IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Fast and Accurate 3D Object Detection for LiDAR-camera-based Autonomous Vehicles using One Shared Voxel-based Backbone

LI-HUA WEN, (Member, IEEE), KANG-HYUN JO, (Senior Member, IEEE)
The Graduate School of Electrical Engineering, University of Ulsan, 44610, Korea (e-mail: wenlihuawlh@gmail.com, acejo@ulsan.ac.kr)
Corresponding author: Kang-Hyun Jo (e-mail: acejo@ulsan.ac.kr).

This work was supported by University of Ulsan.

ABSTRACT Currently, many kinds of LiDAR-camera-based 3D object detectors have been developed with two heavy neural networks to extract view-specific features, while a LiDAR-camera-based 3D detector with only one neural network has not been implemented. To tackle this issue, this paper first presents an early-fusion method to exploit both LiDAR and camera data for fast 3D object detection with only one backbone, achieving a good balance between accuracy and efficiency. We propose a novel point feature fusion module to directly extract point-wise features from raw RGB images and fuse them with their corresponding point cloud with no backbone. In this paradigm, the backbone that extracts RGB image features is abandoned to reduce the large computation cost. Our method first voxelizes a point cloud into a 3D voxel grid and utilizes two strategies to reduce information loss during voxelization. The first strategy is to use a small voxel size (0.05m, 0.05m, 0.1m) in X-axis, Y-axis, and Z-axis, respectively, while the second one is to project the feature (e.g. intensity or height information) of point clouds onto RGB images. Numerous experiments evaluated on the KITTI benchmark suite show that the proposed approach outperforms state-of-the-art LiDAR-camera-based methods on the three classes in 3D performance (Easy, Moderate, Hard): cars (88.04%, 77.60%, 76.23%), pedestrians (66.65%, 60.49%, 54.51%), and cyclists (75.87%, 60.07%, 54.51%). Additionally, the proposed model runs at 17.8 frames per second (FPS), which is almost 2× faster than state-of-the-art fusion methods for LiDAR and camera.

INDEX TERMS LiDAR-camera-based 3D detector, single stage, one backbone, point-wise fusion, KITTI benchmark

I. INTRODUCTION

WITH the rapid development of autonomous vehicles, three-dimensional (3D) object detection has become more important, whose purpose is to perceive the size and accurate location of objects in the real world. Currently, an intelligent car is equipped with at least one LiDAR apparatus, one radar and one RGB camera. Note that radar is now widely used in companies, however, only a few researchers use it to validate a new algorithm. Hence, this paper focuses on LiDAR and camera for 3D object detection. LiDAR is employed to collect the surrounding 3D data, referred to as a point cloud, and the camera is used to capture a high-resolution RGB image. The two devices provide two important and different types of data. However, it is non-trivial to highly efficiently and quickly extract and fuse the features of the point cloud and RGB image.

Recently, feature extraction [1]–[4] with deep learning has drawn much attention. For the RGB image, a general 2D convolutional neural network (CNN) can be used to extract its features. For the point cloud however, it is difficult to extract its features due to its irregular distribution and sparse contributions. Before the advent of highly-efficient graphics processing units (GPUs), representative studies [5]–[10] have converted point clouds into 2D dense images or structured voxel-grid representations and utilized 2D neural networks to extract the corresponding feature from the converted 2D image. With the development of computer technology, the authors in [11]–[14] directly utilized a multi-layer perceptron

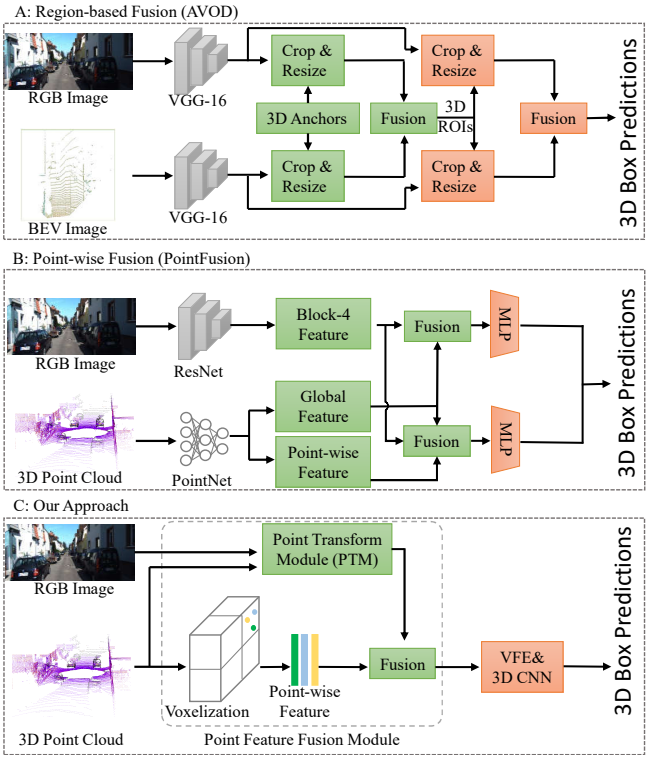

Figure 1: A comparison of two mainstream LiDAR-camera-based 3D detectors with the proposed approach. Both detectors A and B employ one backbone (VGG-16 or ResNet) to extract the features of the RGB image. Conversely, our proposed method directly extracts the pointwise features from the raw RGB image without a backbone.

(MLP) to aggregate features from point clouds. Shi et al. [15] encoded the point cloud natively in a graph using the points as the graph vertices.

To leverages the mutual advantages of point clouds and the RGB image, some researchers have attempted to fuse view-specific region of interest (ROI) features. Currently, there are two mainstream fusion methods. The first is to fuse two view-specific features, as shown in Figure 1A. The other method is pointwise feature fusion, as shown in Figure 1B. Chen et al. [5] and Ku et al. [6] directly fuse the ROI feature maps output with the two backbones of the point cloud and RGB image, respectively. On the other hand, Xu et al. [16] and Sindagi et al. [17] fuse pointwise features. These methods achieve better performance compared with LiDAR-based methods; however, their inference time is usually intolerable for application in real-time autonomous driving systems.

To deal with the above issues, this paper proposes a novel point-wise fusion strategy between point clouds and RGB images, illustrated in Figure 1C. The proposed method directly extracts pointwise features from the raw RGB image based on the raw point cloud first. Then, it fuses the two pointwise features and feeds them into a 3D neural network. The structure, as shown in Figure 2, has only one backbone to extract features, making the proposed model much faster than state-of-the-art LiDAR and camera fusion methods.

