Box-Aware Feature Enhancement for Single Object Tracking on Point Clouds

Chaoda Zheng[1,†], Xu Yan[1,†], Jiantao Gao[2], Weibing Zhao[1], Wei Zhang[3], Zhen Li[1,*], Shuguang Cui[1]

[1]The Chinese University of Hong Kong (Shenzhen), Shenzhen Research Institute of Big Data,
[2]Research Institute of USV Engineering, Shanghai University, [3]Baidu Inc

{chaodazheng@link., xuyan1@link., lizhen@}cuhk.edu.cn

Abstract

*Current 3D single object tracking approaches track the target based on a feature comparison between the target template and the search area. However, due to the common occlusion in LiDAR scans, it is non-trivial to conduct accurate feature comparisons on severe sparse and incomplete shapes. In this work, we exploit the ground truth bounding box given in the first frame as a strong cue to enhance the feature description of the target object, enabling a more accurate feature comparison in a simple yet effective way. In particular, we first propose the **BoxCloud**, an informative and robust representation, to depict an object using the point-to-box relation. We further design an efficient box-aware feature fusion module, which leverages the aforementioned BoxCloud for reliable feature matching and embedding. Integrating the proposed general components into an existing model P2B [27], we construct a superior **box-aware tracker (BAT)**[1]. Experiments confirm that our proposed BAT outperforms the previous state-of-the-art by a large margin on both KITTI and NuScenes benchmarks, achieving a **12.8%** improvement in terms of precision while running ~**20%** faster.*

1. Introduction

Single object tracking (SOT) in 3D scene has a broad spectrum of practical applications, such as autonomous driving [20], semantic understanding [38, 37] and assistive robotics [21, 7]. Given a 3D bounding box (BBox) of an object as the template in the first frame, the SOT task is to keep track of this object across all frames. In real scenes, LiDAR becomes a popular 3D sensor due to its precise measurement, reasonable cost and insensitivity to ambient light variations. In this paper, we focus on SOT on LiDAR data, which can be viewed as 3D point clouds in general.

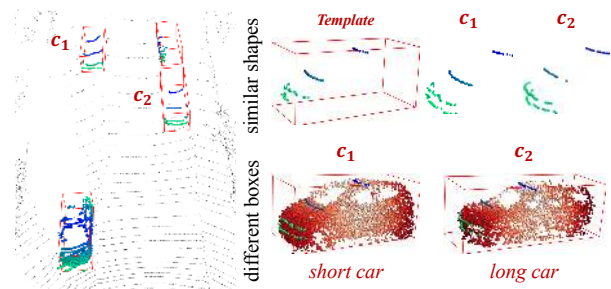Due to the moving environment and self-occlusion, point

---
* Corresponding author: Zhen Li. † Equal first authorship.
[1] https://github.com/Ghostish/BAT

Figure 1. **Motivation behind BAT.** Partial scans of two different cars (*i.e.* c1, c2) collected by a LiDAR sensor both have highly similar shapes with the target template (**1st row**), though their 3D BBoxes are quite different in size (**2nd row**). With explicit consideration of object bounding-boxes, our method can elegantly overcome such ambiguities.

clouds generated by a LiDAR system are inevitably irregular and incomplete, making the SOT task very challenging. In 3D SOT, feature comparison plays an important role. The general idea to locate the target object is based on measuring the feature similarity between some candidate regions and the object template (initialized as the point cloud in the first given BBox). For example, SC3D [13] uses the exhaustive search or Kalman Filter to generate a set of candidate shapes at the current frame, and comparing them to the template using a siamese network. The candidate with the maximum similarity is chosen to be the target object for the frame. Inspired by the success of the siamese region proposal network (RPN) [17] in 2D SOT, P2B [27] proposes a point-based correlation based on the pair-wise feature comparison. P2B executes such a correlation between the template and the search area to output the target-specific search features, on which a 3D RPN is applied to obtain the final target proposals. However, the features used for comparison are extracted from pure LiDAR point clouds, which face the following defects: 1) They do not encode the size information of objects. Since objects in LiDAR scans are mostly incomplete, it is hard to infer an object's size only from the partial point cloud. 2) They cannot capture the explicit part-aware structure information within each object

BBox, *e.g.* some part belongs to the car front while others belong to the sunroof as Figure 1 shows. Therefore, the feature comparison among such features may bring considerable ambiguities which weaken the tracking performance.

What is a good representation for feature matching under 3D SOT? We revisit this problem by pointing out that the size and the part priors of the target object can be directly inferred from the template BBox given at the first frame. Based on this observation, we propose to address the above issues by explicitly utilizing the BBox to enhance the object features. Thus, we propose the BoxCloud, a robust and informative object representation depicting the point-to-box relation. Instead of using the xyz coordinate, it represents an object point via a canonical *box coordinate*, where the i-th dimension corresponds to the Euclidean distance between the object point and its i-th box point (*i.e.* the corner or center of a BBox). Unlike the original LiDAR point cloud, a BoxCloud is defined based on both the object and its BBox. Therefore, it naturally encodes the size and part information of an object. Note that we can directly compute the BoxCloud for a target template using the BBox given at the first frame. After the supervised training using ground-truth object BBoxes, the BoxClouds of objects in the search area can be easily predicted for the inference usage.

Based on the BoxCloud representation, the box-aware feature fusion (BAFF) module is further proposed to perform a correlation between the template and the search area to generate the target-specific search area features. It first measures the similarity between the search area and the template according to their BoxClouds. After such effective feature comparison, the BAFF module aggregates the top-k similar template points into each corresponding searching point, yielding a high-quality target-specific search area. Finally, we construct a Box-Aware Tracker (BAT) by integrating the two proposed components into P2B [27]. By taking the auxiliary 3D BBox as input, BAT captures more shape constraints and part-aware information, no matter whether the input shape is partial or not, enabling effective and robust tracking on LiDAR point clouds.

