# LiDAR-Camera-Based Deep Dense Fusion for Robust 3D Object Detection

Lihua Wen and Kang-Hyun Jo(✉)

School of Electrical Engineering, University of Ulsan, Ulsan, South Korea
wenlihua@islab.ulsan.ac.kr, acejo@ulsan.ac.kr

**Abstract.** For the camera-LiDAR-based three-dimensional (3D) object detection, image features have rich texture descriptions and LiDAR features possess objects' 3D information. To fully fuse view-specific feature maps, this paper aims to explore the two-directional fusion of arbitrary size camera feature maps and LiDAR feature maps in the early feature extraction stage. Towards this target, a deep dense fusion 3D object detection framework is proposed for autonomous driving. This is a two stage end-to-end learnable architecture, which takes 2D images and raw LiDAR point clouds as inputs and fully fuses view-specific features to achieve high-precision oriented 3D detection. To fuse the arbitrary-size features from different views, a multi-view resizes layer (MVRL) is born. Massive experiments evaluated on the KITTI benchmark suite show that the proposed approach outperforms most state-of-the-art multi-sensor-based methods on all three classes on moderate difficulty (3D/BEV): Car (75.60%/88.65%), Pedestrian (64.36%/66.98%), Cyclist (57.53%/57.30%). Specifically, the DDF3D greatly improves the detection accuracy of hard difficulty in 2D detection with an 88.19% accuracy for the car class.

**Keywords:** Two directional fusion · 3D object detection · Autonomous driving

## 1 Introduction

This paper focuses on 3D object detection, which is a fundamental and key computer vision problem impacting most intelligent robotics perception systems including autonomous vehicles and drones. To achieve robust and accurate scene understanding, autonomous vehicles are usually equipped with various sensors (e.g. camera, Radar, LiDAR) with different functions, and multiple sensing modalities can be fused to exploit their complementary properties. However, developing a reliable and accurate perception system for autonomous driving based on multiple sensors is still a very challenging task.

Recently, 2D object detection with the power of deep learning has drawn much attention. LiDAR-based 3D object detection also becomes popular with deep learning. Point clouds generated by LiDAR to capture surrounding objects and return accurate depth and reflection intensity information to reconstruct the objects. Since the sparse and unordered attributes of point clouds, representative works either convert raw point clouds into bird-eye-view (BEV) pseudo images [1–4], 2D front view images [2], or structured voxels grid representations [5–7]. Some references [8–10] directly deal with raw point clouds by multi-layer perceptron (MLP) to estimate the 3D object and localization. However, due to the sparsity of point clouds, these LiDAR-based approaches suffer information loss severely in long-range regions and when dealing with small objects.

On the other hand, 2D RGB images provide dense texture descriptions and also enough information for small objects based on high resolution, but it is still hard to get precise 3D localization information due to the loss of depth information caused by perspective projection, particularly when using monocular camera [11–13]. Even if using stereo images [14], the accuracy of the estimated depth cannot be guaranteed, especially under poor weather, dark and unseen scenes. Therefore, some approaches [15–19] have attempted to take the mutual advantage of point clouds and 2D images. However, they either utilize *Early Fusion*, *Late Fusion*, or *Middle Fusion* is shown in Fig. 1 to shallowly fuse two kinds of features from 2D images and point clouds. Their approaches make the result inaccurate and not stable.

