





Graph Neural Network and Spatiotemporal Transformer Attention for 3D Video Object Detection From Point Clouds

Junbo Yin, Jianbing Shen , Senior Member, IEEE, Xin Gao , David J. Crandall , and Ruigang Yang 

Abstract—Previous works for LiDAR-based 3D object detection mainly focus on the single-frame paradigm. In this paper, we propose to detect 3D objects by exploiting temporal information in multiple frames, i.e., point cloud *videos*. We empirically categorize the temporal information into short-term and long-term patterns. To encode the short-term data, we present a Grid Message Passing Network (GMPNet), which considers each grid (i.e., the grouped points) as a node and constructs a k -NN graph with the neighbor grids. To update features for a grid, GMPNet iteratively collects information from its neighbors, thus mining the motion cues in grids from nearby frames. To further aggregate long-term frames, we propose an Attentive Spatiotemporal Transformer GRU (AST-GRU), which contains a Spatial Transformer Attention (STA) module and a Temporal Transformer Attention (TTA) module. STA and TTA enhance the vanilla GRU to focus on small objects and better align moving objects. Our overall framework supports both online and offline video object detection in point clouds. We implement our algorithm based on prevalent anchor-based and anchor-free detectors. Evaluation results on the challenging nuScenes benchmark show superior performance of our method, achieving first on the leaderboard (at the time of paper submission) without any “bells and whistles.” Our source code is available at <https://github.com/shenjianbing/GMP3D>.

Index Terms—Point cloud, 3D video object detection, autonomous driving, graph neural network, transformer attention

1 INTRODUCTION

THE past several years have witnessed an explosion of interest in 3D object detection due to its vital role in autonomous driving perception. Meanwhile, LiDAR sensors are becoming indispensable instruments in 3D perception because they provide accurate depth information in complex scenarios such as dynamic traffic environments and adverse weather conditions. The majority of 3D object detectors are dedicated to detection in *single-frame* LiDAR point clouds, i.e., predicting oriented 3D bounding boxes frame-by-frame. However, little work has considered detecting in multi-frame point clouds sequences, i.e., point cloud *videos*. In the newly-released

nuScenes [2] dataset, a point cloud video is defined by a driving scene containing around 400 point cloud frames captured within 20 seconds. In general, point cloud videos are readily available in practical applications and can provide rich spatiotemporal coherence. For example, object point clouds from a single frame may be truncated or sparse due to occlusions or long-distance sampling, while other frames could contain complementary information for recognizing the object. Therefore, *single-frame* 3D object detectors have inherent limitations compared to 3D *video* object detectors (See Fig. 1).

Existing 3D video object detectors typically exploit temporal information in a straightforward way, directly concatenating the point clouds from other frames to a reference frame and then performing detection in the reference frame [3], [4], [5]. This converts the video object detection task to a single-frame object detection task. However, such a simple approach has several limitations. First, in driving scenarios, there is often significant motion blur that causes inaccurate detections. One solution might be to apply ego-motion transformations for these frames, but while this can alleviate the ego-motion, it cannot remedy the motion blur caused by moving objects. Second, this concatenation-based approach usually simply encodes the temporal information with an additional point cloud channel such as timestamp, ignoring the rich feature relations among different frames. Third, as the label information is only provided in the reference frame, such methods suffer more information loss when leveraging long-term point cloud data. Apart from the concatenation-based approach, other papers [6] apply temporal 3D convolutional networks on point cloud videos. Nevertheless, they will encounter temporal information collapse when aggregating features over multiple frames [8].

- Junbo Yin is with the School of Computer Science, Beijing Institute of Technology, Beijing 100811, China. E-mail: yinjunbo@bit.edu.cn.
- Jianbing Shen is with the State Key Laboratory of Internet of Things for Smart City, Department of Computer and Information Science, University of Macau, Macau 999078, China. E-mail: shenjianbingcg@gmail.com.
- Xin Gao is with the Computer, Electrical, and Mathematical Sciences and Engineering (CEMSE) Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia. E-mail: xin.gao@kaust.edu.sa.
- David J. Crandall is with the School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408 USA. E-mail: djcran@indiana.edu.
- Ruigang Yang is with the University of Kentucky, Lexington, KY 40507 USA. E-mail: ryang@cs.uky.edu.

Manuscript received 19 April 2021; revised 4 September 2021; accepted 13 October 2021. Date of publication 9 November 2021; date of current version 30 June 2023.

This work was supported in part by the FDCT under Grants 0154/2022/A3, SKL-IOTSC(UM)-2021-2023, MYRG-CRG2022-00013-IOTSC-ICI, and SRG2022-00023-IOTSC.

(Corresponding author: Jianbing Shen.)

Recommended for acceptance by C. G. Snoek.

Digital Object Identifier no. 10.1109/TPAMI.2021.3125981

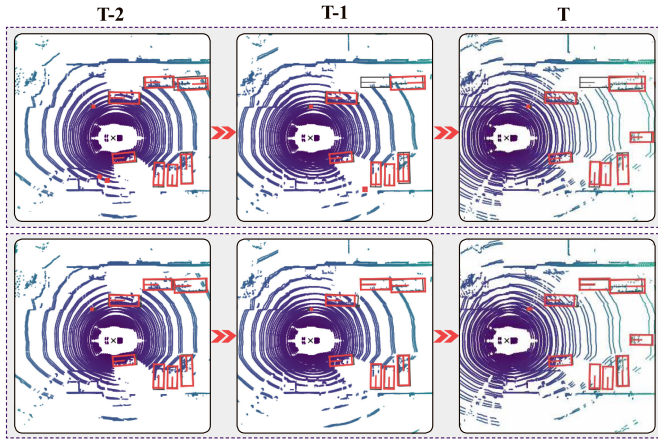


Fig. 1. A representative challenging case in autonomous driving scenarios. A typical single-frame 3D object detector, e.g. [3], suffers from false negatives, due to occlusion (top row). In contrast, our 3D video object detector predicts correctly (bottom row). Red and grey denote the predicted and ground truth boxes, respectively.

How to effectively exploit the temporal information in point cloud videos remains an open challenge. We argue that there are two forms of temporal information: short-term and long-term patterns. In the nuScenes [2] dataset, for example, short-term patterns are point cloud sequences captured within 0.5 seconds (around 10 frames), whereas long-term patterns refer to sequences of 1 to 2 seconds involving dozens of frames. In this work, we aim to present a more effective and general 3D video object detection framework by enhancing prevalent 3D object detectors with both the short-term and long-term temporal cues. In particular, the long-term point clouds are first divided into several short-term ones. We separately encode each short-term sequence with a short-term encoding module, and then adaptively fuse the output features with a long-term aggregation module.

For handling the short-term point cloud data, our short-term encoding module follows the paradigm of modern grid-based detectors [3], [5], [9], [10] that directly concatenate point clouds from nearby frames to a reference frame, but adopts a novel grid feature encoding strategy that models the relations of grids in nearby frames. More precisely, these detectors typically divide the point clouds into regular grids such as voxels and pillars, with each grid containing a fixed number of point clouds. Then PointNet-like modules (e.g., the Voxel Feature Encoding Layer in [9] and the Pillar Feature Network in [3]) are used to extract grid-wise feature representations. A potential problem of these PointNet-like modules is that they focus only on the *individual* grid and ignore the relationships among different grids. Intuitively, certain grids in nearby frames may encode the same object parts, which can be used to improve detection accuracy.

To this end, we propose a graph-based network, Grid Message Passing Network (GMPNet), which iteratively propagates information over different grids. GMPNet treats each *non-empty* grid as a graph node and builds a k -NN graph with the nearby k grids. The relationships among these grids as viewed as edges. The core idea of GMPNet is to iteratively update the node features via the neighbors along the edges and hence mine the rich relations among different grid nodes. At each iteration, a node receives pairwise messages from its neighbors, aggregates these messages, and then updates its

node features. After several iterations, messages from distant grids can be accessed. This flexibly enlarges the receptive field in a non-euclidean manner. Compared with previous PointNet-like modules, GMPNet effectively encourages information propagation among different grids, while also capturing short-term motion cues of objects.

After obtaining individual features extracted by the short-term encoding module, we then turn to Convolutional Gated Recurrent Unit networks (ConvGRU [11]) to further capture the dependencies over these features in our long-term aggregation module. ConvGRUs have shown promising performance in the 2D video understanding field. They adaptively link temporal information over input sequences with a memory gating mechanism. In each time step, the current memory is computed by considering both the current input and the previous memory features. The updated memory is then employed to perform the present task. However, there are two potential limitations when directly applying ConvGRU on multi-frame point cloud sequences. First, modern grid-based detectors tend to transform the point cloud features into the bird's eye view, and the resultant object resolution is much smaller than that in 2D images. For example, with a grid size of $0.25^2 m^2$ and stride of 8, objects such as pedestrians and vehicles occupy just around 1 to 3 pixels in the output feature maps. Therefore, when computing the current memory features, the background noise will inevitably accumulate and decrease detection performance. Second, the spatial features of the current input and the previous memory are not well aligned across frames. Though we could use the ego-pose information to align the static objects over multiple point cloud frames, moving objects still incur motion blur that impairs the quality when updating the memory.

To address these challenges, we present Attentive Spatio-temporal Transformer GRU (AST-GRU), which enhances the vanilla ConvGRU with a Spatial Transformer Attention (STA) module and a Temporal Transformer Attention (TTA) module. In particular, the STA module is derived from [12], [13] and is devised to attend to small objects with the spatial context information. It acts as an intra-attention mechanism, where both the keys and queries reside in the input features. STA views each pixel as a query and the neighbors as keys. By attending the query with context keys, STA can better distinguish foreground pixels from the background ones, and thus emphasize meaningful objects in current memory features.

