

# LiDAR-based Online 3D Video Object Detection with Graph-based Message Passing and Spatiotemporal Transformer Attention

Junbo Yin<sup>1,2</sup>, Jianbing Shen<sup>1,4\*</sup>, Chenye Guan<sup>2,3</sup>, Dingfu Zhou<sup>2,3</sup>, Ruigang Yang<sup>2,3,5</sup>

<sup>1</sup>Beijing Lab of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, China

<sup>2</sup>Baidu Research <sup>3</sup>National Engineering Laboratory of Deep Learning Technology and Application, China

<sup>4</sup>Inception Institute of Artificial Intelligence, UAE <sup>5</sup>University of Kentucky, Kentucky, USA

{yinjuno, shenjianbingcg}@gmail.com <https://github.com/yinjunbo/3DVID>

## Abstract

Existing LiDAR-based 3D object detectors usually focus on the single-frame detection, while ignoring the spatiotemporal information in consecutive point cloud frames. In this paper, we propose an end-to-end online 3D video object detector that operates on point cloud sequences. The proposed model comprises a spatial feature encoding component and a spatiotemporal feature aggregation component. In the former component, a novel Pillar Message Passing Network (PMPNet) is proposed to encode each discrete point cloud frame. It adaptively collects information for a pillar node from its neighbors by iterative message passing, which effectively enlarges the receptive field of the pillar feature. In the latter component, we propose an Attentive Spatiotemporal Transformer GRU (AST-GRU) to aggregate the spatiotemporal information, which enhances the conventional ConvGRU with an attentive memory gating mechanism. AST-GRU contains a Spatial Transformer Attention (STA) module and a Temporal Transformer Attention (TTA) module, which can emphasize the foreground objects and align the dynamic objects, respectively. Experimental results demonstrate that the proposed 3D video object detector achieves state-of-the-art performance on the large-scale nuScenes benchmark.

## 1. Introduction

LiDAR-based 3D object detection plays a critical role in a wide range of applications, such as autonomous driving, robot navigation and virtual/augmented reality [11, 46]. The majority of current 3D object detection approaches [42, 58, 6, 62, 24] follow the *single-frame* detection paradigm, while few of them perform detection in the point cloud *video*. A point cloud video is defined as a temporal sequence of point cloud frames. For instance, in the nuScenes

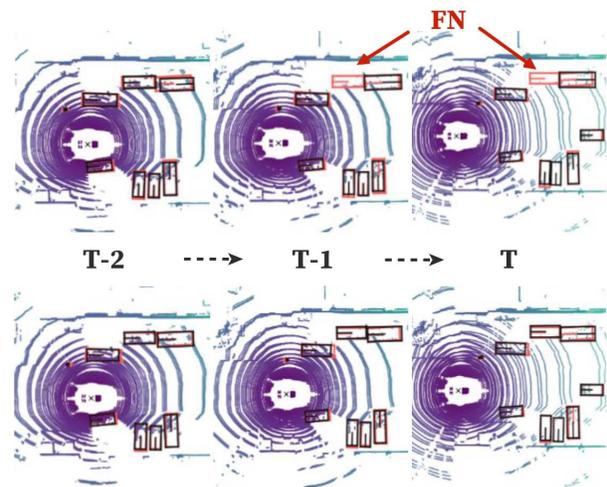


Figure 1: **Occlusion situation in autonomous driving scenarios.** Typical single-frame 3D object detector, e.g. [24], often leads to false-negative (FN) results (top row). In contrast, our online 3D video object detector can handle this (bottom row). The grey and red boxes denote the predictions and ground-truths, respectively.

dataset [4], 20 point cloud frames can be captured per second with a modern 32-beam LiDAR sensor. Detection in single frame may suffer from several limitations due to the sparse nature of point cloud. In particular, occlusions, long-distance and non-uniform sampling inevitably occur on a certain frame, where a single-frame object detector is incapable of handling these situations, leading to a deteriorated performance, as shown in Fig 1. However, a point cloud video contains rich spatiotemporal information of the foreground objects, which can be explored to improve the detection performance. The major concern of constructing a 3D video object detector is how to model the spatial and temporal feature representation for the consecutive point cloud frames. In this work, we propose to integrate a graph-based spatial feature encoding component with an attention-aware spatiotemporal feature aggregation component, to capture the video coherence in consecutive point

\*Corresponding author: Jianbing Shen.

cloud frames, which yields an end-to-end online solution for the LiDAR-based 3D video object detection.

Popular single-frame 3D object detectors tend to first discretize the point cloud into voxel or pillar grids [62, 56, 24], and then extract the point cloud features using stacks of convolutional neural networks (CNNs). Such approaches incorporate the success of existing 2D or 3D CNNs and usually gain better computational efficiency compared with the point-based methods [42, 37]. Therefore, in our spatial feature encoding component, we also follow this paradigm to extract features for each input frame. However, a potential problem with these approaches lies in that they only focus on a locally aggregated feature, *i.e.*, employing a PointNet [39] to extract features for *separate* voxels or pillars as in [62] and [24]. To further enlarge the receptive fields, they have to apply the stride or pooling operations repeatedly, which will cause the loss of the spatial information. To alleviate this issue, we propose a novel graph-based network, named Pillar Message Passing Network (PMPNet), which treats a *non-empty* pillar as a graph node and adaptively enlarges the receptive field for a node by aggregating messages from its neighbors. PMPNet can mine the rich geometric relations among different pillar grids in a discretized point cloud frame by iteratively reasoning on a  $k$ -NN graph. This effectively encourages information exchanges among different spatial regions within a frame.

