IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

Accurate 3D Single Object Tracker with Local-to-Global Feature Refinement

Baojie Fan*, Kai Wang*, Wuyang Zhou, Yushi Yang, Kaiwei Ma, Guoping Jiang

***Abstract*—3D single object tracking in point clouds is an essential task in robotics and autonomous driving. Many astonished trackers only adopt the voting-based region proposal network (RPN) to regress the object's location. However, they suffer from heavy outlier votes and ignore the global semantic features of targets. To resolve the problems, we propose a two-stage RPN module with local-to-global feature refinement for accurate tracking in point clouds. Specifically, the deep Hough voting is applied to obtain coarse proposals in the first stage. In the second stage, we design a local feature refinement (LFR) module and a global feature refinement (GFR) module to realize accurate localization jointly. The LFR module excludes noisy outliers in disordered point clouds and obtains refined local features for coarse proposals. After that, the GFR module explores the relationships among all proposals to weigh the proposal-wise global context features. Integrating the proposed two-stage RPN module into the previous method BAT [1], we develop a coarse-to-fine 3D single object tracker in point clouds abbreviated as C2FT. Extensive experiments on KITTI and nuScene benchmarks demonstrate that C2FT achieves favorable performance with a real-time speed (∼50 FPS). Furthermore, the proposed LFR and GFR modules are generalized and can be easily integrated into other trackers.**

***Index Terms*—3D single object tracker, point clouds, local-to-global, siamese network**

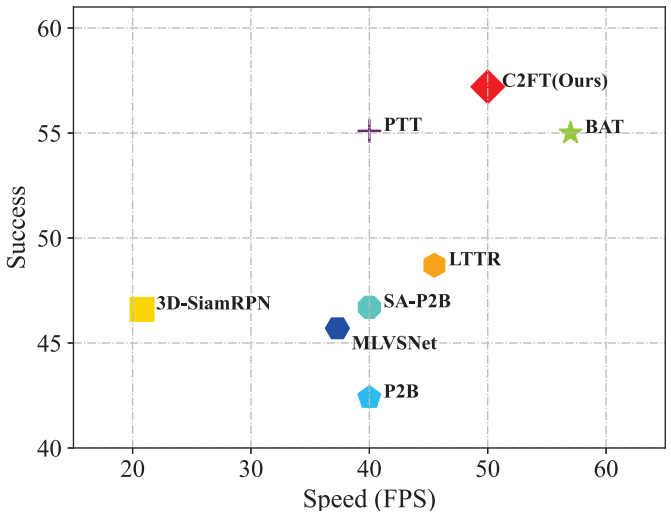

Fig. 1: Comparison with state-of-the-art trackers on KITTI [13] benchmark. We focus on Success and running speed metrics. Our C2FT achieves favorable performance and real-time running speed.

I. Introduction

SINGLE object tracking (SOT) in point clouds is vital to numerous practical applications, such as autonomous driving [2], [3], mobile robots [4], [5], and autonomous drones [6]. LIDAR [7] and cameras [8] are two commonly used sensors in 3D vision tasks. Compared to cameras, LIDAR sensor describes geometric information of 3D objects more accurately and is less sensitive to illumination. Thus, we focus on performing 3D SOT in point clouds scanned by LIDAR sensor. Given the target object's location in the point clouds of the first frame, we aim at predicting the target's location in the point clouds across all frames.

3D SOT in point clouds has its own challenges. For example, point clouds are sparse and irregular. These characteristics limit the direct application of CNNs-based tracking methods in 3D SOT. Some efforts have been made in this field. The pioneering SC3D [9] first introduces the Siamese network into 3D SOT. Inspired by SC3D, P2B [10] utilizes PointNet++ [11] as backbone and adopts a VoteNet [12] based RPN module to generate proposals. BAT [1] shares the same RPN module with P2B but takes advantage of free template bounding box information. In this voting-based RPN module, the proposal's reliability heavily depends on the quality of the votes. However, the point clouds in autonomous driving scenes are extremely sparse and disordered. About 34% cars in the KITTI dataset hold fewer than 50 points, and 51% of the cars hold fewer than 100 points. Simply applying the voting-based RPN suffers from heavy outlier votes and geometric point noise. Furthermore, this RPN module lacks the consideration of contextual information between surrounding objects. It treats all proposals individually and equally, ignoring the contextual information among these proposals.

To address the above problems, we propose a coarse-to-fine 3D single object tracker with local-to-global feature refinement. It achieves favorable performance as shown in Fig.1. Our pipeline is shown in Fig.2. Specifically, PointNet++ [11] is applied as the backbone. The template bounding box is utilized to compute the template BoxClouds. The BoxClouds for search area seeds can be predicted by a multi-layer perception (MLP). We fuse template features and search area features based on BoxClouds comparison. After that, the deep Hough voting is used to obtain coarse proposals. However, the outlier votes greatly affect the accuracy of Hough voting. Therefore, we design a local feature refinement (LFR) module to capture

* Authors with equal contribution.
Baojie Fan, Kai Wang, Wuyang Zhou, Yushi Yang, Kaiwei Ma and Guoping Jiang are with the College of Automation and College of Artificial Intelligence, Nanjing University of Posts and Telecommunications.
Corresponding author: Baojie Fan (email: jobfbj@gmail.com).

local features. The coarse proposals and the target-specific fused seeds are fed to the module. We define key seeds inside proposals that can roughly represent the position and shape of the objects and aggregate their features. Then, we modify the compact generalized non-local network (CGNL) [14] as the global feature refinement (GFR) module to capture contextual information. This module explores the relationships among proposals and enhances their features in a global manner. An MLP layer is applied to make the final predictions based on the refined features.

