A Lightweight and Detector-free 3D Single Object Tracker on Point Clouds

Yan Xia, Qiangqiang Wu, Wei Li, Antoni B. Chan *Senior Member, IEEE*, Uwe Stilla *Senior Member, IEEE*

***Abstract*—Recent works on 3D single object tracking treat the task as a target-specific 3D detection task, where an off-the-shelf 3D detector is commonly employed for the tracking. However, it is non-trivial to perform accurate target-specific detection since the point cloud of objects in raw LiDAR scans is usually sparse and incomplete. In this paper, we address this issue by explicitly leveraging temporal motion cues and propose DMT, a Detector-free Motion-prediction-based 3D Tracking network that completely removes the usage of complicated 3D detectors and is lighter, faster, and more accurate than previous trackers. Specifically, the motion prediction module is first introduced to estimate a potential target center of the current frame in a point-cloud-free manner. Then, an explicit voting module is proposed to directly regress the 3D box from the estimated target center. Extensive experiments on KITTI and NuScenes datasets demonstrate that our DMT can still achieve better performance (∼10% improvement over the NuScenes dataset) and a faster tracking speed (i.e., 72 FPS) than state-of-the-art approaches without applying any complicated 3D detectors. Our code will be released publicly.**

***Index Terms*—point clouds, detector-free, explicit voting, 3D single object tracking.**

I. Introduction

SINGLE object tracking (SOT) is a key task in the field of computer vision, which has wide downstream applications in outdoor and indoor scenarios, ranging from autonomous driving [1], [2], robot vision [3]–[6], and intelligent transportation systems [7]. For example, an autonomous pedestrian-following robot should accurately track its master for efficient crowd-following control. Another example is autonomous landing by unmanned aerial vehicles, in which the drone must track the target and know the exact distance and pose of the target in order to land safely [8]. In indoor environments, tracking methods [5], [6], [9] can provide the six-degrees-of-freedom (6DoF) pose of an object for robust robotics manipulation. Given an initial bounding box of a template object in the first frame from images or LiDAR scans, the aim of SOT is to estimate its location by identifying the trajectory across all frames. In the past decade, a variety of image-based trackers (e.g., Siamese neural networks [10]) have shown promising performance in the 2D tracking community. However, the performance of image-based methods often suffers in degraded situations, e.g., when facing drastic lighting changes [11], [12]. As a possible remedy, 3D point clouds collected from LiDAR provide detailed depth and geometric information, which is inherently invariant to lighting changes [13], making it more robust when tracking across frames taken from different illumination environments.

The main challenges of learning-based approaches for 3D SOT trackers are four-fold: 1) a point cloud is structurally unordered compared with images, and thus the network must be permutation-invariant [11]; 2) a point cloud is incomplete because of occlusion or self-occlusion, and thus the network must be insensitive to different resolutions of input point clouds [14]; 3) the scanned point clouds of different objects might have quite similar shapes [15], and thus the network must be insensitive to shape ambiguities; 4) a point cloud has an unstructured nature and thus applying the convolutional operation is difficult [16].

In 3D SOT, the typical solutions follow a Siamese network-based methodology, i.e., comparing the feature similarity between some search regions and the template object. SC3D [17] is a pioneering 3D tracker, which first enriches geometric features from sparse point clouds using a shape completion network [18], and then executes template matching with target proposals generated by Kalman filtering [19]. However, SC3D is not an end-to-end network and also cannot meet the real-time requirement. To address these concerns, P2B [12] first calculates the point-based correlation between the template and the search area, and then applies a Siamese region proposal network (RPN) [20] to detect the final target proposal. Following this, BAT [15] explores the free box information to enhance the target-specific search feature. MLVSNet [21] proposes performing voting on multi-level features to get more vote centers. With breakthroughs in transformer-based vision methods, the authors of PTT [11] introduce a transformer module to further refine the target-specific search feature. These methods all use historical information to decide the search area, sample seeds in an implicit strategy, and then apply the RPN module (VoteNet [22]) to detect the target in the search space. Although this improves search results, the usage of the RPN module is still complicated and burdensome on the whole. Furthermore, the previous 3D trackers ignore one key point: *the coarse target center in the current frame can be directly predicted in a point-cloud-free way by explicitly exploring the historical information. The predicted center can further serve as strong prior knowledge for the final 3D target box prediction.*

To fully utilize this prior knowledge, we propose a novel

Manuscript received...

Yan Xia and Uwe Stilla are with the School of Engineering and Design, Technical University of Munich, 80333 Munich, Germany (e-mail: yan.xia@tum.de, stilla@tum.de).

Qiangqiang Wu and Antoni B. Chan are with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: qiangqwu2-c@my.cityu.edu.hk, abchan@cityu.edu.hk).

Wei Li is with the X-Lab of Inceptio, Shanghai, China (e-mail: li-weimcc@gmail.com)

Yan Xia and Qiangqiang Wu have equal contribution.

lightweight and detector-free 3D single object tracking network named DMT (Detector-free Motion prediction-based 3D Tracking). Specifically, we first develop a motion prediction module to estimate the 3D coordinates of a potential target center in the current frame using previous frames. Although the estimated center is coarse, it can provide strong prior information as guidance. Thus, we further design an explicit voting layer only consisting of several multi-layer perceptron (MLP) layers to refine the target center with the desired position and rotation.

To summarize, the main contributions of our work are:

- To the best of our knowledge, we are the first to completely remove the usage of complicated 3D detectors or proposal generation in 3D single object trackers. We demonstrate that object motion is a useful cue in 3D SOT, which permits less complex tracking models while still achieving state-of-the-art performance. Our method can serve as a simple yet strong baseline in the 3D SOT community.
- We propose a new lightweight and detector-free 3D single object tracking network based on motion prediction, called DMT, and purely applies to point clouds. With the guidance of center priors, an explicit voting module only consisting of several MLP layers is designed to generate accurate 3D positions and the rotation in the X-Y plane.
- We conduct experiments on the KITTI [23] and NuScenes [24] benchmark datasets to demonstrate the superiority of DMT over other state-of-the-art 3D SOT methods. Notably, the performance on the NuScenes datasets achieves a $\sim$10% improvement on average, while running faster and lighter than the previous state-of-the-art methods.

II. Related Work

The goal of object tracking is to locate the object in successive frames using raw data collected from various sensors, which can be 2D images or 3D point clouds. Numerous methods for tracking objects in 2D or 3D spaces have been developed, which are divided into two categories based on the different data.

A. 2D Single Object Tracking

2D SOT is a basic computer vision task with a long history spanning decades. The representative deep tracking framework is built on deep Siamese networks [25]. The pioneering work is SiamFC [25], which treats visual tracking as a general template-matching problem and performs favorably in terms of both tracking performance and speed. Based on SiamFC, many improvements have been proposed. SiamDW [26] adopts very deep neural networks (e.g., ResNet [27]) as the backbone for Siamese tracking. To handle large-scale appearance variations, SiamRPN [28] and SiamRPN++ [29] employ region proposal networks (RPNs) for scale regression. In addition, much effort is being made to build a robust target appearance model, including UpdateNet [30], MemTrack [31], and DSiam [32]. Kim *et al*. [33] presents a strong discriminative appearance model via a novel pooling module. Recent progress on 3D SOT (e.g., P2B [12] and BAT [15]) follows a bounding box regression-based framework, which is mainly inspired by the 2D tracker SiamRPN. However, the data source in 3D tracking is totally different from the images used in 2D tracking. Directly regressing target bounding boxes is still limited when the scanned point clouds are sparse. In this work, we alleviate this problem by incorporating temporal and spatial tracking information for bounding box regression.

Motion prediction has also been well explored in 2D object tracking in videos. However, 2D motion prediction is generally unreliable due to the scale changes, perspective effects, and inconsistent motion caused by viewing a 2D projection of an object moving in a 3D scene. Indeed, most modern deep trackers use a simple learning-free motion prior (e.g., cosine window in SiamFC), and rely on the more reliable 2D appearance features. There are a few 2D trackers that use the motion module to *assist* with object detection, e.g., motion-conditioned detection [34]–[36] for associating objects in consecutive frames and motion-guided multiple proposal generation [37], [38]. Notably, these trackers still require an object detector module (e.g., RPN) performing on a per-frame basis. In contrast to 2D SOT, we show that motion cues in 3D point cloud tracking are more reliable and can be exploited to build lightweight trackers that do not use complex detectors, while still achieving state-of-the-art performance.

