Knowledge-Based Systems 259 (2023) 110085

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# Dynamic graph transformer for 3D object detection

Siyuan Ren [a], Xiao Pan [b], Wenjie Zhao [a,*], Binling Nie [c], Bo Han [a]

[a] School of Aeronautics and Astronautics, Zhejiang University, Hangzhou, 310000, Zhejiang, China
[b] College of Computer Science and Technology, Zhejiang University, Hangzhou, 310000, Zhejiang, China
[c] Hangzhou Dianzi University, Hangzhou, 310000, Zhejiang, China

ARTICLE INFO

*Article history:*
Received 19 July 2022
Received in revised form 22 October 2022
Accepted 29 October 2022
Available online 4 November 2022

*Keywords:*
3D object detection
Point cloud
Transformer
Graph structure learning
Automatic driving

ABSTRACT

LiDAR-based 3D detection is critical in autonomous driving perception systems. However, point-based 3D object detection that directly learns from point clouds is challenging owing to the sparsity and irregularity of LiDAR point clouds. Existing point-based methods are limited by fixed local relationships and the sparsity of distant and occluded objects. To address these issues, we propose a dynamic graph transformer 3D object detection network (DGT-Det3D) based on a dynamic graph transformer (DGT) module and a proposal-aware fusion (PAF) module. The DGT module is built on a dynamic graph and graph-aware self-attention module, which adaptively concentrates on the foreground points and encodes the graph to capture long-range dependencies. With the DGT module, DGT-Det3D has better capability to detect distant and occluded objects. To further refine the proposals, our PAF module fully integrates the proposal-aware spatial information and combines it with the point-wise semantic features from the first stage. Extensive experiments on the KITTI dataset demonstrate that our approach achieves state-of-the-art accuracy for point-based methods. In addition, DGT brings significant improvements when combined with state-of-the-art methods on the Waymo open dataset.


## 1. Introduction

Accurate 3D object detection is a core component of 3D perception tasks in many real-world applications, such as robotics, autonomous driving, etc. Point clouds are the most commonly used data representations for 3D object detection. Although the rapid development of convolutional neural networks (CNNs) have promoted 2D object detection performance [1–4] significantly, the irregular data format and sparse property of point clouds make it challenging to directly extend 2D detection paradigms to 3D object detection.

Recently, many studies have been conducted to solve the problem of 3D object detection on point clouds, which can be mainly divided into *voxel-based* and *point-based*. *Voxel-based* methods convert irregular point clouds to regular representations, such as BEV maps [5–7] or 3D voxels [8,9], and subsequently extract features by using 2D/3D convolutional networks for detection, which have been widely investigated and have achieved great success. However, voxel-based methods introduce quantization errors [10–14] during voxelization and BEV projection, which causes spatial misalignment of the voxel and therefore leads to unsatisfactory detection capability toward distant and occluded objects. By contrast, *point-based* methods [14–18] can extract fine-grained spatial information from point clouds for accurate localization, which is important for distant and occluded objects in large input spaces. Moreover, many *voxel-based* and *point-voxel-based* methods [10–13,19] benefit from *point-based* methods that introduce fine-grained point cloud spatial information via auxiliary tasks or two stages. This paper focuses on point-based methods that aim to facilitate the development of 3D object detection.

It is noted that the existing point-based methods still suffer from the following limitations. (i) *Fixed local relationships*. Although PointNet-like architectures [15,16] can capture certain local structures, the fixed relationships during training cause foreground points on the object edge to establish a misleading relationship with the background points [20], thereby affecting the feature quality of the foreground points. (ii) *Sparsity of distant and occluded objects*. Owing to the sparsity of LiDAR point clouds, the number of obtained point clouds for distant and occluded objects is limited, making such objects difficult to detect. (iii) *Accurate point clouds are not fully utilized*. Notably, PointRCNN [17] demonstrates that an extra refinement module is crucial for better performance. However, it does not fully exploit the geometric information of proposals with raw points and semantic features from the first stage.

* Corresponding author.
*E-mail addresses:* ren_siyuan@zju.edu.cn (S. Ren), zhaowenjie8@zju.edu.cn (W. Zhao).

https://doi.org/10.1016/j.knosys.2022.110085
0950-7051/

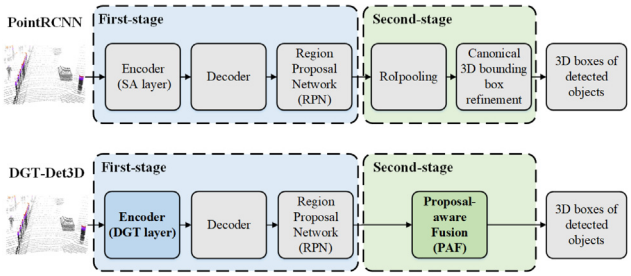


**Fig. 1.** Comparison with the baseline method [17].

To address these issues, we propose a novel two-stage dynamic graph transformer 3D object detection network (DGT-Det3D). Fig. 1 illustrates the comparison with the baseline method [17]. The main contributions of this study are as follows:

In the first stage, we propose a plug-and-play dynamic graph transformer (DGT) module composed of *dynamic graph update (DGU)*, *graph-aware self-attention (GSA)*, and *feed-forward network (FFN)*. (i) *Dynamic graph update.* First, to improve the local relationships and enable the learning process to focus on the more valuable foreground data, we build a dynamic graph [20] for each point to learn connections adaptively, which is different from the commonly used fixed graph convolution network [21] and ball query in PointNet++ [16]. (ii) *Graph-aware self-attention.* Next, a GSA module is proposed to completely capture the long-range dependencies [22,23] between points, improving the robustness toward distant and occluded objects with sparse points. (iii) *Feed-forward network.* Finally, an FFN with a residual operator is employed to aggregate point-wise semantic features to generate high-quality proposals.

In the second stage, to fully exploit the information of accurate raw points and high-quality proposals, which has been ignored in previous methods [17,19], we propose the proposal-aware fusion (PAF) module that can fuse the position information of points with the geometric information of proposals and better integrate spatial and semantic features.

Experimental results on KITTI and Waymo open datasets show that our proposed plug-and-play DGT can be easily integrated with other methods, and it outperforms the commonly used set abstraction (SA) [16] module in other state-of-the-art (SOTA) point-based methods by significant margins. When integrating the DGT with the proposed PAF, the proposed DGT-Det3D achieves better performance than other SOTA point-based methods on the KITTI dataset.

Our contributions are summarized as follows:

- We propose a plug-and-play DGT module, which can enrich local spatial and long-range context features to boost the capability of a point-based 3D backbone, especially for distant and occluded objects.
- We propose a PAF module to completely encode the accurate position information of raw points and the natural geometric property of proposals. Furthermore, spatial and semantic features are combined to refine classification and 3D regression.
- We validate the effectiveness of the plug-and-play DGT on KITTI and Waymo open datasets. When DGT is integrated with the proposed PAF, the proposed DGT-Det3D achieves SOTA performance among point-based methods on the KITTI dataset.

## 2. Related work

### 2.1. 3D object detection from point clouds

**Voxel-based method.** Voxel-based detectors transform unordered point clouds into regular data representations. VoxelNet [24] is a pioneering end-to-end network that densely voxelizes point clouds and proposes a voxel feature encoding layer to extract features. SECOND [8] introduces 3D sparse convolution for processing voxel efficiency. PointPillars [7] encode points as pseudo-images and rapidly use standard 2D convolution processing. CIA-SSD [25] uses a lightweight BEV network combined with IoU-aware confidence rectification and DI-NMS. Voxel-RCNN [9] uses a voxel RoI pooling module to extract RoI features directly from voxel features. CenterPoint [26] proposes an anchor-free 3D object detection head.

Although these voxel-based methods can achieve good detection performance with promising efficiency, they introduce quantitative information loss, which affects the detection performance for long-distance and small objects. Furthermore, they suffer from a trade-off between efficiency and precision depending on the size of the voxel.

**Point-based method.** Point-based detectors directly process raw point clouds to generate 3D boxes that retain precise position information. The proposed PointNet [15], PointNet++ [16] allows point features to be directly learned from raw point clouds. F-PointNet [27] and F-Conv [28] aggregate point-wise features from frustums. PointRCNN [17] generates 3D proposals based on the PointNet++ [16] backbone and utilizes point cloud RoI pooling to extract the 3D region features of each proposal for bounding box refinement. STD [29] further extends the proposal refinement by transferring sparse point features into a dense voxel representation. 3DSSD [18] proposes a single-stage anchor-free 3D object detector, which introduces a novel sampling strategy for feature and spatial distance fusion. The IA-SSD [14] identifies the sampling issue in existing point-based detectors and proposes an efficient point-based 3D detector by introducing two learning-based instance-aware downsampling strategies.