The key contributions of this work are as follows:

- This paper presents an early-fusion method to exploit both LiDAR and camera data for fast multi-class 3D object detection with only one backbone, achieving a good balance between accuracy and efficiency.
- This paper proposes a highly-efficient pointwise feature fusion module, which directly extracts the RGB image point feature based on a point cloud and fuses the extracted RGB image point feature with the corresponding feature of the point cloud.
- This paper also enhances 3D object detection with an RGB^+ image, which preserves the information projected from its corresponding point cloud.

The presented one-stage 3D multi-class object detection framework outperforms state-of-the-art LiDAR-camera-based methods on the KITTI benchmark [18] both in terms of the speed and accuracy.

II. RELATED WORK

This section starts by reviewing recent works in applying convolutional neural networks (CNNs) to 3D object detection based on LiDAR, and then focuses on methods specific to multi-modal 3D object detection from point clouds and RGB images.

A. LIDAR-BASED 3D OBJECT DETECTION

Recently, there have been three main 3D object detectors based on LiDAR: voxel-based detectors, point-based detectors, and graph-based detectors. Voxel-based methods [7], [8], [19]–[21] first voxelize the raw point cloud over a given range and then utilize a 3D CNN or 2D CNN to extract features. Different from other existing methods, Muresan et al. [21] employs a 4-beam LiDAR to collect point clouds. Unlike VoxelNet [7], Yan et al. [19] replaced a 3D CNN by a 3D sparse convolutional network, and Lang et al. [20] directly organized point clouds in vertical columns (pillars) to generate 2D BEV images. Point-based detectors [11]–[14] directly deal with the raw point cloud. Charles et al. [11] pioneered the method used to deal with each point independently using their shared MLPs. Based on PointNet [11], Charles et al. [12] further introduced the metric space distances to learn local features with increasing contextual scales. Yang et al. [13] abandoned the upsampling layers in PointNet++ to boost the inference speed. The proposed method voxelizes a point cloud using a dynamic voxelization method compared with the hard voxelization method in [7] and aims to avoid information loss during voxelization.

B. MULTI-MODAL 3D OBJECT DETECTION

3D Object detection in point clouds and RGB images is a fusion problem. As such, it is natural to extract the RGB image feature and the point cloud feature with two different backbones, respectively, which is the paradigm present in all

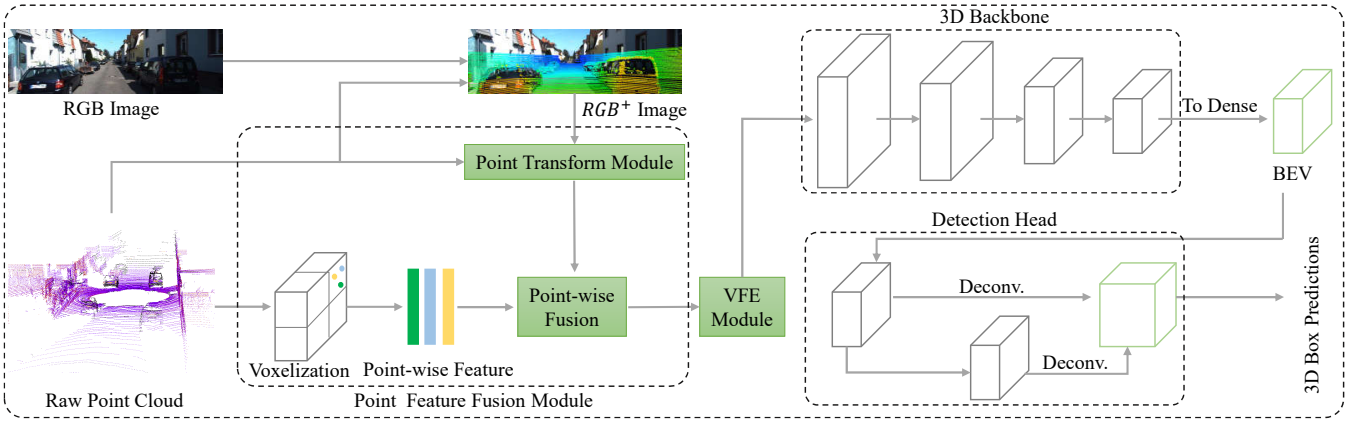

Figure 2: The architecture of the proposed one-stage 3D object detection network for the LiDAR and camera. It mainly includes the input data, the point feature fusion module, the 3D backbone, and the detection head. The gray box and green box represent the convolutional block and feature map, respectively.

previous works [5], [6], [9], [10], [16], [17], [22]–[25]. Obviously, by employing two heavy backbones, these approaches are very slow and consume a great deal of memory. In the paradigm, these methods are designed to either study how to fuse or how to improve accuracy based on state-of-the-art fusion methods, e.g., AVOD [6] changes the feature generation method in MV3D [5] from hand-crafted techniques to automation to improve the running speed of the model. According to different fusion methods, these methods can be divided into two categories: pointwise fusion [16], [17] and region of interest (ROI)-based fusion [5], [6], [22]–[26]. Daraei et al. [27] . Compared with the ROI-based fusion, pointwise fusion is more flexible. Inspired by pointwise fusion, this article will explore whether it is possible to directly aggregate the point features of the raw RGB image with point cloud features. Different from the previous methods, the proposed method only has one backbone. Additionally, the proposed model takes the RGB^+ image as the input, instead of using an RGB image.

In this paper, we first present an early-fusion method to exploit both LiDAR and camera data for fast 3D object detection with only one backbone, and it achieves a good balance between accuracy and efficiency. Thanks to the novel pointwise feature fusion module, which makes the fusion between LiDAR and camera data high efficient. To further improve the detection performance, we propose the RGB^+ image as the input.

III. PROPOSED APPROACH

The proposed model, as shown in Figure 2, takes point clouds and RGB images as inputs and predicts oriented 3D bounding boxes for cyclists, pedestrians, and cars. This model includes four main parts: (1) A point feature fusion module that extracts the point features from the RGB image and fuses the extracted features with the corresponding point cloud features, (2) a voxel feature encoder (VFE) module and a 3D backbone to process the fused pointwise features into a high-level representation, (3) a detection head that regresses and classifies the 3D bounding boxes, and (4) a loss function.

A. POINT FEATURE FUSION MODULE

The fusion module, shown in Figure 3, consists of three submodules: the point transform module, the voxelization of point clouds, and the pointwise fusion module. Since this module involves the input of raw data, before introducing the module, the input data is first introduced.

