

PTT: Point-Trajectory Transformer for Efficient Temporal 3D Object Detection

Kuan-Chih Huang¹ Weijie Lyu¹ Ming-Hsuan Yang^{1,2} Yi-Hsuan Tsai²

¹University of California, Merced ²Google

Abstract

Recent temporal LiDAR-based 3D object detectors achieve promising performance based on the two-stage proposal-based approach. They generate 3D box candidates from the first-stage dense detector, followed by different temporal aggregation methods. However, these approaches require per-frame objects or whole point clouds, posing challenges related to memory bank utilization. Moreover, point clouds and trajectory features are combined solely based on concatenation, which may neglect effective interactions between them. In this paper, we propose a point-trajectory transformer with long short-term memory for efficient temporal 3D object detection. To this end, we only utilize point clouds of current-frame objects and their historical trajectories as input to minimize the memory bank storage requirement. Furthermore, we introduce modules to encode trajectory features, focusing on long short-term and future-aware perspectives, and then effectively aggregate them with point cloud features. We conduct extensive experiments on the large-scale Waymo dataset to demonstrate that our approach performs well against state-of-the-art methods. Code and models will be made publicly available at <https://github.com/kuanchihhuang/PTT>.

1. Introduction

Detecting 3D objects [6–8, 13, 21] is a crucial problem for autonomous driving, enabling vehicles to perceive and interpret their surroundings accurately. With the advent of LiDAR sensors, significant strides have been made in enhancing the precision of 3D object detection. Nevertheless, the inherent sparsity of point cloud data captured by LiDAR sensors poses a significant challenge. The sparse data complicates accurately identifying and localizing objects, particularly in challenging occlusion scenarios.

Recent research studies [1, 5, 30, 35] have shifted their focus towards utilizing multi-frame LiDAR data as input to enhance detection performance. This approach leverages point cloud sequences captured over time as the vehicle moves in real-world scenarios. Compared to single-frame information, these sequences offer a more comprehensive

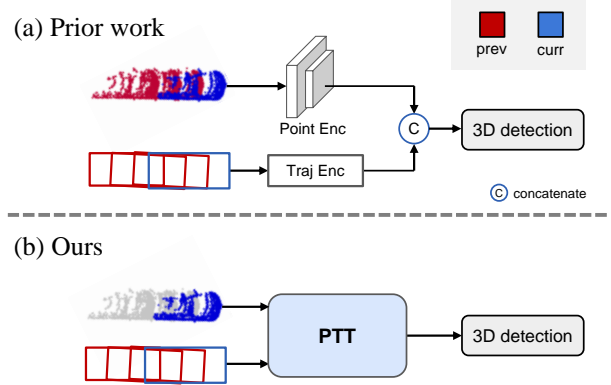


Figure 1. **Different approaches for temporal 3D object detection.** (a) Existing methods [1, 5] require per-frame point clouds as input, resulting in more memory overhead. In addition, the straightforward concatenation of point and trajectory features overlooks their interactions across features. (b) Our approach minimizes the space requirement for the memory bank by utilizing only the current frame’s point cloud as input. Moreover, we introduce a point-trajectory transformer (PTT) to effectively integrate point and trajectory features. Note that the gray point clouds indicate that they are not utilized).

and denser representation of the surrounding environment.

One straightforward method to tackle sequential data is concatenating all multi-frame LiDAR as input [30, 35], followed by the original 3D detection pipeline, which provides performance benefit via the denser information. To better model the information of objects instead of the whole point cloud, recent methods [1, 5] typically adopt a two-stage approach. In the first stage, they generate bounding box candidates and select objects’ point clouds using region proposal networks. Subsequently, these approaches concatenate point features with proposal features for 3D object detection, as illustrated in Figure 1(a).

Nevertheless, these proposal-based techniques suffer from two main limitations. First, sampling object point clouds for each frame in sequential modeling requires a significant memory burden for a long sequence during the training and testing phases. Second, these methods focus on the design of networks for point cloud sequences, which

often neglects the significance of the interaction between point and proposal features. In this paper, we propose PTT, a Point-Trajectory Transformer for efficient temporal 3D object detection to handle these two challenges (see Figure 1(b)).

We emphasize the significance of the multi-frame proposal trajectory. Instead of storing sampled point clouds from all previous LiDARs, we only input the current-frame object point cloud. We employ PTT to model the relationship between single-frame point clouds and multi-frame proposals, facilitating efficient utilization of rich multi-frame LiDAR data with reduced memory overhead. Our motivation is recognizing that previous proposals can inherently approximate the object’s information without using the entire object point cloud. Consequently, we can effectively leverage historical information without requiring more memories. While previous methods necessitate a large memory bank to store point cloud sequences, our pipeline processes concise information from more frames with limited memory usage. This enables PTT to capture object trajectories over a longer time duration (*e.g.*, 64 frames for ours *vs.* 8 frames for MSF [5]).

Furthermore, unlike previous approaches that concatenate proposal trajectories with point cloud features, we introduce the point-trajectory transformer with long short-term memory to model the relationship between single-frame point clouds and multi-frame proposals. In this approach, we partition information obtained from multiple frames into long-term and short-term encoding to enhance representations over temporal information. In addition, we infer the future possible trajectory of the proposals to generate future-aware point features. Finally, we design an aggregator to facilitate interactions between point and trajectory features.

