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

1. Overview

In this supplementary material, we provide more analysis experiments in Section 2. Then we describe the architecture of BAT-Vanilla in Section 3.

2. More Analysis Experiments

Other BoxCloud Design. Our BoxCloud depicts the point-to-box relation using Euclidean distance. An alternative design is replacing the Euclidean distance with the *point-to-point offset*. We denote BoxCloud in this form as $\mathring{B} \in \mathbb{R}^{N\times 27}$, where each row of $\mathring{B}$ is a concatenation of 9 three-dimensional offsets. Compared to our original design, $\mathring{B}$ is three times larger but contains almost the same amount of information as ours. As shown in Table 1, using $\mathring{C}$ incurs a performance decline, which implies that the extra dimension of $\mathring{C}$ puts a burden on the training of the network. In contrast, the proposed BoxCloud is more compact and effective.

Table 1. **Ablation on BoxCloud Designs.**

BoxCloud Designs	Success	Precision
$\mathring{C}$ (offset)	58.8	75.3
C (original)	**60.5**	**77.7**

Performance on Long/Short Term Tracking. We further follow different schemas to generate search areas. 1) To test the long-term tracking performance, we generate all the search areas based on previous results predicted by the models. In this setup, the trackers' ability to handle error accumulation and recover from failure is assessed. 2) For short-term performance evaluation, we use the ground-truth location of the target in the previous frame to generate the next search area. In this case, the tracker does not have to handle the error introduced by its last prediction and only need to focus on the "on time tracking" task.

The results of the car category for all the competitors are shown in Table 2. Overall, our BAT outperforms P2B and SC3D in both setups. In particular, BAT shows more notable superiority in long term setup. This implies that the performance of our method is more stable and robust across time, while the other two methods are more likely to suffer from tracking failure. In the realistic tracking scenario, it is impossible to obtain the "previous ground-truth". Hence, it is more proper to use long-term performance to evaluate a practical tracking system.

Table 2. **Comparison on long/short term tracking performance for car**. The right two columns differ in their ways to generate search area. **Bold** denotes the best performance.

	Method	Long Term	Short Term
Success	SC3D [1]	41.3	64.6
	P2B [2]	56.2	82.4
	BAT(Ours)	**60.5**	**83.5**
Precision	SC3D [1]	57.9	74.5
	P2B [2]	72.8	90.1
	BAT(Ours)	**77.7**	**90.5**

Table 3. **Different ways for template generation**. Methods are compared on the Car category. "First & Previous" denotes "The first GT and Previous result". **Bold** denotes the best performance, and underline shows our default setting.

Source of template	Success			Precision		
	BAT	P2B	SC3D	BAT	P2B	SC3D
The First GT	**51.8**	46.7	31.6	**65.5**	59.7	44.4
Previous result	**59.2**	53.1	25.7	**75.6**	68.9	35.1
First & Previous	**60.5**	56.2	34.9	**77.7**	72.8	49.8
All previous results	**55.8**	51.4	41.3	**71.4**	66.8	57.9

Template Generation Strategy. During the testing, our default setting for template generation is to merge the target in the first frame (the ground truth) with the previous result predicted by the network. For consistent comparison, we further test our method under another three template generation strategies, which uses "the first ground-truth", "the previous result", and "all previous results" respectively to generate the template. The results are listed in Table 3. BAT maintains notable advantages regardless of any strategies. It is worth mentioning that our BAT defeats P2B by the largest margin (~7%) under the "previous result" strategy. This