Overall, our main contributions are as follows:

- We propose an accurate 3D single object tracker in point clouds with local-to-global feature refinement, which is trained in an end-to-end manner.
- We design a local feature refinement (LFR) module to capture local features for coarse proposals, and a global feature refinement (GFR) module to weigh the proposal-wise global context features. They work collaboratively to refine the coarse proposals and can be easily integrated into other trackers.
- Extensive experiments demonstrate that C2FT achieves favorable performance on the widely used KITTI [13] and nuScenes [15] benchmarks.

The rest of this paper is organized as follows. In section II, we introduce the related work. Section III describes the proposed method in detail. In section IV , we validate the superior performance of our tracker on KITTI [13] and nuScenes [15] datasets through extensive experiments. And we conclude in section V.

II. Related Work

This section will briefly discuss the related works in 3D single object tracking, 2D Siamese tracking, and voting-based 3D object detection.

A. 2D Siamese Tracking

Methods based on Siamese network have made rapid development in 2D single object tracking [16]–[23]. These works regard the task as a similarity comparison issue. The pioneering work SiamFC [16] contains two branches with sharing parameters to get the feature map of the template image and the search image. Then, the similarity between these two feature maps is evaluated through cross-correlation operation. SiamRPN [17] introduces the RPN module proposed in Faster R-CNN [24] to regress the bounding box more accurately. Thus, the multi-scale search strategy proposed in SiamFC is no longer required. SiamRPN++ [19] aims to arm Siamese trackers with deeper networks, such as ResNet-50 or deeper networks. SiamMask [20] and D3S [21] devote to finding a unifying approach for both object tracking and segmentation in video sequences. Then, some anchor free trackers [22], [23] regress bounding boxes directly without pre-defined anchor boxes. However, all these methods cannot be directly applied to 3D object tracking in point clouds because typical CNNs are not well suited to process irregularly point clouds.

B. 3D Single Object Tracking

3D single object tracking in point clouds has made rapid developments in recent years. The pioneering work SC3D [9] firstly introduces the Siamese network into 3D SOT and designs an auto-encoder for feature extraction. The cosine similarity is used for similarity comparison. However, SC3D can't be trained in an end-to-end manner and the exhaustive search strategy used for proposals generating in SC3D is quite inefficient. P2B [10] adopts PointNet++ [11] as the backbone and fuses the template seeds and search area seeds to get target specific features. Equipped with a VoteNet [12] based RPN module, P2B can generate proposals more accurately and efficiently. MLVSNet [25] applies a multi-layer voting strategy instead of one-layer voting in P2B. 3D-SiamRPN [3] firstly proposes cross-correlation operation in point clouds for feature fusing and utilizes the RPN module to output the 3D bounding box directly. BAT [1] makes full use of the template 3D bounding box. Specifically, BAT computes the Euclidean distance between every point and box points (eight corners and center), which is called BoxCloud. The BoxCloud is used to measure the shape similarity and get target-specific features. Equipped with the same RPN module, BAT boosts the tracking performance compared to P2B. PTT [26] applies transformer network for 3D single object tracking. It embeds two transformer blocks into P2B to improve the RPN module. LTTR [27] proposes a feature fusion network based on the transformer network. It explores the relationship among all point clouds.

C. Voting-Based 3D Object Detection

VoteNet [12] firstly adapts the classic Hough voting strategy to 3d object detection. It uses PointNet++ [11] to extract point features. Then, the voting module generates a vote for each point that indicates the offset to the object's center. After that, these virtual centers are grouped and aggregated with points nearby to form clusters. At last, the MLP layer is applied to predict the final 3D bounding box. Inspired by VoteNet, ImVoteNet [28] makes full use of image information to help 3D object detection in point clouds. MLCVNet [29] introduces attention mechanism into VoteNet to capture contextual information. Two self-attention modules are utilized to strengthen the seeds feature and cluster features. Then, BRNet [30] adapts the back-tracing strategy in conventional Hough voting and makes a more reliable voting module compared to the original VoteNet. Pointformer [31] proposes a transformer model for point cloud feature learning. The performance improved remarkably after replacing PointNet++ with Pointformer in VoteNet.

III. METHODOLOGY

In 3D single object tracking, the target object's location is usually represented by a 7D vector $\mathbf{V} = [x, y, z, w, l, h, \theta] \in \mathbb{R}^7$. Specifically, $(x, y, z) \in \mathbb{R}^3$ represents the 3D coordinates of the target object center, $(w, l, h) \in \mathbb{R}^3$ represents the width, length, and height of the object, and θ represents the heading angle (X-Y plane). It is worth noting that the size information of the target is given by the template bounding box in the first