After obtaining the spatial features of each input frame, we assemble these features in our spatiotemporal feature aggregation component. Since ConvGRU [1] has shown promising performance in the 2D video understanding field, we suggest an Attentive Spatiotemporal Transformer GRU (AST-GRU) to extend ConvGRU to the 3D field through capturing dependencies of consecutive point cloud frames with an attentive memory gating mechanism. Specifically, there exist two potential limitations when considering the LiDAR-based 3D video object detection in autonomous driving scenarios. First, in the bird’s eye view, most foreground objects (*e.g.*, cars and pedestrians) occupy small regions, and the background noise is inevitably accumulated as computing the new memory in a recurrent unit. Thus, we propose to exploit the Spatial Transformer Attention (STA) module, an intra-attention derived from [48, 53], to suppress the background noise and emphasize the foreground objects by attending each pixel with the context information. Second, when updating the memory in the recurrent unit, the spatial features of the two inputs (*i.e.*, the old memory and the new input) are not well aligned. In particular, though we can accurately align the static objects across frames using the ego-pose information, the dynamic objects with large motion are not aligned, which will impair the quality of the new memory. To address this, we propose a Temporal Transformer Attention (TTA) module that adaptively captures the object motions in consecutive

frames with a temporal inter-attention mechanism. This will better utilize the modified deformable convolutional layers [65, 64]. Our AST-GRU can better handle the spatiotemporal features and produce a more reliable new memory, compared with the vanilla ConvGRU. To summarize, we propose a new LiDAR-based online 3D video object detector that leverages the previous long-term information to improve the detection performance. In our model, a novel PMPNet is introduced to adaptively enlarge the receptive field of the pillar nodes in a discretized point cloud frame by iterative graph-based message passing. The output sequential features are then aggregated in the proposed AST-GRU to mine the rich coherence in the point cloud video by using an attentive memory gating mechanism. Extensive evaluations demonstrate that our 3D *video* object detector achieves better performance against the *single-frame* detectors on the large-scale nuScenes benchmark.

## 2. Related Work

**LiDAR-based 3D Object Detection.** Existing works on 3D object detection can be roughly categorized into three groups, which are LiDAR-based [42, 58, 62, 24, 61, 56], image-based [22, 54, 26, 34, 25] and multi-sensor fusion-based [5, 29, 30, 21, 38] methods. Here, we focus on the LiDAR-based approaches since they are less sensitive to different illumination and weather conditions. Among them, one category [62, 57, 24] typically discretizes the point cloud into regular grids (*e.g.*, voxels or pillars), and then exploits the 2D or 3D CNNs for features extraction. Another category [42, 58, 6] learns 3D representations directly from the original point cloud with a point-wise feature extractor like PointNet++ [39]. It is usually impractical to directly apply the point-based detectors in scenes with large-scale point clouds, for they tend to perform feature extraction for every single point. For instance, a keyframe in nuScenes dataset [4] contains 300,000 point clouds, which are densified by 10 non-keyframe LiDAR sweeps within 0.5 s. Operating on point clouds with such a scale will lead to non-trivial computation cost and memory demand. In contrast, the voxel-based methods can tackle this kind of difficulty for they are less sensitive to the number of points. Zhou *et al.* [62] first apply the end-to-end CNNs for voxel-based 3D object detection. They propose to describe each voxel with a Voxel Feature Encoding (VFE) layer, and utilize cascade 3D and 2D CNNs to extract the deep features. Then a Region Proposal Network (RPN) is employed to obtain the final detection results. After that, Lang *et al.* [24] further extend [62] by projecting the point clouds to the the bird’s eye view and encoding each discretized grid (named pillars) with a Pillar Feature Network (PFN).

Both the VFE layers and the PFN only take into account *separate* voxels or pillars when generating the grid-level representation, which ignores the information exchange in

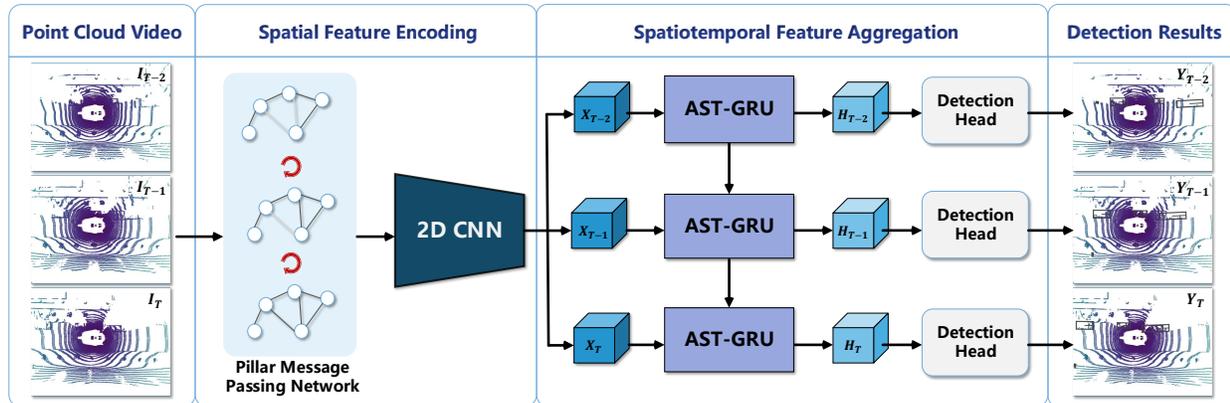


Figure 2: **Our online 3D video object detection framework** includes a spatial feature encoding component and a spatiotemporal feature aggregation component. In the former component, a novel PMPNet (§3.1) is proposed to extract the spatial features of each point cloud frame. Then, features from consecutive frames are sent to the AST-GRU (§3.2) in the latter component, to aggregate the spatiotemporal information with an attentive memory gating mechanism.

larger spatial regions. In contrast, our PMPNet encodes the pillar feature from a global perspective by graph-based message passing, and thus promotes the representation with the non-local property. Besides, all these *single-frame* 3D object detectors can only process the point cloud data frame-by-frame, lacking the exploration of the temporal information. Though [33] applies temporal 3D ConvNet on point cloud sequences, it encounters the feature collapse issue when downsampling the features in the temporal domain. Moreover, it cannot deal with long-term sequences with multi-frame labels. Our AST-GRU instead captures the long-term temporal information with an attentive memory gating mechanism, which can fully mine the spatiotemporal coherence in the point cloud video.

**Graph Neural Networks.** Graph Neural Networks (GNNs) are first introduced by Gori *et al.* [13] to model the intrinsic relationships of the graph-structured data. Then Scarselli *et al.* [41] extend it to different types of graphs. Afterward, GNNs are explored in two directions in terms of different message propagation strategies. The first group [28, 19, 60, 36, 40] uses the gating mechanism to enable the information to propagate across the graph. For instance, Li *et al.* [28] leverage the recurrent neural networks to describe the state of each graph node. Then, Gilmer *et al.* [12] generalizes a framework to formulate the graph reasoning as a parameterized message passing network. Another group [3, 15, 9, 17, 27] integrates convolutional networks to the graph domain, named as Graph Convolutional Neural Networks (GCNNs), which update node features via stacks of graph convolutional layers. GNNs have achieved promising results in many areas [9, 10, 51, 2, 52] due to the great expressive power of graphs. Our PMPNet belongs to the first group by capturing the pillar features with a gated message passing strategy, which is used to construct the spatial representation for each point cloud frame.

