3D Siamese Voxel-to-BEV Tracker for Sparse Point Clouds

Anonymous Author(s)
Affiliation
Address
email

Abstract

3D object tracking in point clouds is still a challenging problem due to the sparsity of LiDAR points in dynamic environments. In this work, we propose a Siamese voxel-to-BEV tracker, which can significantly improve the tracking performance in sparse 3D point clouds. Specifically, it consists of a Siamese shape-aware feature learning network and a voxel-to-BEV target localization network. The Siamese shape-aware feature learning network can capture 3D shape information of the object to learn the discriminative features of the object so that the potential target from the background in sparse point clouds can be identified. To this end, we first perform template feature embedding to embed the template's feature into the potential target and then generate a dense 3D shape to characterize the shape information of the potential target. For localizing the tracked target, the voxel-to-BEV target localization network regresses the target's 2D center and the z-axis center from the dense bird's eye view (BEV) feature map in an anchor-free manner. Concretely, we compress the voxelized point cloud along z-axis through max pooling to obtain a dense BEV feature map, where the regression of the 2D center and the z-axis center can be performed more effectively. Extensive evaluation on the KITTI tracking dataset shows that our method significantly outperforms the current state-of-the-art methods by a large margin.

1 Introduction

Object tracking is an essential task in computer vision and has been widely in various applications, such as autonomous vehicle, mobile robotics, and augmented reality. In the past few years, many efforts [1, 2, 3, 4] have been made on 2D object tracking from RGB data. Recently, with the development of 3D sensor such as LiDAR and Kinect, 3D object tracking [5, 6, 7, 8, 9] has attracted more attention. Lately, some pioneer works [10, 11, 12] have focused on point cloud based 3D object tracking. However, due to the sparsity of 3D point clouds, 3D object tracking on point clouds is still a challenging task.

Few works are dedicated to 3D object tracking with only point clouds. As a pioneer work, SC3D [13] is the first 3D Siamese tracker that performs matching between the template and candidate 3D target proposals generated by Kalman filtering [10]. Furthermore, a shape completion network is used to enhance shape information of candidate proposals in sparse point clouds, thereby improving the accuracy of matching. However, SC3D cannot perform the end-to-end training, and consumes much time when matching exhaustive candidate proposals. Towards these concerns, Qi *et al.* [12] proposed an end-to-end framework termed P2B, which first localizes potential target centers in the search area via Hough voting [14], and then aggregates vote clusters to generate target proposals. Nonetheless, when facing sparse scenes, P2B may not be able to track the object accurately, or even lose the tracked object. On the one hand, it adopts random sampling to generate initial seed points, which further exacerbates the sparsity of point clouds. On the other hand, it is difficult to generate high-quality

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

target proposals on sparse 3D point clouds. Although SC3D has enhanced shape information of candidate proposals, the poor-quality candidate proposals obtained from sparse point clouds still degrade tacking performance.

As shown in Fig. 1, we count the number of points on KITTI's cars. It can be found that 51% of cars have less than 100 points, and only 7% of cars have more than 2500 points. When facing sparse point clouds, it is difficult to distinguish the target from the background due to the sparsity of point clouds. Therefore, how to improve the tracking performance in sparse scenes should be considered. Our intuition consists of two folds. First, enhancing shape information of target will provide discriminative information to distinguish the target from the background, especially in sparse point clouds. Second, due to the sparsity of the point cloud, it is difficult to regress the target center in 3D space. We hence consider compressing the sparse 3D space into a dense 2D space, and perform center regression in the dense 2D space to improve tracking performance.

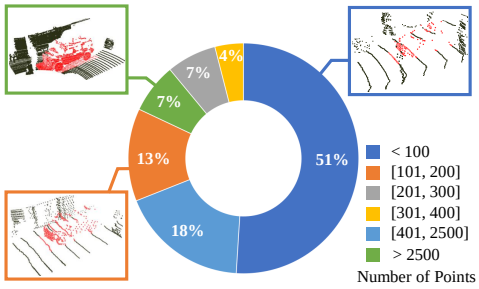


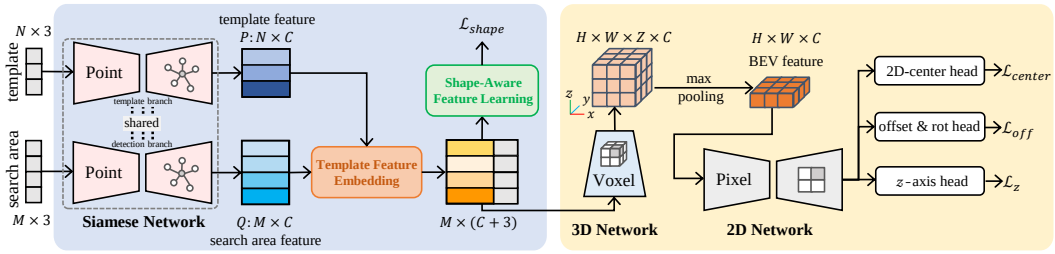
Figure 1: Statistics of the number of points on KITTI's cars. Cars are colored in red.

In this paper, we propose a novel Siamese voxel-to-BEV tracker, which aims to improve the tracking performance in sparse point clouds. We illustrate our framework in Fig. 2. We first feed the template and search area into the Siamese network to extract point features, respectively. Then, we perform template feature embedding to embed the template's feature into the potential target in the search area by adaptively learning the correlation between the template and search area. In this way, we can strengthen relationship between the template and search area in order to better localize the potential target. After that, we implement shape-aware feature learning to enhance the shape information of the potential target to improve the discrimination of point features. Here, we exploit a generation network to characterize the shape information by generating a dense point cloud from a sparse point cloud. For localizing the 3D target, we develop a voxel-to-BEV target localization method, which can improve the accuracy of target center detection in sparse point clouds. In order to reduce the difficulty of target center detection in sparse 3D space, we decompose the 3D center into a 2D center and a z-axis center, and regress them in a dense bird's eye view (BEV) space, where each item can be regressed more effectively and accurately.

