Correlation Pyramid Network for 3D Single Object Tracking

Mengmeng Wang[1], Teli Ma[1], Xingxing Zuo[2], Jiajun Lv[1], Yong Liu[1*]

[1]Zhejiang University [2]Technical University of Munich
{mengmengwang, lvjiajun314}@zju.edu.cn, telima9868@gmail.com
xingxing.zuo@tum.de, yongliu@iipc.zju.edu.cn *

Abstract

3D LiDAR-based single object tracking (SOT) has gained increasing attention as it plays a crucial role in 3D applications such as autonomous driving. The central problem is how to learn a target-aware representation from the sparse and incomplete point clouds. In this paper, we propose a novel Correlation Pyramid Network (CorpNet) with a unified encoder and a motion-factorized decoder. Specifically, the encoder introduces multi-level self attentions and cross attentions in its main branch to enrich the template and search region features and realize their fusion and interaction, respectively. Additionally, considering the sparsity characteristics of the point clouds, we design a lateral correlation pyramid structure for the encoder to keep as many points as possible by integrating hierarchical correlated features. The output features of the search region from the encoder can be directly fed into the decoder for predicting target locations without any extra matcher. Moreover, in the decoder of CorpNet, we design a motion-factorized head to explicitly learn the different movement patterns of the up axis and the x-y plane together. Extensive experiments on two commonly-used datasets show our CorpNet achieves state-of-the-art results while running in real-time.

1. Introduction

This paper focuses on 3D LiDAR single object tracking (SOT), which is emerged in recent years but is an essential task for 3D applications like autonomous driving, robotics, and surveillance system with the development of 3D sensors like LiDAR. The task aims at tracking a specific target in a video by giving the corresponding 3D target bounding box in the first frame. It is challenging since the target will undergo several changes like occlusions and fast motions, and be sparse and incomplete.

Prior arts mainly addressed the above challenges with a conventional *extractor-matcher-decoder* paradigm. The extractor is used to encode the features of the template and the search areas. The matcher is employed to build the template-search relationship and enhance the potential target features, i.e., embedding the template features into the features of the search region, which is also called correlation operation. The decoder is leveraged to generate bunches of 3D target proposals based on the features from the matcher. Since the modern backbones [23, 24] have become the mainstream and even default choices for the extractor, most trackers in this paradigm dedicate to design more robust and elaborate matchers [5, 16, 25, 36, 38], and more powerful decoders [16, 25]. Despite their great success, we find they still suffer from two limitations. (1) The relationship between the template and the search features is always modeled only in the matcher, which is not sufficient for completely cross-source interaction and target enhancement. (2) The point downsampling in the commonly used backbones will inevitably exacerbate the sparsity of

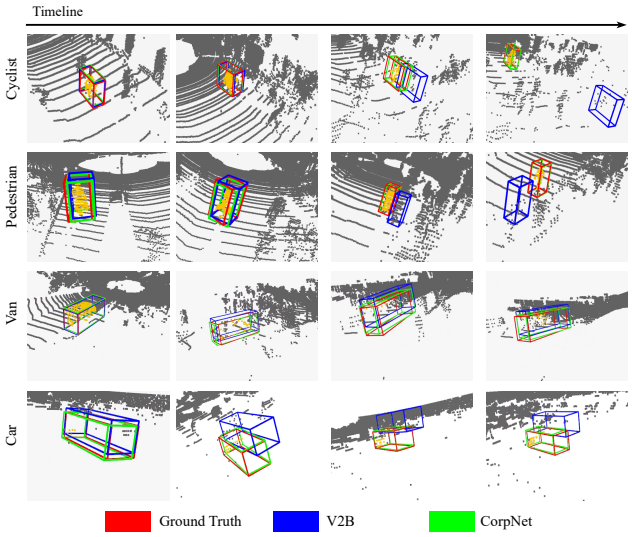

Figure 1. Visualization results of the four different categories. The points of the targets are highlighted in yellow. The red boxes are ground truth bounding boxes. The green boxes are the objects tracked by our CorpNet, while the blue boxes are the results of V2B.

*Yong Liu is the corresponding author.

the point clouds.

The first limitation is mainly caused by the conventional paradigm structure, which separates the extractor and matcher, and makes the matcher be responsible for the template feature enhancement. Inspired by the 2D SOT methods [1, 7, 13, 15, 18, 21, 30, 34], previous methods in this paradigm [9, 11, 16, 17, 25–27, 36, 38] always employ a Siamese-like backbone in the extractor that independently embeds the template and search frames without any intermediate interaction. They then design various matchers to fuse the template features into the search features. However, a standalone matcher is redundant, and the extracted high-level features used in the matcher are not sufficient. M^2-Track [37] has realized part of this problem and proposed a motion-centric paradigm to avoid the Siamese-like backbone and predicts the motion directly. Nevertheless, they still need a motion transformation module to integrate the template information into the search representations, and a two-stage refinement is used to ensure the performance. Inspired by the recent trend in 2D SOT [3, 6, 29, 31], our key insight is that the extractor should be responsible for feature representation and matching simultaneously, so no extra matcher is needed.

For the second limitation, we have noticed that the backbone of previous extractors in 3D SOT remains unexplored. The default configuration is the Siamese-like PointNet [23]/PointNet++ [24], which are not originally designed for 3D SOT. For example, the most commonly-used PointNet++ usually downsamples the input points (typically 1024 or 512 points) by 8×, remaining fewer points (128 or 64 points) for matcher and decoder, and making the intermediate features from the extractor even much sparser than the already sparse input. Actually, the feature embedding backbone plays a core role in object tracking but is overlooked in the previous 3D trackers. It needs to provide a discriminative target representation of the input sparse point clouds, with the around background that inevitably includes distractors and noises. To ease this problem, PTTR [38] proposes a relation-aware sampling strategy for preserving more template-relevant points. However, the total preserved points are still sparse, so they need a two-stage refinement to keep the performance. We emphasize this problem from another perspective, which is to adjust the architecture of the backbone and keep the points from all the stages to formulate multi-scale representations, strengthening the representational capacity of the proposed framework.