Our main contributions can be summarized as follows:

- To the best of our knowledge, we are the first to use free box information to boost the performance on the 3D SOT task. Specifically, we improve the feature comparison by designing a size-aware and part-aware BoxCloud feature, which is not only interpretable but also robust to sparseness and incompleteness.
- We propose a dedicated box-aware feature fusion module to generate better target-specific search areas in a box-aware manner.
- Experiments verify that our BAT achieves significant improvement over the state-of-the-arts on two benchmarks (*i.e.* KITTI and NuScenes), especially on extremely sparse data.

2. Related Work

2D Siamese Tracking. Recently, Siamese based methods [1, 36, 19, 16, 17, 43, 31, 42, 2] have demonstrated their advantages over those based on the traditional Discriminative Correlation Filter (DCF) [10, 9, 3, 35, 8] in 2D visual object tracking. Siamese based trackers formulate the visual object tracking as a feature matching problem. By utilizing the pre-trained feature extraction networks, siamese based trackers first project the target and search image onto a hidden embedding space and then compute their mutual similarity. Although 2D siamese based trackers have made remarkable progress in recent years, they focus on image patches and thus cannot be applied on point clouds directly.

3D Single Object Tracking. Early 3D SOT methods [23, 4, 14, 18, 15] focus on the RGB-D information and tend to employ the 2D siamese architecture. SC3D [13] proposes the first 3D siamese tracker on point clouds and introduces an auxiliary branch for shape completion to process the incomplete shapes. Although the object semantics can be learned via the shape completion task, SC3D lacks the ability of detecting the object relations in the search space. Moreover, it has low efficiency in target proposal generation and cannot be trained in an end-to-end manner. The follow-up [40] accelerates SC3D by leveraging the SiamRPN [17] to generate target proposals from the bird eye views. However, its RPN only operates in 2D, and thus it cannot sense the 3D object relations. To address these, P2B [27] adapts the SiamRPN [17] to the 3D case where pure point clouds are processed. Firstly, the target object information is fused into the search space. Following that, an object detection network is applied to the search space to detect the target. Equipped with a state-of-the-art detection network, *i.e.* VoteNet [24], P2B achieves significant improvement in terms of both accuracy and efficiency. Very recently, Feng *et al.* [11] presents a two-stage object re-track framework for 3D point clouds, which can re-track the lost object at the coarse stage. However, none of these methods exploit the BBox provided in the first frame as an additional cue. The information loss incurred by the incompleteness of shapes cannot be compensated, and their performance drops drastically when they deal with incomplete scans.

3D Multi-Object Tracking. Most of the 3D Multi-Object Tracking (MOT) methods follow the tracking-by-detection paradigm [32, 6, 39, 41, 22, 33]. Unlike SOT where the 3D BBox of the target is provided in the first frame, the MOT tracker determines the number of objects in all frames by running an independent detector [29, 30, 28]. After that, the tracking is done by performing data association on the detection results (*i.e.* linking the detected object BBoxes across all frames to obtain full trajectories), according to the motion estimation. The motion of the objects can be estimated using the handcrafted Kalman filter [32, 22], or some learned features [41, 39].

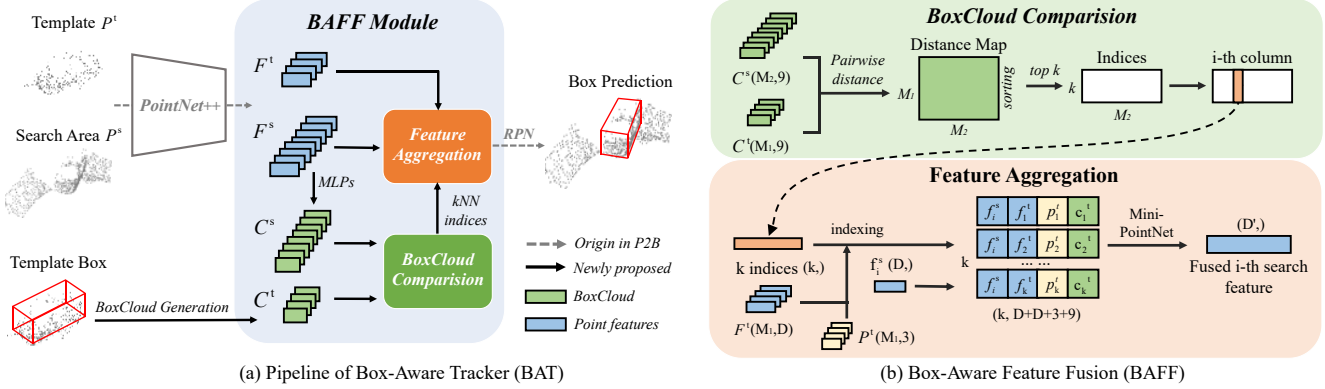(a) Pipeline of Box-Aware Tracker (BAT)   (b) Box-Aware Feature Fusion (BAFF)

Figure 2. **Box-Aware Tracker (BAT) pipeline and Box-Aware Feature Fusion (BAFF) module.** Part (a) illustrates the pipeline of BAT. The *RPN* component denotes the 3D RPN used in [27], which is exploited to generate the final 3D target proposals. Part (b) illustrates the workflow of BAFF module. It first uses the distance map of BoxClouds as the metric to find out the k-nearest neighbors in the template point cloud with respect to each search area point. Then, a Mini-PointNet is used to aggregate the retrieved neighbors' features.