In summary, we propose a novel Siamese voxel-to-BEV tracker (V2B-Tracker), which can significantly improve tracking performance, especially in sparse point clouds. We develop a Siamese shape-aware feature learning network that can introduce shape information to enhance the discrimination of the potential target in the search area. We develop a voxel-to-BEV target localization network, which can accurately detect the 3D target's center in the dense BEV space compared to sparse 3D space. Extensive results show that our method has achieved new state-of-the-art results on the KITTI tracking dataset [15].

2 Related Work

2D object tracking. Numerous schemes [16, 10, 17, 18, 19] have been presented and achieved impressive results in 2D object tracking. Early works are mainly based on correlation filtering. As a pioneer work, MOSSE [20] presents stable correlation filters for visual tracking. After that, correlation-based methods use Circulant matrices [21], kernelized correlation filters [22], continuous convolution filters [23], factorized convolution operators [3] to improve tracking performance. In recent years, Siamese-based methods [24, 25] have been more popular in the tracking field. In [2], Bertinetto *et al.* proposed SiamFC, a pioneering work that combines naive feature correlation with a fully-convolutional Siamese network for object tracking. Subsequently, some improvements [26, 27, 28, 29, 30] are made to Siamese trackers, such as combining with a region proposal network [31, 32, 33, 34] or an anchor-free FCOS detector [35], using a deeper architecture [36] or two-branch structure [37], exploiting attention [38, 39] or self-attention [40], applying triplet loss [41]. However, these methods are specially designed for 2D object tracking, so they cannot be directly applied to 3D point clouds. Inspired by [2, 31], we develop a Siamese tracking-by-detection framework for 3D object tracking task.



Siamese Shape-Aware Feature Learning Network **Voxel-to-BEV Target Localization Network**

Figure 2: The architecture of V2B-Tracker. Given a template and search area, we first use Siamese network to obtain template and search area features. We then perform template feature embedding and shape-aware feature learning to enhance the ability to distinguish the target from the background. Finally, we perform voxel-to-BEV target localization to detect the 3D object center from the BEV.

3D object tracking. Different from 2D object tracking, 3D object tracking leverages the geometric information in the 3D space to track objects instead of relying on the 2D appearance properties of the objects. As a pioneer work, Song *et al.* [42] first proposed a unified 100 RGB-D video dataset, which opened up a new research direction for RGB-D tracking [43, 44, 45]. Based on RGB-D information, 3D object tracking methods [46, 47, 7] usually combines techniques from the 2D tracking with additional depth information. However, RGB-D tracking also relies on RGB information, and it may fail when the RGB information is degraded. Recent efforts [48, 49] begin to use LiDAR point clouds for 3D object tracking. Among them, SC3D [13] is the first 3D Siamese tracker for point clouds. Following it, Re-Track [11] is a two-stage framework that re-tracks the lost objects of the coarse stage in the fine stage. Lately, Qi *et al.* [12] proposed P2B, which solves the problem that SC3D cannot perform end-to-end training and consumes a lot of time. P2B adopts the tracking-by-detection scheme, which uses VoteNet [14] to generate proposals and selects the proposal with the highest score as the target. However, when facing very sparse scenarios, it may be difficult to generate high-quality proposals, thereby degrading tracking performance.

Deep learning on point clouds. With the introduction of PointNet [50], 3D deep learning on point clouds has stimulated the interest of researchers. Existing methods can be mainly divided into: point-based [51, 52, 53], volumetric-based [54, 55], graph-based [56, 57, 58, 59], and view-based [60, 61, 62] methods. However, volumetric-based and view-based methods lose fine-grained geometric information due to voxelization and projection, while graph-based methods take much time due to graph construction. As a result, the 3D tracking networks in [13, 11, 12] are point-based methods. But they may fail when the point cloud is very sparse or there are few points on the target. This motivates us to focus on improving tracking performance under sparse point clouds.

3 Method

Our work is specifically designed for 3D object tracking in sparse point clouds. We first present the Siamese shape-aware feature learning network to enhance the discrimination of the potential target in search area (Sec. 3.1). We then localize the target by voxel-to-BEV target localization network (Sec. 3.2).

3.1 Siamese Shape-Aware Feature Learning Network

3.1.1 Template Feature Embedding

Here, we perform template feature embedding to establish a correlation between the template and search area in order to better localize the potential target in the search area. Suppose the size of the template is N, and the size of the search area is M (generally, $M > N$). We first use the Siamese network to extract point features, denoted by $P = \{\boldsymbol{p}_i\}_{i=1}^{N}$ and $Q = \{\boldsymbol{q}_j\}_{j=1}^{M}$, respectively. The Siamese network consists of template branch and detection branch. The two branches both adopt PointNet++ [51] as backbone and share parameters. In this way, two branches encoded by the same network will contribute to subsequent object tracking. Then, we develop a network to adaptively learn the correlation between the template and search area to embed template's feature into the potential target in the search area.