### 3. Model Architecture

In this section, we elaborate on our online 3D video object detection framework. As shown in Fig. 2, it consists of a spatial feature encoding component and a spatiotemporal feature aggregation component. Given the input sequences  $\{I_t\}_{t=1}^T$  with  $T$  frames, we first convert the point cloud coordinates from the previous frames  $\{I_t\}_{t=1}^{T-1}$  to the current frame  $I_T$  using the GPS data, so as to eliminate the influence of the ego-motion and align the static objects across frames. Then, in the spatial feature encoding component, we extract features for each frame with the Pillar Message Passing Network (PMPNet) (§3.1) and a 2D backbone, producing sequential features  $\{X_t\}_{t=1}^T$ . After that, these features are fed into the Attentive Spatiotemporal Transformer Gated Recurrent Unit (AST-GRU) (§3.2) in the spatiotemporal feature aggregation component, to generate the new memory features  $\{H_t\}_{t=1}^T$ . Finally, a RPN head is applied on  $\{H_t\}_{t=1}^T$  to give the final detection results  $\{Y_t\}_{t=1}^T$ . Some network architecture details are provided in §3.3.

#### 3.1. Pillar Message Passing Network

Previous point cloud encoding layers (*e.g.*, the VFE layers in [62] and the PFN in [24]) for voxel-based 3D object detection typically encode each voxel or pillar separately, which limits the expressive power of the grid-level representation due to the small receptive field of each local grid region. Our PMPNet instead seeks to explore the rich spatial relations among different grid regions by treating the non-empty pillar grids as graph nodes. Such design effectively reserves the non-Euclidean geometric characteristics of the original point clouds and enhance the output pillar features with a non-locality property.

Given an input point cloud frame  $I_t$ , we first uniformly discretize it into a set of pillars  $\mathcal{P}$ , with each pillar uniquely associated with a spatial coordinate in the x-y plane as

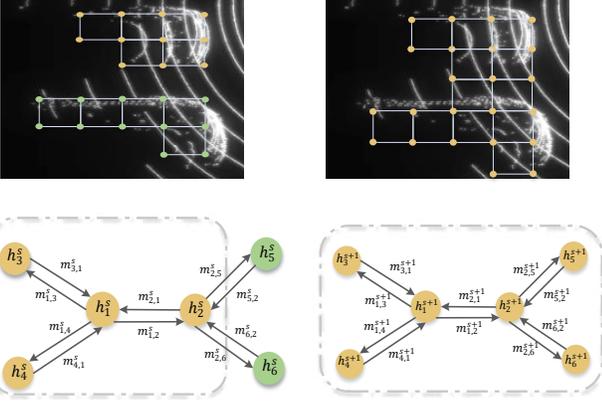


Figure 3: **Illustration of one iteration step for message propagation**, where  $h_i$  is the state of node  $v_i$ . In step  $s$ , the neighbors for  $h_1$  are  $\{h_2, h_3, h_4\}$  (within the gray dash line), presenting the pillars in the top car. After aggregating messages from the neighbors, the receptive field of  $h_1$  is enlarged in step  $s + 1$ , indicating the relations with nodes from the bottom car are modeled.

in [24]. Then, PMPNet maps the resultant pillars to a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where node  $v_i \in \mathcal{V}$  represents a non-empty pillar  $P_i \in \mathcal{P}$  and edge  $e_{i,j} \in \mathcal{E}$  indicates the message passed from node  $v_i$  to  $v_j$ . For reducing the computational overhead, we define  $\mathcal{G}$  as a  $k$ -nearest neighbor ( $k$ -NN) graph, which is built from the geometric space by comparing the centroid distance among different pillars.

To explicitly mine the rich relations among different pillar nodes, PMPNet performs iterative message passing on  $\mathcal{G}$  and updates the nodes state at each iteration step. Concretely, given a node  $v_i$ , we first utilize a pillar feature network (PFN) [24] to describe its initial state  $h_i^0$  at iteration step  $s = 0$ :

$$h_i^0 = F_{\text{PFN}}(P_i) \in \mathbb{R}^L, \quad (1)$$

where  $h_i^0$  is a  $L$ -dim vector and  $P_i \in \mathbb{R}^{N \times D}$  presents a pillar containing  $N$  LiDAR points, with each point parameterized by  $D$  dimension representation (*e.g.*, the XYZ coordinates and the received reflectance). The PFN is realized by applying fully connected layers on each point within the pillar, then summarizing features of all points through a channel-wise maximum operation. The initial node state  $h_i^0$  is a locally aggregated feature, only including points information within a certain pillar grid.

Next, we elaborate on the message passing process. One iteration step of message propagation is illustrated in Fig. 3. At step  $s$ , a node  $v_i$  aggregates information from all the neighbor nodes  $v_j \in \Omega_{v_i}$  in the  $k$ -NN graph. We define the incoming edge feature from node  $v_j$  as  $e_{j,i}^s$ , indicating the relation between node  $v_i$  and  $v_j$ . Inspired by [55], the incoming edge feature  $e_{j,i}^s$  is given by:

$$e_{j,i}^s = h_j^s - h_i^s \in \mathbb{R}^L, \quad (2)$$

which is an asymmetric function encoding the local neigh-

bor information. Accordingly, we have the message passed from  $v_j$  to  $v_i$ , which is denoted as:

$$m_{j,i}^{s+1} = \phi_\theta([h_i^s, e_{j,i}^s]) \in \mathbb{R}^{L'}, \quad (3)$$

where  $\phi_\theta$  is parameterized by a fully connected layer, which takes as input the concatenation of  $h_i^s$  and  $e_{j,i}^s$ , and yields a  $L'$ -dim feature.

