remote sensing

MDPI

Article

FusionPillars: A 3D Object Detection Network with Cross-Fusion and Self-Fusion

Jing Zhang [1,2,3,*], Da Xu [1,2], Yunsong Li [1,2], Liping Zhao [4] and Rui Su [5]

[1] State Key Laboratory of Integrated Service Network, Xidian University, Xi'an 710071, China; 21011210308@stu.xidian.edu.cn (D.X.); ysli@mail.xidian.edu.cn (Y.L.)
[2] School of Telecommunication Engineering, Xidian University, Xi'an 710071, China
[3] Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China
[4] National Defense Science and Technology Innovation Research Institute, Beijing 100071, China; 17710268181@163.com
[5] Xi'an Termony Electronic Technology Co., Ltd., Xi'an 710031, China; surui@termony.com
* Correspondence: jingzhang@xidian.edu.cn; Tel.: +86-298-820-3116

Abstract: In the field of unmanned systems, cameras and LiDAR are important sensors that provide complementary information. However, the question of how to effectively fuse data from two different modalities has always been a great challenge. In this paper, inspired by the idea of deep fusion, we propose a one-stage end-to-end network named FusionPillars to fuse multisensor data (namely LiDAR point cloud and camera images). It includes three branches: a point-based branch, a voxel-based branch, and an image-based branch. We design two modules to enhance the voxel-wise features in the pseudo-image: the Set Abstraction Self (SAS) fusion module and the Pseudo View Cross (PVC) fusion module. For the data from a single sensor, by considering the relationship between the point-wise and voxel-wise features, the SAS fusion module self-fuses the point-based branch and the voxel-based branch to enhance the spatial information of the pseudo-image. For the data from two sensors, through the transformation of the images' view, the PVC fusion module introduces the RGB information as auxiliary information and cross-fuses the pseudo-image and RGB image of different scales to supplement the color information of the pseudo-image. Experimental results revealed that, compared to existing current one-stage fusion networks, FusionPillars yield superior performance, with a considerable improvement in the detection precision for small objects.

Keywords: object detection; point cloud; computer vision

Citation: Zhang, J.; Xu, D.; Li, Y.; Zhao, L.; Su R. FusionPillars: A 3D Object Detection Network with Cross-Fusion and Self-Fusion. *Remote Sens.* **2023**, *15*, 2692. https://doi.org/10.3390/rs15102692

Academic Editor: Dong Liu

Received: 25 April 2023
Revised: 19 May 2023
Accepted: 20 May 2023
Published: 22 May 2023

1. Introduction

In recent years, with the improvement of the average precision of LiDAR, the application of LiDAR in cars and robots has been receiving improved research interest (Figure 1). Consequently, the question of how to use LiDAR for target detection has become an important research topic.

LiDAR has its own numbering and a fixed vertical angle. To obtain a full range of environmental information, LiDAR performs rotational motion at a constant angular speed and emits lasers to gather information through the reflected points. In addition to the distance of the reflected points, information regarding the occurrence time and horizontal angle (Azimuth) is recorded. In this manner, the coordinates of all reflected points form a point cloud.

As 3D (three-dimensional) data, the point cloud includes both depth and geometric information. However, due to its unordered, sparse, and non-uniform distribution, it is a special challenge to efficiently use this information for object detection.

Based on the aforementioned characteristics of the point cloud, its object detection methods can be classified into three categories: (1) point-based methods [1–4], (2) voxel-based methods [5–8], and (3) projection-based methods [9–14].

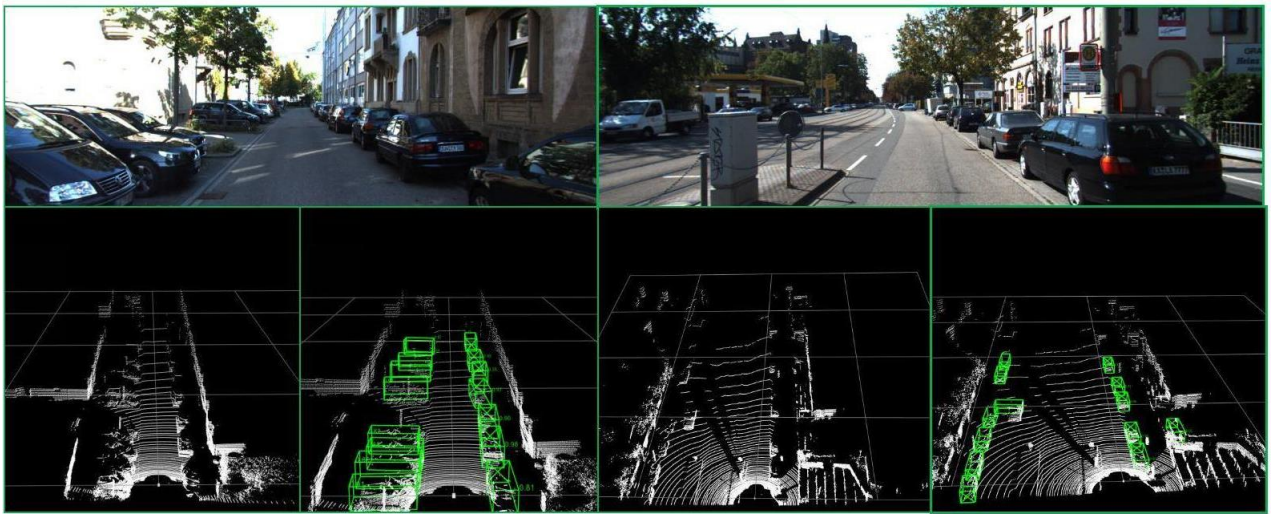


Figure 1. Visualization of detection results. The top of the figure is the RGB image from the camera. The lower left part of the figure shows the original point cloud. The lower right part of the figure shows the detection results. The experiment is based on the KITTI dataset [15].

However, a point cloud is less effective in detecting distant objects and small objects because of its missing color information. Thus, researchers have considered fusing RGB images into a point cloud to improve detection precision. According to the different fusion levels for the objects, the fusion strategies can be classified into three categories: (1) early fusion [16–20], (2) deep fusion [21–24], and (3) late fusion [25–27].

Currently, the detection of 3D objects is focused on large-scale objects, such as cars. As large-scale objects, cars have more feature points and are easier to detect. However, in practical application scenarios, small objects (pedestrians and cyclists) are the most easily ignored because of factors such as occlusion and distance.

The existing fusion network also presents some problems in need of improvement. F-pointnet [28] does not make use of point cloud information, so the effects of illumination and occlusion make the missed detection more serious. PMPF [20] is still ineffective in finding the relationship between pixels and point cloud key points. The proposed method in [14] lacks the use of multi-scale intermediate feature maps, which makes it impossible to effectively use the context semantic information. Consequently, we hope for a strategy to efficiently fuse the information of point cloud and RGB images and utilize multi-scale information, thereby improving detection average precision (AP).

In this study, inspired by the deep fusion strategy and oriented to practical industrial needs, we propose a novel one-stage small model 3D object detection network, FusionPillars. It is designed to improve the detection precision of small objects by supplementing the spatial information and color information of the pseudo-image through cross-fusion and self-fusion. FusionPillars proposes two fusion modules: the Set Abstraction Self (SAS) fusion module and Pseudo View Cross (PVC) fusion module.

In the point-based branch, the most direct motivation of the SAS fusion module is to exploit the advantages of the voxel-based method and the point-based method. Point-based methods enable the complete utilization of structural information of the object. Voxel-based methods employ spatial convolution to extract multi-scale and multi-level local feature information efficiently. At the same time, the SAS fusion module effectively reduces the impact of the disadvantages of the two methods. Set Abstraction [2] down-samples point clouds to effectively reduce memory resource consumption. Furthermore, the self-fusion of two different point cloud representations compensates for the lack of voxel-wise features (pseudo-image) and enables features to more strongly represent the spatial information of small objects.