Bounding Box Utilization in 3D. In the 3D SOT task, though the BBox of the object is provided in the first frame, the previous methods [13, 27] only expand the BBox with a fixed magnification and crop out the point cloud inside as a template. In 3D detection, [30] predicts intra-object part locations of all 3D points within a proposal, using the free-of-charge part supervisions derived from 3D ground-truth BBoxes. However, it only focuses on part-aware information while ignores the sizes of BBoxes. Moreover, it only uses BBoxes as supervision instead of the network input. In contrast, our method explicitly takes the given target BBox as input to enhance the target features with both part-aware and size-aware cues.

3. Method

3.1. Problem Statement

A 3D BBox is represented as $(x, y, z, w, l, h, \theta) \in \mathbb{R}^7$, where (x, y, z) stands for the coordinate of the BBox center, (w, l, h) is the BBox size and θ is the heading angle (the rotation around the *up*-axis). A point cloud is represented as $\{p_i\}_{i=1}^N$, where each point p_i is a vector of its (x, y, z) coordinate and N is the number of points.

In 3D single object tracking (SOT), the BBox of the target in the first frame is given to the tracker. The goal of the tracker is to locate the same target in the search area frame by frame. Notice that in the 3D case, any BBox is *amodal* (covering the entire object even if only part of it is visible). A template point cloud $P^t = \{p_i^t\}_{i=1}^{N_t}$ is generated by cropping and centering the target in the first frame with the given BBox B^t. For any search area $P^s = \{p_i^s\}_{i=1}^{N_s}$ in other frames, previous trackers take in the pair (P^t, P^s) and output the amodal BBox of the target in the search area.

Formally, previous trackers can be formulated as:

$$track : \mathbb{R}^{N_t \times 3} \times \mathbb{R}^{N_s \times 3} \to \mathbb{R}^4, \qquad (1)$$

$$track(P^t, P^s) \mapsto (x, y, z, \theta). \qquad (2)$$

Note that the output has only 4 elements since we do not need to re-predict the size of the 3D BBox, which is unchanged across all frames.

Since previous works compare the search area only with the template that is usually incomplete, they lack the ability to capture the size and part information of the target object. Therefore, they are prone to produce false target proposals when similar objects are present. In this work, we aim at designing a box-aware tracker *track_box-aware* that exploits the amodal BBox of the target:

$$track_{box\text{-}aware} : \mathbb{R}^7 \times \mathbb{R}^{N_t \times 3} \times \mathbb{R}^{N_s \times 3} \to \mathbb{R}^4, \qquad (3)$$

$$track_{box\text{-}aware}(B^t, P^t, P^s) \mapsto (x, y, z, \theta). \qquad (4)$$

3.2. Box-Aware Tracker (BAT)

Our box-aware tracker is built upon P2B [27]. Following the siamese paradigm, P2B first generates a target-specific search area by executing a point-wise correlation between the template and the search features, which are extracted by a shared backbone. The correlation is based on the point-wise similarity and utilizes PointNet [25] to achieve permutation invariance. Then it applies a VoteNet [24] based RPN to generate target proposals. Specifically, a voting module votes the seed points to potential target centers. Besides, a seed-wise targetness score is also predicted to regularize the feature learning. These potential target centers are then grouped into clusters using ball query [26], which are finally turned into target proposals through another PointNet. Each proposal is a 5D vector containing the coordinates of the target center (3D), the rotation in the horizontal plane

(1D) and the targetness score (1D). The proposal with the highest targetness score is chosen as the final prediction.

Our work focus on improving the first part of P2B [27] because generating a better target-specific search area is critical for robust tracking. The overall architecture of BAT is shown in Figure 2 (a). Given a template point clouds $P^t \in \mathbb{R}^{N_1 \times 3}$ and a search area $P^s \in \mathbb{R}^{N_2 \times 3}$, a shared PointNet++ [26] is utilized to extract features from both of them, resulting in $F^t \in \mathbb{R}^{M_1 \times D}$ and $F^s \in \mathbb{R}^{M_2 \times D}$. Besides, the BoxCloud $C^t \in \mathbb{R}^{M_1 \times 9}$ (Section 3.3) is obtained from the template and its BBox. The BoxCloud of search area C^s is predicted by a multi-layer perceptron (MLP). After that, through BoxCloud comparison and feature aggregation sub-modules (Section 3.4), we obtain the target-specific search area $\hat{F}_s \in \mathbb{R}^{M_2 \times D}$. After getting $\hat{F}_s$, we yield the final target proposals using the same way as [27].

3.3. BoxCloud Representation

Previous works only use the BBox given at the first frame to crop and center the target during the pre-processing, ignoring that it contains rich shape information about the target. For a BBox $(x, y, z, w, l, h, \theta)$ of an object, its (w, l, h) indicates the size of the full object, even if part of the object is occluded or truncated. Its (x, y, z, θ) indicates the object coordinate system. Knowing the size and the object coordinate system, we can infer the intra-object location of each point, i.e. the part information. Based on these observations, we design the BoxCloud representation to fully utilize such shape priors.

A BoxCloud is defined by the point-to-box relation between an object point cloud P and its BBox B. For each point p_i in P, we first calculate its Euclidean distance to each of the eight corners and the center of B, resulting in nine distances. Then we arrange them in a predefined order with respect to the object coordinate system, generating a 9-D vector c_i (as shown in Figure 3). We denote c_i as the *box coordinate* of p_i with respect to B. A BoxCloud is a set of box coordinates and can be formulated as follows:

$$C = \{c_i \in \mathbb{R}^9 \mid c_{ij} = ||p_i - q_j||_2, \ \forall j \in [1, 9]\}_{i=1}^N, \quad (5)$$

where $q_{j(j \neq 9)}$ is the j-th corner and q_9 is the center of B.

