Research Article

Scale-Aware Attention-Based PillarsNet (SAPN) Based 3D Object Detection for Point Cloud

Xiang Song ¹, Weiqin Zhan,² Xiaoyu Che,³ Huilin Jiang,¹ and Biao Yang ³

¹School of Electronic Engineering, Nanjing Xiaozhuang University, Nanjing 211171, China

²Department of Information Science and Engineering, Changzhou University, Changzhou 213000, China

³National Engineering Research Center of Road Maintenance Technologies, Beijing 100095, China

Correspondence should be addressed to Biao Yang; yb6864171@cczu.edu.cn

Received 6 May 2020; Revised 31 August 2020; Accepted 5 October 2020; Published 22 October 2020

Academic Editor: Ioannis Kostavelis

Copyright © 2020 Xiang Song et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Three-dimensional object detection can provide precise positions of objects, which can be beneficial to many robotics applications, such as self-driving cars, housekeeping robots, and autonomous navigation. In this work, we focus on accurate object detection in 3D point clouds and propose a new detection pipeline called scale-aware attention-based PillarsNet (SAPN). SAPN is a one-stage 3D object detection approach similar to PointPillar. However, SAPN achieves better performance than PointPillar by introducing the following strategies. First, we extract multiresolution pillar-level features from the point clouds to make the detection approach more scale-aware. Second, a spatial-attention mechanism is used to highlight the object activations in the feature maps, which can improve detection performance. Finally, SE-attention is employed to reweight the features fed into the detection head, which performs 3D object detection in a multitask learning manner. Experiments on the KITTI benchmark show that SAPN achieved similar or better performance compared with several state-of-the-art LiDAR-based 3D detection methods. The ablation study reveals the effectiveness of each proposed strategy. Furthermore, strategies used in this work can be embedded easily into other LiDAR-based 3D detection approaches, which improve their detection performance with slight modifications.

1. Introduction

Locating other agents around autonomous vehicles (AVs) is a difficult technological challenge. Cameras provide a cheap solution to detect and track vehicles, cyclists, and pedestrians. However, they cannot estimate accurate distances between AVs and moving agents. AVs rely on several sensors to have a better perception of the environment. LiDAR (Light Detection and Ranging) is arguably the most significant of all sensors. A LiDAR uses a laser scanner to measure the distance to the environment, thereby generating a sparse cloud representation. Based on the sparse cloud representation, AVs can locate themselves and understand accurately the trajectories of the surrounding agents, which allows them to have better motion plans. To achieve this, 3D object detection using the LiDAR point cloud plays a key role.

Following the tremendous advances in deep learning approaches for computer vision, a large body of literature

has investigated object detection methods. However, most studies are designed for 2D object detection. While many similarities can be observed between the 2D and 3D object detection methods, a key difference exists: the point cloud is a sparse and structureless 3D data, whereas the image is a dense 2D data with a fixed structure. Because of this difference, 3D object detection cannot trivially lend itself to standard image convolutional pipelines, which are used commonly in 2D object detection.

Some of the early works generated structured 3D point cloud data by interpolation and then used 3D convolutions [1] for object detection to use standard image convolutional pipelines. However, such methods incur huge costs in terms of time and memory. Hence, to avoid the use of 3D convolutions, the 3D point cloud is projected into a 2D image plane, such as the front view [2] or the bird's eye view (BEV) [3, 4]. The latter is commonly used; however, BEV is extremely sparse, making the direct application of 2D

convolutions inefficient and impractical. A common workaround for this problem is to partition the whole space into voxels [5] or pillars [6]. Then, the hand-crafted features or automatically engineered features, usually with the aid of PointNet [7], can be extracted from voxels or pillars to represent the point cloud as feature maps. Finally, standard image convolutional pipelines can be used on these feature maps for 3D object detection. A detailed review of object detection using LiDAR data will be discussed in the next section.

In this work, a scale-aware attention-based PillarsNet (SAPN), which is an improved version of PointPillar that enables end-to-end learning with only 2D convolutional layers, is proposed. PointPillar segmented the whole space into pillars and then generated a pseudo-image with the extracted pillar-level features. PointPillar can perform 3D detection at high speed. However, the detection accuracy is lower than that of several state-of-the-art methods [5, 6, 8]. Our study offers mainly three contributions to boost the detection accuracy. First, we proposed a scale-aware feature engineering approach, which uses multiresolution pillars and a multiscale backbone jointly. Second, we introduced a spatial-attention mechanism to highlight the object-related information in the feature maps of the backbone, which can improve the detection performance. Lastly, we used the SE-attention to reweight the features fed into the detection head, which can then estimate the positions and orientations of vehicles, cyclists, and pedestrians accurately.