Point-based approaches avoid voxelization induced quantization errors by learning features directly from the raw point sets. However, the insufficient learning capacity and inefficient time consumption limit the detection performance of point-based methods.

**Point-Voxel-based method.** Several point-voxel hybrid methods have been proposed to solve the problems associated with the voxel-based and point-based methods. PV-RCNN [19] is a SECOND [8] extension that uses raw point position information and voxel representations to provide spatial context. SA-SSD [13] uses an auxiliary network parallel to the backbone to regress box centers and semantic classes to augment features. HVPR [30] introduces a memory module to boost point-based features and increase efficiency. However, these hybrid approaches require more hand-crafted feature designs. In addition, it is challenging to achieve a balance between accuracy and computational efficiency.

### 2.2. Graph convolutional networks in 3D object detection

Recently, certain methods introduce graph convolutional networks (GCNs) [31–33] for learning point clouds. The graph contains accurate position information of raw point clouds. It constructs topological information among points, which can obtain richer features through message passing between vertices and edges. DGCNN [20] proposes EdgeConv to learn local point cloud features in a dynamic KNN graph. EPFM-Conv [34] integrates a graph-based method and point-based strategy to extract rich

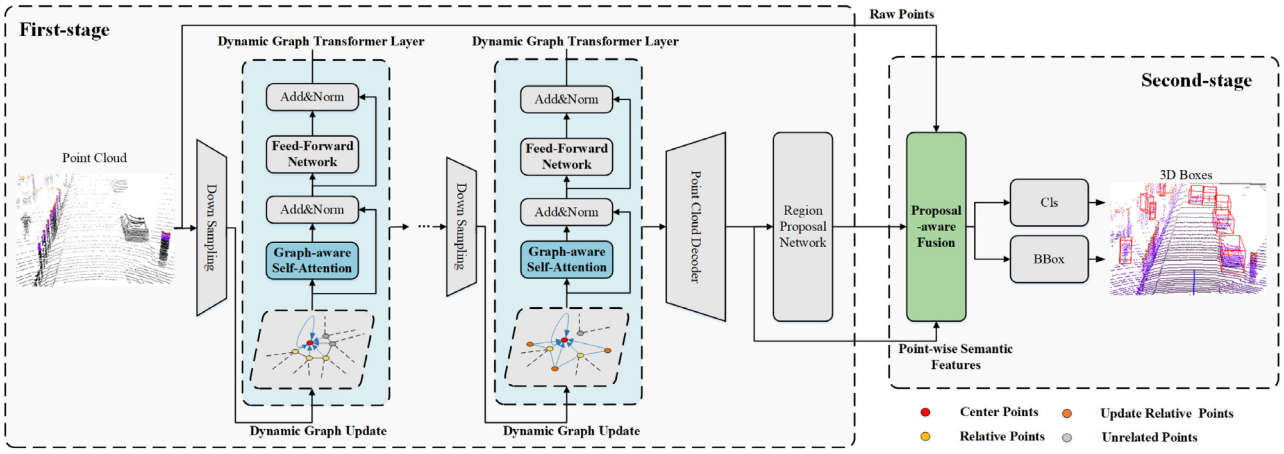


**Fig. 2.** The architecture of our proposed DGT-Det3D. The whole network consists of two major stages: In the first stage, we build a dynamic graph for the raw point clouds and use the 3D DGT module to capture long-range context and learn spatial information at each encoder layer. Then, decoder layers produce point-wise semantic features for RPN generating 3D proposals. In the second stage, the PAF module fuses fully encoded spatial and semantic features to refine the 3D proposals for final prediction.

point cloud features. PointGCN [35] leverages GCNs to refine proposals. Point-GNN [21] designs a one-stage graph neural network with an auto-registration mechanism and an NMS with box merging and scoring. It can be observed that GCNs have great potential for processing irregular and sparse point clouds.

### 2.3. Transformer in 3D object detection

Recently, machine translation and natural language processing [36,37] have achieved considerable much success using transformer [22] models and self-attention mechanisms [36]. Furthermore, they significantly contribute to the field of computer vision [38–40]. Considering the properties of point clouds, a transformer with positional encoding is particularly suitable for 3D point clouds, which can encode positional information, long-range context, and permutation invariance. PT [41], PCT [42], 3DERT [43], PAT [44], and CSANet [45] introduce self-attention or transformers to build point representations for end-to-end point-cloud classification and segmentation. TA-Net [46] is a 3D object detector that leverages shared transformation functions across all voxels to aggregate the context from a local point-voxel neighborhood using standard attention. CT3D [12] performs proposal-aware embedding and uses a channel-wise transformer for the point features to capture the long-range context within each proposal. SA-Det3D [47] explores two self-attention variants to augment the context-aware features. VoTr [23] proposes a sparse voxel transformer module and submanifold voxel transformer module. Unlike the aforementioned methods, the proposed DGT module combines the advantages of a dynamic graph and self-attention to fully encode local features and improve the detection ability of distant and occluded objects.

## 3. Method

### 3.1. Overview

We propose a novel two-stage dynamic graph transformer 3D object detection framework (DGT-Det3D) as illustrated by Fig. 2. In the first stage, we introduce a plug-and-play DGT encoder module to process the dynamic graph in the feature space in a hierarchical manner. Based on the encoder–decoder structure, multiple DGT layers serve as the 3D backbone to obtain high-quality semantic features. The proposed DGT consists of three parts: (i) A dynamic graph update (DGU) module is built to connect each point with K-nearest neighbors in the feature space. (ii) A graph-aware self-attention (GSA) module is introduced to aggregate the edge features associated with all connected vertices and capture long-range context features. In addition, a dual-branch relative positional encoding (DRPE) module is proposed to capture precise position and geometric information. (iii) A feed-forward network (FFN) is adopted to learn the global features. For decoder processing, we adopt feature propagation (FP) layers, such as [16] to learn point-wise semantic features. Finally, a region proposal network (RPN) generates high-quality segmentation masks, classification scores, and 3D proposals for the second refinement stage.

In the second stage, to further exploit the advantages of point-based methods with accurate point cloud position information, a proposal-aware fusion (PAF) module is proposed. PAF exploits the geometric information of the proposal with raw points and fuses it with one-stage semantic features to refine the proposals. Finally, we adopt a detection head for 3D object prediction.

### 3.2. Dynamic graph transformer

This section introduces the details of the proposed DGT module. As shown in Fig. 2, each DGT layer mainly consists of a dynamic graph update, graph-aware self-attention, and feed-forward neural network.

**Dynamic Graph Update.** For the 3D object detection task, a few foreground points are required for high performance. Most current point-based methods employ an SA module [16] for point downsampling and feature grouping. However, as illustrated in the right side of Fig. 3, the SA module only aggregates the features from the fixed ball query area without distinguishing the local relationships between the foreground and background points. Such fixed aggregation leads to the entanglement between foreground and background point features. Therefore, the SA module performs sub-optimally at the following classification and regression stages.

To address this issue, we introduce a dynamic graph update (DGU) module to enrich the representation power of foreground point clouds, as shown on the left of Fig. 3. First, we adopt a hierarchical downsampling strategy [16] to expand the receptive field and reduce the quadratic complexity of the GSA. In addition, a fusion sampling strategy [18] comprising the farthest point sampling on feature and Euclidean space is employed in the last two DGT layers to preserve more foreground points.

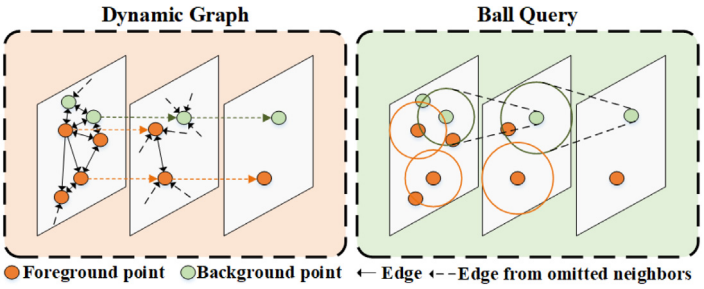


**Fig. 3.** Comparison between our proposed dynamic graph update (DGU) module and the ball query strategy. The ball query considers fixed local relationships and leads to the entanglement between foreground and background features. However, our DGU module decouples foreground and background features and concentrates more on foreground features via a dynamic k-NN graph.