B. 3D Single Object Tracking

Early 3D SOT methods [39]–[42] generally rely on the RGB-D information and employ the 2D Siamese tracking architecture. Though these methods are effective in certain situations, they do not fully explore 3D geometric clues. SC3D [17] is a pioneering work for point-cloud-based tracking, which regularizes the latent spaces of the template point cloud and search candidates using a shape completion network. However, this method is time-consuming since it uses Kalman filtering for the target proposal generation. Moreover, it ignores the local geometric information of each target proposal. PSN [43] leverages 3D Siamese network for single-person tracking. However, it cannot predict the orientation and size of the target. F-Siamese tracker [44] explores RGB images to produce 2D region proposals to reduce the 3D point cloud searching space. However, its performance depends more on the 2D tracker. 3DSiamRPN [45] combines a 3D Siamese network and a 3D RPN to track a single object, but the one stage RPN network limits its performance. P2B [12] fuses the target object information into 3D search space and then adopts a state-of-the-art object detection network (VoteNet) to detect the target. Following this, BAT [15] proposes adding the bounding box information provided in the first frame as an additional cue. MLVSNet [21] performs Hough voting on multi-level features to get more vote centers. PTT [11] introduces the transformer architecture to enhance the target-specific feature extracted in P2B. However, these methods all use the RPN to regress the bounding box of the target, which is inspired by their 2D SOT counterparts [26], [28], [29]. In this paper, we show that complex detectors can be removed from 3D SOT by better leveraging more reliable 3D motion prediction, and still achieving state-of-the-art performance.

III. Problem Statement

Let $B_{init} = \{x, y, z, h, w, l, \theta\}$ be a known 3D bounding box of the object in the first frame, where (x, y, z) are the center coordinates of the 3D bounding box, (h, w, l) are the height, width, and length respectively, and θ is the orientation of the bounding box. Further, let $Q = \{Q_i\}_{i=1}^{M}$ be a query point cloud created by cropping and centering the object in the first frame with B_{init}. Q_i is a 3D point in the Q. We define the single object tracking task as locating the same object in the search point cloud $P = \{P_i\}_{i=1}^{N}$ given the B_{init} frame by frame. M and N are the number of points in the query point cloud and search point cloud, respectively. Formally, previous state-of-the-art 3D single object trackers [12], [15] can be formulated as:

$$Tracker\left(Q, P, B_{init}\right) \rightarrow (\hat{x}, \hat{y}, \hat{z}, \hat{\theta}), \tag{1}$$

where $Q \in \mathbb{R}^{M\times 3}$, $P \in \mathbb{R}^{N\times 3}$, and $B_{init} \in \mathbb{R}^{7}$. Notably, we only predict the center coordinates and orientation $(\hat{x}, \hat{y}, \hat{z}, \hat{\theta})$ of the target since the height, width, and length of the object are assumed to be the same in other frames.

Previous trackers employ off-the-shelf detectors on scanned point clouds for target detection. They may easily drift when the point clouds are relatively sparse or incomplete. In this paper, we propose predicting the potential target center in a point-cloud-free way, that fully explicitly leverages motion cues from previous target states $S_{prev} = \{S_1, S_2, \cdots, S_{t-1}\}$, where the state S_t is the predicted center coordinates in the t-th frame. The whole process is formulated as:

$$Tracker\left(Q, P, B_{init}, \mathcal{M}(S_{prev})\right) \rightarrow (\hat{x}, \hat{y}, \hat{z}, \hat{\theta}), \tag{2}$$

where $\mathcal{M}(\cdot)$ is a motion prediction function that estimates a potential target center in the current frame based on previous target states.

IV. Methodology

The overall network architecture of our DMT is shown in Fig. 1. Given the query and search point cloud with coordinates denoted as Q and P, and an initial bounding box B_{init}, we first use the backbone to extract target-specific features following [15], as introduced in Section IV-A. Unlike previous studies, we propose a motion prediction module to estimate a potential target center in the current frame based on previous target states S_{prev}, with details described in Section IV-B. Afterward, an explicit voting module is adopted to modify the coordinates of the coarse predicted center and predict the orientation in Section IV-C. The loss function is presented in Section IV-D. The training strategy and implementation details are explained in Section IV-E. To highlight the simplicity of our method, we also sketch the detailed flow in Algorithm 1.

A. Backbone

The aim of the backbone network is to generate an enhanced target-specific search feature by fusing the template's target information into the search area points. We adopt the box-aware feature fusion (BAFF) module in [15] as our backbone[1], as shown in Fig. 3. The template and search area are first fed into PointNet++ [46] to obtain their features. Then the BAFF module help augment the search area with target-specific features, which includes BoxCloud [15] comparison and feature aggregation sub-modules. A BoxCloud is defined by the point-to-box relation between an object point cloud and its 3D bounding box. For each point p_i in this point cloud, nine Euclidean distances from the p_i to each of the eight corners and the center of the bounding box are calculated. As shown in Fig. 2, every point is represented by a 9D vector c_i. Formally, a BoxCloud C_{bc} can be formulated as follows:

$$C_{bc} = \left\{c_i \in \mathbb{R}^9 \mid c_{ij} = \left\|p_i - q_j\right\|_2, \quad \forall j \in [1, 9]\right\}_{i=1}^{N}, \tag{3}$$

where $q_{j(j\neq 9)}$ is the j-th corner and j_9 is the center of the bounding box.

BoxCloud comparison. Given the feature of a search area $F_s = \{f_i^s\}_{i=1}^{M_1}$ obtained by PointNet++, we predict the 9D BoxCloud coordinates $C_{bc}^s = \{c_i^s \in \mathbb{R}^9\}_{i=1}^{M_1}$ from each point feature f_i^s via MLP, where M_1 is the number of points in C_{bc}^s. The prediction is supervised by a BoxCloud loss, presented in Sec. IV-D. Then we compare the pairwise distance between the predicted C_{bc}^s and the BoxCloud $C_{bc}^t = \{c_i^t\}_{i=1}^{M_2}$ of the template, as shown in Fig. 3, where M_2 is the number of points in C_{bc}^t. Following [15], we adopt the simple l_2 distance as the distance metric. After obtaining the distance map, we sort and select the top k most similar template points for each point in the search area. The i-th column of the distance map in Fig. 3 represents the indices of the k nearest neighbors of the i-th search point p_i^s.

Feature aggregation. After getting the top k template features, we hope to fuse them into the search area. Considering the feature of a template F_t extracted from PointNet++, the corresponding spatial 3D coordinates P_t, and 9D BoxCloud coordinate C_{bc}^t of the template points, we construct more informative k tuples $\left\{\left[f_j^t, p_j^t, c_j^t, f_i^s\right], \forall j = 1, \cdots, k\right\}$. Finally, a mini-PointNet is used to obtain the aggregated feature of the search point from these pairs, which can be formulated as follows:

$$\hat{f}_i^s = G \odot \left\{MLP(\left[f_j^t, p_j^t, c_j^t, f_i^s\right)\right\}_{j=1}^{k}), \tag{4}$$

where $G\odot$ is a max-pooling operation. Finally, we can get the effective target-specific search feature $\hat{F}_s = \{\hat{f}_i^s\}_{i=1}^{M_2}$.

Algorithm 1 The Workflow of DMT

Input: Points Q in query, points P in search area, an initial bounding box B_{init}, previous target states S_{prev}, and target-specific search feature f.

1: **Potential target center generation.** Given S_{prev}, predict a coarse target center C_{coarse} in the current frame using a motion prediction module.
2: **Explicit voting.** Feed f and C_{coarse} into an explicit voting module to estimate the target-specific point feature $\hat{f}$ of the target center.
3: **Final box regression.** Regress the 3D bounding box of the target based on $\hat{f}$ using a prediction head.

Output: The 3D bounding box of the target.

[1]Our framework is not restricted to BAFF, and any suitable backbone could be used.