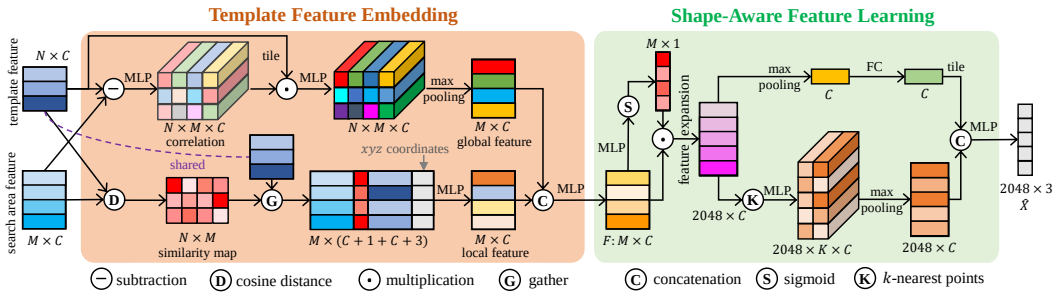


Figure 3: The architecture of template feature embedding and shape-aware feature learning.

Template global feature embedding. Since the search includes some noisy points ($i.e.$, background), it is desired that the potential target in the search area can obtain template's global information, while the rest cannot. Since the Multi-Layer Perceptron (MLP) network introduced in [50, 51] is used to approximate the general function on the point cloud, we hence use MLP to adaptively learn the correlation between the template and search area. As shown in the left half of Fig. 3, the correlation is formulated as:

$$\boldsymbol{w}_{ij} = f_{corr}(\boldsymbol{p}_i, \boldsymbol{q}_j) = \mathrm{MLP}(\boldsymbol{p}_i - \boldsymbol{q}_j), \forall \boldsymbol{p}_i \in P, \boldsymbol{q}_j \in Q \tag{1}$$

where $\boldsymbol{p}_i - \boldsymbol{q}_i$ characterizes the difference between the two feature vectors and $\boldsymbol{w}_{ij}$ is the correlation weight between two points. We then embed the template's global feature into $\boldsymbol{q}_j$ in the search area, and the embedded function f_{emb} is written as:

$$\boldsymbol{q}_j^{'} = f_{emb}(\boldsymbol{q}_j, \boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N) = \mathrm{MLP}(\underset{i=1,2,\ldots,N}{\mathrm{MAX}} \{\boldsymbol{p}_i \cdot \boldsymbol{w}_{ij}\}), \forall \boldsymbol{q}_j \in Q \tag{2}$$

where MAX represents the max pooling function and $\boldsymbol{w}_{ij}$ is the correlation weight learned by MLP. The obtained $\boldsymbol{q}_j^{'} \in \mathbb{R}^C$ considers the correlation of all template points, and characterizes the target's global feature through the max pooling function.

Template local feature embedding. We also embed the template's local feature into the potential target in the search area. As shown in the left half of Fig. 3, we first obtain the similarity map between the template and search area by computing the cosine distance between them. The similarity function f_{sim} is formulated as:

$$\boldsymbol{s}_{ij} = f_{sim}(\boldsymbol{p}_i, \boldsymbol{q}_j) = \frac{\boldsymbol{p}_i^\top \cdot \boldsymbol{q}_j}{\|\boldsymbol{p}_i\|_2 \cdot \|\boldsymbol{q}_j\|_2}, \forall \boldsymbol{p}_i \in P, \boldsymbol{q}_j \in Q \tag{3}$$

where $\boldsymbol{s}_{ij}$ indicates the similarity between points i and j. Then, we assign each search area's point with its most similar template's local feature, which is written as:

$$\boldsymbol{q}_j^{''} = \mathrm{MLP}([\boldsymbol{q}_j, \boldsymbol{s}_{kj}, \boldsymbol{p}_k, \boldsymbol{x}_k]), k = \underset{i=1,2,\ldots,N}{\mathrm{argmax}} \{f_{sim}(\boldsymbol{p}_i, \boldsymbol{q}_j)\}, \forall \boldsymbol{q}_j \in Q \tag{4}$$

where k indicates the index of the maximum value of similarity, and $\boldsymbol{s}_{kj}$, $\boldsymbol{x}_k$ are the corresponding maximum value and 3D coordinate, respectively. $[\cdot, \cdot, \cdot, \cdot]$ represents the concatenation operator. We hence obtain the embedded feature $\boldsymbol{q}_j^{''} \in \mathbb{R}^C$ of j-th point in the search area after using MLP. Since we adopt PointNet++ [51] as backbone, $\boldsymbol{p}_k$ preserves local context of the template point cloud. Note that $\boldsymbol{s}_{kj}$ provides the correlation to distinguish whether the point j is similar to the template point i. Therefore, after using MLP, we can embed template's local feature into the potential target in the search area. Finally, we concatenate the obtained global and local feature maps to obtain an enhanced feature map $F = \{\boldsymbol{f}_j\}_{j=1}^{M}, \boldsymbol{f}_j = \mathrm{MLP}[\boldsymbol{q}_j^{'}, \boldsymbol{q}_j^{''}]$.

3.1.2 Shape-Aware Feature Learning

In the sparse search area, 3D object shape information contributes to improve the ability to distinguish the potential target from the background. As a result, after the template feature embedding, we consider generating a dense 3D point cloud to characterize the shape information in order to enhance the discrimination of the potential target.

Dense ground truth processing. To obtain dense 3D point cloud ground truth, we first crop and center points lying inside the target's ground truth bounding box in all frames. We then concatenate all cropped and centered points to generate a dense aligned 3D point cloud, denoted by $X = \{\boldsymbol{x}_i\}_{i=1}^{2048}$, where $\boldsymbol{x}_i$ is the 3D position, and we fix the number of points to 2048 by randomly discarding and duplicating points.