Considering the above two limitations, we propose a novel **Cor**relation **P**yramid **Net**work (CorpNet). To be specific, the encoder of CorpNet introduces Self Attention (SA) and Cross Attention (CA) modules in multiple stages of the backbone, strengthening the frame representation at multiple levels by SA and facilitating sufficient interaction to replace the original matcher by CA. Afterward, to cope with the sparsity brought by the downsampling operation, we formulate a correlation pyramid architecture in the encoder to reserve as many points as possible. More specifically, a lateral correlation pyramid structure is devised to effectively combine the point features from all the stages, which have different amounts of points and feature dimensions. Then the pyramidally fused features are voxelized to a volumetric representation and fed to the decoder. The main branch of the encoder deeply embeds the cross-source feature in multiple layers, and the lateral correlation pyramid extensively and directly combines correlated features from low level to high level, resulting in sufficient target-aware feature extraction. Moreover, our CorpNet builds a new decoder based on [16] to process the powerful representation from the encoder, considering that the motion of the *x*-*y* plane and the *z* axis are not exactly identical. Finally, as shown in Fig. 1, with the proposed CorpNet, we achieve state-of-the-art performance on two widely adopted datasets (i.e., KITTI [10] and NuScenes [2]).

The main contributions of our paper can be summarized as follows:

- We propose a novel **Cor**relation **P**yramid **Net**work dubbed CorpNet, which integrates multi-level self attentions to enrich the representations and multi-scale cross attentions to empower sufficient interaction/matching between them.
- We emphasize the sparsity problem of the downsampling operation by a new correlation pyramid structure that fuses the hierarchical correlated features to reserve features of all the points inside different stages.
- We design a new motion-factorized decoder to explicitly decouples the prediction of the *x*-*y* plane and *z* axis for their different motion patterns.

2. Related Works

3D object tracking [4, 11, 14, 22, 33] works with 3D input data like point clouds, stereo images and even monocular images. Here we discuss the task of 3D single object tracking task. This is a new task that has emerged in recent years, which is first defined in SC3D [11]. Inspired by the structure of 2D SOT, previous methods [5, 9, 11, 16, 25, 26, 32, 36, 38], of 3D SOT all inherit a *extractor-matcher-decoder* paradigm. As a pioneer, SC3D lays the foundation of this paradigm with simple components, which matches cosine similarity (matcher) of features between candidates and target (Siamese-like extractor) and regularizes the training using shape completion (decoder).

The following trackers mainly focus on improving SC3D from two folds. The first dedicates to design more robust matchers [5, 16, 25, 27, 36–38]. For instance, MLVSNet [27] uses the CBAM module [28] to enhance the vote

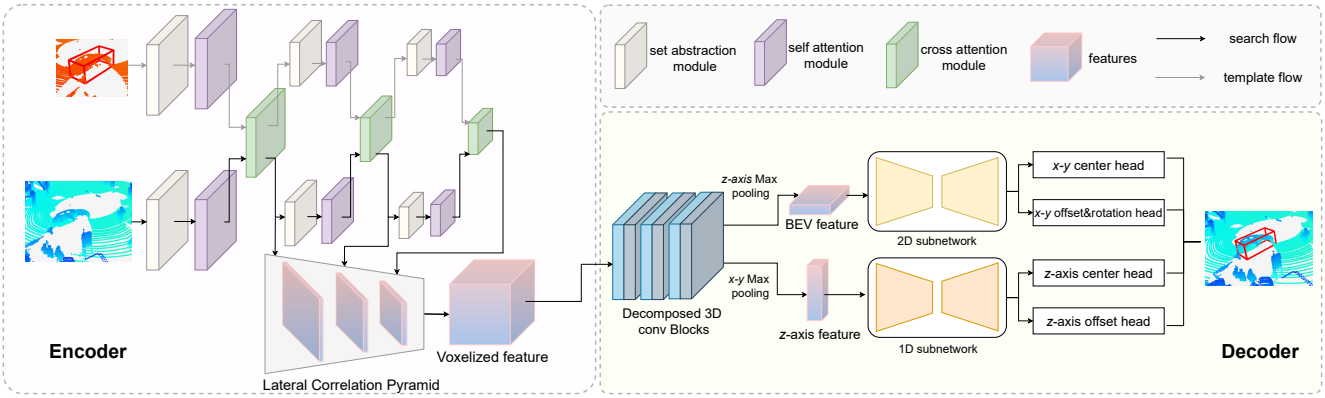

Figure 2. The overall architecture of the proposed CorpNet. Given template and search point clouds and the BBox of the template, a unified encoder is used to extract the representations and match the template and search area simultaneously. A lateral correlation pyramid structure is proposed to handle the sparsity and incomplete challenges by leveraging the hierarchical correlated features. Then a decomposed decoder is designed to obtain the motion of the x-y plane and z-axis separately. The whole pipeline is end-to-end trained with only a single stage.

cluster features with channel attentions and spatial attentions. V2B [16] employs the global and local template feature embedding to strengthen the correlation between the template and search area. M^2-Track introduces a motion-centric paradigm and uses input merge and motion transformation to combine the template and search features instead of conventional correlation operation in the matcher. However, a two-stage refinement is needed to ensure the performance for the lack of appearance matching. The prior effort in the matcher still struggles with standalone matchers, which could not explicitly benefit the extractor and make the structures redundant. In the second fold, the trackers [9, 16, 25–27] have tried to improve the decoder part. P2B [25] employs Hough Voting to predict the target location and many methods follow [36] or improve [26, 27] this manner. LTTR [9] and V2B [16] use center-based regressions to predict several object properties. Even though the matcher and decoder are explored a lot, we find the extractor is always neglected since the modern backbones [23, 24] become the mainstream and default choice. We figure out that the extractor is crucial for a powerful representation to play as the fundamental of the matcher and decoder. Therefore, we shed light on this point and propose a new single-stage correlation pyramid network to explore a unified backbone specified for the 3D SOT task and merge the extractor and matcher together.