After computing all the pair-wise relations between  $v_i$  and the neighbors  $v_j \in \Omega_{v_i}$  of, we summarize the received  $k$  messages with a maximum operation:

$$m_i^{s+1} = \max_{j \in \Omega_i} (m_{j,i}^{s+1}) \in \mathbb{R}^{L'}, \quad (4)$$

Then, we update the node state  $h_i^s$  with  $h_i^{s+1}$  for node  $v_i$ . The update process should consider both the newly collected message  $m_i^{s+1}$  and the previous state  $h_i^s$ . Recurrent neural network and its variants [16, 47] can adaptively capture dependencies in different time steps. Hence, we utilize Gated Recurrent Unit (GRU) [7] as the update function for its better convergence characteristic. The update process is then formulated as follows:

$$h_i^{s+1} = \text{GRU}(h_i^s, m_i^{s+1}) \in \mathbb{R}^L, \quad (5)$$

In this way, the new node state  $h_i^{s+1}$  contains the information from all the neighbor nodes of  $v_i$ . Moreover, a neighbor node  $v_j$  also collects information from its own neighbors  $\Omega_{v_j}$ . Consequently, after the totally  $S$  iteration steps, node  $v_i$  is able to aggregate information from the high-order neighbors. This effectively enlarges the perceptual range for each pillar grid and enables our model to better recognize objects from a global view.

Note that each pillar corresponds with a spatial coordinate in the x-y plane. Therefore, after performing the iterative message passing, the encoded pillar features are then scattered back as a 3D tensor  $\tilde{I}_t \in \mathbb{R}^{W \times H \times C}$ , which can be further exploited by the 2D CNNs. Here, we leverage the backbone network in [62] to further extract features for  $\tilde{I}_t$ :

$$\mathbf{X}_t = F_B(\tilde{I}_t) \in \mathbb{R}^{w \times h \times c}, \quad (6)$$

where  $F_B$  denotes the backbone network and  $\mathbf{X}_t$  is the spatial features of  $\tilde{I}_t$ . Details of the PMPNet and the backbone network can be found in §3.3.

### 3.2. Attentive Spatiotemporal Transformer GRU

Since the sequential features  $\{\mathbf{X}_t\}_{t=1}^T$  produced by the spatial feature encoding component are regular tensors, we can employ the ConvGRU [1] to fuse these features in our spatiotemporal feature aggregation component. However, it may suffer from two limitations when directly applying the ConvGRU. On the one hand, the interest objects are relatively small in the bird's eye view compared with those

in the 2D images (e.g., an average of  $18 \times 8$  pixels for cars with the pillar size of  $0.25^2 m^2$ ). This may cause the background noise to dominate the results when computing the memory. On the other hand, though the static objects can be well aligned across frames using the GPS data, the dynamic objects with large motion still lead to an inaccurate new memory. To address the above issues, we propose the AST-GRU to equip the vanilla ConvGRU [1] with a spatial transformer attention (STA) module and a temporal transformer attention (TTA) module. As illustrated in Fig. 4, the STA module stresses the foreground objects in  $\{\mathbf{X}_t\}_{t=1}^T$  and produces the attentive new input  $\{\mathbf{X}'_t\}_{t=1}^T$ , while the TTA module aligns the dynamic objects in  $\{\mathbf{H}_{t-1}\}_{t=1}^T$  and  $\{\mathbf{X}'_t\}_{t=1}^T$ , and outputs the attentive old memory  $\{\mathbf{H}'_{t-1}\}_{t=1}^T$ . Then,  $\{\mathbf{X}'_t\}_{t=1}^T$  and  $\{\mathbf{H}'_{t-1}\}_{t=1}^T$  are used to generate the new memory  $\{\mathbf{H}_t\}_{t=1}^T$ , and further produce the final detections  $\{\mathbf{Y}_t\}_{t=1}^T$ . Before giving the details of the STA and TTA modules, we first review the vanilla ConvGRU.

**Vanilla ConvGRU.** GRU model [7] operates on a sequence of inputs to adaptively capture the dependencies in different time steps with a memory mechanism. ConvGRU is a variant of the conventional GRU model, which employs convolution operations rather than the fully connected ones, to reduce the number of parameters and preserve the spatial resolution of the input features. ConvGRU has made promising results on many tasks [32, 49, 23, 50], and has shown better results than the LSTM [43] counterparts in terms of the convergence time [8]. More specifically, ConvGRU contains an update gate  $z_t$ , a reset gate  $r_t$ , a candidate memory  $\tilde{\mathbf{H}}_t$  and a new memory  $\mathbf{H}_t$ . At each time step, the new memory  $\mathbf{H}_t$  (also named as the hidden state) is computed based on the old memory  $\mathbf{H}_{t-1}$  and the new input  $\mathbf{X}_t$ , which can be denoted by the following equations:

$$z_t = \sigma(\mathbf{W}_z * \mathbf{X}_t + \mathbf{U}_z * \mathbf{H}_{t-1}), \quad (7)$$

$$r_t = \sigma(\mathbf{W}_r * \mathbf{X}_t + \mathbf{U}_r * \mathbf{H}_{t-1}), \quad (8)$$

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{W} * \mathbf{X}_t + \mathbf{U} * (r_t \circ \mathbf{H}_{t-1})), \quad (9)$$

$$\mathbf{H}_t = (1 - z_t) \circ \mathbf{H}_{t-1} + z_t \circ \tilde{\mathbf{H}}_t, \quad (10)$$

where ‘\*’ and ‘o’ denote the convolution operation and Hadamard product, and  $\sigma$  is a sigmoid function.  $\mathbf{W}, \mathbf{W}_z, \mathbf{W}_r$  and  $\mathbf{U}, \mathbf{U}_z, \mathbf{U}_r$  are the 2D convolutional kernels. When computing the candidate memory  $\tilde{\mathbf{H}}_t$ , the importance of the old memory  $\mathbf{H}_{t-1}$  and the new input  $\mathbf{X}_t$  is determined by the reset gate  $r_t$ , i.e., the information of  $\tilde{\mathbf{H}}_t$  all comes from  $\mathbf{X}_t$  when  $r_t = 0$ . Additionally, the update gate  $z_t$  decides the degree to which the unit accumulates the old memory  $\mathbf{H}_{t-1}$ , to yield the new memory  $\mathbf{H}_t$ . In §4.2, we show that the vanilla ConvGRU has outperformed the simple point cloud merging [4] and the temporal 3D ConvNet [33]. Next, we present how we promote the vanilla ConvGRU with the STA and TTA modules.

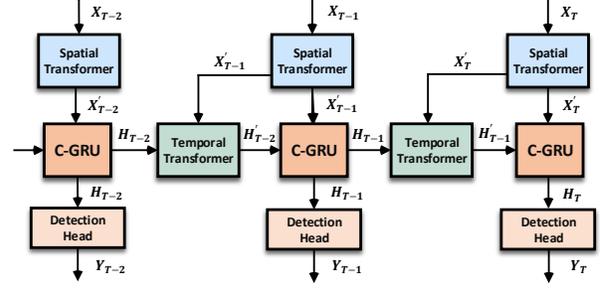


Figure 4: The detailed architecture of the proposed AST-GRU, which consists of a spatial transformer attention (STA) module and a temporal transformer attention (TTA) module. AST-GRU models the dependencies of consecutive frames and produces the attentive new memory  $\{\mathbf{H}_t\}_{t=1}^T$ .