Shape generation for feature learning. As shown in the right half of Fig. 3, we adopt a two-branch shape generation network, which simultaneously considers the global and local information. Suppose the input point feature $F \in \mathbb{R}^{M \times C}$, which embeds template's feature. We first use a gate mechanism to further enhance the potential target and suppress the background, which is formulated as:

$$F^{'} = \sigma(FW^{\top} + b) \circ F \tag{5}$$

where $F^{'} \in \mathbb{R}^{M \times C}$ is the enhanced feature, σ is the *sigmoid* function, $\circ$ is the element-wise product. Besides, $W \in \mathbb{R}^{1 \times C}$ is the weight to be learned. It is expected that the potential target can make a greater contribution to the subsequent shape generation. Then, we execute the feature expansion operation [63] to enlarge the feature map from $M \times C$ to $2048 \times C$, that is, increase the number of points from M to 2048. After that, we use two-branch structure to characterize the shape information of the potential target. One is called the global branch which exploits the max pooling combined with fully connected layers to capture global shape information. The other is called the local branch, which uses EdgeConv [56] to capture local geometric information. After that, we augment each point's local information with a global information, yielding a new feature map of size $2048 \times 2C$. Finally, we adopt MLP to generate 3D coordinates, denoted by $\hat{X} = \{\hat{\boldsymbol{x}}_i\}_{i=1}^{2048}$. To train our shape generation network, we follow [13, 64] and use Chamfer distance (CD) loss to enforce the network to generate a realistic 3D point cloud. The Chamfer distance measures the similarity between the generated point cloud and dense ground truth, which is given by:

$$\mathcal{L}_{shape} = \sum_{\boldsymbol{x}_i \in X} \min_{\hat{\boldsymbol{x}}_j \in \hat{X}} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_j\|_2^2 + \sum_{\hat{\boldsymbol{x}}_j \in \hat{X}} \min_{\boldsymbol{x}_i \in X} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_j\|_2^2 \tag{6}$$

By minimizing the CD loss in training, the generated point cloud can have the shape and local geometric structure similar to the ground truth, so that the shape information can be effectively characterized. Note that the shape generation is only performed during training and will be discarded during testing. Therefore, it does not increase the inference time of object tracking in the test scheme.

It is noted that our shape-aware feature learning is completely different from SC3D [13]. We perform shape-aware feature learning on the entire search area, while SC3D performs shape completion on the template. Besides, we consider the global and local feature of the point cloud, while SC3D only considers the global feature.

3.2 Voxel-to-BEV Target Localization for 3D Object Tracking

Although we have enhanced features of the potential target in the search area, it is still difficult to accurately detect the 3D target center due to the sparsity of 3D point cloud. To further improve the accuracy of target center detection, we disentangle the 3D center into a 2D center and a z-axis center so that each item can be regressed effectively and accurately. We first execute feature aggregation in order to enhance target information, and then detect the 2D center and z-axis center based on bird's eye view (BEV) space.

3.2.1 Voxel-to-BEV Feature Aggregation

As shown in the right half of Fig. 2, we first convert the point cloud into volumetric space. Specifically, we first translate all points into the local coordinate system, and then normalize the points in voxel by averaging the coordinates and features of the points. To enhance target features, we further apply a stack of 3D convolutions to the volumetric space to aggregate the features so that target's voxel can contain sufficient target information. After that, we compress the 3D volumetric representation into the 2D representation by using max pooling along the z-axis, that is, the BEV feature map is obtained. Subsequently, we apply a stack of 2D convolutions to aggregate features so that target's pixel can obtain sufficient target information. Finally, we can obtain a denser BEV feature map that contains rich target information. For more details, please refer to the supplementary material.

3.2.2 Target Localization in BEV

Inspired by [65], we develop a simple yet powerful network to detect the 2D center and the z-axis center based on the obtained dense BEV feature map. As shown in the right half of Fig. 2, it consists of three heads: 2D-center head, offset & rotation head, and z-axis head. The 2D-center head aims to localize 2D center of target on the x-y plane, and z-axis head regresses the target center of the z-axis. Since the 2D center of 2D grid is discrete, we also regress the offset between it and the continuous center. Thus, we use a offset & rotation head to regress offset plus additional rotation.

Target center parameterization. Given the voxel size v and the range of the search area $[(x_{min}, x_{max}), (y_{min}, y_{max})]$ in x-y plane, we can obtain the resolution of the BEV feature map by $H = \lfloor \frac{x_{max}-x_{min}}{v} \rfloor + 1$ and $W = \lfloor \frac{y_{max}-y_{min}}{v} \rfloor + 1$, where $\lfloor \cdot \rfloor$ is the floor operation. Assuming the 3D center (x, y, z) of the target ground truth, we can compute the 2D target center $c = (c_x, c_y)$ in x-y plane by $c_x = \frac{x-x_{min}}{v}$ and $c_y = \frac{y-y_{min}}{v}$. Besides, the discrete 2D center $\tilde{c} = (\tilde{c}_x, \tilde{c}_y)$ is defined by $\tilde{c}_x = \lfloor c_x \rfloor$ and $\tilde{c}_y = \lfloor c_y \rfloor$.

2D-center head. Following [65], we obtain the target center's ground truth $\mathcal{H} \in \mathbb{R}^{H \times W \times 1}$. For the pixel (i, j) in the 2D bounding box, if $i = \tilde{c}_x$ and $j = \tilde{c}_y$, the $\mathcal{H}_{ij} = 1$, otherwise $\frac{1}{d+1}$, where d represents the Euclidean distance between the pixel (i, j) and the target center $(\tilde{c}_x, \tilde{c}_y)$. For any pixel outside the 2D bounding box, $\mathcal{H}_{ij}$ is set to 0. In the training phase, we enforce the predicted map $\hat{\mathcal{H}} \in \mathbb{R}^{H \times W \times 1}$ to approach the ground truth $\mathcal{H}$ by using Focal loss [66]. The modified Focal loss is formulated as:

$$\mathcal{L}_{center} = -\sum \mathbb{I}[\mathcal{H}_{ij} = 1] \cdot (1-\hat{\mathcal{H}}_{ij})^{\alpha} \log(\hat{\mathcal{H}}_{ij}) + \mathbb{I}[\mathcal{H}_{ij} \neq 1] \cdot (1-\mathcal{H}_{ij})^{\beta} (\hat{\mathcal{H}}_{ij})^{\alpha} \log(1-\hat{\mathcal{H}}_{ij}) \quad (7)$$

where $\mathbb{I}(\textit{cond.})$ is the indicator function. If *cond.* is true, then $\mathbb{I}(\textit{cond.}) = 1$, otherwise 0. Besides, we empirically set $\alpha = 2$ and $\beta = 4$ in all experiments.

