IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 November 2023, accepted 26 December 2023, date of publication 4 January 2024,
date of current version 10 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3349614

RESEARCH ARTICLE

G-Fusion: LiDAR and Camera Feature Fusion on the Ground Voxel Space

SHUAI CHENG[1], ZUOTAO NING[1], JUN HU[1], JIAXIN LIU[1,2], WENXING YANG[1,2], LUYANG WANG[1,3], HONGFEI YU[1], AND WEI LIU[1,2]

[1]Neusoft Reach Automotive Technology (Shenyang) Company Ltd., Shenyang 110179, China
[2]School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
[3]Software College, Northeastern University, Shenyang 110819, China

Corresponding author: Wei Liu (lwei@reachauto.com)

This work was supported in part by the Intelligent Control Theories and Key Technologies of Heterogeneous Unmanned Electric Commercial Vehicles Formation under Grant U22A2043, and in part by the Development and Application of Vehicle Cloud Collaboration Self-Evolution Platform for Advanced Automatic Driving under Grant 2022JH1/10400030.

ABSTRACT Detection based on LiDAR and camera fusion is increasingly popular for researchers in the autonomous driving domain. Compared to the camera-only and LiDAR-only methods, the fusion-based methods indeed improve the detection accuracy on public-available datasets. However, due to the complexity of the projection or fusion mechanism, few of these methods can run in real time even on an advanced desktop GPU. Thus, in this paper, we propose a new fusion detection model G-Fusion with a light and fast image view-transform module. According to our receptive field analysis of image feature maps, we directly project image features to only one voxel layer located on the ground, then fuse the LiDAR and image features by concatenation and convolution. With this delicately designed module, G-Fusion greatly boosts the state-of-the-art speed performance on the nuScenes dataset, achieving a good balance with the competitive detection scores. Meanwhile, since the precision of sensor extrinsic parameters is important for most fusion-based methods, we also deeply dig into our model's calibration error tolerance ability and discover the failure noise condition.

INDEX TERMS Autonomous driving, LiDAR, camera, fusion, 3D detection, deep-learning.

I. INTRODUCTION

Perception is essential for an autonomous driving vehicle, especially the 3D detection module that can accurately locate and categorize obstacles in the surrounding area [1], and this module also plays an important role as an upstream module of the planning and control system. Camera and LiDAR are the two kinds of sensors commonly equipped on a self-driving car for perception functions [2]. So, in current research, 3D detection methods usually use these two types of sensors. The camera, as a 2D sensor, can capture rich shape, color, and texture information, which is suitable for semantic information extraction and obstacle classification [3]. However, since the camera misses the 3D depth information, image-based 3D detection always suffers from imprecise location and size regression [4], [5]. While LiDAR collects 3D information and generates point clouds with 3D coordinates and reflection intensity [6]. However, due to the sparsity of the point cloud, LiDAR-based 3D detection is often relatively poor for distant targets [6].

Therefore, LiDAR-Camera fusion is attracting more and more attention for developing more accurate and robust 3D detection methods. The key to this task is to fuse 2D and 3D information effectively and efficiently. Existing fusion methods can be roughly classified into the following three types: proposal-level fusion [7], [8], [9], point-level fusion [10], [11], [12], [13], and mid-level fusion [14], [15], [16], [17], [18]. The proposal-level fusion methods, such as [7], [8], and [9], extract image features by 2D backbones and generate 2D proposals which are then mapped into 3D space. Afterwards, the proposals are refined by the inside point cloud features. However, the 2D proposal usually contains a lot of background noise that causes plenty of wrong detection [19]. To avoid background noise, point-level fusion

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tan.

methods enrich the point cloud data by image segmentation results or project the points onto image pixels. For example, [10] and [20] project the segmentation result into point clouds, [11] uses image segmentation result to separate 3D point clouds, [12], [13] map the point clouds onto an image to enhance the image features. However, according to [18], point-level fusion methods suffer from inaccurate image segmentation results while projecting images to point clouds or missing the geometric information when projecting point clouds to images, as well as the noise of calibration parameters for projection [19]. Thus, [18], as a middle-level fusion method, concatenates the BEV feature maps extracted from both the camera and LiDAR sensors followed by a bird-eye-view (BEV) encoder, which retains both geometric structure and semantic information. This method further boosts the 3D detection state-of-the-art performance, but we consider that its inference speed is relatively slow because the method cannot run in real-time on an advanced desktop GPU (NVIDIA® GeForce® RTX 3090). So, in this work, we propose a new LiDAR-camera fusion 3D detection network G-Fusion with a new view-transform module which is more direct and light.

Similar to BEVFusion [18], we also align the multi-sensory features in the BEV latent level, since this way is capable of maintaining both the geometric structure of LiDAR and the semantic density of the camera [18]. Inspired by the image 3D-detection method FastBEV [21], we propose a light and fast feature projection approach for image view transformation. Specifically, we generate a ground-voxel space catching the image features aligned with the voxel coordinate through LiDAR-camera extrinsic and camera intrinsic matrices. With the help of this module, our method runs two times faster than BEVFusion and three times faster than TransFusion, and the detection accuracy of G-Fusion is also decent on the nuScenes [22] detection task.

In addition, since our view-transform module depends on the calibration extrinsic parameters, to explore the failure boundary, we conduct experiments that add translation and rotation disturbance to the extrinsic parameters, and the experiment is detailed in Sec. V-B.

The main contributions of the G-Fusion are listed below:

- A new view-transform module with a projection mask is designed to faster fuse the image feature to the LiDAR BEV feature space.
- A novel LiDAR-Camera fusion 3D detection method G-Fusion is proposed. The method is light and fast, also with competitive performance on the nuScenes [22] detection task.
- The calibration noise tolerance ability is fully analyzed to verify the usability of the proposed model in real driving scenarios.

II. RELATED WORKS

A. CAMERA-ONLY-BASED 3D DETECTION METHODS

LiDAR point clouds provide accurate depth information but lack the rich semantic content found in images [19]. In addition, due to the high cost of LiDAR [23], many researchers have focused on improving 3D target detection by using images as inputs [24]. They achieve this by either applying a network to learn depth information [25] or employing a multi-view transformation technique to convert images into stereoscopic representations like BEVs or 3D voxels.

LSS [26] uses a depth estimation network to extract implied depth information, generating a depth probability distribution. It projects features pixel by pixel onto the predicted depth distribution, creating camera-truncated cones. These cones are compressed vertically along the ground direction and converted into BEV features, mapping 2D features to BEV space. BEVDet [27] modularizes multi-camera 3D target detection. Specifically, it uses ResNet [28] and Swin Transformer [29] as backbone models, projecting image features into the BEV space using LSS, combining features of different resolutions in the BEV space, and employing the first-stage 3D target detection head of CenterPoint [30].