Next, to update local relationships and focus on foreground points, we propose a dynamic graph. Formally, the $n$ points are denoted by $P = \{p_1, p_2, \ldots, p_n\}$, $p_i \in \mathbb{R}^C$, where each point coordinate and the laser reflection intensity in 3D space are defined as $p_i = [x_i, y_i, z_i, r_i]$. We construct a directed k-nearest neighbor (k-NN) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of $P$ to represent the local point cloud structure. First, we adopt the features corresponding to $P$ to initialize vertices referred to $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, $v_i \in \mathbb{R}^F$. Then, the edges $\mathcal{E} = \left\{e_{ij} : (i,j) \in \mathcal{V} * \mathcal{V}\right\}$ are computed by the k-nearest neighbor vertices, including the self-loop in the feature space, to make connections between similar foreground points. At each layer $l$, we have a different graph $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)})$ on the sampled $P^{(l)}$, and we encode the edge and vertex features of the graph as follows:

$$\begin{aligned} \mathcal{G}^{(l)} &= kNN(P^{(l)}), \\ \mathcal{E}^{(l)} &= \left\{e_{ij} = h(v_i, v_j)\right\}, \\ \mathcal{V}^{(l)\prime} &= \left\{v_i{}' = \sum e_{ij}\right\}, \end{aligned} \tag{1}$$

where $v_i$ is the central vertex of the local graph, and $v_j$ represents the neighbors around $v_i$. $e_{ij}$ is the edge feature learned from vertices $v_i$ and $v_j$ using EdgeConv [20]. Using EdgeConv, we can simultaneously encode local spatial and global semantic information. Then, we aggregate edge features $e_{ij}$ by a sum or maxpooling operation to update vertex features $v_i'$.

The k-nearest neighbor graph is recomputed to update the relations adaptively at each layer. The same number of graph neighbor connections is also maintained for sampling layers with different resolution. This design can obtain a gradually larger receptive field for the local graph. With expansion of the receptive field, DGT can dynamically encode rich long-range contexts. Hence, it is referred to "Dynamic Graph Transformer".

**Graph-aware Self-attention.** Graph-aware self-attention is the core of the DGT module, which is a variant of vector self-attention [48] and targets fully encoding the relationships in a graph. Compared to the commonly used PointNet [15,16] in the SA module, the following three improvements can be observed:

(i) *Subtraction self-attention.* Although the commonly used PointNet captures point-to-point relations in the local region, it has a weak capability to encode features of distant and occluded objects. Thus, we introduce a subtraction vector self-attention to capture long-range dependencies and better explore relations in a graph for detecting distant and occluded objects, as shown in Fig. 4. Unlike scalar dot-product self-attention, our subtraction vector self-attention method can better learn channel-wise attention weights between the query and key embedding in the graph. Additionally, the shared dimensionality between attention weights and dual-branch relative positional encoding features makes it natural to introduce precise position information into the attention weight computing process. The subtraction structure works in concert with the dual-branch relative positional encoding module, which will be introduced in detail.

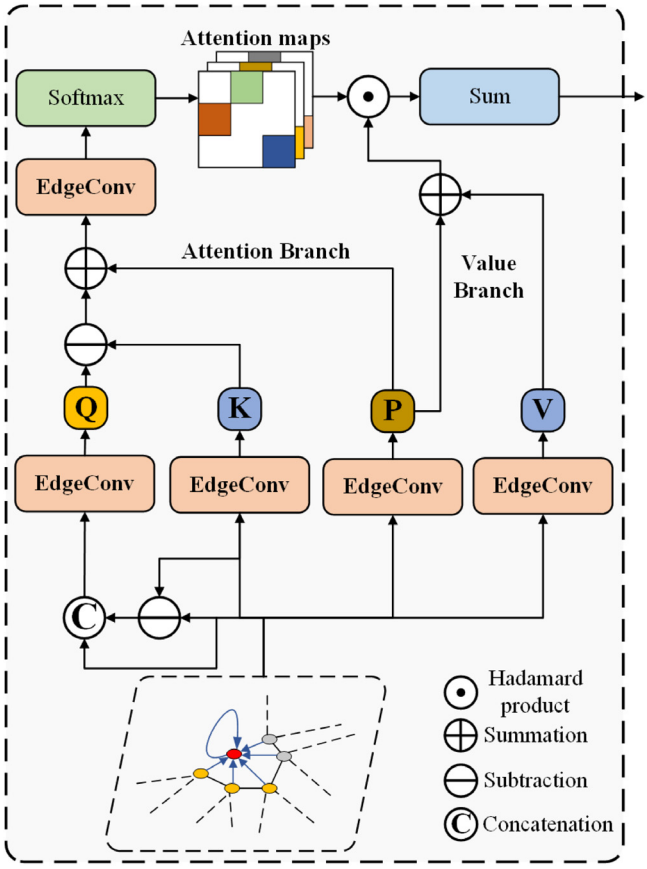


**Fig. 4.** Illustration of the graph-aware self-attention module. The GSA module fully encodes the graph features by EdgeConv and integrates dual-branch relative positional encoding branches to introduce precise spatial information continuously.

(ii) *EdgeConv.* PointNet [15] only encodes global features, whereas PointNet++ [16] is further proposed to learn a fixed local structure. These methods are special cases of EdgeConv [20]. To fully encode local and global graph features simultaneously, we introduce EdgeConv $h(v_i, v_j)$ to replace MLPs in subtraction self-attention, as shown in Fig. 4. Specifically, we compute the query embedding, key embedding, and value embedding as follows:

$$Q = h(v_j - v_i, v_i), K = h(v_j), V = h(v_j), \tag{2}$$

where $Q, K, V$ are the learnable edge transformations for the query, key, and value features, respectively. In particular, to completely learn the local spatial and global semantic information of the graph, we employ the edge function $h(v_j - v_i, v_i) = ReLU(Conv(v_j - v_i) + Conv(v_i))$ for query embeddings. In addition,

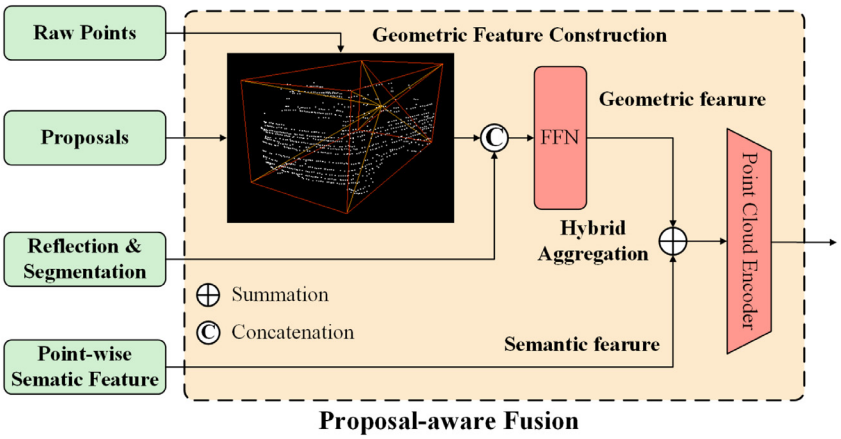


**Fig. 5.** Illustration of proposal-aware fusion module. The PAF module integrates positional and geometric information through the relative position relationship between the raw point cloud and the associated proposal (center and corner points). Then, the spatial representation concatenates with the segmentation results of each point in proposals and a feed-forward network is used to produce spatial features. Next, the spatial and semantic features are fused by a feed-forward network. Finally, multi-DGT layers refine the fused features to get the final features.

we consider the edge function $h(v_j) = ReLU(Conv(v_j))$ to encode key and value embeddings for learning local neighborhood information.

(iii) *Dual-branch relative positional encoding*. Accurate spatial information critically influences on the performance of point cloud detectors. There are many options for encoding position information, including the learned [49] and fixed [22] methods. To provide a more spatial context to the GSA module, we propose dual-branch relative positional encoding (DRPE), which consists of the attention computing branch and value branch. As mentioned in (i) *subtraction self-attention*, the subtraction structure is conducive to directly adding DRPE to the attention and value features. Specifically, DRPE utilizes EdgeConv to learn the relative position features from the related graph vertices, which can be formulated as

$$DRPE(v_i, v_j) = \text{EdgeConv}(v_i - v_j). \tag{3}$$

As shown in Fig. 4, the output of DRPE shares the same dimensions as the Q, K, and V embeddings. Therefore, we send the relative positional encoding features to the attention computing branch together with the value branch. The experiments in indicate that the DRPE module plays a critical role in the GSA module.