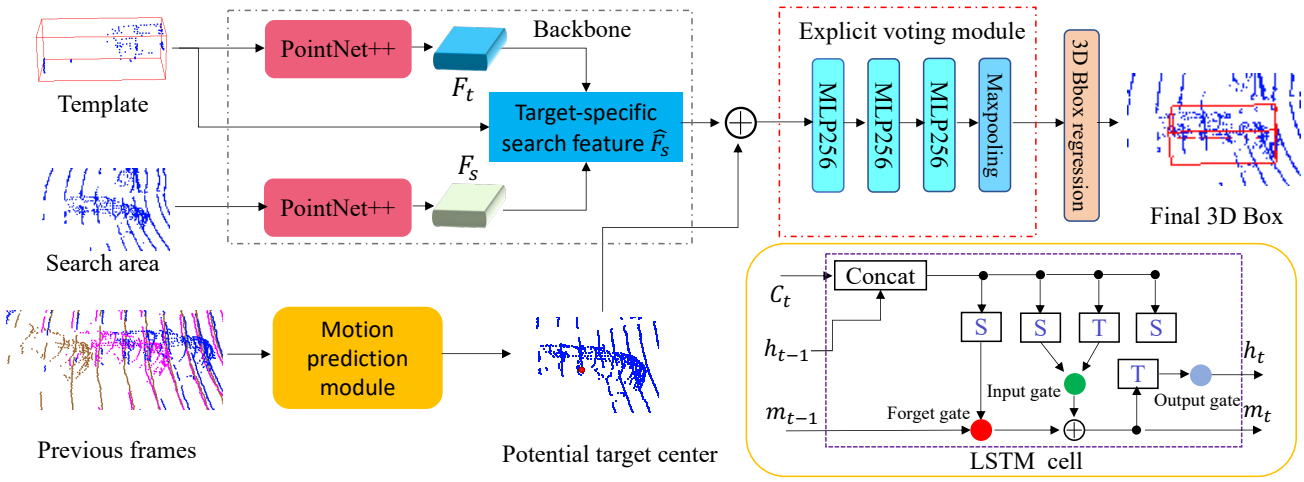


Fig. 1. Overview of DMT. The backbone network first extracts the target-specific features from the template and search area points following [15]. Then the motion prediction module (MPM) estimates the 3D coordinates of a potential target center. Next, the explicit voting module refines the target-specific search feature extracted by the backbone to the coarse predicted center. Finally, a 3D bounding box prediction head regresses the target location. One example of the MPM is an LSTM (lower right corner).

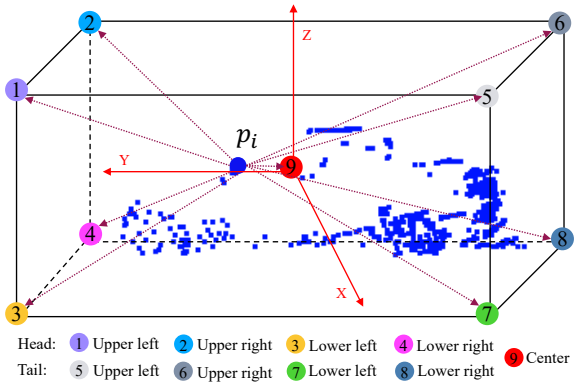


Fig. 2. Illustration of a BoxCloud. The BoxCloud is a set of 9D coordinates. Each 9D coordinate consists of the distances from one point to eight corners and the center of its 3D bounding box.

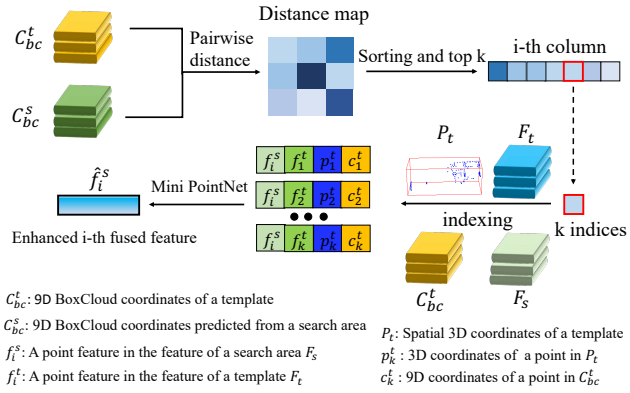


Fig. 3. The workflow of the box-aware feature fusion (BAFF) module. C^s_{bc} is the 9D BoxCloud coordinates, predicted from each search point feature f^s_i via MLP. C^t_{bc}, F_t, P_t are the 9D BoxCloud coordinates, the features, and the spatial 3D coordinates of a template, respectively. The module first generates the distance map between the BoxCloud C^s_{bc} and C^t_{bc} to retrieve the top-k nearest neighbors with respect to each point in the search area. Then, a mini-PointNet is adopted to generate $\hat{f}^s_i$ by aggregating the neighbors' features.

B. Motion prediction module

The previous end-to-end 3D SOT methods [11], [12], [15] heavily rely on point cloud features for target object detection. However, erroneous detection may occur when the point cloud of the target is incomplete [17]. To alleviate this, we propose explicitly leveraging spatio-temporal information for 3D SOT. Specifically, we introduce a motion prediction module (MPM) $\mathcal{M}$ based on previous target states (i.e., predicted 3D target center coordinates in the previous frames) to predict a coarse target center in the current frame. Suppose that we have a tracklet $\{(x_i, y_i, z_i)\}_{i=1}^{t}$ in the previous t frames; the prediction of the target center location in the next $(t+1)$-th frame is formulated as:

$$(\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1}) = \mathcal{M}(\{(x_i, y_i, z_i)\}_{i=1}^{t}). \tag{5}$$

In our general design, common regression or prediction models can be employed as our MPMs for effective target center prediction. Here we introduce several simple yet effective MPMs.

Constant velocity model. The constant velocity model assumes that the target acceleration in the current frame is 0, and the velocity of the target in the current frame should be equal to the velocity in the last frame. Given the target locations in the $(t-1)$ and t-th frames $\{(x_i, y_i, z_i)\}_{i=t-1}^{t}$, the predicted target center coordinates in the $(t+1)$-th frame are calculated as $(2x_t - x_{t-1}, 2y_t - y_{t-1}, 2z_t - z_{t-1})$. Despite the simplicity of this model, we find it also works very well in our DMT.

Sequence-to-sequence prediction model. The goal of our MPM is to predict 3D coordinates based on previously estimated t target coordinates, which is actually a sequence-to-sequence prediction task. A long short-term memory (LSTM) network [47] is a typical sequence-to-sequence prediction model that has been widely used in various sequence prediction tasks. In this paper, we choose a multi-layer LSTM since this naive LSTM model can better validate the effectiveness of our proposed tracking method. The conventional LSTM cell is shown in Fig. 1 (bottom right). More details can be found in [47]. In the implementation, we select the center coordinates of the 10 consecutive frames from the times $t-10$ to t to predict

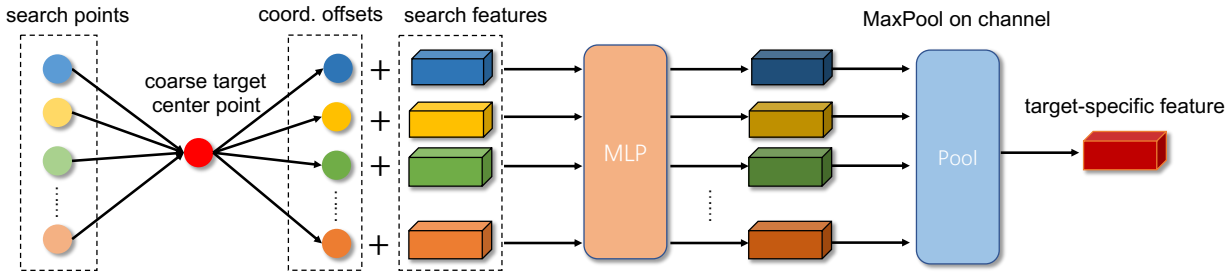


Fig. 4. The overall pipeline of the explicit voting module (EVM). Our EVM first calculates the coordinate offsets between each search point and the coarse predicted target center. Then the offsets are jointly concatenated with the search features for feature modeling via an MLP. Finally, a permutation-invariant max pooling layer is applied to obtain the target-specific feature of the predicted target center point for the final 3D box prediction.

potential target center coordinates in the $(t+1)$-th frame. In the training stage, we prepare multiple training tracklets generated from the KITTI and NuScenes datasets to train the LSTM. In online tracking, we directly use the offline trained LSTM network for motion prediction without further updating.