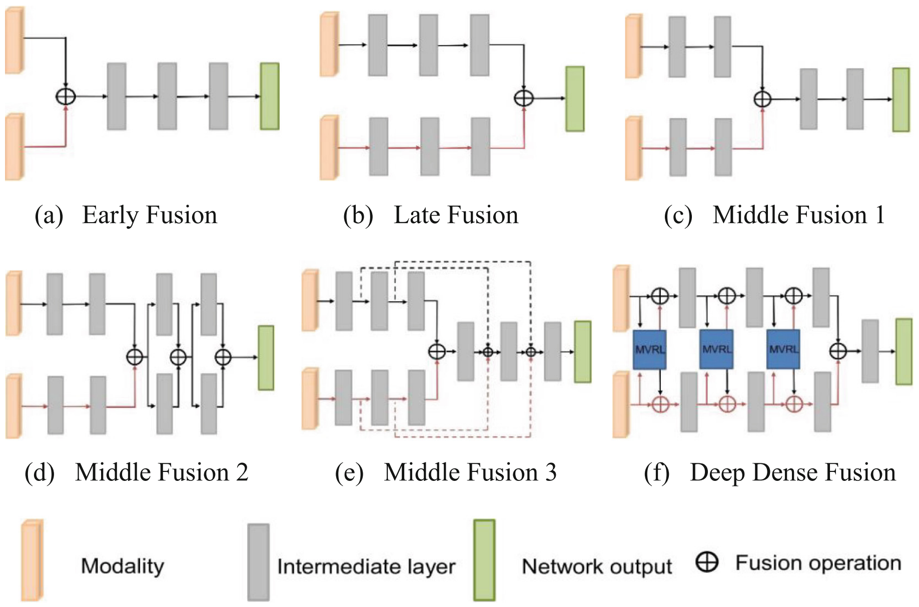

**Fig. 1.** A comparison of existed fusion methods and the deep dense fusion (proposed). Compared with methods (a–e), the deep dense fusion moves forward to the feature extraction phase and becomes denser. The proposed fusion method fully integrates each other's characteristics.

MV3D [2] and AVOD [6] fuse region-based multi-modal features at the region proposal network (RPN) and detection stage, the *local* fusion method causes the loss of semantic and makes its results inaccurate. Conversely, ContFusion [20] proposed a *global* fusion method to fuse BEV features and image features from different feature levels, it verifies the superiority of the full fusion of 2D image and point clouds.

However, ContFusion [20] is only unidirectional fusion. Based on logical experience, a bidirectional fusion will be even more superior than the unidirectional fusion. The challenge lies in the fact that the image feature is dense at discrete state, while LiDAR points are continuous and sparse. Thus, fusing them in both directions is non-trivial.

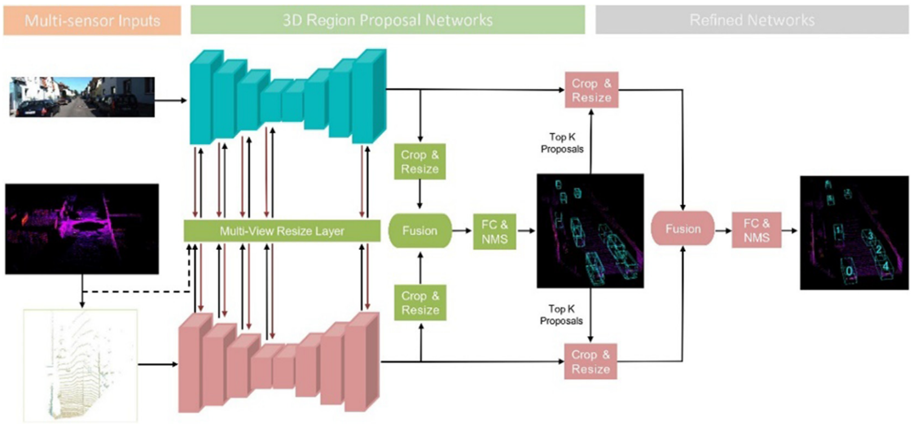

**Fig. 2.** The architecture of deep dense fusion 3D object detection network.

This paper proposes a two-stage multi-sensor 3D detector, called DDF3D, which fuses image feature and BEV feature at different levels of resolution. The DDF3D is an end-to-end learnable architecture consisting of a 3D region proposal subnet (RPN) and a refined detector subnet in the order illustrated in Fig. 2. First, the raw point clouds are partitioned into six-channel pseudo images and 2D images are cropped based on the central region. Second, two identical fully convolutional networks are used to extract view-specific features and fuse them by the MVRL simultaneously. Third, 3D anchors are generated from BEV, and anchor-dependent features from different views are fused to produce 3D non-oriented region proposals. Finally, the proposal-dependent features are fused again and fed to the refined subnetwork to regress dimension, orientation, and classify category.

Here, the contributions in this paper have summarized in 3 points as follows:

1. A highly efficient multi-view resizes layer (MVRL) designed to resize the features from BEV and camera simultaneously, which makes real-time fusion of multiple view-specific feature maps possible.
2. Based on the MVRL, a deep dense fusion method is proposed to fully fuse view-specific feature maps at different levels of resolution synchronously. The fusion method allows different feature maps to be fully fused during feature extraction, which greatly improves the detection accuracy of small size object.
3. The proposed architecture achieves a higher and robust 3D detection and localization accuracy for car, bicycle, and pedestrian class. Especially the proposed architecture greatly improves the accuracy of small classes on both 2D and 3D.