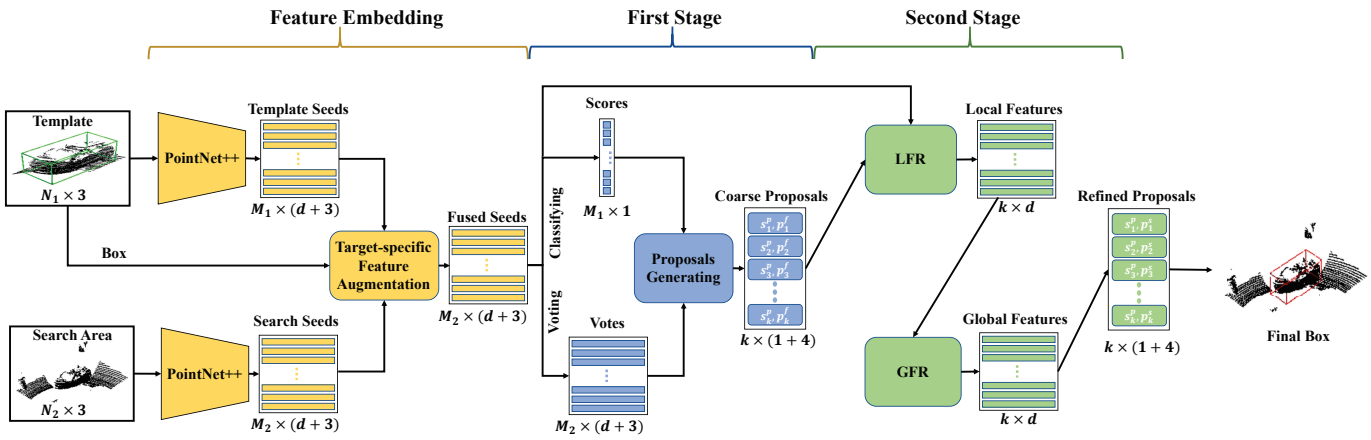

Fig. 2: The main pipeline of our tracker. The network can be divided into three parts: 1) feature embedding part for feature extraction and feature fusion as introduced in III-A. 2) the first stage RPN for coarse proposals generating based on deep Hough voting as introduced in III-B. 3) the second stage RPN for proposals refinement with local feature refinement (LFR) and global feature refinement (GFR) modules as introduced in III-C.

frame. The size of the same object remains unchanged in point clouds across all frames. Thus, our goal is to predict (x, y, z, θ) for a target object. The size information is not needed to be predicted.

Our pipeline is shown in Fig.2. It consists of three main parts: 1) the feature embedding part, 2) the coarse proposals generating part, 3) the proposals refining part. We present the details of these parts in the following sections.

A. Feature Embedding

The feature embedding part is designed for feature extraction and target-specific feature augmentation. We adopt PoineNet++ [11] as the backbone. It has two branches with sharing parameters. The template point clouds $P^t \in \mathbb{R}^{N_1 \times 3}$ and search area point clouds $P^s \in \mathbb{R}^{N_2 \times 3}$ are taken as input. All input point clouds are transformed to the object coordinate system. These two branches extract features $F^t \in \mathbb{R}^{M_1 \times d}$ for the template and $F^s \in \mathbb{R}^{M_2 \times d}$ for the search area. Farthest point sampling (FPS) is used in PointNet++ to sample M_1 seed points for the template and M_2 seed points for the search area. For each point, we concatenate its XYZ coordinates and its feature to obtain a seed represented as $q_i \in \mathbb{R}^{d+3}$. The operations above can be formulated as Eq. 1 and Eq. 2.

$$Q^t = \phi(P^t) \tag{1}$$

$$Q^s = \phi(P^s) \tag{2}$$

where $\phi(\cdot)$ represents feature extraction operation, $Q^t \in \mathbb{R}^{N_1 \times (d+3)}$ is template seeds and $Q^s \in \mathbb{R}^{N_2 \times (d+3)}$ is search area seeds.

After feature extraction, we embed target clues into the search area. BoxClouds $C = \{c_i\}_{i=1}^{N}, c_i \in \mathbb{R}^9$ are proposed which represent the Euclidean distance between object points and box points (eight corners and center). The template BoxClouds can be computed since the template box is given. The search area BoxClouds are predicted through a multi-layer perception layer supervised by a Huber loss L_{bc}. We obtain a distance map after BoxClouds comparison. It indicates the similarity between template seeds and search seeds. Then, four most similar template seeds are selected for each search area seed. A mini-PointNet [32] is utilized to fuse their features. These operations can be summarized as Eq. 3 and Eq. 4.

$$sim = l_2(C^t, C^s) \tag{3}$$

$$Q^f = \varphi(Q^t, Q^s, sim) \tag{4}$$

where $l_2(\cdot)$ denotes l_2 distance, $\varphi(\cdot)$ denotes the feature fusion operation. $C_t \in \mathbb{R}^{M_1 \times 9}$ is template Boxclouds and $C_s \in \mathbb{R}^{M_2 \times 9}$ is search area Boxclouds. $sim \in \mathbb{R}^{M_1 \times M_2}$ represents the distance map between template and search area and $Q_f \in \mathbb{R}^{M_2 \times (d+3)}$ stands for target-specific fused seeds.

B. First Stage: Coarse Proposals Generating

Each fused seed generates a vote $= [\Delta x_i, \Delta f_i]$ through MLPs. The $\Delta x_i \in \mathbb{R}^3$ represents the offset from the seed point to the object center. The $\Delta f_i \in \mathbb{R}^d$ represents the feature offset. A seed-wise targetness score $s_i^s \in \mathbb{R}^1$ is predicted for each seed to regularize feature learning and strengthen the representation of the votes. We random sample k vote centers and gather the neighbor seeds through ball query to obtain clusters. Then, these clusters are sent to MLPs to obtain proposals $T = \{T_i\}_{i=1}^{k}$, $T_i = [s_i^p, p_i^f]$. The $s_i^p \in \mathbb{R}^1$ is the proposal-wise targetness score and $p_i^f \in \mathbb{R}^4$ represents the offset from cluster center to the ground truth object center and rotation in X-Y plane.