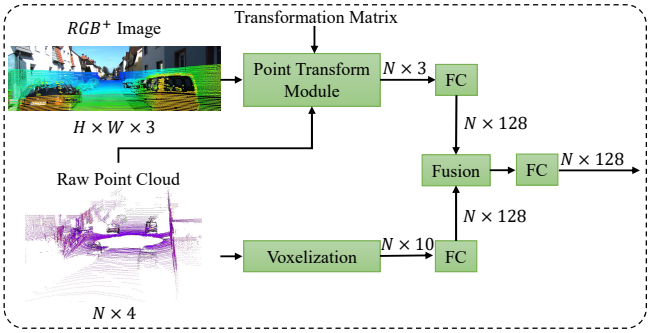

Figure 3: Visualization of the point feature fusion module. N is the number of points in a point cloud, and FC denotes one fully connected layer.

Input Data. This model accepts point clouds and RGB images as the input. To reduce the loss of raw point-cloud information during voxelization, a LiDAR point cloud is projected onto an RGB image and embedded into the image to generate a new image with three channels, called RGB$^+$. The RGB$^+$ object has two typical representations: the RGB$^{\mathbf{I}}$ portion that embeds the intensity of point clouds into an RGB image, and the RGB$^{\mathbf{D}}$ representation that embeds the Z-axis value of point clouds into the image. The detailed process of the RGB$^+$ generation is divided into the three following steps:

(1) First, point clouds (X, Y, Z) are mapped onto the original image $(W \times H)$ plane as follows:

$$\begin{pmatrix} u & v & 1 \end{pmatrix}^T = \mathbf{M} \cdot \begin{pmatrix} X & Y & Z & 1 \end{pmatrix}^T, \tag{1}$$

$$\mathbf{M} = \mathbf{P}_{rect} \cdot \begin{pmatrix} \mathbf{R}_{velo}^{cam} & \mathbf{t}_{velo}^{cam} \\ 0 & 1 \end{pmatrix}, \tag{2}$$

where (u, v) is the image coordinate, $\mathbf{P}_{rect}$ is a project matrix, $\mathbf{R}_{velo}^{cam}$ is the rotation matrix from LiDAR to the camera, $\mathbf{t}_{velo}^{cam}$ is a translation vector, and $\mathbf{M}$ is the homogeneous transformation matrix from LiDAR to the camera.

(2) Second, the points $\{(x, y, z) \mid x \in X, y \in Y, z \in Z\}$ located in the image of size $W \times H$ are kept. Meanwhile, the LiDAR points are projected to the camera coordinates and denoted as (x_c, y_c, z_c):

$$\begin{pmatrix} x_c & y_c & z_c \end{pmatrix}^T = \mathbf{M} \cdot \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T. \tag{3}$$

(3) Finally, z_c is mapped between 0 and 255 and then assigned to the corresponding image coordinate (u, v) to generate the $\text{RGB}^{\mathbf{D}}$ object. Similarly, the intensity of the point cloud for each color channel is mapped between 0 and 255 and then assigned to the corresponding image coordinate (u, v) to obtain the $\text{RGB}^{\mathbf{I}}$ data structure. This process uses the circle function of the OpenCV library.

Point Transform Module. This module extracts point features from the RGB^{+} image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ based on the raw point cloud. First, a point cloud $\mathbf{P} \in \mathbb{R}^{N \times 3}$ is projected onto its corresponding image by Eq. 1 to obtain the corresponding image coordinates (u_i, v_i). Second, the RGB^{+} and the (u_i, v_i) are fed into the image sampler [28], outputting the image point feature $\mathbf{P_i} \in \mathbb{R}^{N \times 3}$, where N is the number of points in the point cloud.

Voxelization. Voxelization divides the point cloud into evenly spaced voxel grids and then generates a many-to-one mapping between 3D points and their corresponding voxels. The details are shown in Figure 4. Currently, there exist two voxelization methods: hard voxelization [7] and dynamic voxelization [29]. Compared with the former, dynamic voxelization makes the detection more stable by preserving all the raw points and voxel information. This work applies the dynamic voxelization method. Given a point cloud $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$, the process assigns N points to a buffer of size $N \times F$, where N is the number of points and F denotes the feature dimension. Specifically, each point $p_i = [x_i, y_i, z_i, r_i]$ (containing the XYZ coordinates and the reflectance value) in a voxel is denoted by its inherent information (x_i, y_i, z_i, r_i), its relative offsets (x_v, y_v, z_v) with respect to the centroid of the points in the voxel, and its relative offsets (x_p, y_p, z_p) with respect to the centroid of the points in the pillar. Finally, the output point-wise feature is $\mathbf{P_v} \in \mathbb{R}^{N \times 10}$, and the resulting size of the 3D voxel grid is $\left(\frac{W}{s_y}, \frac{H}{s_x}, \frac{D}{s_z}\right)$, where (s_y, s_x, s_z) gives the voxel sizes, and (W, H, D) are the ranges along the Y-axis, X-axis, Z-axis, respectively.

Point-wise Fusion. This module fuses the pointwise features $\mathbf{P_i}$ and $\mathbf{P_v}$. Since the dimensions of the two features are different, two fully connected (FC), one for each feature, are used to adjust their dimensions to be the same. There are two common fusion methods for ROIs: addition and concatenation. Therefore, this paper will analyze which fusion method is the most suitable for the pointwise features in Table 3 in the ablation section. After the fusion operation, one FC layer is utilized to further merge the fused features and output the result as $\mathbf{P_f}$.

B. VOXEL FEATURE ENCODER MODULE AND 3D BACKBONE

This section introduces the voxel feature encoder module and the 3D backbone, in that order.

Voxel Feature Encoder Module. Upon completing the pointwise fusion, the fused feature $\mathbf{P_f}$ is transformed through the VFE layer which is composed of a fully connected network (FCN), into a feature space, where information from the point features $\mathbf{f_i} \in \mathbb{R}^m$ can be aggregated to encode the shape of the surface contained within the voxel [7], [8], [17], where $i \in [1, N]$ and m is the feature dimension of a point. The FCN consists of a linear layer followed by a batch normalization layer, and a ReLU layer. An elementwise max-pooling process is used to locally aggregate the transformed features and output a feature $\vec{\mathbf{f}}$ for $\mathbf{P_f}$. Finally, the max-pooled feature $\vec{\mathbf{f}}$ is concatenated with each point feature $\mathbf{f_i}$ to generate the final feature $\mathbf{P_{vfe}}$. his work stacks two such VFE layers and both of the output lengths are 128. This means the shape of $\mathbf{P_{vfe}}$ is $N \times 128$. The details are shown in Figure 4.