DETR3D [31] utilizes the transformer structure, decoding each BEV query as a 3D reference point. Then DETR3D transforms the reference points into the image space and extracts 2D features from relevant image locations for object query refinement. For the training process, it uses set-to-set loss. BEVFormer [32] is also based on transformer. It uses spatial and temporal self-attention to allow BEV queries to aggregate multi-view features and extract temporal information, which improves the estimation speed and the target detection that is heavily occluded.

FCOS3D [33] presents a target detection framework. It translates a 7-degree-of-freedom 3D target into the image domain, detects objects of various sizes through feature pyramid network (FPN) layers, and employs different prediction heads for 3D box prediction based on different attributes. Fast-BEV [21] leverages temporal information through multi-frame feature fusion and introduces the fast-ray transform to quickly transfer image features to the BEV space. Its efficient BEV encoder speeds up on-vehicle inference, proposing a benchmark on an on-vehicle chip.

B. LiDAR-ONLY-BASED 3D DETECTION METHODS

In nowadays autonomous driving systems, LiDAR serves as a commonly used sensor, generating point cloud data known for its characteristics of disorder, sparsity, and irregularity [34]. To streamline data processing, researchers have traditionally converted this point cloud data into either regular 3D voxels [6] or collections of images [35]. However, this practice often results in unwarranted data inflation and potential loss of valuable information [36].

PointNet [35] introduces a groundbreaking deep neural network that offers a unified approach to various 3D recognition tasks. It takes the number of original point clouds and the 3D information of each point as input, preserving the critical alignment invariance of the input data. This method paves the way for the DeepLearning-based point cloud

perception approaches. PointRCNN [37] adopts a two-stage detection methodology for 3D target detection from original point clouds. In the first stage, the network segments the original point cloud into foreground and background points, learning point-by-point features, and simultaneously generating high-quality 3D proposals from the foreground points. In the second stage, a sub-network converts each proposed set of points to canonical coordinates, refining local spatial features. These features are combined with the global semantic features of each point learned in the first stage for precise box refinement and confidence prediction.

Vote3deep [38] presents a unified strategy for rasterizing point clouds into 3D voxels. During this process, each voxel cell is assigned features extracted or designed manually. A voting-based method then estimates the centre and orientation of the point cloud within each voxel, generating candidate bounding boxes. These candidates undergo classification to identify target objects and fine localization for precise target location and shape determination. This method simplifies point cloud analysis but necessitates manual feature selection and design for specific scenarios, potentially overlooking complex point cloud structures. VoxelNet [6] directly processes sparse 3D point clouds by subdividing 3D space into equal-sized voxels and utilizing the network to learn features from nonempty voxels. This approach eliminates the need for labour-intensive handcrafted feature representations, efficiently capturing 3D shape information. However, the substantial computational demands of VoxelNet pose challenges for real-time applications. To address this, SECOND [39] applies improved sparse convolution to radar-based networks, greatly enhancing training and inference speed. Additionally, SECOND introduces a novel data augmentation technique: randomly injecting a Ground Truth (GT) point cloud into the training data, accompanied by collision testing. This approach significantly accelerates convergence speed and performance. Nevertheless, SECOND retains the costly 3D convolution layer.

PointPillars [40] transforms raw point cloud data into a stacked column tensor and a column index tensor. From these stacked columns, the encoder facilitates the learning of features to be reassembled into a 2D pseudo-image. This enables subsequent processing with 2D convolution, achieving further computational efficiency. In the case of CenterPoint [30], the first stage employs a standard 3D backbone to extract map-view features from radar point clouds. It then employs a 2D CNN detection head to locate the object centre and regress the 3D box using center point features. The second stage extracts a point feature from the 3D centre of each face of the predicted 3D box, concatenating these features into an MLP for box refinement and confidence score prediction.

3DETR [41] is an end-to-end trainable transformer that processes point clouds. In its second stage, it extracts a point feature from the centre of each face of the predicted 3D box and concatenates these features into an MLP for box refinement and confidence score prediction. The 3DETR system involves downsampling and set-aggregating the original point cloud through the MLP. In the encoder, 3DETR employs the multiple-layer self-attention to generate a set of per-point features. Following this, in the decoder, these features are combined with Query embeddings to generate a set of 3D frame predictions.

C. LiDAR-CAMERA-BASED 3D DETECTION METHODS

To combine the accurate depth information of LiDAR point clouds with the rich semantic content of images, researchers have explored various integration approaches:

Frustum PointNets [7] leverages a 2D CNN target detection network to generate a 2D region proposal. This proposal is projected into 3D view cones, and PointNet is utilized to segment the point cloud inside each view cone into foreground and background points. Finally, the foreground points are processed through PointNet for 3D box prediction and regression. PointPainting [10] projects object edge information from camera images onto the LiDAR point cloud. This process enriches the point cloud data by providing additional geometric information. By using projected image edge information to guide LiDAR point cloud processing, the network can more accurately distinguish dynamic objects from static environments, to improve the target detection robustness.

EPNet [14] introduces a two-stream Region Proposal Network (RPN). The image stream employs 2D convolution to extract semantic image features, while the geometric stream directly extracts features from the point cloud through Set Abstraction and Feature Propagation layers. These features are then enhanced with corresponding semantic image features at different scales using the LI-Fusion module to obtain more discriminative feature representations. RoarNet [8] employs Faster R-CNN to extract features from input images and predict the target's 3D pose and 2D bounding box. Following this, a geometric agreement search is applied to determine the location prediction. A cylinder-shaped region proposal is set around each location prediction, and the point cloud data in each region proposal is passed into the RPN module to generate a new region proposal. Finally, the point cloud data in the new region proposal is passed into the BRN module for 3D frame regression. Transfusion [19] aims to enhance small object detection using high-resolution images. It initializes the object query by projecting image features into Bird's Eye View (BEV) space and performing cross-attention with LiDAR BEV features. This approach establishes a soft correlation between point cloud and image data, improving robustness on image quality degradation and sensor misalignment.