In the image-based branch, to utilize the color information from RGB image and the depth information from the point cloud, the PVC module cross-fuses the data of the point

cloud and the image. Specifically, through the pseudo-conversion of the images' view, we realize the registration of the pseudo-image and the RGB image; then, cross-fusion is used to supplement the color information of the image into the pseudo-image. Now, the pseudo-image contains both depth information and color information.

In summary, the contributions of this study are as follows:

- The SAS fusion module performs self-fusion by using the point-based method and voxel-based method to strengthen the spatial expression of the pseudo-image.
- The PVC fusion module cross-fuses the pseudo-image and the RGB image through the pseudo-conversion of the view angle to strengthen the color expression of the pseudo-image.
- We aggregate the SAS and PVC modules in the proposed network called FusionPillars, a one-stage end-to-end trainable modal that performs well on the KITTI dataset, with a particularly pronounced improvement in detection precision for small objects.

2. Materials

2.1. Point-Based Methods

Point-based methods directly operate on the raw point cloud. First, a point cloud is converted to sparse representations, and subsequently low-dimensional feature vectors are extracted. Next, these vectors are then aggregated into larger, more complex high-dimensional features. The main advantage of this method is that the structural information of the object is retained intact. Pointnet [1] has designed a highly efficient and effective architecture for solving three key issues: unordered interaction, interaction among points, and invariance under transformations, laying the foundation for subsequent research work [2–4]. PointRCNN [29] is a bottom-up 3D object detection network. It first splits the point cloud into foreground and background to generate a small amount of high-quality 3D proposals, and then combines semantic and local spatial features to predict box coordinates. Instead of using upsampling layers, 3DSSD [30] employs a fusion sampling strategy based on the distance and features and uses a candidate generation layer to generate key points, which are input into an anchor-free regression head to predict the 3D object box.

2.2. Voxel-Based Methods

Voxel-based methods convert the point cloud into regular discrete voxel representations, and then apply convolutional neural networks to predict object categories and 3D bounding boxes. Voxelization enables point clouds to be stored in memory in order, which is conducive to reducing random memory access and increasing the efficiency of data operation. This advantage enables the network to perform better when processing magnitude point clouds. Vote3deep [5] converts point cloud data into hand-crafted voxels for real-time 3D inspection. However, such voxels are not suitable for complex real-world environments. VoxelNet [6] is a voxel-based network framework that can be trained end-to-end. A sparse 4D tensor is encoded by dividing the point cloud into equal voxels. The convolutional middle layers and a Region Proposal Network [31] processes the 4D tensor to generate the 3D detection. SECOND [7] further overcomes the computational barrier of VoxelNet by applying sparse convolution. It also proposes a new loss function and a new data enhancement method. PointPillars [8] learns point cloud features contained in point pillars in the z-direction by PointNet, and encodes these features as a pseudo-image. Then, it employs a well-established 2D image object detection process to predict 3D bounding boxes. Voxel-RCNN [32] has designed a Voxel RoI (Region of Interest) pooling on the basis of PV-RCNN [33], which directly extracts RoI features from voxel features for further refinement.

2.3. Lidar-Camera Fusion Methods

The RGB image (R, G, B) contains color information but lacks depth information. In contrast, the point cloud (x, y, z, r) has depth information but lacks color information. Therefore, a key challenge for fusion methods is how to fuse the color and depth information

by using fusion methods. According to the different fusion objects, fusion strategies can be classified into three categories: early fusion, deep fusion, and late fusion. Early fusion mainly focuses on data-level fusion. It fuses data from different modalities into a single feature vector before inputting them for subsequent operations. The data can be either raw data from the sensors or pre-processed data. Pointpainting [17] adds a new dimension to point cloud data by fusing them with semantic segmentation scores after matrix projection. Deep fusion mainly focuses on feature-level fusion. The raw data of different modalities are first transformed into a high-dimensional feature representation after extracting features, and then some interactive fusion operations are done in different feature layers. EPNet [21] extracts features from the point cloud and image several times and then fuses the features at different scales to improve the corresponding point-wise feature representation. Late fusion mainly focuses on decision-level fusion. The raw data from different modalities are processed with their respective networks to output classification scores, which are fused at the score. CLOCS [27] processes 2D and 3D candidates by converting them into consistent joint detection candidates based on geometric and semantic consistencies.

3. Methods

In this section, we show the specific details of FusionPillars. As shown in the Figure 2, our network architecture consists of three main sub-networks:

1. Feature Extraction Network: it is the preprocessing network for point cloud voxelization.
2. Dual-fusion Backbone: it fuses feature information from secondary branches into features of primary branches.
3. Detection Head: it performs the concatenation operation for feature maps to generate the final feature map and outputs the label and bounding box of the object.

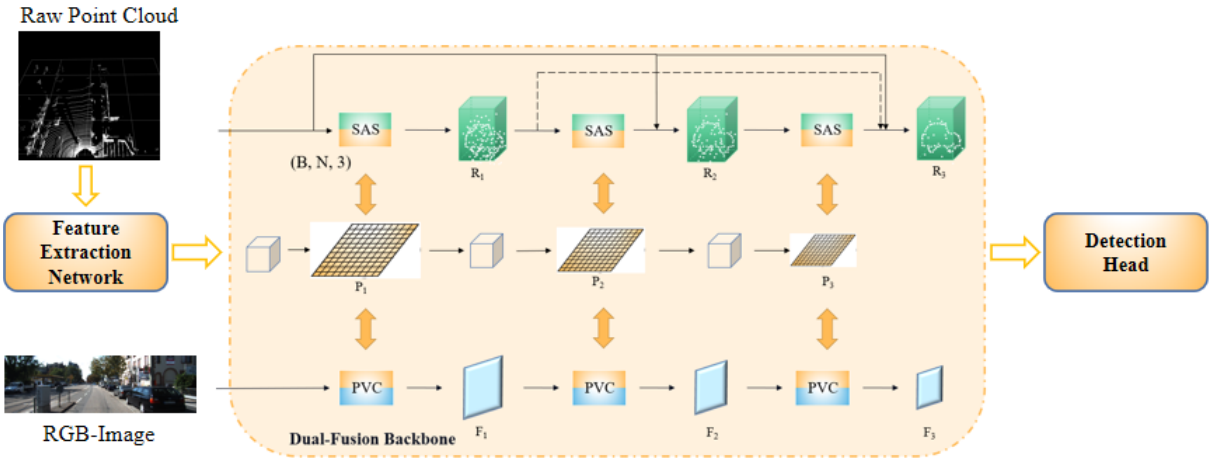


Figure 2. Illustration of the architecture of FusionPillars, which is composed of a point-based branch, a voxel-based branch, and an image-based branch. We employ the SAS fusion module and PVC fusion module to enhance the feature expression of the pseudo-image.

3.1. Feature Extraction Network

To utilize the relationship between features of different heights, we adopt an improved pseudo-image generation network [34] as Feature Extraction Network: at height dimension, a pillar is equally divided into four smaller pillars, and the final pseudo-image is generated after height-dimension and channel-dimension attention to retain the more expressive point cloud spatial information (Figure 3). The process is described as follows:

- The point cloud (x, y, z, r) is separated using a uniform grid network with a size of 0.16 m^2 in the x-y direction. With the grid network as the bottom and the point cloud height ($4 * 1$ m) as the height, the point cloud space is divided into P pillars.
- The arithmetic mean (x_c, y_c, z_c) and the offset (x_p, y_p) from the central point in the x-y direction are calculated, and then the coordinate data ($D = 4$ dimensional) is aug-

mented. Now, the augmented coordinate data are $D = 9$ dimensional $(x, y, z, r, x_c, y_c, z_c, x_p, y_p)$.