3. Method

3.1. Problem Definition

The 3D LiDAR single object tracking (SOT) task is defined as: Given a dynamic 3D sequence of T point clouds $\{\mathcal{P}_i\}_{i=1}^{T}$ and an initial bounding box (BBox) $\mathcal{B}_1 = (x_1, y_1, z_1, w_1, l_1, h_1, \theta_1)$ of a target, our goal is to localize the target BBoxs $\{\mathcal{B}_i\}_{i=2}^{T}$ in sequential frames online, where the subscript stands for the sequence id, (x, y, z) indicates the center coordinate of the BBox, (w, l, h) is the BBox size and θ is the heading angle (the rotation around the *up*-axis). Generally, the BBox size is assumed to keep unchanged across all frames in the 3D scenes even for non-rigid objects (the BBox size for a non-rigid object is defined by its maximum extent in the scene), so we do not need to re-predict the size and simplify $\mathcal{B}_i$ from $\mathbb{R}^7$ to $\mathbb{R}^4$.

More specifically, we follow the conventional tracking pipelines to generate the input consisting of a template area $\mathcal{P}^t = \{\mathbf{P}_i^t\}_{i=1}^{N_t}$ and a search area $\mathcal{P}^s = \{\mathbf{P}_i^s\}_{i=1}^{N_s}$, where $\mathbf{P}_i \in \mathbb{R}^{3+d}$ is a 3D (x, y, z) point with d-dimensional features like intensity and elongation. N_t and N_s are the point number of the template and search areas, respectively. The template $\mathcal{P}^t$ is cropped and centered in the template frames with its corresponding BBox and the search region $\mathcal{P}^s$ is cropped and centered according to $\mathcal{B}_{k-1}$ with an enlarged area in the search frame.

3.2. CorpNet

The overall architecture of the proposed CorpNet is shown in Fig. 2, which consists of a unified encoder and a factorized decoder.

3.2.1 Encoder

The function of the encoder is not only to extract features for the template and search regions but also to simultaneously be responsible for the feature matching or correlation. The popular Siamese-like backbones are not suitable anymore in this situation. We design a unified structure to satisfy the requirements.

As shown in Fig. 2, the encoder of our CorpNet consists of three stages, where each of them contains a set abstraction module [24] to gradually reduce the point number, a self attention (SA) module to enrich the feature representations and a cross attention (CA) module to implement fea-

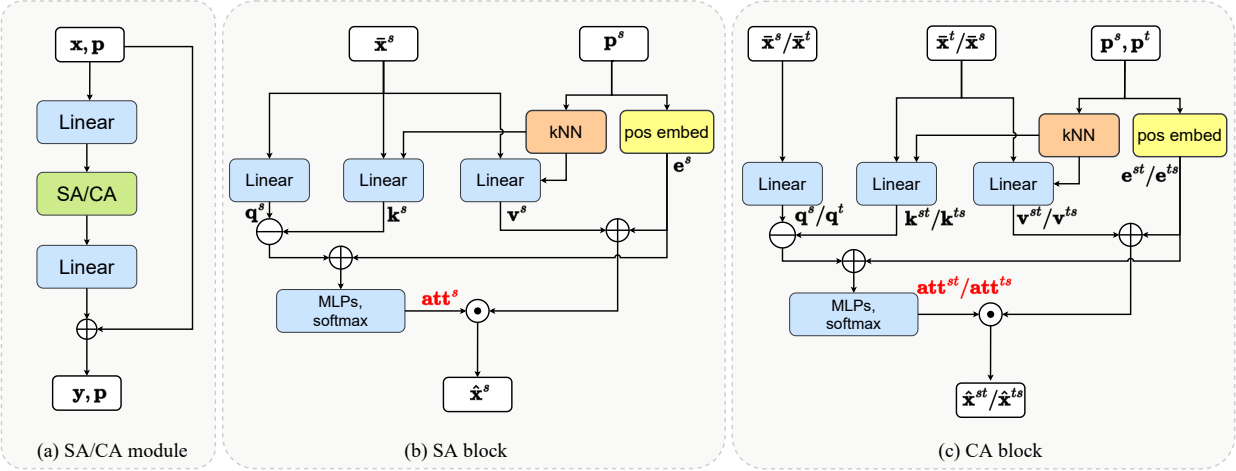

Figure 3. Illustration of the SA/CA module and the SA/CA blocks of the encoder. "att" is short for attention.

ture correlation and interaction. Specifically, the set abstraction modules select a set of points from input points, then group them with the local neighborhood of a ball query and embed the local region patterns into feature vectors with pointwise MLPs (we only use two MLP layers here). Next, each stage's core SA and CA are detailedly introduced as follows.

Most attention mechanisms [8, 12] are global attentions with scalar dot products. It is known that the computation of global attention over all input tokens leads to quadratic complexity. However, the inference speed should be considered in real-time applications for the 3D SOT task. Therefore, a local vectorized self-attention mechanism [35] is leveraged in CorpNet to diminish the computational overhead. In other words, for one specific point, the attention is calculated from several adjacent points around it rather than all the points.