Offset & rotation head. Since the continuous 2D object center is converted into the discrete one through floor operation, we consider regressing the offset of the continuous ground truth center. To improve the accuracy of regression, we consider a square area with radius r around the object center. Here, we also add rotation regression. Given a predicted map $\hat{\mathcal{O}} \in \mathbb{R}^{H \times W \times 3}$, where 3-dim means the 2D coordinate offset plus rotation, the error of offset and rotation is expressed as:

$$\mathcal{L}_{off} = \sum_{\triangle x=-r}^{r} \sum_{\triangle y=-r}^{r} \left| \hat{\mathcal{O}}_{\tilde{c}+(\triangle x, \triangle y)} - [c - \tilde{c} + (\triangle x, \triangle y), \theta] \right| \quad (8)$$

where $\tilde{c}$ and c mean the discrete and continuous position of the ground truth center, respectively. Besides, θ indicates the ground truth rotation angle and $[\cdot, \cdot]$ is the concatenation operation.

z-axis head. We directly regress the z-axis location of the target center from the BEV feature map. Given a predicted map $\hat{\mathcal{Z}} \in \mathbb{R}^{H \times W \times 1}$, we use L_1 loss to compute the error of z-axis center by:

$$\mathcal{L}_z = \left| \hat{\mathcal{Z}}_{\tilde{c}} - z \right| \quad (9)$$

where $\tilde{c}$ is the discrete object center, and z is z-axis center's ground truth.

The final loss of our network is as follows: $\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{shape} + \lambda_2 (\mathcal{L}_{center} + \mathcal{L}_{off}) + \lambda_3 \mathcal{L}_z$, where λ_1, λ_2 and λ_3 are the hyperparameter for shape generation, 2D center and offset regression, and z-axis position regression, respectively. In the experiment, we set $\lambda_1 = 10^{-6}$, $\lambda_2 = 1.0$ and $\lambda_3 = 2.0$. For more details, please refer to the supplementary material.

4 Experiments

4.1 Experimental Settings

Dataset. Since the ground truth of the test set of KITTI tracking dataset [15] cannot be obtained, we follow [13, 12] and use the training set to train and evaluate our method. It contains 21 video sequences and 8 types of objects. Specifically, we use scenes 0-16 for training, scenes 17-18 for validation, and scenes 19-20 for testing. We use it to perform the 3D single object tracking task.

Evaluation metrics. For 3D single object tracking, we use the *Success* and *Precision* defined in the One Pass Evaluation (OPE) [67] to evaluate the tracking performance of different methods. *Success* measures the IoU between the predicted and ground truth bounding boxes, while *Precision* measures the error AUC of the distance between the centers of two bounding boxes.

Implementation details. Following [12], we set the number of points $N = 512$ and $M = 1024$ for the template and search area by randomly discarding and duplicating points. For the backbone network, we use a slightly modified PointNet++ [51], which consists of three set-abstraction (SA) layers (with query radius of 0.3, 0.5, and 0.7) and three feature propagation (FP) layers. For each SA layer passed, the points will be randomly downsampled by half. For the shape generation network, we generate 2048 points. The global branch is the max pooling combined with two fully connected layers,

Table 1: The results of *Success*/*Precision* of different methods in the KITTI tracking dataset. "Mean" represents the average results of four categories.

Method Frame Number	Modality	Car 6424	Pedestrian 6088	Van 1248	Cyclist 308	Mean 14068
AVOD [69]	RGB+LiDAR	63.2 / 69.7	-	-	-	-
SC3D [13]	LiDAR	41.3 / 57.9	18.2 / 37.8	40.4 / 47.0	**41.5 / 70.4**	31.2 / 48.5
P2B [12]	LiDAR	56.2 / 72.8	28.7 / 49.6	40.8 / 48.4	32.1 / 44.7	42.4 / 60.0
Re-Track [11]	LiDAR	58.4 / 73.4	-	-	-	-
V2B-Tracker	LiDAR	**70.5 / 81.3**	**48.3 / 73.5**	**50.1 / 58.0**	40.8 / 49.7	**58.4 / 75.2**

while the local branch only uses one EdgeConv layer. We use a two layer MLP network to generate 3D coordinates. For 3D center detection, the voxel size is set to 0.3 meters in volumetric space. We stack four 3D convolutions (with stride of 2, 1, 2, 1 along the z-axis) and four 2D convolutions (with stride of 2, 1, 1, 2) combined with the skip connections for feature aggregation, respectively. For all experiments, we use Adam [68] optimizer with learning rate 0.001 for training, and the learning rate decays by 0.2 every 6 epochs. It takes about 20 epochs to train our model to convergence. Please refer to the supplementary material for more details.

Training and testing. For training, we combine the points inside the first ground truth bounding box (GTBB) and the points inside the previous GTBB plus the random offset as the template of the current frame. To generate the search area, we enlarge the current GTBB by 2 meters and plus the random offset. For testing, we fuse the points inside the first GTBB and the previous result's point cloud (if exists) as the template. Besides, we first enlarge the previous result by 2 meters in current frame, and then collect the points lying in it to generate the search area.