- The sparsity of the point cloud results in an uneven distribution of the point cloud, which results in a large number of empty pillars. Thus, a threshold is set to randomly sample the pillars with an excessive amount of points, whereas the pillars with too few points are operated zero-padding. In this manner, dense tensors $(D, P, N, 4)$ are created, where N represents the number of points in each pillar.
- Features are learned by using simplified PointNet. Specifically, tensors (C, P, N, B) are generated by processing each point through a linear-layer, BatchNorm [35], and ReLU [36]; then, a maximum pool operation is operated to obtain tensors (C, P, B). B represents the number of batches.
- The features are encoded and scattered back to the locations of the original pillars to create B pseudo-images of size (C, H, W, B), where H and W indicate the height and width of the pseudo-image.
- B pseudo-images (C, H, W, B) are fed into two attention sub-modules to calculate the height-attention weight S and the channel-attention weight T. The final pseudo-image (C, H, W) is then obtained by performing operations such as multiplication and maximum pooling.

$$S = W_2\delta(W_1F) \tag{1}$$

$$T = W_4\delta(W_3F) \tag{2}$$

W_i indicates fully connected layer, F indicates 4 pseudo-images.

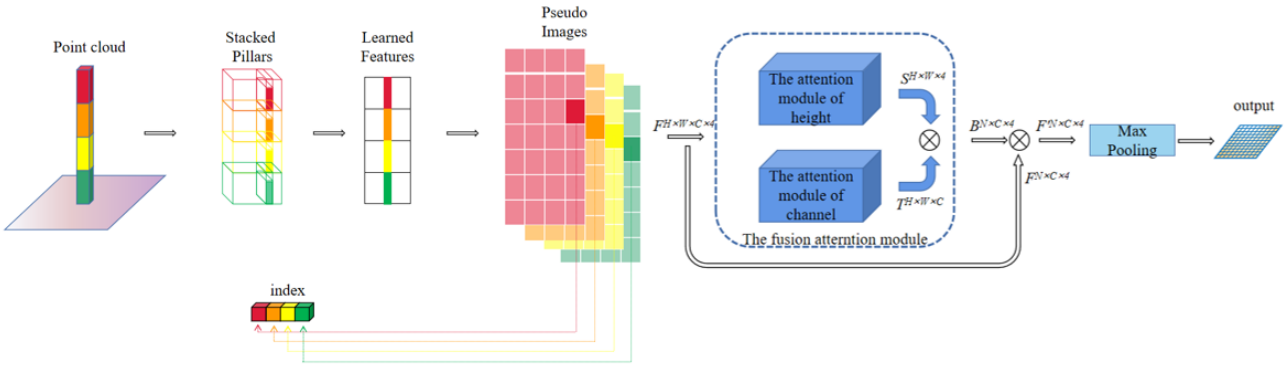


Figure 3. Illustration of the architecture of the the Feature Extraction Network.

3.2. Dual-Fusion Backbone

In the Dual-Fusion Backbone, the input is the raw point cloud and RGB image. We designed a Set Abstraction Self (SAS) fusion module (Figure 4) to fuse two feature representations of a point cloud, and a Pseudo View Cross (PVC) fusion module (Figure 5) to fuse point cloud and RGB images. Moreover, based on the two modules, three branches are designed to utilize multi-scale information (Figure 2): a point-based branch and a voxel-based branch for point cloud, and an image-based branch for RGB images, respectively. Among them, the voxel-based branch acts as the primary branch and other branches as secondary branches. The SAS module is designed with the motivation to integrate the spatial information of the point-based branch into the pseudo-images of the voxel-based branch. The PVC module is designed with the motivation to integrate the color information of the image-based branch into the pseudo-images of the voxel-based branch.

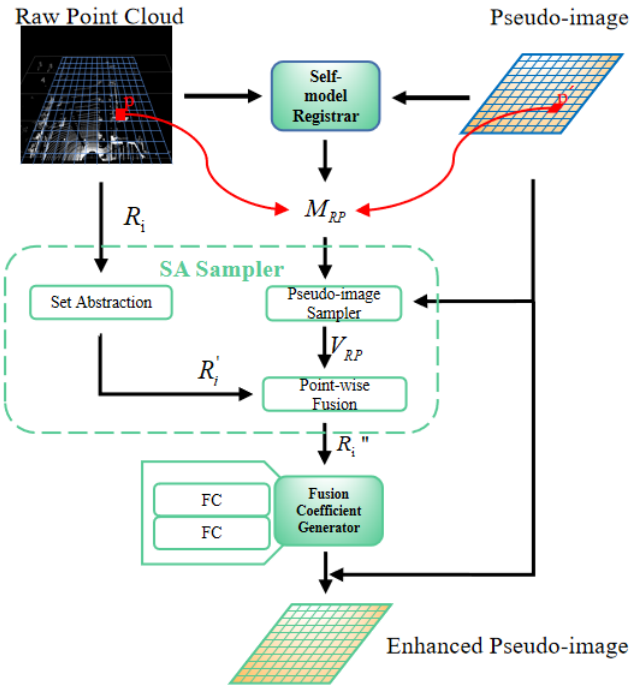


Figure 4. Illustration of the architecture of the SAS fusion module.

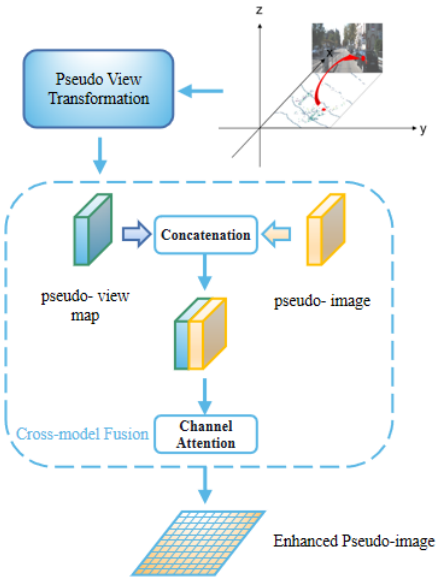


Figure 5. Illustration of the architecture of the PVC fusion module.

3.2.1. Voxel-Based Branch

The voxelized pseudo-images allow irregular and huge point clouds to be stored in an orderly manner and use efficient convolution, ensuring the computational efficiency of the algorithm. The voxel-based branch is the main branch of the dual-fusion backbone, which receives the pseudo-image from the Feature Extraction Network as input. In the feature extraction network, the point cloud is divided into voxels in a spatial division, and a voxel may contain multiple points within it. Then local features are extracted from the point groups within voxels instead of extracting features for each point, which reduces the memory load on subsequent algorithms while ensuring that the pseudo-images contain most of the information of the point cloud.

The voxel-based branch utilizes three lightweight convolution blocks to successively hierarchically generate the feature map P_i with different resolutions. The multi-scale and multi-level local feature information is extracted and utilized by the lightweight convolution

module. Each convolution block comprises a $3 * 3$ convolution layer, a BatchNorm, and a ReLU activation function. The feature maps P_i with different resolutions are sent to the SAS fusion module and PVC fusion module to obtain the enhanced feature maps P_i'.

3.2.2. Point-Based Branch

The raw point cloud contains all three-dimensional features of objects. However, the volume of point cloud data is huge, and direct processing has high hardware requirements and a high time cost. The point-based branch directly operates the raw point cloud and extracts the spatial information to enhance the spatial representation of the pseudo-image.