Furthermore, we describe a TTA module motivated by [14], [15] to align the moving objects. It is composed of modified deformable convolutional layers and behaves as an inter-attention that operates on both the input and previous memory features. TTA treats each pixel in the previous memory as a query and decides the keys by integrating motion information, thus capturing more cues of moving objects. In contrast to the vanilla ConvGRU, the presented AST-GRU effectively improves the quality of the current memory features and leads to better detection accuracy, which is verified on both prevalent anchor-based and anchor-free detectors. Moreover, this also introduces a general methodology that accounts for both online and offline 3D video object detection by flexibly deciding the inputs, e.g., using previous or later frames in a point cloud video. In the offline mode, we further augment our AST-GRU with

a bi-directional feature learning mechanism, which achieves better results in comparison to the online mode.

To summarize, the main contributions of this work are:

- A general point cloud-based 3D video object detection framework is proposed by leveraging both short-term and long-term point cloud information. The proposed framework can easily integrate prevalent 3D object detectors in both online and offline modes.
- We present a novel graph neural network, named GMPNet, to encode short-term point clouds. GMPNet can flexibly mine the relations among different grids in nearby frames and thus capture motion cues.
- To further model the long-term point clouds, an AST-GRU module is introduced to equip the conventional ConvGRU with an attentive memory mechanism, where a Spatial Transformer Attention (STA) and a Temporal Transformer Attention (TTA) are devised to mine the spatiotemporal coherence of long-term point clouds.
- We build the proposed 3D video object detection framework upon both anchor-based and anchor-free 3D object detectors. Extensive evaluations on the large-scale nuScenes benchmark show that our model outperforms all the state-of-the-art approaches on the leaderboard.

This work significantly extends our preliminary conference paper [1] and the improvements are multi-fold. Our previous work [1] only supports 3D object detection in the online mode with an anchor-based detector. In this work, we first provide a more general framework that works with both anchor-based and anchor-free detection settings. In this regard, our approach can be easily incorporated with leading 3D object detectors. Second, we extend our method to both online and offline detection modes, which can benefit more applications with improved accuracy, such as acting as annotation tools. Third, this paper provides a more in-depth discussion on the algorithm with more details including its motivation, technical preliminary, network architecture, and implementation. Fourth, extensive ablation studies are conducted to thoroughly and rigorously assess the broad effectiveness of our model. Finally, we empirically observe that the proposed model outperforms all the algorithms on the nuScenes leaderboard without any “bells and whistles.”

2 RELATED WORK

3D Object Detection. A large amount of work has studied 3D object detection in recent years, in part because of its crucial role in autonomous driving. Existing approaches for 3D object detection can be roughly grouped into three categories.

(1) *2D image-based methods* typically perform detection from monocular [16], [17], [18] or stereo images [19], [20], [21]. These approaches experience significant errors due to loss of depth information, and often turn to geometric priors [22] or additional modules to estimate depth [17] or disparity [19]. Wang *et al.* [21] presented a novel framework by first converting image-based depth maps to pseudo point clouds and then applying the off-the-shelf LiDAR-based detectors.

(2) *3D point cloud-based methods* can leverage LiDAR to measure accurate depth information, and are less sensitive to different illumination and weather conditions. Grid- and point-based methods are the main ways to process the point clouds. The common solution of grid-based methods [1], [3], [9], [23] is to first discretize raw point clouds into regular grids (e.g., voxels [10] or pillars [3]), and then 3D or 2D convolutional networks can be readily applied to extract features. In practice, grid-based approaches are much more efficient than point-based methods, but may suffer from information loss in the quantization process. Point-based methods [24], [25], [26] typically extract features and predict proposals from points with backbones like PointNet++ [27]. They tend to perform better than grid-based methods, but are limited in computational efficiency and memory footprint when the number of point clouds increases. Recently, Shi *et al.* [28] proposed the PV-RCNN detector that combines the merits of both the voxel- and point-based approaches and shows better performance.

(3) *Fusion-based methods* exploit both camera and LiDAR sensors to capture complementary information. MV3D [29] is the seminal work of this family. It takes LiDAR bird’s eye view and front view as well as images as inputs, and combines the proposal-wise features of each view with a deep fusion network. Later, 3D-CVF [30] combined point clouds and images from N multi-view cameras with a cross-view spatial feature fusion strategy. However, the multi-sensor fusion system may not work robustly due to signal synchronization problems. In this paper, we focus on LiDAR-only grid-based approaches since they are more prevalent in current autonomous driving applications.

Spatiotemporal Models in Point Clouds. In contrast to the aforementioned work that performs frame-by-frame detections, we aim to address the multiple-frame 3D object detection by exploiting temporal information. Only a few papers have explored spatiotemporal models in point clouds. Luo *et al.* [6] utilized a temporal 3D ConvNet to aggregate multi-frame point clouds. It performs multiple tasks simultaneously including detection, tracking, and motion prediction, but requires a considerable number of parameters for operating on 4D tensors. Liang *et al.* [7] also addressed the framework of joint detection, tracking, and motion prediction for improving efficiency and accuracy. Later, Choy *et al.* [31] improved the convolutional layers in [6] with sparse operations. However, their approach encounters feature collapse when downsampling the features in the temporal domain, and fails to leverage full label information in all frames.

More recently, Huang *et al.* [32] proposed a contemporary work with ours that focuses on a point-based recurrent gating mechanism. They adopt a Sparse Conv U-Net to extract spatial features in each frame, and then apply LSTMs on points with high semantic scores to fuse information across frames and save computation. Due to the restriction of the backbone, this approach cannot be adapted to state-of-the-art single-frame detectors. In contrast, our work can be easily integrated into prevalent grid-based 3D object detectors [3], [5] by processing the spatiotemporal information in point cloud videos.

It is worth mentioning that some 2D video object detection work [33], [34] explores attention modules for modeling

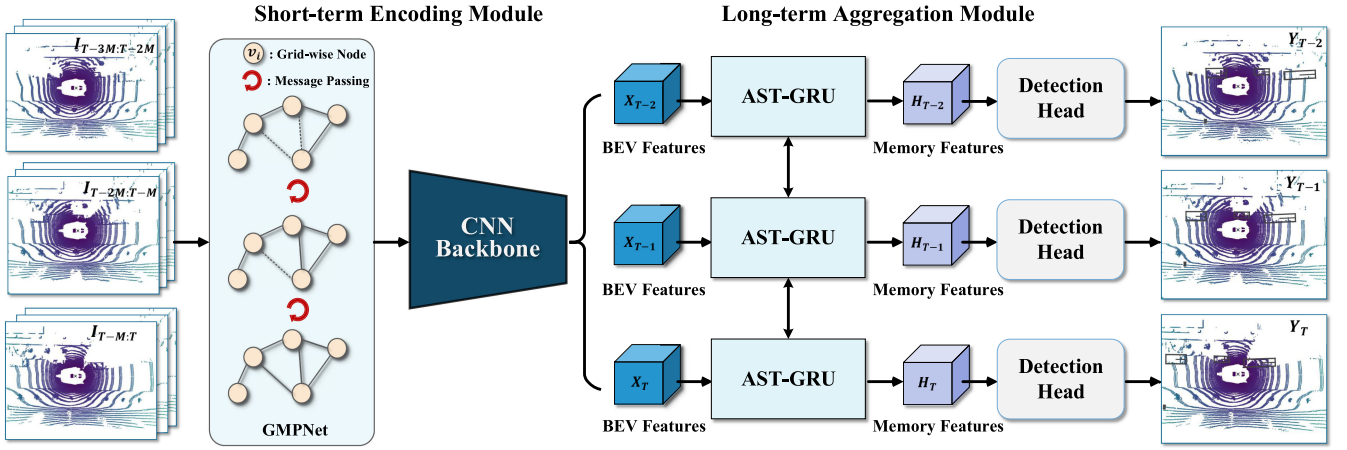


Fig. 2. Schematic of our point cloud-based 3D video object detection framework. The input $T \times M$ frames are first divided into T groups of short-term data with each merging M frames. Then, the short-term encoding module extracts the BEV features for each short-term data with a Grid Message Passing Network (GMPNet) followed by a CNN backbone. Afterwards, the long-term aggregation module further captures the dependencies in these short-term features with an Attentive Spatiotemporal Transformer GRU (AST-GRU). Finally, the detection head receives the enhanced memory features and produces the detection results.

temporal information. However, for 3D point cloud data with a bird's eye view, there are typically more moving objects containing sparse points with shorter lifespans. Our work aims to address these new challenges in 3D video object detection by aggregating point features in consecutive frames.

Graph Neural Networks. The concept of Graph Neural Networks (GNNs) was first proposed by Scarselli *et al.* [35]. They extended the Recurrent Neural Networks (RNNs) and used GNNs to directly process graph-structured data. GNNs encode the underlying relationships among nodes of a graph and mine rich information in the data processing step. Many variants have been developed to improve the representation capability of the original GNNs. Here, we categorize them into two classes according to the type of information propagation step: convolution and gate mechanism.

First, Graph Convolutional Networks (GCNs) [36], [37], [38], [39], [40] generalize convolution to graph data and update nodes via stacks of graph convolutional layers. GCNs typically compute in the spectral domain with graph Fourier transformations. Recently, various applications [41], [42], [43] have been explored using GCNs and have achieved promising results.