Compared with the traditional PointPillar, the main contributions of this paper are as follows: since the original point cloud features produced by LiDAR are sparse at long range, the traditional fixed resolution method of PointPillar only focuses on global features, not local features, which is not conducive to the extraction of remote sparse point cloud information. In this work, the multiresolution pillars can extract remote sparse point cloud information more effectively; the multiscale backbone can realize the fusion of shallow feature (local feature) and deep feature (global feature) in the feature extraction process; the spatial-attention mechanism can highlight the features related to the target detection task. Through the combination of these algorithms, we can more effectively extract features suitable for target detection and then improve the detection performance.

The rest of the paper is as follows. Section 2 reviews some related works of 3D object detection and point cloud processing. The proposed method is described in Section 3. Experimental results and discussion will be given in Section 4 and Section 5, respectively.

2. Related Work

We review related works that perform 3D detection from single modality and multisensor fusion. Furthermore, we review some recent works on how to process the point clouds.

2.1. 3D Detection from Single Modality. Early approaches to perform 3D detection focused on monocular [9] or stereo

[10, 11] images captured by RGB cameras. However, these camera-based solutions suffer from inherent difficulties of estimating depth from images, which lead to poor localization accuracy. More recent 3D detection approaches use range sensors, such as LiDAR [5, 6], which can capture precise depth measurements accurately. However, these LiDAR-based solutions produce sparse observations, especially at long range, and lack details of objects that can be found in images. Simon et al. [12] utilized the YOLO detection approach on the BEV RGB-map generated from the point cloud to perform 3D object detection. However, BEV is extremely sparse. Thus, convolutional neural networks are inefficient to use. Yang et al. [3] presented a PIXOR 3D object detector from BEV of the LiDAR point cloud. Yang et al. [13] proposed IPOD that used a proposal generation module to output proposals based on each point. Both context and local information for each proposal are extracted for better detection performance. PointRCNN [8] proposed a novel proposal generation algorithm that could generate a small number of high-quality 3D proposals by segmenting the point cloud into foreground objects and background. However, processing each point is time-consuming.

Some preprocessing techniques that better utilize the 3D point cloud were proposed. VoxelNet [5] used a feature learning network to partition the space into voxels and transform points within each voxel to a vector representation characterizing the shape information. SECOND [14] utilized a sparse CNN to process the feature extracted by VoxelNet. The output is fed into an RPN to generate the detection results. PointPillars [6] replaced the voxel structure with a new pillar structure that ignores the z-axis information. A pillar network was built to convert the point cloud into a pseudo-image by extracting the pillar-level feature.

2.2. 3D Detection from Multisensor Fusion. Another commonly used strategy for 3D object detection is multisensor fusion. AVs are always equipped with different sensors, such as LiDAR, cameras, GPS, and high precision maps. MV3D [4] and AVOD-FPN [15] are early works that perform 3D object detection based on the fusion of LiDAR and cameras. However, ROI feature fusion occurs only at high-level feature maps and only fuses features at selected object regions instead of dense locations on the feature map. Qi et al. [16] generated 2D object region proposals and extruded them to a 3D viewing frustum. The 3D object bounding box was then predicted using PointNet on the points in the frustum. Wang and Jia [17] used F-PointNet to extract frustum-level features and reformed them as 2D feature maps, which were processed further by a fully convolutional network and detection header for 3D object detection. However, the overall performance is bounded by each stage, which still uses a single sensor. PointNet was also used as the backbone in RoarNet [18], which first estimated the 3D poses of objects via RoarNet-2D. Thereafter, 3D object detection via RoarNet-3D was performed. Xu et al. [19] presented a dense fusion approach to fuse LiDAR features

extracted by PointNet and image features extracted by ResNet. Liang et al. [20] extracted image features via a CNN and then fused them with LiDAR features with continuous fusion in a multiscale manner. Liang et al. [21] proposed a multitask and multisensor fusion approach for 3D object detection. Ground estimation and depth completion are used as auxiliary tasks to improve 3D object detection. Du et al. [22] used 2D detection to select a subset of points. Score maps were utilized to find the car points in the subset. 3D object detection was achieved via a two-stage refinement CNN by using the car points. Generally speaking, 3D object detection from multisensor fusion has better detection accuracy than 3D object detection from a single modality at the cost of time and memory space.

