low inference speed as SC3D relies on a large number of candidates. Differently, P2B [16] explores not relying on many candidates by first using cosine similarity to fuse features of the search region with the template and then adopting the prediction head of VoteNet [17] to generate the final prediction. Following P2B, SA-P2B [18] adds an extra auxiliary network to predict the object structure. In a similar framework, 3D-SiamRPN [19] uses a cross-correlation module for feature matching and an RPN head for final prediction. These methods [15], [16], [18] essentially perform a linear matching process between features in the search domain and the template, which is typically not robust against various challenges in 3D observation, such as random noise, sparsity, and occlusions. Moreover, the inclusion of complex prediction heads as in detection models [17] severely limits their tracking speed, which is a crucial factor for real-time applications.

In this work, we design Point Tracking TRansformer (PTTR), a novel tracking paradigm that achieves high-quality 3D object tracking in a coarse-to-fine fashion. Specifically, PTTR first extracts point features from the template and the search region individually using the PointNet++ [2] backbone. To alleviate the point sparsity issue, which is common for objects at a distance, we propose a sampling strategy termed Relation-Aware Sampling, which can preserve more points that are relevant...
to the given template by leveraging the relation-aware feature similarities between the search region and the template. We then propose a novel Point Relation Transformer equipped with Relation Attention Module to match search and template features and generate a coarse prediction based on the matched feature. PRT first utilizes a self-attention operation to adaptively aggregate point features for the template and the search region individually, and then performs feature matching with a cross-attention operation. Moreover, we propose a lightweight Prediction Refinement Module to refine the coarse prediction with local feature pooling. We highlight that PTTR achieves good efficiency despite the prediction refinement process.

Other than the commonly used point-wise representation in 3D SOT, another type of point cloud representation that has been widely adopted in various 3D perception tasks [20], [21], [22] and achieves great success is the voxel-based representation. Specifically, the raw point clouds are rasterized into voxel cells, hence leading to a structured 3D format. One special form of the voxel representation is the Bird’s-Eye View (BEV), where the height dimension is suppressed and the point cloud is converted to 2D feature maps. The BEV representation was first utilized in deep-learning-based 3D perception by [23] for object detection and is known for its computation efficiency. From the inspection of point cloud tracklets, we find that BEV has a significant potential to benefit 3D tracking. As shown in Fig. 1(a), BEV could better capture motion features compared to the other two views (i.e., Side View and Front View) since object movements largely occur in the horizontal plane in scenarios such as autonomous driving. By compressing the 3D point clouds, BEV naturally filters out the noises in the height dimension, which makes it promising for identifying object motions in a tracking sequence. Nonetheless, voxelization and BEV representations inevitably cause information loss, which could lead to inaccurate localization, especially for smaller objects (e.g., pedestrians). Therefore, it is intuitive to explore the complementary effect of point-wise and BEV representations for the tracking task to exploit their merits.

In light of the above motivations, we further propose a 3D tracking framework named PTTR++ on top of PTTR to exploit the complementary effect between point features and BEV features. As shown in Fig. 1(b), PTTR++ consists of two parallel feature matching branches for the point-wise view and BEV, respectively, where the point branch is directly inherited from PTTR and each branch matches template and search region features with PRT independently. Subsequently, Cross-view Feature Mapping is performed to establish the mapping between point and BEV features. Finally, we propose Selective Feature Fusion to adaptively fuse the mapped point and BEV features for generating the tracking prediction. With the help of the proposed Point-BEV fusion strategy, PTTR++ substantially outperforms PTTR in terms of tracking accuracy with low computational overhead. While we mainly develop our method based on PTTR, we highlight that the proposed Point-BEV fusion is a generic approach that can be easily integrated with other 3D trackers. We experiment on another recently published method M²-Track [24], which is a contemporary work to our conference paper [25], to validate the wide applicability of our proposed fusion approach.

In summary, the contributions of this work are: 1) We design PTTR, a transformer-based 3D point cloud object tracking method, which performs tracking in a coarse-to-fine manner. PTTR consists of a few novel designs, including Relation-Aware Sampling for preserving template-relevant points, Point Relation Transformer for effective feature aggregation and matching, and a lightweight pooling-based refinement module. 2) We further propose PTTR++ on top of PTTR to explore the complementary effect of point-wise and BEV representations for improved tracking performance. PTTR++ employs Cross-view Feature Mapping and Selective Feature Fusion for effective Point-BEV fusion. 3) Comprehensive evaluations over multiple benchmarks show that our proposed methods achieve state-of-the-art performance with competitive inference speed.

This work is an extension of our conference paper [25] published at CVPR’22. Compared with the conference version, we incorporate the following new content. 1) We propose a more advanced tracking framework PTTR++ by incorporating Point-BEV fusion with PTTR to achieve state-of-the-art performance. 2) We show that the proposed fusion strategy is a

Fig. 1. (a) Bird’s-Eye View (BEV) densifies the point cloud and filters out the noises in the height dimension, while most object movements occur in the horizontal plane. Template points from the previous frame are overlaid with search region points from the current frame for illustration. (b) PTTR++ builds on top of PTTR by performing BEV feature matching in parallel with the existing point feature matching to exploit the complementary information of the two representations.
generic approach that can be applied to boost the performance of other tracking methods. 3) We provide a more comprehensive performance evaluation by reporting experimental results on multiple benchmarks and conducting extensive ablation studies.

II. RELATED WORK

3D Object Tracking: 3D object tracking can be roughly divided into two categories: multi-object tracking (MOT) and single-object tracking (SOT). Most MOT approaches adopt a detect-to-track strategy and mainly focus on data association [26], [27], [28] first proposes a 3D detection module to provide the 3D bounding boxes, then uses a 3D Kalman Filter to predict current estimation, and match them using the Hungarian algorithm. [12] proposes to use GNN to model relationships among different objects both spatially and temporally, while [14] applies closest distance matching after speed compensation. SOT methods focus on tracking a single object given a template. Most of the existing 3D SOT approaches utilize the Siamese network architecture to match the template and the search region. As the pioneering deep-learning-based work, SC3D [15] proposes to match feature distance between candidates and target and regularize the training using shape completion. P2B [16] matches search and template features with cosine similarity and employs Hough Voting [17] to predict the current location. SA-P2B [18] proposes to learn the object structure as an auxiliary task. 3D-SiamRPN [19] uses a RPN [29] head to predict the final results. BAT [30] encodes box information in Box Cloud to incorporate structural information. MLVSNNet [31] proposes to perform multi-level Hough voting for aggregating information from different levels. PTT [32] proposes a Point-Track-Transformer module to weigh features’ importance. Most existing SOT methods either utilize cosine similarity or cross-correlation to match the search region and template features, which are essentially linear matching processes and can hardly adapt to complex situations where random noise and occlusions are involved. Moreover, the use of detection model prediction heads leads to high computation overheads. Our proposed PTTR addresses the above limitations. In particular, one recent work [24] proposes a new motion-centric tracking paradigm, which directly predicts the motion state based on overlapped point cloud frames. We empirically show that our proposed Point-BEV fusion method in PTTR++ is also applicable to this paradigm.