Second, multiple papers [44], [45], [46], [47] adopted recurrent gating units to propagate and update information across nodes. For instance, Li *et al.* [44] exploited Gated Recurrent Units (GRUs) to describe node states by aggregating messages from neighbors. After that, Gilmer *et al.* [48] proposed a generalized framework that formulates the graph computation as a parameterized Message Passing Neural Network (MPNN), and these models have proven successful in various tasks involving graph data [49], [50], [51]. Our GMPNet is also inspired by MPNNs, encoding the features of short-term point clouds and mining motion cues among grid-wise nodes by iterative message passing.

3 OUR 3D VIDEO OBJECT DETECTION FRAMEWORK

In this section, we first present the overall pipeline of our method in Section 3.1. Then, we briefly review the general

formulations of Message Passing Neural Network [48] and ConvGRU [11] in Sections 3.2 and 3.3, respectively, since our work builds upon them. Afterwards, we describe the designs of Grid Message Passing Network (GMPNet) and Attentive Spatiotemporal Transformer GRU (AST-GRU) in Sections 3.4 and 3.5, respectively. The main part of this paper assumes online object detection, but in Section 3.6 we present an offline pipeline that learns with a bi-directional ConvGRU, and further improves detection performance. Finally, we provide more detailed information on our network architecture in Section 4.2.

3.1 Overview

As illustrated in Fig. 2, our framework consists of a short-term encoding module and a long-term aggregation module. Assuming that the input long-term point cloud sequence $\{I_t\}_{t=1}^{T \times M}$ contains $T \times M$ frames in total, we first divide it into multiple short-term clips $\{I_t\}_{t=1}^T$, with each I_t containing concatenated point clouds within M nearby frames (e.g., $M = 10$ in nuScenes). Then, LiDAR pose information is leveraged to align point clouds over frames in $\{I_t\}_{t=1}^T$ to eliminate the influence of ego-motion. Next, each I_t is quantized into evenly-distributed grids, and forwarded to the short-term encoding module to extract bird's eye view features $\{X_t\}_{t=1}^T$. Here, GMPNet is applied before the 2D CNN backbone to capture the relations among grids. After that, in the long-term aggregation module, AST-GRU learns long-term dependencies of sequential features $\{X_t\}_{t=1}^T$. In particular, at each time step t , the unit receives current input features X_t as well as memory features H_{t-1} , and produces updated memory features H_t with an attentive gating mechanism. In this way, H_t preserves information from both previous and current frames, which facilitates the subsequent object detection task. Finally, detection heads such as classification and regression can be used on H_t to obtain final detections $\{Y_t\}_{t=1}^T$.

For training the 3D video object detector, both anchor-based [3], [10] and anchor-free [5], [52] loss functions can be applied. Furthermore, our framework works in both online and offline inference modes: in online mode, only previous

frames are employed to help detect in the current frame, while in offline mode, both previous and future frames are used, which further improves detection results.

3.2 Message Passing Neural Networks

Message Passing Neural Networks (MPNNs) [48] unify various promising neural networks that operate on graph-structured data [44], [45], [53], [54]. Formally, they define a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes $v_i \in \mathcal{V}$ and edges $e_{i,j} \in \mathcal{E}$. The core of MPNN is to iteratively pass messages between different nodes and mine the diverse relations of them. At each time step t , let h_i^t denote the state features of node v_i , and $e_{j,i}^t$ represents edge features of $e_{i,j}$ that describe the information passed from node v_j to v_i . MPNN aggregates information for node v_i with the neighbors $v_j \in \Omega_{v_i}$, and infers updated state features h_i^{t+1} based on the received messages. It runs for K time steps and thus captures long range interactions among the nodes.

More specifically, MPNN includes a message function $M(\cdot)$ and a node update function $U(\cdot)$. At each time step t , $M(\cdot)$ summarizes the message $m_{j,i}^{t+1}$ passed from the neighbor node $v_j \in \Omega_{v_i}$ to v_i , then obtains the aggregated message features m_i^{t+1} . Notice that the message features $m_{j,i}^{t+1}$ are computed by considering both node state features and edge features, which is denoted as follows:

$$\begin{aligned} m_i^{t+1} &= \sum_{j \in \Omega_i} m_{j,i}^{t+1} \\ &= \sum_{j \in \Omega_i} M(h_i^t, h_j^t, e_{j,i}^t). \end{aligned} \quad (1)$$

Then, according to the collected message m_i^{t+1} , the update function $U(\cdot)$ refines the previous state features h_i^t for node v_i , and produces the updated state features h_i^{t+1}

$$h_i^{t+1} = U(h_i^t, m_i^{t+1}). \quad (2)$$

In MPNN, both $M(\cdot)$ and $U(\cdot)$ are parameterized by weight-sharing neural networks and all the operations can be learnt with gradient-based optimization. After one time step, a node accesses information from its neighbor nodes. After K time step, the information from long-range nodes can be obtained. In this work, we extend MPNN to the context of LiDAR point clouds by treating grids as nodes. Our proposed GMPNet can encode grid-wise features by mining spatiotemporal relations in short-term point clouds.

3.3 ConvGRU Network

Gated Recurrent Unit (GRU) models [55] are streamlined variants of Recurrent Neural Networks (RNNs) [56], [57], [58], which were originally devised for machine translation and video understanding by capturing the dependencies of input sequences. They simplify the computation of RNNs and achieve comparable performance. Later, Ballas *et al.* [11] proposed convolutional GRUs (ConvGRUs), which use convolutional layers instead of the fully-connected ones in the original GRU. This not only preserves better spatial resolution of the input features, but also significantly reduces the number of parameters. ConvGRU has shown promising results on many vision tasks [8], [59], [60], [61]. Basically, ConvGRU contains an update gate z_t , a reset gate r_t , a

candidate memory \tilde{H}_t , and a current memory H_t . At each time step, it computes the current memory H_t (or the hidden state) according to the previous memory H_{t-1} and the current input X_t

$$\begin{aligned} z_t &= \sigma(W_z * X_t + U_z * H_{t-1}), \\ r_t &= \sigma(W_r * X_t + U_r * H_{t-1}), \\ \tilde{H}_t &= \tanh(W * X_t + U * (r_t \circ H_{t-1})), \\ H_t &= (1 - z_t) \circ H_{t-1} + z_t \circ \tilde{H}_t, \end{aligned} \quad (3)$$

where $*$ denotes the convolution operation, \circ is the Hadamard product and σ is as a sigmoid activation function. W, W_z, W_r and U, U_z, U_r are the 2D convolutional kernels. The reset gate r_t determines how much of the past information from H_{t-1} to forget so as to produce the candidate memory \tilde{H}_t . For example, the information of \tilde{H}_t all comes from the current input X_t when $r_t = 0$. The update gate z_t decides the degree to which the current memory H_t accumulates the previous information from H_{t-1} . Our AST-GRU significantly improves vanilla ConvGRUs by integrating with STA and TTA modules, which enforces the network to focus on meaningful objects in long-term point clouds.

3.4 Grid Message Passing Network

As introduced in Section 3.1, in the short-term encoding module, we aggregate the K short-term point cloud frames by concatenating them into a single frame. To extract features on this merged frame, dominant approaches tend to quantize the point clouds into regular grids such as voxels or pillars, and then use modules like Voxel Feature Encoding Layer [9] or the Pillar Feature Network [3] to encode the grids. These modules typically consider an *individual* grid, which fails to capture the spatiotemporal relations between the nodes from nearby frames, as well as limiting the expressive power due to the local representation. To this end, we propose Grid Message Passing Network (GMPNet) to mine the spatiotemporal relations of the grids, which results in a non-local representation. GMPNet views each *non-empty* grid as a node, and the relations between grids as edges. It iteratively passes messages between the grids and updates the grid-level representation accordingly. Besides, GMPNet exploits the non-euclidean characteristics for point cloud representation, providing a complementary perspective compared with the CNN backbone.

Specifically, given a merged point cloud frame I_t , we first uniformly discretize it into a set of grids \mathcal{V} . The grid can be either a voxel or a pillar, determined by the baseline detectors. Then a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed, where each node $v_i \in \mathcal{V}$ represents a non-empty grid and each edge $e_{j,i} \in \mathcal{E}$ holds the information passed from node v_j to v_i . Furthermore, we define \mathcal{G} as a k -nearest neighbor (k -NN) graph to save computations, which means that each node can directly access information from the K spatial neighbors (also known as the first order neighbors). The goal of GMPNet is to adaptively update feature representation h_i for each grid-wise node v_i by integrating information from long-range nodes, i.e., the higher-order neighbors. Given a grid-wise node v_i , we first use a simplified PointNet [27] module to abstract its initial state features h_i^0 , which map a set of points within v_i to an L -dim vector. The simplified

PointNet is composed of fully connected layers $f(\cdot)$ and a max-pooling operation:

$$h_i^0 = \max\{f(V_i)\} \in \mathbb{R}^L, \quad (4)$$

where $V_i \in \mathbb{R}^{N \times D}$ denotes the grid v_i with N point clouds parameterized by D -dim representation (e.g., the XYZ coordinates and the reflectance of LiDAR).