Though simple, BoxCloud features own the following properties over the geometric point clouds (i.e. spatial xyz coordinates of point clouds):

Size-Awareness. The full size of an object can be directly inferred from its BoxCloud regardless of shape incompleteness. This helps to distinguish similar object parts of incomplete scans from objects of different sizes.

Part-Awareness. Each c_i indicates the intra-object location of p_i, because c_i encodes the relative position of p_i with respect to the corners and center. For example, if $\arg\min_j c_{ij} = 1$, which mean p_i is closest to q_1, we know p_i belongs to the upper left part of the object head.

| Box Coordinate | c_{i1} | c_{i2} | c_{i3} | c_{i4} | c_{i5} | c_{i6} | c_{i7} | c_{i8} | c_{i9} |

Head: ◯ upper left ◯ upper right ● lower left ◯ lower right
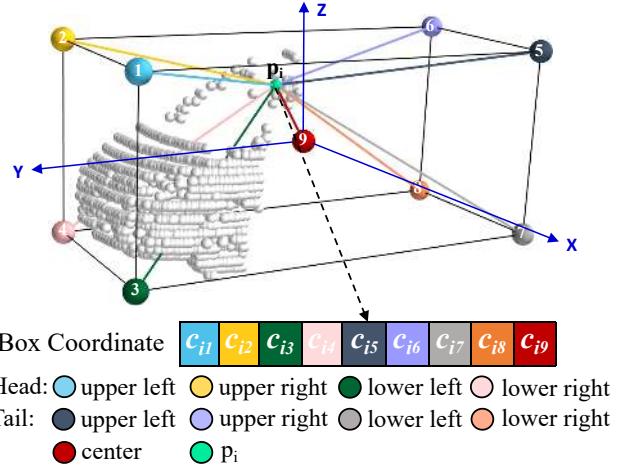Tail: ● upper left ◯ upper right ◯ lower left ◯ lower right
● center ● p_i

Figure 3. **BoxCloud representation.** BoxCloud depicts the distances between object points and box points (i.e. corners and the center of a 3D BBox). Box points are arranged in a predefined order. The figure shows the *box coordinate* of an object point p_i. The object and its BBox are transformed to the object coordinate system with x-axis pointing to the left, y-axis pointing to the front and z-axis pointing upward.

3.4. Box-Aware Feature Fusion

The goal of the Box-Aware Feature Fusion (BAFF) module is to generate an enhanced target-specific search area by augmenting the search area with the template, which enables us to locate the target in the search area using a 3D RPN. BAFF can also be regarded as performing a correlation on search area features using the template features as the kernel. BAFF mainly consists of the *BoxCloud Comparison* and the *Feature Aggregation* sub-modules, which are described in details as below.

3.4.1 BoxCloud Comparison

BAFF depends on BoxCloud comparison to conduct a reliable comparison between the template and the search area. Considering the BoxCloud features of the search area are unknown during testing, we first utilize a point-wise MLP to predict the BoxClouds of objects in the search areas under the supervision of ground truth BBoxes.

Given the features of a search area $F^s = \{f_i^s \in \mathbb{R}^d\}_{i=1}^M$ extracted by the backbone network, the MLP takes each feature point f_i^s as input, and outputs its 9-D box coordinate c_i^s. The predicted c_i^s is explicitly supervised by a Huber loss (i.e. smooth-L1 regression loss):

$$\mathcal{L}_{bc} = \frac{1}{\sum_i m_i} \sum_i ||c_i^s - \hat{c}_i^s|| \cdot m_i, \quad (6)$$

where $\{\hat{c}_i^s\}_{i=1}^M \in \hat{C}^s$ are ground truth *box coordinates* from the BoxCloud of the search area pre-calculated before train-

ing. $\{m_i\}_{i=1}^M$ is a binary mask indicating whether the i-th point is inside an object BBox ($m_i = 1$) or not ($m_i = 0$).

As shown in Figure 2 (b), after obtaining the predicted BoxCloud C^s of the search area, we can conduct BoxCloud comparison between the predicted C^s of the search area and the BoxCloud C^t of the template. The BoxCloud comparison is simply based on the pairwise l_2 distance:

$$Dist \in \mathbb{R}^{M_1 \times M_2} = Pairwise(C^t, C^s), \quad (7)$$

where $Pairwise(\cdot, \cdot)$ denotes the pairwise l_2 distance between the two input BoxClouds. M_1 and M_2 are the numbers of points in C^t and C^s respectively. $Dist$ denotes a M_1 by M_2 distance map whose entity $Dist_{i,j}$ represents the distance between c_i^t and c_j^s.

Compared with the feature comparison using the extracted features [27], measuring the shape similarity using BoxCloud is not only more reliable and interpretable but also more efficient. This is because the dimension of a learned feature must be much higher than nine (the dimension of the BoxCloud point) to encode sufficient information. However, a higher dimension inevitably brings additional computational costs.

After that, BAT takes the resulting distance map as a guide to sift out the k most similar template points for each search point. The k-NN grouping is illustrated in Figure 2 (b). BAT selects the top-k most similar template points for each search area point, resulting in an indices matrix where the i-th column contains the indices of the k nearest neighbors of the i-th search point in terms of BoxCloud distance. The k-NN grouping helps to discard false matchings and avoid noise interference from all template points, promoting the subsequent feature aggregation. Besides, using a smaller k helps to gain noticeable speed up, which provides an alternative option for performance and latency trade-off (see Section 4.3).