Specifically, the core of the GSA can be formulated as

$$\begin{aligned} y_i = Sum(Norm(v_i + \sum_j softmax(h(Q - K + DRPE(v_i, v_j))) \\ \odot \ (V + DRPE(v_i, v_j)))), \end{aligned} \tag{4}$$

where $y_i$ is the output feature of the GSA. $DRPE(\cdot)$ denotes a DRPE module. $Sum(\cdot)$ is a sum operation for aggregating local graph features into a central vertex.

**Feed-forward Network.** Finally, a conventional FFN and residual operator are used to aggregate point-wise semantic features with global dependencies. The FFN is formulated as

$$z_i = Norm(y_i + FFN(y_i)), \tag{5}$$

where $z_i$ denotes the output feature of the DGT. $FFN(\cdot)$ is the FFN that consists of two linear layers with ReLU activation. *Norm* denotes the BatchNorm function.

### 3.3. Proposal-aware fusion module

In the second stage, we further refine the 3D bounding box proposals obtained from the RPN. For each proposal, the point cloud RoI pooling operation [17] pools 3D points and their associated features to obtain RoI features directly. However, they do not fully utilize the *geometric information* of the proposal. The proposal-to-point encoding in [12] is modulated by the proposal information. However, it mainly serves voxel-based methods and ignores the *semantic features* acquired in the first stage. Furthermore, some methods [50–52] have been proposed to fuse multi-views data. For improving the localization accuracy, we propose a proposal-aware fusion (PAF) module to exploit fine-grained spatial and semantic information by fusing *geometric features* $f_g$ and *semantic features* $f_s$. PAF consists mainly of geometric feature construction (GFC) and hybrid aggregation (HA). GFC uses a small manual design to encode the points in the proposal to obtain geometric information. Then, HA uses a feed-forward network to fuse geometric features with the foreground point semantic features obtained in the first stage. Finally, HA adopts multiple DGT layers to produce features for prediction, which significantly enhances the refinement module.

**Geometric Feature Construction.** The most important advantage of point-based approaches is that they provide precise point cloud location information. To take full advantage of the precise point cloud location information, we further explore the geometric information of the proposals with raw points. First, we randomly sample N = 512 keypoints $P_k = \{p_1, \ldots, p_N\}$ within each proposal. As shown in Fig. 5, to encode the proposal shape information, we calculate the relative coordinates between the center of the proposal $p^c$ and each keypoint $p_i$, defined as $\Delta p_i^c = p_i - p^c$, $\Delta p_i^c \in \mathbb{R}^3$. In addition, to encode proposal size information, we calculate the relative coordinates for each keypoint $p_i$ with the $n$-th corner $p^n, n = 1, \ldots, 8$ of the corresponding 3D proposal, defined as $\Delta p_i^n = p_i - p^n$, $\Delta p_i^n \in \mathbb{R}^3$. The reflection $r_i \in \mathbb{R}^1$ and segmentation scores $s_i \in \mathbb{R}^1$ of each keypoint are concatenated with the calculated proposal-aware point relative coordinates. We integrate the raw point cloud, segmentation, and proposal information into a new representation, which is formulated as

$$f_i = [\Delta p_i^c, \Delta p_i^1, \ldots, \Delta p_i^8, r_i, s_i], f_i \in \mathbb{R}^{29}. \tag{6}$$

**Hybrid Aggregation.** A feed-forward network encodes the new representation $f_i$ into the feature space to learn geometric features $f_g$, and we add the geometric features to the semantic features $f_s$ to obtain the full encoded features. Finally, we introduce three DGT layers to encode the fused features to obtain a discriminative feature vector $f_{output}$ for the final confidence classification and box regression, as follows:

$$f_g = FFN(f_i), \tag{7}$$

$$f_{output} = DGT(f_g + f_s). \tag{8}$$

### 3.4. Loss function

Our DGT-Det3D is trained using an end-to-end strategy, and the objective function consists of an RPN loss and a Refinement loss.

The overall loss is the sum of the RPN loss and box refinement loss, computed as follows:

$$L_{total} = L_{rpn} + L_{ref}. \tag{9}$$

**RPN Loss.** The segmentation head utilizes ground-truth boxes to generate segmentation masks to achieve segmentation tasks. Furthermore, to address the imbalance between the number of foreground and background points, we adopt focal loss [4] as

$$\begin{aligned} L_{seg} &= \frac{1}{N_{fg}} \sum_{i}^{N} -\alpha(1-l_i)^{\gamma} \log(l_i), \\ where,\ l_i &= \begin{cases} l_i, & \text{if } l_i = 1 \\ 1-l_i, & \text{otherwise,} \end{cases} \end{aligned} \tag{10}$$

where $l_i$ is the foreground/background probability at each point. We refer to the default parameter settings of the original study [17], e.g. $\alpha = 0.25$ and $\gamma = 2$.

Then, we generate and regress 3D proposals from foreground point segmentation. Each ground truth bounding box and a matching positive anchor are parameterized as $(x_g, y_g, z_g, h_g, w_g, l_g, \theta_g)$ and $(x_a, y_a, z_a, h_a, w_a, l_a, \theta_a)$, respectively, where $x, y, z$ represent the center location of the ground truth box or anchor, $h, w, l$ are the length, width, and height of the ground truth box or anchor, respectively, and $\theta$ is the yaw rotation around the $Z$-axis. We encode the regression targets $\delta_g \in \mathbb{R}^7$ as follows:

$$\begin{aligned} &\delta_x = \frac{x_g - x_a}{d_a}, \delta_y = \frac{x_g - x_a}{d_a}, \delta_z = \frac{z_g - z_a}{l_a}, \\ &\delta_h = \log(\frac{h_g}{h_a}), \delta_w = \log(\frac{w_g}{w_a}), \delta_l = \log(\frac{l_g}{l_a}), \\ &\delta_\theta = \theta_g - \theta_a, \end{aligned} \tag{11}$$

where $d_a = \sqrt{w_a^2 + h_a^2}$. The corresponding regression output is denoted as $\delta \in \mathbb{R}^7$.

The box regression loss is formulated as a smooth L1 loss [2] of

$$L_{reg} = \frac{1}{N_{fg}} \sum_{i}^{N_c} \mathbb{1}[l_i = 1]\text{Smooth-L1}(\delta, \delta_g), \tag{12}$$

where $N_{fg}$ denotes the number of foreground points.

The loss of the RPN is a combination of segmentation and box regression loss, defined as

$$L_{rpn} = \beta_{seg} L_{seg} + \beta_{reg} L_{reg}, \tag{13}$$

where $\beta_{seg}$ and $\beta_{reg}$ are weights to balance the RPN stage's segmentation and box regression loss, respectively.

**Refinement Loss.** In the refinement stage, we use the binary cross-entropy loss to compute the classification loss as follows:

$$L_{cls} = \frac{1}{N} \sum_{i}^{N_c} p_g \cdot log(p_i) + (1 - p_g) \cdot log(1 - p_i), \tag{14}$$

where $i$ indexes the foreground points, $p_g$ is the target, and $p_i$ is the prediction.

Moreover, box regression loss [2] adopts a smooth L1 loss consistent with the RPN regression loss, which is defined as

$$L_{reg} = \frac{1}{N_{fg}} \sum_{i}^{N_c} \mathbb{1}[l_i = 1]\text{Smooth-L1}(\delta, \delta_g). \tag{15}$$

The loss of the refinement sub-network is combined with the classification loss and box regression loss:

$$L_{ref} = \zeta_{cls} L_{cls} + \zeta_{reg} L_{reg}, \tag{16}$$

where $\zeta_{cls}$ and $\zeta_{reg}$ are the weights for balancing the classification and box regression loss of the refinement stage, respectively.

## 4. Experiments

In this section, we first present implementation details. Then, we provide quantitative and qualitative results with discussions on the KITTI and Waymo open datasets. Finally, we conduct extensive ablation studies to validate each component of our approach.

### 4.1. Experimental setup

**KIITI Datasets.** We evaluate our method on the KITTI dataset for 3D object detection. The KITTI dataset contains 7481 images and LiDAR samples for training and 7518 images for testing. We perform the experiments on two major classes: cars and cyclists. For each class, the benchmark considers three levels of difficulty: easy, moderate, and hard, based on the object size, occlusion level, and truncation. Following the common protocol, we further split the original training data into 3712 training samples and 3769 validation samples. For evaluation of the test set, the average precision (AP) metric is calculated using 40 recall positions with the 3D IoU of the car and cyclist, 0.7 and 0.5, respectively. For a fair comparison with other approaches, the results on the val set are calculated with 11 recall positions. Our results are compared with different methods for both validation samples and test samples.