4.2 Results

Quantitative results. We compare our method with current state-of-the-art methods, including SC3D [13], P2B [12], and Re-Track [11]. The quantitative results are listed in Tab. 1. Following [13, 12], we report the performance of four categories, including car, pedestrian, van, and cyclist, and their average results. As one can see from the table, our method is significantly better than other methods on the mean results of four categories. On the car category, our method can even improve the *Success* from 58.4% (Re-Track) to 70.5% (V2B-Tracker). In addition, SC3D achieves the highest performance in the cyclist category. Since the training samples of cyclists is far less than the other three categories, it is difficult to effectively track cyclists, especially for those tacking-by-detection methods, such as P2B and our V2B-Tracker. However, SC3D uses exhaustive search to generate numerous candidate proposals for template matching, so it can perform well with few training samples.

Table 2: The results of *Success*/*Precision* of different methods on the **sparse** scenarios.

Method	SC3D [13]	P2B [12]	V2B-Tracker
Car	37.9 / 53.0	56.0 / 70.6	**64.7 / 77.4**
Pedestrian	20.1 / 42.0	33.1 / 58.2	**50.8 / 74.2**
Van	36.2 / 48.7	41.1 / 46.3	**46.8 / 55.1**
Cyclist	**50.2 / 69.2**	24.1 / 28.3	30.4 / 37.2
Mean	32.7 / 49.4	47.2 / 63.5	**58.0 / 73.2**

To verify the effectiveness of our method for tracking in sparse scenes, we count the performance of SC3D, P2B, and our V2B-Tracker in sparse scenes of the KITTI tracking dataset. We filter out sparse scenes for evaluation according to the number of points lying in the target bounding boxes in the test set. Specifically, the conditions for the sparse scenes are: ≤ 150 (car), ≤ 100 (pedestrian), ≤ 150 (van), and ≤ 100 (cyclist), respectively. For the four categories, the number of selected frames are 3293 (car), 1654 (pedestrian), 734 (van), and 59 (cyclist), respectively. In Tab. 2, we report the results of *Success* and *Precision*. Note that we run the official code to obtain the results of SC3D and P2B. As one can see from the table, our V2B-Tracker can significantly outperform P2B for all four categories, which further demonstrates the effectiveness of our V2B-Tracker in improving tracking performance in sparse point clouds. The results on the cyclist category are poor due to few training samples.

Visual results. As shown in Fig. 4, we compare the tracking results of our method with P2B. Specifically, we plot a couple sparse and dense scenarios on the car category of the KITTI tracking dataset. It can be clearly seen from the figure that compared with P2B, our V2B-Tracker can track the targets more accurately in both sparse and dense scenes. Especially in the sparse scene, our

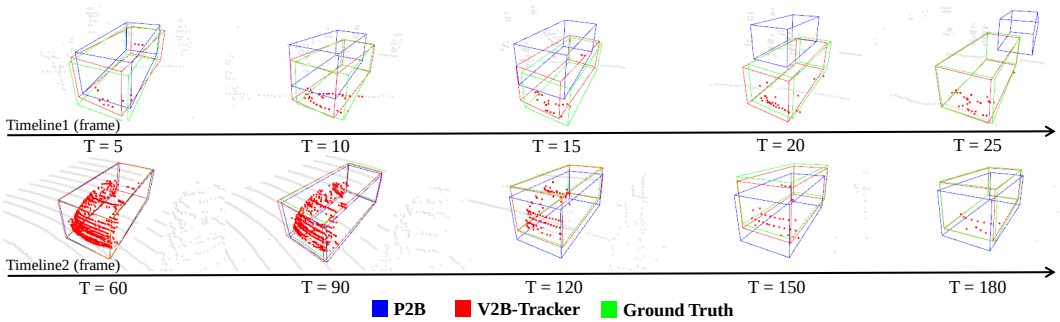


Figure 4: Visualization results of sparse (the first-row sequence) and dense (the second-row sequence) scenes on the car category. The green boxes are ground truth bounding boxes. The red boxes are the objects tracked by our V2B-Tracker, while the blue boxes are the objects tracked by P2B. In addition, we mark the points of cars to red for better identification.

V2B-Tracker can effectively track the targets while P2B fails to track the targets. The visualization results demonstrate the effectiveness of our V2B-Tacker for sparse scenes. For more visualization results, please refer to the supplementary material.

Computational cost. We also compare our V2B-Tracker with SC3D and P2B in terms of parameters, FLOPs, FPS, and training time. We conduct experiments on the car category in the KITTI tracking dataset. For a fair comparison, we use a single TITAN RTX GPU to train and test all codes under the PyTorch [70] platform. We report the results in Tab. 3. One can see that our method is comparable to P2B in terms of FPS. Because the shape generation network will be discarded in the test scheme. Although our FPS is comparable to P2B, our tracking performance is significantly better than P2B. Since SC3D uses the exhaustive search to generate numerous candidate proposals for template matching, it consumes much time, and the FPS is low. Besides, the reported training time represents the time required for the model to converge. The training time of our method is shorter than SC3D and P2B. On the one hand, our method can learn discriminative features through shape-aware feature learning, so as to accelerate the convergence. On the other hand, it is easier to detect the center in the 2D space than in the 3D space.

Table 3: The computational cost of different methods.

Method	Param	FLOPs	FPS	Training time
SC3D [13]	6.45 M	20.07 G	2	∼13 h
P2B [12]	1.34 M	4.28 G	17	∼13 h
V2B-Tracker	1.35 M	5.47 G	13	∼8 h

4.3 Ablation Study

Template feature embedding. We study the impact of template feature embedding on tracking performance. As shown in Tab. 4, we report the results on the car category. It can be seen that without using the template feature embedding (dubbed as "w/o template feature"), the performance will be greatly reduced from 70.5 / 81.3 to 63.9 / 73.9 by a large margin. Since template feature embedding builds the relationship between the template and search area, it will contribute to identify the potential target from the background in the search area. Therefore, when the template feature embedding is absent, the performance will be greatly reduced, which further demonstrates the effectiveness of the proposed template feature embedding for improving tracking performance.