## 2 The Proposed Architecture

The main innovation of proposed DDF3D, depicted in Fig. 2, is to fully fuse view-specific features simultaneously based on the MVRL, and the fused features are fed into the next convolutional layers at BEV stream and camera stream respectively, the detailed procedure is shown in Fig. 1f. After feature extractor, both feature maps are fused again and the 3D RPN is utilized to generate 3D non-oriented region proposals, which are fed to the refined detection subnetwork for dimension refinement, orientation estimation, and category classification.

**Birds Eye View Representation** Like MV3D [2] and AVOD [15], a six-channel BEV map is generated by encoding the height and density in each voxel of each LiDAR frame. Especially, the height is the absolute height relative to the ground. First, the raw point clouds are located in $[-40, 40] \times [0, 70]$ m and limited to the field of camera view. Along X and Y axis, the point clouds are voxelized at the resolution of 0.1 m. Then, the voxelized point clouds are equally sliced 5 slices between $[-2.3, 0.2]$ m along the Z axis. Finally, the point density in each cell computed as $min\left(1.0, \frac{\log(N+1)}{\log 64}\right)$, where N is the number of points in a pillar. Note that the density features are computed for the whole point clouds while the height feature is computed for 5 slices, thus a $700 \times 800 \times 6$ BEV feature is generated for each LiDAR frame. In addition to output a feature map, each LiDAR frame also outputs the voxelized points to construct the MVRL.

### 2.1 The Feature Extractor and Multi-view Resize Layer

This section will introduce the feature extractor and MVRL. The MVRL is used to resize the view-specific features at a different resolution, then view-specific features are concatenated with the features resized from different views. Finally, the fused features are fed into the next convolutional layers.

**The Multi-view Resize Layer.** To fuse feature maps from different perspectives is not easy since the feature maps from different views are of different sizes. Also, fusion efficiency is a challenge. So, the multi-view resize layer is designed to bridge multiple intermediate layers on both sides to resize multi-sensor features at multiple scales with highly efficient. The inputs of MVRL contains three parts: the source BEV indices $I_{bev/ori}$ and LiDAR points $P_{ori}$ obtained during a density feature generation, the camera feature $f_{cam}$, and the BEV feature $f_{bev}$. The workflow of the MVRL shown in Fig. 3. The MVRL consists of data preparation and bidirectional fusion. In data preparation, the voxelized points $P_{ori}$ are projected onto the camera plane, the process is formulated as Eq. 1 and Eq. 2, and the points $P_{cam}$ in original image size $360 \times 1200$ are kept. The points $P_{cam/fusion}$ in image size $H_i \times W_i$ are used to obtained image indices $I_{camcam/fusion}$ based on bilinear interpolation. A new BEV index $I_{bev/fusion}$ are obtained based on BEV indices $I_{bev/ori}$ and BEV size $H_b \times W_b$. Then, a sparse tensor $T_s$ with $H_b \times W_b$ shape is generated by image indices $I_{cam/fusion}$ and BEV indices $I_{bev/fusion}$. Finally, a feature multiplies the sparse tensor to generate the feature which can be fused by a camera feature map or an image feature map formulated as Eq. 3 and Eq. 4.

$$(\mathrm{u}, \mathrm{v})^{\mathrm{T}} = \mathrm{M} \cdot (\mathrm{x}, \mathrm{y}, \mathrm{z})^{\mathrm{T}}, \tag{1}$$

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

$$\mathrm{f}_{\mathrm{b2c}} = \mathrm{S}\left(\mathrm{Matmul}\left(\mathrm{T}_{\mathrm{s}}^{-1}, \mathrm{f}_{\mathrm{bev}}\right)\right), \tag{3}$$

$$\mathrm{f}_{\mathrm{c2b}} = \mathrm{R}\left(\mathrm{Matmul}\left(\mathrm{T}_{\mathrm{s}}, \mathrm{G}\left(\mathrm{f}_{\mathrm{cam}}, \mathrm{I}_{\mathrm{cam/fusion}}\right)\right)\right), \tag{4}$$

where (x, y, z) is a LiDAR point coordinate and (u, v) is image coordinate, $\boldsymbol{P_{rect}}$ is a project matrix, $\boldsymbol{R^{cam}_{velo}}$ is the rotation matrix from LiDAR to the camera, $\boldsymbol{t^{cam}_{velo}}$ is a translation vector, **M** is the homogeneous transformation matrix from LiDAR to the camera, **S** and **G** represent scatter operation and gather operation, respectively, **Matmul** means multiplication, **R** means reshape operation, $\mathrm{f}_{\mathrm{b2c}}$ is the feature transferred from BEV to the camera, Conversely, $\mathrm{f}_{\mathrm{c2b}}$ is the feature transferred from the camera to BEV.