3D Backbone. The 3D backbone takes the feature $\mathbf{P_{vfe}}$ and it's corresponding index of 3D coordinates $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ as inputs. The backbone is widely used in [30], [31] and has twelve 3D sparse convolutional layers and is divided into four stages according to feature resolution, as shown in Figure 5. The four-stage feature resolutions in the order of (W, H, D) are $(1600, 1408, 41)$, $(800, 704, 21)$, $(400, 352, 11)$, and $(200, 176, 2)$. Specifically, each stage has two kinds of 3D convolutional layers: the submanifold convolution [19] and the sparse convolution. The former does not generate new points and shares the point coordinate indices in each stage; hence, the submanifold convolution runs very fast. The latter is a sparse version of the dense 3D convolution. Usually, these two convolutions are used in conjunction to achieve the speed/accuracy balance. The details and numbers of input and output channels are illustrated in Figure 5. The sparse feature map after the 3D sparse convolution needs to be converted into the dense feature map $\mathbf{F_d} \in \mathbb{R}^{200 \times 176 \times 256}$. The detailed configuration is given in Table 1.

C. DETECTION HEAD

The input data of the detection head is the dense feature map $\mathbf{F_d}$. The detection head is comprised of three convolution blocks. Block 1 has five 2D convolutional layers and outputs the feature map $\mathbf{F_1} \in \mathbb{R}^{100 \times 88 \times 128}$. Similarly,

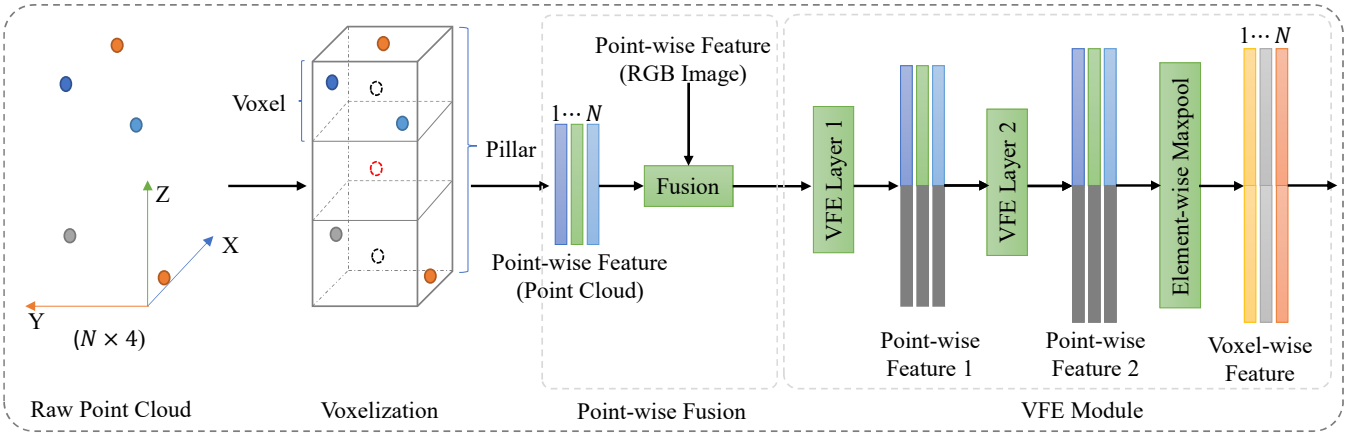

Figure 4: The details for the voxelization, pointwise fusion, and VFE module. The black circle and red circle are the centroid of a voxel and a pillar, respectively. The black circle and red circle are only for demonstration and are all virtual. To understand the figure more clearly, the number of points N only takes the value of five.

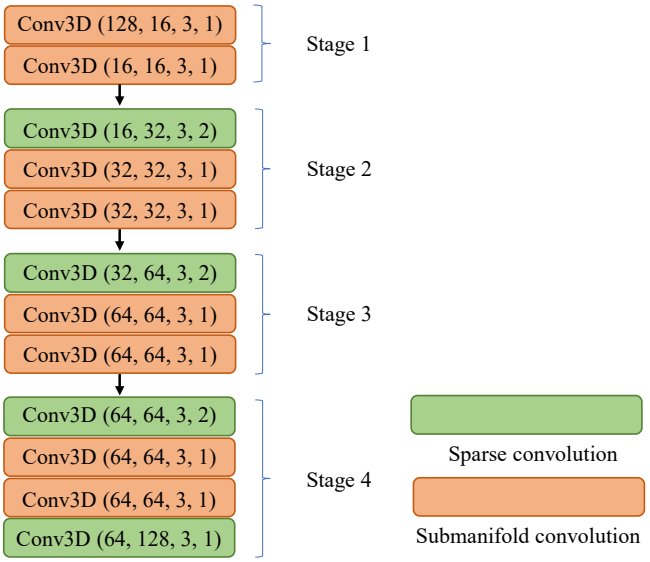

Figure 5: The 3D backbone architecture. Conv3D (cin, cout, k, s) denotes a convolutional block, where the parameters cin, cout, k, and s represent the input-channel numbers, the output-channel numbers, the kernel size, and the stride, respectively. Each block consist of a 3D convolutional layer followed by a batch normalization layer and a ReLU layer.

block 2 also has five 2D convolutional layers and takes the feature map $\mathbf{F_1}$ as input and outputs the feature map $\mathbf{F_2} \in \mathbb{R}^{50\times44\times256}$. Block 3 has two transpose layers and one 2D convolutional layer. $\mathbf{F_1}$ and $\mathbf{F_2}$ are transposed as the feature map $\mathbf{F_3} \in \mathbb{R}^{100\times88\times256}$ and the feature map $\mathbf{F_4} \in \mathbb{R}^{100\times88\times256}$, respectively. Finally, the feature maps $\mathbf{F_3}$ and $\mathbf{F_4}$ are concatenated as the feature map $\mathbf{F} \in \mathbb{R}^{100\times88\times512}$. The feature map $\mathbf{F}$ is mapped to three desired learning targets: (1) a classification score map $\mathbf{F_{score}} \in \mathbb{R}^{100\times88\times18}$, (2) a box regression map $\mathbf{F_{box}} \in \mathbb{R}^{100\times88\times42}$, and (3) a direction regression map $\mathbf{F_{dir}} \in \mathbb{R}^{100\times88\times12}$. The detailed configuration is given in Table 1.

Table 1: The network configuration for the 3D backbone and the detection head. The output sizes (W, H, Depth) and (W, H, Channel) are for the 3D backbone and the detection head, respectively. The structure $[type, size, stride] \times Number$ represents the convolutional type, filter size, stride, and the number of layers.