**Waymo Open Dataset.** We adopt the DGT module to enhance the backbone of PV-RCNN [19] and IA-SSD [14]. Then, we evaluate the performance of the improved methods on a large-scale Waymo open dataset, which is more challenging than the KITTI dataset. The Waymo Open Dataset contains 798 sequences with nearly 158k Lidar samples in the training set and 202 sequences with almost 40k Lidar samples in the validation set. The objects are split into two difficulty levels, where the *LEVEL*_1 objects have more than five points and the *LEVEL*_2 objects have at least one LiDAR point. For a fair comparison, we only use DGT modules to replace the SA modules of the PV-RCNN and IA-SSD, while keeping the remaining consistent.

**Network Architecture.** As shown in Fig. 2, the details of the DGT network architecture are represented. For consistency of the input dimensions, we follow [17], which randomly samples $N = 16\,384$ points from all points in each scene as input. In the first stage, we construct a dynamic graph and process it through four DGT layers to extract local spatial and long-range semantic features. The four DGT layers downsample and group points into (4096, 1024, 256, 128). In each layer, we recompute the k-nearest neighbor (k = 24) graph as the input of a DGT module. The four DGT layers generate multi-scale features with dimensions (64, 128, 256, 512). Finally, point-wise semantic features are obtained from four feature propagation layers for subsequent segmentation and 3D proposal bounding box generation.

In the second stage, each proposal randomly samples 512 points as the input of the proposal-aware fusion module. Our

**Table 1**
Performance comparisons on the KITTI 3D object detection *test* set with AP calculated by 40 recall positions. The evaluation metric is Average Precision (AP) with an IoU threshold of 0.7 for cars and 0.5 for cyclists. The best results in point-based methods are shown in bold, and the best in all results are underlined.

| | Method | Stage | Car-3D detection | | | Cyclist-3D detection | | |
|---|---|---|---|---|---|---|---|---|
| | | | Easy | Moderate | Hard | Easy | Moderate | Hard |
| Voxel-based | VoxelNet [24] | 1-stage | 77.82 | 64.17 | 57.51 | 61.22 | 48.36 | 44.37 |
| | SECOND [8] | 1-stage | 83.34 | 72.55 | 65.82 | 71.33 | 52.08 | 45.83 |
| | PointPillars [7] | 1-stage | 82.58 | 74.31 | 68.99 | 77.10 | 58.65 | 51.92 |
| | TANet [46] | 2-stage | 83.81 | 75.38 | 67.66 | 73.84 | 59.86 | 53.46 |
| | Part-$A^2$ [53] | 2-stage | 87.81 | 78.49 | 73.51 | – | – | – |
| | SA-SSD [13] | 1-stage | 88.75 | 79.79 | 74.16 | – | – | – |
| | CIA-SSD [25] | 1-stage | 89.59 | 80.28 | 72.87 | – | – | – |
| Point-voxel | Fast Point R-CNN [54] | 2-stage | 85.29 | 77.40 | 70.24 | – | – | – |
| | STD [29] | 2-stage | 87.95 | 79.71 | 75.09 | 78.69 | 61.59 | 55.30 |
| | HVPR [53] | 1-stage | 86.38 | 77.92 | 73.04 | – | – | – |
| | PV-RCNN [19] | 2-stage | 90.25 | 81.43 | 76.82 | 78.60 | 63.71 | 57.65 |
| | CT3D [12] | 2-stage | 87.83 | 81.77 | 77.16 | – | – | – |
| Point-based | PointRCNN [17] | 2-stage | 86.96 | 75.64 | 70.70 | 74.96 | 58.82 | 52.53 |
| | Point-GNN [21] | 1-stage | 88.33 | 79.47 | 72.29 | **78.60** | 63.48 | 57.08 |
| | 3DSSD [18] | 1-stage | **88.36** | 79.57 | 74.55 | – | – | – |
| | IA-SSD [14] | 1-stage | 88.34 | 80.13 | 75.04 | 78.35 | 61.94 | 55.70 |
| | **DGT-Det3D (ours)** | **2-stage** | 87.89 | **80.68** | **76.02** | 78.06 | **64.80** | **58.08** |

**Table 2**
Performance comparison on the KITTI *val* set with AP calculated by 11 recall positions for car class.

| Method | $AP_{3D}$ (%) | | |
|---|---|---|---|
| | Easy | Moderate | Hard |
| VoxelNet [24] | 81.97 | 65.46 | 62.85 |
| PointPillars [7] | 86.62 | 76.06 | 68.91 |
| 3DSSD [18] | 89.71 | 79.45 | 78.67 |
| Part-$A^2$ [53] | 89.47 | 79.47 | 78.54 |
| STD [9] | 89.70 | 79.80 | **79.30** |
| SA-SSD [13] | **90.15** | 79.91 | 78.78 |
| PV-RCNN [19] | 89.35 | 83.69 | 78.70 |
| PointRCNN [17] | 88.88 | 78.63 | 77.38 |
| DGT-Det3D | 89.65 | **84.82** | 78.76 |

**Table 3**
Performance of DGT-Det3D on the KITTI *val* set with AP calculated by 40 recall positions for car class.

| IoU | 3D mAP | | | BEV mAP | | |
|---|---|---|---|---|---|---|
| Thresh. | Easy | Moderate | Hard | Easy | Moderate | Hard |
| 0.7 | 92.44 | 85.16 | 82.94 | 95.41 | 91.28 | 89.35 |

PAF module processes proposal-aware geometric features by an FFN (64,128) and fuses produced geometric features with RPN semantic features through an addition operation. Then, three DGT layers are employed to produce a single feature vector for predicting object confidence classification and 3D bounding box regression.

**Data Augmentation.** We use some widely used data augmentation strategies [8,19] to prevent overfitting in the training process. We introduce a commonly used copy-and-paste strategy to improve the model performance. Furthermore, we apply random flipping along the $X$-axis, global scaling with a random uniform scale sampled from [0.95, 1.05] and global rotation with a random degree sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$ for all points.

**Training and Inference Details.** DGT-Det3D is trained end-to-end by the ADAM optimizer with an initial learning rate of 0.01 and a batch size of 16 for 80 epochs on 8 GTX 2080Ti GPUs for the KITTI dataset. For the second proposal refinement stage, 128 proposals are randomly selected with 512 sampling keypoints. We consider a proposal to train the classification head as positive if the 3D IoU with ground-truth boxes is above 0.6. To train the box regression head, if a proposal has at least 0.55 3D IoU with ground-truth boxes, it is considered positive. Four sub-loss weights $\beta_{seg}$, $\beta_{reg}$, $\zeta_{cls}$, and $\zeta_{reg}$ are set to 1 for the best performance. Finally, we employ an NMS threshold of 0.1 to remove the redundant boxes in post-processing.

### 4.2. Comparisons on the KITTI dataset

By submitting our results to the KITTI online test server, we compare our DGT-Det3D against the baseline PointRCNN [17] and state-of-the-art methods on the KITTI test set. As shown in Table 1, in 3D detection tasks, our method outperforms most of the competitors. By capturing the long-range context and taking full advantage of spatial information, our DGT-Det3D demonstrates a significant performance improvement over PointRCNN [17] in the detection of each category and each difficulty level by car +0.93%, +5.04%, +5.32%, and cyclist +3.10%, +5.98%, +5.55%. Furthermore, our DGT-Det3D surpasses previous state-of-the-art point-based detectors IA-SSD [14] in the moderate and hard difficulty levels by car +0.55%, +0.98%, and cyclist +2.86%, +2.38%.

Furthermore, we provide the performance of 3D and bird's eye view BEV car detection results with AP calculated by recalling 11 positions on the val split for further comparison, as shown in Table 2. Table 3 presents the 3D object detection and BEV car detection results of the proposed DGT-Det3D, which are calculated by recalling 40 positions with an (IoU) threshold of 0.7. Given the val and test set results, we can observe that DGT-Det3D achieves state-of-the-art car performance compared to all point-based methods. In addition, for cyclist and hard-level car detection, DGT-Det3D achieves a comparable result to the state-of-the-art 3D detection method.