PI-RCNN [12] introduces a new fusion module called PACF. This module selects the nearest k points within a specific range around each 3D point, projects them onto the image feature map, retrieves corresponding semantic features from the image, and combines them with the geometric offsets of the 3D points. The semantic and geometric characteristics of these k points are then fused

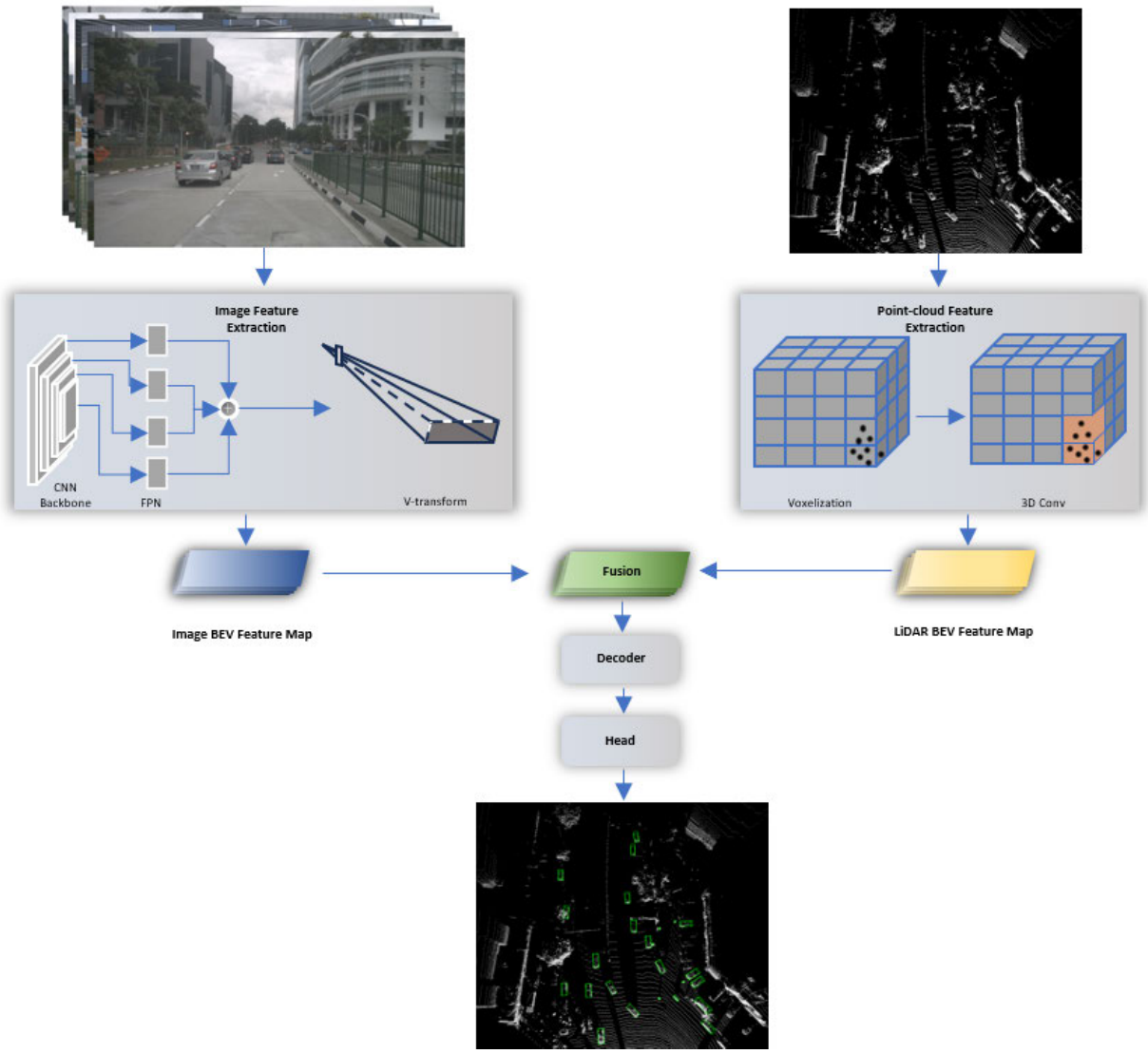


FIGURE 1. The overview of G-Fusion. The top-right block shows the process of voxel BEV feature map extraction. Upon receiving input from LiDAR, the point cloud first goes through a 3D-convolutional backbone and generates a 3D voxel feature map. Then within this feature map, we integrate the height dimension into channels and derive a 2D feature map as a voxel BEV representation. The top-left block illustrates the generation of the image BEV feature map. A CNN backbone is utilized to meticulously extract image features, followed by an FPN structure. Leveraging our proposed v-transform module, the image features are projected into the BEV space and generate an image BEV feature map. Then the voxel and image BEV feature maps are fused via concatenation and a convolutional layer. Finally, the decoder and head further decode the fused features and predict the detection bounding boxes for different obstacle categories.

through a convolutional layer to obtain image features with higher resolution and more semantic information. BEVFusion [18] unifies images and point clouds in a shared BEV space. Instead of mapping a single modality sensor to another, it converts multi-modal features into a unified BEV representation to preserve the semantic density information of the camera and the geometry information of the LiDAR.

III. PROPOSED METHOD

The overview of our method is presented in Fig. 1. The pipeline consists of five modules: feature extraction module, view-transform module, fusion module, decoder module, and detection head module. The structure and function of these modules are illustrated in detail in the following subsections.

A. FEATURE EXTRACTION MODULE FOR IMAGES AND POINT-CLOUDS

The input of the pipeline consists of both images and point clouds. When given an image input, a pre-trained CNN backbone (for example, ResNet [28]) is employed and followed by an FPN to extract the image features. For the point clouds, we choose a 3D-convolutional backbone (for example VoxelNet [6]) to generate a 3D voxel feature map. Then in this feature map, we merge the height and the channel dimensions and derive a LiDAR BEV feature map.

B. VIEW-TRANSFORM AND FUSION MODULE

This part aims to project the image features into a ground voxel space and fuse with the LiDAR BEV feature map.

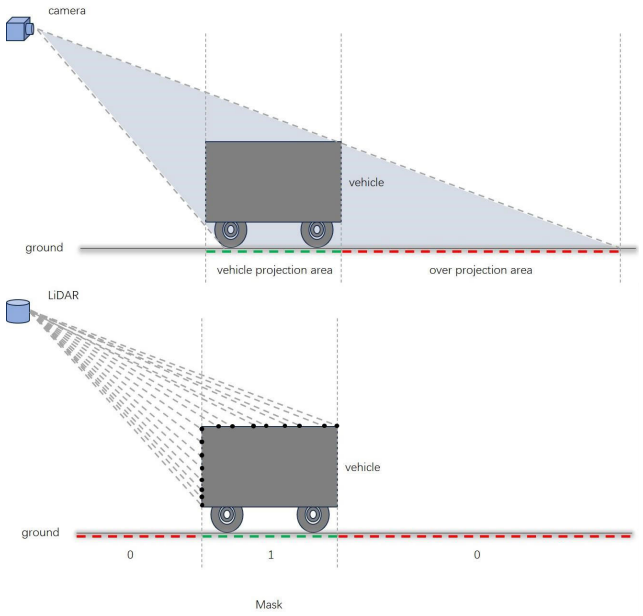


FIGURE 2. Illustration of projection mask. The upper figure shows the image features on the ground voxel space. The green dash line indicates the features projecting to the correct object 3D position, while the red line demonstrates the over-projection areas where the image features are projected beyond the real object boundary. To erase these over-projection areas, we calculate a projection mask. The mask is generated with the help of the LiDAR feature map. As shown in the lower figure, we set 1 and 0 on the mask depending on if there exists LiDAR features, for example, the features reflected from the vehicle. Therefore, utilizing the mask would be able to avoid the projected voxel space carrying many over-projection noises.