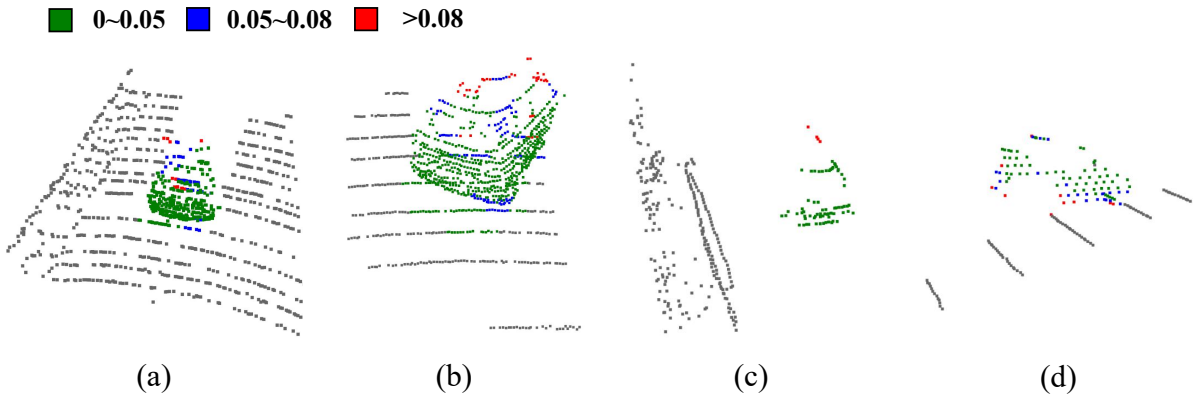

Figure 1. **Visualization of BoxCloud predictions.** 4 *car* cases with different sparsity are presented. Points are colored according to its corresponding BoxCloud predictions MSE errors. The greens are points with MSE errors less than 0.05; The blues are points with MSE errors in the range between 0.05 and 0.08; while the reds denotes points with MSE errors higher than 0.08. It is obvious that most predictions are with small MSE errors.

also shows that our long-term performance is much better and robust than that of P2B.

Table 4. **Comparison with MOT methods on KITTI dataset.**

Class	Method	AB3D [3]	PC3T [4]	BAT
Car	Succ/Prec	37.5 / 42.3	51.9 / 59.2	**60.5 / 77.7**
Ped.	Succ/Prec	17.6 / 27.3	23.6 / 34.1	**42.1 / 70.1**

Comparison with MOT Approaches. To illustrate the superiority of single object tracking (SOT) upon 3D multi-object tracking (MOT) methods, we evaluate 3D MOT methods with 3D SOT metrics. Since multiple objects need to be tracked in 3D MOT, we first find out the corresponding relation between the multiple objects with the single one. Specifically, for each object in SOT, we search its nearest neighbor in all objects of MOT. Hence, each MOT task can be transferred into several SOT tasks, and the metrics of SOT can play a normal role. For each object in MOT, if its tracked identity is changed in a specific frame (the tracker infer the error relation), we will stop the further tracking.

We compare the two most representative methods in 3D MOT, AB3DMOT [3] and PC3T [4], where they rank 25 and 4 in the KITTI MOT leaderboard in the car category, respectively. As shown in Table 4, BAT significantly performs better than state-of-the-art 3D MOT methods. Since multiple objects need to be tracked, they cannot use target-specific feature augmentation to enhance the template representation. Moreover, since they use detection to obtain all objects in the scene, their speed is much slower than ours, where both two MOT methods cannot achieve real-time speed.

Visualization of BoxCloud Learning. BAT is trained to predict the BoxClouds of the points in search areas. In this part, we compare the BoxClouds predicted by our trained BAT with the ground-truths. We use the Mean Square Error (MSE) as the metric to evaluate the performance of BoxCloud learning. For each predicted BoxCloud point, we calculate the MSE between the corresponding ground-truth and the predicted BoxCloud point (only the target points are considered). Figure 2 shows the distribution of MSEs of all the predictions in our KIITI test split. Most of the MSEs are less than 0.1, which implies a high accuracy of the BoxCloud prediction.

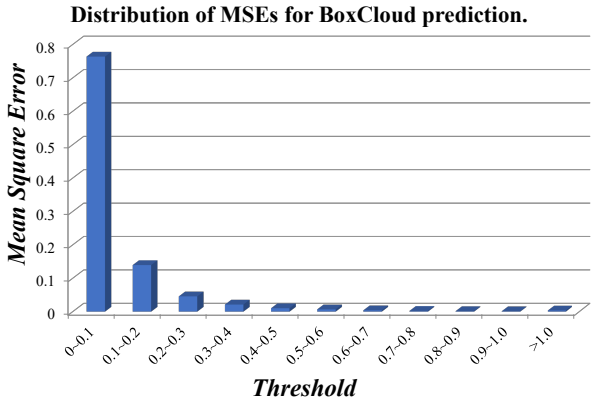

Figure 2. Distribution of MSEs for BoxCloud prediction.

We further visualize several cases of BoxCloud predictions in Figure 1. As shown in figures, our BAT can generally obtain accurate BoxCloud predictions. Some biases may occur in the edges between objects and backgrounds. This is because our training strategy only supervises the BoxCloud of the object rather than the whole search area. Nevertheless, such slight prediction biases have little impact on our BoxCloud comparison and the following tracking process, since they are filtered out by the k-NN grouping.