Regression model. Traditional learning-based regression models can also be employed as MPMs. In this paper, we try several basic regression models, including linear regression, ridge regression, Gaussian processor regression, and RANSAC regression. The training for the above models is similar to the LSTM-based MPM, i.e., using the generated tracklet training data for training in an offline manner.

The above basic MPMs can roughly predict the potential target center coordinates based on the previous states. The prediction is not always reliable since the previous target states may be noisy (i.e., the predicted target center does not match the ground truth), or the target changes position in an unexpected way. To alleviate this problem, we propose a lightweight explicit voting module to further refine the MPM prediction.

C. Explicit voting module

Before going into the details of our proposed explicit voting module (EVM), we give a short review of the RPN module (VoteNet) used by previous trackers [11], [12], [15], [21]. The architecture of VoteNet includes two aspects: 1) Hough voting to convert the search area seeds into possible target centers; and 2) clustering neighboring potential target centers to obtain the final target center. To generate the potential target centers, VoteNet estimates the coordinate offsets between each search seed and ground-truth target center, which aims to push the predicted possible target centers and ground-truth target center to be as close as possible. In our DMT, the above two steps can be removed since the coarse target center location in the current frame is provided by our MPM, which makes our method simpler and lighter.

The overall pipeline of our proposed explicit voting module is shown in Fig. 4. As can be seen, after obtaining the coarse target center coordinates $(\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1})$ estimated by the MPM and the target-specific search feature, the goal of our EVM is to estimate effective features at $(\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1})$. In the design of the EVM, we use coordinate offsets as explicit voting signals to estimate the target center feature. Specifically, we first calculate the coordinate offset between the estimated target center and each search point. We then concatenate the coordinate offset with the search point feature to obtain a candidate voting feature $f \in \mathbb{R}^{C+3}$, where C denotes the feature dimension. Suppose there are N search points with N corresponding candidate voting features $\{f_i\}_{i=1}^{N}$. The explicit target coordinate voting is formulated as:

$$\bar{f}_i = \mathrm{MLP}(f_i), \quad \hat{f} = \mathrm{MaxPool}(\{\bar{f}_i\}_{i=1}^{N}), \tag{6}$$

where $\bar{f}_i \in \mathbb{R}^C$, and $\hat{f} \in \mathbb{R}^C$ are the final estimated target-specific feature at the estimated target center, which is obtained by applying the max pooling operation on the channel dimension of each feature vector in $\{\bar{f}_i\}_{i=1}^{N}$. The estimated feature $\hat{f}$ is finally fed into a prediction head (i.e., MLP) to regress the bounding box of the target.

In the training stage, given a ground-truth target center location in a frame, we randomly sample diverse points around the ground-truth center. For stable training, the maximum distance between the sampled points and the ground-truth center should not be too large, and here we set it to 0.75 meters. During training, our EVM learns to estimate target-specific features of the sampled points that are effective for predicting the final bounding box. Note that the diverse sampled points can effectively mimic the noisy predictions of MPM, which makes our DMT less sensitive to noise in the predicted target track.

D. Loss function

Following [15], our training loss includes three components: classification loss, box-cloud loss, and regression box loss. The first two losses enhance the target-specific feature extracted by the backbone, while the latter supervises the estimated 3D bounding box. In addition, we add a motion prediction loss to train the MPM (except for the constant velocity model).

Point-wise classification loss. Following [12], we note that only search points located on the surface of a ground-truth target are useful in the EVM, and thus labeled as positives, while all others are negatives. Therefore, a standard binary cross entropy loss L_{cla} is adopted to classify the search points.

BoxCloud loss. The BoxCloud features [15] in the search area are unknown in the inference stage, so we need to predict

the 9D BoxCloud coordinate C_{bc} in the search area, which is supervised by a smooth-L1 regression loss.

$$\mathcal{L}_{bc} = \frac{1}{\sum_i E_i} \sum_{i=1}^{N} \left\| C_{bc}^i - \hat{C}_{bc}^i \right\| \cdot E_i, \tag{7}$$

where $\hat{C}_{bc}$ are ground-truth BoxCloud coordinates pre-calculated before training. E_i is a binary mask, which indicates whether the i-th point is inside an object BBox or not.

3D box regression loss. The final result of our network is to predict the 3D box parameters $C_{bbox} = \{\hat{x}, \hat{y}, \hat{z}, \hat{\theta}\}$. Following previous work, we adopt Huber (smooth-L1 loss) to supervise the regression.

$$\mathcal{L}_{bbox} = \left\| C_{bbox} - \hat{C}_{bbox} \right\|, \tag{8}$$

where $\hat{C}_{bbox}$ is the ground-truth bounding box of the target.

Motion prediction loss. When training an MPM, we hope the distance between the predicted center coordinates of the target and the ground truth is as small as possible. In this paper, we use the mean squared error loss $\mathcal{L}_v$ for supervision:

$$\mathcal{L}_v = \left\| C_{cen}^{t+1} - \hat{C}_{cen}^{t+1} \right\|_2, \tag{9}$$

where $C_{cen}^{t+1} = (\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1})$ (see Eq. (5)) is the predicted target center coordinates at the $(t+1)$-th frame, and $\hat{C}_{cen}^{t+1}$ is the corresponding ground-truth coordinates.

Note that we first train the MPM with $\mathcal{L}_v$, and then we use the following combined loss to train the backbone network, EVM, and the prediction head:

$$L = \alpha L_{cla} + \beta L_{bc} + \gamma L_{bbox}, \tag{10}$$

where α, β, and γ are hyperparameters to balance their relationship. Here we set $\alpha = 0.2$, $\beta = 1.0$, $\gamma = 0.2$.

E. Implementation details

We follow previous 3D trackers [12], [15] to generate templates and search point clouds in both the training and testing stages. To fairly compare with recent trackers equipped with online detectors, we use the same target-specific search feature generation method in BAT [15], which makes the predictions of BAT and our DMT both based on the same augmented search features.

Search area generation. In practice, the object movement between two consecutive frames is relatively small, so searching the entire frame for the target is unnecessary. Following [15], we look for the target near the previous object location to generate 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.

Network architecture. In the proposed MPM, we use one LSTM layer with 50 hidden units as the motion predictor. The input tracklet length is set to 10, meaning that we use target states in the previous 10 frames for prediction. The model size of this LSTM model is about 50K, which is extremely light. The EVM is implemented as a three-layer MLP with 256 hidden units, where the first two layers are followed by a 1D batch normalization layer and a ReLU activation layer. We use the same backbone and box prediction head as P2B [12] and BAT [15].

Training. In the training stage, we first generate tracklet training data (i.e., each tracklet contains the target center coordinates in every 10 frames and the corresponding ground-truth target center coordinates in the next frame) to train the LSTM network. The batch size is set to the overall dataset size, and the learning rate and training epochs are respectively set to 1e-3 and 8,000. The whole training takes only 28 seconds for the car category of the KITTI dataset, which is efficient. After training the LSTM network in the offline manner, we use it for online testing without further modifications. The proposed DMT is trained for 60 epochs using the Adam optimizer with a batch size of 100. The learning rate is initialized as 1e-3 and decayed by 0.5 in every 5 epochs.

Testing. During testing, we apply the trained DMT to infer the 3D bounding boxes of a given target within tracklets frame by frame. For the current frame, the template is updated by fusing the point clouds in the first given BBox and in the previously estimated BBox. To obtain the search area, we enlarge the previously estimated BBox by 2 meters in the current frame and collect the points within the enlarged BBox.

V. Experiments

In this section, we first describe the experimental settings. Next, we present experiments on the KITTI and NuScenes datasets to demonstrate the efficacy of our lightweight 3D SOT tracker, DMT.