The loss function in the coarse proposals generating part can be summarized as Eq.5.

$$L_{fir} = \lambda_1 L_{ss} + \lambda_2 L_{vote} + \lambda_3 L_{sp} + \lambda_4 L_{box} \tag{5}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weights, L_{ss} is a binary cross entropy loss for seed-wise targetness score s_i^s, L_{vote} is a Huber loss for votes, L_{sp} is a binary cross entropy loss for proposal-wise targetness score s_i^p and L_{box} is a Huber loss for p_i^f. Notably, we treat proposals as positive whose center is close to the

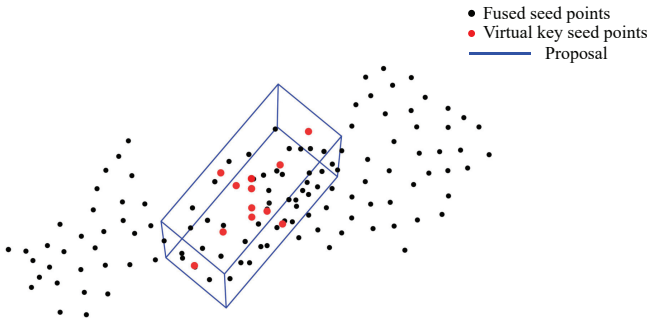

Fig. 3: Illustration of virtual key points generating. Twelve virtual key points are uniformly generated along with six directions from the proposal's center (up, down, left, right, front, back).

center of ground truth (within 0.3 meters) and treat proposals as negative whose center is far away from the center of ground truth (over 0.6 meters). The positive and negative proposals are considered for L_{sp} and only positive proposals are considered for L_{box}.

C. Second Stage: Proposals Refining

We design two sub-modules to refine coarse proposals. The LFR module is designed to capture local features and reduce the impact of noise outliers. Then, the GFR module is proposed to capture extra information among all proposals. With these two modules, we can refine the coarse proposals effectively.

1) Local Feature Refinement: We obtain target-specific fused seeds Q^f after target-specific feature augmentation. These fused seeds are vital to regress the final 3D bounding box. We visualize these seed points in Fig.3. We can find that among these seed points (points in black color), many seed points do not belong to the target object. Furthermore, some of them are far away from the target object. Obviously, the first stage regression based on Hough voting suffers from heavy point noise. Thus, We design a local feature refinement (LFR) module to capture local features for coarse proposals. Given a set of coarse proposals $T = \{T_i\}_{i=1}^{k}$, $T_i = [s_i^p, p_i^f]$, we first define twelve virtual key points $v_i \in \mathbb{R}^3$ for each proposal. To be specific, as shown in Fig.3, with the offset $p_i^f \in \mathbb{R}^4$, we can easily compute the proposal's center and orientation. We define two virtual key points uniformly along one direction from the center to one face of the proposal. Total twelve virtual key points along with six directions (up, down, left, right, front, back). These virtual key points roughly represent the size and orientation of a proposal. However, all these key points are virtual, we need to connect them to actual seeds to get features. We define key seeds $\left\{ q_j^f \;\middle|\; \left\| q_j^f - v_i \right\| \le R \right\}$ and aggregate features of key seeds for a virtual key point. Then all features of 12 virtual key points are fused in a fixed order to obtain local features for a proposal that is less affected by point noise since most outliers are excluded. The operation above can be formulated as Eq. 6 and Eq. 7.

$$V = Gen(T_i) \tag{6}$$

$$F_i^p = \chi(V, Q^f) \tag{7}$$

where $Gen\,(\cdot)$ is the key point generation operation, $V = \{v_i\}_{i=1}^{12}$, $v_i \in \mathbb{R}^3$ is key points set generated for a proposal. χ is the key seeds aggregating operation realized by a MLP layer. F_i^p is the local feature for a proposal.

2) Global Feature Refinement: In P2B [10] and BAT [1], each proposal is processed by a MLP layer independently. However, considering features from other objects brings extra information on the object relationships, which has been proved to be helpful in object detection [33]. In such a way, the final predicted result is not only determined by its own but also affected by object relationships. As shown in Fig.4 (a), it's hard to recognize the pedestrian when we only focus on itself. But it becomes much easier when we focus on the global scene as shown in Fig.4 (b).

Fig. 4: The motivation for the GFR module. It is much easier to recognize and locate the target when we consider the whole scene.

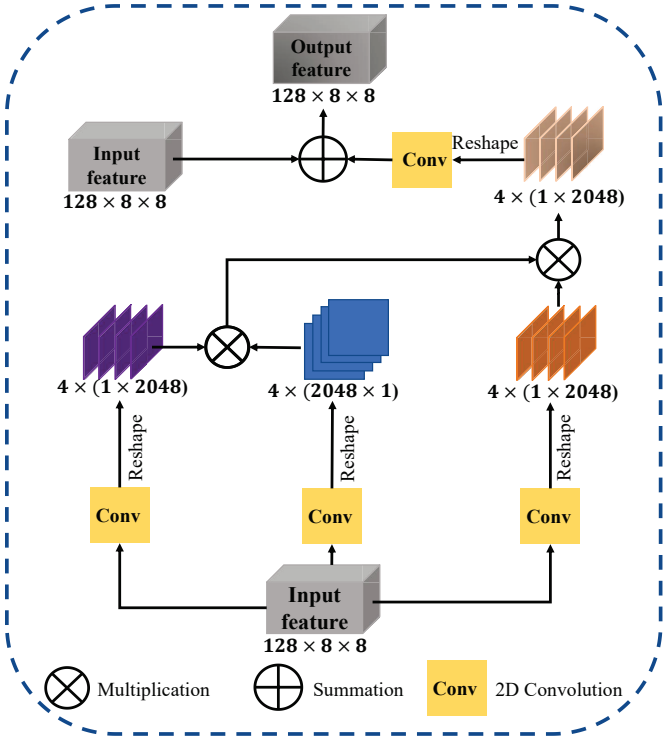

Fig. 5: The structure of the proposed GFR module. A typical self-attention structure is designed to enhance the input features.