Let $\mathbf{x}_i, \mathbf{p}_i$ denote the feature, location of the i-th point, respectively. To illustrate the commonalities of SA and CA modules, we omit the superscript s/t of the search/template region here. A SA/CA module (Fig. 3(a)) consists of a linear layer f_1, a SA/CA block, another linear layer f_2 and a residual connection. Formally,

$$\begin{aligned} \bar{\mathbf{x}}_i &= f_1(\mathbf{x}_i), \\ \hat{\mathbf{x}}_i &= \texttt{SA/CA}(\bar{\mathbf{x}}_i), \\ \mathbf{y}_i &= f_2(\hat{\mathbf{x}}_i) + \mathbf{x}_i. \end{aligned} \tag{1}$$

In the SA/CA block (Fig. 3(b/c)), we obtain the query, key and value $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ features as:

$$\begin{aligned} \mathbf{q}_i &= f_q(\bar{\mathbf{x}}_i), \\ \mathbf{k}_i &= f_k(\texttt{kNN}(\mathbf{p}_i), \bar{\mathbf{x}}_i), \\ \mathbf{v}_i &= f_v(\texttt{kNN}(\mathbf{p}_i), \bar{\mathbf{x}}_i) \end{aligned} \tag{2}$$

where f_q, f_k, f_v are linear projections and $\texttt{kNN}$ denotes the k nearest neighbors. The position encoding is defined as:

$$\mathbf{e}_{ij} = f_e(\mathbf{p}_i - \mathbf{p}_j) \tag{3}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the 3D position for points i and j. f_e is a linear projection. Then the core local attention mechanism could be formulated as:

$$\hat{\mathbf{x}}_i = \sum_{j=1}^{k} \sigma\left(\frac{1}{\sqrt{k}} f_a(\mathbf{q}_i - \mathbf{k}_{ij} + \mathbf{e}_{ij})\right) \odot (\mathbf{v}_{ij} + \mathbf{e}_{ij}) \tag{4}$$

where k stands for the k nearest neighbors, f_a is a two-layer MLPs, $\odot$ represents the element-wise multiplication and σ is a softmax function. As illustrated in Fig. 3(b), in SA module, both $\texttt{kNN}$ and $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ are calculated from a same source (the template or search region), and the output of the SA block of Eq. 1 is modified to:

$$\begin{aligned} \mathbf{y}_i^s &= f_2(\hat{\mathbf{x}}_i^s) + \mathbf{x}_i^s, \\ \mathbf{y}_i^t &= f_2(\hat{\mathbf{x}}_i^t) + \mathbf{x}_i^t \end{aligned} \tag{5}$$

where s stands for search region and t is the template region. While in the CA module (Fig. 3(c)), $\texttt{kNN}$ is calculated with different sources. Also, $\mathbf{q}_i$ and $(\mathbf{k}_i, \mathbf{v}_i)$ are calculated crossly to interact between the template and the search regions and absorb useful target-related features. More specifically, we have:

$$\begin{aligned} \mathbf{y}_i^s &= f_2(\hat{\mathbf{x}}_i^{st}) + \mathbf{x}_i^s, \\ \mathbf{y}_i^t &= f_2(\hat{\mathbf{x}}_i^{ts}) + \mathbf{x}_i^t. \end{aligned} \tag{6}$$

From Eq. 5 and Eq. 6, it can be seen that the original features are aggregated with the corresponding transformed/attended one in SA and sufficiently interacted with each other source in CA. By stacking multiple stages, the features of both the template and search regions are gradually concentrated on the beneficial target-relevant features.

3.2.2 Lateral Correlation Pyramid

In the traditional *extractor-matcher-decoder* paradigm, researchers always overlook the importance of the extractor,

which always has several downsampling operations [23,24] that are used to reduce the model size but exacerbate the sparsity problem. Besides, due to the separation of the extractor and matcher, using the highest-level to do the matching operation in the matcher is a default configuration [9,16,25,36,38], which hinders useful multi-scale combination. Differently, we break the above limitations and try to explore a pyramid structure to equip our CorpNet with rich semantics at all feature levels, as shown in Fig. 4.

The feature pyramid is exploited in 2D detection [19] but remains unexplored in 3D SOT. In particular, to compensate for the exacerbated sparsity caused by the downsampling of the set abstraction module and keep as many points as possible, we formulate a correlation pyramidal encoder by leveraging the encoder's pyramidal feature hierarchy, which has semantics from low to high levels. The inherent multi-scale correlated features output from the CA modules of all stages are fused together. Note that the point amount of these features is different due to the downsampling of the set abstraction modules. Also, the feature dimension and the semantic level are different. To this end, for the feature of each stage $\mathbf{F}^k \in \mathbb{R}^{N_k \times C_k}$, $k \in \{1,2,3\}$, we first unify the feature dimension by a 1D convolution block which consists of a 1D convolution layer, a BN layer, a ReLU activation, and another 1D convolution layer. Then the obtained features are concatenated with the last features of the pyramid along the point amount dimension. The output features of the lateral pyramid $\mathbf{F} \in \mathbb{R}^{(N_1+N_2+N_3)\times C}$ will then be voxelized as a volumetric representation $\mathbf{F}^m \in \mathbb{R}^{C\times H\times L\times W}$ by averaging the 3D coordinates and features of the points in the same voxel bin. Given the voxel size (v_x, v_y, v_z) and the range of the search region $[(x_{min}, x_{max}), (y_{min}, y_{max}), (z_{min}, z_{max})]$, the resolution (W, L, H) of the $\mathbf{F}^m$ is:

$$\begin{aligned} W &= \lfloor \frac{x_{max} - x_{min}}{v_x} \rfloor + 1, \\ L &= \lfloor \frac{y_{max} - y_{min}}{v_y} \rfloor + 1, \\ H &= \lfloor \frac{z_{max} - z_{min}}{v_z} \rfloor + 1 \end{aligned} \tag{7}$$

where $\lfloor \cdot \rfloor$ indicates the floor operation. Since only the search region representations are fed into the decoder for prediction in our CorpNet, thus the lateral correlation pyramid serves only for the multi-level embeddings of the search region.