Next, we elaborate the message passing and node state updating in GMPNet, following the formulation presented in Section 3.2. At time step s , v_i aggregates information from its neighbors $v_j \in \Omega_{v_i}$ according to Eq. (1). The incoming edge features $e_{j,i}^s$ are defined as

$$e_{j,i}^s = h_j^s - h_i^s \in \mathbb{R}^L, \quad (5)$$

which is an asymmetric function encoding the local neighbor information. Accordingly, the message passed from v_j to v_i is denoted as

$$m_{j,i}^{s+1} = \phi_\theta([h_i^s, e_{j,i}^s]) \in \mathbb{R}^{L'}, \quad (6)$$

where ϕ_θ is a fully connected layer denoting the message function $M(\cdot)$ in Eq. (1). It receives the concatenation of h_i^s and $e_{j,i}^s$ and yields an L' -dim feature. We then summarize the received messages from K neighbors with a max-pooling operation

$$m_i^{s+1} = \max_{j \in \Omega_i} \{m_{j,i}^{s+1}\} \in \mathbb{R}^{L'}. \quad (7)$$

Afterwards, with the collected message m_i^{s+1} , we update the state features h_i^s for node v_i in terms of Eq. (2). Here, a GRU [55] is used as the update function $U(\cdot)$ to adaptively preserve the information in different time steps

$$h_i^{s+1} = \text{GRU}(h_i^s, m_i^{s+1}) \in \mathbb{R}^L. \quad (8)$$

After one time step, v_i contains information from the neighbors $v_j \in \Omega_{v_i}$. Moreover, the neighbor node v_j also holds information from its own neighbors Ω_{v_j} . Therefore, v_i can aggregate information from long-range nodes after a total of S time steps. The message passing and node updating processes are illustrated in Fig. 3.

We obtain the final grid-wise feature representation v_i by applying another fully connected layer ϕ'_θ on h_i^S

$$v_i = \phi'_\theta(h_i^S) \in \mathbb{R}^L. \quad (9)$$

After that, all the grid-wise features are then scattered back to a regular tensor \tilde{I}_t , e.g., $\tilde{I}_t \in \mathbb{R}^{W \times H \times L}$ with the PointPillars [3] baseline. Finally, we apply the CNN backbone in [9] to further extract features for I_t

$$X_t = F_B(\tilde{I}_t) \in \mathbb{R}^{w \times h \times c}, \quad (10)$$

where F_B is the backbone network and X_t is the obtained short-term features of I_t . This leads to a reduced resolution for I_t , facilitating the subsequent long-term aggregation module. More details of the GMPNet and the CNN backbone can be found in Section 4.2.

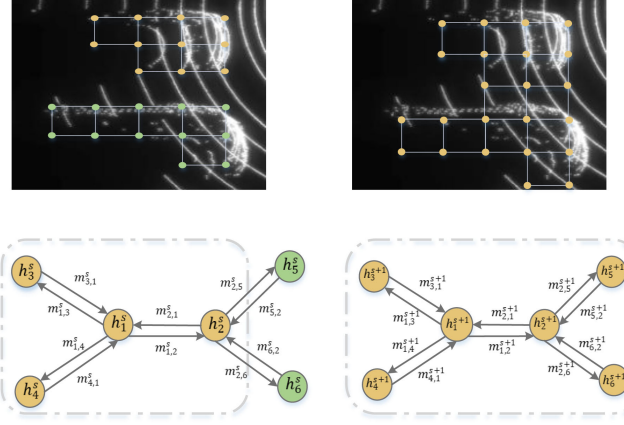


Fig. 3. Illustration of one iteration step for message propagation, where h_i is the state of node v_i . In step s , the neighbors for h_1 are $\{h_2, h_3, h_4\}$ (within the gray dashed line), presenting the grids of the top car. After receiving messages from the neighbors, the receptive field of h_1 is enlarged in step $s+1$. This indicates the relations with the bottom car are modeled after message propagation.

3.5 Attentive Spatiotemporal Transformer GRU

Recall that we have divided the input point cloud sequences $\{I_t\}_{t=1}^{T \times M}$ into multiple short-term clips $\{I_t\}_{t=1}^T$, with each clip I_t including concatenated point clouds from M neighbor frames. Then, in the short-term encoding module, we independently described the short-term features X_t for each I_t . Here, we further capture the long-term dynamics over the sequential features $\{X_t\}_{t=1}^T$ in the long-term aggregation module by exploiting the Attentive Spatiotemporal Transformer GRU (AST-GRU). AST-GRU is an extension of ConvGRU that adaptively mines the spatiotemporal dependencies in $\{X_t\}_{t=1}^T$. In particular, it improves the vanilla ConvGRU in two ways, inspired by the transformer attention mechanism [12], [62]. First, dominant approaches usually perform detection on birds-eye view feature maps [3], [5], [9], [10]. However, objects of interest are much smaller in such views compared with front views in 2D images. For example, using PointPillars [3] with a voxel size of $0.25^2 m^2$, a vehicle (e.g., with size of $4m \times 2m$) remains only 2 pixels in X_t after the feature extractor with common stride of 8. This causes difficulties for the recurrent unit, because the background noise inevitably accumulates across time in H_t .

To address this problem, we propose a spatial transformer attention (STA) module by attending each pixel in X_t with its rich spatial context, helping the detectors to focus on foreground objects. Second, for a recurrent unit, the received current input X_t and previous memory H_{t-1} are not spatially aligned. Although we have aligned the static objects with the ego-pose information, dynamic objects with large motion create an inaccurate current memory H_t . Therefore, a temporal transformer attention (TTA) module is introduced to align the moving objects in H_t , which exploits modified deformable convolutional networks to capture the motion cues in H_{t-1} and X_t . Next, we elaborate the workflow of our STA and TTA modules.

Spatial Transformer Attention. Motivated by the transformer attention in [12], we propose STA to stress the foreground pixels in X_t and suppress the background ones. In particular, each pixel $x_q \in X_t$ is considered as a query input and its context pixels $x_k \in \Omega_{x_q}$ are viewed as keys. Since

both the queries and keys are from the same feature map X_t , STA can be grouped into an intra-attention family. For each query, its corresponding keys are embedded as values and the output of STA is the weighted sum of the values. We compute the weight of values by comparing the similarity between the query and the corresponding key, such that the keys that have the same class as the query could contribute to the output.

Formally, given an input query-key pair $x_q, x_k \in X_t$, we first map them into different subspaces with linear layers $\phi_Q(\cdot)$, $\phi_K(\cdot)$ and $\phi_V(\cdot)$ to obtain embedded feature vectors for the query, key, and value, respectively. Then, the attentive output y_q for a query x_q is calculated as

$$y_q = \sum_{k \in \Omega_q} A(\phi_Q(x_q), \phi_K(x_k)) \cdot \phi_V(x_k), \quad (11)$$

where $A(\cdot, \cdot)$ is the function to compute the attention weight. In practice, STA needs to compute the attention for all the query positions simultaneously. Therefore, we replace ϕ_K , ϕ_Q , and ϕ_V with convolutional layers, Φ_K , Φ_Q , and Φ_V , such that STA can be optimized with matrix multiplication. Specifically, the input features X_t are first embedded as K_t , Q_t and $V_t \in \mathbb{R}^{w \times h \times c'}$ through Φ_K , Φ_Q and Φ_V . Then, we define the weight function $A(\cdot, \cdot)$ as a dot-product operation followed by a softmax layer to measure the similarity of query-key pairs. To compute the attention weight \tilde{A} , we reshape K_t and Q_t to $l \times c'$ ($l = w \times h$) for saving computation

$$\tilde{A} = \text{softmax}(Q_t \cdot K_t^T) \in \mathbb{R}^{l \times l}. \quad (12)$$

Afterwards, we multiply the attention weight \tilde{A} by values V_t to obtain the attention output $\tilde{A} \cdot V_t$. Next, the shape of the output is recovered back to $w \times h \times c'$, and head layers W_{out} are employed to determine the specific mode of attention. Finally, we obtain the attention features X'_t via a residual operation [63]. These steps can be summarized as

$$X'_t = W_{\text{out}} * (\tilde{A} \cdot V_t) + X_t \in \mathbb{R}^{w \times h \times c}, \quad (13)$$

where attention head W_{out} also maps the feature subspace of $\tilde{A} \cdot V_t$ (e.g., c' -dim) back to the original space (e.g., c -dim in X_t). The resultant output X'_t aggregates the information from the context pixels and thus can better focus on small foreground objects while suppressing background noise accumulated in memory features.

Temporal Transformer Attention. The basic idea of TTA is to align the spatial features of H_{t-1} and X'_t so as to give a more accurate current memory H_t . Here we use modified deformable convolutional layers [14], [15] as a special instantiation of the transformer attention. TTA treats each pixel $h_q \in H_{t-1}$ as a query and determines the positions of keys by attending the current input X'_t , thus capturing the temporal motion information of moving objects. TTA belongs to the inter-attention since it involves both H_{t-1} and X'_t .

Specifically, we first describe the regular deformable convolutional network. Let w_m denote the convolutional kernel with size 3×3 , and $p_m \in \{(-1, -1), (-1, 0), \dots, (1, 1)\}$ indicate the predefined offsets of the kernel in $M = 9$ grids. Given a pixel-wise input $h_q \in H_{t-1}$, the output h'_q of this

deformable convolutional layer can be expressed as

$$h'_q = \sum_{m=1}^M w_m \cdot h_{q+p_m+\Delta p_m}, \quad (14)$$

where Δp_m is the deformation offset learnt through another regular convolutional layer Φ_R to model spatial transformations, i.e., $\Delta p_m \in \Delta P_{t-1} = \Phi_R(H_{t-1}) \in \mathbb{R}^{w \times h \times 2r^2}$, where $2r^2$ represents the offsets in both x and y directions.

Following the perspective of transformer attention in Eq. (11), we could also reformulate Eq. (14) as

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

where h'_q is the attentive output of the query h_q by a weighted sum on the $M = 9$ values denoted by $\phi_V(h_k)$. Notice that ϕ_V is an identity function here, such that $\phi_V(h_k) = h_k$. The attention weight is calculated through the kernel weight w_m followed by a bilinear interpolation function $G(\cdot, \cdot)$:

$$G(a, b) = \max(0, 1 - |a - b|), \quad (16)$$

which decides the sampling positions of the keys in the supporting regions Ω_q .