3.4.2 Feature Aggregation

Feature aggregation aims to fuse the top-k related template features into the search area. For the i-th point p_i^s in the search area, we use the i-th column in the indices matrix to select its k nearest template points. A mini-PointNet [25] is then applied to aggregate these k pairs $\{[p_j^t; f_j^t; c_j^t; f_i^s] \in \mathbb{R}^{2D+3+9}, \forall j = 1, \ldots, k\}$. Here $p_j^t \in \mathbb{R}^3$, $f_j^t \in \mathbb{R}^D$, $c_j^t \in \mathbb{R}^9$ denote the spatial coordinate, extracted feature and *box coordinate* of the j-th template point, respectively. $f_i^s \in \mathbb{R}^D$ is the extracted feature of the corresponding search point. Formally, the mini-PointNet can be formulated as follows:

$$\hat{f}_i^s = \texttt{MaxPool}(\{\texttt{MLP}([p_j^t; f_j^t; c_j^t; f_i^s])\}_{j=1}^k), \quad (8)$$

where $\texttt{MLP}(\cdot)$ is a multi-layer perceptron and $\texttt{MaxPool}(\cdot)$ is the max-pooling across points. Finally, the template fea-

tures are effectively and reliably encoded in the newly acquired target-specific search area $\hat{F}^s = \{\hat{f}_i^s\}_{i=1}^{M_2}$.

3.5. Implementation

Search Area Generation. In practice, the object movement between two consecutive frames is relatively small and hence there is no need to search the target in the whole frame. All we need is to search the target in the neighborhood of the previous object location. We follow this idea to generate the search areas for training and testing. During both training and testing, templates and their BBoxes are transformed to the object coordinate system before being sent to the model.

Training. We generate training samples from any two consecutive frames at the time $t - 1$ and t and use them to train BAT in an end-to-end manner. The template is generated by merging the point clouds inside the first given BBox and $(t - 1)$-th target BBox. We randomly shift the (t)-th target BBox, enlarger it by 2 meters in each direction and collect the points inside to generate the search area. Our loss used for training the RPN is the same as that in [27], denoting it as $\mathcal{L}_{rpn}$, and the final loss for our model is $\mathcal{L} = \mathcal{L}_{bc} + \lambda \mathcal{L}_{rpn}$. We used grid search to tune the λ and finally set it to 1. In fact, changing λ does not affect the performance too much. Our network is trained for 60 epochs using Adam optimizer with a batch size of 96. The learning rate of the network is initialized as 0.001 and is divided by 5 every 12 epochs.

Testing. During testing, we track a given target across all frames in a sequence. This is achieved by applying the trained BAT frame by frame. For the current frame, we update the template by merging the point clouds in the first given BBox and the previous predicted BBox. W e enlarge the previous predicted BBox by 2 meters and collect the points inside to form the new search area.

Architecture. For the model architecture, k is set as 4 in the BAFF module and the Mini-PointNet is constructed with a three-layer MLP, where the channel numbers of hidden layers are identically 256. We randomly sample $N_1 = 512$ points for each template point cloud P^t and $N_2 = 1024$ points for each search area P^s. They are then fed into a PointNet++ (as that in P2B [27]) to obtain corresponding features $F^t \in \mathbb{R}^{128 \times 256}$ and $F^s \in \mathbb{R}^{64 \times 256}$, which will be used for the subsequent processing. All experiments are conducted using NVIDIA RTX-2080 GPUs.

4. Experiments

In this section, we further present our experiment setting and superior results from different aspects.

4.1. Experimental Setting

We conduct extensive experiments on two widely-adopted datasets (*i.e.* KITTI [12] and NuScenes [5]) to val-

Table 1. Performance comparison among our BAT and the state-of-the-art methods on the KITTI (left) and NuScenes (right) dataset, where the instance number of each category is shown under category names. **Bold** denotes the best performance.

	Approach	KITTI					NuScenes				
	Category	Car	Pedestrian	Van	Cyclist	Mean	Car	Truck	Trailer	Bus	Mean
	Frame Number	*6424*	*6088*	*1248*	*308*	*14068*	*32302*	*8646*	*2297*	*2215*	*45460*
Success	SC3D [13]	41.3	18.2	40.4	41.5	31.2	30.6	23.5	27.4	23.6	28.7
	SC3D-RPN [40]	36.3	17.9	-	**43.2**	-	-	-	-	-	-
	P2B [27]	56.2	28.7	40.8	32.1	42.4	34.6	25.2	30.0	28.4	32.3
	BAT(Ours)	**60.5**	**42.1**	**52.4**	33.7	**51.2**	**36.8**	**28.6**	**31.8**	**30.2**	**34.7**
Precision	SC3D [13]	57.9	37.8	47.0	70.4	48.5	35.9	24.8	24.8	21.8	32.5
	SC3D-RPN [40]	51.0	47.8	-	**81.2**	-	-	-	-	-	-
	P2B [27]	72.8	49.6	48.4	44.7	60.0	37.6	25.2	26.7	27.6	34.2
	BAT (Ours)	**77.7**	**70.1**	**67.0**	45.4	**72.8**	**39.5**	**28.4**	**30.5**	**29.5**	**36.4**

idate the effectiveness of the proposed BAT method. Both datasets contain point clouds scanned by LiDAR sensors.

Datasets. For KITTI dataset, it contains 21 outdoor scenes and 8 types of targets. The NuScenes dataset is more challenging, containing 1000 driving scenes across 23 object classes with annotated 3D BBoxes. Furthermore, there are much more objects in a scene, and thus tracking on NuScenes dataset is more challenging since the target is easily submerged by other objects. Since the labels of the test sets are inaccessible, we follow the same settings as in [13, 27] to divide the KITTI training set into training/valid/test splits. We use the training set of NuScenes for training and test on its validation set.