Network	Output Size	Name	Layer
3D Backbone	(1600, 1408, 41)	Stage1	S_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(800, 704, 21)	Stage2	S_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(400, 352, 11)	Stage3	S_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(200, 176, 2)	Stage4	S_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
Detection Head	(100, 88, 128)	Block1	[Conv2D, 3, s1]x4 Conv2D, 3, s2
	(50, 44, 256)	Block2	[Conv2D, 3, s1]x4 Conv2D, 3, s1
	(100, 88, 512)	Block3	DeConv2D, 3, s1 DeConv2D, 3, s2 Conv2D, 3, s1

D. LOSS FUNCTION

This work utilizes the same loss functions in PointPillars [20] and SECOND [19]. The 3D ground truth boxes and anchors are parameterized as $(x, y, z, l, w, h, \theta)$, where (x, y, z) denote the box's center, (l, w, h) represent the box's size, and θ is the yaw rotation around the Z-axis. The corresponding regression residuals between the 3D anchors and ground truth

are defined as follows:

$$\begin{aligned} \Delta x &= \frac{x^g - x^a}{d^a}, \quad \Delta y = \frac{y^g - y^a}{d^a}, \\ \Delta h &= log(\frac{h^g}{h^a}), \quad \Delta\theta = sin(\theta^g - \theta^a), \end{aligned} \tag{4}$$

where the superscripts g and a represent the ground truth box and the anchor, respectively. The variable $d^a = \sqrt{(w^a)^2 + (l^a)^2}$ is the diagonal of the base of the anchor box.

The regression loss function is as follows:

$$\mathcal{L}_{reg} = \sum_{b} Smooth_{\mathcal{L}1}(\Delta b), \tag{5}$$

where the input dimensions are $b \in (x, y, z, w, l, h, \theta)$ and $Smooth_{\mathcal{L}1}$ is the smooth $\mathcal{L}1$ loss function in the Fast R-CNN module.

Since the yaw angle $\theta \in [-\Pi, \Pi]$ has two directions $\{+, -\}$, and the angle regression loss cannot distinguish the directions. A softmax classification loss is utilized to compute the discretized direction loss [19], $\mathcal{L}_{dir}$. If the yaw angle θ around the Z-axis of the ground truth is greater than zero, the direction is positive; otherwise, the direction is negative.

For the object classification loss, the focal loss [32] is used:

$$\mathcal{L}_{cls} = -\alpha_a(1 - p^a)^\gamma log(p^a), \tag{6}$$

where p^a is the class probability of an anchor, $\alpha = 0.25$, and $\gamma = 2$. The total loss can be formulated as follows:

$$\mathcal{L}oss = \frac{1}{N_{pos}}(\beta_1\mathcal{L}_{box} + \beta_2\mathcal{L}_{cls} + \beta_3\mathcal{L}_{dir}), \tag{7}$$

where N_{pos} is the number of positive anchors and $\beta_1 = 2.0$, $\beta_2 = 1.0$, and $\beta_3 = 0.2$. For the car class, an anchor is defined as positive if it has a 2D IoU greater than 0.60 (pedestrian/cyclist is 0.35) with its paired ground truth. If it has a 2D IoU less than 0.45 (pedestrian/cyclist is 0.2), the anchor is labeled as negative. The other anchors are ignored when computing the loss.

IV. EXPERIMENTS

This section introduces the dataset, the experimental settings, and the results in detail.

A. DATASET

The proposed model is trained and evaluated on the KITTI dataset [18]. The KITTI object dataset possesses 7,518 testing frames and 7,481 training frames. Each frame is comprised of a point cloud, stereo RGB images (the left image and the right image), and calibration data. In this research, only a point cloud and the left image with their calibration data are used. To impartially compare the proposed approach with existing methods, the training dataset is divided into two subsets (training subset and validation subset) based on the same criteria, and the ratio of the two subsets is 1:1.

For KITTI's criteria, according to the size, truncation, and occlusion classes of objects, all objects are grouped into three difficulty classes: easy (E), moderate (M), and hard (H). Before October 8th, 2019, KITTI's object detection metric was defined as the 11-point average precision (AP) metric. Since then, the metric has been defined by 40 recall positions. Compared with the 11-point AP, the 40-point AP more properly assesses the quality of an algorithm based on the infinite approximation. Intersection-over-Union (IoU) is the generic evaluation criterion for object detection. In the evaluation of 2D, 3D, and bird's eye view (BEV) detection, the IoU is at the threshold of 0.7 for the car class and 0.5 for the pedestrian/cyclist class. For the average orientation similarity (AOS) we follow the approach in [18] and define the AOS as:

$$AOS = \frac{1}{N} \sum_{r \in \{0, 0.1, \cdots, 1\}} \max_{\tilde{r}:\tilde{r} \geq r} s\left(\tilde{r}\right), \tag{8}$$

$$s\left(r\right) = \frac{1}{|D\left(r\right)|} \sum_{D(r)} \frac{1 + cos\Delta_\theta^i}{2} \delta_i, \tag{9}$$

where $N \in \{11, 40\}$, $r = \frac{TP}{TP+FN}$ is the PASCAL object detection recall, TP means the true positive, FN is the false negative, s is the orientation similarity, $D\left(r\right)$ represents the set of all object detections at recall, and Δ_θ^i is the difference in angle between estimated and ground truth orientation of detection i, $\delta \in \{0, 1\}$ is the penalty factor.

B. EXPERIMENTAL SETTINGS

The proposed model is an end-to-end 3D detector for three classes: the car, pedestrian, and cyclist. When designing the anchors for the three classes, different classes employ different sizes (w, l, h). The sizes $(1.6, 3.9, 1.56)$, $(0.6, 0.8, 1.73)$, and $(0.6, 1.76, 1.73)$ are for the car, the pedestrian, and the cyclist, respectively. Note that each anchor has two directions $\{0°, 90°\}$, which means that each location has six anchors. The detection area in the point cloud is $\{(x, y, z) \mid x \in [0, 70.4], y \in [-40, 40], z \in [-3, 1]\}$.

The framework is based on Pytorch and programmed by the python language. This model is trained from scratch based on Adam optimizer. The whole network is trained with a batch of size 10 and the initial learning rate is 0.003 for 80 epochs on one TITAN RTX GPU. This work also adopts the cosine annealing learning rate for the learning rate decay. The entire training time is around 12 hours.

For data augmentation, this work employs the widely used augmentations found in [7], [19], [20], including global scaling $[0.95, 1.05]$, global rotation around the Z-axis $[-45°, 45°]$, and the random flipping along the X-axis.