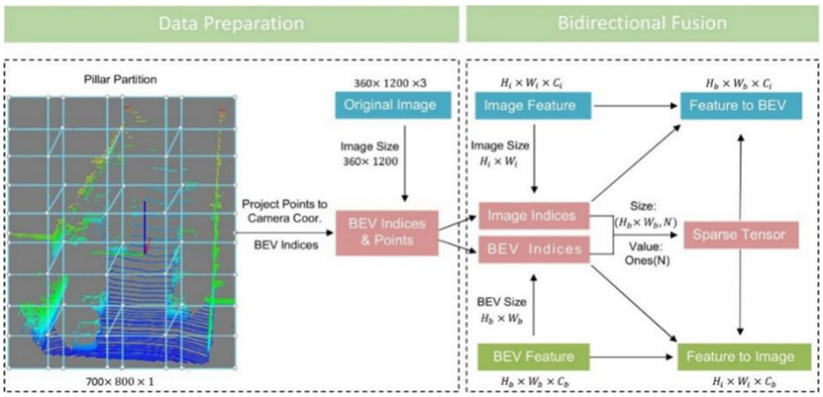

**Fig. 3.** Multi-view resize layer: it includes data preparation and bidirectional fusion.

**The Feature Extractor.** The backbone network follows a two-stream architecture [22] to process multi-sensor data. Specifically, it uses two identical CNNs to extract features from both of 2D image and BEV representation in this paper. Each CNNs includes two parts: an encoder and a decoder. VGG-16 [23] is modified and simplified as the encoder. The convolutional layers from conv-1 to conv-4 are kept, and the channel number is reduced by half. In the feature extraction stage, the MVRL is used to resize two-side features. A little of information is retained for small classes such as cyclists and pedestrians in the output feature map. Therefore, inspired by FCNs [24] and Feature Pyramid Network (FPN) [25], a decoder is designed to up-sample the features back to the original input size. To fully fuse the view-specific features, The MVRL is used again to resize features after decoding. The final feature map has powerful semantics with a high resolution, and are fed into the 3D RPN and the refined subnetwork.

### 2.1.1 3D Region Proposal Network

**3D Anchor Generation and Fusion.** Unlike MV3D [2], this paper directly generates 3D plane-based anchors like AVOD [15] and MMF [22]. The 3D anchors are parameterized by the centroid $\left(c_x, c_y, c_z\right)$ and axis aligned dimensions $\left(d_x, d_y, d_z\right)$. The $\left(c_x, c_y\right)$ pairs are sampled at intervals of 0.5 m in the BEV, while $c_z$ is a fixed value that is determined according to the height of the sensor related to the ground plane. Since this paper does not regress the orientation at the 3D proposal stage, the $\left(d_x, d_y, d_z\right)$ are transformed from (*w, l, h*) of the prior 3D boxes based on rotations. Furthermore, the (*w, l, h*) are determined by clustering the training samples for each class. For the car case, each location has two sizes of anchors. While each location only has one size of anchor for pedestrians and cyclists.

**3D Proposal Generation.** AVOD [15] reduces the channel number of BEV and image feature maps to 1, and aims to process anchors with a small memory overhead. The truncated features are used to generate region proposals. However, the rough way loses most of the key features and causes proposal instability. To keep proposal stability and small memory overhead, we propose to apply a $1\ \times 1$ convolutional kernel on the view-specific features output by the decoder, and the output number of channels is the same as the input. Each 3D anchor is projected onto the BEV and image feature maps output by the $1\ \times 1$ convolutional layer to obtain two corresponding region-based features. Then, these features are cropped and resized to equal-length vectors, e.g. $3 \times 3$. These fixed-length feature crop pairs from two views fused by concatenation operation.