We conduct extensive experiments on the Waymo Open Dataset [24] to show the efficiency and effectiveness of the proposed PTT. For example, our method can achieve favorable performance against state-of-the-art schemes, using longer frames with lower memory requirements. Moreover, we present detailed ablation studies and analyses to demonstrate the usefulness of proposed components in PTT. The main contributions of this work are:

- We propose a temporal 3D object detection framework by efficiently considering only the point cloud in the current frame, along with longer historical proposal trajectories.
- We introduce a point-trajectory transformer with long short-term memory and future encoding that efficiently extracts features from single-frame point clouds and multi-frame trajectories, eliminating the need for storing multi-frame point clouds.
- We present comprehensive results with analysis to demonstrate the effectiveness of our method, allowing

longer temporal information with lower cost while still achieving favorable results against other approaches.

2. Related Work

Single-Frame 3D Object Detection. Recent 3D object detection methods mainly focus on learning effective feature representations of point clouds. They can be categorized into two categories: point-based and voxel-based approaches. Point-based techniques [16, 21, 29, 32] directly operate on point clouds in continuous space, extracting semantic and geometric features. In [21], PointRCNN presents a two-stage approach, initially segmenting foreground points and generating object-wise proposals, followed by refinement in canonical coordinates. 3DSSD [11] introduces a fusion sampling technique utilizing feature distance to ensure comprehensive information preservation. Point-GNN [23] uses a graph neural network to create a more condensed representation of point cloud data. Furthermore, Pointformer [15] introduces a transformer module that incorporates local and global attention, operating directly on point clouds.

Voxel-based methods [28, 31, 34] rasterize point clouds into fixed-size spatial voxels and use 2D or 3D CNNs to extract dense features. Second [28] introduces a sparse convolution method to enhance the efficiency of point cloud processing. VoxelNet [34] encodes voxel features from the raw point cloud using a dense region proposal network for 3D object detection. In [14], Voxel-RCNN employs voxel Region-of-Interest (RoI) pooling to extract voxel features within proposals, facilitating subsequent refinement. Moreover, CenterPoint [31] transforms the sparse output from a backbone network into a dense Bird’s Eye View (BEV) feature map. The model predicts a dense heatmap representing the center locations of objects using the information from this dense feature map. Our approach uses CenterPoint [31] as the proposal generation network to predict 3D proposals for each frame. Subsequently, we introduce a temporal transformer to aggregate points and trajectory features for refining the proposals effectively.

Temporal 3D Object Detection. Compared with 3D object detectors solely from a single frame, including temporal information in multi-frame scenarios [1, 5, 12, 30] yields a more dense and comprehensive understanding of the surrounding environment. Fast-and-furious [12] concatenates hidden feature maps of multi-frame point clouds to aggregate temporal information. 3D-MAN [30] utilizes an attention mechanism to align 3D objects from various views and employs a memory bank to store and aggregate temporal information for processing lengthy point cloud sequences. Furthermore, Offboard3D [17] and MPPNet [1] associate the detected boxes from individual frames of the sequence as proposal trajectories and extract high-quality proposal features by sampling sequential point clouds along these

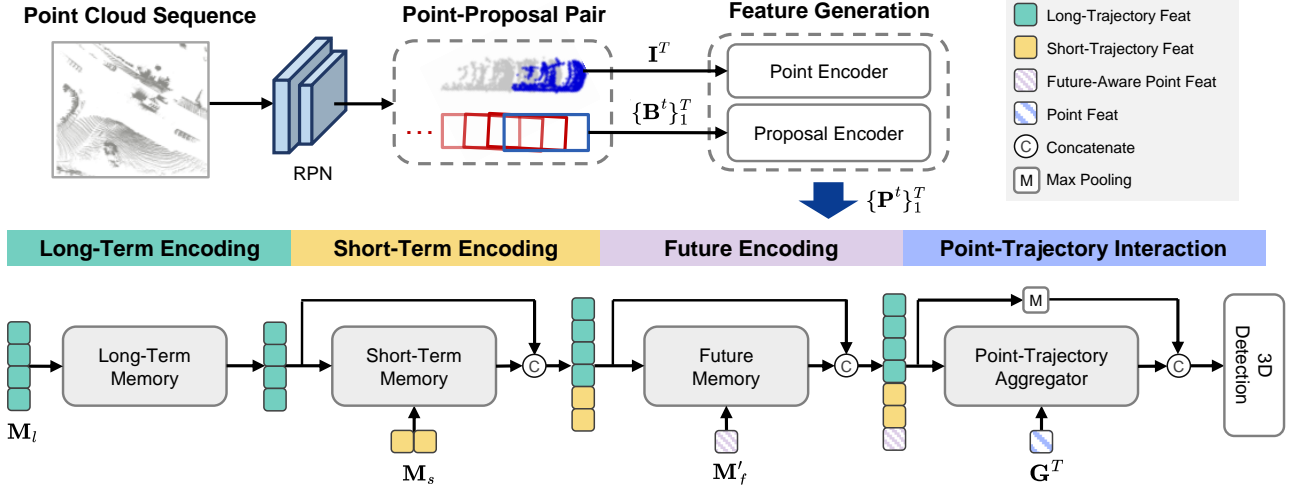


Figure 2. **Overall framework of the proposed Point-Trajectory Transformer (PTT).** First, we utilize a region proposal network (RPN) at timestamp T to generate proposals \mathbf{B}^T for each frame, sample the corresponding point-of-interest \mathbf{I}^T , and connect past T -frame 3D proposals to form proposal trajectories $\{\mathbf{B}^1, \dots, \mathbf{B}^T\}$. Then, we take the single-frame point cloud for each object and its previous multi-frame trajectory as input to generate point-trajectory features \mathbf{P}^t , which avoid storing per-frame points to mitigate memory bank’s overhead (Section 3.3). Finally, we present a point-trajectory transformer (PTT) to fuse features, which consists of four components: Long-term and Short-term encoders for extracting two types of features, a future-aware module for extracting future-aware point features, and a point-trajectory aggregator for adaptive interaction between trajectory and current frame’s point features \mathbf{G}^T (Section 3.4).

trajectories to enhance 3D detection performance. In CenterFormer [35], features of multi-frame objects are fused with object queries through cross-attention operations. On the other hand, MSF [5] generates proposals in the current frame and propagates them to explore features in earlier frames, offering efficiency and suitability for online detection systems. Nevertheless, the backward propagation process still necessitates a substantial memory bank to store the point cloud sequence, which may be less practical for on-device deployment.