Finally, we demonstrate the advantages of DGT-Det3D by comparing the proposed DGT-Det3D and PointRCNN with qualitative results, as shown in Fig. 6. Specifically, the upper area predicts DGT-Det3D, and the ground truth is projected onto the images. The central area is the prediction of PointRCNN and the ground truth in the 3D field. The lower area predicts DGT-Det3D and the ground truth in the 3D field. It can be observed that our DGT-Det3D can detect serious occlusions and long-distance objects that contain only a small number of point clouds. We attribute

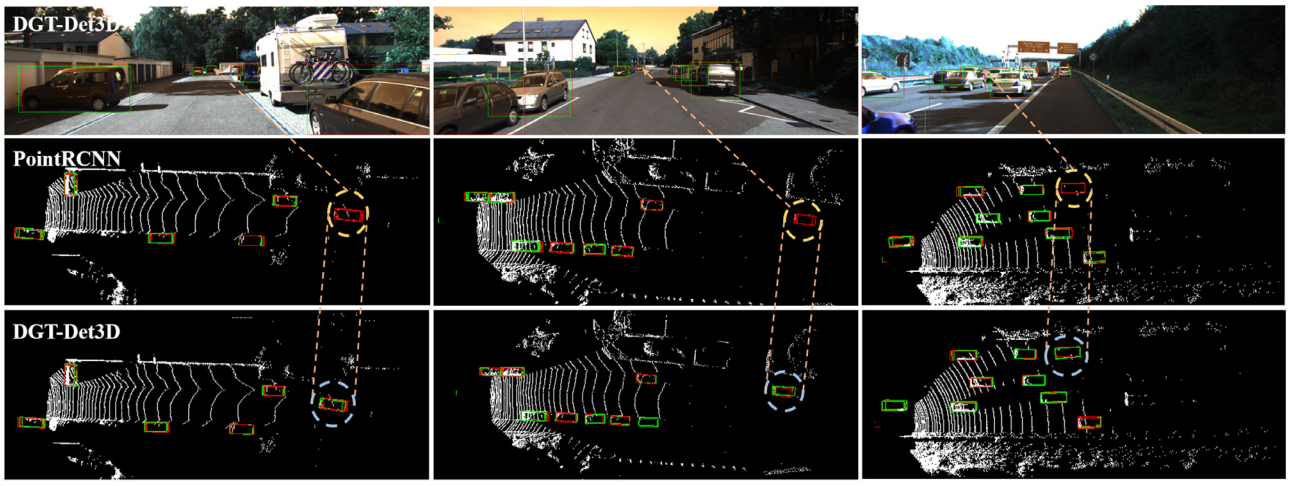


**Fig. 6.** Qualitative 3D detection results of DGT-Det3D on the KITTI *val* set. Specifically, the upper area predicts DGT-Det3D and ground truth in images. The central area predicts PointRCNN and ground truth in the 3D field. The lower area is the prediction of DGT-Det3D and ground truth in the 3D field. *Red* boxes are the ground truth boxes. *Green* boxes are the predicted boxes. Inside the *golden dashed* boxes are the distant or occluded targets that are not detected by the baseline method, and inside the *blue dashed* boxes are the results detected by DGT-Det3D overcoming the problem. The *orange* arrows point to the location in the corresponding images of the objects.

**Table 4**
Performance comparisons of IA-SSD and PV-RCNN with DGT module on the Waymo Open Dataset 3D object detection with 202 validation sequences.

| Method | Rep. | Veh. (LEVEL 1) | | Veh. (LEVEL 2) | | Ped. (LEVEL 1) | | Ped. (LEVEL 2) | | Cyc. (LEVEL 1) | | Cyc. (LEVEL 2) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | mAP | mAPH | mAP | mAPH | mAP | mAPH | mAP | mAPH | mAP | mAPH | mAP | mAPH |
| PointPillars [7] | Voxel-based | 60.67 | 59.79 | 52.78 | 52.01 | 43.49 | 23.51 | 37.32 | 20.17 | 35.94 | 28.34 | 34.60 | 27.29 |
| SECOND [8] | Voxel-based | 68.03 | 67.44 | 59.57 | 59.04 | 61.14 | 50.33 | 53.00 | 43.56 | 54.66 | 53.31 | 52.67 | 51.37 |
| Part-A2 [53] | Voxel-based | 71.82 | 71.29 | 64.33 | 63.82 | 63.15 | 54.96 | 54.24 | 47.11 | 65.23 | 63.92 | 62.61 | 61.35 |
| PV-RCNN [19] | Point-voxel | 74.06 | 73.38 | 64.99 | 64.38 | 62.66 | 52.68 | 53.80 | 45.14 | 63.32 | 61.71 | 60.72 | 59.18 |
| Voxel-RCNN [9] | Voxel-based | 75.59 | – | 66.59 | – | – | – | – | – | – | – | – | – |
| VoTr-TSD [23] | Voxel-based | 74.95 | 74.25 | 65.91 | 65.29 | – | – | – | – | – | – | – | – |
| IA-SSD [14] | Point-based | 70.53 | 69.67 | 61.55 | 60.80 | 69.38 | 58.47 | 60.30 | 50.73 | 67.67 | 65.30 | 64.98 | 62.71 |
| DGT-IA | Point-based | 72.23 | 71.44 | 63.56 | 62.86 | 69.30 | 59.29 | 60.75 | 51.86 | 67.76 | 65.46 | 65.21 | 62.99 |
| *Improvement* | – | *+1.7* | *+1.77* | *+2.01* | *+2.06* | *−0.08* | *+0.82* | *+0.45* | *+1.13* | *+0.09* | *+0.16* | *+0.23* | *+0.28* |
| PV-RCNN (CenterHead) [19] | Point-voxel | 75.95 | 75.43 | 68.02 | 67.54 | 75.94 | 69.40 | 67.66 | 61.62 | 70.18 | 68.98 | 67.73 | 66.57 |
| DGT-PV | Point-voxel | **77.00** | **76.47** | **68.37** | **67.88** | **77.30** | **70.92** | **69.06** | **63.14** | **71.56** | **70.30** | **69.05** | **67.82** |
| *Improvement* | – | *+1.05* | *+1.04* | *+0.35* | *+0.34* | *+1.36* | *+1.52* | *+1.4* | *+1.52* | *+1.38* | *+1.32* | *+1.77* | *+1.25* |

**Table 5**
Performance comparisons on the Waymo Open Dataset 3D object detection with 202 validation sequences for vehicle detection by distance.

| Method | Rep. | Veh. (LEVEL 1) mAP by distance | | | | Veh. (LEVEL 2) mAP by distance | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Overall | 0 m–30 m | 30 m–50 m | 50 m–inf | Overall | 0 m–30 m | 30 m–50 m | 50 m–inf |
| PointPillars [7] | Voxel-based | 56.62 | 81.01 | 51.57 | 29.94 | – | – | – | – |
| MVF [55] | Voxel-based | 62.93 | 86.30 | 60.02 | 36.02 | – | – | – | – |
| PV-RCNN [19] | Point-voxel | 70.60 | 91.92 | 69.21 | 42.17 | 65.36 | 91.58 | 65.13 | 36.45 |
| Voxel-RCNN [9] | Voxel-based | 75.59 | **92.49** | 74.09 | 53.51 | 66.59 | 91.74 | 67.89 | 40.80 |
| VoTr-TSD [23] | Voxel-based | 74.95 | 92.28 | 73.36 | 51.09 | – | – | – | – |
| PV-RCNN (CenterHead) [19] | Point-voxel | 75.95 | 92.45 | 74.52 | 53.02 | 63.56 | 91.59 | 68.44 | 40.79 |
| DGT-PV | Point-voxel | **77.00** | 92.45 | **75.41** | **54.47** | **68.37** | **91.61** | **68.45** | **42.28** |
| *Improvement* | – | *+1.05* | *+0.0* | *+0.89* | *+1.45* | *+0.35* | *+0.02* | *+0.01* | *+1.49* |

this improvement to the dynamic graph, which retains valuable foreground features, and the transformer, which improves long-range context learning.