Vision Transformers: Transformer [33] was first proposed as an attention-based building block in machine translation to replace the RNN architecture. Recently, a number of works [34], [35], [36], [37], [38], [39], [40] apply transformer on 2D vision tasks and achieve great success. Most of these attempts divide the images into overlapping patches and then regard each patch as a token to further apply the transformer architecture. In the 3D domain, PCT [41] generates positional embedding using 3D coordinates of points and adopts transformer with an offset attention module to enrich features of points from its local neighborhood. Point Transformer [42] adopted vectorized self-attention network [43] for local neighbors and designed a Point Transformer layer that is order-invariant to suit point cloud processing. [44] proposes SortNet to gather spatial information from point clouds, which sorts the points by learned scores to achieve order invariance. All of these works focus on shape classification or part segmentation tasks. The attention mechanism in transformer offers correlation modeling with global receptive field, which makes it a good candidate for the 3D tracking problem where feature matching is required. We propose a novel transformer-based module to utilize the attention mechanism for feature aggregation and matching.

BEV-based 3D Perception: The bird’s-eye view (BEV) is a type of compact representation of the 3D space where the height dimension is compressed to form 2D-like dense features. The BEV representation is known for its computational efficiency and it provides a physically interpretable medium for fusing multi-modal information such as multi-view images, radar, and point clouds. A series of LiDAR-based approaches [14], [21], [22], [23], [45], [46], [47], [48] adopt this representation for various perception tasks such as object detection and segmentation. BEV features are typically generated from voxelization with subsequent feature downsampling or pillarization [23].

Recently, BEV-based approaches have received unprecedented attention in the field of multi-view camera-based 3D perception for autonomous driving. Based on the way BEV features are constructed, studies from this line of research can be broadly categorized into depth-based approach and network-based approach. Depth-based methods [49], [50], [51], [52], [53], [54] perform depth estimation and project image features to BEV based on the depth distribution. Differently, network-based methods [55], [56], [57], [58], [59] project 3D positions to images to extract image features and build the BEV representation. Given that BEV features can be conveniently generated from different input modalities, a number of studies [60], [61], [62], [63] naturally extend BEV to multi-modal perception by fusing image and point features in the BEV space. Our work is inspired by the success of these BEV-based perception methods. Different from the aforementioned multi-modal approaches, our method fuses different views of the single point cloud modality.

Multi-view Feature Fusion: Point clouds can be represented in different views and some recent studies [64], [65], [66], [67], [68], [69] attempt to fuse two or more different views for better performance for tasks such as classification and detection. PVCNN [64] proposes to use convolution on voxelized features to replace the expensive ball-query operations in PointNet++ [2] and improve the efficiency. PV-RCNN [65] proposes to integrate voxel-based and point-based networks with two aggregation modules: voxel-to-keypoint encoding for fusing multi-scale semantic features into points and keypoint-to-grid RoI pooling for aggregating local context information. RPVNet [66] designs a network with three branches for voxel, point, and range image, respectively. The voxel and range image views are converted to the point view for multi-view feature propagation at different stages. The fused point features are then projected back to other views. A gating mechanism is used to measure the importance of different views and adaptively aggregate features. FusionNet [69] proposes a deep fusion architecture where neighborhood aggregation is applied to process point and voxel features, and inner-voxel aggregation is used for point-level
and voxel-level feature interaction. To our best knowledge, it is the first attempt to adapt multi-view fusion to the 3D tracking task, where feature matching is involved. Our proposed PTTR++ differs from these methods by using the BEV representation and an attention-based selective fusion method based on cross-view mapping.

III. PTTR: 3D POINT CLOUD TRACKING WITH TRANSFORMER

A. Overview

Given a 3D point cloud sequence, 3D object tracking aims to estimate the object location and orientation in each point cloud observation, i.e., the search region point cloud \( P^s \in \mathbb{R}^{N_s \times 3} \), by predicting a bounding box conditioned on a template point cloud \( P^t \in \mathbb{R}^{N_t \times 3} \). To this end, we propose PTTR, a novel coarse-to-fine framework for 3D object tracking. As shown in Fig. 2, PTTR performs 3D point cloud tracking with three main steps: 1) Feature Extraction (Section III-B), 2) Attention-based Feature Matching (Section III-C), and 3) Prediction Refinement (Section III-D).

Feature Extraction: Following previous methods [15], [16], [18], [19], we employ PointNet++ [2] as the backbone to extract multi-scale point features from the template and the search region. However, important information loss may occur during the point sampling process in the original PointNet++. Therefore, we propose a novel Relation-Aware Sampling method to preserve more points relevant to the given template by leveraging relation-aware feature similarities.

Attention-based Feature Matching: Different from previous methods that use cosine similarity [15], [16], [18] or linear correlation [19] for matching the template and the search region, we utilize novel attention operations and propose Point Relation Transformer (PRT). PRT first utilizes a self-attention operation to adaptively aggregate point features for the template and the search region individually, and then performs feature matching with cross-attention. The coarse prediction is generated based on the output of PRT.

Prediction Refinement: The coarse prediction is further refined with a lightweight Prediction Refinement Module, which results in a coarse-to-fine tracking framework. Based on the coarse predictions, we first conduct a Point Offset operation for seed points from the search region to estimate their corresponding seed points in the template. Subsequently, we employ a Local Pooling operation for the seed points from both point clouds respectively, and then concatenate the pooled features with the matched features from PRT for generating the final prediction.

B. Relation-Aware Feature Extraction

As one of the most successful backbones, PointNet++ [2] introduces a hierarchical architecture with multiple distance-farthest point sampling (D-FPS) and ball query operations, which effectively exploits multi-scale point features. Most existing 3D object tracking methods [16], [18], [19] use PointNet++ for feature extraction. However, it has a non-negligible disadvantage for object tracking: the D-FPS sampling strategy used in PointNet++ tends to generate random samples that are uniformly distributed in the Euclidean space, which often leads to important information loss during the sampling process. In particular, the search region point cloud often has a much larger size than the template, and therefore D-FPS sampling inevitably keeps a substantial portion of background points and leads to sparse point distribution for the object of interest, which further challenges the subsequent object localization in feature matching. To alleviate this issue, previous methods resort to random point sampling [16], [18] or feature-farthest point sampling (F-FPS) [3]. However, the problem of substantial foreground information loss during sampling is not fully resolved.