C. RESULTS

Most of the LiDAR-camera-based methods only provide the results in the KITTI validation dataset for three classes, hence, this work first compares the results in the validation dataset. In addition, for the car class, this paper also compares the results based on the KITTI testing dataset.

Table 2: Performance comparison using the KITTI validation dataset. The results are evaluated by the mean Average Precision with 11 recall positions. For easy understanding, the top result is highlighted in bold for each column in each class and the second best is shown in blue. I and L denote the RGB image and LiDAR, respectively.

Class	Method	Publish Year	Input	FPS	3D Performance (%)				BEV Performance (%)			
					Easy	Moderate	Hard	mAP	Easy	Moderate	Hard	mAP
Car	VoxelNet [7]	2017	L	4.3	81.97	65.46	62.85	70.15	89.60	84.81	78.57	84.32
	SECOND [19]	2018	L	20.0	87.43	76.48	69.10	77.67	89.96	87.07	79.66	85.56
	PointRCNN [33]	2018	L	10.0	88.88	78.63	77.38	81.63	-	-	-	-
	F-PointRCNN [34]	2019	L	15.4	89.12	79.00	77.48	81.87	90.12	88.10	86.24	88.15
	MV3D [5]	2017	I+L	2.8	71.29	62.68	56.56	63.51	86.55	78.10	76.67	80.44
	F-PointNet [22]	2017	I+L	5.9	83.76	70.92	63.65	72.78	88.16	84.02	76.44	82.87
	PC-CNN [35]	2018	I+L	2.0	57.63	51.74	51.39	53.59	83.61	77.36	69.61	76.86
	AVOD [6]	2018	I+L	12.5	83.11	74.02	67.84	74.99	-	-	-	-
	AVOD-FPN [6]	2018	I+L	10.0	84.41	74.44	68.65	75.83	89.37	86.09	79.13	84.86
	ContFusion [23]	2018	I+L	16.7	86.32	73.25	67.81	75.79	**95.44**	87.34	82.43	**88.40**
	MVX-Net [17]	2019	I+L	6.7	85.50	73.30	67.40	75.40	89.50	84.90	79.00	84.47
	MCF3D [24]	2019	I+L	6.3	84.11	75.19	74.23	77.84	88.82	86.11	79.31	84.75
	TAO3D [10]	2020	I+L	9.0	85.12	76.23	74.46	78.60	89.64	86.23	85.60	87.16
	KDA3D [36]	2020	I+L	7.7	**88.45**	**78.85**	**78.46**	**81.92**	90.18	**87.69**	**86.93**	88.27
	Proposed	-	I+L	**17.8**	88.04	77.60	76.23	80.62	89.75	86.97	85.42	87.38
Ped.	VoxelNet [7]	2017	L	4.3	57.86	53.42	48.87	53.38	65.95	61.05	56.98	61.33
	AVOD-FPN [6]	2018	I+L	10.0	-	58.80	-	-	-	-	-	-
	F-PointNet [22]	2018	I+L	5.9	**70.00**	61.32	53.59	61.64	**72.38**	66.39	59.57	66.11
	MCF3D [24]	2019	I+L	7.7	68.54	**64.93**	**59.47**	**64.31**	68.56	64.98	59.55	64.36
	KDA3D [36]	2020	I+L	8.3	63.34	60.12	54.36	59.27	70.21	**67.34**	**61.24**	**66.26**
	Proposed	-	I+L	**17.8**	66.65	60.49	54.51	60.55	71.67	64.22	61.03	65.64
Cyc.	VoxelNet [7]	2017	L	4.3	67.17	47.65	45.11	53.31	74.41	52.18	50.49	59.02
	AVOD-FPN [6]	2018	I+L	10.0	-	49.70	-	-	-	-	-	-
	F-PointNet [22]	2018	I+L	5.9	77.15	56.49	53.37	62.34	**81.82**	60.03	56.32	66.06
	MCF3D [24]	2019	I+L	7.7	**78.18**	51.06	50.43	59.89	78.18	51.09	50.45	59.91
	KDA3D [36]	2020	I+L	8.3	77.19	57.43	54.56	63.06	79.39	60.31	57.14	65.61
	Proposed	-	I+L	**17.8**	75.87	**60.07**	**55.87**	**63.94**	81.03	**63.50**	**61.06**	**68.53**

Table 3: Performance comparison using the KITTI testing dataset. The results of cars are evaluated by the mean Average Precision with 40 recall positions. The top performance is highlighted in bold only for the mAP columns and FPS column, and the second-best is shown in blue.

Method	FPS	AP_{BEV} (IoU = 0.7)				AP_{3D} (IoU = 0.7)				$AP_{2D}(IoU = 0.7)$			
		Easy	Moderate	Hard	mAP	Easy	Moderate	Hard	mAP	Easy	Moderate	Hard	mAP
MV3D [5]	2.8	86.00	76.90	68.50	77.13	71.10	62.40	55.10	62.87	96.47	90.83	78.63	88.64
F-PointNet [22]	5.9	88.70	84.00	75.30	82.67	81.20	70.40	62.20	71.27	95.85	95.17	85.42	**92.15**
AVOD [6]	12.5	86.80	85.40	77.70	83.30	73.60	65.80	58.40	65.93	95.17	89.88	82.83	89.29
AVOD-FPN [6]	10.0	88.50	83.80	77.90	83.40	81.90	71.90	66.40	73.40	94.70	88.92	84.13	89.25
ContFusion [23]	16.7	94.07	85.35	75.88	**85.10**	83.68	68.78	61.67	71.38	-	-	-	-
MVX-Net [17]	6.7	89.20	85.90	78.10	84.40	83.20	72.70	65.20	73.70	-	-	-	-
Proposed	**17.8**	89.61	85.08	80.42	85.04	81.11	72.93	67.24	**73.76**	95.37	92.15	87.54	91.69

This work achieves competitive results compared with other state-of-the-art methods, the details are illustrated in Table 2. The results are mainly compared with the LiDAR and RGB camera-based methods. Usually, the LiDAR-based methods run much faster than the LiDAR-camera-based approaches. To show the superiority of the proposed model in speed, the classic LiDAR-based methods are also listed in Table 2. As can be seen, the proposed model mainly competes with MCF3D [22] and KDA3D [36] in comprehensive performance. For the cyclist class, the proposed model outperforms the KDA3D [36]. In the car class, our model is slightly inferior to the KDA3D [36]. However, the speed of our model runs 2× faster than KDA3D. Note that the proposed model is an end-to-end multi-class detector, however MCF3D [24] and KDA3D [36] train two models for the car class, and the pedestrian/cyclist classes, respectively. F-PointNet [22] is actually a LiDAR-based method that utilizes the location of the object in the 2D RGB image to quickly guide the model convergence.