Two similar branches [15] of 256-dimension fully connected layers take the fused feature crops as input to regress 3D proposal boxes and perform binary classification. The regression branch is to regress $\left(\Delta c_x, \Delta c_y, \Delta c_z, \Delta d_x, \Delta d_y, \Delta d_z\right)$ between anchors and target proposals. The classification branch is to determine an anchor to capture an object or background based on a score. Note that we divide all 3D anchors into negative, positive, ignore by projecting 3D anchors and corresponding ground-truth to BEV to calculate the 2D IoU between the anchors and the ground truth bounding boxes. For the car class, anchors with IoU less than 0.3 are considered negative anchors, while ones with IoU greater than 0.5 are considered positive anchors. Others are ignored. For the pedestrian and cyclist classes, the object anchor IoU threshold is reduced to 0.45. The ignored anchors do not contribute to the training objective [21].

The loss function in 3D proposal stage is defined as follows:

$$Loss = \lambda L_{cls} + \gamma L_{box}, \tag{5}$$

where $L_{cls}$ is the focal loss for object classification and $L_{box}$ is the smooth $l1$ loss for 3D proposal box regression, $\lambda = 1.0$, $\gamma = 5.0$ are the weights to balance different tasks.

Followed by two task-specific branches, 2D non-maximum suppression (NMS) at an IoU threshold of 0.8 in BEV is used to remove redundant 3D proposals and the top 1,024 3D proposals are kept during the training stage. At inference time, 300 3D proposals are kept for the car class, and 1,024 3D proposals are used for cyclist and pedestrian class.

## 2.2 The Refined Network

The refined network aims to further optimize the detection based on the top $K$ non-oriented region proposals and the features output by the two identical CNN to improve the final 3D object detection performance. First, the top $K$ non-oriented region proposals are projected onto BEV and image feature maps output by feature extractors to obtain two corresponding region-based features. The region-based features are cropped and resized to 7 × 7 equal-length shapes. Then, the paired fixed-length crops from two views are fused with element-wise mean method. The fused features are fed into a three parallel fully connected layers for outputting bounding box regression, orientation estimation, and category classification, simultaneously. MV3D [2] proposes an 8-corner encoding, however, it does not take into account the physical constraints of a 3D bounding box. Like AVOD [15], a plane-based 3D bounding box is represented by a 10-dimensional vector to remove redundancy and keep the physical constraints. Ground truth boxes and 3D anchors are defined by $(x_1 \cdots x_4, y_1 \cdots y_4, h_1, h_2, \theta)$. The corresponding regression residuals between 3D anchors and ground truth are defined as follows:

$$\Delta \mathrm{x} = \frac{\mathrm{x}_\mathrm{c}^\mathrm{g} - \mathrm{x}_\mathrm{c}^\mathrm{a}}{\mathrm{d}^\mathrm{a}}, \Delta \mathrm{y} = \frac{\mathrm{y}_\mathrm{c}^\mathrm{g} - \mathrm{y}_\mathrm{c}^\mathrm{a}}{\mathrm{d}^\mathrm{a}}, \Delta \mathrm{h} = \log\left(\frac{\mathrm{h}^\mathrm{g}}{\mathrm{h}^\mathrm{a}}\right), \tag{6}$$

$$\Delta\theta = \sin(\theta^\mathrm{g} - \theta^\mathrm{a}), \tag{7}$$

where $d^a = \sqrt{(x_1 - x_2)^2 + (y_4 - y_1)^2}$ is the diagonal of the base of the anchor box.

The localization loss function and orientation loss function [7] as follows:

$$\mathrm{L_{box}} = \sum\nolimits_{\mathrm{b} \in \left(\mathrm{x_1 \cdots x_4, y_1 \cdots y_4, h_1, h_2}, \theta\right)} \mathrm{Smooth_{L1}}(\Delta \mathrm{b}), \tag{8}$$

$$\mathrm{L_{dir}} = \sum \mathrm{Smooth_{L1}}(\Delta\theta). \tag{9}$$

For the object classification loss, the focal loss is used:

$$\mathrm{L_{cls}} = -\alpha_\mathrm{a}(1 - \mathrm{p}^\mathrm{a})^\gamma \log(\mathrm{p}^\mathrm{a}), \tag{10}$$

where $\mathrm{p}^\mathrm{a}$ is the class probability of an anchor, we set $\alpha$ = 0:25 and $\gamma$ = 2, the total loss for the refined network is, therefore,

$$\mathrm{Loss} = \frac{1}{\mathrm{N_{pos}}}(\beta_1 \mathrm{L_{box}} + \beta_2 \mathrm{L_{cls}} + \beta_3 \mathrm{L_{dir}}), \tag{11}$$

Where $\mathrm{N_{pos}}$ is the number of positive anchors and $\beta_1$ = 5.0, $\beta_2$ = 1.0, $\beta_3$ = 1.0.