Relation-Aware Sampling: In this work, we propose to use a novel sampling method dubbed Relation-Aware Sampling (RAS) to preserve more points relevant to the given template by considering relational semantics. Our key insight is that the region of interest in the search region point cloud should have similar semantics to the template. Therefore, points in the search region with higher semantic feature similarities to the template points are more likely to be foreground points. Specifically, given the template point features \( \mathbf{X}^t \in \mathbb{R}^{N_t \times C} \) and search region point features \( \mathbf{X}^s \in \mathbb{R}^{N_s \times C} \), we first calculate the pairwise point feature distance matrix \( \mathbf{D} \in \mathbb{R}^{N_s \times N_t} \):

\[
D_{ij} = ||\mathbf{x}_i^s - \mathbf{x}_j^t||_2, \quad \forall \mathbf{x}_i^s \in \mathbf{X}^s, \quad \forall \mathbf{x}_j^t \in \mathbf{X}^t,
\] (1)

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
where $\| \cdot \|_2$ denotes L2-normalization, and $N_s$ and $N_i$ are the current number of points from the search region and the template, respectively. Subsequently, we compute the minimum distance $V \in \mathbb{R}^{N_s}$ by considering the distance between each point from the search region and its nearest point from the template in the feature space:

$$V_i = \min_{j=1}^{N_i} (D_{ij}), \quad \forall i \in \{1, 2, \ldots, N_s\}. \quad (2)$$

Following previous methods [15, 16, 18, 19], we update the template point cloud for each frame by using the tracking result from the previous search point cloud. In the case that low-quality tracking predictions are encountered in difficult situations, the newly formed templates might mislead RAS and lead to unfavorable sampling results. Moreover, the inclusion of background information offers useful contextual information for the localization of the tracked object. To improve the robustness of the sampling process, we adopt a similar strategy as in [3] to combine our proposed RAS with random sampling. In practice, we sample half of the points with RAS, while the rest of the points are obtained via random sampling. We show the effects of different sampling approaches in Fig. 3. It can be observed that the proposed sampling method can preserve the most foreground points.

C. Relation-Enhanced Feature Matching

Existing 3D object tracking methods perform feature matching between the search region and the template with cosine similarity [15, 16, 18] or linear correlation [19]. Motivated by the success of transformer models in computer vision applications [10, 33, 41, 42], we aim to leverage attention-based mechanisms to improve the accuracy and robustness of 3D point cloud tracking, which is an under-explored area in prior research. Note that although PTT [32] utilizes transformer in their model, they still match the template and search point cloud by cosine similarity, and the transformer module is only used for feature enhancement.

Relation Attention Module: Inspired by recent works studying feature matching [70, 71, 72, 73], we propose a Relation Attention Module (RAM) (shown in Fig. 4) to adaptively aggregate features by predicted attention weights. Firstly, RAM employs linear projection layers to transform the input feature vectors “Query”, “Key” and “Value”. Instead of naively calculating the dot products between “Query” and “Key”, RAM predicts the attention map by calculating the cosine distances between the two sets of L2-normalized feature vectors. With the help of L2-normalization, RAM can prevent the dominance of a few feature channels with extremely large magnitudes. Subsequently, the attention map is normalized with a Softmax operation. In order to sharpen the attention weights and meanwhile reduce the influence of noise [41], we employ the offset attention to predict the final attention map by subtracting the query features from the previously normalized attention map. Consequently, the proposed RAM can be formulated as:

$$\text{Attn} (Q, K, V) = \phi (Q - \text{softmax}(A) \cdot (W_o V)), \quad (3)$$

where $\phi$ represents the linear layer and ReLU operation applied to the output features, the attention matrix $A \in \mathbb{R}^{N_q \times N_k}$ is obtained by:

$$A = \overline{Q} \cdot \overline{K}^\top, \quad \overline{Q} = \frac{W_q Q}{\|W_q Q\|_2}, \quad \overline{K} = \frac{W_k K}{\|W_k K\|_2}, \quad (4)$$

where $\| \cdot \|_2$ is the L2-norm, $Q, K, V$ represent the input “Query”, “Key” and “Value” respectively, and $W_q, W_k$ and $W_o$ denote the corresponding linear projections.
Point Relation Transformer: By incorporating RAM, we propose the Point Relation Transformer (PRT) module to adaptively exploit the correlations between point features for context enhancement. The PRT module first performs a self-attention operation for the search region and the template features, respectively. Subsequently, PRT employs a cross-attention operation for gathering cross-contextual information between the two point clouds. Both operations use global attention, where all input point feature vectors are considered tokens. Formally, PRT is formulated as:

\[
\bar{X}^s = \text{Attn}(X^s), \quad \text{and} \quad \bar{X}^t = \text{Attn}(X^t), \quad (5)
\]

\[
\hat{X}^s = \text{Attn}(\bar{X}^s, \bar{X}^t, \bar{X}^t), \quad (6)
\]

where Attn(Q, K, V) denotes our proposed Relation Attention Module, \(\bar{X}^s\) denotes the matched features, and \(\hat{X}^s\) and \(\hat{X}^t\) denote the enhanced search and template features respectively. By using global self-attention, the exploited features can obtain a global understanding of the current observation. Note that the self-attention uses the same point features as \(Q, K, V\), and both self-attention operations share weights so as to project the search region and template features into the same latent space. Thereafter, the cross-attention performs pairwise matching between query tokens \(\hat{X}^s\) and key tokens \(\hat{X}^t\), which exploits cross-contextual information for \(\hat{X}^s\) by capturing correlations between the two sets of point features. Based on the relation-enhanced point features \(\hat{X}^s\), we generate the coarse prediction results for 3D object tracking.

D. Coarse-to-Fine Tracking Prediction

The majority of the existing point tracking approaches adopt prediction heads of detection models to generate the predictions, e.g., P2B [16] adopts the clustering and voting operations of VoteNet [17] and 3D-SiamRPN [19] uses an RPN [29], [74] head. However, these prediction heads introduce extra computation overheads, which largely limit their efficiency. To circumvent this issue, we propose a novel coarse-to-fine tracking scheme. The coarse prediction \(Y^c\) is predicted by directly regressing the relation-enhanced features \(\hat{X}^s\) from the proposed PRT module with Multi-Layer-Perceptron (MLP). Remarkably, \(Y^c\) provides faithful tracking predictions for most cases and also surpasses the tracking performance of prior methods.

Prediction Refinement Module: To further refine the tracking predictions, we propose a lightweight Prediction Refinement Module (PRM) to generate the final predictions \(Y^f\) based on \(Y^c\). Specifically, we use the sampled points from the search region point cloud as seed points, and then we estimate their correspondences in the template by using an offset operation for \(Y^c\). Subsequently, we encode local discriminative feature descriptors for the seed points from both sources, which is achieved by using Local Pooling operations to group neighboring point features. The neighboring features are grouped with ball-query operations with a fixed radius \(r\). Lastly, we concatenate \(\hat{X}^s\) with the pooled features from the source and the target, based on which we generate the final prediction \(Y^f\):

\[
Y^f = \gamma \left( \left[ F^s, F^t, \hat{X}^s \right] \right), \quad (7)
\]

where \(F^s\) and \(F^t\) are the pooled features from the search region and the template respectively, \([\cdot]\) denotes concatenation, and \(\gamma\) represents the MLP networks. We highlight that even with the refinement stage, our proposed method still achieves competitive inference speed thanks to the lightweight design.