Table 4: The ablation study results of different components on the car category.

Module	*Success*/*Precision*
w/o template feature	63.9 / 73.9
w/o shape information	67.6 / 78.2
our default setting	**70.5 / 81.3**

Shape-aware feature learning. For sparse point clouds, we further introduce shape-aware feature learning to enhance the ability to distinguish the potential target from the background in the search area. As shown in Tab. 4, we conduct experiments to demonstrate the effectiveness of the shape information. It can be seen from the table that if shape-aware feature learning (dubbed as "w/o shape information") is not used, the performance will drop from 70.5 / 81.3 to 67.6 / 78.2. Since the shape

generation network can capture 3D shape information of the object to learn the discriminative features of potential target so that it can be identified from the search area.

Different voxel sizes. We compress the voxelized point cloud into a BEV feature map for subsequent target center detection. Since the scope of object tracking is a large area, the size of voxel will affect the size of BEV feature map, thereby affecting the tracking performance. We hence study the impact of different voxel sizes on the tracking performance. Specifically, we consider four sizes, which are 0.1, 0.2, 0.3, and 0.4 meters. The *Success*/*Precision* results of the four sizes are 52.5 / 62.2 (0.1), 68.1 / 79.4 (0.2), **70.5 / 81.3** (0.3), and 69.1 / 80.0 (0.4), respectively. When the voxel size is set to 0.3 meters, we achieve the best performance. A larger voxel size will increase the sparsity and cause the loss of the target's details. A smaller voxel size will increase the size of the BEV feature map, thereby increasing the difficulty of detecting the center.

Voxel-to-BEV target localization. Different from SC3D [13] and P2B [12], our V2B-Tracker adopts another route to localize potential target in object tracking. SC3D performs matching between the template and the exhaustive candidate 3D proposals to select the most similar proposal as the target. P2B uses VoteNet [14] to generate 3D target proposals, and select the proposal with the highest score as the target. However, when facing sparse point clouds, it is hard to generate high-quality proposals, so these methods may not be able to track the object effectively. Our V2B-Tracker is an anchor-free method that does not require generating numerous 3D proposals. Therefore, our method can overcome the above concern. In order to prove this, we use VoteNet instead of voxel-to-BEV target localization to conduct experiments in the KITTI tracking dataset. In Tab. 5, we report the results of different detection methods in the sparse scenarios. Likewise, we filter out sparse scenes in the test set for evaluation according to the number of points (refer to the setting of Tab. 2). It can be found that the results of VoteNet are lower than that of voxel-to-BEV target localization, which further demonstrates the effectiveness of our method in sparse point clouds.

Table 5: The ablation study results (*Success*/*Precision*) of different detection schemes on the **sparse** scenarios.

Module	VoteNet	Voxel-to-BEV
Car	56.9 / 72.0	**64.7 / 77.4**
Pedestrian	35.3 / 62.1	**50.8 / 74.2**
Van	30.7 / 39.0	**46.8 / 55.1**
Cyclist	23.9 / 30.0	**30.4 / 37.2**

Template generation scheme. Following [13, 12], we study the impact of different template generation schemes on tracking performance. As shown in Tab. 6, we report the results of four schemes on the car category in the KITTI tracking dataset. It can be seen from the table that our V2B-Tracker outperforms SC3D and P2B in all schemes by a large margin. In experiments, when the first ground truth and previous result are used to generate the template, we can obtain the highest performance. Compared with SC3D and P2B, our method can yield stable results on four template generation schemes, which further demonstrates that our method can consistently generate accurate tracking results in all frames.

Table 6: The results of different template generation schemes of different methods on the car category.

Scheme	SC3D [13]	P2B [12]	V2B-Tracker
The First GT	31.6 / 44.4	46.7 / 59.7	**67.8 / 79.3**
Previous result	25.7 / 35.1	53.1 / 68.9	**70.0 / 81.3**
The First GT & Previous result	34.9 / 49.8	56.2 / 72.8	**70.5 / 81.3**
All previous results	41.3 / 57.9	51.4 / 66.8	**69.8 / 81.2**

5 Conclusion

In this paper, we proposed a Siamese voxel-to-BEV tracker (V2B-Tracker), which can significantly improve the tracking performance in sparse point clouds. Our V2B-Tracker consists of the Siamese shape-aware feature learning network and the voxel-to-BEV target localization network. For sparse point clouds, the Siamese shape-aware feature learning network exploits the generation network to characterize the shape information of the target in order to improve the discrimination of features. For localizing the target, we convert the sparse 3D space into a dense BEV space and then regress the target's 2D center and the z-axis center from it, respectively. Compared with sparse 3D space, regressing target center from the dense 2D feature can improve the tracking performance, especially in the sparse scenes. Rich experiments have demonstrated the effectiveness of our method.

References

[1] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *ICCV workshops*, 2015.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional Siamese networks for object tracking. In *ECCV*, 2016.

[3] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *CVPR*, 2017.

[4] Jack Valmadre, Luca Bertinetto, Joao Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[5] K. Lebeda, S. Hadfield, and R. Bowden. 2D or not 2D: Bridging the gap between tracking and structure from motion. In *ACCV*, 2014.

[6] T. Whelan, Renato F. Salas-Moreno, B. Glocker, A. Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35:1697 – 1716, 2016.

[7] Martin Rünz and L. Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *ICRA*, 2017.