Firstly, similar to [21], we build a look-up table based on the projection matrix calculated from the extrinsic and intrinsic matrices. Secondly, with the table, we project the features on the image feature map to the ground voxels to be the image BEV feature map. The ground voxels are in a voxel space where the width and length resolution is the same as the LiDAR BEV feature map, and the layer number of this voxel space is only one and located H meters lower than the LiDAR coordinate origin, where H is the distance between the LiDAR sensor and the ground. When the slopes are ignored, this voxel space falls exactly on the ground, so we call it ground-voxel space.

Fourthly, to filter the over-projection area on the ground-voxel space, as described in Fig. 2, a mask is generated from the LiDAR BEV feature map. For details, since the LiDAR features are sparse and fall on real-existing obstacles, we set 1 on the mask where the LiDAR features exist. Finally, we concatenate the image and LiDAR BEV feature maps and fuse them through a convolutional layer.

C. DECODER AND DETECTION HEAD MODULES

In the decoder module, we use a feature-pyramid structure to generate a dense representation in different receptive fields from the sparse features on the fusion result mentioned above, which could be beneficial to the detection of both large objects (trucks, buses) and small obstacles (such as traffic cones and pedestrians). we follow the detection head in [19] to predict the size, location, and orientation of the object bounding boxes.

IV. EXPERIMENT AND ANALYSIS

A. SETUP

1) DATASET SETTING

We choose nuScenes dataset [22] to evaluate our proposed method. This is because nuScenes provides a rich set of urban road scene data collected from LiDAR and a set of cameras. All these cameras capture six views surrounding the collection vehicle by 360 degrees.

2) EVALUATION METRICS

There are standard metrics for nuScenes datasets, which are mean average precision (mAP) and nuScenes detection score (NDS). The mAP metric evaluates the precision and recall of prediction centre distance in the BEV view from the ground truths with 4 thresholds 0.5, 1, 2, and 4 meters. The NDS metric evaluates the comprehensive performance considering mAP and several average errors towards translation, scale, orientation, velocity, and attribute.

In addition, we also consider running speed as an essential factor in evaluating detection performance. So, in addition to mAP and NDS, following [44], we apply a new metric, called NDS on each Computational Unit (NCU), which can evaluate the running speed and detection performance comprehensively. This metric is calculated as Eq. 1.

$$NCU = NDS \times log(\frac{Max\ Method\ Latency}{Method\ Latency} + 9) \quad (1)$$

where *Max Method Latency* indicates the running time of the slowest method in the comparison group and *Method Latency* is the latency of the current method compared.

3) TRAIN SETTING

Our method is implemented in PyTorch. We choose ResNet [28] as the image backbone network and VoxelNet [6] as the voxel backbone network. In the training process, the images are down-sampled and cropped to 256 × 704. The vertical resolution of the LiDAR in nuScenes is 32-line, which is relatively sparse compared to the novel 128-line LiDARs. So, to provide a dense point cloud in the training process, we apply a commonly used data augmentation method that projects and merges the LiDAR points of sweep frames into their key frames. We use a one-cycle scheduler, similar to [45], with a max learning rate of 0.001 for 20 epochs. The optimizer is AdamW [46] with a weight decay of 0.01. The training process is completed on a workstation with 8 × NVIDIA® GeForce® RTX 3090 GPUs.

B. COMPARISON WITH STATE-OF-THE-ART METHODS

To quantitatively evaluate our proposed pipeline, we compare G-Fusion with nuScenes benchmark methods, which depend on different sensory conditions. In detail: PointPillars [40], SECOND [39], and CenterPoint [30] are on LiDAR-only;

TABLE 1. Experimental results of the comparison group on nuScenes validation dataset, where C and L mean camera and LiDAR, respectively. R18 and R50 represent different layer numbers of ResNet backbones.

	Venue	Sensor	MACs (G)	Latency (ms)	mAP	NDS	NCU
M^2BEV [42]	-	C	-	-	41.7	47.0	-
BEVFormer [32]	ECCV 2022	C	-	-	41.6	51.7	-
FastBEV (R50) [21]	-	C	-	-	33.4	47.3	-
PointPillars [40]	CVPR 2019	L	65.5	34.4	52.3	61.3	79.5
SECOND [39]	Sensors 2018	L	85.0	69.8	52.6	63.0	72.8
CenterPoint [30]	CVPR 2021	L	153.5	80.7	59.6	66.8	75.7
PointPainting [10]	CVPR 2020	C+L	370.0	158.5	65.8	69.6	73.4
MVP [20]	NeurIPS 2021	C+L	371.7	187.1	66.1	70.0	72.9
FUTR3D [43]	CVPR 2023	C+L	1069	371.7	64.5	68.3	68.3
TransFusion [19]	CVPR 2022	C+L	485.8	156.6	67.5	71.3	75.2
BEVFusion (R50) [18]	ICRA 2023	C+L	296.9	116.3	65.3	68.9	74.8
BEVFusion (Swin-T) [18]	ICRA 2023	C+L	253.2	119.2	68.5	71.4	77.4
G-Fusion (R18)	-	C+L	111.0	49.2	62.6	67.0	81.7
G-Fusion (R50)	-	C+L	164.9	50.8	64.1	67.8	82.2

TABLE 2. Latency and MACs of each module in G-Fusion (ResNet50). The running time of the proposed view-transform module is only around 6% of the whole pipeline.

Module Name	Latency (ms)	MACs (G)
LiDAR Backbone	38.06	0.35
Camera Backbone & FPN	3.44	108.18
View-transform	3.11	0.67
Fusion	0.13	25.12
Decoder	1.05	15.29
Head	4.98	15.30
Total	50.8	164.9

M^2BEV [42], BevFormer [32], FastBEV [21] are on Camera-only; PointPainting [10], FUTR3D [43], MVP [20], Transfusion [19], and BEVFusion [18] are on LiDAR-Camera. In addition to comparing the detection metrics, we also measured the inference latency of networks running on a single GPU.