Since the interest objects have moved from H_{t-1} to X'_t , our TTA integrates the information in X'_t to compute a refined offset $\Delta p_m \in \Delta P_{t-1}$, and therefore adjust the sampling positions of the keys accordingly. In particular, we define a *motion map* with the difference of H_{t-1} and X'_t , and then compute ΔP_{t-1} as

$$\Delta P_{t-1} = \Phi_R([H_{t-1}, H_{t-1} - X'_t]) \in \mathbb{R}^{w \times h \times 2r^2}, \quad (17)$$

where Φ_R is a regular convolutional layer with the kernel size 3×3 to predict the offsets, and $[\cdot, \cdot]$ is the concatenation operation. The intuition of the *motion map* is that feature responses of static objects are very low since they have been spatially aligned in H_{t-1} and X'_t , while the responses of moving objects remain high. By combining the salient features in the *motion map*, TTA can focus more on moving objects. Afterwards, the output ΔP_{t-1} is used to determine the regions of keys and further attend H_{t-1} for all the queries $q \in w \times h$ in terms of Eq. (15), yielding a temporally attentive memory H'_{t-1} . Additionally, we could stack multiple such modified deformable convolutional layers to further refine H'_{t-1} . In our implementation, two layers are employed, where the latter layer receives H'_{t-1} and updates it similar to Eq. (17).

Consequently, we have the temporally attentive previous memory H'_{t-1} and the spatially attentive current input X'_t . This leads to an enhanced current memory H_t which contains richer spatiotemporal information and produces better detection results Y_t after being equipped with detection heads (see Fig. 4). We have described our AST-GRU in an online detection setting. In Section 3.6, we present an offline detection setting by exploiting bidirectional AST-GRU.

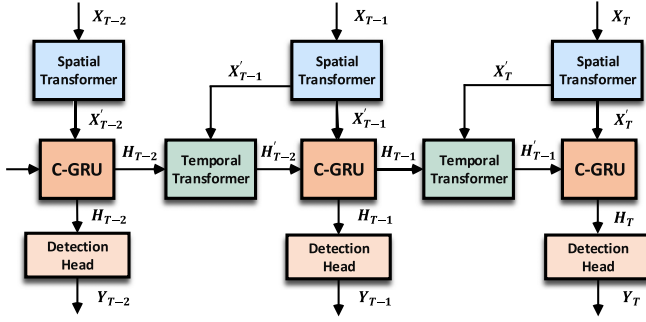


Fig. 4. Illustration of our proposed AST-GRU in the online mode. It consists of a spatial transformer attention (STA) module and a temporal transformer attention (TTA) module. AST-GRU captures the dependencies from a long-term perspective and produces the enhanced memory features $\{H_t\}_{t=1}^T$.

3.6 Offline 3D Video Object Detection

With the proposed AST-GRU, we have achieved better performance for online 3D video object detection. However, we can provide a stronger 3D video object detector by exploring both the past and future frames, which will facilitate more casual applications in autonomous driving. For example, when annotating LiDAR frames for autonomous driving scenes, it is easy to access future frames in a scene. Our offline 3D video object detector could act as an annotation tool by providing more accurate initial 3D bounding boxes. This may greatly increase the productivity of annotators compared with using a frame-by-frame detector. To this end, we further adapt AST-GRU to the bidirectional offline setting by capturing temporal information in both past and future frames.

Concretely, by considering the attentive current input X'_t and the previous memory H'_{t-1} , we have acquired the forward memory features H'_t . To obtain a more powerful representation for the current frame, we can better integrate the backward information from future features H'_{t+1} . Similar to Eq. (3), we use the following equations to capture the rich temporal information preserved in H'_{t+1} and compute the backward memory features H^b_t

$$\begin{aligned} z_t^b &= \sigma(W_z^b * X'_t + U_z^b * H'_{t+1}), \\ r_t^b &= \sigma(W_r^b * X'_t + U_r^b * H'_{t+1}), \\ \tilde{H}_t^b &= \tanh(W^b * X'_t + U^b * (r_t^b \circ H'_{t+1})), \\ H_t^b &= (1 - z_t^b) \circ H'_{t+1} + z_t^b \circ \tilde{H}_t^b, \end{aligned} \quad (18)$$

where W^b, W_z^b, W_r^b and U^b, U_z^b, U_r^b are sets of learnable parameters included in the backward AST-GRU. The reset gate r_t^b and update gate z_t^b measure the importance of the future information H'_{t+1} to the current input X'_t .

In this way, our bidirectional AST-GRU obtains memory features from both forward and backward units. Then, we combine these features with a concatenation operation to aggregate the information and obtain an enhanced memory H'_t , which is calculated as

$$H'_t = [H_t^f, H_t^b] \in \mathbb{R}^{w \times h \times 2c}. \quad (19)$$

Afterwards, detection heads can be directly applied on H'_t to produce 3D detection results. The proposed bidirectional AST-GRU exploits the spatiotemporal dependencies from

both the past and future, which offers complementary cues for detection in the current frame. In Section 4.4, we show that it outperforms the unidirectional AST-GRU by a large margin.

4 EXPERIMENTAL RESULTS

We empirically evaluate our algorithm on the large-scale nuScenes benchmark. Since our framework is agnostic to the detectors, we demonstrate the performance of our algorithm based on two prevalent 3D object detectors: anchor-based PointPillars [3] and anchor-free CenterPoint [5]. The detailed network architecture and main detection results are presented in Sections 4.2 and 4.3, respectively. Afterwards, we provide comprehensive ablation studies to assess each module of our algorithm in Section 4.4.

4.1 3D Video Object Detection Benchmark

Since point cloud videos are not available in the commonly-used KITTI benchmark [74], we turn to the more challenging nuScenes benchmark [2] to evaluate our algorithm and compare with other approaches on the leaderboard. The nuScenes dataset is composed of 1,000 driving scenes (i.e., video clips), with each scene containing around 400 frames of 20 seconds in length. The annotations are conducted every 10 frames for 10 object classes, and the training set provides 7 times as many annotations as the KITTI dataset. There are 700 scenes (28,130 annotations) for training, 150 scenes (6,019 annotations) for validation and 150 scenes (6,008 annotations) for testing. nuScenes utilizes a 32-beam LiDAR with a 20Hz capture frequency, resulting in approximately 30k points in each frame with a full 360-degree view. The official evaluation protocol for 3D object detection defines an NDS (nuScenes detection score) metric, which is a weighted sum of mean Average Precision (mAP) and several True Positive (TP) metrics. In contrast to the definition in KITTI, mAP in nuScenes measures the varying center distance (i.e., 0.5m, 1m, 2m and 4 m) between the predictions and ground truths in the bird's eye view. Other TP metrics consider multiple aspects to measure the quality of the predictions, i.e., box location, size, orientation, attributes and velocity.

To build our point cloud-based video object detector, we follow the common implementation in nuScenes benchmark by merging the 10 previous non-keyframe sweeps (0.5s) to corresponding keyframes with ego-pose information. These sweeps are deemed as short-term data, while the merged keyframes are viewed as long-term data. For our online model, two previous keyframes (1s) are used to detect in the current frame. As for the offline model, we use 0.5s temporal information from both previous and future keyframes.

4.2 Implementation Details

Architecture. We achieve 3D video object detection using both anchor-based and anchor-free LiDAR-based detectors. For the anchor-based baseline, we choose the official PointPillars [3] provided by nuScenes benchmark. For the anchor-free baseline, the CenterPoint models [5] with PointPillars or VoxelNet backbones are adopted in our framework. Specifically, for each merged point cloud keyframe, we define 5-dim input features $(x, y, z, r, \Delta t)$ for the points,

where r is the LiDAR intensity and Δt formulates the time lag to the keyframe that ranges from $0s$ to $0.5s$. We consider the points whose coordinates locate within $[-61.2, 61.2] \times [-61.2, 61.2] \times [-10, 10]$ meters along the X, Y and Z axes. The grid size for grouping points is set as $0.25 \times 0.25 \times 8$ in PointPillars backbone. While in VoxelNet backbone, we adopt a smaller size, $0.075 \times 0.075 \times 0.2$, to obtain better detection performance and compare with other approaches on the leaderboard. In the subsequent paragraphs, we mainly elaborate the architecture details based on the PointPillars backbone due to space limitations, and all the modifications can be applied on VoxelNet backbone accordingly.

In the short-term encoding module, we implement GMPNet by sampling 16,384 grid-wise nodes with Farthest Point Sampling [27], and build the k -NN graph with $K = 20$ neighbors. A grid v_i contains $N = 60$ points with $D = 5$ representations, which is embedded into feature space with channel number $L = 64$ to get the node features. This is achieved by a 1×1 convolutional layer followed by a max-pooling operation, which yields the initial state features $G^0 \in \mathbb{R}^{16,384 \times 64}$ (Eq. (4)) for all the nodes. Then, in the iteration step s , we obtain the message features $M^s \in \mathbb{R}^{16,384 \times 64}$ by performing 1×1 convolutional layer and max-pooling over the neighbors based on the edge features $E^s \in \mathbb{R}^{16,384 \times 20 \times 128}$ (Eqs. (5) to (7)). Afterwards, the updated node state $G^{s+1} \in \mathbb{R}^{16,384 \times 64}$ is computed by applying GRU on G^s and M^s with linear layers (Eq. (8)). After $S = 3$ iteration steps, GMP produces the final node state $G^3 \in \mathbb{R}^{16,384 \times 64}$ according to Eq. (9), and broadcasts it to the bird's eye view. Next, we apply Region Proposal Network (RPN) [9] as the 2D backbone to further extract the features in the bird's eye view. RPN is composed of several convolutional blocks, with each defined as a tuple (S, Z, C) . S represents the stride of each block, while Z denotes the kernel size of each convolutional layer and C is the number of output channels. The output features of each block are resized to the same resolution via upsampling layers and concatenated together, so as to merge the semantic information from different levels.