A. Dataset

The KITTI dataset [23] includes raw point clouds scanned by the Velodyne HDL-64E rotating 3D laser scanner and annotations for object instances in the form of 3D bounding boxes. The tailored dataset contains 21 outdoor scenes and 8 categories of targets. Following [12], we generate tracklets for target instances within all videos and split the KITTI training set into three parts: scenes 00-16, scenes 17-18, and scenes 19-20 for the training, validation, and test sets, respectively, since the annotations of the test set in KITTI are inaccessible. Furthermore, we also conduct experiments on the more challenging dataset NuScenes [24]. The NuScenes dataset includes 1000 outdoor scenes and 23 categories of objects with annotated 3D bounding boxes. Specifically, the NuScenes dataset contains 32,302 frames in the car category, which is five times larger than the KITTI dataset. Following [15], the training set of NuScenes is used for training, and the validation set is used for testing.

Sparsity of point clouds. Although there are (on average) $\sim$120k points in each frame of raw LiDAR data, the points on the target object might be quite sparse due to occlusion and LiDAR defects [12]. Thus we count the number of points in the pedestrian category of the KITTI benchmark, as shown in Fig. 5. About 36% of pedestrians have fewer than 100 points, and this sparsity introduces great challenges to 3D single object tracking based on point clouds.

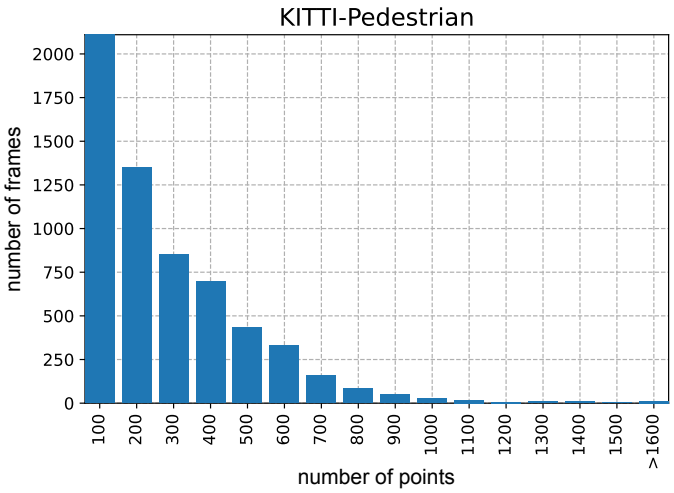


Fig. 5. Long-tailed distribution of the frame-wise number of points in KITTI-Pedestrian, which shows the sparsity of target points.

B. Evaluation metric

Following [12], [15], we apply One Pass Evaluation (OPE) [48] to measure the Success and Precision of different approaches. For a predicted bounding box and a ground-truth bounding box, "Success" is defined as the intersection over union (IOU) between them. "Precision" is defined as the AUC for the distance error curve from 0 to 2m, which is measured between the centers of the two boxes. The success and precision metrics, respectively, measure the box overlap and center distance error between the predicted bounding box and the ground-truth bounding box.

C. Comparison with State-of-the-arts

We compare our network with the state-of-the-art methods: SC3D [18], its follow-up SC3D-RPN [49], FSiamese [44], 3DSiamRPN [45], P2B [12], MLVSNet [21], PTT [11], and BAT [15]. For a fair comparison, we use the same evaluation metrics. In this paper, the default setting of the MPM is an LSTM prediction model. Fig. 6 and Table I show the success and precision of each network on the KITTI and NuScenes datasets. The success and precision values for other methods are those reported in their published papers [11], [12], [15], [18], [21], [44], [45], [49]. We first quantitatively evaluate our network on KITTI, and then extend the comparisons to NuScenes.

Comparisons on KITTI. Following [12], [15], we generate the search area centered on the previous result in the inference stage to meet the requirement of real scenarios. The results in Table I show that the proposed DMT outperforms other 3D trackers significantly. Specifically, when we mix all categories together to test the average performance following previous trackers, our average performance is 55.1, outperforming BAT by $\sim$4% on Success, indicating the effectiveness of the proposed DMT. When compared with PTT for the rigid object (e.g., Van) tracking, DMT has a significant advantage ($\sim$10%) over PTT in the less-frequent van category in terms of the success metric. However, DMT does not achieve the highest performance in the more-frequent Car category. The transformer-based tracker PTT can learn better features of rigid objects since it has complex network architectures and more parameters but relies on more data to train the networks. Qualitative results are given in Section V-E.

To demonstrate its generalizability for non-rigid object tracking, we compare it with other trackers on Pedestrian and Cyclist. For Pedestrian, we observe that DMT outperforms BAT and PTT by $\sim$8% and $\sim$6% on Precision respectively, indicating the effectiveness of our tracking pipeline. Amazingly, DMT outperforms BAT and PTT by a large margin for the cyclist category, achieving about $\sim$47%/$\sim$45% improvement for Precision. This phenomenon can be explained as follows: 1) The amount of training and testing samples is extremely small; 2) Our method DMT is less sensitive to interference with non-rigid objects in the search area; 3) DMT is simple yet effective, thus relying on less data to train better networks. The visualized results are shown in Fig. 7. This also demonstrates that our method can achieve better performance, especially when having less data compared with BAT.

Comparisons on NuScenes. For the Car category, DMT achieves the best performance of 43.8/48.3 for Success/Precision, exceeding the performance of the current state-of-the-art method BAT [15] by $\sim$7%/$\sim$9%, respectively. Notably, for the Truck and Trailer categories, DMT achieves $\sim$23% and $\sim$20% improvements over BAT for Precision, which demonstrates that our motion-guided pipeline is more effective, especially on the more challenging dataset. Moreover, for the Bus category, which has the fewest training samples, our DMT still outperforms BAT by a large margin of 8% in terms of the Success metric. Compared with the baseline method BAT, the performance of our DMT shows significant improvements ($\sim$10% on average) in terms of all categories. Note that PTT/MLVSNet did not present results on NuScenes in their papers.

D. Computational cost analysis

In this section, we analyze the required computational resources of different 3D trackers in terms of the number of parameters, floating point operations (FLOPs), and running speed. For a fair comparison, here we test our method on all KITTI-Car frames with a single NVIDIA RTX3090 GPU. As shown in Table II and Fig. 6 (Left), our method uses less time per frame with fewer FLOPs compared with other trackers. Notably, despite the fact that our network includes an LSTM model, the number of parameters in our model are the same as P2B, while our model is significantly faster (57% improvement in FPS) and simpler (36% improvement in FLOPs) using the same RTX3090 GPU. In addition, the running speed of MLVSNet is close to ours. However, our DMT is lighter (i.e., with fewer model parameters) and can achieve much better performance on the KITTI dataset (see Table I), demonstrating that our method is simple yet effective.

E. Results visualization

According to the different categories and difficulties of the targets, we select and visualize some advantageous cases of

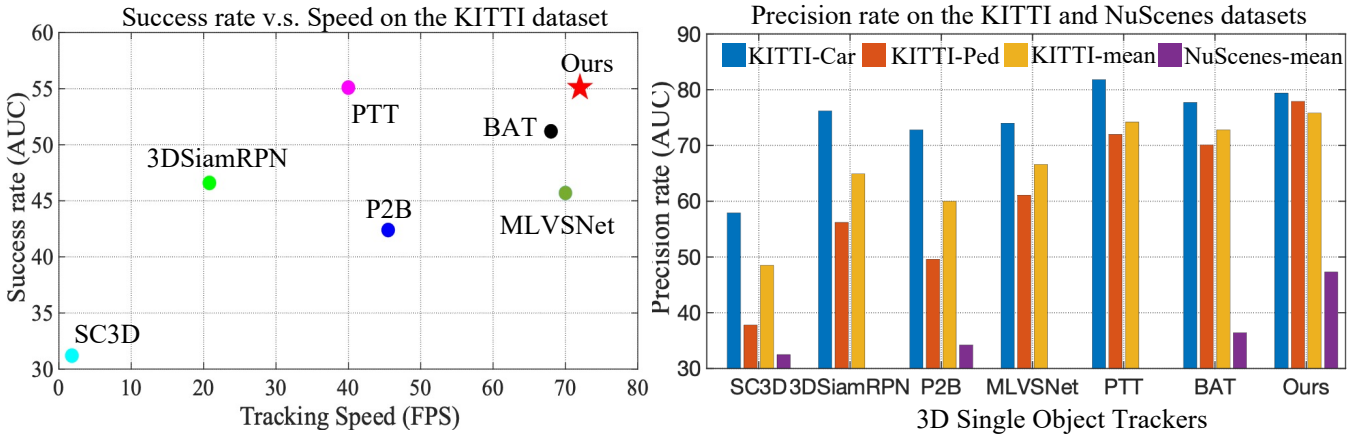


Fig. 6. (Left) Tracking accuracy vs. speed for the Car category of the KITTI benchmark. Our DMT outperforms state-of-the-art 3D single-object trackers in terms of both tracking accuracy and speed. (Right) Precision comparison for KITTI-Car, KITTI-Pedestrian, KITTI-mean, and NuScenes-mean.