The experiment result is shown in Tab. 1. Firstly, G-Fusion outperforms the three camera-only methods by more than 30% (the percentages in the experimental section all represent the percentage increase compared to other methods). We consider that this is because even though these camera-only-based methods are with delicately designed 2D-3D view-transform approaches, the lack of depth information in images still highly harms the 3D perception capability. Secondly, compared to LiDAR-only based methods, G-Fusion is 7% better than the best CenterPoint on mAP with lower MACs and shorter inference latency, which shows the improvement brought by the image branch and also proves the light computational burden of our proposed view-transform module (also see Tab. 2). Thirdly, looking at the LiDAR-camera-based methods, with the same image backbone ResNet-50, the mAP and NDS of G-Fusion are under 2% lower than BEVFusion. But considering the inference latency, G-Fusion is over 100% faster than BEVFusion. Thus, the NCU of our method is much higher than BEVFusion. Overall, our method achieves the highest NCU among the SoTAs, which illustrates that we find the best balance between detection performance and running speed.

Furthermore, we also compile statistics on the time consumption and computational cost of each module in G-Fusion, as detailed in Table 2. From the table, it can be seen that the time consumption of our view-transform module only accounts for 6% of the entire pipeline, and aside from the projection operation, the computational cost of this module is negligible.

V. ABLATION STUDY

A. IMAGE FEATURES

From the image view-transform process described in Sec. III-B, it is easy to find that G-Fusion only utilizes a part of the image features which can be projected to the ground voxels. This is different from those methods, such as BEVFusion [18] and TransFusion [19], which use all the image features. We will explain the reason for our design by answering two questions:

- Why are fewer features used?
- Why only ground voxels?

To the first question: We consider that after CNN backbones, the receptive field of the feature map is large, so the feature vectors located on neighbouring pixels usually carry similar context information. Therefore, only a few of the feature vectors on the object would be enough to represent the object's feature information. To verify our theory, we visualize the cosine similarity of neighbouring image features using a superimposed heat map, as shown in Fig. 3. From the figure, we can see that all the feature vectors on or even a bit out of the bus carry similar information to the yellow star-marked vector. So, as Fig. 3, even though not all the feature vectors on the bus can be projected onto the ground voxels and fused with the LiDAR features, it is already enough to indicate that there is a bus with semantic and geometric information from the image branch.

To the second question: We think that less projection means less calculation. The view-transform modules of many methods, such as FastBEV [21], project image features onto the voxel layers. We analyze this as redundant. Firstly,

FIGURE 3. Cosine similarity of the features on the image feature map. We input the images into the CNN backbone ResNet-18 [28] and derive latent feature maps with a 256-channel vector on each pixel. Then we choose one vector located on the vehicle (the yellow star) as the query and calculate the cosine similarity between the query and others. The temperature of the heatmaps visualizes the similarity values.

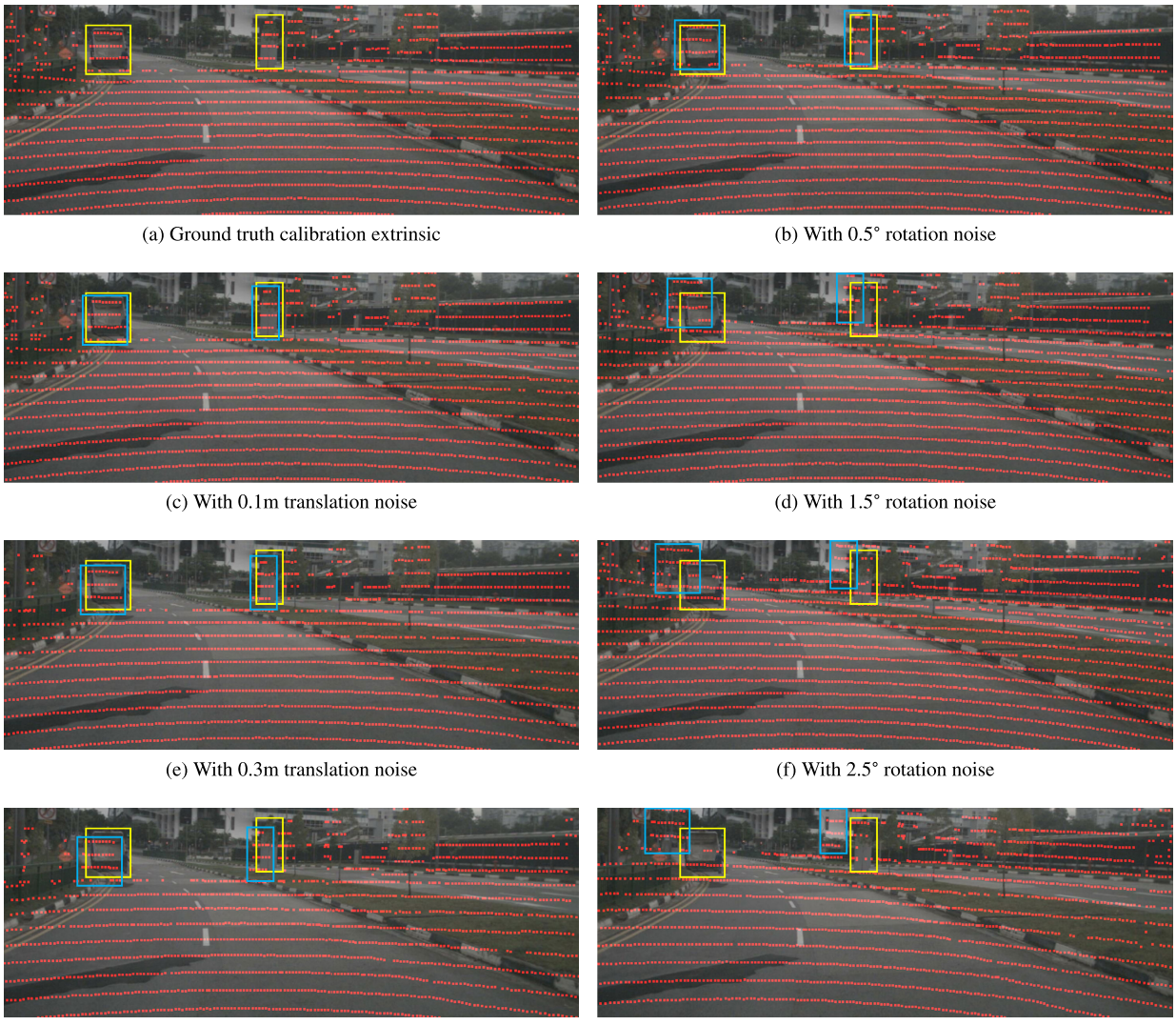
TABLE 3. Experimental result on nuScenes validation dataset, analyzing the influence of different layer numbers and layer heights in the view-transform process. In the last row we also ablate the projection mask (w/o Proj-m) mentioned in Sec. III-B.