Despite promising advances, most methods mentioned above sample objects’ point clouds or the entire scene’s point clouds in each frame, which may demand a large memory bank to store, especially when dealing with long sequences. In contrast, our proposed PTT achieves memory-efficient online detection by employing a point-trajectory transformer to fuse information from the current frame’s point cloud and multi-frame proposal trajectory.

3. Proposed Approach

3.1. Framework Overview

In this paper, we propose a Point-Trajectory Transformer (PTT) for efficient temporal 3D object detection. The overall framework is shown in Figure 2. We employ a region proposal network (RPN) to produce proposals for each frame, sample the corresponding points of interest based on the proposals, and establish connections among past T -frames’ 3D proposals to construct box trajectories. To

mitigate the memory bank’s overhead, our approach stores solely the single-frame point cloud and multi-frame proposals (see further analysis in Section 3.2).

We first encode two types of features, point-to-proposal and proposal-to-proposal features, by sampling the current frame’s point cloud and multi-frame trajectories (Section 3.3). Then, we present our proposed point-trajectory transformer to fuse sequential features, which mainly consists of the following modules: 1) Long-term and short-term memory modules are designed to encode two distinct types of sequential features. 2) A future encoder is employed to predict the object’s future trajectory, facilitating the encoding of future-aware features. 3) We introduce a point-trajectory aggregator to integrate an object’s current-frame point features with its trajectory features (Section 3.4).

3.2. Analysis of Memory Bank’s Size

Recent state-of-the-art temporal 3D object detectors often adopt a two-stage approach [1, 5]. In the initial stage, they generate 3D box candidates using the region proposal network and subsequently sample points based on these boxes for integrating information across temporal frames. However, these approaches need to store either extensive raw point cloud data [5] or point clouds of identified objects [1] based on the length of frames. This leads to potential memory issues, especially with a large number of frames.

We compare our approach with MPPNet [1] and MSF [5] regarding the space complexity of the memory bank in Ta-

ble 1. Assuming that each object requires $\mathcal{O}(N)$ for storing its point clouds and $\mathcal{O}(O)$ for the proposal in each frame, where the average number of objects in the scene is k . Here, $\mathcal{O}(F)$ denotes the average space complexity for the per-frame point cloud input. For the region proposal network, we employ CenterPoint [31] with an input of 4 frames of accumulated point clouds. This implies that all the approaches require $\mathcal{O}(4F)$ space complexity to deal with the input.

Regarding the point cloud of objects, MPPNet [1] with 16 frames (default setting) needs to store per-frame object points, so the memory complexity is $\mathcal{O}(16kN)$, which will increase the complexity based on the frame number. For MSF [5] with 8-frames input, the memory complexity for dealing with the point clouds is $\mathcal{O}(8F)$ since they need to keep the entire sequential point cloud for propagating the proposals into all previous frames to find previous bounding box candidates. Instead, our approach only samples the current-frame object point cloud, which only requires $\mathcal{O}(kN)$ space complexity without being proportional to the frame number, allowing us to leverage longer frames (e.g., 64).

For the complexity of proposals, MPPNet, MSF, and our PTT are $\mathcal{O}(16kO)$, $\mathcal{O}(kO)$, and $\mathcal{O}(64kO)$, respectively, where MSF’s complexity is not proportional to the frame number. Although our space complexity for the proposal part is more significant due to the use of 64 frames, compared with the point cloud, the memory bank size for the proposal is much smaller. For instance, each object stores a point cloud with $128 \text{ points} \times 4 \text{ dimensions}$, including 3D location and intensity. In contrast, a proposal for each object only requires 9 dimensions, encompassing the 3D center point, 3D size, heading, and 2D BEV velocity.

Assuming there are $k = 200$ objects and the average point number in the point clouds is 160,000 for each frame, we can calculate that $\mathcal{O}(F)$ and $\mathcal{O}(kN)$ are approximately $355\times$ and $55\times$ larger than $\mathcal{O}(kO)$, respectively. We can then obtain that the memory bank size of our approach is the lowest, which indicates that our approach efficiently minimizes the memory bank’s storage requirement while maximizing the leveraged frame number.

3.3. Point-Trajectory Feature Generation

Unlike the prior work [1, 5] keeping per-frame objects or entire point clouds, we advocate using single-frame points and multi-frame proposals, a more storage-efficient solution as discussed in Section 3.2. In this section, we propose to encode two kinds of features: Point-to-Proposal and Proposal-to-Proposal features.