Training Loss: Our PTTTR is trained in an end-to-end manner. The coarse prediction \(Y^c\) and the final prediction \(Y^f\) are in the same form that each contains a classification component \(Y_{cls} \in \mathbb{R}^{N_x \times 1}\) and a regression component \(Y_{reg} \in \mathbb{R}^{N_x \times 4}\), where \(N_x\) denotes the number of sampled points from the search region. \(Y_{cls}\) predicts the objectiveness of each point, and \(Y_{reg}\) consists of the predicted offsets along each axis \(\{\Delta x, \Delta y, \Delta z\}\) with an additional rotation angle offset \(\Delta \theta\). For each prediction, we use a classification loss \(L_{cls}\) defined by binary cross-entropy, and a regression loss \(L_{reg}\) in the form of mean square error. Consequently, our overall loss function is formulated as:

\[
\mathcal{L}_{total} = \mathcal{L}_{cls} \left( Y^c_{cls}, Y^{gt}_{cls} \right) + \mathcal{L}_{reg} \left( Y^c_{reg}, Y^{gt}_{reg} \right) + \lambda \left( \mathcal{L}_{cls} \left( Y^f_{cls}, Y^{gt}_{cls} \right) + \mathcal{L}_{reg} \left( Y^f_{reg}, Y^{gt}_{reg} \right) \right), \quad (8)
\]

where \(Y^{gt}\) denotes the corresponding ground truth, and \(\lambda\) is a weight coefficient.

E. Implementation Details

Template and Search Region: During training, we use the ground truth bounding box to crop the point cloud to form the template. In order to simulate the disturbances the model might encounter, we add random distortions to augment the bounding boxes with a range of [-0.3 to 0.3] along the x, y, and z axes. For both training and testing, we extend the box with a ratio of 0.1 to include nearby background points. We enlarge the template bounding box by 2 meters in all directions to form the search region.

Model Details: We use PointNet++ [2] with 3 set-abstraction layers as the backbone. The radius of these SA layers is set to 0.3, 0.5, and 0.7 meters, respectively. In the first stage, we use a 3-layer MLP for classification and regression, respectively. Each layer is followed by a BN [75] layer and a ReLU [76] activation layer. In PRM, the Local Pooling is conducted with a ball-query operation and a grouping operation [2] with a radius of 1.0 m. After pooling, we obtain the concatenated features that are fed into a 5-layer MLP for generating the final predictions.

IV. PTTR++: EXPLORING POINT-BEV FUSION FOR 3D POINT CLOUD TRACKING

A. Overview

As introduced in Section I, motivated by the favorable properties of the Bird’s-Eye View in capturing object motion and its potential to complement the point-wise representation, we further propose a new 3D tracking framework on top of PTTR and name it as PTTR++, which explores Point-BEV fusion for point cloud tracking. As shown in Fig. 5, PTTR++ consists of two stages, namely Dual Branch Feature Matching (in Section IV-B) and Point-BEV Feature Fusion (in Section IV-C).
In the first stage, PTTR++ performs template-search feature matching for the point and BEV features independently with transformer operations. In the second stage, it first maps the features from one branch to another, and then adaptively fuses the mapped futures to generate the tracking prediction. Note that we do not include the Prediction Refinement Module in PTTR++ as the focus is the exploration of the synergy of the two point cloud representations. Our empirical results demonstrate that PTTR++ achieves significantly improved performance and competitive efficiency without incorporating PRM thanks to the strong complementary effect of the two representations.

B. Dual Branch Feature Matching

To exploit the complementary information in the point-wise view and its corresponding BEV representation, we introduce two parallel feature-matching branches, namely the point branch and the BEV branch. Each branch consists of a backbone network that extracts features for the template and search region respectively, as well as a transformer-based matching module that performs template and search region feature matching. In this stage, the two branches encode features individually.

The point branch is inherited from PTTR that PRT is used to perform feature matching between template and search point features as introduced in Section III. We denoted the matched point features as \( F_P \). For the BEV branch, we first perform the pillarization process to convert the raw point cloud into BEV features. As Fig. 5 illustrates, we first divide the x-y plane into evenly spaced grid to form the pillars. For each pillar, the points that fall inside the pillar are first augmented with the arithmetic mean of the point coordinates \((x_c, y_c, z_c)\) and the distance from the pillar center \((x_p, y_p)\). Following BAT [30], we additionally add object size information to the points by attaching the ground truth bounding box size \((w, h, l)\) to each point. As a result, together with the original point coordinates \((x, y, z)\), the point features have 11 dimensions in total. We then use a linear layer to encode the pillars followed by a max-pooling operation to obtain the pillar features of shape \((P, C)\), where \(P\) is the number of pillars and \(C\) is the number of feature channels. Note that the pillarization process is efficient for sparse point clouds since only non-empty pillars need to be processed. Subsequently, the pillars are scattered back to their original location inside the feature maps to obtain template and search region BEV features of shape \((H, W, C)\), where \(H\) and \(W\) indicate the 2D grid size. By considering each element inside the feature maps as a token, similar to the point branch, we employ the proposed PRT to match BEV features. Differently, we use the sinusoidal function [33] to generate positional embeddings for each BEV feature token. The output of the matching module is then processed by a BEV backbone consisting of convolution layers to obtain the final matched BEV features \( F_B \). As shown in Fig. 5, the BEV branch differs from the point branch in that the BEV matching module is positioned before the BEV backbone which consists of convolution layers that downsample the feature maps. It allows the feature matching to be performed at higher resolution, which potentially leads to more accurate localization. We empirically verify that the proposed setting brings improved tracking performance. We refer the readers to our ablation studies in Section V-C2 for details.

C. Point-BEV Feature Fusion

To fuse the matched features from the point and BEV branches, we first leverage Cross-view Feature Mapping (CFM) to transform features from one branch to the other (e.g., from the point branch to the BEV branch). Afterward, we propose...
a Selective Feature Fusion (SFF) operation to adaptively fuse the mapped cross-view features. In this stage, we focus on combining the advantages of the point-based view and the BEV representation for tracking prediction.

Cross-View Feature Mapping: CFM can be performed in both directions, namely Point-to-BEV mapping and BEV-to-Point mapping, and we introduce both of them here. The Point-to-BEV mapping is achieved via grid-based average pooling, which is similar to the pillarization process introduced in Section IV-B. Based on the corresponding point coordinates \( \{ s_k \} \) of the point features \( \{ F_P^k \} \) as well as the range and grid size of the target BEV representation, the Point-to-BEV mapping can be described as:

\[
F_{h,w}^{P \rightarrow B} = \frac{1}{N_{h,w}} \sum_{k=1}^{K} 1_{s_k \in P_{h,w}} F_k^P
\]

where \( h \) and \( w \) index the target BEV feature maps \( F^{P \rightarrow B} \), \( P \) is the set of pillars corresponding to the grid, \( K \) is the number of points, \( N_{h,w} \) is the number of points within \( P_{h,w} \), and \( 1 \) is the indicator function.

The BEV-to-Point mapping can be regarded as the reserve process of Point-to-BEV mapping. Instead of assigning the same features to all the points that fall inside the same pillar, we use bilinear interpolation to resample point features from the BEV features based on their corresponding x-y coordinates. The resampled point features are denoted as \( F^{B \rightarrow P} \).