Layer number	Layer height	mAP	NDS
0	-	60.0	65.6
3	[-0.20, -0.95, -1.70]	62.5	67.1
2	[-0.20, -1.70]	62.6	66.9
1	[-0.20]	62.2	66.8
1	[-0.95]	62.3	66.6
1	[-1.70]	62.6	67.0
1 (w/o Proj-m)	[-1.70]	62.2	66.3

as discussed above, because of the large receptive field of the image feature maps after a CNN backbone, the image features projected to the voxels at the same vertical column may carry very similar information. Secondly, to fuse with LiDAR BEV feature maps, the projected voxel layers need to be vertically compressed into BEV feature maps. Compressing with a simple summarizing operation is light but would harm the feature representation, while concatenation and convolutional structures would bring heavy computational overhead. Therefore, we only choose one voxel layer for projection. But why do we choose the ground-voxel layer instead of the others? Because although the higher voxel layer easily carries the features from the big objects such as buses and trucks, it is hard to get the small objects like traffic cones. The much lower voxel layer is unreasonable since things under the ground are ignored in the detection tasks. Thus, we only choose the ground-voxel layer.

We conduct experiments to support our consideration, and the detection performances are shown in Tab. 3. With only one voxel layer, we compare the layers with heights between -1.7 m and -0.2 m, which are located between the ground and the upper half of the common objects (cars, pedestrians), and the detection score is shown in the three rows of the table. It can be seen that the detection performance with the ground-voxel layer is better than that with the -0.2 m and -1.0 m layers, on average by 0.3 on NDS and mAP. The result illustrates the rationality of our chosen height. In the upper middle two rows, we compare the model performance with different layer numbers. As described above, we have two ways to compress the projection layers into one layer BEV feature. Here we choose summarizing operation. Because of the numerous feature channels in each project layer, the concatenation and convolutional structures are expensive and will cause memory management problems for GPU. From the scores, we see that increasing the voxel layers does not lead to performance improvement, so more projection voxel layers are proved to be redundant. In addition, the last row ablates the projection mask described in Sec. III-B, and the performance decay proves the effectiveness of this design.

What is more, the top row shows the performance of the model trained under LiDAR-only conditions, and the lower detection scores indicate that the camera



(a) Ground truth calibration extrinsic
(b) With 0.5° rotation noise
(c) With 0.1m translation noise
(d) With 1.5° rotation noise
(e) With 0.3m translation noise
(f) With 2.5° rotation noise
(g) With 0.5m translation noise
(h) With 2.5° rotation and 0.5m translation noise

FIGURE 4. Visualize calibration noise. We project LiDAR points onto an image and highlight them with red spots. We selected two distinct markers and labelled them with yellow boxes. The corresponding projected point clouds are enclosed with blue boxes. As observed, with increasing noise levels, the discrepancy between the green and blue boxes continues to widen.

branch works effectively with the proposed view-transform module.

B. INFLUENCE FROM CALIBRATION ERROR

Detection by multi-sensor fusion needs accurate extrinsic calibration between sensors. However, there are unavoidable vibrations in real autonomous driving scenarios, leading to biases in extrinsic parameters [47]. Thus, in this subsection, we experiment to analyze the impact of calibration errors at different levels. The experiment is divided into two parts. In the first part, the noise is added both to the training and validation process, then we observe the decline in the detection performance. In the second part, we train our model with ground-truth calibration extrinsic but add noise to the data during the validation process. In these two parts, the noise contains both translation noise and rotation noise. The translation noise is a uniform distribution on x, y, and z axes. The rotation noise is also set as a uniform distribution on all the *row*, *yaw*, and *pitch* axes.

For the first part, the experimental result of the first part is shown in Tab.4. In this table, the models are evaluated under two conditions: with and without noise. From the statistical results we can see that with the training noise level shown in the table, the without-noise validation performance has barely dropped, with less than 0.5% under the two metrics. The with-noise validation scores decline slightly higher than above but no more than 1% decreases on both mAP and NDS. These phenomena indicate that our model can tolerate the calibration noise under the $\pm$50 cm translation and $\pm$2.5° rotation bias level for training and inference. Furthermore, we can see that in training with $\pm$10 cm translation noise and $\pm$0.5° rotation noise, the without-noise validation performance does not decline but even shows little increase. This illustrates that adding low-level calibration

TABLE 4. Performance decay of our model with increasing calibration noise on both training and validation datasets. Both the translation noise and rotation noise adhere to a uniform distribution, measured in the centimetre scale and degree scale, respectively.

Training Noise						Validation Noise						mAP	NDS
Translation(±cm)			Rotation(±deg)			Translation(±cm)			Rotation(±deg)				
x	y	z	roll	yaw	pitch	x	y	z	roll	yaw	pitch		
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	62.6	67.0
10.0	10.0	10.0	0.5	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	62.7	67.1
20.0	20.0	20.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	62.6	66.9
30.0	30.0	30.0	1.5	1.5	1.5	0.0	0.0	0.0	0.0	0.0	0.0	62.5	66.9
40.0	40.0	40.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	62.3	66.7
50.0	50.0	50.0	2.5	2.5	2.5	0.0	0.0	0.0	0.0	0.0	0.0	62.4	66.9
10.0	10.0	10.0	0.5	0.5	0.5	10.0	10.0	10.0	0.5	0.5	0.5	62.6	67.0
20.0	20.0	20.0	1.0	1.0	1.0	20.0	20.0	20.0	1.0	1.0	1.0	62.4	66.8
30.0	30.0	30.0	1.5	1.5	1.5	30.0	30.0	30.0	1.5	1.5	1.5	62.3	66.8
40.0	40.0	40.0	2.0	2.0	2.0	40.0	40.0	40.0	2.0	2.0	2.0	62.1	66.5
50.0	50.0	50.0	2.5	2.5	2.5	50.0	50.0	50.0	2.5	2.5	2.5	62.1	66.7