**Spatial Transformer Attention.** The core idea of the STA module is to attend each pixel-level feature  $\mathbf{x} \in \mathbf{X}_t$  with a rich spatial context, to better distinguish a foreground object from the background noise. Basically, a transformer attention receives a query  $\mathbf{x}_q \in \mathbf{X}_t$  and a set of keys  $\mathbf{x}_k \in \Omega_{\mathbf{x}_q}$  (e.g., the neighbors of  $\mathbf{x}_q$ ), to calculate an attentive output  $\mathbf{y}_q$ . The STA is designed as an intra-attention, which means both the query and key are from the same input feature  $\mathbf{X}_t$ .

Formally, given a query  $\mathbf{x}_q \in \mathbf{X}_t$  at location  $q \in w \times h$ , the attentive output  $\mathbf{y}_q$  is computed by:

$$\mathbf{y}_q = \sum_{k \in \Omega_q} A(\phi_Q(\mathbf{x}_q), \phi_K(\mathbf{x}_k)) \circ \phi_V(\mathbf{x}_k), \quad (11)$$

where  $A(\cdot, \cdot)$  is the attention weight.  $\phi_K, \phi_Q$  and  $\phi_V$  are the linear layers that map the inputs  $\mathbf{x}_q, \mathbf{x}_k \in \mathbf{X}_t$  into different embedding subspaces. The attention weight  $A(\cdot, \cdot)$  is computed from the embedded query-key pair  $(\phi_Q(\mathbf{x}_q), \phi_K(\mathbf{x}_k))$ , and is then applied to the neighbor values  $\phi_V(\mathbf{x}_k)$ .

Since we need to obtain the attention for all the query-key pairs, the linear layers,  $\phi_K, \phi_Q$  and  $\phi_V$ , are then achieved by the convolutional layers,  $\Phi_K, \Phi_Q$  and  $\Phi_V$ , to facilitate the computation. Specifically, the input features  $\mathbf{X}_t$  are first embedded as  $\mathbf{K}_t, \mathbf{Q}_t$  and  $\mathbf{V}_t \in \mathbb{R}^{w \times h \times c'}$  through  $\Phi_K, \Phi_Q$  and  $\Phi_V$ . Then, we adjust the tensor shapes of  $\mathbf{K}_t$  and  $\mathbf{Q}_t$  to  $l \times c'$ , where  $l = w \times h$ , in order to compute the attention weight:

$$\tilde{\mathbf{A}} = \text{softmax}(\mathbf{Q}_t \cdot \mathbf{K}_t^T) \in \mathbb{R}^{l \times l} \quad (12)$$

where  $A(\cdot, \cdot)$  is realized as a softmax layer to normalize the attention weight matrix. After that,  $\tilde{\mathbf{A}}$  is employed to aggregate information from the values  $\mathbf{V}_t$  through a matrix multiplication, generating the attentive output  $\tilde{\mathbf{A}} \cdot \mathbf{V}_t$ , with the tensor shape recovered to  $w \times h \times c'$ . Finally, we obtain the spatially enhanced features  $\mathbf{X}'_t$  through a residual operation [14], which can be summarized as:

$$\mathbf{X}'_t = \mathbf{W}_{\text{out}} * (\tilde{\mathbf{A}} \cdot \mathbf{V}_t) + \mathbf{X}_t \in \mathbb{R}^{w \times h \times c} \quad (13)$$

where  $\mathbf{W}_{\text{out}}$  is the output layer of the attention head that maps the embedding subspace ( $c'$ -dim) of  $\tilde{\mathbf{A}} \cdot \mathbf{V}_t$  back to the original space ( $c$ -dim). In this way,  $\mathbf{X}'_t$  contains the information from its spatial context and thus can better focus on the meaningful foreground objects.

**Temporal Transformer Attention.** To adaptively align the features of dynamic objects from  $\mathbf{H}_{t-1}$  to  $\mathbf{X}'_t$ , we apply the modified deformable convolutional layers [65, 64] as a special instantiation of the transformer attention. The core is to attend the queries in  $\mathbf{H}_{t-1}$  with adaptive supporting key regions computed by integrating the motion information.

Specifically, given a vanilla deformable convolutional layer with kernel size  $3 \times 3$ , let  $\mathbf{w}_m$  denotes the learnable weights, and  $p_m \in \{(-1, -1), (-1, 0), \dots, (1, 1)\}$  indicates the predetermined offset in total  $M = 9$  grids. The output  $\mathbf{h}'_q$  for input  $\mathbf{h}_q \in \mathbf{H}_{t-1}$  at location  $q \in w \times h$  can be expressed as:

$$\mathbf{h}'_q = \sum_{m=1}^M \mathbf{w}_m \cdot \mathbf{h}_{q+p_m+\Delta p_m}, \quad (14)$$

where  $\Delta p_m$  is the deformation offset learnt through a separate regular convolutional layer  $\Phi_R$ , *i.e.*,  $\Delta p_m \in \Delta \mathbf{P}_{t-1} = \Phi_R(\mathbf{H}_{t-1}) \in \mathbb{R}^{w \times h \times 2r^2}$ , where the channel number  $2r^2$  denotes the offsets in the x-y plane for the  $r \times r$  convolutional kernel. We can also reformulate Eq. 14 from the perspective of transformer attention as in Eq. 11, such that the attentive output  $\mathbf{h}'_q$  of query  $\mathbf{h}_q$  is given by:

$$\mathbf{h}'_q = \sum_{m=1}^M \mathbf{w}_m \cdot \sum_{k \in \Omega_q} G(k, q + p_m + \Delta p_m) \cdot \phi_V(\mathbf{h}_k), \quad (15)$$

where  $\phi_V$  is an identity function, and  $\mathbf{w}_m$  acts as the weights in different attention heads [64], with each head corresponding to a sampled key position  $k \in \Omega_q$ .  $G(\cdot, \cdot)$  is the attention weight defined by a bilinear interpolation function, such that  $G(a, b) = \max(0, 1 - |a - b|)$ .