In refined network, 3D proposals are only considered in the evaluation of the regression loss if they have at least a 0.65 2D IoU in bird's eye view with the ground-truth boxes for the car class (0.55 for pedestrian/cyclist classes). NMS is used at a threshold of 0.01 to choose out the best detections.

## 3 Experiments and Results

### 3.1 Implementation Details

Due to the 2D RGB camera images are with different size, the images are center-cropped into a uniform size of 1200 × 360. Each point clouds are voxelized as a 700 × 800 × 6 BEV pseudo image. For data augmentation, it flips images, voxelized pseudo images, and ground-truth labels horizontally at the same time with a probability of 0.5 during the training. The DDF3D model is implemented by TensorFlow on one NVIDIA 1080 Ti GPU with a batch size of 1. Adam is the optimizer. The DDF3D model is trained for a total of 120K iterations with the initial learning rate of 0.0001, and decayed by 0.1 at 60K iterations and 90K iterations. The whole training process takes only 14 h, and the DDF3D model is evaluated from 80K iterations to 120K iterations every 5K iterations.

### 3.2 Quantitative Results

**Table 1.** Comparison of the 3D Object and BEV performance of DDF3D with state-of-the-art 3D object detectors.

| Class | Method | Time | 3D AP (%) | | | BEV AP (%) | | |
|---|---|---|---|---|---|---|---|---|
| | | | E | M | H | E | M | H |
| Car | MV3D [2] | 0.36 | 71.29 | 62.68 | 56.56 | 86.55 | 78.10 | 76.67 |
| | AVOD [15] | 0.10 | 84.41 | 74.44 | 68.65 | – | – | – |
| | F-PointNet [17] | 0.17 | 83.76 | 70.92 | 63.65 | 88.16 | 84.02 | 76.44 |
| | ContFusion [20] | 0.06 | 84.58 | 72.33 | 67.50 | **93.84** | 86.10 | **82.00** |
| | MCF3D [16] | 0.16 | 84.11 | 75.19 | **74.23** | 88.82 | 86.11 | 79.31 |
| | Proposed (Ours) | 0.12 | **84.65** | **75.60** | 68.64 | 89.81 | **88.65** | 79.88 |
| Ped. | MV3D [2] | 0.36 | – | – | – | – | – | – |
| | AVOD [15] | 0.10 | – | 58.80 | – | – | – | – |
| | F-PointNet [17] | 0.17 | 70.00 | 61.32 | 53.59 | **72.38** | 66.39 | 59.57 |
| | ContFusion [20] | 0.06 | – | – | – | – | – | – |
| | MCF3D [16] | 0.16 | 68.54 | **64.93** | 59.47 | 68.56 | 64.98 | 59.55 |
| | Proposed (Ours) | 0.12 | **70.04** | 64.36 | **59.55** | 70.05 | **66.98** | **59.66** |
| Cyc. | MV3D [2] | 0.36 | – | – | – | – | – | – |
| | AVOD [15] | 0.10 | – | 49.70 | – | – | – | – |
| | F-PointNet [17] | 0.17 | 77.15 | 56.49 | **53.37** | **81.82** | **60.03** | **56.32** |
| | ContFusion [20] | 0.06 | – | – | – | – | – | – |
| | MCF3D [16] | 0.16 | 78.18 | 51.06 | 50.43 | 78.18 | 51.09 | 50.45 |
| | Proposed (Ours) | 0.12 | **79.19** | **57.53** | 50.99 | 79.19 | 57.30 | 50.99 |

To showcase the superiority of the deep dense fusion method, this paper compares its approach with the existing state-of-the-art fusion methods (MV3D [2], AVOD [15], F-pointNet [17], ContFusion [20], MCF3D [16]) based on the RGB images and point clouds as inputs only. Table 1 shows the comparing results on the 3D and BEV performance measured by the AP. According to KITTI's metric, the DDF3D increases 0.41% in 3D performance and 2.54% in BEV performance in the "Moderate" difficulty on the car class, respectively. For pedestrian/cyclist classes, DDF3D achieves 2.00% growth in BEV performance on the "Moderate" difficulty for pedestrian class and 1.04% growth in 3D performance on the "Moderate" difficulty for cyclist class. In the easy difficulty of 3D performance, DDF3D surpasses the second-best 1.50% for the pedestrian class and 1.01% for the cyclist, respectively. However, F-pointNet [17] is slightly better than DDF3D in BEV performance for cyclist. F-pointNet [17] utilizes the ImageNet weights to fine-tune its 2D detector, whereas DDF3D model is trained from scratch. Some 2D detection results in RGB images, 3D detection results are illustrated in Fig. 4.