TABLE I
RESULTS OF THE SUCCESS AND PRECISION OF DIFFERENT 3D TRACKERS WITH DIFFERENT CATEGORIES ON THE KITTI AND NUSCENES DATASET. 'PED' REPRESENTS 'PEDESTRIAN.'

	Dataset	KITTI					NuScenes				
	Category	Car	Ped	Van	Cyclist	Mean	Car	Truck	Trailer	Bus	Mean
	Frame Number	6424	6088	1248	308	14068	32302	8646	2297	2215	45460
Success (%)	SC3D [17]	41.3	18.2	40.4	41.5	31.2	30.6	23.5	27.4	23.6	28.7
	SC3D-RPN [49]	36.3	17.9	-	43.2	-	-	-	-	-	-
	FSiamese [44]	37.1	16.2	-	47.0	-	-	-	-	-	-
	3DSiamRPN [45]	58.2	35.2	45.6	36.1	46.6	-	-	-	-	-
	P2B [12]	56.2	28.7	40.8	32.1	42.4	34.6	25.2	30.0	28.4	32.3
	MLVSNet [21]	56.0	34.1	52.0	34.3	45.7	-	-	-	-	-
	PTT [11]	**67.8**	44.9	43.6	37.2	**55.1**	-	-	-	-	-
	BAT [15]	60.5	42.1	52.4	33.7	51.2	36.8	28.6	31.8	30.2	34.7
	DMT (Ours)	66.4	**48.1**	**53.3**	**70.4**	**55.1**	**43.8**	**51.3**	**46.8**	**38.2**	**44.0**
Precision (%)	SC3D [17]	57.9	37.8	47.0	70.4	48.5	35.9	24.8	24.8	21.8	32.5
	SC3D-RPN [49]	51.0	47.8	-	81.2	-	-	-	-	-	-
	FSiamese [44]	50.6	32.2	-	77.2	-	-	-	-	-	-
	3DSiamRPN [45]	76.2	56.2	52.8	49.0	64.9	-	-	-	-	
	P2B [12]	72.8	49.6	48.4	44.7	60.0	37.6	25.2	26.7	27.6	34.2
	MLVSNet [21]	74.0	61.1	61.4	44.5	66.6	-	-	-	-	-
	PTT [11]	**81.8**	72.0	52.5	47.3	74.2	-	-	-	-	-
	BAT [15]	77.7	70.1	**67.0**	45.4	72.8	39.5	28.4	30.5	29.5	36.4
	DMT (Ours)	79.4	**77.9**	65.6	**92.6**	**75.8**	**48.3**	**51.1**	**40.3**	**31.9**	**47.3**

TABLE II
COMPUTATIONAL COST REQUIREMENTS OF DIFFERENT 3D SINGLE OBJECT TRACKERS ON KITTI-CAR. * INDICATES THE FPS IS TAKEN FROM THE CORRESPONDING PAPER.

Method	Modality	Params	FLOPs	FPS	Platform
SC3D [17]	LiDAR	-	-	1.8*	1080Ti
FSiamese [44]	LiDAR+RGB	-	-	4.9*	1080Ti
3DSiamRPN [45]	LiDAR	-	-	20.8*	1080Ti
P2B [12]	LiDAR	**5.4M**	4.65G	45.5	3090
MLVSNet [21]	LiDAR	7.6M	-	70.0*	1080Ti
PTT [11]	LiDAR	-	-	45	3090
BAT [15]	LiDAR	5.9M	3.05G	68.0	3090
DMT (Ours)	LiDAR	**5.4M**	**2.98G**	**71.5**	3090

our DMT in Fig. 7. Four frames sorted by time from a full sequence are selected from the Cyclist and Car categories, respectively. For the cyclist target, the point clouds of the target and the tracked results are shown in the top of Fig. 7. In this example, BAT tracks the cyclist wrongly when there are two similar cyclists in the surrounding area. Our method can track the target accurately and tightly, indicating our method is more robust in complex scenarios. Furthermore, we display the tracked results in the Car category, which is shown in the bottom of Fig. 7. Here, BAT fails in the extremely sparse scenes (fewer than 10 points), but our DMT works well, which shows that our proposed method can indeed cope with point sparsity.

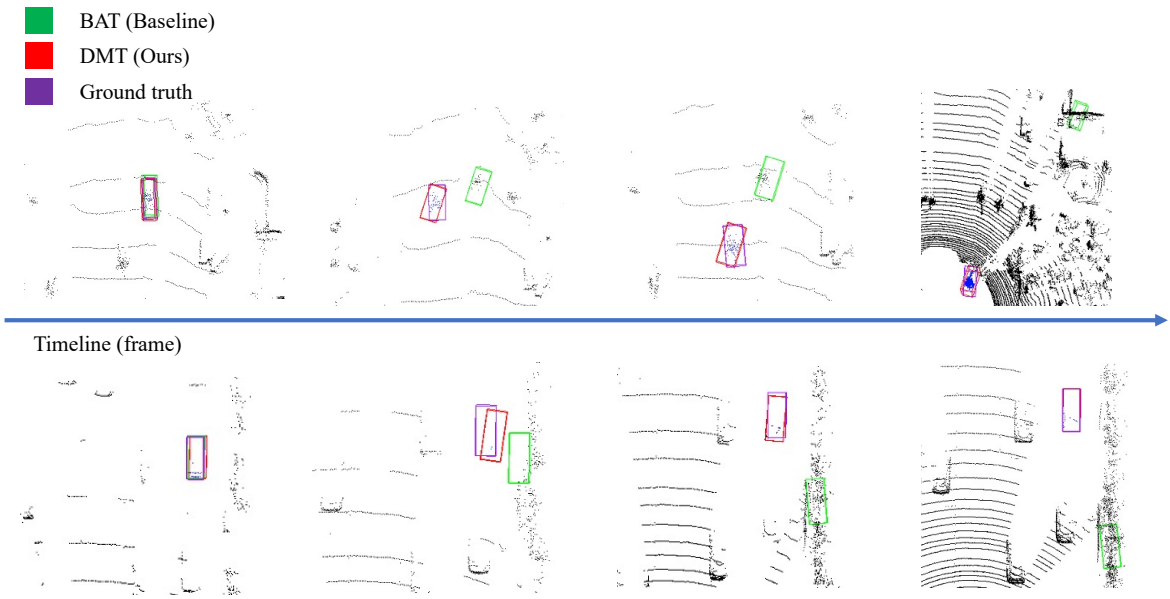


Fig. 7. Visualizations of the example results of DMT compared with BAT. The point clouds of tracked objects are shown in blue. (Top) shows the results for test instances from the Cyclist category. There are two cyclists nearby, and our DMT can maintain the correct track while BAT drifts to the wrong object. (BOTTOM) shows the results for test instances from the Car category. Although the point clouds are extremely sparse ($<$ 10 points), our DMT still tracks the object.

VI. Discussions

In this section, we analyze the effectiveness of the important modules in our DMT, including both the motion prediction module (MPM) and the explicit voting module (EVM). We also discuss the choices of MPM, template generation strategies, sampling distances for training the EVM, the number of sampled training points, and the robustness to object motion patterns.

A. Ablation studies of DMT components

TABLE III
Ablation studies of motion prediction module (MPM) and explicit voting module (EVM) on KITTI-Car.