2.3. Point Clouds Process. A large volume of works study deep learning on point clouds with the increase of their applications in 3D object detection [4, 15] and semantic segmentation [23–25]. PointNet [7] pioneered this route by independently learning each point and gathering the final features with max pooling. PointNet++ [26] reinforced the power of PointNet in capturing local structures, which have been proven important for the success of CNN, by designing a hierarchical feature learning architecture. SO-Net [27] modeled the spatial distribution of point cloud by building a self-organizing map and then conducted hierarchical feature aggregation on individual points. EdgeConv [28] generated edge features that describe the relationships between a point and its neighbors to capture local geometric features while maintaining permutation invariance. Geo-CNN [29] modeled geometric structures between points via directional descriptions. A hierarchical structure was used for scale issues. A relation-shape convolution operator [30] was presented to learn from relation, which encodes the geometric relation of points explicitly.

Many works have focused on the convolutional kernel, which should be designed specially to process irregular 3D points. An octree guided CNN [31] utilized the spherical kernel that preserves translation-invariance and asymmetry properties of the standard 2D convolutional kernel in the 3D point cloud. KPConv [32] could handle the case when input points were not aligned with kernel points. Each point feature was multiplied by all kernel weight matrices with a correlation coefficient depending on its relative position to kernel points. SpiderConv [33] extended convolution operations from regular grids to irregular point sets by parameterizing a family of convolutional filters. SCN [34] used the shape context kernel to capture the contextual shape information from the 3D point cloud. Apart from grouping points from grids or cells, a deep kd-tree-based network [35] was presented to process a 3D point cloud with better scaling performance. KCNet [36] proposed a kernel correlation layer to exploit local geometric structures and used a graph-based pooling layer to exploit local feature structures. Wang et al. [37] leveraged the power of spectral graph CNNs in the PointNet++ framework while adopting a novel graph pooling strategy that aggregates features at graph nodes.

3. Proposed Method

SAPN accepts point cloud as input and predicts 3D bounding boxes for cars, cyclists, and pedestrians. Figure 1 shows that the algorithm contains three main modules: (1) a multiresolution pillar-level feature extraction network that converts the point clouds to a multiscale pseudo-image, (2) a spatial-attention-based convolutional backbone to process the multiscale pseudo-image into high-level representation, and (3) an SE-attention-based detection head that performs 3D object detection. In Figure 1, input is only LiDAR point clouds and outputs are tight 3D bounding boxes with orientations. Notably, we only focus on car objects.

3.1. Pillar-Level Feature Extraction. As discussed above, several one-stage detection approaches extract voxel-level features by using PointNet and then perform object detection by performing 3D convolutions, which are time-consuming. PointPillars [6] first converts the point cloud to a pseudo-image by partitioning the whole space into pillars. The 2D convolutions can be used to extract feature maps from the pseudo-image. Each point in the pillar can be represented by a nine-dimensional pointwise input as follows: $x, y, z, r, x_c, y_c, z_c, x_p$, and y_p , where x, y, z , and r represent the three-dimensional coordinates and reflectance. The five augmented features for each point are x_c, y_c, z_c, x_p , and y_p , where the subscript c denotes distance to the arithmetic mean of all points in the pillar and the subscript p denotes the offset from the pillar center. Then, pillar-level features are extracted by using PointNet. Pseudo-image is generated from the extracted features.

However, the points are not distributed evenly in all targets. Targets close to the LiDAR sensor demonstrate dense points, while targets located far away present sparse points. Most of the 3D object detection approaches capture multiscale features by concatenating feature maps of different scales. We propose to extract multiresolution pillar-level features. As illustrated in Figure 2, the point cloud is discretized into pillars in three different resolutions. $\mathbf{P}_H, \mathbf{P}_M$, and $\mathbf{P}_L$ denote the number of nonempty pillars per sample of high, middle, and low resolutions, respectively. Then, we extract pillar-level features by using PointNet and generate pseudo-image for each resolution. Deconvolution is achieved through a transposed 2D convolution. Parameters in terms of kernel size and stride of convolution and deconvolution are demonstrated below. BatchNorm [38] and ReLU [39] are applied to both downsampled and upsampled features. The sizes of channel number $\mathbf{C}$, width $\mathbf{W}$, and height $\mathbf{H}$ of different feature maps are illustrated in detail in Figure 2. In addition, $\mathbf{C}$ is set to 128; $\mathbf{W}$ and $\mathbf{H}$ are calculated based on the x - y ranges and corresponding resolutions of different targets, which will be specified in Section 4. The input of Figure 2 is the pillars divided by different scales in Figure 2. Firstly, the point cloud space is divided according to the same scale to generate several equal-sized pillars. Then, PointNet is used to extract the features in the pillar. The PointNet algorithm can automatically extract the point cloud features in the pillar. Due to

FIGURE 1: Overview of the proposed SAPN.