3.2.3 Decoder

Since the obtained encoded feature $\mathbf{F}^m \in \mathbb{R}^{C\times H\times L\times W}$ is a voxel representation, we can take advantage of regular convolutions that are commonly used in the images. The most similar decoder is from [16], which first uses 3D convolutions on their encoded features and pools a BEV feature map to regress the final results, including the z-axis location.

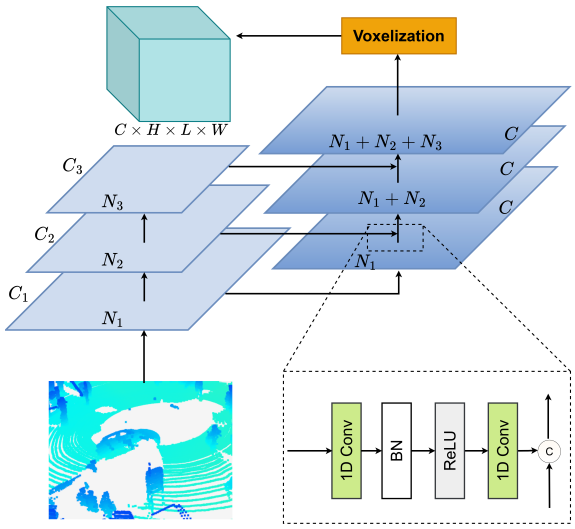

Figure 4. The structure of the lateral correlation pyramid. The left feature maps are obtained by the three stages of the main branch in the encoder. On the right is the lateral correlation pyramid, which combines every correlated feature, feeding through a 1D convolution block and then merging with the features of the last level by concatenation.

In fact, the motion pattern of the x-y plane and the movement in z axis are not exactly the same. Considering this, we decide to separately model them in our decoder as presented in the right part of Fig. 2. First, we disentangle the 3D convolution block into a 2D convolution block (consisting of a 2D convolution layer, a batch normalization layer and a ReLU activation layer) and a 1D convolution block (consisting of a 1D convolution layer, a batch normalization layer, and a ReLU activation layer), operating successively on the BEV and the vertical direction. We can further gain another advantage that is an additional nonlinear rectification between these two operations. This effectively doubles the number of nonlinearities compared to a single 3D convolutions block, thus rendering the model capable of representing more complex functions. After stacking several decomposed 3D convolution blocks, we apply two max pooling operations on the z axis and x-y plane to get the BEV features $\mathbf{F}^{BEV}$ and vertical features $\mathbf{F}^z$, respectively. Mathematically,

$$\begin{aligned} \mathbf{F}^{BEV} &= \texttt{MaxPool}_z \phi(\mathbf{F}^m), \\ \mathbf{F}^z &= \texttt{MaxPool}_{xy} \phi(\mathbf{F}^m) \end{aligned} \tag{8}$$

where ϕ is the decomposed 3D convolution blocks. The subscript of the `MaxPool` means applying max pooling on the corresponding dimension.

Next, different to [16, 17], we employ two separate prediction subnetworks on top of the $\mathbf{F}^{BEV}$ and $\mathbf{F}^z$. Each subnetwork applies a stack of common convolutions blocks

(2D convolution for BEV feature maps and 1D convolution for vertical features) on the dense feature map ($\mathbf{F}^{BEV}$ and $\mathbf{F}^{z}$) to strengthen and adjust the features for sufficient local information in the corresponding feature maps. Then, for each subnetwork, a center classification head is attached to predict the discrete object center. To compensate for the discretization error, we regress the offset of the continuous ground truth center with an offset regression head. The rotation regression is predicted together with the BEV offset regression head. The final training objective is expressed as:

$$\mathcal{L} = \lambda_{cls}(\mathcal{L}_{cls}^{BEV} + \mathcal{L}_{cls}^{z}) + \lambda_{reg}(\mathcal{L}_{reg}^{BEV} + \mathcal{L}_{reg}^{z}) \quad (9)$$

where $\mathcal{L}_{cls}^{*}$ is the focal loss [20] with default hyper-parameters for center classification heads and $\mathcal{L}_{reg}^{*}$ is the L1 loss for the BBox center offset and rotation regression heads.

Ground Truth Construction. Here we take the ground truth construction of BEV heads for instance, and the ground truth of the z-axis heads can be similarly formulated. Let (x, y, z) denote the 3D target location, the target 2D center (c_x, c_y) in the BEV is computed as:

$$\begin{aligned} c_x &= \frac{x - x_{min}}{v_x}, \\ c_y &= \frac{y - y_{min}}{v_y}. \end{aligned} \quad (10)$$

The discrete 2D center is defined by $\hat{c_x} = \lfloor c_x \rfloor$ and $\hat{c_y} = \lfloor c_y \rfloor$. The ground truth of the BEV center classification is $\mathbf{y}_{cls} \in \mathbb{R}^{L \times W}$, where the value in location (i, j) becomes 1 if $i = \hat{c_x}$ and $j = \hat{c_y}$, 0 if (i, j) not in the 2D target BBox, otherwise $\frac{1}{\gamma+1}$, where γ represents the Euclidean distance between the pixel (i, j) and the discrete target center.