### 4.3. Comparisons on the Waymo open dataset

To validate the effectiveness of our proposed plug-and-play DGT module, we conduct experiments on the larger and more challenging Waymo open dataset. We consider the state-of-the-art point-based method IA-SSD and point-voxel-based PV-RCNN as the baseline and replace the SA module with the DGT module, termed DGT-IA and DGT-PV, respectively. We train the two enhanced methods on the training set and evaluate them on the validation set. As shown in Table 4, our DGT module achieves consistent improvement for both baselines, surpassing IA-SSD and PV-RCNN by +1.77%, +1.13%, +0.28% and +0.34%, +1.52%, +1.25% for vehicle, pedestrian, and cyclist LEVEL2 3D mAPH, respectively. Especially, as shown in Table 5, the performance improvement brought by our DGT module is mainly in the 30m-50 m and 50m-inf range by +0.89%, +1.45% and +0.01%, +1.49% on vehicle LEVEL2 3D mAP. Such results indicate that the DGT module can capture long-range relationships and improve the detection of distant and occluded objects. As shown in Fig. 7, we

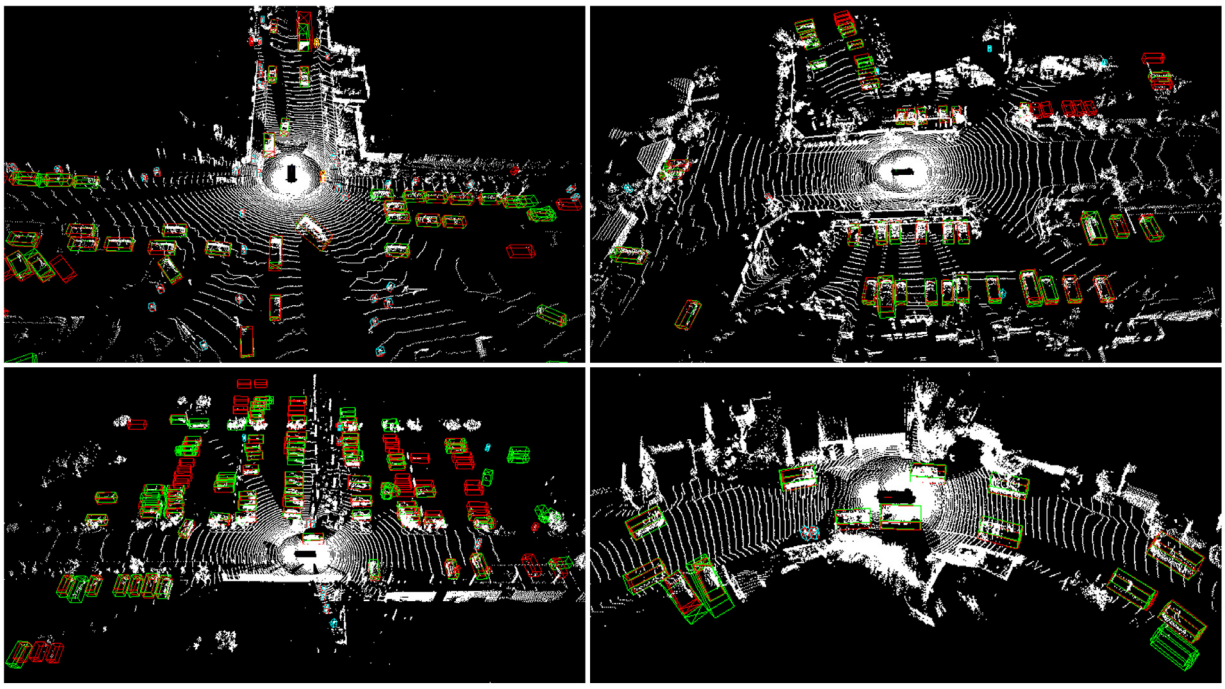

**Fig. 7.** Qualitative 3D detection results of DGT-IA on the Waymo *val* set. Note that, the predicted bounding boxes are shown in *green* for car, *yellow* for cyclist, and *cyan* for pedestrian. *Red* boxes in 3D are the ground truth boxes.

**Table 6**
Ablation for DGT-SSD on KITTI *val* set.

| Methods | Easy | Mod. | Hard |
|---|---|---|---|
| PointRCNN (1-stage) | 57.99 | 51.60 | 49.10 |
| DGT-SSD | **79.29** | **66.44** | **63.50** |

also visualize the qualitative detection results for all categories produced by DGT-IA on the Waymo open dataset validation set.

### 4.4. Ablation studies

In this study, our main contributions lie in two aspects: dynamic graph transformer and proposal-aware fusion. In this section, comprehensive ablation studies are performed to verify the effectiveness of each component. We present a 3D detection AP metric with 40 recall positions on the KITTI val set.

**Effectiveness of DGT-SSD.** Our DGT module significantly improves the one-stage feature encoding capability of point-based methods, which is beneficial for the RPN to generate better proposals. To verify the feature extraction performance of our proposed DGT module, we conduct experiments directly using the first stage of DGT-Det3D as a detector, named DGT-SSD, as shown in Table 6. For the car detection performance, our DGT-SSD achieves a significant improvement over PointNet++ [16] by +21.3%, +14.84%, and +14.4%. Therefore, the RPN of DGT-Det3D can provide high-quality proposals and semantic features for the second-stage refinement.

**Effectiveness of Dynamic Graph Transformer.** From Table 7, we further validate the effectiveness of the proposed DGT. Our DGT module significantly improves the backbone feature encoding performance, resulting in a consistent improvement for the car classes by +0.29%, +2.54%, and +1.89%. We further verify the effect of the dynamic graph and different self-attention mechanisms on the results. As shown in Table 8, we compare our graph-aware self-attention with standard self-attention, which

**Table 7**
Ablation experiments on the effect of DGT, Proposal-aware Fusion on the KITTI *val* set.

| DGT | PAF | Easy | Mod. | Hard |
|---|---|---|---|---|
| – | – | 91.15 | 80.67 | 78.28 |
| ✓ | – | 91.44 | 83.21 | 80.17 |
| – | ✓ | **92.44** | 83.18 | 80.90 |
| ✓ | ✓ | 92.16 | **85.16** | **82.94** |

**Table 8**
Effectiveness of the Graph-aware Self-Attention (GSA) and the standard Self-Attention (standard SA).

| Standard SA | GSA | Easy | Mod. | Hard |
|---|---|---|---|---|
| – | – | 90.77 | 82.46 | 80.43 |
| ✓ | – | 91.95 | 83.17 | 82.72 |
| – | ✓ | **92.16** | **85.16** | **82.94** |

can improve the feature encoding capability to a certain extent, but our method can achieve superior performance. In the DGT module, we aggregate the local point cloud features by building a graph. Therefore, we explore the effect of the number of graph-connected neighbors K on the detection accuracy, efficiency, and memory consumption in Table 9. It can be seen that if K is extremely small, local features cannot be effectively learned, and the performance will be poor, whereas a large K will reduce the efficiency and will not continue to improve the accuracy. Therefore, we choose $k = 24$ to balance accuracy and efficiency. The results demonstrate the effectiveness and robustness of the DGT network.

**Effectiveness of Dual-branch Relative Positional Encoding.** As presented in Table 10, we further investigate the importance of DRPE in the GSA module. The first row of the table presents the baseline performance without any positional encoding. The table second line demonstrates the absolute positional encoding increase in cars by −0.17%, +2.09%, and +2.14%. Then, we introduce and verify the effectiveness of DRPE with an attention computing (AC) branch and a value (V) branch, respectively. The

**Table 9**
Effectiveness of the number of neighbors in dynamic graph.

| K | Easy | Mod. | Hard | Memory |
|---|---|---|---|---|
| k = 1 | 91.22 | 82.77 | 80.54 | **1646** |
| k = 8 | 91.77 | 83.09 | 82.50 | 1728 |
| k = 16 | 91.90 | 85.14 | 82.89 | 1795 |
| k = 24 | **92.16** | **85.16** | **82.94** | 1882 |
| k = 32 | 92.00 | 85.02 | 82.87 | 2028 |

**Table 10**
Ablation for Dual-branch Relative Positional Encoding (DRPE) on KITTI *val* set. "AC" and "V" represent the attention computing branch and value branch, respectively.

| Methods | Easy | Mod. | Hard |
|---|---|---|---|
| Without PE | 91.93 | 80.95 | 80.28 |
| Absolute PE | 91.76 | 83.04 | 82.42 |
| AC | 91.79 | 83.13 | 82.63 |
| V | 91.79 | 83.14 | 82.41 |
| DRPE | **92.16** | **85.16** | **82.94** |

**Table 11**
Efficiency of DGT-Det3D.

| Methods | Stage | Memory (MB) | Runtime (ms) | Params (M) |
|---|---|---|---|---|
| IA-SSD | 1-stage | 1613 | 55 | 2.7 |
| PV-RCNN | 2-stage | 3383 | 111 | 13.1 |
| PointRCNN | 2-stage | 1923 | 116 | **4.0** |
| DGT-Det3D | 2-stage | **1882** | **107** | 7.9 |

AC and V branches achieve +2.95%, +2.35%, and +2.13% precision improvements in the car's moderate and hard difficulty levels, respectively. When we combine both branches, we obtain the best performance with increases of +0.23%, +4.21%, and +2.66%.

**Effectiveness of Proposal-aware Fusion.** In Table 7, we demonstrate the importance of the PAF module, which gains +1.29%, +2.51%, and +2.32% mAP improvement. We can observe that the detection performance can be significantly improved when the geometric features of the proposal are fully encoded and fused with high-quality semantic features from the first stage.