Selective Feature Fusion: After obtaining the mapped features from the 3D point cloud and BEV representations, we propose to adaptively fuse the two sets of features following the similar spirit of SENet [77]. As Fig. 6(a) illustrates, SENet works on 2D image features and it first performs the “squeeze” operation by downsampling the feature maps via global average pooling, followed by the “excitation” operation which generates channel-wise attention weights with linear layers and the sigmoid function. Finally, the attention weights are broadcast and multiplied with the input features as a gating mechanism. Our method differs from SENet in that two sets of features from the point and BEV branches are involved. In order to generate meaningful attention weights that can effectively select informative features from both inputs, we first sum the features from both branches before the squeeze and excitation operations that generate the attention weights. By taking the BEV branch as an example (Fig. 6(b)), we can formulate this process as:

\[
w = \sigma \left( \gamma \left( \text{AvgPool} \left( F^B + F^{P \rightarrow B} \right) \right) \right) \quad (11)
\]

where \( \gamma \) and \( \sigma \) represent linear layers and the sigmoid function, and \( w \) denotes the attention weights. We then multiply \( w \) and \( 1 - w \) to both input features before the summation so that the fusion process essentially acts as a self-gating mechanism to adaptively select useful information from both branches:

\[
\tilde{F}^B = w F^B + (1 - w) F^{P \rightarrow B} \quad (12)
\]

where \( \tilde{F}^B \) denotes the fused features on the BEV branch. On the other hand, for the point branch, as points are unstructured and different points might have distinct feature responses, making it unsuitable to apply a global channel-wise attention weight. We instead remove the global average pooling step to perform channel-wise re-weighting for each individual point as shown in Fig. 6(c). The rest of the fusion process is similar to the BEV branch and the fused point features are denoted by \( \tilde{F}^P \).

For simplicity, we name the attention mechanism for the BEV branch as global attention and the point-wise version (for the point branch) as point-wise attention.

In our empirical analysis, we conduct experiments to evaluate feature fusion on both branches and select BEV-to-Point mapping and fusion on the point branch as our proposed method (as illustrated in Fig. 5) as it achieves more competitive and balanced performance. Please refer to the experimental results in Table VIII for details.

Training Loss: We apply the same prediction head and training loss as PTTR on the fused point features \( F^P \). Note that the Prediction Refinement Module proposed in PTTR is omitted to reduce the computational cost. In addition, we also apply a prediction head on the matched BEV features \( F^B \) to obtain the BEV prediction \( Y^B \) to allow for additional supervision on the BEV branch. The overall loss function is computed as:

\[
L = L_{BCE}(Y^P_{cls}, \hat{Y}^P_{cls}) + \alpha L_{MSE}(Y^P_{reg}, \hat{Y}^P_{reg}) \\
+ L_{BCE}(Y^B_{cls}, \hat{Y}^B_{cls}) + \beta L_{MSE}(Y^B_{reg}, \hat{Y}^B_{reg}), \quad (13)
\]

where \( Y^P \) and \( Y^B \) denote the predictions of both branches, \( \hat{Y} \) indicates the ground truth, and \( \alpha \) and \( \beta \) are the loss coefficients. During inference, prediction is only generated on the point branch.

D. Implementation Details

We mostly follow the settings of PTTR as introduced in Section III-E. For the additional BEV branch, we use a point cloud range of \([(-4.8, 4.8), (-4.8, 4.8), (-1.5, 1.5)]\) meters and a pillar size of 0.3 meters for smaller objects including car, van, pedestrian, and cyclist. For large objects such as trucks, trailers, and buses, we adopt a range of \([(-12.0, 12.0), (-12.0, 12.0), (-4.0, 4.0)]\) meters and a pillar size of 0.5 meters. The point features are projected to 64 dimensions using one-layer MLP after pillarization. In the BEV backbone network, we use 3 convolution blocks with \([128, 128, 256]\) channels respectively.
TABLE I

| Method         | Car 6424 | Pedestrian 6088 | Van 1248 | Cyclist 308 | Average by Class | Average by Frame |
|----------------|----------|-----------------|----------|-------------|------------------|------------------|
| SC3D [15]     | 41.3 / 37.9 | 18.2 / 37.8 | 40.4 / 47.0 | 41.5 / 70.4 | 35.4 / 53.3 | 31.2 / 48.5 |
| P2B [16]      | 56.2 / 72.8 | 28.7 / 49.6 | 40.8 / 48.4 | 32.1 / 44.7 | 39.5 / 53.9 | 42.4 / 60.0 |
| 3D-SiamRPN [19] | 58.2 / 76.2 | 35.2 / 56.2 | 45.7 / 52.9 | 36.2 / 49.0 | 43.8 / 58.6 | 46.6 / 64.9 |
| SA-P2B [18]   | 58.0 / 75.1 | 34.6 / 63.3 | 51.2 / 63.1 | 32.0 / 43.6 | 44.0 / 61.3 | 46.7 / 68.2 |
| MLVSN [31]    | 56.0 / 74.0 | 34.1 / 61.1 | 52.0 / 61.4 | 34.3 / 44.5 | 44.1 / 60.3 | 45.7 / 66.7 |
| BAT [30]      | 60.5 / 77.7 | 42.1 / 70.1 | 52.4 / 67.0 | 33.7 / 45.4 | 47.2 / 65.1 | 51.2 / 72.8 |
| PTT [32]      | 67.8 / 81.8 | 44.9 / 72.0 | 43.6 / 52.5 | 37.2 / 47.3 | 48.4 / 63.4 | 55.1 / 74.2 |
| LTTR [83]     | 65.0 / 77.1 | 33.2 / 56.8 | 35.8 / 45.6 | 66.2 / 89.9 | 50.0 / 67.4 | 48.7 / 65.8 |
| V2B [80]      | 70.5 / 81.3 | 48.3 / 73.5 | 50.1 / 58.0 | 40.8 / 49.7 | 52.4 / 65.6 | 58.4 / 75.2 |

Success / precision are used for evaluation. * indicates results reproduced based on the official implementation. M^2-track++ denotes M^2-track with our proposed Point-BEV fusion. "Improvement" refers to performance gain introduced by our proposed Point-BEV fusion.

and each block has a down-sample factor of 2. In attention-based feature fusion, we follow SENet [77] and use a reduction ratio $r$ of 16. For loss computation, we set the coefficients $\alpha$ and $\beta$ as 100 and 2, respectively.

V. EXPERIMENTS

A. Experimental Settings

Datasets: We evaluate our proposed method on the most commonly used KITTI [78] and nuScenes [79] SOT datasets. For a fair comparison on the KITTI dataset, we follow the data split specified in [15], [16] and use scenes 0-16 for training, 17-18 for validation, and 19-20 for testing. For the nuScenes dataset, we follow the implementation of [30] and report the performance of five categories including Car, Pedestrian, Truck, Trailer, and Bus. The nuScenes dataset only provides annotation for 1 in 10 consecutive frames, and the annotated frames are defined as keyframes. The evaluation is performed on keyframes only.