4. Experiment

4.1. Experiment Setups

Implementation details. We set both N_t and N_s to 1024 for the input template and search regions by randomly duplicating and discarding points. In the encoder, the set abstraction modules from PointNet++ [24] are simplified into 2 MLP layers to decrease parameters, diminishing the input points to 512, 256, 128 ones, respectively. The radius of these layers is set to 0.3, 0.5, and 0.7 meters by default. Parameters of the set abstraction modules and SA modules are shared for the template and search regions. The correlated features after the lateral correlation pyramid are all with 64 channels. In the voxelization process, we set the region $[(x_{min}, x_{max}), (y_{min}, y_{max}), (z_{min}, z_{max})]$ to [(-5.6,5.6),(-3.6,3.6),(-2.4,2.4)] to cover most target points. The voxel size (v_x, v_y, v_z) is set to (0.3,0.3,0.3). For the decoder, three decomposed 3D blocks are stacked before the pooling operations of Eq. 8. The two subnetworks include three 2D convolution blocks and three 1D convolution blocks for feature aggregation, respectively. The classification loss has a weight λ_{cls} of 1 and the regression loss has a weight λ_{reg} of 1. The radius r is set to 2. For all experiments, we use the Adam optimizer with an initial learning rate of 0.001 for training, and the learning rate decays by 0.2 every six epochs by default. It takes about 20 epochs to train our model. In the inference, CorpNet runs at 36 FPS.

Training and Testing. In the training stage, we use the points chosen from the ground truth BBox in the first frame and the ground truth BBox dealt by a random shift from the last frame as the template. The search region is generated with ground truth BBox enlarged by 2 meters plus with the random shift. In the testing stage, we use the points inside the BBox of the first frame and the last predicted BBox as the template. The search region is generated by the last predicted BBox enlarged by 2 meters.

4.2. Comparison with State-of-the-Art Trackers

4.2.1 Comparison on KITTI

KITTI [10] contains 21 training sequences and 29 test sequences. We follow previous works [11, 36, 37] to split the training set into train/val/test splits due to the inaccessibility of the test labels. We use scenes 0-16 for training, scenes 17-18 for validation, and scenes 19-20 for testing.

We compare the proposed CorpNet with current state-of-the-art methods, from the pioneering SC3D [11] to the most recent STNet [17]. As shown in Tab. 1, our CorpNet performs significantly better than other methods on the mean results of four categories. We yield the best results on most categories. We find that our sufficient self aggregations and cross interactions, as well as the correlation pyramid in the encoder, make our CorpNet learn effectively on the data-rare categories like Cyclist and Van. While most previous methods like V2B [16], BAT [36] and PTT [26] are hard to handle these classes. The second best M2Track [37] also proposes to change the Siamese-like pipeline by constructing a spatial-temporal point cloud to avoid the Siamese-like backbone and predicts the motion directly. Nevertheless, they still need a motion transformation module to integrate the template information into the search representations, and a two-stage refinement is used to improve the performance. As a result, we achieve better performance than M2Track with a single stage. Besides, compared with V2B [16] which is the decoder baseline of our method, our proposed encoder and the improved decoder make CorpNet yield better performance in all categories.

4.2.2 Experiments on NuScenes

NuScenes [2] has 1000 scenes, which are divided into 700/150/150 scenes for train/val/test. Officially, the train

Table 1. Comparison among our CorpNet and the state-of-the-art methods on the **KITTI** datasets. Mean shows the average result weighed by frame numbers. **Bold** and underline denote the best performance and the second-best performance, respectively.

Methods	Car (6424)		Cyclist (308)		Van (1248)		Pedestrian (6088)		Mean (14068)	
	Success	Precision	Success	Precision	Success	Precision	Success	Precision	Success	Precision
SC3D [11]	41.3	57.9	41.5	70.4	40.4	47.0	18.2	37.8	31.2	48.5
SC3D-RPN [32]	36.3	51.0	43.0	81.4	-	-	17.9	47.8	-	-
P2B [25]	56.2	72.8	32.1	44.7	40.8	48.4	28.7	49.6	42.4	60.0
MLVSNet [27]	56.0	74.0	34.3	44.5	52.0	61.4	34.1	61.1	45.7	66.6
3DSiamRPN [9]	58.2	76.2	36.1	49.0	45.6	52.8	35.2	56.2	46.6	64.9
LTTR [5]	65.0	77.1	66.2	89.9	35.8	45.6	33.2	56.8	48.7	65.8
PTT [26]	67.8	81.8	37.2	47.3	43.6	52.5	44.9	72.0	55.1	74.2
BAT [36]	60.5	77.7	33.7	45.4	52.4	67.0	42.1	70.1	51.2	72.8
V2B [16]	70.5	81.3	40.8	49.7	50.1	58.0	48.3	73.5	58.4	75.2
PTTR [38]	65.2	77.4	65.1	90.5	52.5	61.8	50.9	81.6	58.4	77.8
STNet [17]	72.1	84.0	73.5	93.7	58.0	70.6	49.9	77.2	61.3	80.1
M2Track [37]	65.5	80.8	73.2	93.5	53.8	**70.7**	**61.5**	**88.2**	62.9	**83.4**
CorpNet	**73.6**	**84.1**	**74.3**	**94.2**	**58.7**	66.5	55.6	82.4	**64.5**	82.0

Table 2. Comparison among our CorpNet and the state-of-the-art methods on the **NuScenes** datasets. Mean shows the average result weighed by frame numbers. **Bold** and underline denote the best performance and the second-best performance, respectively.