FIGURE 2: Pipelines of (a) the multiresolution pillar-level feature extraction and (b) the simplified PointNet. Different colors of pseudo-images represent different scales (best viewed in color).

the characteristics of PointNet, it is suitable for small-scale spatial feature extraction and has a good feature extraction effect for the divided pillar. As presented in Figure 2(b), we use a simplified version of PointNet that consists of two feature propagation modules. The pointwise input vector is fed into the first feature propagation module, in which a fully connected layer is applied followed by BatchNorm and ReLU to generate a pointwise feature vector and then a max-pooling layer is applied to generate locally aggregated feature vector. These two vectors are concatenated point-by-point to generate the point-wise concatenated feature vector, which is fed into the second feature propagation module. The output of the second feature propagation module is fed into the fully connected and max-pooling layers in sequence to generate the pillar-level feature vector. The lengths of all

vectors are specified below. Finally, three pseudo-images are resized and concatenated into a fused one, which will be used as the input of the spatial-attention-based backbone.

3.2. Spatial-Attention-Based Backbone Pillar-Level. The backbone used in this work is similar to the improved region proposal networks (RPN) proposed in VoxelNet [5]. The input to the backbone is the previously generated concatenated pseudo-images. The architecture of the backbone is demonstrated in Figure 3. The backbone contains two subnetworks: one top-down network that downsamples feature maps at increasingly small spatial resolution by convolutional operations and another network upsamples and concatenates features maps of different resolutions.

FIGURE 3: Pipeline of the spatial-attention-based multiscale backbone. Different textures represent feature maps of different scales after spatial-attention procedures (best viewed in color).

Upsampling is also performed by deconvolution (transposed 2D convolution). BatchNorm and ReLU are applied to both downsampled and upsampled features.

To better capture effective spatial information, a spatial-attention operation is used in each scale of the feature map before concatenation. Taking the top branch as an example, the size of the input feature is assumed to be $C \times H \times W$ for simplification. Figure 4 depicts the pipeline of the spatial-attention operation in detail. First, we fed the input into two 1×1 convolutional layers to generate two new feature maps G_1 and G_2 , respectively, where $\{G_1, G_2\} \in R^{C \times H \times W}$. We reshape G_1 to $R^{C \times (H \times W)}$ and transposed it. The matrix multiplication is then performed between the transposition of G_1 and G_2 . The Softmax function σ is used to calculate the spatial-attention matrix $W_{sa} \in R^{(H \times W) \times (H \times W)}$, which encodes the spatial highlight that can be found in the input. Finally, the output is generated by performing matrix multiplication between G_2 and W_{sa} :

$$W_{sa} = \sigma(G_1^T \times G_2). \quad (1)$$

3.3. SE-Attention-Based Detection Head. Generally speaking, SE-attention-based detection head automatically obtains the importance of each feature channel through network self-learning and then improves the useful features according to the importance degree and suppresses the features which are not useful for the current task. Instead of creating a new spatial dimension, it merges feature channels by explicitly modeling the interdependence between feature channels.

In this work, the Single Shot Detector (SSD) [40] setup is used to perform 3D object detection. Similar to SSD, the prior boxes are matched to the ground truth by using 2D Intersection over Union (IoU) [41]. As illustrated in Figure 5, the concatenated features are used as input. We used a SE-attention mechanism [42] to reweight the input to select significant features automatically. The SE-attention

mechanism is achieved by squeeze and excitation operations. Given an input $P \in R^{H \times W \times C}$ at the squeeze step, the global average pooling is used to generate a channelwise vector $s \in R^C$. Then, the following excitation step captures channelwise dependencies by learning the attention mechanism, which is achieved via two fully connected layers as follows:

$$e = \text{ReLU}(W_2 \delta(W_1 s)), \quad (2)$$

where δ and $\text{ReLU}()$ refer to the sigmoid and ReLU functions, respectively. $W_1 \in R^{(C/r) \times C}$ and $W_2 \in R^{C \times (C/r)}$ are two learnable weights and r is the reduction ratio, which is set empirically to 16 in this work. The reweighted features are then generated by reweighting the concatenated features with activation e as follows:

$$\tilde{P}_i = e_i \cdot P_i, \quad (3)$$

where the subscript i indicates the i^{th} channel. Finally, each of the features is enhanced or weakened by the reweighting operation. Afterward, the reweighted features are fed into three branches in a multitask learning manner. The box regression branch is used to estimate the 3D bounding boxes of targets by regressing several residuals. The classification branch is used to calculate the probability of the target being a given class. The direction classifier is used to determine the orientations of 3D bounding boxes.