Evaluation Metrics: We follow existing works [16], [80] and use Success and Precision as defined in one pass evaluation (OPE) [81] as the evaluation metrics. Specifically, Success measures the IoU between the predicted box and the ground truth, while Precision computes the AUC of the distance between prediction and ground truth box centers. We report the performance of each object category as well as the average over all classes.

Training and Testing: For the KITTI tracking dataset, we train the model for 160 epochs with a batch size of 64. We use Adam optimizer [82] with an initial learning rate of 0.001 and reduce it by 5 every 40 epochs. For nuScenes [79], we train the model for 30 epochs with a batch size of 128. We set the initial learning rate to 0.001 and divide it by 5 every 6 epochs. During testing, we use the previous prediction result as the next template. In line with [15], [16], we use the ground truth bounding box as the first template.

B. Benchmarking Results

Results on KITTI: We compare PTTR and PTTR++ with existing state-of-the-art 3D SOT methods. In particular, to validate that the proposed Point-BEV fusion in PTTR++ is a generic approach, we integrate it with a recent method M^2-Track [24] and denote it as M^2-Track++. M^2-Track is a contemporary work to our conference version, which differs from existing matching-based methods in that it predicts the motion state based on overlapped point cloud frames. We refer the readers to [24] for details. We incorporate the proposed Point-BEV fusion to the targetness prediction stage that voxelization is performed on the overlapped point cloud input to form BEV features, which are then fused with point features to generate the segmentation scores. As reported in Table I, PTTR outperforms all existing matching-based methods while PTTR++ further boosts the average performance by a large margin of 6.0/4.7 in success and precision, respectively. On top of M^2-Track, our simple integration of the proposed Point-BEV fusion strategy also brings a notable improvement of 3.4/3.6 on average, showing the wide applicability of the proposed approach.

Results on nuScenes: Table III compares the tracking performance on the nuScenes dataset. Our proposed PTTR again demonstrates competitive performance and outperforms existing matching-based methods. In addition, PTTR++ further improves the tracking accuracy as compared to PTTR by a significant margin of 8.9/11.06 in average success and precision and achieves state-of-the-art performance. We find that the proposed Point-BEV fusion is especially beneficial to larger objects such as trucks and buses.

Inference Time: Efficiency has always been an important aspect of object tracking due to its time-sensitive applications. We compare the inference time of our method with existing methods with available open-source implementations on the KITTI dataset. We run all the models on the same machine and report the inference time in Table II. It can be observed that our
methods have comparable inference time to existing methods. In particular, PTTR++ only adds a small proportion of inference time on top of PTTR despite having an extra BEV branch. This is because the BEV branch is lightweight thanks to the efficient pillarization process and convolution operations as opposed to the expensive point grouping operation.

C. Analysis Experiments

To evaluate the effectiveness of the components proposed in PTTR and PTTR++, we conduct extensive ablation studies on the KITTI [78] dataset and report the experimental results.

1) Ablation Studies on PTTR: Point Sampling Methods: We compare our proposed Relation-Aware Sampling (RAS) method with existing sampling approaches, including random sampling [16], distance-farthest point sampling (D-FPS) [2] and feature-farthest point sampling (F-FPS) [3]. As shown in Table IV, RAS yields the best performance with a clear margin. By utilizing RAS, our method achieves an increase of 5.5/8.8 in success/precision as compared to the random sampling baseline. Small objects usually consist of fewer points, and hence are more sensitive to the point sparsity challenge. For the pedestrian class, which is the class of the smallest object size, RAS significantly boosts the results from 36.6/59.9 to 50.9/81.6.

Model Components: We conduct experiments to investigate the effectiveness of the proposed Point Relation Transformer (PRT) and Prediction Refinement Module (PRM). For the ablation studies on PRT, we replace PRT with cosine similarity for feature correlation computation as in existing methods [15], [16], [18]. We also compare the performance w/o PRM.

The experimental results of comparing methods are reprinted from [30]. Success/precision are used for evaluation. "Improvement" refers to performance gain introduced by our proposed Point-BEV fusion.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 8. Visualization of tracking results from individual branches. Our proposed PTTR++ generates more robust tracking predictions as compared to the point branch and the BEV branch by exploiting the complementary information of both representations.

**TABLE VI**

Ablation Studies on Relation Attention

| Offset | Norm | Car | Pedestrian | Van | Cyclist | Average |
|--------|------|-----|------------|-----|---------|---------|
| ✓      | 65.4 / 69.0 | 36.6 / 65.1 | 34.6 / 36.6 | 35.9 / 78.8 | 45.6 / 62.6 |
| ✓ ✓    | 56.6 / 69.1 | 60.3 / 67.3 | 54.3 / 59.6 | 63.7 / 90.3 | 52.2 / 71.6 |
| ✓ ✓ ✓  | 63.7 / 75.3 | 47.1 / 73.5 | 53.0 / 60.4 | 64.1 / 89.5 | 57.0 / 74.7 |
| ✓ ✓ ✓ ✓ | 65.2 / 77.4 | 51.9 / 81.6 | 52.5 / 61.8 | 65.1 / 90.5 | 58.4 / 77.8 |

We investigate the effectiveness of the two major modifications in our proposed relation attention module. Offset refers to offset attention and norm refers to feature normalization.

the coarse predictions are further corrected in the refinement stage, especially when point sparsity or large movements are present.

Components of Relation Attention: The main differences between our proposed Relation Attention and regular transformer attention are the L2-normalization applied on the query and key features and the offset attention. Ablation studies on each component are reported in Table VI. Both two operations improve the model performance, especially the L2-normalization. It reveals that the cosine distance facilitates point cloud feature matching.

2) Ablation Studies on PTTR++: Effectiveness of Point-BEV Fusion: We conduct a series of experiments to examine the effectiveness and the working mechanism of our proposed Point-BEV fusion. As Table VII shows, experiments #1 and #2 study the single-branch setting where both feature extraction and matching are performed on either point-wise view or BEV alone. The default dual-branch setting (#5) in PTTR++ outperforms both single-branch settings by large margins, demonstrating the complementary nature of the point view and BEV. Note the point-only branch and the BEV-only branch are advantageous on different classes e.g., the success/precision of the BEV branch is 4.8/1.8 lower than the point branch on the pedestrian class. To further verify that the performance improvements do come from the complementary effect of two matching methods instead of the representation power of both views, we conduct experiments #3 and #4 by extracting both point-wise and BEV features and fusing the two types of features before the matching process. Subsequently, matching is performed either on the point branch (#3) or the BEV branch (#4) only. We can see that no clear performance gains are obtained under such settings even when Point-BEV fusion is performed before matching. The results validate that the main source of the performance gain is the complementary effect of the two matching processes. We also qualitatively compare PTTR++ with the single-branch setting. As Fig. 8 shows, PTTR++ generates more robust and accurate tracking predictions by making use of the complementary information from the point-wise and BEV representations.