Point-to-Proposal Features. Consider the current frame’s points-of-interest \mathbf{I}^T and the previous T -frame box trajectories $\{\mathbf{B}^1, \dots, \mathbf{B}^T\}$ for a specific object, our objective is to encode the point features based on historical proposals. We first generate point-to-proposal features based on the offset

Method	Frame Num	Space Complexity		
		RPN	Point	Proposal
MPPNet [1]	16	$\mathcal{O}(4F)$	$\mathcal{O}(16kN)$	$\mathcal{O}(16kO)$
MSF [5]	8	$\mathcal{O}(4F)$	$\mathcal{O}(8F)$	$\mathcal{O}(kO)$
PTT (Ours)	64	$\mathcal{O}(4F)$	$\mathcal{O}(kN)$	$\mathcal{O}(64kO)$

Table 1. **Comparisons with various methods regarding space complexity.** We utilize CenterPoint [31] with 4-frame input as the region proposal network. F and N denote the numbers representing the whole point cloud and individual objects, respectively. k is the average object number in a scene. O represents the dimension of the proposal. See Section 3.2 for more details.

between points and points of the bounding box (i.e., center and eight corners) $\{\mathbf{b}_i^t \in \mathcal{C}_3(\mathbf{B}^t) : i = 0, \dots, 8\}$ [18]:

$$\mathbf{G}^t = \text{MLP}(\text{Concat}(\{\mathbf{I} - \mathbf{b}_i^t\}_{i=0}^8, \Delta t)) \in \mathbb{R}^{N \times C}, \quad (1)$$

where N is number of the sampled points, C is the feature dimension, and Δt is the time difference between \mathbf{b}^t to the current frame. Then $\{\mathbf{G}^t\}_1^T$ denotes the sequential Point-to-Proposal features that encode the point features based on its proposal at different frame t . To this end, we are not required to store each frame’s point cloud for the object but can still preserve its geometric and motion features related to previous frames.

Proposal-to-Proposal Features. By considering proposals from the previous T frames, we can encode the relative location of the proposals compared to the current frame, enhancing the model’s ability to capture the motion pattern more effectively. The Proposal-to-Proposal features can be defined as:

$$\mathbf{F}^t = \text{MLP}(\text{Concat}(\mathbf{B}_{xyz}^t - \mathbf{B}_{xyz}^T, \mathbf{B}_{whl\theta}^t, \Delta t)) \in \mathbb{R}^C, \quad (2)$$

where \mathbf{B}_{xyz} and $\mathbf{B}_{whl\theta}$ represent the attributes of the proposal, including the center location (x, y, z) and the size of the bounding box (w, l, h) respectively, along with the heading θ . To this end, $\{\mathbf{F}^t\}_1^T$ is the sequential Proposal-to-Proposal features.

Sequential Point-Trajectory Features. After deriving the Point-to-Proposal and Proposal-to-Proposal features from (1) and (2), we can combine them into integrated features. Specifically, we compress the per-frame Point-to-Proposal features using a MaxPool operation and concatenate them with the corresponding Proposal-to-Proposal features to acquire per-frame point-trajectory features:

$$\mathbf{P}^t = \text{MLP}(\text{Concat}(\mathbf{F}^t, \text{MaxPool}(\mathbf{G}^t))). \quad (3)$$

In this manner, we can use $\{\mathbf{P}^t\}_1^T$ to represent the sequential point-trajectory information, which is utilized for further feature learning.

3.4. Point-Trajectory Transformer

After acquiring the sequential point-trajectory features, our objective is to devise a point-trajectory transformer with long short-term memory to encode these features. Our transformer comprises four modules: long-term memory, short-term memory, future-aware encoder, and point-trajectory aggregator module.

Long-Term Memory. Contrary to prior approaches that may be limited to handling short sequences, our memory-efficient solution can manage significantly longer sequences. We partition the feature sequence into long-term and short-term memories to capture sequential information effectively across various periods. Considering the total frame length T , we separate the sequential point-trajectory features $\{\mathbf{P}^t\}$ into the short memory as $\mathbf{M}_s = \{\mathbf{P}^T, \dots, \mathbf{P}^{T-m_s+1}\}$ with a short frame number m_s , and $\mathbf{M}_l = \{\mathbf{P}^{m_l}, \dots, \mathbf{P}^1\}$ with a long frame number m_l , where $m_l + m_s = T$.

For the long-term memory $\mathbf{M}_l \in \mathbb{R}^{m_l \times C}$, we utilize the multi-head self-attention encoder layers to encode the long-term features:

$$\begin{aligned} \mathbf{Q} &= \mathbf{M}_l \mathbf{W}_q, \mathbf{K} = \mathbf{M}_l \mathbf{W}_k, \mathbf{V} = \mathbf{M}_l \mathbf{W}_v, \\ \hat{\mathbf{M}}_l &= \text{FFN}(\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V})), \end{aligned} \quad (4)$$

where \mathbf{W}_* denotes the learnable parameters for the long-term memory. We utilize one linear layer followed by the ReLU activation to construct our feed-forward network FFN (see [27] for details about the self-attention layer MultiHead). To this end, we can obtain the aggregated long-term trajectory features $\hat{\mathbf{M}}_l$.

Short-Term Memory. Short-term features are crucial for learning 3D bounding boxes in the current frame, as they capture the dynamic status close to the current frame. We use the short-term memory \mathbf{M}_s as a query to extract valuable trends in historical sequences from the aggregated long-term memory $\hat{\mathbf{M}}_l$. We exploit the transformer module to model the relationship between short- and long-term memories:

$$\begin{aligned} \mathbf{Q}^s &= \mathbf{M}_s \mathbf{W}_q^s, \mathbf{K}^l = \hat{\mathbf{M}}_l \mathbf{W}_k^l, \mathbf{V}^l = \hat{\mathbf{M}}_l \mathbf{W}_v^l, \\ \hat{\mathbf{M}}_s &= \text{FFN}(\text{MultiHead}(\mathbf{Q}^s, \mathbf{K}^l, \mathbf{V}^l)), \end{aligned} \quad (5)$$

where $\hat{\mathbf{M}}_s$ denotes the interactive short-term memory features. Then, we connect one FC layer to learn the residual offset between the current frame's proposal and the corresponding ground truth bounding box. This ensures that the learned short-term features can represent short historical motion cues.