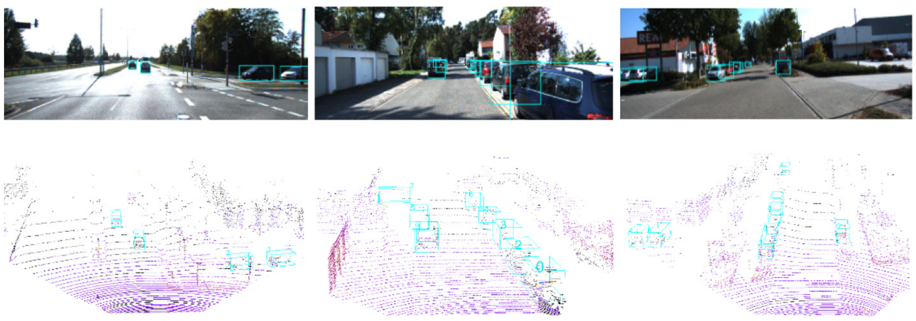

**Fig. 4.** Visualizations of DDF3D results on RGB images, point clouds.

## 3.3 Ablation Study

To analyze the effects of optimal deep dense fusion, an ablation is conducted on KITTI's validation subset with massive experiments. Table 2 shows the effect of varying different combinations of the deep dense fusion method on the performance measured by the AP. As shown in Fig. 2, Each encoder has 4 convolution blocks in order: Conv1, Conv2, Conv3, Conv4. Each decoder also has 4 deconvolution blocks in order: Deconv1, Deconv2, Deconv3, Deconv4. To ensure the DDF3D high efficiency, the combinations of deep dense fusion are only designed shown in Table 2.

**Table 2.** Ablation study for the combinations of the deep dense method on KITTI's validation subset. All results are in moderate difficulty in the car class.

| Deep dense fusion | | | | | 2D | 3D | BEV |
|---|---|---|---|---|---|---|---|
| Deconv4 | Conv4 | Conv3 | Conv2 | Conv1 | | | |
| | | | | | 86.93 | 72.43 | 85.78 |
| √ | | | | | 88.11 | 72.73 | 86.06 |
| √ | √ | | | | 88.28 | 74.02 | 86.25 |
| √ | √ | √ | | | **88.96** | **75.60** | **88.65** |
| √ | √ | √ | √ | | 88.57 | 73.55 | 86.08 |
| √ | √ | √ | √ | √ | 87.44 | 72.18 | 86.49 |

To explore the effects of fusion method in different directions, two more sets of experiments are conducted based on the best combinations in Table 2. The first set of the experiment only projects features from BEV to the camera view. In contrast, the second set of the experiment only projects feature from camera view to BEV. Table 3 demonstrates that two-way fusion method is better than one-way fusion. The effect of different fusion methods on it is very limited for 2D and BEV performance, but they have a significant impact on the accuracy of 3D detection.

**Table 3.** Ablation study for the fusion method in different directions. **B2C** means the one-way fusion from BEV to the camera view. **C2B** means the one-way fusion from the camera view to BEV. **Both** mean bidirectional fusions.

| Method | 2D (%) | | | 3D (%) | | | BEV (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | E | M | H | E | M | H | E | M | H |
| B2C | 90.00 | 87.97 | 86.33 | 83.12 | 74.18 | 68.08 | 89.30 | 85.17 | 78.84 |
| C2B | 89.67 | 87.07 | 86.15 | 82.08 | 72.56 | 66.96 | 88.51 | 84.98 | 78.77 |
| Both | **90.33** | **88.96** | **88.19** | **84.65** | **75.60** | **68.64** | **89.81** | **88.65** | **79.88** |

Besides, the DDF3D model converges faster and the experimental values keep steadily after 80K iterations. Based on the attribute, the model can be checked good or not good within 12 h. Figure 5 shows the evaluation results are extracted every 5K iterations from 80K iterations to 120K iterations on the validation subset.