The supporting key regions  $\Omega_q$  play an important role in attending  $\mathbf{h}_q$ , which are determined by the deformation offset  $\Delta p_m \in \Delta \mathbf{P}_{t-1}$ . In our TTA module, we compute  $\Delta \mathbf{P}_{t-1}$  not only through  $\mathbf{H}_{t-1}$ , but also through a *motion map*, which is defined as the difference of  $\mathbf{H}_{t-1}$  and  $\mathbf{X}'_t$ :

$$\Delta \mathbf{P}_{t-1} = \Phi_R([\mathbf{H}_{t-1}, \mathbf{H}_{t-1} - \mathbf{X}'_t]) \in \mathbb{R}^{w \times h \times 2r^2}, \quad (16)$$

where  $\Phi_R$  is a regular convolutional layer with the same kernel size as that in the deformable convolutional layer, and  $[\cdot, \cdot]$  is the concatenation operation. The intuition is that, in the *motion map*, the features response of the static objects is very low since they have been spatially aligned in  $\mathbf{H}_{t-1}$  and  $\mathbf{X}'_t$ , while the features response of the dynamic objects remains high. Therefore, we integrate  $\mathbf{H}_{t-1}$  with the *motion map*, to further capture the motions of dynamic

objects. Then,  $\Delta \mathbf{P}_{t-1}$  is used to select the supporting key regions and further attend  $\mathbf{H}_{t-1}$  for all the query regions  $q \in w \times h$  in terms of Eq. 15, yielding a temporally attentive memory  $\mathbf{H}'_{t-1}$ . Since the supporting key regions are computed from both  $\mathbf{H}_{t-1}$  and  $\mathbf{X}'_t$ , our TTA module can be deemed as an inter-attention.

Additionally, we can stack multiple modified deformable convolutional layers to get a more accurate  $\mathbf{H}'_{t-1}$ . In our implementation, we adopt two layers. The latter layer takes as input  $[\mathbf{H}'_{t-1}, \mathbf{H}'_{t-1} - \mathbf{X}'_t]$  to predict the deformation offset according to Eq. 16, and the offset is then used to attend  $\mathbf{H}'_{t-1}$  via Eq. 15. Accordingly, we can now utilize the temporally attentive memory  $\mathbf{H}'_{t-1}$  and the spatially attentive input  $\mathbf{X}'_t$  to compute the new memory  $\mathbf{H}_t$  in the recurrent unit (see Fig. 4). Finally, a RPN detection head is applied on  $\mathbf{H}_t$  to produce the final detection results  $\mathbf{Y}_t$ .

### 3.3. Network Details

**PMPNet.** Our PMPNet is an end-to-end differentiable model achieved by parameterizing all the functions with neural networks. Given a discretized point cloud frame  $\mathbf{I}_t \in \mathbb{R}^{P \times N \times D}$  with  $P$  pillar nodes,  $F_{\text{PFN}}$  is first used to generate the initial node state  $G^0 \in \mathbb{R}^{P \times L}$  for all the nodes (Eq. 1), which is realized by a  $1 \times 1$  convolutional layer followed by a max pooling layer that operates on the  $N$  points. In each iteration step  $s$ , the edge features from the  $K$  neighbor nodes are first collected as  $\tilde{G}^s \in \mathbb{R}^{P \times K \times 2L}$  (Eq. 2) with a concatenation operation. Then the message functions (Eq. 3 and Eq. 4) map the collected features  $\tilde{G}^s$  to  $M^s \in \mathbb{R}^{P \times L'}$ , through a  $1 \times 1$  convolutional layer followed by a max pooling layer performing on the  $K$  messages. The update function (Eq. 5) then updates the node state using a GRU with fully connected layers, by considering both the  $G^s \in \mathbb{R}^{P \times L}$  and  $M^s \in \mathbb{R}^{P \times L'}$ , and outputs  $G^{s+1} \in \mathbb{R}^{P \times L}$ . After  $S$  iteration steps, we get the final node state  $G^S \in \mathbb{R}^{P \times L}$ , and scatter it back to a 3D tensor  $\tilde{\mathbf{I}}_{t \in \mathbb{R}^{W \times H \times L}}$  (Eq. 6).

**Backbone Module.** As in [62], we utilize a 2D backbone network to further extract features for  $\tilde{\mathbf{I}}_{t \in \mathbb{R}^{W \times H \times L}}$ , which consists of three blocks of fully convolutional layers. Each block is defined as a tuple  $(S, Z, C)$ . All the blocks have  $Z \times Z$  convolutional kernels with output channel number  $C$ . The first layer of each block operates at stride  $S$ , while other layers have stride 1. The output features of each block are resized to the same resolution via upsampling layers and then concatenated together to merge the semantic information from different feature levels.

**AST-GRU Module.** In our STA module, all the linear functions in Eq. 11 and  $\mathbf{W}_{\text{out}}$  in Eq. 13 are  $1 \times 1$  convolution layers. In our TTA module, the regular convolutional layers, the deformable convolutional layers and the ConvGRU all have learnable kernels of size  $3 \times 3$ .

Method	Car	Pedestrian	Bus	Barrier	T.C.	Truck	Trailer	Moto.	Cons.	Bicycle	Mean
VIPLICT [35]	71.9	57.0	34.1	38.0	27.3	20.6	26.9	20.4	3.3	0.0	29.9
MAIR [44]	47.8	37.0	18.8	51.1	48.7	22.0	17.6	29.0	7.4	<b>24.5</b>	30.4
PointPillars [24]	68.4	59.7	28.2	38.9	30.8	23.0	23.4	27.4	4.1	1.1	30.5
SARPNET [59]	59.9	69.4	19.4	38.3	44.6	18.7	18.0	29.8	11.6	14.2	32.4
WYSIWYG [18]	79.1	65.0	46.6	34.7	28.8	30.4	40.1	18.2	7.1	0.1	35.0
Tolist [35]	79.4	71.2	42.0	<b>51.2</b>	47.8	<b>34.5</b>	34.8	36.8	9.8	12.3	42.0
<b>Ours</b>	<b>79.7</b>	<b>76.5</b>	<b>47.1</b>	48.8	<b>58.8</b>	33.6	<b>43.0</b>	<b>40.7</b>	<b>18.1</b>	7.9	<b>45.4</b>

Table 1: **Quantitative detection results on the nuScenes 3D object detection benchmark.** T.C. presents the traffic cone. Moto. and Cons. are short for the motorcycle and construction vehicle, respectively. Our 3D *video* object detector outperforms the *single-frame* detectors, achieving state-of-the-art performance on the leaderboard.