[8] Ugur Kart, J. Kämäräinen, and Jiri Matas. How to make an rgbd tracker? In *ECCV Workshops*, 2018.

[9] Y. Liu, X. Jing, Jianhui Nie, Hao Gao, Jun Liu, and Guo-Ping Jiang. Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in rgb-d videos. *IEEE Transactions on Multimedia*, 21:664–677, 2019.

[10] Neil Gordon, B Ristic, and S Arulampalam. Beyond the Kalman filter: Particle filters for tracking applications. *Artech House, London*, 830(5):1–4, 2004.

[11] Tuo Feng, Licheng Jiao, Hao Zhu, and Long Sun. A novel object re-track framework for 3D point clouds. In *ACM MM*, 2020.

[12] Haozhe Qi, C. Feng, Zhiguo Cao, F. Zhao, and Yang Xiao. P2B: Point-to-box network for 3D object tracking in point clouds. *CVPR*, 2020.

[13] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completion for 3D Siamese tracking. In *CVPR*, 2019.

[14] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3D object detection in point clouds. In *ICCV*, 2019.

[15] Andreas Geiger, Philip Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012.

[16] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese" time delay neural network. In *NeurIPS*, 1993.

[17] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomáš Vojíř, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, 2016.

[18] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[19] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016.

[20] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010.

[21] Joao F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*, 2012.

[22] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2014.

[23] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016.

[24] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019.

[25] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

[26] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018.

[27] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019.

[28] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, 2020.

[29] Zhipeng Zhang, Houwen Peng, Jianlong Fu, Bing Li, and Weiming Hu. Ocean: Object-aware anchor-free tracking. *arXiv preprint arXiv:2006.10721*, 2020.

[30] Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *CVPR*, 2021.

[31] Christoph Feichtenhofer, A. Pinz, and Andrew Zisserman. Detect to track and track to detect. In *ICCV*, 2017.

[32] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[33] Yuechen Yu, Yilei Xiong, Weilin Huang, and M. Scott. Deformable siamese attention networks for visual object tracking. In *CVPR*, 2020.

[34] P. Voigtlaender, Jonathon Luiten, P. Torr, and B. Leibe. Siam R-CNN: Visual tracking by re-detection. In *CVPR*, 2020.

[35] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Visual tracking by tridentalign and context embedding. In *ACCV*, 2020.

[36] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

[37] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold Siamese network for real-time object tracking. In *CVPR*, 2018.

[38] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. Learning attentions: residual attentional Siamese network for high performance online visual tracking. In *CVPR*, 2018.

[39] Wenzhang Zhou, Longyin Wen, L. Zhang, Dawei Du, T. Luo, and Y. Wu. SiamMan: Siamese motion-aware network for visual tracking. *arXiv preprint arXiv:1912.05515*, 2019.

[40] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *CVPR*, 2021.

[41] Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *ECCV*, 2018.

[42] Shuran Song and Jianxiong Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *ICCV*, 2013.

[43] Adel Bibi, Tianzhu Zhang, and Bernard Ghanem. 3D part-based sparse tracker with automatic synchronization and registration. In *CVPR*, 2016.

[44] Ye Liu, Xiao-Yuan Jing, Jianhui Nie, Hao Gao, Jun Liu, and Guo-Ping Jiang. Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in RGB-D videos. *IEEE Transactions on Multimedia*, 21(3):664–677, 2018.

[45] Ugur Kart, A. Lukezic, M. Kristan, J. Kämäräinen, and Jiri Matas. Object tracking by reconstruction with view-specific discriminative correlation filters. In *CVPR*, 2019.

[46] Luciano Spinello, Kai Arras, Rudolph Triebel, and Roland Siegwart. A layered approach to people detection in 3D range data. In *AAAI*, 2010.

[47] Matthias Luber, Luciano Spinello, and Kai O Arras. People tracking in RGB-D data with on-line boosted target models. In *IROS*, 2011.

[48] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018.

[49] Sukai Wang, Yuxiang Sun, Chengju Liu, and Ming Liu. Pointtracknet: An end-to-end network for 3D object detection and tracking from point clouds. *IEEE Robotics and Automation Letters*, 5(2):3206–3212, 2020.

[50] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[51] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[52] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on
$mathcalX$-transformed points. In *NeurIPS*, 2018.

[53] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[54] Charles R Qi, Hao Su, Matthias Niebner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3D data. In *CVPR*, 2016.

[55] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-Voxel cnn for efficient 3D deep learning. In *NeurIPS*, 2019.

[56] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[57] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018.

[58] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *CVPR*, 2019.

[59] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019.

[60] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015.

[61] Hao Su, Fan Wang, Eric Yi, and Leonidas J Guibas. 3D-assisted feature synthesis for novel views of an object. In *ICCV*, 2015.

[62] Ze Yang and Liwei Wang. Learning relationships for multi-view 3d object recognition. In *ICCV*, 2019.

[63] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *CVPR*, 2018.

[64] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017.

[65] Runzhou Ge, Zhuangzhuang Ding, Y. Hu, Yu Wang, Sijia Chen, L. Huang, and Y. Li. AFDet: Anchor free one stage 3D object detection. *arXiv preprint arXiv: 2006.12671*, 2020.

[66] Tsung-Yi Lin, Priyal Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:318–327, 2020.

[67] M. Kristan, Jiri Matas, A. Leonardis, Tomás Vojír, R. Pflugfelder, G. Fernández, G. Nebehay, F. Porikli, and Luka Cehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:2137–2155, 2016.

[68] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

[69] Jason T Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander. Joint 3D proposal generation and object detection from view aggregation. In *IROS*, 2018.

[70] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please refer to the supplementary material.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please refer to the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Refer to the experimental part of the main paper.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Refer to the experimental part of the main paper.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]