Specifically, the point-based branch directly extracts features of the point cloud with the idea of hierarchical fashion. We accomplish point cloud refinement and self-fusion with the pseudo-image P_i by three SAS fusion modules. Point cloud refinement selects key points in small neighborhoods and samples them to reduce computational effort, achieving efficiency. Self-fusion extracts and assigns spatial information to the corresponding feature map of the voxel-based branch.

For description, the output of every SAS fusion module is recorded as $R_i (i = 0, 1, 2)$. In addition, under the premise of ensuring the maximum information transmission between point-wise feature R_i in the network, all point-wise features are directly connected. In order to ensure the characteristics of feed-forward, each point-wise feature combines the inputs of all the previous point-wise features, and then passes the output feature to all subsequent modules. Dense connections enable feature reuse, which further enriches feature map information.

3.2.3. Image-Based Branch

Although the camera is vulnerable to the external environment, the RGB images taken by the camera contain the color information of objects, which is also essential and significant for object detection task.

The image-based branch operates the RGB images. Similarly, to extract and fuse semantic information of the images, we employ three PVC fusion modules to the input with the idea of hierarchical processing. In the PVC fusion module, we implement the pseudo-transformation of the images' view and supplement color information for the pseudo-image after registration between the RGB image and the pseudo-image. As can be seen in Figure 2, we record the pseudo-view map of the image as F_i.

3.2.4. Set Abstraction Self (SAS) Fusion Module

Self-fusion refers to the fusion of different feature representations (Voxel and point) within the point cloud. So, we proposed the Set Abstraction Self (SAS) fusion module.

The LIDAR-guided SAS fusion module (Figure 4) refines the raw point cloud and completes the self-fusion of enhanced spatial information by establishing correspondence. It consists of a Self-modal Registrar, a SA Sampler, and a Fusion Coefficient Generator.

Self-modal Registrar establishes firstly the correspondence relationship between pseudo-pixels (pixels of the pseudo-image) and points in space. Two feature representations (point-based feature R_i, pseudo-image feature P_i) are registered to generate an self-modal mapping matrix M_{RP}. In more detail, in the self-modal registrar, for a particular point in the raw point cloud and its corresponding position in the pseudo-image, the self-modal mapping matrix M_{RP} is obtained as follows:

$$p' = M_{RP} * p \tag{3}$$

SA Sampler then takes the point-based feature R_i and the self-modal mapping matrix M_{RP} as inputs to output new point-based feature R_i'' for the next stage.

To avoid the huge amount of operations, the raw point cloud R_i is subjected to the set abstraction operation to realize local feature extraction, generating point-based feature R_i'.

After the correspondence is established, the sampling position p' and the self-modal mapping matrix M_{RP} are used as inputs of the SA sampler. The pseudo-image sampler

generates a point-based feature representation V_{RP} for the point-based feature. Because the sampling locations may lie between adjacent pixels, we apply bilinear interpolation to image features on continuous coordinates, i.e.:

$$V_{RP} = K(F(^{N(p')})) \tag{4}$$

where V_{RP} is the corresponding feature representation of the point p', K indicates the bilinear interpolation function, and $F(^{N(p')})$ indicates the feature of the neighboring pixels at the sampling position p'.

The feature representation V_{RP} and the point-based feature R_i' are fused point by point. In the Point-wise Fusion, we first perform a concatenation operation for V_{RP} and R_i, and then, enter a fully connected layer, adjust it with a $1 * 1$ convolution, and then normalize it to the range of [0, 1] following the BN and ReLU weight graph w. Finally, new point-wise features R_i'' are obtained.

$$R_i'' = \sigma(Wtanh(UV_{RP} + VR_i)) \tag{5}$$

where W,U,and V denote the learnable weight matrices. σ represents the sigmoid activation function.

Fusion Coefficient Generator generates fusion coefficients for point-based features R_i'' finally. It uses two fully connected layers to generate spatial enhancement coefficients. The fusion coefficient is then multiplied with the pseudo-image to obtain the enhanced pseudo image P_i'.

The obtained spatial enhancement coefficient is accumulated with the pseudo-image to obtain the enhanced pseudo image P_i'.

$$P_i' = P_i * FC(FC(R_i'')) \tag{6}$$

The SAS module uses a pseudo-image and raw point cloud to achieve efficient fusion in single modal data so that the pseudo-image has more comprehensive spatial information.

3.2.5. Pseudo View Cross (PVC) Fusion Module

Cross-fusion refers to the fusion of point cloud and RGB images. So, we propose a Pseudo View Cross (PVC) fusion module, which consists of a Pseudo View Transformation and a Cross modal Fusion. The PVC module (Figure 5) provides an effective solution for determining the differences between the RGB image and pseudo-image.

Most voxel-based methods densely divide the space and generate feature maps to ensure high-quality detection. The pseudo-image is created by dividing the grid in the x-y plane and squeezing the z-oriented points. So, the pseudo-image shows a vertical view of the scene. In contrast, combined with the actual scene and camera characteristics, the RGB image shows a frontal view of the current scene. This makes it necessary to fully consider the spatial relationship between the two in the process of fusing feature maps. For this reason, we aim to perform cross-view fusion based on the spatial overlap relationship between the front view and the vertical view.

Specifically, because of the characteristics of LIDAR and camera, objects exhibit spatial correspondence in each pair of images and pseudo images. Based on this fact, First, through the Pseudo View Transformation (Figure 6), a pseudo-view map F_i of the image is obtained pixel by pixel, which size is the same as that of the pseudo-image.

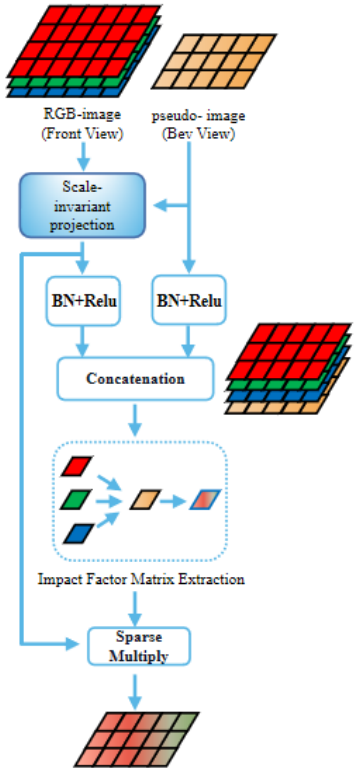


Figure 6. Illustration of the architecture of the Pseudo View Transformation.

Pseudo View Transformation generates pseudo-view mapping by pseudo-view transformation matrix. First, based on the characterization of the LIDAR and the camera, the input RGB image is scaled to be equal in size to the pseudo-image by performing a scale-invariant projection. Subsequently, after BatchNorm and Relu operations are performed in order to attenuate the differences in the numerical meanings of the two images (RGB image and pseudo-image) within the computer. Then, the two images are executed as a concatenation operation. Finally, the following equation is used to extract the pseudo-view transformation matrix M_cs and generate the pseudo-view map F_i mapped to the BEV view with the resized RGB image.

$$M_cs = 0.4 * M_c + 0.6 * M_s; \tag{7}$$

where, we consider that the image receives the influence of channel dimension and spatial dimension in the pseudo-view transformation. Using the channel attention module and spatial attention module to generate the channel transformation influence matrix M_c and spatial transformation influence matrix M_s, respectively. After several experiments, we set the weighting coefficients as 0.4 and 0.6, respectively.