In the long-term aggregation module, we implement the embedding functions Φ_K, Φ_Q, Φ_V and the output layer W_{out} in STA with 1×1 convolutional layers, and all the channel numbers are half of the inputs except W_{out} to save computations. In the TTA module, the regular convolutional layers, the deformable convolutional layers, and the ConvGRU all have learnable kernels of size 3×3 . All these convolutional kernels use the same channel numbers as the input bird's eye view features. We follow the PointPillars [3] baseline to set the anchor-based detection head and calculate anchors for different classes using the mean sizes. Two separate convolutional layers are used for classification and regression, respectively. For the anchor-free head, we refer to the CenterPoint baseline [5] by applying a center heatmap head and an attribute regression head. The former head aims to predict the center location of objects, while the latter head estimates a sub-voxel center location, as well as the box height, size, rotation, and velocity.

Training and Inference. We train our point cloud-based video object detector following the pre-training and fine-tuning paradigm. Specifically, the short-term encoding module is first trained in the same way as a single-frame detector, where one-cycle learning rate policy is used for 20

epochs with a maximum learning rate of 0.003 for PointPillars and 0.001 for CenterPoint. Then, we fine-tune the whole video detector including the long-term aggregation module with a fixed learning rate (0.003 for PointPillars or 0.001 for CenterPoint) for 10 epochs. Corresponding anchor-based or anchor-free loss functions are utilized in each of the long-term keyframes. Adam optimizer [75] is used to optimize the loss functions in both stages. In our framework, we feed at most 3 keyframes (with 30 LiDAR frames) to the model due to memory limitations. At the inference time, we preserve at most 500 detections after Non-Maxima Suppression (NMS) with a score threshold of 0.1. We test our algorithm on a Tesla v100 GPU and observe an average speed of 5 FPS and 1 FPS for PointPillars and VoxelNet backbones, respectively.

4.3 Quantitative and Qualitative Performance

We validate the performance of our 3D video object detection algorithm on the test set of the nuScenes benchmark by comparing it with other state-of-the-art approaches on the leaderboard. As shown in Table 1, our best model with CenterPoint-VoxelNet baseline outperforms all the published methods in terms of nuScenes detection score (NDS), which is the most important metric in nuScenes. Furthermore, we achieve first on the leaderboard at time of paper submission, without any tricks like leveraging information from 2D images.

Specifically, our anchor-based video detection model (PointPillars-VID) improves the PointPillars [3] baseline by nearly 8% NDS and 15% mAP, respectively. This demonstrates the significance of integrating temporal information in 3D point clouds. PointPillars is lightweight compared with other state-of-the-art methods. To further validate the performance of our algorithm, we implement it based on the stronger CenterPoint [5] baseline with VoxelNet backbone. CenterPoint proposes to represent objects as points with an anchor-free detection head and addresses the class imbalance problem inspired by CBGS [71], thus obtaining much better results. The newly released version of CenterPoint includes a two-stage refinement pipeline, while we only use the one-stage version as the baseline. By adopting the video detection strategy, our model (CenterPoint-VID) with offline mode obtains better performance, achieving 71.4% NDS and 65.4 mAP on the leaderboard. This significantly improves the strong baseline by 4.1% NDS and 5.1% mAP. By further integrating the painting strategy proposed by PointPainting [67], our final model (CenterPoint-VID*) gives the best performance on the leaderboard, achieving 71.8% NDS and 67.4 mAP, without any sophisticated ensemble strategies.

In addition to the quantitative results in Table 1, we further provide some qualitative examples. We mainly compare our 3D video object detector with the single-frame detector baseline, i.e., PointPillars [3]. We show specific cases that our model outperforms the single-frame detector, e.g., objects are occluded in certain frames or there are distant or small objects with sparse points. Here, three consecutive frames are shown for each case. The occlusion case is presented in Fig. 1, where our video object detector can handle the occluded objects with the memory features in

TABLE 1
Quantitative Detection Results on the nuScenes 3D Object Detection Benchmark

Method	Publication	Input	NDS	mAP	Car	Truck	Bus	Trailer	CV	Ped	Motor	Bicycle	TC	Barrier
InfoFocus [64]	ECCV 2020	0.5	39.5	39.5	77.9	31.4	44.8	37.3	10.7	63.4	29.0	6.1	46.5	47.8
PointPillars [3]	CVPR 2019	0.5	45.3	30.5	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
WYSIWYG [4]	CVPR 2020	0.5	41.9	35.0	79.1	30.4	46.6	40.1	7.1	65.0	18.2	0.1	28.8	34.7
SARPNET [65]	N.C. 2020	0.5	48.4	32.4	59.9	18.7	19.4	18.0	11.6	69.4	29.8	14.2	44.6	38.3
3DSSD [66]	CVPR 2020	0.5	56.4	42.6	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
PointPainting [67]	CVPR 2020	0.5	58.1	46.4	77.9	35.8	36.2	37.3	15.8	73.3	41.5	24.1	62.4	60.2
ReconfigPP [68]	ARXIV 2020	0.5	59.0	48.5	81.4	38.9	43.0	47.0	15.3	72.4	44.9	22.6	58.3	61.4
PointPillars_DSA [69]	ARXIV 2020	0.5	59.2	47.0	81.2	43.8	57.2	47.8	11.3	73.3	32.1	7.9	60.6	55.3
SSN V2 [70]	ARXIV 2020	0.5	61.6	50.6	82.4	41.8	46.1	48.0	17.5	75.6	48.9	24.6	60.1	61.2
3DCVF [30]	ECCV 2020	0.5	62.3	52.7	83.0	45.0	48.8	49.6	15.9	74.2	51.2	30.4	62.9	65.9
CBGS [71]	ARXIV 2019	0.5	63.3	52.8	81.1	48.5	54.9	42.9	10.5	80.1	51.5	22.3	70.9	65.7
CVCNet [72]	NeurIPS 2020	0.5	64.2	55.8	82.7	46.1	45.8	46.7	20.7	81.0	61.3	34.3	69.7	69.9
HotSpotNet [52]	ECCV 2020	0.5	66.0	59.3	83.1	50.9	56.4	53.3	23.0	81.3	63.5	36.6	73.0	71.6
CyliNet [73]	ARXIV 2020	0.5	66.1	58.5	85.0	50.2	56.9	52.6	19.1	84.3	58.6	29.8	79.1	69.0
CenterPoint [5]	CVPR 2021	0.5	67.3	60.3	85.2	53.5	63.6	56.0	20.0	84.6	59.5	30.7	78.4	71.1
PointPillars-VID (Ours)	CVPR 2020	1.5	53.1	45.4	79.7	33.6	47.1	43.1	18.1	76.5	40.7	7.9	58.8	48.8
CenterPoint-VID (Ours)	-	1.5	71.4	65.4	87.5	56.9	63.5	60.2	32.1	82.1	74.6	45.9	78.8	69.3
CenterPoint-VID* (Ours)	-	1.5	71.8	67.4	87.0	58.0	67.1	60.2	31.0	88.2	76.5	51.2	85.2	69.7

CV is construction vehicle, Motor is motorcycle, and TC is traffic cone. Our 3D video object detector significantly improves the single-frame detectors, and outperforms all the competitors on the leaderboard.

previous frames. In Fig. 5a, we showcase the detection of the distant car (the car on the top right), whose point clouds are especially sparse. Though it is very challenging for the single-frame detectors, our 3D video object detector still improves the detection results thanks to the information from adjacent frames. Similar improvement is observed in Fig. 5b, where a single-frame detector fails to detect the small objects in the right of the ego-car, while our model accurately recognizes these small objects. In a nutshell, compared with the single-frame detector, much fewer false positive (FP) and false negative (FN) detection results are obtained from our video object detector.

4.4 Ablation Studies

In this section, we conduct ablation studies to verify each module of our algorithm. In particular, we choose CenterPoint [5] with PointPillar backbone as the baseline, as it is more flexible and faster to develop than VoxelNet. To verify the effect of each module, experiments are conducted with the full training set, and are evaluated on the validation set. For tuning the hyperparameters in GMPNet and AST-GRU, experiments are performed on a $7\times$ downsampled training set. The short-term module uses point cloud data within 0.5 seconds, while the long-term module takes data in 1.5 seconds. All the modules work in online mode unless explicitly specified. In addition, we also perform experiments to analyze the influence of input data length in the short-term and long-term modules.