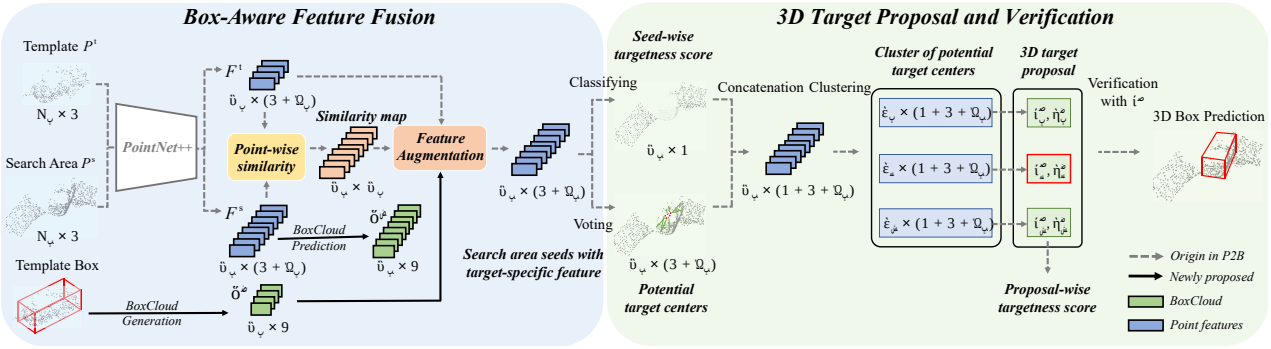

Figure 3. **The overall pipeline of vanilla Box-Aware Tracker (BAT-Vanilla).** The left part is the box-aware feature fusion, which augments the search area with template information. The right part is a VoteNet-based RPN which generates final target proposals from the target-specific search area. For the i-th proposal, s_i^p is its targetness score and p_i^t is its $(x; y; z;\ )$.

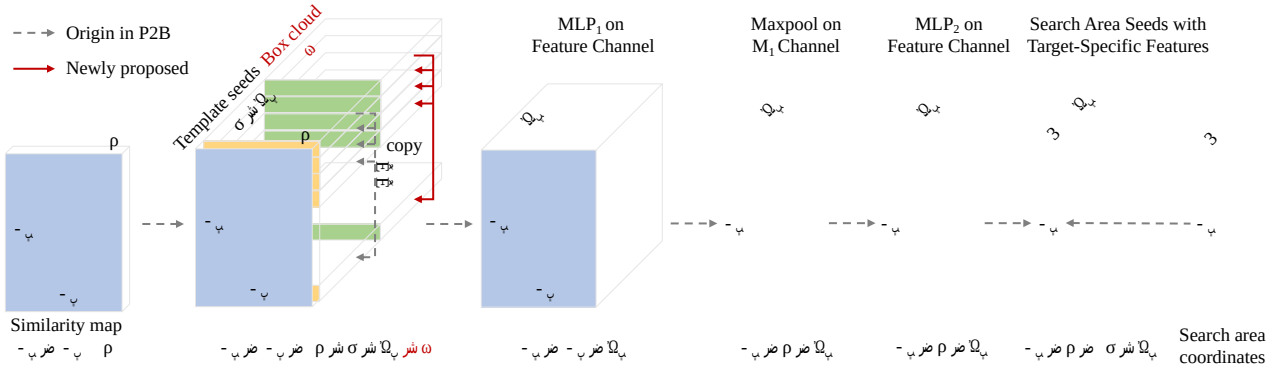

Figure 4. **The detailed architecture of feature augmentation module in the vanilla Box-Aware Tracker (BAT-Vanilla).** The BoxCloud of the template is concatenated together with the corresponding coordinates and the extracted features.

3. Architecture of BAT-Vanilla

The Figure 3 illustrates the architecture of BAT-Vanilla, which directly adds BoxCloud to P2B. After the feature extraction using a shared backbone, a shared MLP (256,256,9) (with layer output sizes 256, 256, 9) is applied to the extracted search area features F^s for BoxCloud prediction. By adding this branch, the F^s is supervised to be box-aware. During the feature augmentation stage, the BoxCloud of the template is simply concatenated together with its coordinates and the extracted features (as shown in Figure 4). The feature augmentation in BAT-Vanilla is almost the same with that in P2B, but introduces additional BoxCloud features as priors.

The right part of Figure 3 illustrates the workflow of the RPN, which is used in both BAT and BAT-Vanilla. A point-wise MLP (256, 256, 3+256) is applied to the target-specific search area features (only the sampled seed points) for object center voting. The MLP takes the per-point feature as input and outputs its offset to the corresponding object center. Besides the coordinate offset (3D), the MLP also predicts a feature offset for each point. But only the coordinate offsets are supervised with the ground-truths. In addition to the voting MLP, another MLP (256, 256, 1) is used to predict a targetness score for each search area seed.

After that, the predicted vote centers and targetness scores are concatenated together and then clustered into k groups through the furthest point sampling and the ball query. Finally, a mini-PointNet is used to produce the final target proposal of each group.

References

[1] 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] 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

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

[4] 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