Methods	Car (15578)		Bicycle (501)		Truck (3710)		Pedestrian (8019)		Mean (27808)	
	Success	Precision	Success	Precision	Success	Precision	Success	Precision	Success	Precision
SC3D [11]	24.5	25.9	16.6	18.8	32.5	30.6	13.8	14.7	22.3	23.2
P2B [25]	32.8	35.2	19.7	26.6	16.2	11.1	19.2	26.6	26.4	29.3
BAT [36]	26.5	28.8	17.8	22.8	16.5	10.6	19.4	28.2	23.0	27.9
V2B [16]	32.9	34.5	20.3	27.5	28.7	23.8	20.1	27.4	28.4	30.9
STNet [17]	35.7	37.2	22.3	29.3	33.5	32.4	20.1	27.8	30.7	33.7
CorpNet	35.0	**38.4**	**26.9**	**43.5**	**39.7**	**36.3**	**21.3**	**33.6**	**31.8**	**36.8**

set is further evenly split into "train track" and "train detect" to remedy overfitting. Following [16], we train our model with "train track" split and test it on the val set.

Note that the NuScenes dataset only labels keyframes and provides official interpolated results for the remaining frames, so there are two configurations for this dataset. The first is from [36] which trains and tests both only on the keyframes. The other one is from [16] which trains and tests on all the frames. These two configurations result in different datasets with different performances. We believe that the motion in key frames is extremely large, which is not in line with the practical applications. Therefore, we follow the second set in this paper. The methods that follow the second way are compared, and the performance evaluated on the keyframes is reported.

In Tab. 2, we report the comparison results on NuScenes. The proposed CorpNet exceeds all the competitors under all categories, mostly by large margins. NuScenes is much more challenging than KITTI due to the label scarcity, pervasive distractors, and drastic appearance changes. CorpNet still surpasses other methods on both rigid (e.g., Car) and non-rigid (e.g., Pedestrian) objects, both small (Bicycle) and large (Truck) objects. Besides, our method significantly improves categories (i,e, Bicycle and Truck) with fewer data. The thorough cross-source interactions and the lateral pyramid helps to learn useful features even though the annotation may be incorrect (interpolated labels) on this dataset.

4.3. Ablation Study

We comprehensively perform ablation studies to analyze each component in our proposed CorpNet on the Car category of the KITTI dataset.

Location of SA modules and CA modules. We study the influence of the SA and CA modules separately in Tab. 3. For both SA and CA, we find that merely adding one module to the last stage has already exceeded all the previous methods. When placing SA and CA on all the three stages, we could obtain the best results. Therefore, we use this configuration in our final CorpNet. These results are intuitive since the representation could be enriched more if the features of all levels are self-aggregated by SA modules. Similarly, the template information could be sufficiently interacted and enhanced if the correlation operations existed in all the stages, yielding better results.

Number of neighbors in the local attention. In Tab. 4,

we investigate the influence of the number of neighbors k, which determines the considered local neighborhood around each point. We use the same setting for SA and CA modules. In our CorpNet, the best results are obtained when k=32. When k is smaller, like 8 and 16, the model will have insufficient context for prediction. More specifically, SA may not have enough information to strengthen its features, and CA can not interact with sufficient related points. When k is larger, like 64, the performance will be deteriorated by excessive noise, which is farther and less relevant. The best choice of k is 32 in CorpNet.

How important are the correlation pyramid? We conduct an ablation study about the correlation pyramid in Tab. 5. When only using the correlated features of stage 3 without the pyramid structure, the performance drops significantly (2.7%) compared with the final CorpNet. When fusing the features of stage 2 and stage 3, the results are still worse than fusing all the three stages (-1.0%). This demonstrates that using the correlation from all stages for the proposed correlation pyramid is important and beneficial to the model's accuracy.

Table 3. Location of SA and CA modules.

Stage 1	Stage 2	Stage 3	SA	CA
		✓	71.2/81.9	70.8/81.9
	✓	✓	71.3/82.8	72.8/83.9
✓	✓	✓	**73.6/84.1**	**73.6/84.1**

Table 4. Number of neighbors k.

k	Success	Precision
8	69.9	81.8
16	71.4	81.4
32	**73.6**	**84.1**
64	72.1	83.6

Table 5. Influence of the pyramid structure.

Stage 1	Stage 2	Staget 3	Success	Precision
		✓	70.9	81.5
	✓	✓	72.6	82.9
✓	✓	✓	**73.6**	**84.1**

Is the z-axis separation beneficial? We now study our contribution in the decoder in Tab. 6, where 3D Conv is short for 3D We now study our contribution in the decoder in Tab. 6, where 3D Conv is short for 3D convolution blocks, BEV stands for the 2D prediction head operating on the BEV feature maps, Decomposed 3D Conv represents our proposed decomposed 3D blocks and z denotes our standalone prediction head for the z-axis features. "3D Conv + BEV" is actually the original decoder implement of V2B [16]. The results show that 1) directly using V2B's decoder improves 1.8% (success) over V2B, which validates the effectiveness of our proposed encoder, 2) only factorizing the 3D convolution block but using only the BEV head is not working, and we attribute to the fact that decomposed 3D convolution first explicitly embeds the z-axis information but then pools it will damage the decomposed features and lead to failures. 3) factorizing the 3D convolution block and adding a separate z-axis prediction head together is beneficial and further improve the performance with 1.3% in success, which verifies our initial consideration that the motion pattern of the x-y plane and the movement in z axis are not exactly the same.

Table 6. z-axis separation in the decoder.

Configurations	Success	Precision
3D Conv + BEV	72.3	81.7
3D Conv + BEV+z	70.6	82.1
Decomposed 3D Conv + BEV	59.8	76.1
Decomposed 3D Conv + BEV+z	**73.6**	**84.1**

4.4. Computational Cost

We analyze the computational cost in Tab. 7. The results are tested under the same PyTorch platform and a single TITAN RTX GPU on the Car category of KITTI. We can see that the speed of our method is real-time and comparable with V2B, but our CorpNet performs better than V2B with 3.1% in success. Although BAT and P2B infer faster than our CorpNet, our performance is significantly better than them (+17.4% and +13.1% in success). For the training time, our CorpNet, V2B and BAT all converge fast with about 8 hours, while SC3D and P2B require more time. Besides, the parameters and FLOPs of CorpNet are also comparable with other methods.