Metrics. For evaluation metrics, we follow [13] to use One Pass Evaluation (OPE) [34] to measure Success and Precision. For a predicted BBox and ground-truth (GT) BBox, "Success" is defined using the AUC for intersection over union between them. "Precision" is defined as AUC for distance between two boxes' centers from 0 to 2m.

4.2. Comparison with State-of-the-arts

We compare our method against SC3D [13], its follow-up [40] and P2B [27]. We do not include [11] in our main table because it does not provide open source codes and only has published results on car category of KITTI, which are 58.4/73.4 in terms of Success/Precision.

Performance across Categories. Table 1 summarizes the results in the KITTI and NuScenes dataset. For the KITTI dataset, we report the published results from corresponding papers. For the NuScenes dataset, since there are no published results, we evaluate [13, 27] using their official open-source codes.

The left part of the table records the results on the KITTI dataset, where BAT shows a significant performance gain over existing methods, outperforming P2B by over 10% on average. For the pedestrian category, BAT even achieves
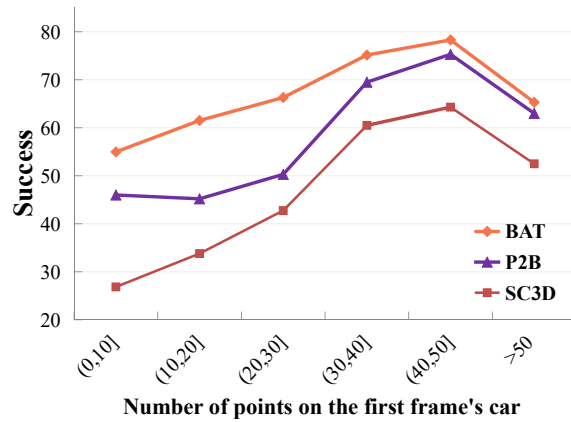

Figure 4. **Robustness test.** Methods are evaluated on sequences split by the number of points in the first frame's car. We use the average Success for each interval (horizontal axis) as the evaluation metric.

about 20% improvement over P2B, which strongly demonstrates the effectiveness of our box-aware tracking pipeline. Moreover, for the van category with only 1248 instances, BAT still achieves satisfactory results, outperforming the other two methods by a large margin. This implies that the additional box information helps to reduce the demand for training data. For the category of cyclist, which has an extremely small amount of training samples, our results still out perform P2B notably. Furthermore, as shown in the right part of Table 1, BAT is consistently superior to the other two competitors on four main classes in NuScenes, which is a more challenging dataset with more instances and object-interference.

Robustness to Sparsity. In realistic applications, objects collected by LiDAR sensors are mostly sparse and incomplete. According to [27], about 34% of cars in KITTI hold fewer than 50 points. Therefore, the robustness against sparsity is an indispensable property for practical trackers.
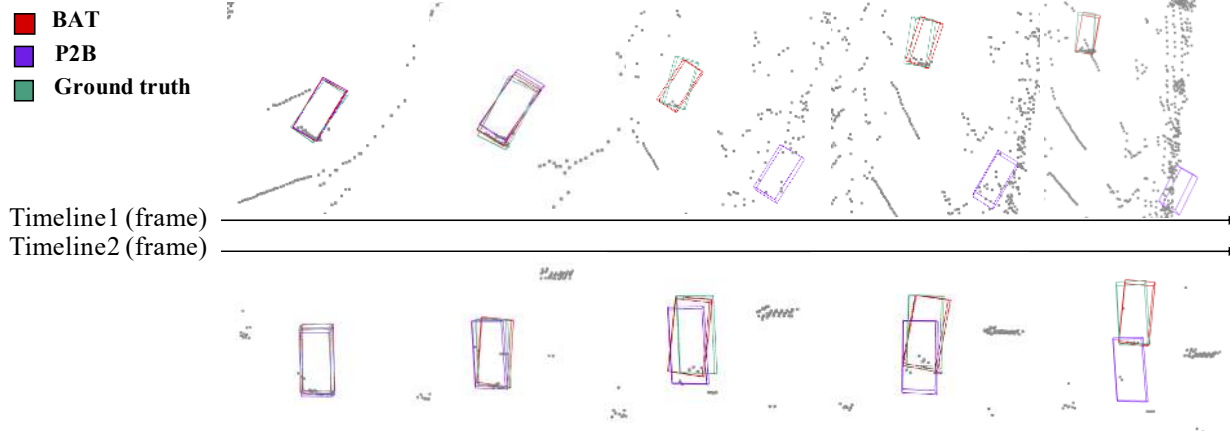
Figure 5. **Advantageous cases of our BAT compared with P2B**. We can observe BAT's advantage over P2B in extremely sparse scenarios.
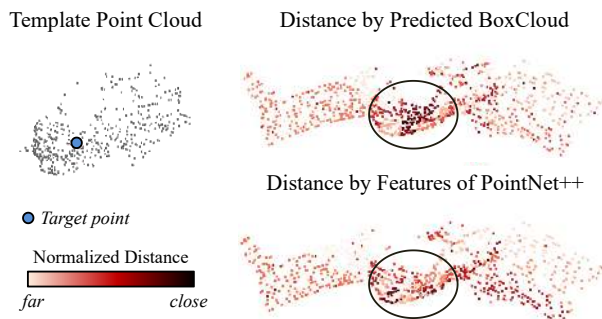

Figure 6. **Pairwise distances measured on extracted features and BoxClouds.** Distances are calculated between each point in the search area and the blue point in the template.