**Efficiency of DGT-Det3D.** Finally, we evaluate the computational and memory efficiencies of the proposed DGT-Det3D. For a fair comparison, we re-implement several representative approaches and report the memory, speed, and parameters on the same platform. As shown in Table 11, we achieve higher efficiency than 2-stage methods, such as PV-RCNN and PointRCNN. Our DGT-Det3D can be accelerated by reducing the number of graph neighbors k, such as k = 16 in Table 9, which slightly affects performance.

## 5. Conclusion

In this paper, we propose a robust and effective two-stage dynamic graph transformer 3D object detection network (DGT-Det3D), which processes point clouds directly. In the first stage, our DGT backbone builds graph connections among points and dynamically updates graph connections to concentrate on more valuable foreground data. The DGT module is then adopted to encode graph features for capturing long-range context and extracting precise spatial information. In the second stage, we propose a proposal-aware module to fully fuse the geometric and semantic features and refine the final classification and regression of the 3D box. Quantitative and qualitative experiments on the KITTI dataset demonstrate that DGT-Det3D learns robust and discriminatory features and can detect distant and occluded objects. Notably, DGT is a plug-and-play module that achieves consistent improvements in state-of-the-art point-based methods on the large-scale Waymo Open Dataset, particularly for distant and occluded objects. The results demonstrate the potential of graph-based transformers for 3D detection tasks.

## CRediT authorship contribution statement

**Siyuan Ren:** Conceptualization, Methodology, Software, Validation, Investigation, Visualization, Writing – original draft. **Xiao Pan:** Writing – original draft, Validation, Visualization. **Wenjie Zhao:** Writing – review & editing, Supervision, Funding acquisition. **Binling Nie:** Validation. **Bo Han:** Supervision, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The authors do not have permission to share data.

## Acknowledgments

This work was supported by the Key Research and Development Program of Zhejiang Province, China (Grant No. 2020C05001), the Shanghai Aerospace Science and Technology Innovation Fund, China (Grant No. SAST2019-10), the National Natural Science Foundation of China (Grant No. 61703366), the Fundamental Research Funds for the Central Universities, China (Grant No. 2021QNA4030).

## References

[1] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: European Conference on Computer Vision, Springer, 2016, pp. 21–37.
[2] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Syst. 28 (2015) 91–99.
[3] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.
[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.
[5] X. Chen, H. Ma, J. Wan, B. Li, T. Xia, Multi-view 3d object detection network for autonomous driving, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1907–1915.
[6] J. Ku, M. Mozifian, J. Lee, A. Harakeh, S.L. Waslander, Joint 3d proposal generation and object detection from view aggregation, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1–8.
[7] A.H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, O. Beijbom, Pointpillars: Fast encoders for object detection from point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12697–12705.
[8] Y. Yan, Y. Mao, B. Li, Second: Sparsely embedded convolutional detection, Sensors 18 (10) (2018) 3337.
[9] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, H. Li, Voxel R-CNN: Towards high performance voxel-based 3D object detection, 2020, arXiv preprint arXiv:2012.15712.
[10] Z. Li, F. Wang, N. Wang, Lidar r-cnn: An efficient and universal 3d object detector, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 7546–7555.
[11] J.S. Hu, T. Kuai, S.L. Waslander, Point density-aware voxels for lidar 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8469–8478.
[12] H. Sheng, S. Cai, Y. Liu, B. Deng, J. Huang, X.-S. Hua, M.-J. Zhao, Improving 3D object detection with channel-wise transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2743–2752.

[13] C. He, H. Zeng, J. Huang, X.-S. Hua, L. Zhang, Structure aware single-stage 3d object detection from point cloud, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11873–11882.
[14] Y. Zhang, Q. Hu, G. Xu, Y. Ma, J. Wan, Y. Guo, Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 18953–18962.
[15] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 652–660.
[16] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017, arXiv preprint arXiv:1706.02413.
[17] S. Shi, X. Wang, H. Li, Pointrcnn: 3d object proposal generation and detection from point cloud, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 770–779.
[18] Z. Yang, Y. Sun, S. Liu, J. Jia, 3Dssd: Point-based 3d single stage object detector, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11040–11048.
[19] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, H. Li, Pv-rcnn: Point-voxel feature set abstraction for 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10529–10538.
[20] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Trans. Graph. (TOG) 38 (5) (2019) 1–12.
[21] W. Shi, R. Rajkumar, Point-gnn: Graph neural network for 3d object detection in a point cloud, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1711–1719.
[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[23] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, C. Xu, Voxel transformer for 3D object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 3164–3173.
[24] Y. Zhou, O. Tuzel, Voxelnet: End-to-end learning for point cloud based 3d object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.
[25] W. Zheng, W. Tang, S. Chen, L. Jiang, C.-W. Fu, Cia-ssd: Confident iou-aware single-stage object detector from point cloud, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 3555–3562.
[26] T. Yin, X. Zhou, P. Krahenbuhl, Center-based 3d object detection and tracking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11784–11793.
[27] C.R. Qi, W. Liu, C. Wu, H. Su, L.J. Guibas, Frustum pointnets for 3d object detection from rgb-d data, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 918–927.
[28] Z. Wang, K. Jia, Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2019, pp. 1742–1749.
[29] Z. Yang, Y. Sun, S. Liu, X. Shen, J. Jia, Std: Sparse-to-dense 3d object detector for point cloud, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1951–1960.
[30] J. Noh, S. Lee, B. Ham, Hvpr: Hybrid voxel-point representation for single-stage 3d object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14605–14614.
[31] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.
[32] Y. Yang, Z. Guan, J. Li, W. Zhao, J. Cui, Q. Wang, Interpretable and efficient heterogeneous graph convolutional network, IEEE Trans. Knowl. Data Eng. (2021).
[33] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen, A. Mian, Bi-CLKT: Bi-graph contrastive learning based knowledge tracing, Knowl.-Based Syst. 241 (2022) 108274.
[34] Z. Du, H. Ye, F. Cao, 3D mixed CNNs with edge-point feature learning, Knowl.-Based Syst. 221 (2021) 106985.
[35] J. Zarzar, S. Giancola, B. Ghanem, Pointrgcn: Graph convolution networks for 3D vehicles detection refinement, 2019, arXiv preprint arXiv:1911.12236.
[36] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv preprint arXiv:1409.0473.
[37] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, 2020, arXiv preprint arXiv:2010.11929.
[39] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, 2021, arXiv preprint arXiv:2103.14030.
[40] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer, 2020, pp. 213–229.
[41] H. Zhao, L. Jiang, J. Jia, P.H. Torr, V. Koltun, Point transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 16259–16268.
[42] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R.R. Martin, S.-M. Hu, PCT: Point cloud transformer, Comput. Vis. Media 7 (2) (2021) 187–199.
[43] I. Misra, R. Girdhar, A. Joulin, An end-to-end transformer model for 3D object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2906–2917.
[44] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, Q. Tian, Modeling point clouds with self-attention and gumbel subset sampling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3323–3332.
[45] G. Wang, Q. Zhai, H. Liu, Cross self-attention network for 3D point cloud, Knowl.-Based Syst. (2022) 108769.
[46] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, X. Bai, Tanet: Robust 3d object detection from point clouds with triple attention, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 11677–11684.
[47] P. Bhattacharyya, C. Huang, K. Czarnecki, Self-attention based context-aware 3D object detection, 2021, arXiv e-Prints arXiv–2101.
[48] H. Zhao, J. Jia, V. Koltun, Exploring self-attention for image recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10076–10085.
[49] P. Shaw, J. Uszkoreit, A. Vaswani, Self-attention with relative position representations, 2018, arXiv preprint arXiv:1803.02155.
[50] C. Xu, Z. Guan, W. Zhao, H. Wu, Y. Niu, B. Ling, Adversarial incomplete multi-view clustering, in: IJCAI, 2019, pp. 3933–3939.
[51] Y. Li, J. Wan, Q. Miao, S. Escalera, H. Fang, H. Chen, X. Qi, G. Guo, Cr-net: A deep classification-regression network for multimodal apparent personality analysis, Int. J. Comput. Vis. 128 (12) (2020) 2763–2780.
[52] C. Xu, W. Zhao, J. Zhao, Z. Guan, X. Song, J. Li, Uncertainty-aware multi-view deep learning for internet of things applications, IEEE Trans. Ind. Inform. (2022).
[53] S. Shi, Z. Wang, J. Shi, X. Wang, H. Li, From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network, IEEE Trans. Pattern Anal. Mach. Intell. (2020).
[54] Y. Chen, S. Liu, X. Shen, J. Jia, Fast point r-cnn, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9775–9784.
[55] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, V. Vasudevan, End-to-end multi-view fusion for 3d object detection in lidar point clouds, in: Conference on Robot Learning, PMLR, 2020, pp. 923–932.