Cross modal Fusion is used to register and fuse the pseudo-view map with the pseudo-image to produce an enhanced pseudo-image. More carefully, the same size pseudo-view map F_i and pseudo-image P_i are concatenated in the channel dimension. Next, using Channel Attention, the image information is fused into the pseudo-image to produce an enhanced pseudo-image.

At this time, the pseudo-image is infused with color information, and the feature expression capability is further enhanced.

3.3. Detection Head

As shown in Figure 7, P_i provides semantic image information at different scales. Three parallel transposed convolutional layers with different steps are employed to recover the resolution of pseudo-images, obtaining three feature maps of the same size as P_1. These feature maps are performed for the concatenate operation and yield a more representative

feature graph P_U. Now, the P_U contains rich semantic image information with different receptive domains. Then, the P_U is sent to an adaptive detection head.

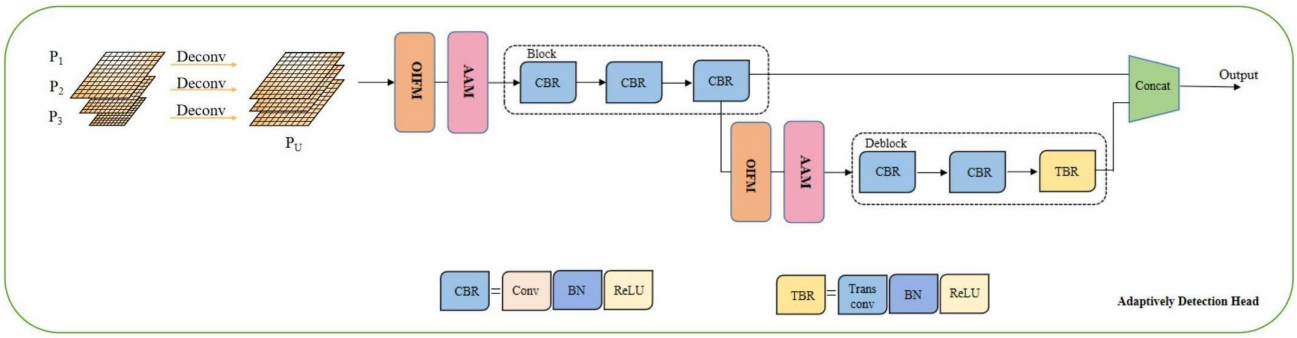


Figure 7. Illustration of the architecture of the Detection Head.

We employ an adaptive detection head [37], which consists of three parts: Head Backbone Network, Original Information Fusion Module (OIFM), and Adaptive Adjustment Module (AAM). It adaptively adjusts the sparse feature maps so that the network detail information is complemented, thus improving the object detection performance.

The Head Backbone Network is a top-down network that aggregates a Block module and a DeBlock module through concatenation operation, allowing the use of information at different scales. These multi-scale feature maps enhance the regression effect of object detection. The Original Information Fusion Module compensates for sparse feature maps by aggregating pseudo-image features. The Adaptive Adjustment Module emphasizes key information by adjusting the size of the feature maps.

3.4. Loss Function

We use the same loss function as the SECOND [7]. It further includes three different loss functions: regression loss, classification loss, and directional loss. The regression loss is used to accurately predict the position of three-dimensional boxes. The classification loss is used to determine the class of the object. Additionally, the directional loss is used to determine the direction of the object.

Regression loss function: In 3D object detection, parameters $(x, y, z, h, w, l, \theta)$ are used to define a 3D bounding boxes a 3D bounding box.The positioning residual coding of ground truth and anchors are defined below.

$$\begin{aligned} \Delta x = \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{d^a} \\ \Delta x = \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{d^a} \\ \Delta w = \log \frac{w^{gt}}{w^a}, \Delta l = \log \frac{l^{gt}}{l^a}, \Delta h = \log \frac{h^{gt}}{h^a} \\ \Delta \theta = \sin\left(\theta^{gt} - \theta^a\right), d^a = \sqrt{(w^a)^2 + (l^a)^2} \end{aligned} \tag{8}$$

The regression loss function can be expressed as:

$$L_{loc} = \sum_{b \in (x,y,z,w,l,h,\theta)} \text{SmoothL1}(\Delta b) \tag{9}$$

Classification loss function: We also use focal loss to deal with the class imbalance problem. The classification loss is defined as:

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

where p^a is the class probability of an anchor. We use the parameters mentioned in the paper: $a = 0.25$ and $\gamma = 2$.

Directional loss function: We use SoftMax as the directional loss function L_{dir} so that the network can learn the directional information of the bounding box. By combining all the loss functions, the total loss is defined as:

$$L = \frac{1}{N_{pos}}(\beta_{loc}L_{loc} + \beta_{cls}L_{cls} + \beta_{dir}L_{dir}) \tag{11}$$

where, N_{pos}represents the number of positive matching boxes, loss weights $\beta_{loc}, \beta_{cls}, \beta_{dir}$ are empirically set to 2.0, 1.0, and 0.2.

4. Experiment

In this section, we present the experimental data, including the experiment environment, the experimental dataset, the experimental settings, and the experimental results.

4.1. Experiment Environment

The following is the experimental environment:

1. CUDA: 10.2
2. Pytorch: 1.10.2
3. Python: 3.6
4. GPU: GeForce RTX 2080Ti

4.2. Experiment Dataset

The KITTI dataset [15] is the largest computer vision algorithm evaluation dataset for autonomous driving scenarios in the world. All of our experiments are based on the KITTI dataset. The KITTI dataset provides us with samples of lidar point cloud and images, which contains 7481 training samples and 7518 test samples. Among them, the training samples are divided into two parts, namely, the training set composed of 3712 samples and the validation set composed of 3769 samples. The dataset includes three categories: car, pedestrian, and cyclist. Additionally, based on the size, the occlusion level, and the truncation of each category, the dataset is stratified into three levels of difficulty level: easy, mod., and hard. Average Precision (AP) and mean Average Precision (mAP) are used as the evaluation indicators in this study. To ensure a fair comparison, we adopted a comprehensive evaluation scheme.

4.3. Experimental Settings

For the feature extraction network, we used the PointPillars setting to set the grid size to 0.4 m $*$ 0.4 m, the maximum number of pillars: P = 12,000, and the maximum number of points per pillar: $N = 100$. For the point-based branch, three set abstraction layers subsample the raw point cloud with sizes of 4096, 1024, and 256. For the voxel-based branch, the enhanced pseudo-images are down-sampled using the $3 * 3$ convolution, and the number of channels after convolution is set as 64, 128, and 256, successively. The number of transposed convolution output channels is 128.

The detection range is [(0, 70), $(-40, 40)$, and $(-3, 1)$]m, respectively. For the car, its anchor has a width, length, and height of $(1.6, 3.9, 1.5)$. Matching uses positive and negative thresholds of 0.6 and 0.45. For the pedestrian and the cyclist, their anchor sizes are (0.6, 0.8, 1.73) and (0.6, 1.76, 1.73), respectively. Additionally, 0.5 and 0.35 are the positive and negative matching thresholds they shared.

4.4. Experimental Results

In this section, our method is compared with typical methods based on the KITTI dataset. Our network detects three categories (cars, pedestrians, and bicycles) and obtains reports that contain different AP at three difficulty levels (easy, moderate, and hard).

4.5. Evaluation Indicators

The IoU indicates the coincidence degree between the network prediction object frame and the original real object frame:

$$IoU = \frac{Detectionresult \cap GroundTruth}{Detectionresult \cup GroundTruth} \tag{12}$$

In this study, the IoU threshold is set as three different values of 0.7, 0.5, and 0.25. On this foundation, the results can be classified into four categories: TP (True Positives), TN (True Negatives), FP (False Positives), FN (False Negatives).