Point-BEV Fusion Methods: We experiment on different Point-BEV fusion methods including addition, global attention (i.e., attention with global average pooling), and point-wise attention. Besides, we perform feature fusion on the point branch and the BEV branch, respectively. As shown in Table VIII, Point-BEV fusion via simple feature addition on either branch leads to significantly improved results as compared to the single-branch results reported in Table VII, which demonstrates the complementary effect of the features from both views. By applying global attention during feature fusion, we observe slightly improved performance on the BEV branch and decreased performance on the point branch as compared to the feature addition fusion method. When point-wise attention is applied, we instead observe increased performance when fusion is performed on the point branch and lowered accuracy on the BEV branch. It shows that global attention is more suitable for BEV features, while point-wise attention is applicable to point features as different points might possess distinct feature responses and it is not advisable to apply unified channel-wise weights to all points. Finally, we select BEV-to-Point mapping and point-wise attention as our default setting since it achieves the best overall accuracy.

BEV Branch Design Choices: As introduced in Section IV-B, the BEV branch differs from the point branch in that we match the template and search region features before the BEV backbone network downsamples the feature maps. It allows BEV feature matching at a higher resolution for more accurate object localization. We compare the tracking accuracy when feature matching is performed under different resolutions. Apart from our default setting (no downsampling before feature matching), we apply layers in the backbone network to downsample the feature maps by 2x and 4x before the matching module and evaluate the performance. As shown in Table IX, matching the BEV features at higher resolution clearly leads to higher overall performance, which validates our proposed design choice.
TABLE VII
ABLATION STUDIES ON THE EFFECTIVENESS OF POINT-BEV FUSION

| # | Point Feature | BEV Feature | Point Matching | BEV Matching | Car       | Pedestrian | Van       | Cyclist | Average |
|---|---------------|-------------|----------------|--------------|-----------|------------|-----------|---------|---------|
| 1 | ✓             |             | ✓              | ✓            | 62.9 / 74.3 | 49.1 / 77.7 | 50.7 / 58.7 | 64.1 / 90.0 | 56.7 / 75.2 |
| 2 | ✓             | ✓           | ✓              | ✓            | 68.5 / 80.7 | 44.3 / 75.9 | 51.3 / 59.4 | 70.1 / 93.4 | 58.6 / 77.4 |
| 3 | ✓             | ✓           | ✓              | ✓            | 65.5 / 78.2 | 50.6 / 78.3 | 48.4 / 54.6 | 64.8 / 90.8 | 57.4 / 75.5 |
| 4 | ✓             |             | ✓              | ✓            | 64.9 / 76.0 | 45.4 / 73.5 | 52.1 / 60.0 | 67.4 / 91.9 | 57.5 / 75.4 |
| 5 | ✓             |             | ✓              | ✓            | 73.4 / 84.5 | 55.2 / 84.7 | 55.1 / 62.2 | 71.6 / 92.8 | 63.8 / 81.0 |

TABLE VIII
ABLATION STUDIES ON POINT-BEV FUSION METHODS

| Fusion Method | Fusion Branch | Car       | Pedestrian | Van       | Cyclist | Average |
|---------------|---------------|-----------|------------|-----------|---------|---------|
| Addition      | ✓             | ✓         | 73.0 / 84.5 | 53.7 / 82.2 | 54.8 / 61.9 | 70.5 / 92.9 | 63.0 / 80.4 |
|               |               |           | 70.8 / 82.5 | 56.7 / 84.8 | 51.5 / 58.6 | 71.6 / 93.6 | 62.7 / 79.9 |
| Global Attention | ✓         | ✓         | 71.5 / 83.3 | 52.0 / 81.1 | 54.3 / 60.5 | 69.8 / 92.7 | 61.9 / 79.4 |
|               |               |           | 69.9 / 82.3 | 56.7 / 85.4 | 51.6 / 58.1 | 74.1 / 94.3 | 63.1 / 80.0 |
| Point-wise Attention | ✓        | ✓         | 73.4 / 84.5 | 55.2 / 84.7 | 55.1 / 62.2 | 71.6 / 92.8 | 63.8 / 81.0 |
|               |               |           | 69.5 / 81.6 | 54.8 / 83.8 | 51.5 / 58.6 | 71.9 / 93.6 | 61.9 / 79.4 |

TABLE IX
ABLATION STUDIES ON BACKBONE DESIGN CHOICES

| Downsample Ratio | Car       | Pedestrian | Van       | Cyclist | Average |
|------------------|-----------|------------|-----------|---------|---------|
| 1x               | 73.4 / 84.5 | 55.2 / 84.7 | 55.1 / 62.2 | 71.6 / 92.8 | 63.8 / 81.0 |
| 2x               | 70.1 / 82.0 | 52.7 / 82.8 | 53.5 / 60.9 | 63.5 / 91.1 | 60.5 / 79.2 |
| 4x               | 67.3 / 79.1 | 49.9 / 77.1 | 50.4 / 56.3 | 69.4 / 92.6 | 59.3 / 76.3 |

Downsample ratio refers to the number of times the BEV features are downsampled before the matching process.

Fig. 9. Examples of failure cases. Our tracking failures mainly occur when the point clouds are too sparse.

VI. LIMITATION

We show in Fig. 9 the failure cases encountered by our model, which mainly occur when the point clouds are too sparse that the model can hardly capture enough patterns to effectively match template and search point clouds. One possible way to further mitigate this issue could be utilizing complementary multi-frame information for object tracking, which can be explored in future research.

VII. CONCLUSION

In this paper, we have proposed PTTR, a novel framework for 3D point cloud single object tracking, which contains a designed Relation-Aware Sampling strategy to tackle point sparsity, a novel Point Relation Transformer for feature matching, and a lightweight Prediction Refinement Module. Moreover, motivated by the advantages of the bird’s-eye view of point cloud in capturing object motion, we have designed a more advanced framework named PTTR++ by exploiting the complementary information in point-wise and BEV representations. The proposed Point-BEV fusion strategy in PTTR++ substantially boosts the tracking performance and it can also be easily integrated with other tracking approaches as a generic method.

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,” 2017, arXiv: 1706.02413.
[3] Z. Yang, Y. Sun, S. Liu, and J. Jia, “3DSSD: Point-based 3D single stage object detector,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11040–11048.
[4] S. Shi, X. Wang, and H. Li, “PointrCNN: 3D object proposal generation and detection from point cloud,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 770–779.
[5] Z. Luo et al., “Unsupervised domain adaptive 3D detection with multi-level consistency,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 8866–8875.
[6] A. Xiao, J. Huang, D. Guan, F. Zhan, and S. Lu, “SynLiDAR: Learning from synthetic LiDAR sequential point cloud for semantic segmentation,” 2021, arXiv:2107.05399.
[7] A. Xiao, J. Huang, D. Guan, and S. Lu, “Unsupervised representation learning for point clouds: A survey,” 2022, arXiv:2202.13589.
[8] L. Landrieu and M. Simonovsky, “Large-scale point cloud semantic segmentation with superpoint graphs,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4558–4567.
[9] J. Ren, L. Pan, and Z. Liu, “Benchmarking and analyzing point cloud classification under corruptions,” 2022, arXiv:2202.03777.
[10] L. Pan et al., “Variational relational point completion network,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 8524–8533.
[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] Y. Wang, K. Kitani, and X. Weng, “Joint object detection and multi-object tracking with graph neural networks,” in Proc. IEEE Int. Conf. Robot. Automat., 2021, pp. 13 708–13 715.
[13] X. Weng, Y. Wang, Y. Man, and K. Kitani, “Graph neural networks for 3D multi-object tracking,” 2020, arXiv: 2008.09506.