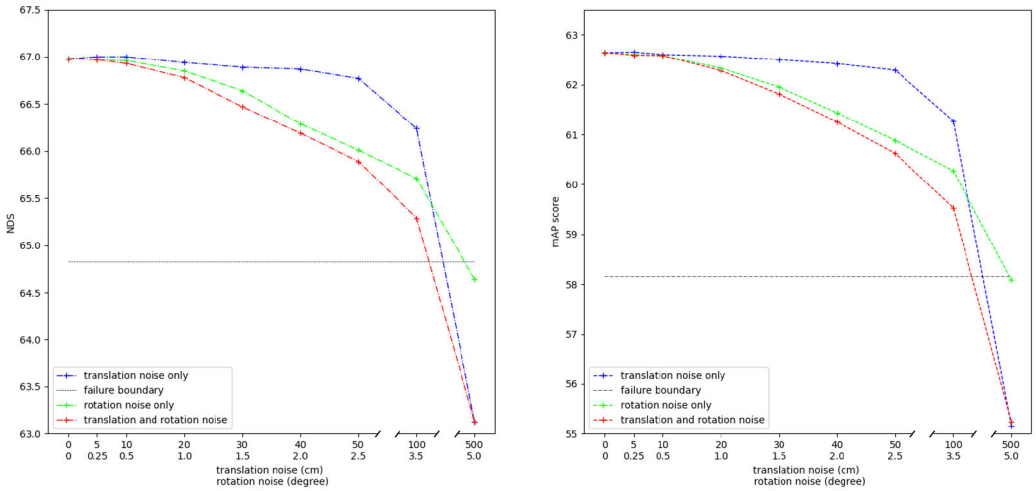


FIGURE 5. Qualitatively visualize the statistics in Tab. 5. When evaluating the performance under increasing noise conditions, it becomes evident that the camera branch demonstrates a notable decline in detection capability. But, as long as the noise level is maintained below thresholds of 100 cm and 3.5°, this camera branch can provide a positive contribution to the overall detection performance.

noise to train data can be taken as a data augmentation strategy to improve model performance.

For the second part, training with ground-truth calibration extrinsic, we separately analyze the translation and rotation noises of the calibration parameters and observe the performance decay. To evaluate the fused feature, we make a definition of the model failure boundary. The boundary is set as the inference performance on the nuScenes dataset when removing the camera input, which is under the condition that the model is trained with both LiDAR and camera branches with accurate calibration parameters. Lowering the boundary means that the bad calibration extrinsic results in the image information contribute negatively to the fusion network. The score of the failure boundary is shown in Fig. 5 with black dash lines. Noise condition and detection scores are shown in Tab. 5 and Fig. 5, quantitatively and qualitatively. From the table and graphs, we see that with lower than 20 cm translation noise, the performance barely changes, and increasing the noise to 50 cm, the model performance only drops less than 0.6%, which is far away from the failure boundary. Therefore, our method demonstrates strong resistance to translation noise. However, as the pure rotation calibration noise rises, our method suffers an observable performance decline. With the bias of less than 0.5°, the detection scores do not change, but when the noise level increases to 2.5°, the detection mAP drops by nearly 2.4%, and the performance passes the failure limit before 5°. With both the translation and rotation noise, we find that the model is mostly influenced by the angle error because the detection score under combined noise is nearly as same as under pure rotation noise. In conclusion, under calibration noise lower than 100 cm and 3.5° bias on three axes, the camera branch can work positively and contribute to the detection accuracy, which we believe is robust enough to deal with the vibration disturbances in real autonomous driving scenarios.

TABLE 5. Decline in our model performance with increasing calibration noise only on the validation dataset. Same as Tab. 4, the noise in translation and rotation both align with a uniform distribution, quantified in terms of centimetres and degrees, respectively.

	Translation Noise (±cm)			Rotation Noise (±deg)			mAP	NDS
	x	y	z	roll	yaw	pitch		
w/o Camera Branch	-	-	-	-	-	-	58.3	65.2
Default	0.0	0.0	0.0	0.0	0.0	0.0	62.6	67.0
Translation Only	5.0	5.0	5.0	0.0	0.0	0.0	62.7	67.0
	10.0	10.0	10.0	0.0	0.0	0.0	62.6	67.0
	20.0	20.0	20.0	0.0	0.0	0.0	62.6	66.9
	30.0	30.0	30.0	0.0	0.0	0.0	62.5	66.9
	40.0	40.0	40.0	0.0	0.0	0.0	62.4	66.9
	50.0	50.0	50.0	0.0	0.0	0.0	62.3	66.8
	100.0	100.0	100.0	0.0	0.0	0.0	61.3	66.2
	500.0	500.0	500.0	0.0	0.0	0.0	55.2	63.1
Rotation Only	0.0	0.0	0.0	0.25	0.25	0.25	62.6	67.0
	0.0	0.0	0.0	0.5	0.5	0.5	62.6	67.0
	0.0	0.0	0.0	1.0	1.0	1.0	62.3	66.9
	0.0	0.0	0.0	1.5	1.5	1.5	62.0	66.6
	0.0	0.0	0.0	2.0	2.0	2.0	61.4	66.3
	0.0	0.0	0.0	2.5	2.5	2.5	60.9	66.0
	0.0	0.0	0.0	3.5	3.5	3.5	60.3	65.7
	0.0	0.0	0.0	5.0	5.0	5.0	58.1	64.6
Translation and Rotation	5.0	5.0	5.0	0.25	0.25	0.25	62.6	67.0
	10.0	10.0	10.0	0.5	0.5	0.5	62.6	66.9
	20.0	20.0	20.0	1.0	1.0	1.0	62.3	66.8
	30.0	30.0	30.0	1.5	1.5	1.5	61.8	66.5
	40.0	40.0	40.0	2.0	2.0	2.0	61.3	66.2
	50.0	50.0	50.0	2.5	2.5	2.5	60.6	65.9
	100.0	100.0	100.0	3.5	3.5	3.5	59.5	65.3
	500.0	500.0	500.0	5.0	5.0	5.0	55.2	63.1

VI. CONCLUSION

This paper designs a lightweight 3D detection method G-Fusion based on LiDAR and Camera. According to the analysis of the receptive field of image feature maps, a light feature projection module is proposed and highly accelerates the image view transformation compared with the other fusion detection methods. Also, G-Fusion achieves competitive performance on nuScenes datasets with two times faster running speed than the SoTAs. In addition, considering practical application scenarios, we conduct experiments to analyze the tolerance ability of G-Fusion to the LiDAR-Camera calibration bias. Experimental results show that the camera branch can positively support the detection performance as long as translation and rotation bias are lower than 100 cm and 3.5°, proving the usability of our method in real driving scenarios.

ACKNOWLEDGMENT

The authors would like to thank Qǐxī Zhào for his assistance in the project.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[3] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth, "Efficient multi-cue scene segmentation," in *Proc. German Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2013, pp. 435–445.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6526–6534.

[5] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3061–3070.

[6] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[7] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[8] K. Shin, Y. P. Kwon, and M. Tomizuka, "RoarNet: A robust 3D object detection based on RegiOn approximation refinement," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 2510–2515.

[9] R. Nabati and H. Qi, "CenterFusion: Center-based radar and camera fusion for 3D object detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 1526–1535.

[10] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4603–4611.

[11] S. Xu, D. Zhou, J. Fang, J. Yin, Z. Bin, and L. Zhang, "FusionPainting: Multimodal fusion with adaptive attention for 3D object detection," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 3047–3054.