Additionally, the precision represents the correct ratio of all predicted objects:

$$precision = \frac{TP}{TP + FP} \tag{13}$$

Recall is defined as the proportion of all positive samples in the test set that are correctly identified, i.e.,

$$recall = \frac{TP}{TP + FN} \tag{14}$$

AP is the area under the precision-recall curve. To verify the effectiveness of our algorithm, mAP is used as the main evaluation indicator in this paper to compare with existing algorithms.

4.5.1. Results with Single-Modal Networks

In Table 1, compared with the 3D detection networks of the single point cloud including MV3D [38], TANet [39], Point-GNN [40], and LSNet [41]. Our network achieves the best results in detecting small objects (pedestrians and cyclists). The mAP (mean Average Precision) of small objects under all difficulties has been improved greatly.

Table 1. The comparison with 3D detection network of single point cloud on the KITTI test set.

Benchmark	Network	Cars			Pedestrains			Cyclists		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
BEV	MV3D	66.77	52.73	51.31	N/A	N/A	N/A	N/A	N/A	N/A
	VoxelNet	89.35	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
	SECOND	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
	PointPillars	89.46	86.65	83.44	57.89	53.05	49.73	**82.36**	63.63	60.31
	PointRCNN	85.94	75.76	68.32	49.43	41.78	38.63	**73.93**	59.60	53.59
	H23D RCNN	92.85	88.87	86.07	58.14	50.43	46.72	82.76	67.90	60.49
	Point-GNN	93.11	89.17	83.90	55.36	47.07	44.61	81.17	67.28	59.67
	LSNet	92.12	85.89	80.80	N/A	N/A	N/A	N/A	N/A	N/A
	FusionPillars	**92.15**	**88.00**	**85.53**	**62.33**	**55.46**	**50.13**	**87.63**	**66.56**	**62.67**
3D	MV3D	71.09	62.35	55.12	N/A	N/A	N/A	N/A	N/A	N/A
	VoxelNet	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
	SECOND	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
	PointPillars	83.68	74.56	71.82	53.32	47.76	44.80	**71.82**	56.62	52.98
	PointRCNN	85.94	75.76	68.32	49.43	41.78	**38.63**	73.93	59.60	53.59
	TANet	83.81	75.38	67.66	54.92	46.67	42.42	73.84	59.86	53.46
	H23D RCNN	90.43	81.55	77.22	52.75	45.26	41.56	78.67	62.74	55.78
	Point-GNN	88.33	79.47	72.29	51.92	43.77	40.14	78.60	63.48	57.08
	LSNet	86.13	73.55	68.58	N/A	N/A	N/A	N/A	N/A	N/A
	FusionPillars	**86.96**	**75.74**	**73.03**	**55.87**	**48.42**	**45.42**	**80.62**	**59.43**	**55.76**

On the BEV benchmark, compared to the one-stage classic network, PointPillars, mAP is improved by an average of 2.66%. Among them, for small objects, the mAP is improved by 2.97% on average. Additionally, the mAP for cyclists with easy difficulty exhibits the maximum improvement (5.27%). Compared to the two-stage classic network PointRCNN, the mAP is improved by an average of 11.50%. Among them, for small objects, the mAP is improved by 11.30% on average. Additionally, the AP for cyclists with easy difficulty is improved the most (13.70%).

On the 3D benchmark, compared to the one-stage classic network, PointPillars, mAP is improved by an average of 2.65%. Among them, for small objects, the mAP is improved by 3.04% on average. Additionally, the performance of cyclists with easy difficulty is boosted from 71.82% to 80.62%. Compared to the two-stage classic network PointRCNN, AP is improved by an average of 3.81%. Among them, for small objects, the mAP is improved by 4.76% on average. Additionally, the performance for pedestrians with hard difficulty is boosted the most (6.79%).

In order to prove the innovation and effectiveness of FusionPillars, we made a further comparison with PointPillars on all benchmarks. Based on Table 2, we can see that our network has a particularly strong detection effect on small objects. The detection AP of pedestrians has improved by an average of 1.36%, and the maximum improvement is 4.44% (easy, BEV). The detection precision of cyclists has improved by an average of 3.81%, and the maximum improvement is 8.80% (easy, 3D).

Table 2. The comparison of pedestrians and cyclists on the all benchmarks.

Benchmark	Network	Pedestrians			Cyclists		
		Easy	**Mod.**	**Hard**	**Easy**	**Mod.**	**Hard**
BBOX	PointPillars	59.54	56.14	54.29	86.23	70.24	66.87
	FusionPillars	63.58	58.21	54.55	90.88	73.18	69.99
	Delta	**4.04**	**2.07**	**0.26**	**4.65**	**2.94**	**3.11**
BEV	PointPillars	57.89	53.05	49.73	82.36	63.63	60.31
	FusionPillars	62.33	55.46	50.13	87.63	66.56	62.67
	Delta	**4.44**	**2.41**	**0.40**	**5.27**	**2.93**	**2.36**
3D	PointPillars	53.32	47.76	44.80	71.82	56.62	52.98
	FusionPillars	55.87	48.42	45.42	80.62	59.43	55.76
	Delta	**2.55**	**0.66**	**0.62**	**8.80**	**2.81**	**2.78**
AOS	PointPillars	45.06	42.51	41.08	85.67	67.98	64.59
	FusionPillars	46.44	41.98	39.06	90.43	70.49	67.38
	Delta	**1.38**	**−0.53**	**−2.02**	**4.76**	**2.51**	**2.79**

4.5.2. Results with Multi-Modal Networks

FusionPillars is a one-stage multi-modal network. So, in this section, we compare FusionPillars with other fusion networks (LIDAR and RGB images).

Firstly, we compare it with other one-stage fusion networks (FusionRCNN [24], F-PointNet [28], HDNet [42], Cont-Fuse [43], and MVX-Net [44]) to demonstrate that FusionPillars are the better performing one-stage fusion network.

As can be seen from Table 3, FusionPillars achieves the best performance for various difficulty objects. Among them, compared with the latest algorithm, MVX-Net, its mAP is improved by 3.0%, 2.1%, and 7.4%, respectively, on the BEV benchmark; its mAP is improved by 3.8%, 3.0%, and 7.8%, respectively, on the 3D benchmark.

Table 4 shows the comparison results with the two-stage fusion networks (AVOD-FPN [9], IPOD [45], F-ConvNet [46], PointPainting [17], and $H^2$3D-RCNN [13]) under the BEV benchmark. The two-stage fusion network has one more refinement stage com-

pared to the one-stage network, including FusionPillars, and it has the advantage of high average precision and the disadvantage of slow speed. As can be seen from the table, although FusionPillars is only a one-stage network, it achieves the best detection of small objects in several indicators, and it is not weaker than most two-stage fusion networks for the detection of car objects.

Table 3. The comparison with 1-Stage multi-modal networks on the BEV and 3D benchmark.

Benchmark	Network	Easy	Mod.	Hard
BEV	F-PointNet	88.7	84	75.3
	HDNet	89.1	86.6	78.3
	Cont-Fuse	88.8	85.8	77.3
	MVX-Net	89.2	85.9	78.1
	FusionRCNN	89.9	86.45	79.32
	FusionPillars	**92.2**	**88.0**	**85.5**
3D	F-PointNet	81.2	70.4	62.2
	HDNet	N/A	N/A	N/A
	Cont-Fuse	82.5	66.2	64.0
	MVX-Net	83.2	72.7	65.2
	FusionPillars	**87.0**	**75.7**	**73.0**

Table 4. The comparison with 2-stage multi-modal networks on the BEV benchmark.