Future-Aware Activation. Future information is also important for 3D estimation in the current frame by improving feature learning. Considering a 3D proposal box with

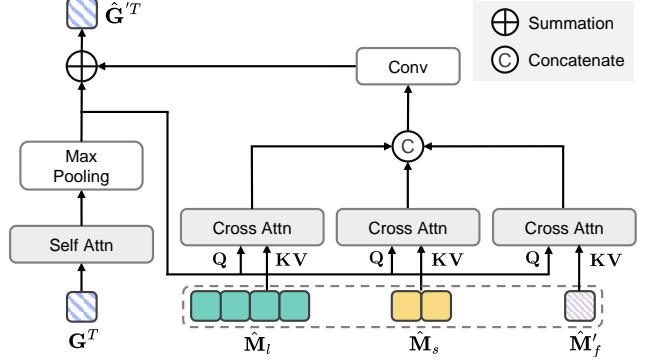


Figure 3. **Point-Trajectory Aggregator.** Current-frame point cloud features are squeezed and interact with long-term, short-term, and future memory.

$(x, y, z, w, l, h, \theta)$ with the estimated velocity (v_x, v_y) at frame T , we can infer the box's state at future time difference Δt as $\mathbf{B}^{T+\Delta t} = (x + \Delta t \cdot v_x, y + \Delta t \cdot v_y, z, w, l, h, \theta + \Delta t \cdot v_\theta)$, where v_θ can be calculated by the difference within previous frames: $v_\theta = (\theta_T - \theta_1)/T$.

Next, we can utilize (3) to derive the future-aware features $\mathbf{M}_f = \{\mathbf{P}^{T+m_f}, \dots, \mathbf{P}^{T+1}\}$ for m_f future frames based on the inferred future box $\{\mathbf{B}^{T+t}\}_{t=1}^{m_f}$. We then concatenate the long-term $\hat{\mathbf{M}}_l$ and the short-term memory $\hat{\mathbf{M}}_s$ to generate the whole memory features $\mathbf{M} = [\hat{\mathbf{M}}_l, \hat{\mathbf{M}}_s]$, where $[\cdot, \cdot]$ means the feature concatenation operation along the temporal dimension. Afterward, since the inferred future boxes may be inaccurate, we concatenate the learnable queries $\mathbf{q}_f \in \mathbb{R}^{m_f \times C}$ to the future features along the channel dimension, followed by the convolution operation: $\mathbf{M}'_f = \text{Conv}(\text{Concat}(\mathbf{q}_f, \mathbf{M}_f))$, facilitating additional learning for potential correction. Then the concatenated features \mathbf{M}'_f are fed to the multi-head self-attention encoder layers:

$$\begin{aligned} \mathbf{Q}^f &= \mathbf{M}'_f \mathbf{W}_q^f, \mathbf{K}^f = \mathbf{M} \mathbf{W}_k^f, \mathbf{V}^f = \mathbf{M} \mathbf{W}_v^f, \\ \hat{\mathbf{M}}'_f &= \text{FFN}(\text{MultiHead}(\mathbf{Q}^f, \mathbf{K}^f, \mathbf{V}^f)), \end{aligned} \quad (6)$$

where $\hat{\mathbf{M}}'_f$ is the future-aware features. Similar to the short-term memory encoder, we connect the feature to one FC layer that learns the residual offset, aiding the model in capturing meaningful representations.

Point-Trajectory Aggregator. After obtaining the enhanced sequential point-trajectory features based on long-term, short-term, and future encoding, we integrate them with the current frame's point features into the proposed point-trajectory aggregator as shown in Figure 3.

For the current frame's point cloud features $\mathbf{G}^T \in \mathbb{R}^{N \times C}$, we first employ a self-attention encoding block, followed by the max-pooling operation to obtain the compressed point features $\hat{\mathbf{G}}^T \in \mathbb{R}^C$. Subsequently, three distinct cross-attention modules are used to interact with