[12] L. Xie, "PI-RCNN: An efficient multi-sensor 3D object detector with point-based attentive cont-conv fusion module," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, 2020, pp. 12460–12467.

[13] H. Yang, C. Shi, Y. Chen, and L. Wang, "Boosting 3D object detection via object-focused image fusion," 2022, *arXiv:2207.10589*.

[14] T. Huang, Z. Liu, X. Chen, and X. Bai, "EPNet: Enhancing point features with image semantics for 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, Aug. 2020, pp. 35–52.
[15] Y. Li, A. W. Yu, T. Meng, B. Caine, J. Ngiam, D. Peng, J. Shen, Y. Lu, D. Zhou, Q. V. Le, A. Yuille, and M. Tan, "DeepFusion: LiDAR-camera deep fusion for multi-modal 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 17161–17170.
[16] Y. Li, Y. Chen, X. Qi, Z. Li, J. Sun, and J. Jia, "Unifying voxel-based representation with transformer for 3D object detection," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 18442–18455.
[17] H. Wang, H. Tang, S. Shi, A. Li, Z. Li, B. Schiele, and L. Wang, "UniTR: A unified and efficient multi-modal transformer for bird's-eye-view representation," 2023, *arXiv:2308.07732*.
[18] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 2774–2781.
[19] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "TransFusion: Robust LiDAR-camera fusion for 3D object detection with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 1080–1089.
[20] T. Yin, X. Zhou, and P. Krähenbühl, "Multimodal virtual point 3D detection," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34. Red Hook, NY, USA: Curran Associates, 2021, pp. 16494–16507.
[21] B. Huang, Y. Li, E. Xie, F. Liang, L. Wang, M. Shen, F. Liu, T. Wang, P. Luo, and J. Shao, "Fast-BEV: Towards real-time on-vehicle bird's-eye view perception," 2023, *arXiv:2301.07870*.
[22] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "NuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11618–11628.
[23] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 228–242, Feb. 2008.
[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.
[25] Z. Yin and J. Shi, "GeoNet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1983–1992.
[26] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, Aug. 2020, pp. 194–210.
[27] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "BEVDet: High-performance multi-camera 3D object detection in bird-eye-view," 2021, *arXiv:2112.11790*.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[29] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9992–10002.
[30] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11779–11788.
[31] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "DETR3D: 3D object detection from multi-view images via 3D-to-2D queries," in *Proc. Conf. Robot Learn.*, 2022, pp. 180–191.
[32] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 1–18.
[33] T. Wang, X. Zhu, J. Pang, and D. Lin, "FCOS3D: Fully convolutional one-stage monocular 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 913–922.
[34] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.
[35] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.
[36] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6410–6419.
[37] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.
[38] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1355–1361.
[39] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct. 2018.
[40] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12689–12697.
[41] I. Misra, R. Girdhar, and A. Joulin, "An end-to-end transformer model for 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2886–2897.
[42] E. Xie, Z. Yu, D. Zhou, J. Philion, A. Anandkumar, S. Fidler, P. Luo, and J. M. Alvarez, "M^2BEV: Multi-camera joint 3D detection and segmentation with unified birds-eye view representation," 2022, *arXiv:2204.05088*.
[43] X. Chen, T. Zhang, Y. Wang, Y. Wang, and H. Zhao, "FUTR3D: A unified sensor fusion framework for 3D detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2023, pp. 172–181.
[44] M. Zaffar, S. Ehsan, M. Milford, and K. McDonald-Maier, "CoHOG: A light-weight, compute-efficient, and training-free visual place recognition technique for changing environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1835–1842, Apr. 2020.
[45] T. Cazenave, J. Sentuc, and M. Videau, "Cosine annealing, Mixnet and swish activation for computer go," in *Advances in Computer Games*. Cham, Switzerland: Springer, 2021, pp. 53–60.
[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.
[47] H. Jie, Y. Zhou, Z. Ning, Q. Zhao, L. Liu, and J. Hu, "DyLESC: A dynamic LiDAR extrinsic self-calibration method for intelligent driving vehicles," in *Proc. IEEE 7th Int. Conf. Intell. Transp. Eng. (ICITE)*, Nov. 2022, pp. 185–190.

SHUAI CHENG received the M.S. and Ph.D. degrees from the Changchun University of Science and Technology, Shenyang, China, in 2012 and 2016, respectively.

His current research interests include deep learning, object tracking, and trajectory prediction in complex traffic scenes.

ZUOTAO NING received the B.S. degree in software engineering from Dalian Jiaotong University, Dalian, China, in 2013.

He is currently an Assistant Researcher. His research interests include the environmental perception of autonomous driving vehicles and mobile robots. His recent work has involved exploring and developing efficient and accurate point cloud processing techniques in intelligent driving applications.

JUN HU received the Ph.D. degree in computer science and engineering from Northeastern University, Shenyang, China, in 2023.

Currently, he is the Deputy Director of the Automatic Driving Business Line with Neusoft Reachauto Corporation (Shenyang) and a Senior Engineer. His current research interests include assisted driving and automatic driving, image processing, pattern recognition, and the vision-based perception of complex traffic scenes.

JIAXIN LIU received the B.S. and M.S. degrees from Northeastern University, Shenyang, China, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree in computer science and technology.

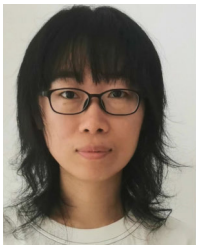
He is also an Intern Engineer with Reachauto, Neusoft Corporation, working on deep learning applications in the field of autonomous driving. His research interests include computer vision, deep learning, and model compression.

WENXING YANG received the M.S. degree in computer science from Northeastern University, Shenyang, China, in 2019, where he is currently pursuing the Ph.D. degree in computer science with the School of Computer Science and Engineering.

His research interests include visual perception and prediction related to autonomous driving.

LUYANG WANG received the master's degree from the College of Computer Science and Engineering, Northeast University, in 2011.

His main research interests include image processing, data mining, and artificial intelligence.



HONGFEI YU received the M.S. degree in applied mathematics and the Ph.D. degree in computer application technology from Northeastern University, in 2008 and 2014, respectively.

Her main research interests include computer vision and autonomous driving, image processing, and pattern recognition with applications to intelligent video surveillance and advanced driver assistance systems.

WEI LIU received the M.S. and Ph.D. degrees in control theory and control engineering from Northeastern University, Shenyang, China, in 2001 and 2005, respectively.

He is currently a professor-level Senior Engineer with Research Academy, Northeastern University. He is also the Director of the Intelligent Vision Laboratory, Neusoft Corporation, China. His research interests include computer vision, image processing, and pattern recognition with applications to intelligent video surveillance and advanced driver assistance systems.