[14] T. Yin, X. Zhou, and P. Krahenbuhl, “Center-based 3D object detection and tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 6329–6338.

[15] 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.

[16] X. Zhou, L. Wang, Z. Yuan, K. Xu, and Y. Ma, “Structure aware 3D single object tracking of point cloud,” J. Electron. Imag., vol. 30, no. 4, 2021, Art. no. 043010.

[17] 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, Feb. 2021.

[18] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, “Volumetric multi-view CNNs for object classification on 3D data,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 5648–5656.

[19] Z. Fang and S. Zhou, “End-to-end learning for point cloud based 3D object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4490–4499.

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

[21] 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/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 12 697–12 705.

[22] C. Zheng et al., “Beyond 3D Siamese tracking: A motion-centric paradigm for 3D single object tracking in point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 8111–8120.

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

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

[25] H. Karunasekera, H. Wang, and H. Zhang, “Multiple object tracking with attention to appearance, structure, motion and size,” IEEE Access, vol. 7, pp. 10 0432–10 0434, 2019.

[26] X. Weng and K. Kitani, “A baseline for 3D multi-object tracking,” 2019, arXiv:1907.03961.

[27] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 91–99.

[28] C. Zheng et al., “Box-aware feature enhancement for single object tracking on point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 13199–13208.

[29] Z. Wang, Q. Xie, Y.-K. Lai, J. Wu, K. Long, and J. Wang, “MLVNet: Multi-level voting Siamese network for 3D visual tracking,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 3101–3110.

[30] J. Shan, S. Zhou, F. Fang, and Y. Cui, “PTT: Point-track-transformer module for 3D single object tracking in point clouds,” in Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst., 2021, pp. 1310–1316.

[31] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.

[32] Z. Liu et al., “Swin transformer: Hierarchical vision transformer using shifted windows,” 2021, arXiv:2103.14030.

[33] X. Chu et al., “Twins: Revisiting the design of spatial attention in vision transformers,” 2021, arXiv:2104.13840.

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

[35] X. Zhu, W. Li, B. Li, L. Wang, and J. Dai, “Deformable DTR: Deformable transformers for end-to-end object detection,” 2020, arXiv:2010.04159.

[36] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “SegFormer: Simple and efficient design for semantic segmentation with transformers,” 2021, arXiv:2105.12600.

[37] G. Zhang, Z. Luo, K. Cui, S. Lu, and E. P. Xing, “Meta-DETR: Image-level few-shot detection with inter-class correlation exploitation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 11, pp. 12832–12843, Nov. 2023.
[67] J. Noh, S. Lee, and B. Ham, “HVPR: Hybrid voxel-point representation for single-stage 3D object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 14605–14614.

[68] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou, “PV-RAFT: Point-voxel correlation fields for scene flow estimation of point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 6954–6963.

[69] F. Zhang, J. Fang, B. Wah, and P. Torr, “Deep fusionnet for point cloud semantic segmentation,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 644–665.

[70] N. Wang, W. 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.

[71] H. Wang et al., “CosFace: Large margin cosine loss for deep face recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 5265–5274.

[72] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, “NormFace: L2 hypersphere embedding for face verification,” in Proc. 25th ACM Int. Conf. Multimedia, 2017, pp. 1041–1049.

[73] T. Chen, S. Komblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 1597–1607.

[74] 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. 8971–8981.

[75] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 448–456.

[76] A. F. Agarap, “Deep learning using rectified linear units (ReLU),” 2018, arXiv:1803.03575.

[77] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 7132–7141.

[78] 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.

[79] H. Caesar et al., “nuScenes: A multimodal dataset for autonomous driving,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11621–11631.

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

[81] M. Kristan et al., “A novel performance evaluation methodology for single-target trackers,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 11, pp. 2137–2155, Nov. 2016.

[82] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[83] Y. Cui, Z. Fang, J. Shan, Z. Gu, and S. Zhou, “3D object tracking with transformer,” 2021, arXiv:2110.14921.

**Zhipeng Luo** received the BEng degree in mechanical engineering and MSc degree in computing from the National University of Singapore, in 2015 and 2018. He is currently working toward the PhD degree with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, under the supervision of Dr. Shijian Lu. He has published multiple top conference papers in the field of computer vision. His research interests include computer vision, object detection, and object tracking.

**Changjing Zhou** received the BEng degree from Shanghai JiaoTong University, in 2018, and the MSc degree from Nanyang Technological University, in 2021. He is currently a research scientist with SenseTime. He has published multiple top conference papers in the field of computer vision and his research interests include computer vision and image processing.

**Yueru Luo** received the MSc degree from Nanyang Technological University, in 2021. She is currently a research assistant with the Chinese University of Hong Kong, Shenzhen. She is currently working toward the PhD degree with CUHK, SZ and her research interests include computer vision, segmentation, and detection.

**Haiyu Zhao** received the bachelor’s degree from Beihang University. He is currently a senior researcher with SenseTime International Pte. Ltd. Previously, he was a research assistant with the Chinese University of Hong Kong Supervised by Prof. Xiaogang Wang. He has published more than 15 papers (with more than 1,900 citations) on top-tier conferences in relevant fields, including CVPR, ICCV, ECCV, NeurIPS, and ICLR.
Ziwei Liu received the PhD degree from CUHK / Multimedia Lab, advised by Prof. Xiaou Tang and Prof. Xiaogang Wang, in 2017. He is a Nanyang assistant professor with the School of Computer Science and Engineering (SCSE), Nanyang Technological University, with MMLab@NTU. Previously, he was a research fellow (2018-2020) in CUHK (with Prof. Dahua Lin) and a postdoc researcher (2017-2018) with UC Berkeley (with Prof. Stella X. Yu). His research interests include computer vision, machine learning and computer graphics. He is fortunate to have internships at Microsoft Research and Google Research. His works include Burst Denoising, CelebA, DeepFashion, Fashion Landmarks, DeepMRF, Voxel Flow, Long Tail, Compound Domain, and Wildlife Conservation.

Shijian Lu received the PhD degree in electrical and computer engineering from the National University of Singapore. He is currently an associate professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His major research interests include image and video analytics, visual intelligence, and machine learning. He has published more than 100 international refereed journal and conference papers and co-authored over 10 patents in these research regions. He is currently an associate editor for the journal Pattern Recognition (PR). He has also served in the program committee of a number of conferences, e.g., the Area Chair of the International Conference on Document Analysis and Recognition (ICDAR) 2017 and 2019, the Senior Program Committee of the International Joint Conferences on Artificial Intelligence (IJCAI) 2018 and 2019, etc.