Method	Frames	ALL (3D mAPH)		VEH (3D AP/APH)		PED(3D AP/APH)		CYC(3D AP/APH)	
		L1	L2	L1	L2	L1	L2	L1	L2
SECOND [28]	1	63.05	57.23	72.27/71.69	63.85/63.33	68.70/58.18	60.72/51.31	60.62/59.28	58.34/57.05
IA-SSD [33]	1	64.48	58.08	70.53/69.67	61.55/60.80	69.38/58.47	60.30/50.73	67.67/65.30	64.98/62.71
PointPillar [8]	1	63.33	57.53	71.60/71.00	63.10/62.50	70.60/56.70	62.90/50.20	64.40/62.30	61.90/59.90
LiDAR R-CNN [10]	1	66.20	60.10	73.50/73.00	64.70/64.20	71.20/58.70	63.10/51.70	68.60/66.90	66.10/64.40
RSN [26]	1	-	-	75.10/74.60	66.00/65.50	77.80/72.70	68.30/63.70	-	-
PV-RCNN [19]	1	69.63	63.33	77.51/76.89	68.98/68.41	75.01/65.65	66.04/57.61	67.81/66.35	65.39/63.98
Part-A2 [22]	1	70.25	63.84	77.05/76.51	68.47/67.97	75.24/66.87	66.18/58.62	68.60/67.36	66.13/64.93
VoTR [14]	1	-	-	74.95/74.25	65.91/65.29	-	-	-	-
SWFormer-1f [25]	1	-	-	77.8/77.3	69.2/68.8	80.9/72.7	72.5/64.9	-	-
PillarNet [4]	1	74.60	68.43	79.09/78.59	70.92/70.46	80.59/74.01	72.28/66.17	72.29/71.21	69.72/68.67
PV-RCNN++ [20]	1	75.21	68.61	79.10/78.63	70.34/69.91	80.62/74.62	71.86/66.30	73.49/72.38	70.70/69.62
FSD [3]	1	77.4	70.8	79.2/78.8	70.5/70.1	82.6/77.3	73.9/69.1	77.1/76.0	74.4/73.3
3D-MAN [30]	16	-	-	74.53/74.03	67.61/67.14	-	-	-	-
SST-3f [2]	3	-	-	78.66/78.21	69.98/69.57	83.81/80.14	75.94/72.37	-	-
SWFormer-3f [25]	3	-	-	79.4/78.9	71.1/70.6	82.9/79.0	74.8/71.1	-	-
CenterFormer [35]	4	77.0	73.2	78.1/77.6	73.4/72.9	81.7/78.6	77.2/74.2	75.6/74.8	73.4/72.6
CenterFormer [35]	8	77.3	73.7	78.8/78.3	74.3/73.8	82.1/79.3	77.8/75.0	75.2/74.4	73.2/72.3
MoDAR [9]	92	-	-	81.0/80.5	73.4/72.9	83.5/79.4	76.1/72.1	-	-
MPPNet [1]	4	79.83	74.22	81.54/81.06	74.07/73.61	84.56/81.94	77.20/74.67	77.15/76.50	75.01/74.38
MPPNet [1]	16	80.40	74.85	82.74/82.28	75.41/74.96	84.69/82.25	77.43/75.06	77.28/76.66	75.13/74.52
MSF [5]	4	80.20	74.62	81.36/80.87	73.81/73.35	85.05/82.10	77.92/75.11	78.40/77.61	76.17/75.40
MSF [5]	8	80.65	75.46	82.83/82.01	75.76/75.31	85.24/82.21	78.32/75.61	78.52/77.74	76.32/75.47
PTT (Ours)	32	81.11	75.48	83.31/82.82	75.83/75.35	85.91/82.94	78.89/76.00	78.29/77.57	75.81/75.11
PTT (Ours)	64	81.32	75.71	83.71/83.21	76.26/75.78	85.93/82.98	78.90/76.02	78.51/77.79	76.01/75.32

Table 2. **Performance comparisons on the validation set of the Waymo Open Dataset.** The metrics are 3D AP and APH for both L1 and L2 difficulties. We use **bold** numbers to highlight the best results.

the trajectory memories: long-term, short-term, and future-aware memories, with point features serving as queries to generate three different queried memories: $\hat{\mathbf{M}}_{lp}$, $\hat{\mathbf{M}}_{sp}$, and $\hat{\mathbf{M}}_{fp}$, where \mathbf{M}_{*p} indicates the point-interacted features for long-term, short-term, and future encoding, respectively.

After that, we concatenate these three features and apply a convolution layer, followed by the summation of the point features to obtain the final integrated point features:

$$\hat{\mathbf{G}}'^T = \hat{\mathbf{G}}^T + \text{Conv}([\hat{\mathbf{M}}_{lp}, \hat{\mathbf{M}}_{sp}, \hat{\mathbf{M}}_{fp}]), \quad (7)$$

where $\hat{\mathbf{G}}'^T$ is the enhanced point features for the current frame. Furthermore, we apply the Max Pooling operation to obtain final trajectory features $\mathbf{M}_{\text{Traj}} = \text{MaxPool}([\hat{\mathbf{M}}_l, \hat{\mathbf{M}}_s, \hat{\mathbf{M}}_f])$ and concatenate it with the enhanced point features $\hat{\mathbf{G}}'^T$ for the following 3D box learning.

Loss Functions. Finally, the learned features are sent to the detection head for 3D box estimation. The overall detection loss \mathcal{L}_{det} comprises the confidence prediction loss $\mathcal{L}_{\text{conf}}$ and the box regression loss \mathcal{L}_{reg} , formulated as:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{conf}} + \alpha \mathcal{L}_{\text{reg}}, \quad (8)$$

where α represents the weight for balancing two losses. Our confidence prediction loss $\mathcal{L}_{\text{conf}}$ and box regression loss \mathcal{L}_{reg} are implemented using the binary cross-entropy loss and box regression loss respectively, similar to those employed in CT3D [18].

4. Experiments

4.1. Experimental Setup

Dataset and Evaluation Metrics. We evaluate the proposed PTT method on the Waymo Open Dataset (WOD) [24]. WOD comprises 1,150 sequences, partitioned into 798 training, 202 validation, and 150 testing sequences. Each sequence spans 20 seconds and is captured by a 64-line LiDAR sensor at 10Hz. Evaluation of the WOD dataset employs mean average precision (mAP) and mAP weighted by heading accuracy (mAPH) as metrics. The dataset includes three object categories: “Vehicle,” “Pedestrian,” and “Cyclist”. These classes are further divided into two difficulty levels, LEVEL1 and LEVEL2. LEVEL1 pertains to objects with more than 5 points, while LEVEL2 includes objects with fewer than 5 points but at least 1 point.

Implementation Details. We utilize the CenterPoint [31] as our region proposal network (RPN) to generate first-stage high-quality 3D proposals, which take four adjacent frames at input and add one additional head for object velocity prediction similar to [1, 5]. Building upon these proposals, we train our proposed PTT for 6 epochs, utilizing the ADAM optimizer with an initial learning rate of 0.003 and a batch size of 4. We set an IoU matching threshold of 0.5 for generating the proposed trajectories in the first stage. Also, we apply proposal-centric box jitter augmentation to the per-frame 3D proposal box, following the approach outlined in

Long	Short	L1	L2
		mAP / mAPH	mAP / mAPH
✗	✗	79.61/78.22	74.21/72.79
✓	✗	80.00/78.59	74.33/72.98
✗	✓	79.96/78.55	74.77/72.93
✓	✓	81.38/80.04	75.88/74.59

Table 3. **Ablation study on the effectiveness of the long-term and short-term memory.** The mean AP and APH are reported for L1 and L2 difficulties.