**Detection Head.** The detection head in [62] is applied on the attentive memory features. In particular, the smooth L1 loss and the focal loss [31] count for the object bounding box regression and classification, respectively. A cross-entropy loss is used for the orientation classification. For the velocity regression required by the nuScenes benchmark, a simple L1 loss is adopted and shows substantial results.

## 4. Experimental Results

**3D Video Object Detection Benchmark.** We evaluate our algorithm on the challenging nuScenes 3D object detection benchmark [4], since the KITTI benchmark [11] does not provide the point cloud videos. nuScenes is a large-scale dataset with a total of 1,000 scenes, where 700 scenes (28,130 samples) are for training and 150 scenes (6,008 samples) are for testing, resulting  $7\times$  as many annotations as the KITTI. The samples (also named as keyframes) in each video are annotated every 0.5 s with a full 360-degree view, and their point clouds are densified by the 10 non-keyframe sweeps frames, yielding around 300,000 point clouds with 5-dim representation  $(x, y, z, r, \Delta t)$ , where  $r$  is the reflectance and  $\Delta t$  describes the time lag to the keyframe (ranging from 0 s to 0.45 s). Besides, nuScenes requires detecting objects for 10 classes with full 3D boxes, attributes and velocities.

**Implementation Details.** For each keyframe, we consider the point clouds within range of  $[-50, 50] \times [-50, 50] \times [-5, 3]$  meters along the X, Y and Z axes. The pillar resolution on the X-Y plane is  $0.25^2 m^2$ . The pillar number  $P$  used in PMPNet is 16,384, sampled from the total 25,000 pillars, with each pillar containing most  $N = 60$  points. The input point cloud is a  $D = 5$  dimensions representation  $(x, y, z, r, \Delta t)$ , which are then embedded into  $L = L' = 64$  dimensions feature space after the total  $S = 3$  graph iteration steps. The convolutional kernels in the 2D backbone are of size  $Z = 3$  and the output channel number  $C$  in each block is  $(64, 128, 256)$ . The upsampling layer has kernel size 3 and channel number 128. Thus, the final features map produced by the 2D backbone has a resolution of  $100 \times 100 \times 384$ . We calculate anchors for different

classes using the mean sizes and set the matching threshold according to the class instance number. The coefficients of the loss functions for classification, localization and velocity prediction are set to 1, 2 and 0.1, respectively. NMS with IOU threshold 0.5 is utilized when generating the final detections. In both training and testing phases, we feed most 3 consecutive keyframes to the model due to the memory limitation. The training procedure has two stages. In the first stage, we pre-train the spatial features encoding component using the one-cycle policy [45] with a maximum learning rate of 0.003. Then, we fix the learning rate to 0.0002 in the second stage to train the full model. We train 50 epochs for both stages with batch size 3. Adam optimizer [20] is used to optimize the loss functions.

### 4.1. Quantitative and Qualitative Performance

We present the performance comparison of our algorithm and other state-of-the-art approaches on the nuScenes benchmark in Table 1. PointPillars [24], SARPNET [59], WYSIWYG [18] and Tolist [35] are all voxel-based single-frame 3D object detectors. In particular, PointPillars is used as the baseline of our model. WYSIWYG is a recent algorithm that extends the PointPillars with a voxelized visibility map. Tolist uses a multi-head network that contains multiple prediction heads for different classes. Our 3D video object detector outperforms these approaches by a large margin. In particular, we improve the official PointPillars algorithm by 15%. Please note that there is a severe class imbalance issue in the nuScenes dataset. The approach in [63] designs a class data augmentation algorithm. Further integrating with these techniques can promote the performance of our model. But we focus on exploring the spatiotemporal coherence in the point cloud video, and handling the class imbalance issue is not the purpose in this work. In addition, we further show some qualitative results in Fig. 5. Besides the occlusion situation in Fig. 1, we present another case of detecting the distant car (the car on the top right), whose point clouds are especially sparse, which is very challenging for the single-frame detectors. Again, our 3D video object detector effectively detects the distant car using the attentive temporal information.



(a) Detection results from the single-frame 3D object detector [24].



(b) Detection results from our 3D video object detector.

Figure 5: **Detections for the distant cars.** The grey and red boxes indicate the predictions and ground-truths, respectively.

## 4.2. Ablation Study

In this section, we investigate the effectiveness of each module in our algorithm. Since the training samples in the nuScenes is  $7\times$  as many as those in the KITTI (28,130 vs 3,712), it is non-trivial to train multiple models on the whole dataset. Hence, we use a mini train set for validation purposes. It contains around 3,500 samples uniformly sampled from the original train set. Besides, PointPillars [24] is used as the baseline detector in our model.

First, we evaluate our PMPNet in the spatial feature encoding component, by replacing the PFN in PointPillars with PMPNet. As shown in Table 2, it improves the baseline by 2.05%. Second, we validate the ability of each module in the spatiotemporal feature aggregation component (*i.e.*, ConvGRU, STA-GRU and TTA-GRU) through adding each module to the PointPillars. We can see that all these modules achieve better performance than the single-frame detector. Moreover, we compare AST-GRU with other video object detectors. Since each keyframe in nuScenes contains point clouds merged by previous 10 non-keyframe sweeps, the PointPillars baseline, trained on the merged keyframes, can be deemed as the simplest video object detector. Our AST-GRU improves it by 5.98%. Then, we compare AST-GRU with the temporal 3D ConvNet-based method by implementing the late feature fusion module in [33]. Temporal 3D ConvNet can only access a single keyframe label (0.5 s) during training, and aggregating more labels instead impairs the performance. According to Table 2, the 3D ConvNet-based method surpasses the PointPillars baseline by 1.82%.

Components	Modules	Performance	
		mAP	$\Delta$
Single-frame 3D Object Detector	PointPillars (PP)	21.30	-
	PP + PMPNet	23.35	+2.05
3D Video Object Detector	PP + 3D ConvNet	23.12	+1.82
	PP + ConvGRU	23.83	+2.53
	PP + STA-GRU	25.23	+3.93
	PP + TTA-GRU	25.32	+4.02
	PP + AST-GRU	27.28	+5.98
	<b>Full Model</b>	<b>29.35</b>	<b>+8.05</b>

Table 2: **Ablation study for our 3D video object detector.** PointPillars [24] is the reference baseline for computing the relative improvement ( $\Delta$ ).