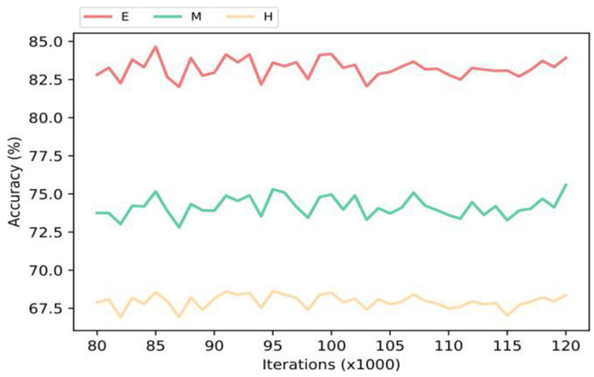

**Fig. 5.** 3D detection accuracy of DDF3D for car class from 80K iterations to 120K iterations. The *light coral color*, *medium aquamarine color*, and *Navajo white color* denote the *Easy*, *Moderate*, *Hard* difficulty respectively. (Color figure online)

## 4 Conclusion

This work proposed DDF3D, a full fusion 3D detection architecture. The proposed architecture takes full consideration of the mutual advantages of RGB images and point clouds in the feature extraction phase. The deep dense fusion is two-directional fusion at the same time. A high-resolution feature extractor with the full fusion features, the proposed architecture greatly improves 3D detection accuracy, specifically for small objects. Massive experiments on the KITTI object detection dataset, DDF3D outperforms the state-of-the-art existing method in among of 2D, 3D, and BEV.

## References

1. Li, B., Zhang, T., Xia, T.: Vehicle detection from 3D lidar using fully convolutional network. In: Robotics: Science and Systems XII (2016)
2. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6526–6534 (2017)
3. Caltagirone, L., Scheidegger, S., Svensson, L., Wahde, M.: Fast LIDAR-based road detection using fully convolutional neural networks. In: 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, pp. 1019–1024 (2017)
4. Yang, B., Luo, W., Urtasun, R.: PIXOR: real-time 3D object detection from point clouds. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, pp. 7652–7660 (2018)
5. Zhou, Y., Tuzel, O.: VoxelNet: end-to-end learning for point cloud based 3D object detection. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4490–4499 (2018)
6. Wen, L., Jo, K.-H.: Fully convolutional neural networks for 3D vehicle detection based on point clouds. In: Huang, D.-S., Jo, K.-H., Huang, Z.-K. (eds.) ICIC 2019. LNCS, vol. 11644, pp. 592–601. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26969-2_56
7. Yan, Y., Mao, Y., Li, B.: Second: sparsely embedded convolutional detection. Sensors **18**, 3337 (2018)
8. Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J.: Pointnet: deep learning on point sets for 3d classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77–85 (2017)
9. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems, vol. 30, pp. 5099–5108 (2017)
10. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: PointPillars: fast encoders for object detection from point clouds. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, pp. 12689–12697 (2019)
11. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3D object detection for autonomous driving. In: IEEE CVPR (2016)
12. Xu, B., Chen, Z.: Multi-level fusion based 3D object detection from monocular images. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, pp. 2345–2353 (2018)

13. He, T., Soatto, S.: Mono3D++: Monocular 3D vehicle detection with two-scale 3D hypotheses and task priors. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8409–8416, July 2019
14. Li, P., Chen, X., Shen, S.: Stereo R-CNN based 3d object detection for autonomous driving. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7636–7644 (2019)
15. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3D proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–8 (2018)
16. Wang, J., Zhu, M., Sun, D., Wang, B., Gao, W., Wei, H.: MCF3D: multi-stage complementary fusion for multi-sensor 3D object detection. IEEE Access **7**, 90801–90814 (2019)
17. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3D object detection from RGB-D data. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 918–927 (2018)
18. Xu, D., Anguelov, D., Jain, A.: PointFusion: deep sensor fusion for 3D bounding box estimation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, pp. 244–253 (2018)
19. Du, X., Ang, M.H., Karaman, S., Rus, D.: A general pipeline for 3D detection of vehicles. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 3194–3200 (2018)
20. Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3D object detection. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11220, pp. 663–678. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01270-0_39
21. Girshick, R., Fast R-CNN. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015)
22. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3D object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, pp. 3431–3440 (2015)
25. Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944 (2017)