PointRCNN [21]. Each proposal is randomly sampled with 128 raw LiDAR points. The feature dimension for point clouds and trajectories is configured as 128. We set α as 2 following CT3D [18] to balance the overall loss terms. During training, intermediate supervision is implemented by adding a loss to the output of each learning block, and all the intermediate losses are summed to train the model. At the testing stage, only bounding boxes and confidence scores predicted from the last block are utilized. All the experiments are implemented in Pytorch on an NVIDIA 3090 GPU. Following MPPNet [1] to reduce computational cost, we conduct ablation studies by using a light-cost setting of training our PTT with 32 frames for 3 epochs. More details and results are presented in the supplementary materials.

4.2. Main Results

Comparisons with single-frame methods. In Table 2, we compare our proposed PTT with different state-of-the-art methods on the Waymo validation set. First, in comparison to state-of-the-art single-frame methods, e.g., FSD [3], our PTT with 64-frame sequences notably enhances the overall 3D mAPH (LEVEL 2), achieving improvements of 5.6%, 6.9%, and 2.0% on vehicles, pedestrians, and cyclists categories, respectively.

Comparisons with multi-frame methods. Second, our approach performs favorably against several multi-frame methods in most evaluation metrics. Compared with two top-performing multi-frame methods, MPPNet [1] and MSF [5], our approach consistently performs better in 11 out of 14 evaluation metrics with the same region proposal network [31].

Runtime Performance. Specifically, our PTT approach, employing 64 frames, performs better than MPPNet [1] using 16 frames, while consuming less memory overhead (see Table 1) since MPPNet needs to store per-frame sampled object point clouds. In addition, MPPNet introduces an MLP-Mixer module to interact points among sequences, incurring more computational overhead. In contrast, our point-trajectory transformer efficiently handles long sequential data. For instance, while MPPNet takes approximately 1100ms to process 16-frame sequential data, our

Method	Vehicle		Pedestrian		Cyclist	
	L1	L2	L1	L2	L1	L2
w/o Future	81.31	74.00	81.80	74.73	75.98	73.82
w/ Future	81.50	74.32	82.25	75.22	76.38	74.24

Table 4. **Effectiveness of the proposed future encoding.** APH on L1 and L2 difficulties for three categories are reported.

PTT only requires about 150ms to run 64-frame information on the same devices. This efficiency demonstrates the effectiveness of our approach.

Furthermore, when comparing our approach using 64 frames with MSF [5] using 8 frames, our method also achieves better performance across most metrics. It is worth mentioning that, although our PTT exhibits a comparable runtime to MSF (approximately 160ms), their approach necessitates retaining the entire sequential point cloud to propagate the current frame’s proposals into previous frames. This requirement results in a larger memory bank, potentially causing memory overhead and limiting its applicability to longer sequences.

4.3. Ablation Study

Importance of long-term and short-term encoding. We show the ablation study of our long-term and short-term encoding in Table 3. Overall, using both encodings yields the most favorable results, which shows the complementary properties of these two memory encoding ways. The reason is that the long-term memory allows the model to capture the long-range dependencies, while the short-term memory focuses on the recent state, capturing immediate context. Adding both provides a more comprehensive view of temporal patterns.

Effectiveness of the proposed future encoding. As demonstrated in Table 4, the inclusion of additional future encoding enhances the performance of our proposed PTT, leading to an increase of 0.32%, 0.49%, and 0.42% in APH across vehicle, pedestrian, and cyclist categories within the Level 2 difficulty. This observation validates the value of latent information from future timestamps, aiding the model in contributing to the learning process via the availability of longer trajectory information spanning from the past to the near future.

Effectiveness of the proposed Point-to-Proposal and Proposal-to-Proposal features. We investigate the effectiveness of the proposed point-to-proposal and proposal-to-proposal features in Table 5. We show that jointly applying both features achieves the optimal L1 (mAP/mAPH) and L2 (mAP/mAPH) scores. Utilizing proposal-to-proposal features alone is better than using point-to-proposal features, as the information of the proposals provides more compre-

Point-to-Proposal	Proposal-to-Proposal	L1	L2
		mAP / mAPH	mAP / mAPH
✗	✗	79.99/78.58	74.34/72.98
✓	✗	80.13/78.71	74.46/73.10
✗	✓	81.15/79.79	75.58/74.27
✓	✓	81.38/80.04	75.88/74.59

Table 5. **Effectiveness of the proposed Point-to-Proposal and Proposal-to-Proposal features**, which are defined in (1) and (2), respectively. We report the mean AP and APH for the L1 and L2 difficulties.

Method	Vehicle AP / APH	Pedestrian AP / APH	Cyclist AP / APH
MLP	72.22/71.74	76.51/73.65	74.19/73.46
Transformer	74.83/74.32	77.93/75.22	74.89/74.24

Table 6. **Analysis of different methods to encode the trajectory**. AP and APH scores of L2 difficulties for three categories are reported. Transformer means our proposed point-trajectory transformer (PTT) as discussed in Section 3.4.

SA	CA	L1	L2
		mAP / mAPH	mAP / mAPH
✗	✗	80.00/78.58	74.31/72.95
✓	✗	81.12/79.78	75.63/74.34
✗	✓	81.15/79.79	75.58/74.27
✓	✓	81.38/80.04	75.88/74.59

Table 7. **Ablation study on different attention mechanisms in the proposed point-trajectory aggregator**. “SA” and “CA” denote the self-attention and cross-attention operations, respectively.

hensive information for representing an object.