Figure 4 showcases the impressive robustness of BAT. Our method achieves over 55% success rate even when dealing with targets holding less than 10 points. This result is 10% higher than P2B and 30% higher than SC3D. For targets with fewer than 30 points, BAT still holds a huge performance advance over the previous SOTA. Even the performance gap narrows as the number of points approaches 50, our BAT still outperforms P2B by over about 2% success rate. This figure verifies that our BAT works not only for sparse point clouds, but also for dense ones. We observe that the performance of all methods drops with > 50 points, which is counter-intuitive. To clarify this, we show that the average length for sequences with >50 points is much longer than that of sequences with ≤ 50 points (86.3 vs 47.7), making the task much more difficult regardless of more points. The performance drop is mainly due to the accumulated errors when tracking on longer sequences. Figure 5 visualizes BAT's advantage over P2B in extremely sparse scenarios, where BAT's predictions hold tight to the ground truth boxes when P2B tracks off course or even fails.

BoxCloud Comparison v.s. Feature Comparison. Our proposed BoxCloud is a more effective representation compared with features extracted by neural networks. Figure 6 shows a case using PointNet++ to extract the features of the template and search point clouds. It shows that the target points in the search area have relatively low similarities in this case, while the similarities calculated using BoxClouds are much more satisfying. This result strongly proves that BoxCloud is much more reliable for feature comparison.

4.3. Ablation Study

In this section, comprehensive experiments are conducted to validate the design of the BoxCloud and the BAFF module. Behavior analysis of the proposed method is presented as well. All ablated experiments are conducted on the car class of the KITTI dataset as P2B [27].

Our box-aware feature module is specially designed for BoxCloud processing. However, since the BoxCloud provides additional clue for incomplete shapes, directly feeding it to the network must help to improve the performance. To validate this hypothesis, we add an extra branch to the pipeline of P2B, constructing a simple extension called **BAT-Vanilla**. It predicts the BoxCloud of the search area points. The BoxCloud of the template is also concatenated to the spatial coordinates for the feature fusion, while the other parts of P2B remain unchanged. Results in the bottom part of Table 2 confirm that the BoxCloud does help to improve the tracking performance. Specifically, BAT-Vanilla defeats P2B by a noticeable margin. Besides, the last two rows of Table 2 prove the effectiveness of our BAFF module, implying that the BoxCloud comparison and kNN-grouping techniques help to further exploit the useful clues encoded in the BoxClouds.

4.3.1 Analysis on Box-Aware Feature Fusion (BAFF)

The Choice of k. The value of k is a hyper-parameter of BAFF. Figure 7 shows that a smaller k (*i.e.* $k = 2$) helps
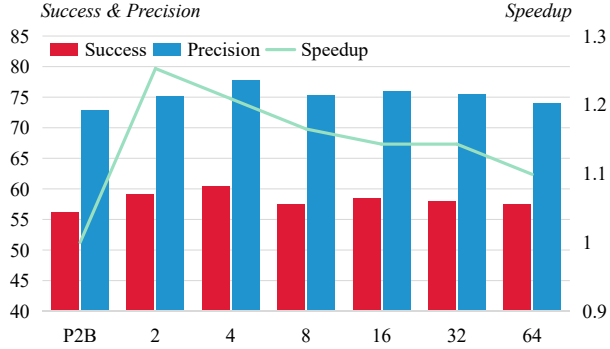
Figure 7. **Performance for different k values.** The left y-axis shows success and precision, while the right illustrates the speedup with respect to P2B.

to speed up our model with little negative impact on the performance. Furthermore, setting a larger k does not improve the performance, which implies that the k-NN grouping in BAFF helps to reject false matching points in the template. With the default setting $k = 4$, BAT achieves the best performance with a satisfactory high speedup, about 20% faster than P2B. Also, it should be noted that BAT is stably superior to P2B under different k values in terms of both performance and speed. Even with $k = 64$ (*i.e.* using all the template points same with P2B), it is still about 10% faster than P2B. Note that here we only consider the time of model forwarding with batch size 96 when calculating the speedup. We do not consider the pre- and post-processing time because these operations are not optimized on GPUs. During the inference where sequences are processed frame by frame, our BAT processes one frame with only 17.5ms on average, achieving 57 FPS.

The Choice of Feature Aggregation. In Eqn. (8) of BAT, the feature we send for aggregation is $[p_j^t; f_j^t; c_j^t; f_i^s]$. In the upper part of Table 2, other settings for the feature aggregation are listed. Firstly, without concatenating the template BoxCloud, our model suffers from a great performance decrease since it does not introduce useful clues to enhance the feature fusion. Furthermore, we observe that adding additional information does not bring any improvement. Specifically, the results of adding the predicted BoxCloud of the search area, pairwise distances in Eqn. (7) and coordinates of the search point cloud all show drastic drops. A possible explanation for this is that such features bring vast redundancies, which may confuse the feature fusion.

Effectiveness of BoxCloud Comparison. To further illustrate the effectiveness of the BoxCloud comparison, two settings are used to validate as follows: 1) *BAT without Box-Cloud Comparison*: We replace BoxCloud with feature (*i.e.* the feature extracted by PointNet++) to compute the distance map. 2) *P2B with BoxCloud Comparison*: Keeping the BoxCloud comparison, we follow the way in P2B [27] for subsequent feature aggregation. As shown in the middle

Table 2. **Results of different ablations.** The upper part shows different concatenation strategies for the feature aggregation sub-module (FA) and the middle part illustrates the effectiveness of the BoxCloud Comparison. As for the lower part, it shows our baseline (P2B), BAT-Vanilla (in Section 4.3) and the best results.

method	Success	Precision
FA without C^t	58.9	75.3
FA with additional C^s	56.7	74.7
FA with additional $Dist$	58.4	75.4
FA with additional P^s	56.7	73.4
BAT without BoxCloud Comparison	57.1	72.9
P2B with BoxCloud Comparison	58.3	74.1
P2B (Baseline)	56.2	72.8
BAT-Vanilla (P2B + BoxCloud)	59.0	74.6
BAT (Ours)	**60.5**	**77.7**

part of Table 2, when using features of PointNet++ in comparison, the performance of BAT drops significantly, which demonstrates that the BoxCloud Comparison is the key of BAFF. While the result of P2B with BoxCloud Comparison implies that a different feature aggregation styles can also benefit from the BoxCloud comparison. The overall results show that the BoxCloud comparison and the BAFF module can benefit from each others.