Network	Cars			Pedestrains			Cyclists		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D	86.62	78.93	69.8	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN	90.99	84.82	79.62	58.49	50.32	46.98	69.39	57.12	51.09
IPOD	89.64	84.62	79.96	60.88	49.79	45.43	78.19	59.4	51.38
F-ConvNet	89.69	83.08	74.56	58.9	50.48	46.72	82.59	68.62	60.62
PointPainting	92.45	88.11	83.36	58.7	49.93	46.29	83.91	71.54	62.97
$H^2$3D RCNN	92.85	88.87	86.07	58.14	50.43	46.72	82.76	67.90	60.49
FusionPillars	**92.15**	**88.00**	**85.53**	**62.33**	**55.46**	**50.13**	**87.63**	**66.56**	**62.67**

4.6. Ablation Studies

We have conducted extensive experiments on the 3D benchmark of the KITTI dataset. We verify the validity of SAS modules and PVC modules by comparing mAP. The result of PointPillars is set as a baseline.

As can be seen from Table 5, on the 3D benchmark, SAS modules improve the mAP of individual objects. Compared to the baseline, the mAP of pedestrians is boosted by 0.27%, and the mAP of cyclists is boosted by 0.45%. In the SAS module, we also design dense connections. Combined with dense connections, the detection capability of SAS modules is further strengthened, with mAP for cars, pedestrians, and cyclists increasing to 76.75%, 48.96%, and 61.52%, respectively.

In the case of a single PVC module, compared with the baseline, our network improves the mAP of three types of objects from 0.01% to 2.09%. The detection mAP of cyclists is significantly improved (2.09%), and that of pedestrians is improved by 0.39%.

After the combination of the two modules, all the detection results have been further boosted. Finally, the mAP of FusionPillars for three types of objects are boosted to 77.42%, 49.12%, and 63.95%, respectively.

Table 5. Ablation experiments on the 3D benchmark.

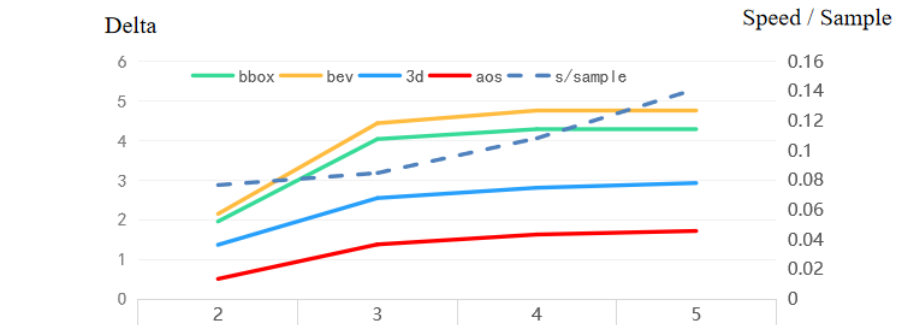
SAS	Dense	PVC	Cars	Pedestrains	Cyclists
✘	✘	✘	76.68	48.62	60.47
✔	✘	✘	76.71	48.89	60.92
✔	✔	✘	76.75	48.96	61.52
✘	✘	✔	76.69	49.01	62.56
✔	✔	✔	77.86	49.37	63.95

To verify the effectiveness of the improved feature extraction network and detection head in FusionPillars, we conducted additional ablation studies (Table 6) based on Table 5. Where ✘ represents the use of the corresponding module in the original PointPillars, and ✔ represents the use of the improved module from other papers. The deployment of improved network architecture improved by 0.72%, 0.54%, and 1.32% in each of the three object categories.

Table 6. Ablation experiments for network architecture.

Fea. Ext. Net.	Dua. Bac.	Det. Hea.	Car.	Ped.	Cyc.
✘	✔	✘	77.86	49.37	63.95
✔	✔	✘	77.88	49.52	64.47
✘	✔	✔	77.93	49.47	64.81
✔	✔	✔	78.58	49.91	65.27

SAS modules and PVC modules can be deployed at different depths of the feature map, which means that the network can continuously deepen the pyramid backbone structure as long as the resolution allows. Figure 8 analyzes the detection speed and improvement of AP by different layers. We choose (pedestrians, easy) as an indicator to analyze the impact of pyramid layers on the network. The four solid lines represent the detection AP under the four benchmarks (BBOX, BEV, 3D, AOS), and the dotted line represents the time to detect each sample. From the trend of solid line and dotted line, it can be seen that from the third layer, although increasing the number of layers can improve the detection AP, the improvement is limited and tends to be flat; at the same time, the time consuming for detecting each sample increase significantly. The reasons are analyzed as follows. The detection of large objects depends on the small-scale feature map at the pyramid backbone, because the small-scale feature map contains more obvious feature information of large objects. Unlike the big objects, most of the semantic information of small objects is stored in the large feature map. The smaller the feature map is, the weaker the small object is. Therefore, in order to give consideration to the detection AP and speed, we set the number of pyramid layers of the branch to 3.



	2	3	4	5
bbox	1.96	4.04	4.29	4.31
bev	2.15	4.44	4.76	4.79
3d	1.37	2.55	2.81	2.93
aos	0.51	1.38	1.63	1.72
s/sample	0.0768	0.0849	0.1083	0.1415

Figure 8. The impact of different layers.

4.7. Effectiveness Analysis

This subsection analyzes the effectiveness of the method by analyzing comparative and ablation experiments.

The motivation of the method is supplementing object features and improving the ability to expressions in order to expect to improve small object detection average precision. As can be seen from Tables 1 and 4, we not only have particularly significant improvement in small objects, but also the detection average precision of large objects is guaranteed. Among them, there are some results that we did not achieve the best performance. The reason for this is that some networks are two-stage networks. The two-stage networks refine the proposal in the second stage to obtain the final detection results, and although the detection average precision is improved, the detection speed becomes slower. Our network, FusionPillars, is a one-stage network that is only faster, but also performs better.

The comparison experiment (Table 2) between FusionPillars and baseline illustrates that the introduction of multi-modality indeed complements the features of the object, improves the information representation of the feature map, and achieves the improvement of small object detection average precision.

Comparative experiments (Table 3) with other classical 1-stage fusion networks illustrate the superior performance of FusionPillars.

In addition, the progressive improvement of the ablation studies (Table 5) demonstrates the effectiveness of the two main fusion modules (PVC module and SAS module) operating independently and operating jointly for the network. The experimental data in Table 6 show the effectiveness of introducing the improved network.

5. Conclusions

In this paper, we propose a novel one-stage multi-modal detection network, FusionPillars. It consists of a feature extraction network, a dual-fusion backbone, and a detection head. The feature extraction network preprocesses the raw point cloud to generate a pseudo-image. The dual-fusion backbone includes the SAS module and the PVC fusion module. The SAS Fusion Module self-fuses point-wise features and voxel-wise features, enhancing the spatial representation of the pseudo-image. The PVC fusion module cross-fuses the pseudo-image and RGB image, and the semantic information of the RGB image is given to the pseudo-image so that the pseudo-image has color expression capability. These two modules effectively enhance the feature expression ability of small objects of the pseudo-image by performing self-fusion and cross-fusion of point clouds and images. A large number of experiments have verified the effectiveness of the SAS fusion module and PVC fusion module.

For a long time, the biggest challenge faced by multi-modal network researchers has been how to overcome the differences in two or more different data characteristics. So in the next step, our research focus will still be on finding the relationships between data with different characteristics in order to achieve better fusion methods to improve detection accuracy.