Aspect	Modules	Performance	
		mAP	$\Delta$
Input Lengths (Full Model)	T=1	16.84	-
	T=2	19.34	+2.50
	T=3	20.27	+3.43
	T=4	20.77	+3.93
	<b>T=5</b>	<b>21.52</b>	<b>+4.68</b>

Table 3: **Ablation study for the input lengths.** Detection results with one input frame are used as the reference baseline.

In contrast, our AST-GRU further outperforms it by 4.16%, which demonstrates the importance of the long-term temporal information. Finally, the full model with the PMPNet achieves the best performance.

Finally, we analyze the effect of the input sequence length. Since each keyframe contains quantities of point clouds that will increase the memory demand, we conduct this experiment without using the point clouds in non-keyframe sweeps. Experimental results with different input lengths are shown in Table. 3, which demonstrates that using the previous long-term temporal information (2.5 s) can consistently gain better performance in 3D object detection.

## 5. Conclusion

This paper proposed a new 3D video object detector for exploring the spatiotemporal information in point cloud video. It has developed two new components: spatial feature encoding component and spatiotemporal feature aggregation component. We first introduce a novel PMPNet that considers the spatial features of each point cloud frame. PMPNet can effectively enlarge the receptive field of each pillar grid through iteratively aggregating messages on a  $k$ -NN graph. Then, an AST-GRU module composed of STA and TTA is presented to mine the spatiotemporal coherence in consecutive frames by using an attentive memory gating mechanism. The STA focuses on detecting the foreground objects, while the TTA aims to align the dynamic objects. Extensive experiments on the nuScenes benchmark have proved the better performance of our model.

## References

- [1] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016. 2, 4, 5
- [2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016. 3
- [3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014. 3
- [4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 2, 5, 7
- [5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 2
- [6] Yilun Chen, Shu Liu, Xiaoqiang Shen, and Jiaya Jia. Fast point r-cnn. In *ICCV*, 2019. 1, 2
- [7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Conference on Empirical Methods in Natural Language Processing*, 2014. 4, 5
- [8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 5
- [9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016. 3
- [10] Lifeng Fan, Wenguan Wang, Siyuan Huang, Xinyu Tang, and Song-Chun Zhu. Understanding human gaze communication by spatio-temporal graph reasoning. In *ICCV*, 2019. 3
- [11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 7
- [12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017. 3
- [13] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, 2005. 3
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5
- [15] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 3
- [16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4
- [17] Jilin Hu, Jianbing Shen, Bin Yang, and Ling Shao. Infinitely wide graph convolutional networks: Semi-supervised learning via gaussian processes. *arXiv preprint arXiv:2002.12168*, 2020. 3
- [18] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *CVPR*, 2020. 7
- [19] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016. 3
- [20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 7
- [21] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018. 2
- [22] Jason Ku, Alex D Pon, and Steven L Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *CVPR*, 2019. 2
- [23] Qiuxia Lai, Wenguan Wang, Hanqiu Sun, and Jianbing Shen. Video saliency prediction using spatiotemporal residual attentive networks. *TIP*, 29:1113–1126, 2019. 5
- [24] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1, 2, 3, 4, 7, 8
- [25] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *CVPR*, 2019. 2
- [26] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *CVPR*, 2019. 2
- [27] Tao Li, Zhiyuan Liang, Sanyuan Zhao, Jiahao Gong, and Jianbing Shen. Self-learning with rectification strategy for human parsing. In *CVPR*, 2020. 3
- [28] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016. 3
- [29] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019. 2
- [30] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018. 2
- [31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 7
- [32] Mason Liu and Menglong Zhu. Mobile video object detection with temporally-aware feature maps. In *CVPR*, 2018. 5
- [33] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018. 3, 5, 8
- [34] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *CVPR*, 2019. 2
- [35] nuTonomy. nuscenes 3d object detection challenge. <https://www.nuscenes.org/object-detection?externalData=all&mapData=all&modalities=Any/>. 7
- [36] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association*

- for *Computational Linguistics*, 5:101–115, 2017. 3
- [37] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 2
- [38] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018. 2
- [39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2
- [40] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *ECCV*, 2018. 3
- [41] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2008. 3
- [42] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 2
- [43] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, 2015. 5
- [44] Andrea Simonelli, Samuel Rota Buló, Lorenzo Porzi, Manuel López-Antequera, and Peter Kotschieder. Disentangling monocular 3d object detection. In *ICCV*, 2019. 7
- [45] Leslie N Smith. Cyclical learning rates for training neural networks. In *WACV*, 2017. 7
- [46] Xibin Song, Peng Wang, Dingfu Zhou, Rui Zhu, Chenye Guan, Yuchao Dai, Hao Su, Hongdong Li, and Ruigang Yang. Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving. In *CVPR*, 2019. 1
- [47] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014. 4
- [48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2
- [49] Wenguan Wang, Xiankai Lu, Jianbing Shen, David J Crandall, and Ling Shao. Zero-shot video object segmentation via attentive graph neural networks. In *ICCV*, 2019. 5
- [50] Wenguan Wang, Jianbing Shen, Ming-Ming Cheng, and Ling Shao. An iterative and cooperative top-down and bottom-up inference network for salient object detection. In *CVPR*, 2019. 5
- [51] Wenguan Wang, Yuanlu Xu, Jianbing Shen, and Song-Chun Zhu. Attentive fashion grammar network for fashion landmark detection and clothing category classification. In *CVPR*, 2018. 3
- [52] Wenguan Wang, Hailong Zhu, Jifeng Dai, Yanwei Pang, Jianbing Shen, and Ling Shao. Hierarchical human parsing with typed part-relation reasoning. In *CVPR*, 2020. 3
- [53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2
- [54] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019. 2
- [55] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):146, 2019. 4
- [56] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2
- [57] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 2
- [58] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *ICCV*, 2019. 1, 2
- [59] Yangyang Ye, Houjin Chen, Chi Zhang, Xiaoli Hao, and Zhaoxiang Zhang. Sarpnet: Shape attention regional proposal network for lidar-based 3d object detection. *Neurocomputing*, 379:53–63, 2020. 7
- [60] Victoria Zayats and Mari Ostendorf. Conversation modeling on reddit using a graph-structured lstm. *Transactions of the Association for Computational Linguistics*, 6:121–132, 2018. 3
- [61] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *International Conference on 3D Vision*, 2019. 2
- [62] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1, 2, 3, 4, 6, 7
- [63] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. 7
- [64] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *ICCV*, 2019. 2, 6
- [65] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 2, 6