5. Conclusion

In this paper, we enhance the SOT on LiDAR point clouds by fully exploiting the free bounding-box information. Two main components of our work, the BoxCloud representation and the box-aware feature fusion module, are designed specially for effective object representation and box-aware target-specific search area generation. Each of them independently improves the tracking performance while benefits from each other when they are applied jointly. We extend P2B with the two proposed components, constructing BAT. With such small modifications over P2B, our BAT outperforms the current state-of-the-art methods by a large margin, especially on sparser object tracking. We believe that BoxCloud provides a flexible and powerful tool for future work to enhance their performance.

Acknowledgment

References

[1] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *Eur. Conf. Comput. Vis.*, pages 850–865, 2016. 2

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Int. Conf. Comput. Vis.*, pages 6182–6191, 2019. 2

[3] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *Eur. Conf. Comput. Vis.*, pages 483–498, 2018. 2

[4] Adel Bibi, Tianzhu Zhang, and Bernard Ghanem. 3d part-based sparse tracker with automatic synchronization and registration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1439–1448, 2016. 2

[5] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11621–11631, 2020. 5

[6] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *arXiv preprint arXiv:2001.05673*, 2020. 2

[7] Andrew I Comport, Éric Marchand, and François Chaumette. Robust model-based tracking for robot vision. In *International Conference on Intelligent Robots and Systems*, volume 1, pages 692–697. IEEE, 2004. 1

[8] Kenan Dai, Dong Wang, Huchuan Lu, Chong Sun, and Jianhua Li. Visual tracking via adaptive spatially-regularized correlation filters. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4670–4679, 2019. 2

[9] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6638–6646, 2017. 2

[10] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *Eur. Conf. Comput. Vis.*, pages 472–488. Springer, 2016. 2

[11] Tuo Feng, Licheng Jiao, Hao Zhu, and Long Sun. A novel object re-track framework for 3d point clouds. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 3118–3126, 2020. 2, 6

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3354–3361, 2012. 5

[13] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completion for 3d siamese tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1359–1368, June 2019. 1, 2, 3, 6

[14] Ugur Kart, Joni-Kristian Kamarainen, and Jiri Matas. How to make an rgbd tracker? In *Eur. Conf. Comput. Vis.*, pages 148–161, 2018. 2

[15] Ugur Kart, Alan Lukezic, Matej Kristan, Joni-Kristian Kamarainen, and Jiri Matas. Object tracking by reconstruction with view-specific discriminative correlation filters. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1339–1348, 2019. 2

[16] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4282–4291, 2019. 2

[17] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8971–8980, 2018. 1, 2

[18] 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 Trans. Multimedia*, 21(3):664–677, 2018. 2

[19] Alan Lukezic, Jiri Matas, and Matej Kristan. D3s-a discriminative single shot segmentation tracker. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7133–7142, 2020. 2

[20] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3569–3577, 2018. 1

[21] Eiji Machida, Meifen Cao, Toshiyuki Murao, and Hiroshi Hashimoto. Human motion tracking of mobile robot with kinect 3d sensor. In *2012 Proceedings of SICE Annual Conference (SICE)*, pages 2207–2211. IEEE, 2012. 1

[22] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *IEEE Int. Conf. Robotics Automation*, pages 9552–9557. IEEE, 2019. 2

[23] Alessandro Pieropan, Niklas Bergström, Masatoshi Ishikawa, and Hedvig Kjellström. Robust 3d track-

ing of unknown objects. In *IEEE Int. Conf. Robotics Automation*, pages 2410–2417. IEEE, 2015. 2

[24] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Int. Conf. Comput. Vis.*, pages 9277–9286, 2019. 2, 3

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017. 3, 5

[26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Adv. Neural Inform. Process. Syst.*, pages 5099–5108, 2017. 3, 4

[27] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6329–6338, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[28] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10529–10538, 2020. 2

[29] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–779, 2019. 2

[30] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 2, 3

[31] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1328–1338, 2019. 2

[32] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IROS*, 2020. 2

[33] Hai Wu, Wenkai Han, Chenglu Wen, Xin Li, and Cheng Wang. 3d multi-object tracking in point clouds based on prediction confidence-guided data association. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 2

[34] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2411–2418, 2013. 6

[35] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Joint group feature selection and discriminative filter learning for robust visual object tracking. In *Int. Conf. Comput. Vis.*, pages 7950–7960, 2019. 2

[36] 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. 2

[37] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3101–3109, 2021. 1

[38] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5589–5598, 2020. 1

[39] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2

[40] Jesus Zarzar, Silvio Giancola, and Bernard Ghanem. Efficient bird eye view proposals for 3d siamese tracking. *arXiv preprint arXiv:1903.10168*, 2019. 2, 6

[41] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust multi-modality multi-object tracking. In *Int. Conf. Comput. Vis.*, pages 2365–2374, 2019. 2

[42] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4591–4600, 2019. 2

[43] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Eur. Conf. Comput. Vis.*, pages 101–117, 2018. 2