The proposed model is also evaluated using the more challenging dataset: the KITTI testing dataset. In Table 3, this part only compares the proposed method with state-of-the-art methods in three aspects: BEV, 3D, and 2D. Since it requires a great deal of data to compare these three performances, here, the results are simply compared based on the mean average precision (mAP). For the 3D performance, the proposed model has the best performance. For the BEV and 2D performances, the proposed method is the second-best, but the overall performance of the proposed method outperforms state-of-the-art methods when taking accuracy and speed into account. The results of the proposed method can be retrieved on the KITTI website based on the name of the proposed method, PFF3D.

Figure 6 presents some qualitative results. As can be seen

Table 4: Effect of the point feature fusion module. The results are from the 'Moderate' difficulty category. The best result is highlighted in bold for each column.

Method			Cars (%)				Pedestrians (%)				Cyclists (%)			
Addition	Concatenation	FC	2D	AOS	BEV	3D	2D	AOS	BEV	3D	2D	AOS	BEV	3D
			89.27	88.72	85.04	77.00	70.69	33.91	62.74	56.20	64.13	59.55	59.19	55.71
√			**89.74**	**89.39**	85.84	76.60	70.26	56.75	62.80	57.65	64.64	58.40	61.34	59.72
	√		89.54	85.84	**89.14**	77.01	69.99	56.59	62.71	57.74	65.19	62.08	61.74	**60.51**
√		√	89.72	89.29	86.97	**77.60**	**71.87**	**60.20**	64.22	**60.16**	**67.21**	**63.95**	**63.50**	60.07
	√	√	89.70	89.27	86.27	77.12	69.51	58.02	**65.89**	59.74	64.74	61.50	60.87	59.37

Table 5: Effect of the proposed framework. The 'Time' column denotes the training time and the 'Memory' is the memory needed when the model is run for four batch sizes. The 'Rtime' column denotes the runtime. 'R' and 'V1' represent ResNet and ResNetV1d, respectively. '2D Image Branch' denotes that if the model use a full 2D image detection branch. The results of the cars are in the 'Moderate' difficulty category for the BEV and 3D.

2D Image Branch	Method	Time (hour)	Memory (MB)	Rtime (FPS)	BEV (%)	3D (%)
Yes	R101	28.0	19,500	9.5	85.95	76.82
	R50	23.5	12,550	11.0	86.29	76.92
	V1-50	25.0	12,700	10.6	85.51	76.48
	VGG11	16	11900	12.0	85.96	76.44
No	Ours	**11.5**	**4200**	**17.8**	86.17	**76.93**

in the figures, each object can be detected by the proposed model and the predicted bounding boxes are well-matched with their corresponding ground truth boxes. Even in very complex scenes, the proposed model can detect objects quite well, as shown in the last two rows of Figure 6.

V. ABLATION STUDIES

This section analyzes the proposed methods individually by conducting ablation experiments using the KITTI validation dataset.

1) Effect of the Point Feature Fusion Module

This section analyzes the point feature fusion module based on the three classes in detail. In Table 4, the 'Addition' and 'Concatenation' represent the respective addition and concatenation fusion methods. The parameter'FC' means the fully connected layer followed after the fusion operation, as shown in Figure 3. The experimental results show that the combination of the addition operation and FC of the proposed module is best for the three classes: the car, pedestrian, and cyclist. The data in the first row give the results of the proposed method when only taking a point cloud as input. Compared with the LiDAR-based method (the first row), the proposed method (the fourth row) achieves 0.45%, 0.57%, 1.83%, and 0.6% gains in the 2D, AOS, BEV, and 3D performance, respectively. Compared with the performance improvement of cars, the proposed model is more helpful for improving the identification of pedestrians and cyclists.

2) Effect of the Proposed Framework

The proposed 3D object detection framework is the first to directly project the raw RGB point features to a point cloud, as shown in Figure 2. The proposed approach is not without precedent but was discovered through experiments. Inspired by MVX-Net [17], we simply wanted to implement a lightweight design based on two backbones. One backbone was intended for 2D detection and the other one for 3D detection. First, ResNet-101 [37] was chosen as the backbone to extract features from RGB images. The results were as expected but the testing model ran very slowly. Then, ResNet-101 was replaced by ResNet-50 [37], and the model ran a little faster but the accuracy was almost the same. When using ResNetV1d-50 [38], the result was almost the same as the result for ResNet-50 [37]. These results are thought-provoking. Hence, we boldly propose to map the raw point features of the RGB image to the point cloud without the 2D detection branch. The experimental results in Table 5 demonstrate that the proposed method is feasible. As can be seen in Table 5, the proposed approach not only drastically reduces the memory requirements for model operation, but also reduces the time of model training by half. It can be said that the proposed framework is lightweight, memory-saving, and energy-saving.

3) Effect of RGB$^+$

Table 6: Effect of RGB$^+$. The results of the car class are in the 'Moderate' difficulty category.

Class	Input	2D (%)	AOS (%)	BEV (%)	3D (%)
Car	RGB	89.67	89.25	86.62	77.48
	RGB^I	89.51	88.90	86.27	76.92
	RGB^D	**89.72**	**89.29**	**86.97**	**77.60**
Ped.	RGB	70.38	46.09	**65.89**	58.99
	RGB^I	69.75	54.80	65.66	58.91
	RGB^D	**72.42**	**60.64**	64.22	**60.16**
Cyc.	RGB	62.19	58.33	57.83	55.53
	RGB^I	**67.52**	62.19	62.87	**61.37**
	RGB^D	67.21	**63.95**	**63.50**	60.07

The RGB$^+$ construct includes two representations: the RGB$^{\mathbf{I}}$ and RGB$^{\mathbf{D}}$. In Table 6, there are three sets of experiments each for the car, pedestrian, and cyclist. The variable for each set of experiments is the input image. For the car class, the results of RGB$^{\mathbf{D}}$ outperform the results of RGB and RGB$^{\mathbf{I}}$ in all aspects. For the pedestrian and cyclist classes, the results of RGB$^{\mathbf{D}}$ surpasses both the results of RGB and

Figure 6: Qualitative results of the proposed method using the KITTI validation dataset. In the RGB images, the red, cyan, and yellow color represent the predictions for the car, pedestrian, cyclist, respectively. In the point cloud images, the green color denotes the ground truth, and the red color represents the prediction. The results in the point cloud images are used for a qualitative comparison.