Method	MPM	EVM	Success	Precision
BAT [15]			60.5	77.7
DMT_MP	✓		-	37.0
DMT_EV		✓	54.0	64.1
DMT	✓	✓	**66.4**	**79.4**

We first conduct an ablation study on the necessity of the EVM and MPM. All studies are conducted on KITTI-Car. We remove the EVM and the MPM in our network one by one, which is denoted as DMT_MP and DMT_EV. Both variations have the same structure as DMT except for the removed module. The baseline model is BAT.

The results are shown in Table III. We obtain four conclusions from these results. (1) The potential target center estimated by the MPM is extremely inaccurate, only achieving 37% for Precision. Note that the MPM in our network cannot regress the orientation of the target, and thus we cannot compute the Success value. (2) The precision without the EVM is 37% (DMT_MP), and with EVM is 79.4% (DMT), which demonstrates that EVM can estimate an effective target-specific point feature to further refine the prediction of the MPM. (3) Comparing DMT_EV with BAT, the performance of DMT_EV degrades about 6% and 13% in terms of Success and Precision, respectively. This is consistent with our expectation that we use a simpler explicit voting module, removing the complicated RPN module. (4) Our full pipeline achieves the best performance, which demonstrates the two modules are mutually beneficial and necessary. In addition, even if the MPM provides inaccurate results, DMT achieves satisfactory performance due to the explicit voting module.

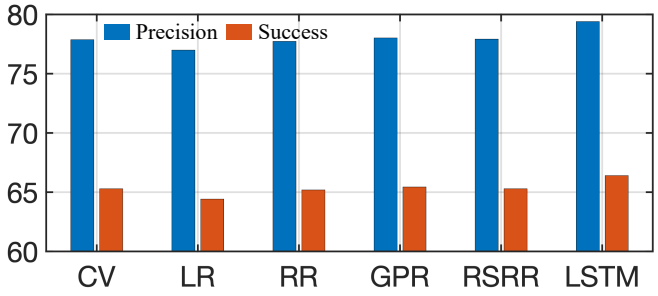


Fig. 8. Comparison of using various regression or prediction models as our motion prediction module on KITTI-Car.

B. The choice of motion prediction module

In Fig. 8, we compare various types of motion prediction models on KITTI-Car. The compared models include constant velocity (CV), linear regression (LR), ridge regression (RR), Gaussian process regression (GPR), RANSAC with ridge regression, and LSTM models. The LR, RR, GPR, and RSRR models are trained in the same way as the LSTM model, i.e., using the same sampled tracklets from the training data in

KITTI-Car for offline training. These models are then applied to motion prediction during online testing without further updating. In Fig. 8, the differences between the various models are not significant, which implies that our DMT is not sensitive to the MPM selection. This is because our EVM is trained to predict GT bounding boxes from diverse sampled locations in the training stage, which makes it less sensitive to noisy predicted target center locations. The sequence-to-sequence prediction LSTM model achieves the best Precision (79.4%) and Success (66.4%) due to its better sequence modeling ability.

C. Template generation strategy

We next explore the performance of our DMT with four template generation strategies following [15], including "the first ground truth," "the previous result," "the first ground-truth and previous result," and "all previous results." "The first ground truth" generates a template using the target in the first frame (the ground truth). "The previous result" uses the result in the last frame predicted by the network, while "all previous results" concatenates the points in all previous results. To update the template efficiently, the default setting is "the first ground truth and previous result", which concatenates the target in the first frame with the prediction result in the last frame.

Table IV shows the Success/Precision results with different settings for different trackers on KITTI-Car. Note that P2B, BAT, and DMT use the same PointNet++ backbone. The specific design in our DMT enables it to achieve better tracking performance than the other trackers under different template generation settings. Specifically, DMT achieves the best performance when using the "all previous" strategy, outperforming BAT and P2B by large margins (∼8% and ∼12%, respectively). Another finding is that P2B, BAT and our DMT all report degraded results under the "all previous" setting since these trackers did not train the networks using all previous results for efficiency, while SC3D did. Despite this, the superior overall performance of DMT in Table IV suggests that DMT better utilizes motion cues from all previous predictions compared with BAT.

TABLE IV
DIFFERENT STRATEGIES FOR TEMPLATE GENERATION. 3D TRACKERS ARE EVALUATED ON KITTI-CAR.

	Method	The First GT	Previous Result	First & Previous	All Previous
Success	SC3D [17]	31.6	25.7	34.9	41.3
	P2B [12]	46.7	53.1	56.2	51.4
	BAT [15]	51.8	59.2	60.5	55.8
	DMT (Ours)	**54.3**	**63.8**	**66.4**	**63.5**
Precision	SC3D [17]	44.4	35.1	49.8	57.9
	P2B [12]	59.7	68.9	72.8	66.8
	BAT [15]	65.5	75.6	77.7	71.4
	DMT (Ours)	**67.2**	**76.7**	**79.4**	**75.9**

D. Sampling distance for training EVM

In this section, we explore the network performance with different sampling distances (i.e., the distances between the sampled points and the ground-truth center) in the training of the EVM. As mentioned in Section IV-C, the distance should not be too large to maintain stable training. We conduct an ablation experiment on KITTI-Car, choosing the distance values from 0.65 to 0.95. As shown in Table V, the performance of DMT reaches its peak with a distance value of 0.75. When the distance expands to 0.95, the performance steadily degrades. This implies the distances between the sampled points and the ground-truth center are still a little large so some outliers are picked. On the other hand, the network performance drops when the distance is set to 0.65. Thus, in this paper, we fix the values to 0.75 for the best performance.

TABLE V
SAMPLING DISTANCE ANALYSIS FOR DMT. WE EVALUATE DMT ON KITTI-CAR.

Distance(m)	Success(%)	Precision(%)
0.65	64.0	77.0
0.75	**66.4**	**79.4**
0.85	63.0	77.5
0.95	63.0	76.8

E. Number of sampled training points

In the practical implementation, we sample various points around the ground-truth target center to mimic motion predictions during the online tracking process. In this section, we study how the number of sampled points affects the final tracking performance. Specifically, we vary the number of sampled points and report the corresponding performance on KITTI-Car in Table VI. We find that sampling dense points (i.e., 64) leads to better performance because dense sampling provides more comprehensive cases for training a more robust EVM. We also notice that the performance is not saturated, implying that better performance can be obtained by sampling a larger number of points.

TABLE VI
SAMPLING POINT NUMBER ANALYSIS FOR DMT. WE EVALUATE DMT ON KITTI-CAR.

Number	Success(%)	Precision(%)
8	61.1	75.0
16	62.2	75.7
32	64.5	78.0
64	**66.4**	**79.4**

F. Robustness test for object motion patterns

To better demonstrate the effectiveness of DMT on complex motion patterns, Fig. 9(a) shows the comparison of our DMT and BAT on tracklets with different motion complexities. Here, motion complexity is defined as the average error of a simple constant velocity model. Our method still performs better than the RPN-based 3D tracker BAT when the motion complexity increases, which demonstrates the robustness of our method to complicated motion patterns. The reason is that we randomly sample diverse points when training the EVM, which makes our method more effectively handle various motion patterns. To further demonstrate the superiority clearly,

we also visualize one tracklet of a pedestrian having a complex trajectory in Fig. 9(b). DMT can track the target accurately despite the complicated motion pattern.

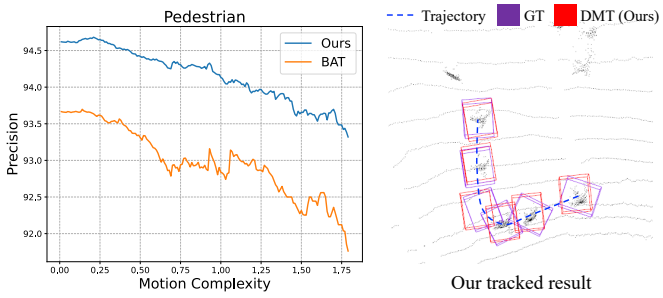


Fig. 9. (a) Comparison of BAT and our DMT under various motion complexity on KITTI-Pedestrian. (b) Example results of DMT for complex motion patterns.