Table 7. The Computational cost of different trackers.

Method	Parameters	FLOPs	FPS	Training time	Success
SC3D [11]	6.45 M	20.07 G	6	∼13 h	41.3
P2B [25]	1.34 M	4.28 G	48	∼13 h	56.2
BAT [36]	1.47 M	5.53 G	54	∼8 h	60.5
V2B [16]	1.35 M	5.47 G	39	∼8 h	70.5
CorpNet	1.95 M	7.02 G	36	∼8 h	**73.6**

5. Conclusion

This paper proposes a novel correlation pyramid network (CorpNet) for the 3D single object tracking task, which merges the extractor and matcher of the traditional pipeline to jointly learn the target-aware representation, enabling the extractor and matcher to benefit each other. Particularly, CorpNet integrates multi-level self attentions and cross attentions to enrich the template and search region features, and sufficiently realize their fusion and interaction, respectively. Furthermore, a lateral correlation pyramid structure is designed in the encoder to handle the sparsity problem from the downsampling operations by combining the hierarchical correlated features of all the points that existed in the encoder. The decoder of our method has a new z-axis-separated structure, which explicitly learns the movement of the z axis and the x-y plane. Finally, our method achieves state-of-the-art results on two commonly-used datasets (KITTI and NuScenes).

References

[1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[3] Boyu Chen, Peixia Li, Lei Bai, Lei Qiao, Qiuhong Shen, Bo Li, Weihao Gan, Wei Wu, and Wanli Ouyang. Backbone is all your need: A simplified architecture for visual object tracking. *arXiv preprint arXiv:2203.05328*, 2022.

[4] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. Robust 3d object tracking from monocular images using stable parts. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1465–1479, 2017.

[5] Yubo Cui, Zheng Fang, Jiayao Shan, Zuoxu Gu, and Sifan Zhou. 3d object tracking with transformer. *arXiv preprint arXiv:2110.14921*, 2021.

[6] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13608–13618, 2022.

[7] Xingping Dong, Jianbing Shen, Dongming Wu, Kan Guo, Xiaogang Jin, and Fatih Porikli. Quadruplet network with one-shot learning for fast visual object tracking. *IEEE Transactions on Image Processing*, 28(7):3516–3527, 2019.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Zheng Fang, Sifan Zhou, Yubo Cui, and Sebastian Scherer. 3d-siamrpn: an end-to-end learning method for real-time 3d single object tracking using raw point cloud. *IEEE Sensors Journal*, 21(4):4995–5011, 2020.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[11] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completion for 3d siamese tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1359–1368, 2019.

[12] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

[13] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[14] Hou-Ning Hu, Yung-Hsu Yang, Tobias Fischer, Trevor Darrell, Fisher Yu, and Min Sun. Monocular quasi-dense 3d object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[15] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[16] Le Hui, Lingpeng Wang, Mingmei Cheng, Jin Xie, and Jian Yang. 3d siamese voxel-to-bev tracker for sparse point clouds. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] Le Hui, Lingpeng Wang, Linghua Tang, Kaihao Lan, Jin Xie, and Jian Yang. 3d siamese transformer network for single object tracking on point clouds. *arXiv preprint arXiv:2207.11995*, 2022.

[18] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[21] Xiankai Lu, Chao Ma, Jianbing Shen, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep object tracking with shrinkage loss. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[22] Jonah Ong, Ba-Tuong Vo, Ba-Ngu Vo, Du Yong Kim, and Sven Nordholm. A bayesian filter for multi-view 3d multi-object tracking with occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[25] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6329–6338, 2020.

[26] Jiayao Shan, Sifan Zhou, Zheng Fang, and Yubo Cui. Ptt: Point-track-transformer module for 3d single object tracking in point clouds. In *2021 IEEE/RSJ International Conference*

on Intelligent Robots and Systems (IROS), pages 1310–1316. IEEE, 2021.

[27] Zhoutao Wang, Qian Xie, Yu-Kun Lai, Jing Wu, Kun Long, and Jun Wang. Mlvsnet: Multi-level voting siamese network for 3d visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3101–3110, 2021.

[28] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[29] Fei Xie, Chunyu Wang, Guangting Wang, Yue Cao, Wankou Yang, and Wenjun Zeng. Correlation-aware deep tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8751–8760, 2022.

[30] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, pages 12549–12556, 2020.

[31] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *European Conference on Computer Vision*, pages 341–357. Springer, 2022.

[32] Jesus Zarzar, Silvio Giancola, and Bernard Ghanem. Efficient bird eye view proposals for 3d siamese tracking. *arXiv preprint arXiv:1903.10168*, 2019.

[33] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenyu Liu, and Wenjun Zeng. Voxeltrack: Multi-person 3d human pose estimation and tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[34] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4591–4600, 2019.

[35] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

[36] Chaoda Zheng, Xu Yan, Jiantao Gao, Weibing Zhao, Wei Zhang, Zhen Li, and Shuguang Cui. Box-aware feature enhancement for single object tracking on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13199–13208, 2021.

[37] Chaoda Zheng, Xu Yan, Haiming Zhang, Baoyuan Wang, Shenghui Cheng, Shuguang Cui, and Zhen Li. Beyond 3d siamese tracking: A motion-centric paradigm for 3d single object tracking in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8111–8120, 2022.

[38] Changqing Zhou, Zhipeng Luo, Yueru Luo, Tianrui Liu, Liang Pan, Zhongang Cai, Haiyu Zhao, and Shijian Lu. Pttr: Relational 3d point cloud object tracking with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8531–8540, 2022.