We design a global feature refinement (GFR) module to capture proposal-level contextual information. It is built upon the typical self-attention mechanism. The structure is shown clearly in Fig.5. The core operation of this module can be formulated as Eq. 8.

$$F^r = f(\theta(F^p), \psi(F^p))\upsilon(F^p) \tag{8}$$

After local feature refining, we obtain proposal features $F^p \in \mathbb{R}^{128\times64}$. We first reshape it into $F^p \in \mathbb{R}^{128\times8\times8}$. Then we transform it through three 2×2 convolutions ($\theta(\cdot)$, $\psi(\cdot)$,$\upsilon(\cdot)$). We obtain the attention weights by computing the product between features transformed by $\theta(\cdot)$ and $\psi(\cdot)$. This operation is presented by $f(\cdot)$. We use the attention weights to weight features transformed by $\upsilon(\cdot)$. Especially, the correlations of two positions across channels are taken into consideration. The grouping operation in [14] is preserved. We divided channels of input features into 4 groups. We perform Eq. 8 for each group and then concatenate them to obtain enhanced features F^r.

TABLE I: Extensive comparisons with state-of-the-art trackers on KITTI [13] dataset.

	Category	Car	Pedestrian	Van	Cyclist	Mean
	Frame Number	6424	6088	1248	308	14068
Success	SC3D [9]	41.3	18.2	40.4	41.5	31.2
	SC3D-RPN [34]	36.3	17.9	-	**43.2**	-
	P2B [10]	56.2	28.7	40.8	32.1	42.4
	MLVSNet [25]	56.0	34.1	52.0	34.3	45.7
	3D-SiamRPN [3]	58.2	35.2	45.7	36.2	46.6
	BAT [1]	65.4	45.7	52.4	33.7	55.0
	PTT [26]	**67.8**	44.9	43.6	37.2	55.1
	C2FT (Ours)	67.0	**48.6**	**53.4**	38.0	**57.2**
Precision	SC3D [9]	57.9	37.8	47.0	70.4	48.5
	SC3D-RPN [34]	51.0	47.8	-	**81.2**	-
	P2B [10]	72.8	49.6	48.4	44.7	60.0
	MLVSNet [25]	74.0	61.1	61.4	44.5	66.6
	3D-SiamRPN [3]	76.2	56.2	52.9	49.0	64.9
	BAT [1]	78.9	74.5	**67.0**	45.4	75.2
	PTT [26]	**81.8**	72.0	52.5	47.3	74.2
	C2FT (Ours)	80.4	**75.6**	66.1	48.7	**76.4**

TABLE II: Extensive comparisons with state-of-the-art trackers on nuScene [15] dataset.

	Category	Car	Truck	Trailer	Bus	Mean
	Frame Number	64159	13587	3352	2953	84051
Success	SC3D [9]	22.3	30.7	35.3	29.4	24.4
	P2B [10]	38.8	43.0	49.0	33.0	39.7
	BAT [1]	40.7	45.3	52.6	35.4	41.8
	C2FT (Ours)	**40.8**	**48.4**	**58.5**	**40.5**	**42.7**
Precision	SC3D [9]	21.9	27.8	28.1	24.1	23.2
	P2B [10]	43.2	41.6	40.1	27.4	42.2
	BAT [1]	43.3	42.6	44.9	28.0	42.7
	C2FT (Ours)	**43.8**	**46.6**	**51.8**	**36.6**	**44.3**

3) Final Box Verification: We send the enhanced featureF^r to a MLP layer, generating refined proposals $T = \{T_i\}_{i=1}^{k}$, $T_i = [s_i^p, p_i^s]$. p_i^s represents the offset from the center of coarse proposals to the target object center. The refined proposal with the highest proposal-wise targetness score is chosen as the final 3D bounding box. The loss function for the second stage is a Huber loss L_{sec}. Only the refined results of positive proposals in the first stage are considered for L_{sec}.

IV. Experiments

In this section, we present our experiments in detail. First, we briefly introduce the experiment settings. Then, we compare our tracker with other state-of-the-art trackers to demonstrate the favorable performance of our tracker. After that, we conduct ablation experiments to show the effectiveness of different modules in our tracker.

A. Experimental Settings

1) Dataset: We conduct extensive experiments on the widely used KITTI [13] and nuScene datasets [15]. They mainly focus on the autonomous driving scene. KITTI has 21 outdoor scenes and we focus on tracking 4 categories: car, pedestrian, van, and cyclist. We follow SC3D [9] to split the training scene as follows: scenes 0-16 for training, scenes 17-18 for validating, and scenes 19-20 for testing. For each category, tracklets were generated across all frames and the template 3D bounding box is only given in the first frame of a sequence. Compared to KITTI, nuScenes is a more challenging dataset that contains 1000 scenes and 23 classes. We focus on car, truck, trail, and bus categories. We use the training set for training and the validation set for testing. Tracklets are ignored if there are no points in the bounding box of the first frame.