3.4. Loss Function. As illustrated in Figure 4, three branches exist in the detection head. Thus, the network is trained in a multitask learning manner. We used the same loss functions as proposed in the SECOND [14]. For the box regression branch, we defined the 3D bounding box by X, Y, Z, w, l, h , and θ , which represent the three-dimension center, width, length, height, and yaw rotation of the box, respectively. The residuals between anchors and ground truth 3D boxes are defined as follows:

FIGURE 4: Pipeline of the spatial-attention operation.

FIGURE 5: Pipeline of the SE-attention-based detection head.

$$\begin{aligned}
 \Delta X &= \frac{X_g - X_a}{d_a}, \\
 \Delta Y &= \frac{Y_g - Y_a}{d_a}, \\
 \Delta Z &= \frac{Z_g - Z_a}{h_a}, \\
 \Delta w &= \log \frac{w_g}{w_a}, \\
 \Delta l &= \log \frac{l_g}{l_a}, \\
 \Delta w &= \log \frac{w_g}{w_a}, \\
 \Delta \theta &= \sin(\theta_g - \theta_a),
 \end{aligned} \tag{4}$$

where subscripts g and a represent ground truth and anchor boxes. The diagonal of the base of the anchor box represented by $d_a = \sqrt{w_a^2 + l_a^2}$ is used to normalize ΔX and ΔY . Afterward, the localization loss of the box regression branch is defined by the SmoothL1 function as follows:

$$L_{loc} = \sum_{res \in (X, Y, Z, w, l, h, \theta)} \text{SmoothL1}(\Delta res). \tag{5}$$

The abovementioned box regression scheme treats boxes in the opposite direction as being the same. Thus, a direction classification branch is used by adding a direction classifier to the reweighted features. Similar to the SECOND, a Softmax classification loss is used to calculate the direction loss L_{dir} .

For the classification branch, the focal loss proposed in RetinaNet [43] is utilized to restrain the extreme imbalance

between a mass of anchors and a spot of ground truths. The classification loss is defined as follows:

$$L_{cls} = -\alpha(1 - p_i)^\gamma \log(p_i), \tag{6}$$

where p_i indicates the class probability of the i^{th} sample predicted by the network. Hyperparameters α and γ are set to 0.25 and 2 according to RetinaNet.

Finally, the total loss function is defined by summing up all loss functions defined previously in a weighted manner as follows:

$$L = \frac{1}{N_{pos}} (\beta_{loc} L_{loc} + \beta_{cls} L_{cls} + \beta_{dir} L_{dir}), \tag{7}$$

where N_{pos} is the number of positive anchors and constant weights β_{loc} , β_{dir} , and β_{cls} are set to 2, 0.2, and 1 respectively, according to SECOND.

We use the Adam optimizer to optimize the total loss function for 200 epochs. The initial learning rate is 0.001 and the learning rate is decayed by a factor of 0.8 every 20 epochs.

4. Experimental Results

In this section, we present our experimental setups (dataset, detailed settings, and data augmentation), ablation studies, and quantitative and qualitative evaluations.

4.1. Experimental Setups. All evaluations are performed on the KITTI object detection benchmarking dataset [44] that consists of samples which have both images and point clouds. We only use point clouds for object detection, while several state-of-the-art methods used for comparison utilize both LiDAR point clouds and RGB images. The benchmarking dataset contains 7,481 training and 7,518 testing samples. Similar to PointPillar, we split the official training samples into 3,712 training and 3,769 validation samples. Since the ground truth objects are only annotated if they are visible in the image, we follow the same rule proposed in [5] of only using the points that are visible when they are projected into the image.