Effectiveness of the proposed point-trajectory transformer. We demonstrate the effectiveness of our proposed point-trajectory transformer (PTT) in Table 6. We compare the performance of the PTT with an alternative model where we replace the point-trajectory transformer with a multi-layer perceptron (MLP) of comparable parameter count. The results indicate a noteworthy reduction in AP and APH across all three categories. Specifically, in the context of vehicle detection, there is a significant decrease of 2.61 in AP and 2.58 in APH. This indicates the importance of the role played by our proposed point-trajectory transformer in effectively capturing and integrating sequential information for improved object detection accuracy.

Effectiveness of the attention mechanisms in the proposed point-trajectory aggregator. Table 7 shows the importance of different attention designs in the point-trajectory aggregator that involves self-attention and cross-attention mechanisms. The results show that simultaneous usage of both attention operations in the proposed point-

Frame Length	L1	L2
	mAP / mAPH	mAP / mAPH
4	80.47/79.06	74.78/73.43
8	80.56/79.25	74.95/73.66
16	81.13/79.80	75.58/74.29
32	81.38/80.04	75.88/74.59
64	81.54/80.21	76.05/74.75
128	81.46/80.11	75.95/74.68

Table 8. **Comparisons for different lengths of input frames**. We report the mean APH scores among three categories for the level 1 and level 2 difficulties.

trajectory aggregator achieves the best performance across all L1 (mAP), L1 (mAPH), L2 (mAP), and L2 (mAPH) metrics.

Effectiveness of different lengths of the input frame. We compare the performance of our method across different numbers of past frames, as presented in Table 8. The results of mAPH on both L1 and L2 consistently show improvements as the number of past frames increases from 4 to 64. This emphasizes the importance of incorporating information from an extended time range and demonstrates the effectiveness of our method in integrating this valuable sequential data. Furthermore, the performance shows a slight decline when utilizing 128 past frames, suggesting the inclusion of redundant information. Consequently, we choose 32 and 64 frames as a trade-off.

5. Conclusions

In this paper, we present a Point-Trajectory Transformer (PTT) for efficient temporal 3D object detection. We find that 1) leveraging multi-frame point clouds can lead to memory overhead, and 2) considering multi-frame proposal trajectories can be efficient and effective, where these two observations are not widely studied in prior works. To this end, our PTT efficiently establishes connections between single-frame point clouds and multi-frame proposals, facilitating the utilization of rich LiDAR data with reduced memory overhead. By exploiting fewer points for learning effective representations, our approach allows processing information from more frames without encountering memory issues. Meanwhile, we propose long-term, short-term, and future-aware encoders to enhance feature learning over temporal information and enable the ability to generate future-aware features via future trajectories, followed by the proposed point-trajectory aggregator to integrate point clouds and proposals effectively. Our method performs favorably against state-of-the-art approaches on the challenging Waymo Open Dataset, using more frames but smaller memory overhead and faster runtime, showcasing the effectiveness of the proposed approach.

References

- [1] Xuesong Chen, Shaoshuai Shi, Benjin Zhu, Ka Chun Cheung, Hang Xu, and Hongsheng Li. Mppnet: Multi-frame feature intertwining with proxy points for 3d temporal object detection. In *European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 3, 4, 6, 7
- [2] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6
- [3] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fully Sparse 3D Object Detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 6, 7
- [4] Chao Ma Guangsheng Shi, Ruifeng Li. Pillarnet: Real-time and high-performance pillar-based 3d object detection. *European Conference on Computer Vision (ECCV)*, 2022. 6
- [5] Chenhang He, Ruihuang Li, Yabin Zhang, Shuai Li, and Lei Zhang. Msf: Motion-guided sequential fusion for efficient 3d object detection from point cloud sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 3, 4, 6, 7
- [6] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1
- [7] Kuan-Chih Huang, Tsung-Han Wu, Hung-Ting Su, and Winston H. Hsu. Monodtr: Monocular 3d object detection with depth-aware transformer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1
- [8] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 6
- [9] Yingwei Li, Charles R. Qi, Yin Zhou, Chenxi Liu, and Dragomir Anguelov. Modar: Using motion forecasting for 3d object detection in point cloud sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 6
- [10] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar r-cnn: An efficient and universal 3d object detector. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 6
- [11] Shujie Luo, Hang Dai, Ling Shao, and Yong Ding. M3dssd: Monocular 3d single stage object detector. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2
- [12] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional ne. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [13] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Xin Fan, and Wanli Ouyang. Accurate monocular object detection via color-embedded 3d reconstruction for autonomous driving. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 1
- [14] Jiageng Mao, Yujing Xue, Minzhe Niu, et al. Voxel transformer for 3d object detection. *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2, 6
- [15] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2
- [16] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2
- [17] Charles R. Qi, Yin Zhou Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2
- [18] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 4, 6, 7
- [19] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 6
- [20] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *International Journal of Computer Vision (IJCV)*, 2023. 6
- [21] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Point-rcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 7
- [22] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 6
- [23] Weijing Shi and Ragunathan (Raj) Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2
- [24] Pei Sun, Henrik Kretschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 6
- [25] Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds.

In *European Conference on Computer Vision (ECCV)*, 2022. 6

- [26] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 6
- [27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 5
- [28] Bo Li Yan Yan, Yuxing Ma. Second: Sparsely embedded convolutional detection. In *Sensor*, 2018. 2, 6
- [29] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Ji-aya Jia. STD: sparse-to-dense 3d object detector for point cloud. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2
- [30] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3d-man: 3d multi-frame attention network for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 6
- [31] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 4, 6, 7
- [32] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2
- [33] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6
- [34] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [35] Zixiang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh. Centerformer: Center-based transformer for 3d object detection. In *European Conference on Computer Vision (ECCV)*, 2022. 1, 3, 6