2) Implementation Details: We set the number of template point clouds $N_1 = 512$ and that of search area point clouds $N_2 = 1024$. PointNet++ [11] is modified to contain three set-abstraction layers. Their receptive radius are set as 0.3m, 0.5m, and 0.7m respectively. The backbone generates $M_1 = 64$ target seeds and $M_2 = 128$ search area seeds. We set the dimensions of seed features $d = 256$. $k = 64$ proposals are generated and the dimensions of proposal features d_1 are set to 128 in the GFR module. R in the LFR module for key seeds definition is set to 0.2m. We fuse point clouds centered on the template box and previous results to generate the template area. To generate the search area, we enlarge the previous result by 2 meters and sample point clouds inside it to generate search area. The network is end-to-end trained using Adam optimizer for 60 epochs. The initial learning is 0.001 and decreased by 5 times every 12 epochs. The total loss $L_{total} = \lambda_0 L_{bc} + \lambda_1 L_{ss} + \lambda_2 L_{vote} + \lambda_3 L_{sp} + \lambda_4 L_{box} + \lambda_5 L_{sec}$, we set $\lambda_0 = 1.0, \lambda_1 = 0.2, \lambda_2 = 1.0, \lambda_3 = 1.5, \lambda_4 = 0.2, \lambda_5 = 0.2$ respectively.

B. Comparison With State-of-the-art trackers

1) Performance Comparison: Here we compare C2FT with other state-of-the-art trackers that take point clouds only as input. Comparison results on KITTI dataset and nuScenes dataset are shown in TABLE I and TABLE II. On the KITTI dataset, we make the top performance in both pedestrian and van categories. Compared to our baseline [1], C2FT shows a significant improvement. The mean Success is improved from 55.0 to 57.2 and the mean Precision is improved from 75.2 to 76.4. On the nuScenes dataset, we achieve the best performance in all categories. The mean Success is 0.9 higher and the mean Precision is 1.6 higher compared to BAT. The visualization comparisons are shown in Fig.7 and Fig.9. The shown categories are car, pedestrian, cyclist, van, bus, car,

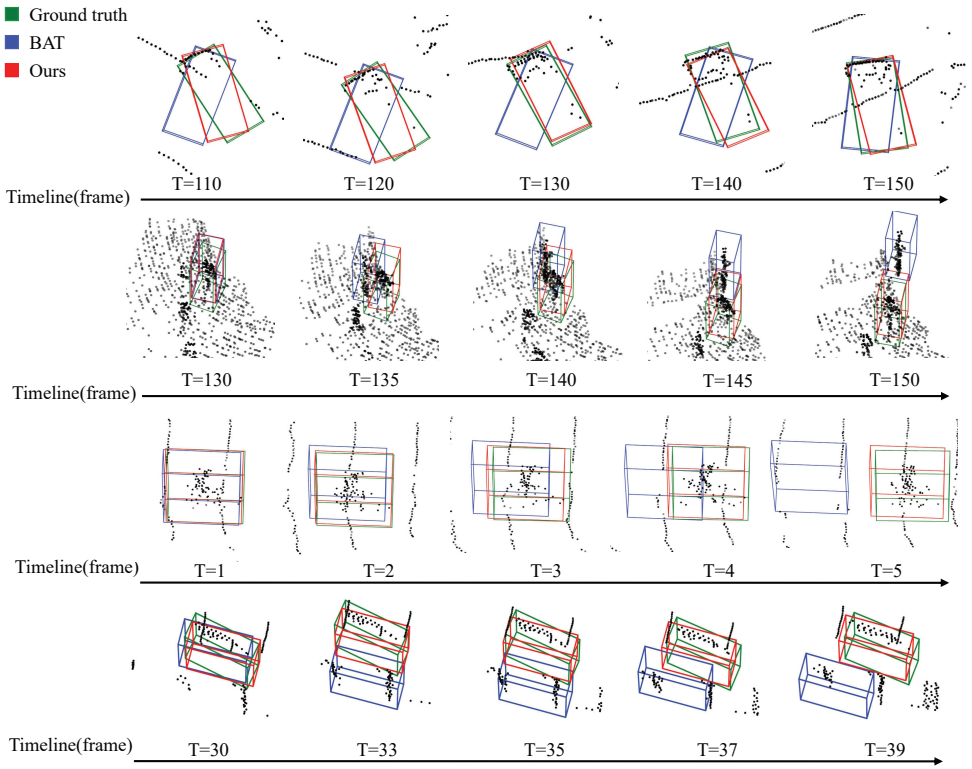

Fig. 6: Advantage cases of our tracker compared to BAT [1] on KITTI.