$\text{RGB}^{\mathbf{I}}$ in some aspects. Hence, the $\text{RGB}^{\mathbf{D}}$ image is beneficial for improving 3D object detection.

VI. CONCLUSION

This paper is the first to propose a lightweight, memory-saving, and energy-saving framework for 3D object detection based on LiDAR and an RGB camera. Different from the existing frameworks, the proposed framework only employs one backbone to extract features from a point cloud and RGB image. The framework benefits from the proposed module, i.e., the point feature fusion module. The fusion module directly extracts the point features of RGB images and fuses them with the corresponding point cloud features. The experimental results using both the KITTI validation dataset and testing dataset demonstrate that the proposed method significantly improves the speed (17.8 FPS) of LiDAR-camera-based 3D object detection compared with other state-of-the-art approaches. Note that the proposed native model can achieve an inferring speed 17.8 FPS.

In the future, the proposed method will be directly used in the point-based methods [13], thereby achieving breakthroughs in both accuracy and speed.

References

[1] D. Hong, L. Gao, N. Yokoya, J. Yao, J. Chanussot, Q. Du, and B. Zhang, "More diverse means better: Multimodal deep learning meets remote-sensing imagery classification," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–15, 2020.

[2] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–13, 2020.

[3] X. Wu, D. Hong, J. Chanussot, Y. Xu, R. Tao, and Y. Wang, "Fourier-based rotation-invariant feature boosting: An efficient framework for geospatial object detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 2, pp. 302–306, 2020.

[4] X. Wu, W. Li, D. Hong, J. Tian, R. Tao, and Q. Du, "Vehicle detection of multi-source remote sensing data using active fine-tuning network," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 167, pp. 39–53, 2020.

[5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6526–6534.

[6] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and 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)*, 2018, pp. 1–8.

[7] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[8] L. Wen and K.-H. Jo, "Fully convolutional neural networks for 3d vehicle detection based on point clouds," in *Intelligent Computing Theories and Application*, 2019, pp. 592–601.

[9] L. Wen and J. Kang-Hyun, "Lidar-camera-based deep dense fusion for robust 3d object detection," in *Intelligent Computing Methodologies*. Springer International Publishing, 2020, pp. 133–144.

[10] L. Wen and K. H. Jo, "Three-attention mechanisms for one-stage 3d object detection based on lidar and camera," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[11] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[12] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5099–5108.

[13] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 037–11 045.

[14] L. Wen, X. T. Vo, and K.-H. Jo, "3d saccadenet: A single-shot 3d object detector for lidar point clouds," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 2020, pp. 1225–1230.

[15] W. Shi and R. Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1708–1716.

[16] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 244–253.

[17] V. A. Sindagi, Y. Zhou, and O. Tuzel, "Mvx-net: Multimodal voxelnet for 3d object detection," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7276–7282.

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

[19] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct 2018.

[20] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 689–12 697.

[21] M. P. Muresan, S. Nedevschi, and I. Giosan, "Real-time object detection using a sparse 4-layer lidar," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2017, pp. 317–322.

[22] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.

[23] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Computer Vision – ECCV 2018*, 2018, pp. 663–678.

[24] J. Wang, M. Zhu, D. Sun, B. Wang, W. Gao, and H. Wei, "Mcf3d: Multi-stage complementary fusion for multi-sensor 3d object detection," *IEEE Access*, vol. 7, pp. 90 801–90 814, 2019.

[25] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7337–7345.

[26] J. Yoo, Y. Kim, J. Kim, and J. Choi, "3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection," in *Computer Vision – ECCV 2020 - 16th European Conference, 2020, Proceedings*, 2020, pp. 720–736.

[27] M. H. Daraei, A. Vu, and R. Manduchi, "Velocity and shape from tightly-coupled lidar and camera," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 60–67.

[28] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 2017–2025.

[29] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, "End-to-end multi-view fusion for 3d object detection in lidar point clouds," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 100, 30 Oct–01 Nov 2020, pp. 923–932.

[30] C. He, H. Zeng, J. Huang, X. S. Hua, and L. Zhang, "Structure aware single-stage 3d object detection from point cloud," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 870–11 879.

[31] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 526–10 535.

[32] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[33] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 770–779.

[34] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point r-cnn," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9774–9783.

[35] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A general pipeline for 3d detection of vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3194–3200.

[36] J. Wang, M. Zhu, B. Wang, D. Sun, H. Wei, C. Liu, and H. Nie, "Kda3d: Key-point densification and multi-attention guidance for 3d object detection," *Remote Sensing*, vol. 12, no. 11, p. 1895, 2020.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[38] K. He, X. Zhang, S. Ren, e. B. Sun, Jian", J. Matas, N. Sebe, and M. Welling, "Identity mappings in deep residual networks," in *Computer Vision – ECCV 2016*, 2016, pp. 630–645.

LI-HUA WEN (S'15) received a bachelor's degree in vehicle engineering from the School of Automotive Engineering, Shanghai University of Engineering Science, Shanghai, in 2015. He is currently working toward the Ph.D. degree in electrical and computer engineering with the Graduate School of Electrical Engineering, University of Ulsan, Ulsan, South Korea.

Since 2013, he was an engineer with Shanghai Automobile Gear Works, Commercial Aircraft Corporation of China, Ltd., and Hyundai Commercial Vehicle (China) Company, Ltd, Ziyang, China. His research interests include image processing, pattern recognition, computer vision, machine learning, and 3-D object detection for intelligent vehicles.

KANG-HYUN JO (Senior Member, IEEE) received a Ph.D. degree in computer controlled machinery from Osaka University, Osaka, Japan, in 1997. After a year of experience with Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea, as a Postdoctoral Research Fellow, he joined the School of Electrical Engineering, University of Ulsan, Ulsan, South Korea. He is currently serving as the Faculty Dean with the School of Electrical Engineering, University of Ulsan, Ulsan, South Korea. His research interests include computer vision, robotics, autonomous vehicle, and ambient intelligence.

Dr. Jo has served as the Director or an AdCom Member with the Institute of Control, Robotics and Systems, The Society of Instrument and Control Engineers, and the IEEE IES Technical Committee on Human Factors Chair, AdCom Member, and the Secretary until 2019. He has also been involved in organizing many international conferences, such as International Workshop on Frontiers of Computer Vision, International Conference on Intelligent Computation, International Conference on Industrial Technology, International Conference on Human System Interactions, and the Annual Conference of the IEEE Industrial Electronics Society. He is currently an Editorial Board Member for international journals, such as the International Journal of Control, Automation, and Systems and Transactions on Computational Collective Intelligence.