Comparison With Other Temporal Encoders. Our algorithm benefits from both the short-term and long-term point cloud sequence information, since GMPNet and AST-GRU modules are devised to handle these two different temporal patterns, respectively. Here, we compare our modules with other temporal-based design strategies. In particular, for both the short-term and long-term modules, the concatenation-based approach is set as the baseline by merging point clouds from 0.5 seconds or 1.5 seconds. This can be viewed

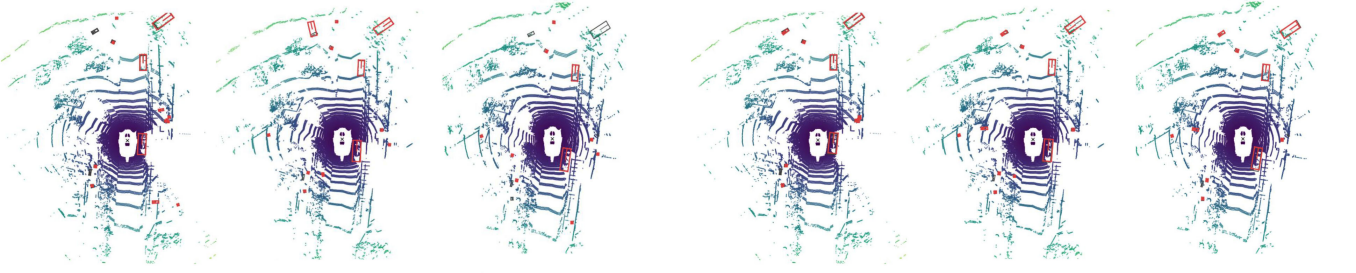
as the simplest temporal encoder. As shown in Table 2, our GMPNet improves the baseline by 1.53% mAP. This demonstrates that GMPNet could effectively mine the motion features in short-term point clouds by exchanging messages among grids from nearby frames, while the baseline encoder, e.g., the Pillar Feature Network, only considers each grid independently. Furthermore, to verify the effectiveness of aggregating long-term point clouds, we compare our AST-GRU with 3D temporal ConvNet [6] and vanilla ConvGRU [11] methods. Since the 3D ConvNet could only enforce optimization on a single keyframe, it thus achieves worse results than the ConvGRU [11]. Though the ConvGRU can exploit multi-frame features, it ignores the influence of noisy backgrounds and the misalignment in spatial features. In contrast, our proposed AST-GRU module improves these two approaches by 5.12% mAP and 1.42% mAP respectively, according to Table 2.

We also implement a new tracking-based method named *Tracklet Fusion* that follows the ideas in [6], [7] to check whether explicit object tracking could help detection. *Tracklet Fusion* integrates the detections and tracklets through NMS,

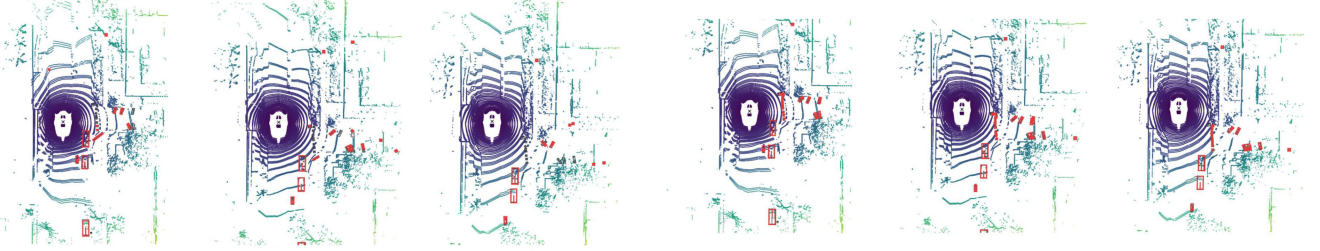
TABLE 2
Ablation Study of Our 3D Video Object Detector

Components	Modules	Performance mAP	Δ
Short-term Point Cloud Encoding	Concatenation [5]	48.25	-
	GMPNet	49.78	+1.53
Long-term Point Cloud Aggregation	Concatenation [5]	49.35	-
	Tracklet Fusion [3]	50.09	+0.74
	3D ConvNet [6]	52.84	+3.49
	ConvGRU [11]	56.54	+7.19
	STA-GRU	57.54	+8.19
	TTA-GRU	57.13	+7.78
	AST-GRU	57.96	+8.61
	Full Model (online)	58.78	+9.43
	Full Model (offline)	59.37	+10.02

CenterPoint-pillar with concatenated point clouds [5] is the reference baseline for computing the relative improvement (Δ).



(a) The single-frame detector (left) fail to detect the car on the top right in the third frame, while our method (right) could address this.

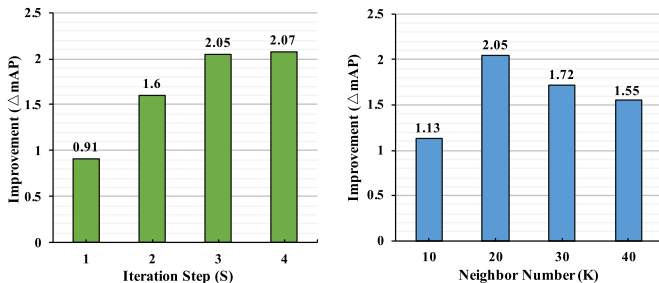


(b) The single-frame detector (left) have difficulty in detecting the small objects in the right of the ego-car (zoom in for better view).

Fig. 5. *Qualitative results of 3D video object detection.* We compare our algorithm (the right three frames in each case) with the single-frame 3D object detector (the left ones). The red and grey boxes indicate the predictions and ground-truths, respectively.

and improves the baseline by 0.74%. It demonstrates the importance of spatiotemporal features aggregation. It is worth mentioning that all the designs in AST-GRU give better performance. For example, the STA module enhances the ConvGRU by 1.00% mAP, while the TTA module further advances the performance by 0.59%. The overall model containing both GMPNet and AST-GRU surpasses the baseline by 9.43% mAP. Moreover, we obtain the best model by integrating the offline strategy, further improving the online model by 0.59%. This shows that the information from future frames can further boost detection performance. Next, we ablate some crucial designs in GMPNet and TTA, as well as the influence of the length of input point cloud sequences.

Hyperparameters in GMPNet. Our GMPNet enables a grid to capture a flexible receptive field via iteratively propagating messages on a k -NN graph. Given a grid-wise node, both the number of first-order neighbors (denoted as K) and the total iteration steps (denoted as S) have an influence on the receptive field as well as the final performance. In order to clearly demonstrate the impact of these two parameters, we show the performance change by varying S and K .



(a) Importance of S ($K=20$).

(b) Importance of K ($S=3$).

Fig. 6. *Ablation study for GMPNet.* We fix one parameter and vary the other to ablate the importance of iteration step S and neighbor node number K .

According to Fig. 6a, we observe that $S = 3$ has already obtained a sufficient receptive field, and further increasing S does not help improve the results. In Fig. 6b, we show that $K = 20$ achieves the best performance, while a larger K instead degrades performance. We infer that $K = 20$ and $S = 3$ have captured an appropriate receptive field, and a much larger receptive field may confuse the detector.

Different Design Strategies in TTA. Our TTA aligns the features of the dynamic objects by applying a modified deformable convolutional network. Thus, there are several factors affecting the final results, e.g., the inputs for computing supporting regions and the number of layers in TTA. In our implementation, we integrate a *motion map* into the inputs and use two layers for TTA. Here, we give a detailed analysis of these aspects. In Table 3, w/o *motion map* denotes that TTA takes only the previous memory feature H_{t-1} as input, while w/ current input represents that TTA receives the concatenation of H_{t-1} and X_t' . Both designs give

TABLE 3
Detailed Analysis of the Input Choices and the Layer Number in TTA Module

Aspect	Modules	Performance	
		mAP	Δ
Inputs ($M=2$)	Full Model	27.28	0
	w/o <i>motion map</i>	26.03	-1.25
	w/ current input	26.27	-1.01
Layer Number	$M=1$	26.55	-0.73
	$M=2$	27.28	0
	$M=3$	26.42	-0.86

The full model is viewed as the reference for computing the relative performance (Δ).

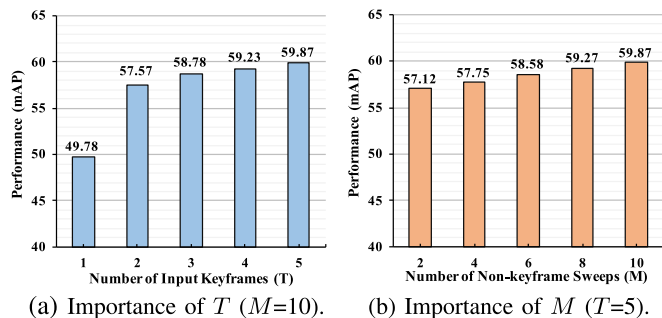


Fig. 7. Ablation study for the input length of the short-term and long-term modules. In (a) and (b), different keyframes T or non-keyframe sweeps M are used to evaluate the performance.

decreased performance, demonstrating the effectiveness of the *motion map*. In addition, the model using two modified deformable convolutional layers achieves the best performance. We infer that the deeper layers lead to difficulty in optimizing the whole network.

Input Length of Point Cloud Frames. Finally, we analyze the effect of the input sequence length. For datasets like nuScenes, labels are only available on keyframes, and non-keyframe sweeps do not have annotations. It is interesting to explore the short-term features in non-keyframe sweeps for further improving the performance. Thus, our GMPNet is developed to implicitly capture the temporal information in short-term non-keyframe sweeps, while applying AST-GRU to explicitly leverage the labels in long-term keyframes. We now evaluate the importance of the number of keyframes and non-keyframe sweeps. In Fig. 7a, we fix $M = 10$ and increase T . When $T = 1$, GMPNet is adopted as the baseline. When $T > 1$, AST-GRU is further used to capture long-term features. Our model with 3 keyframes already achieves good enough results, and a larger T brings slightly consistent improvement. In Fig. 7b, T is fixed as 5 and we increase M from 2 to 10. Obvious improvements can be seen with larger M . In particular, when the number of total frames $T \times M$ is the same, similar gains could be achieved. This indicates that short-term features are as crucial as long-term features in 3D video object detection. For balancing between the efficiency and accuracy, we adopt $T = 3$ and $M = 10$ in the final model with the VoxelNet backbone.