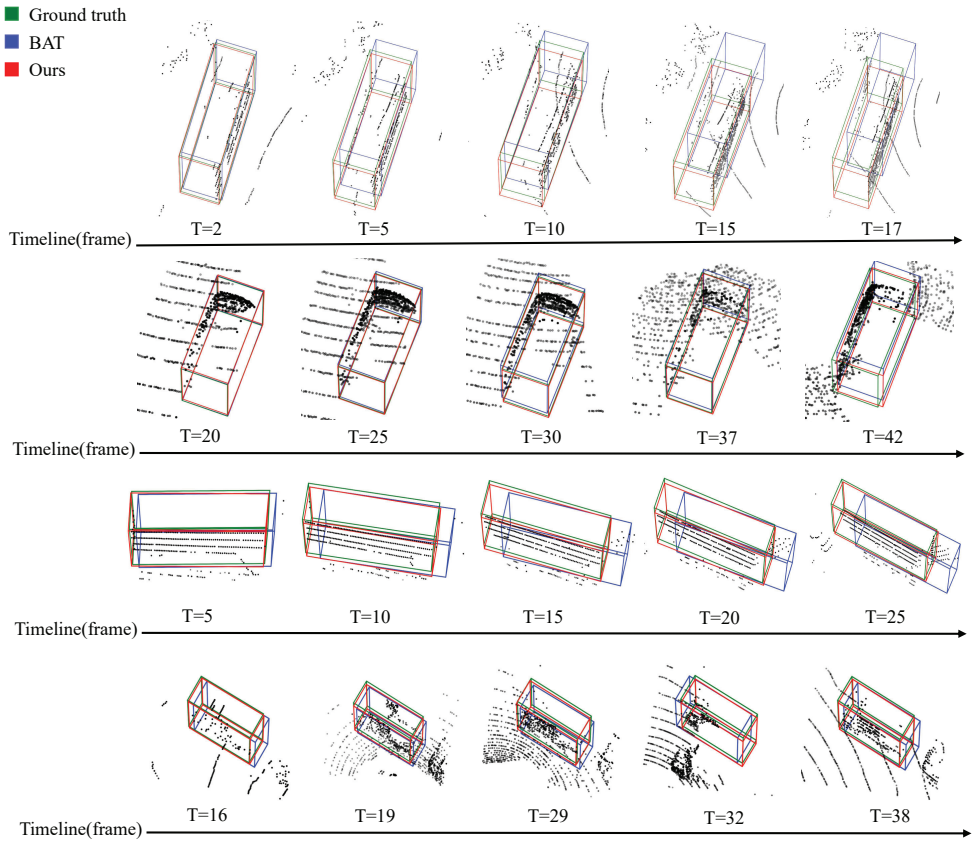

Fig. 7: Advantage cases of our tracker compared to BAT [1] on nuScenes.

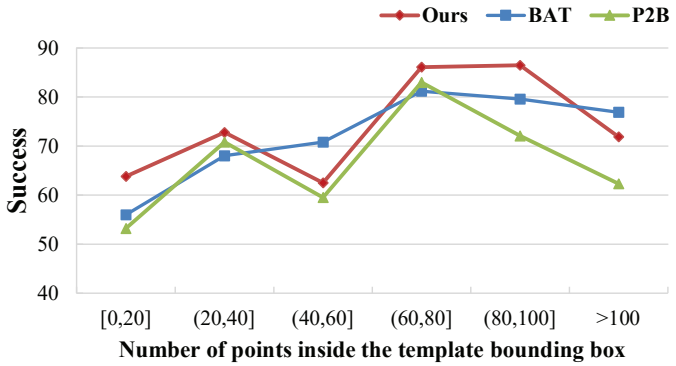

Fig. 8: Comparison with BAT [1] and P2B [10] while there are few informative points in the first frame. The average Success is calculated to reflect the robustness in sparse scenes.

Fig. 9: Visualization in a sparse scene. The green box represents ground truth, and the red box represents the tracking result. In this scene, the target car ID is 63 in scene 4 of the KITTI dataset. The number of points for the target car is about 20.

trailer, and truck from top to bottom. We can find that our tracker holds the target tightly when BAT is more likely to lose the target.

2) Robustness in Sparse Scenes: The number of informative template points clouds greatly affects the performance of the 3D single object trackers. In KITTI [13] dataset, We split 120 sequences in car category into six intervals according to the number of points inside the template bounding box. Success is the evaluation metric. As shown in Fig.8, in extremely sparse scenes with $\leq$ 40 points, C2FT performs better than other state-of-the-art trackers. Furthermore, C2FT takes the lead in four intervals among all five intervals with $\leq$ 100 points. We visualize the tracking process in a sparse scene in Fig.9. It can be seen that C2FT performs well in such a challenging scenes.

3) Running Speed: We first calculate the time to test all frames in car category in KITTI and then average it by total frame numbers. We only consider the time for network forward propagation. As a result, C2FT takes 19.9ms to process a frame, achieving 50FPS approximately on a single NVIDIA 2080Ti GPU. It means that C2FT runs at a real-time speed. Our baseline BAT [1] achieves 57FPS as mentioned in their published results. C2FT is only 7 FPS slower compared to BAT but more precise.

C. Ablation Experiments

1) Two Stages: We design a two-stage RPN module to regress a more accurate 3D bounding box. To validate the effectiveness of the second stage, we choose proposals in different stages with the highest targetness score as the final prediction. The results in the car category on KITTI are shown in TABLE III. The results of the second stage outperform that of the first stage significantly. This indicates the effectiveness of the second stage we designed.

TABLE III: Results in different stages.

	Coarse proposals	Refined proposals
Success	65.6	67.0
Precision	**78.4**	**80.4**

TABLE IV: Results after equipping BAT [1] with different modules.

	Category	Car	Pedestrian	Van	Cyclist	Mean
Success	BAT	65.4	45.7	52.4	33.7	55.0
	+LFR	66.2	46.1	52.6	35.8	55.6
	+LFR+GFR	**67.0**	**48.6**	**53.4**	**38.0**	**57.2**
Precision	BAT	78.9	74.5	**67.0**	45.4	75.2
	+LFR	79.2	74.1	66.0	46.4	75.1
	+LFR+GFR	**80.4**	**75.6**	66.1	**48.7**	**76.4**

2) The LFR Module and GFR Module: We take BAT [1] as our baseline. We equip our baseline with the LFR module and the GFR module to test their effects. The results are shown in TABLE IV. It can be seen that both the LFR module and GFR module bring significant improvements. These two modules totally improve mean Success from 55.0 to 57.2, and mean Precision from 75.2 to 76.4.

The number of virtual key points in the LRF module is crucial. If the number setting is too small, some key information will be lost. If the number setting is too large, point noise will be introduced. We do experiments to verify the performance with different numbers, the results are shown in TABLE V. We obtain the best performance when the number is set to 12.