VII. Conclusion

In this paper, we propose DMT, a novel lightweight and detector-free network for 3D single object tracking. We design a motion prediction module for predicting a potential target center, explicitly leveraging spatial-temporal correlations from previous frames to explore prior knowledge. In addition, we propose a simplified voting module to accurately regress the 3D box with the guidance of the potential target center. Experiments show that our DMT method is lighter, faster, and simpler and improves the tracking performance over state-of-the-art methods significantly. According to discussions on experimental results, the explicit voting module based on a potential target center is an advantage of our method. We hope that our work will inspire more investigation into lightweight, detector-free 3D single-object trackers.

VIII. Acknowledgments

This research was supported by the China Scholarship Council. We would like to thank Dr. Tianyu Yang at Tencent for his suggestions and support.

References

[1] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

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

[4] A. I. Comport, É. Marchand, and F. Chaumette, "Robust model-based tracking for robot vision," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1. IEEE, 2004, pp. 692–697.

[5] M. Stoiber, M. Sundermeyer, and R. Triebel, "Iterative corresponding geometry: Fusing region and depth for highly efficient 3d tracking of textureless objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6855–6865.

[6] M. Stoiber, M. Pfanne, K. H. Strobl, R. Triebel, and A. Albu-Schäffer, "Srt3d: A sparse region-based 3d object tracking approach for the real world," *International Journal of Computer Vision*, vol. 130, no. 4, pp. 1008–1030, 2022.

[7] F. Zheng, L. Shao, and J. Han, "Robust and long-term object tracking with an application to vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3387–3399, 2018.

[8] S. Jiayao, S. Zhou, Y. Cui, and Z. Fang, "Real-time 3d single object tracking with transformer," *IEEE Transactions on Multimedia*, 2022.

[9] N. Merrill, Y. Guo, X. Zuo, X. Huang, S. Leutenegger, X. Peng, L. Ren, and G. Huang, "Symmetry and uncertainty-aware object slam for 6dof object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 901–14 910.

[10] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proceedings of the European Conference on Computer Vision*. Springer, 2016, pp. 850–865.

[11] 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.

[12] 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.

[13] Y. Xia, Y. Xu, S. Li, R. Wang, J. Du, D. Cremers, and U. Stilla, "Soe-net: A self-attention and orientation encoding network for point cloud based place recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 348–11 357.

[14] Y. Xia, Y. Xu, C. Wang, and U. Stilla, "Vpc-net: Completion of 3d vehicles from mls point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 174, pp. 166–181, 2021.

[15] 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.

[16] Y. Xia, Y. Xia, W. Li, R. Song, K. Cao, and U. Stilla, "Asfm-net: Asymmetrical siamese feature matching network for point completion," in *Proceedings of the ACM International Conference on Multimedia*, 2021, pp. 1938–1947.

[17] 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.

[18] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International Conference on Machine Learning*. PMLR, 2018, pp. 40–49.

[19] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech House, 2003.

[20] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[21] 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.

[22] 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.

[23] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[24] 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.

[25] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proceedings of the European Conference on Computer Vision*. Springer, 2016, pp. 850–865.

[26] Z. Zhang and H. Peng, "Deeper and wider siamese networks for real-time visual tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4591–4600.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[28] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[29] 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.

[30] L. Zhang, A. Gonzalez-Garcia, J. v. d. Weijer, M. Danelljan, and F. S. Khan, "Learning the model update for siamese trackers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4010–4019.

[31] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 152–167.

[32] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017, pp. 1763–1771.

[33] C. Kim, L. Fuxin, M. Alotaibi, and J. M. Rehg, "Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9553–9562.

[34] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5390–5399.

[35] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *Proceedings of the European Conference on Computer Vision*. Springer, 2020, pp. 474–490.

[36] J. Sun, Y. Xie, S. Zhang, L. Chen, G. Zhang, H. Bao, and X. Zhou, "You don't only look once: Constructing spatial-temporal memory for integrated 3d object detection and tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3185–3194.

[37] J. Wang and Y. He, "Motion prediction in visual object tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2020, pp. 10 374–10 379.

[38] Y. Liu, R. Li, Y. Cheng, R. T. Tan, and X. Sui, "Object tracking using spatio-temporal networks for future prediction location," in *European Conference on Computer Vision*. Springer, 2020, pp. 1–17.

[39] A. Asvadi, P. Girao, P. Peixoto, and U. Nunes, "3d object tracking using rgb and lidar data," in *IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2016, pp. 1255–1260.

[40] A. Bibi, T. Zhang, and B. Ghanem, "3d part-based sparse tracker with automatic synchronization and registration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1439–1448.

[41] U. Kart, J.-K. Kamarainen, and J. Matas, "How to make an rgbd tracker?" in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[42] U. Kart, A. Lukezic, M. Kristan, J.-K. Kamarainen, and J. Matas, "Object tracking by reconstruction with view-specific discriminative correlation filters," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1339–1348.

[43] Y. Cui, Z. Fang, and S. Zhou, "Point siamese network for person tracking using 3d point clouds," *Sensors*, vol. 20, no. 1, p. 143, 2020.

[44] H. Zou, J. Cui, X. Kong, C. Zhang, Y. Liu, F. Wen, and W. Li, "F-siamese tracker: A frustum-based double siamese network for 3d single object tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2020, pp. 8133–8139.

[45] 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.

[46] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

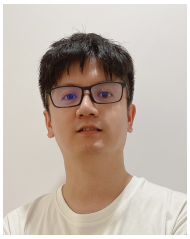
[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

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

Yan Xia is currently pursuing a Ph.D. degree at the School of Engineering and Design, Technical University of Munich, Munich, Germany. He has published several articles in CVPR, ACM MM, IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and ISPRS JOURNAL OF PHOTOGRAMMETRY AND REMOTE SENSING. His current research interests include 3D vision, deep learning, and point cloud processing.

Qiangqiang Wu received a BS degree from the School of Information and Electronic Engineering, Zhejiang Gongshang University in 2016, and the MS degree in Computer Science from Xiamen University in 2019. He is working toward a Ph.D. degree in the Department of Computer Science at the City University of Hong Kong, Hong Kong, China. His research interests include computer vision and machine learning.



Wei Li Wei Li received his Ph.D. degree from the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics. He is the senior staff engineer at Inceptio. His research interests include robotics, computer vision, and computer graphics. He has published 10+ papers in top journals and conferences, including Science Robotics / SIGGRAPH / TVCG / AAAI / CVPR / ICRA.

Antoni B. Chan received his BS and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. He is currently a Full Professor in the Department of Computer Science at the City University of Hong Kong. His research interests include computer vision, machine learning, pattern recognition, and music analysis.

Uwe Stilla (Senior Member, IEEE) was born in Cologne, Germany, in 1957. He received the Dipl.-Ing. degree in electrical engineering from Gesamthochschule Paderborn, Paderborn, Germany, in 1980, and the Dipl.-Ing. degree in biomedical engineering and the Ph.D. degree from the University of Karlsruhe, Karlsruhe, Germany, in 1987 and 1993, respectively. From 1990 to 2004, he was with the Research Institute of Optronics and Pattern Recognition Forschungsgesellschaft für Angewandte Naturwissenschaften (FGAN)-Fraunhofer Institut fuer Optronik und Mustererkennung (FOM), Ettlingen, Germany. Since 2004, he has been a Professor with the Technical University of Munich (TUM), Munich, Germany, and the Head of the Department of Photogrammetry and Remote Sensing. He was the Dean of Studies from 2005 to 2016 and the Vice Dean from 2008 to 2013 with the Faculty of Civil, Geo, and Environmental Engineering. His research interests include image analysis in the field of photogrammetry and remote sensing, and his publication list shows more than 500 entries. Dr. Stilla is the Chair of the International Society for Photogrammetry and Remote Sensing (ISPRS) Working Group II/III Pattern Analysis in Remote Sensing, a Principal Investigator of the International Graduate School of Science and Engineering (IGSSE), the Vice Director of the Commission for Geodesy and Glaciology [Kommission fuer Erdmessung und Glaziologie (KEG)], Bavarian Academy of Science and Humanities (BAdW), Munich, and the President of the German Society of Photogrammetry, Remote Sensing and Geoinformation (DGPF). He has been the Organizer and the Chair of multiple conferences related to ISPRS and IEEE.