5 CONCLUSION

We have presented a new framework for 3D video object detection in point clouds. Our framework formulates the temporal information with short-term and long-term patterns, and devises a short-term encoding module and a long-term aggregation module to address these two temporal patterns. In the former module, a GMPNet is introduced to mine the short-term object motion cues in nearby point cloud frames. This is achieved by iterative message exchanging in a k -NN graph: a grid updates its feature by integrating information from k neighbor grids. In the latter module, an AST-GRU is proposed to further aggregate long-term features. AST-GRU includes a STA and a TTA, which are designed to handle small objects and align moving objects, respectively. Our point cloud video-based framework can work in either online or

offline modes, depending on the applications. We also tested our framework with both anchor-based and anchor-free 3D object detectors. Evaluation results on the nuScenes benchmark demonstrate the superior performance of our framework.

ACKNOWLEDGMENTS

A preliminary version of this work has appeared in CVPR 2020 [1].

REFERENCES

- [1] J. Yin, J. Shen, C. Guan, D. Zhou, and R. Yang, "LiDAR-based online 3D video object detection with graph-based message passing and spatiotemporal transformer attention," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11495–11504.
- [2] H. Caesar et al., "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11618–11628.
- [3] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12697–12705.
- [4] P. Hu, J. Ziegler, D. Held, and D. Ramanan, "What you see is what you get: Exploiting visibility for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10998–11006.
- [5] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3D object detection and tracking," 2020, *arXiv:2006.11275*.
- [6] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3569–3577.
- [7] M. Liang et al., "PnPNet: End-to-end perception and prediction with tracking in the loop," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11550–11559.
- [8] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6450–6459.
- [9] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 4490–4499, 2018.
- [10] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337.
- [11] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," in *Proc. Int. Conf. Learn. Representations*, 2016.
- [12] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
- [13] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7794–7803.
- [14] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets V2: More deformable, better results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9300–9308.
- [15] X. Zhu, D. Cheng, Z. Zhang, S. Lin, and J. Dai, "An empirical study of spatial attention mechanisms in deep networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6687–6696.
- [16] Y. Chen, L. Tai, K. Sun, and M. Li, "MonoPair: Monocular 3D object detection using pairwise spatial relationships," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12090–12099.
- [17] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6850–6860.
- [18] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3D object detection leveraging accurate proposals and shape reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11859–11868.
- [19] P. Li, X. Chen, and S. Shen, "Stereo R-CNN based 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7636–7644.

- [20] Y. Chen, S. Liu, X. Shen, and J. Jia, "DSGN: Deep stereo geometry network for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12536–12545.
- [21] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8437–8445.
- [22] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2147–2156.
- [23] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7652–7660.
- [24] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.
- [25] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1951–1960.
- [26] Q. Meng, W. Wang, T. Zhou, J. Shen, Y. Jia, and L. Van Gool, "Towards a weakly supervised framework for 3D point cloud object detection and annotation," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Mar. 3, 2021, doi: [10.1109/TPAMI.2021.3063611](https://doi.org/10.1109/TPAMI.2021.3063611).
- [27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
- [28] S. Shi *et al.*, "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10526–10535.
- [29] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6526–6534.
- [30] J. H. Yoo, Y. Kim, J. S. Kim, and J. W. Choi, "3D-CVF: Generating joint camera and lidar features using cross-view spatial feature fusion for 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 720–736.
- [31] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3075–3084.
- [32] R. Huang *et al.*, "An LSTM approach to temporal 3D object detection in LiDAR point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 266–282.
- [33] Y. Chen, Y. Cao, H. Hu, L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10334–10343.
- [34] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
- [35] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.
- [36] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014.
- [37] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.
- [38] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3837–3845.
- [39] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, 2011.
- [40] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.
- [41] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
- [42] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.*, 2018, pp. 593–607.
- [43] H. Gao and S. Ji, "Graph U-nets," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2083–2092.
- [44] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016.
- [45] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: Moving beyond fingerprints," *J. Comput.-Aided Mol. Des.*, vol. 30, no. 8, pp. 595–608, 2016.
- [46] V. Zayats and M. Ostendorf, "Conversation modeling on reddit using a graph-structured LSTM," *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 121–132, 2018.
- [47] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence N-ary relation extraction with graph LSTMs," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 101–115, 2017.
- [48] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.
- [49] W. Wang, X. Lu, J. Shen, D. J. Crandall, and L. Shao, "Zero-shot video object segmentation via attentive graph neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9235–9244.
- [50] S. Qi, W. Wang, B. Jia, J. Shen, and S.-C. Zhu, "Learning human-object interactions by graph parsing neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 407–423.
- [51] C. Si, Y. Jing, W. Wang, L. Wang, and T. Tan, "Skeleton-based action recognition with spatial reasoning and temporal stack learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 103–118.
- [52] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille, "Object as hot-spots: An anchor-free 3D object detection approach via firing of hotspots," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 68–84.
- [53] D. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.
- [54] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4502–4510.
- [55] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014.
- [56] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.
- [57] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 843–852.
- [58] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.
- [59] M. Liu and M. Zhu, "Mobile video object detection with temporally-aware feature maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5686–5695.
- [60] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7445–7454.
- [61] Z. Gan *et al.*, "Semantic compositional networks for visual captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1141–1150.
- [62] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.
- [63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [64] J. Wang, S. Lan, M. Gao, and L. S. Davis, "InfoFocus: 3D object detection for autonomous driving with dynamic information modeling," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 405–420.
- [65] Y. Ye, H. Chen, C. Zhang, X. Hao, and Z. Zhang, "Sarpnet: Shape attention regional proposal network for LiDAR-based 3D object detection," *Neurocomputing*, vol. 379, pp. 53–63, 2020.
- [66] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11037–11045.
- [67] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential fusion for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4603–4611.
- [68] T. Wang, X. Zhu, and D. Lin, "Reconfigurable voxels: A new representation for lidar-based point clouds," 2020, *arXiv:2004.02724*.
- [69] P. Bhattacharyya, C. Huang, and K. Czarnecki, "Self-attention based context-aware 3D object detection," 2021, *arXiv:2101.02672*.
- [70] X. Zhu, Y. Ma, T. Wang, Y. Xu, J. Shi, and D. Lin, "SSN: Shape signature networks for multi-class object detection from point clouds," 2020, *arXiv:2004.02774*.

- [71] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3D object detection," 2019, *arXiv:1908.09492*.
- [72] Q. Chen, L. Sun, E. Cheung, and A. L. Yuille, "Every view counts: Cross-view consistency in 3D object detection with hybrid-cylindrical-spherical voxelization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.
- [73] M. Rapoport-Lavie and D. Raviv, "It's all around you: Range-guided cylindrical network for 3D object detection," 2020, *arXiv:2012.03121*.
- [74] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis Pattern Recognit.*, 2012, pp. 3354–3361.
- [75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.



Junbo Yin is currently working toward the PhD degree with the School of Computer Science, Beijing Institute of Technology, Beijing, China. His research interests include Lidar-based 3D object detection and segmentation, self supervised learning, and visual object tracking.



Jianbing Shen (Senior Member, IEEE) is currently a full professor with the State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), Department of Computer and Information Science, University of Macau, and also the head of Centre for Artificial Intelligence and Robotics, University of Macau. Before that, he acted as the Lead Scientist with the Inception Institute of Artificial Intelligence, Abu Dhabi, United Arab Emirates, and the chief scientist with the Research Institute of Autonomous Driving at NavInfo Technology. He published more

than 200 top journal and conference papers, and his Google scholar citations are about 21200 times with H-index 73. He was rewarded as the Highly Cited Researcher by the Web of Science in 2020–2022, and also the most cited Chinese researchers by the Elsevier Scopus in 2020–2022. His research interests include computer vision, self-driving cars, deep learning, smart city, and intelligent systems. He is/was an associate editor of *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems*, *Pattern Recognition*, and other journals.



Xin Gao received the PhD degree in computer science from University of Waterloo, Waterloo, ON, Canada, in 2009. He is currently a full professor of computer science with Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia, the associate director of the Computational Bioscience Research Center, KAUST, and an adjunct faculty member with the David R. Cheriton School of Computer Science, University of Water-

loo. He has coauthored more than 200 research articles in the fields of machine learning and bioinformatics. His research interests include building computational models, developing machine learning methods, and designing efficient and effective algorithms, with particular a focus on applications to key open problems in biology.



David J. Crandall received the BS and MS degrees in computer science and engineering from Pennsylvania State University, State College in 2001 and the PhD degree in computer science from Cornell University, in 2008. He is currently an associate professor with the School of Informatics and Computing, Indiana University. His research interests include computer vision, machine learning, and data mining. He is currently an associate editor for the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and the *IEEE Transactions on Multimedia*. He was the recipient of National Science Foundation CAREER Award and a Google Faculty Research Award. .



Ruigang Yang received the MS degree from Columbia University in 1998 and the PhD degree from the University of North Carolina, Chapel Hill, in 2003. He is currently a full professor of computer science with the University of Kentucky. His research interests include computer vision and computer graphics, in particular in 3D reconstruction and 3D data analysis. He has authored or coauthored more than 100 papers, which, according to Google Scholar, has received close to 16,800 citations with an h-index of 61. He was the area chair for premium vision conferences, including ICCV or CVPR, and was a program chair for CVPR 2021. He is currently an associate editor for the *IEEE Transactions on Pattern Analysis and Machine Intelligence*. He was the recipient of the number of awards, including the U.S. National Science Foundation Faculty Early Career Development (CAREER) Program Award in 2004, Best Demonstration Award at CVPR 2007, and the Deans Research Award at the University of Kentucky in 2013.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.