TABLE V: Results of setting different numbers of virtual points in the car category on KITTI.

Number of virtual points	Success	Precision
6	64.2	75.4
12	**67.0**	**80.4**
18	64.1	77.9

3) Generalization: To validate the generalization of the proposed LFR and GFR modules, we also integrate it into P2B [10]. The results are shown in TABLE VI. We can find that the two proposed modules bring more significant improvement to P2B. Especially in the pedestrian category, the Success improved from 28.7 to 35.2 and the Precision improved from 49.6 to 59.3.

TABLE VI: Results after equipping P2B [10] with different modules.

	Category	Car	Pedestrian	Van	Cyclist	Mean
Success	P2B	56.2	28.7	40.8	32.1	42.4
	+LFR	57.4	33.5	44.6	34.3	45.4
	+LFR+GFR	**58.2**	**35.2**	**46.1**	**36.2**	**46.7**
Precision	P2B	72.8	49.6	48.4	44.7	60.0
	+LFR	73.4	58.6	52.4	47.3	64.6
	+LFR+GFR	**73.9**	**59.3**	**54.5**	**48.0**	**65.3**

V. Conclusion

In this paper, we propose a coarse-to-fine 3D single object tracker in point clouds. We aim at improving the voting-based RPN module in other state-of-the-art trackers, such as P2B and BAT. After integrating the proposed LFR and GFR module into BAT, we construct a superior tracker called C2FT. Experiments on the KITTI and nuScenes benchmarks show that C2FT achieves outstanding performance and runs at a real-time speed. We expect that this work will inspire others to explore more appropriate RPN modules for 3D SOT in point clouds. Furthermore, incomplete point clouds heavily limit the tracking performance, and the RGB images provide rich texture details to complement point clouds. We will devote to fusing RGB images and point clouds to improve the tracking performance in 3D scenes.

References

[1] C. Zheng, X. Yan, J. Gao, W. Zhao, W. Zhang, Z. Li, and S. Cui, "Box-aware feature enhancement for single object tracking on point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 199–13 208.

[2] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[3] Z. Fang, S. Zhou, Y. Cui, and S. Scherer, "3d-siamrpn: An end-to-end learning method for real-time 3d single object tracking using raw point cloud," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4995–5011, 2020.

[4] A. I. Comport, É. Marchand, and F. Chaumette, "Robust model-based tracking for robot vision," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1. IEEE, 2004, pp. 692–697.

[5] E. Machida, M. Cao, T. Murao, and H. Hashimoto, "Human motion tracking of mobile robot with kinect 3d sensor," in *2012 Proceedings of SICE Annual Conference (SICE)*. IEEE, 2012, pp. 2207–2211.

[6] J. Ye, C. Fu, Z. Cao, S. An, G. Zheng, and B. Li, "Tracker meets night: A transformer enhancer for uav tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3866–3873, 2022.

[7] S. Wang, P. Cai, L. Wang, and M. Liu, "Ditnet: End-to-end 3d object detection and track id assignment in spatio-temporal world," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3397–3404, 2021.

[8] Y. Liu, Y. Yixuan, and M. Liu, "Ground-aware monocular 3d object detection for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 919–926, 2021.

[9] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3d siamese tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1359–1368.

[10] H. Qi, C. Feng, Z. Cao, F. Zhao, and Y. Xiao, "P2b: Point-to-box network for 3d object tracking in point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6329–6338.

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[12] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.

[13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[14] K. Yue, M. Sun, Y. Yuan, F. Zhou, E. Ding, and F. Xu, "Compact generalized non-local network," *arXiv preprint arXiv:1810.13125*, 2018.

[15] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[16] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *ECCV 2016 Workshops*, 2016, pp. 850–865.

[17] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8971–8980.

[18] D. Gordon, A. Farhadi, and D. Fox, "Re^3: Re al-time recurrent regression networks for visual tracking of generic objects," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 788–795, 2018.

[19] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.

[20] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1328–1338.

[21] A. Lukezic, J. Matas, and M. Kristan, "D3s-a discriminative single shot segmentation tracker," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7133–7142.

[22] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, "Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 12 549–12 556.

[23] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese box adaptive network for visual tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6668–6677.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[25] Z. Wang, Q. Xie, Y.-K. Lai, J. Wu, K. Long, and J. Wang, "Mlvsnet: Multi-level voting siamese network for 3d visual tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3101–3110.

[26] J. Shan, S. Zhou, Z. Fang, and Y. Cui, "Ptt: Point-track-transformer module for 3d single object tracking in point clouds," *arXiv preprint arXiv:2108.06455*, 2021.

[27] Y. Cui, Z. Fang, J. Shan, Z. Gu, and S. Zhou, "3d object tracking with transformer," in *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*. BMVA Press, 2021, p. 317.

[28] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, "Imvotenet: Boosting 3d object detection in point clouds with image votes," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4404–4413.

[29] Q. Xie, Y.-K. Lai, J. Wu, Z. Wang, Y. Zhang, K. Xu, and J. Wang, "Mlcvnet: Multi-level context votenet for 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 447–10 456.

[30] B. Cheng, L. Sheng, S. Shi, M. Yang, and D. Xu, "Back-tracing representative points for voting-based 3d object detection in point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8963–8972.

[31] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, "3d object detection with pointformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7463–7472.

[32] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[33] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3588–3597.

[34] J. Zarzar, S. Giancola, and B. Ghanem, "Efficient bird eye view proposals for 3d siamese tracking," *arXiv preprint arXiv:1903.10168*, 2019.