Author Contributions: Conceptualization, J.Z. and Y.L.; Methodology, J.Z. and D.X.; Software, D.X.; Validation, R.S.; Formal analysis, Y.L. and L.Z.; Investigation, L.Z.; Resources, R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded as follows: Spark funding under Grant HHJJ-2022-0101, Wuhu and Xidian University special fund for industry-university-research cooperation (Project No.: XWYCXY-012021019); General project of key R&D Plan of Shaanxi Province (Project No.: 2022GY-060).

Acknowledgments: Thanks to the Karlsruhe Institute of Technology and the Toyota Technological Institute for providing the KITTI dataset. Thanks to the three reviewers for their helpful and constructive comments on our work. Thanks to every editor for their hard work on this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
2. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
3. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic graph cnn for learning on point clouds. *Acm Trans. Graph. (Tog)* **2019**, *38*, 1–12. [CrossRef]
4. Wang, Y.; Chao, W.L.; Garg, D.; Hariharan, B.; Campbell, M.; Weinberger, K.Q. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8445–8453.
5. Engelcke, M.; Rao, D.; Wang, D.Z.; Tong, C.H.; Posner, I. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1355–1361.
6. Zhou, Y.; Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4490–4499.
7. Yan, Y.; Mao, Y.; Li, B. Second: Sparsely embedded convolutional detection. *Sensors* **2018**, *18*, 3337. [CrossRef] [PubMed]
8. Lang, A.H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; Beijbom, O. Pointpillars: Fast encoders for object detection from point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 12697–12705.
9. Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; Waslander, S.L. Joint 3d proposal generation and object detection from view aggregation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–8.
10. Yang, B.; Luo, W.; Urtasun, R. Pixor: Real-time 3d object detection from point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7652–7660.
11. Liang, M.; Yang, B.; Chen, Y.; Hu, R.; Urtasun, R. Multi-task multi-sensor fusion for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 7345–7353.
12. Liang, Z.; Zhang, M.; Zhang, Z.; Zhao, X.; Pu, S. Rangercnn: Towards fast and accurate 3d object detection with range image representation. *arXiv* **2020**, arXiv:2009.00206.
13. Deng, J.; Zhou, W.; Zhang, Y.; Li, H. From multi-view to hollow-3D: Hallucinated hollow-3D R-CNN for 3D object detection. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 4722–4734. [CrossRef]
14. Sugimura, D.; Yamazaki, T.; Hamamoto, T. Three-dimensional point cloud object detection using scene appearance consistency among multi-view projection directions. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 3345–3357. [CrossRef]
15. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
16. Xu, D.; Anguelov, D.; Jain, A. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 244–253.
17. Vora, S.; Lang, A.H.; Helou, B.; Beijbom, O. Pointpainting: Sequential fusion for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 4604–4612.
18. Xie, L.; Xiang, C.; Yu, Z.; Xu, G.; Yang, Z.; Cai, D.; He, X. PI-RCNN: An efficient multi-sensor 3D object detector with point-based attentive cont-conv fusion module. In Proceedings of the AAAI Conference on Artificial Intelligence, Hilton, NY, USA, 7–12 February 2020; Volume 34, pp. 12460–12467.
19. Wang, J.; Li, J.; Shi, Y.; Lai, J.; Tan, X. AM^{3}Net: Adaptive Mutual-Learning-Based Multimodal Data Fusion Network. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 5411–5426. [CrossRef]
20. Zhang, Y.; Liu, K.; Bao, H.; Zheng, Y.; Yang, Y. PMPF: Point-Cloud Multiple-Pixel Fusion-Based 3D Object Detection for Autonomous Driving. *Remote Sens.* **2023**, *15*, 1580. [CrossRef]
21. Huang, T.; Liu, Z.; Chen, X.; Bai, X. Epnet: Enhancing point features with image semantics for 3d object detection. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Cham, Swizerland, 2020; pp. 35–52.
22. Yoo, J.H.; Kim, Y.; Kim, J.; Choi, J.W. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Cham, Swizerland, 2020; pp. 720–736.
23. Li, Y.; Yu, A.W.; Meng, T.; Caine, B.; Ngiam, J.; Peng, D.; Shen, J.; Lu, Y.; Zhou, D.; Le, Q.V.; et al. Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 17182–17191.
24. Xu, X.; Dong, S.; Xu, T.; Ding, L.; Wang, J.; Jiang, P.; Song, L.; Li, J. FusionRCNN: LiDAR-Camera Fusion for Two-Stage 3D Object Detection. *Remote Sens.* **2023**, *15*, 1839. [CrossRef]
25. Kim, T.; Ghosh, J. Robust detection of non-motorized road users using deep learning on optical and LIDAR data. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 271–276.

26. Liu, J.; Zhang, S.; Wang, S.; Metaxas, D.N. Multispectral deep neural networks for pedestrian detection. *arXiv* **2016**, arXiv:1611.02644.
27. Pang, S.; Morris, D.; Radha, H. CLOCs: Camera-LiDAR object candidates fusion for 3D object detection. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 10386–10393.
28. Qi, C.R.; Liu, W.; Wu, C.; Su, H.; Guibas, L.J. Frustum pointnets for 3d object detection from rgb-d data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 918–927.
29. Shi, S.; Wang, X.; Li, H. Pointrcnn: 3d object proposal generation and detection from point cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 770–779.
30. Yang, Z.; Sun, Y.; Liu, S.; Jia, J. 3dssd: Point-based 3d single stage object detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11040–11048.
31. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef] [PubMed]
32. Deng, J.; Shi, S.; Li, P.; Zhou, W.; Zhang, Y.; Li, H. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 1201–1209.
33. Shi, S.; Guo, C.; Jiang, L.; Wang, Z.; Shi, J.; Wang, X.; Li, H. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10529–10538.
34. Zhang, J.; Xu, D.; Wang, J.; Li, Y. An Improved Detection Algorithm For Pre-processing Problem Based On PointPillars. In Proceedings of the 2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, 23–25 October 2021; pp. 1–6.
35. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 448–456.
36. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010.
37. Zhang, J.; Wang, J.; Xu, D.; Li, Y. HCNET: A Point Cloud Object Detection Network Based on Height and Channel Attention. *Remote Sens.* **2021**, *13*, 5071. [CrossRef]
38. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1907–1915.
39. Liu, Z.; Zhao, X.; Huang, T.; Hu, R.; Zhou, Y.; Bai, X. Tanet: Robust 3d object detection from point clouds with triple attention. In Proceedings of the AAAI Conference on Artificial Intelligence, Hilton, NY, USA, 7–12 February 2020; Volume 34, pp. 11677–11684.
40. Shi, W.; Rajkumar, R. Point-gnn: Graph neural network for 3d object detection in a point cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1711–1719.
41. Wang, M.; Chen, Q.; Fu, Z. Lsnet: Learned sampling network for 3d object detection from point clouds. *Remote Sens.* **2022**, *14*, 1539. [CrossRef]
42. Yang, B.; Liang, M.; Urtasun, R. Hdnet: Exploiting hd maps for 3d object detection. In Proceedings of the Conference on Robot Learning, PMLR, Zürich, Switzerland, 29–31 October 2018; pp. 146–155.
43. Liang, M.; Yang, B.; Wang, S.; Urtasun, R. Deep continuous fusion for multi-sensor 3d object detection. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 641–656.
44. Sindagi, V.A.; Zhou, Y.; Tuzel, O. Mvx-net: Multimodal voxelnet for 3d object detection. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 7276–7282.
45. Yang, Z.; Sun, Y.; Liu, S.; Shen, X.; Jia, J. Ipod: Intensive point-based object detector for point cloud. *arXiv* **2018**. arXiv:1812.05276.
46. Wang, Z.; Jia, K. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 1742–1749.