Three x - y resolutions are used, namely, 0.08, 0.16, and 0.32 m in the multiresolution feature encoder network. The max numbers of pillars (P) and points per pillar (N) are 12,000 and 100, respectively, according to [6]. We use the same anchors and matching strategy as those in [5]. Each anchor is described by its width, length, height, and z center. Two orientations, 0 and 90 degrees, are applied in each anchor. The 2D IoU is used to indicate whether anchors are matched to ground truth. A positive match is either the highest IoU or above a positive threshold. A negative match is below the negative threshold. All other anchors are ignored in calculating the loss. Axis aligned nonmaximum suppression (NMS) is used to remove redundant bounding boxes at inference time. Notably, the KITTI benchmarking dataset requires detections of cars, cyclists, and pedestrians. Some detailed settings for different targets are given as follows.

In this work, we focus only on detecting cars. Thus, the x , y , and z ranges are (0, 70.4), (-40, 40), and (-3, 1) meters.

The width, length, and height of a car anchor are 1.6, 3.9, and 1.5 m, respectively. The z center is set to -1 m. The positive/negative thresholds used for matching are set to 0.6 and 0.45, respectively.

We perform all our experiments on a computing platform with an AMD2900X CPU and an NVIDIA 2080TI GPU. All experiments are performed on Ubuntu18.04 system with the Pytorch framework.

4.2. Data Augmentation. We used the same data augmentation strategy as in [14] for a good performance on the KITTI dataset. First, a lookup table is created to associate the ground truth 3D boxes. Point clouds fall inside these boxes. Then, for each sample, 15 ground truth 3D boxes for cars are randomly selected and placed into the current point clouds. Each ground truth box is rotated (uniformly drawn from $[-\pi/20, \pi/20]$) and translated (x , y , and z drawn from $N(0, 0.25)$ independently) to further enrich the training set. Finally, a random mirroring flip along the x -axis, as well as a global rotation and scaling, is applied to all boxes and point clouds. A global translation with x , y , and z drawn from $N(0, 0.2)$ is performed to simulate localization noise.

4.3. Evaluation of the Multiresolution Strategy. We evaluated the multiresolution strategy by selecting three learned pseudo-images from different resolutions. Demonstrated in Figures 6(a) and 6(b) are the source and corresponding BEV images, respectively. Figures 6(c)–6(e) are pseudo-images chosen from high, middle, and low resolutions. Then, we can perform a scale-aware analysis of the scene and objects. For better understanding, we zoom-in the regions that contain two car objects from different resolutions of pseudo-images, as shown in Figures 6(f)–6(h). Notably, there are more than two cars as shown in (a). However, a few laser points can be detected in other car regions. Thus, we only choose the first two car objects for analysis. Car positions in the pseudo-images are opposite to those in the source image since the used mirror operation of data augmentation. Obviously, car objects from high-resolution pseudo-image contain abundant details, whereas car objects from low-resolution pseudo-image have strong edges. These complementary representations are beneficial to the subsequent 3D object detection.

4.4. Evaluation of the Spatial-Attention Strategy. We evaluated the spatial-attention strategy by comparing two feature maps w/o spatial-attention operations. As illustrated in Figure 7, feature maps that contain activations of car objects are selected manually and several strong activations are labeled with ellipses. The red ellipse indicates the correct activation and the yellow ellipse indicates the wrong one. In Figure 7(a), two strong activations belong to two car objects being observed. However, in Figure 7(b), strong activations belong to complex backgrounds being observed. Car objects are found to have weaker activations than complex backgrounds, which decreases the detection accuracy. We conclude from the comparison that the spatial-attention strategy

highlights object activations while restraining background noise, thus improving the detection performance. The effectiveness of multiresolution and spatial-attention strategies is further discussed in the ablation study.

4.5. Qualitative Evaluations. Qualitative evaluations are illustrated in Figures 8 and 9. Notably, only LiDAR point clouds are used for training. The 3D bounding box predictions from the BEV and image perspective are demonstrated. Figure 8 illustrates several successful detection results, with tight oriented 3D bounding boxes. Vehicles in these scenes are accurately detected and located even under partial occlusions. Following the rule proposed in PointPillars, van or truck existing in the scenes are not detected. Figure 8 shows three common failure modes due to different reasons. The guardrail is recognized wrongly as a vehicle in the first scene. In the middle scene, a vehicle is missed because of heavy occlusion and few point clouds are projected onto the missed vehicle. In the last scene, vehicles far away are hard to detect because point clouds are extremely sparse in such a far distance and thus cannot represent vehicle shapes.

4.6. Quantitative Evaluations. We followed the official KITTI evaluation protocol, where the IoU threshold for car samples is 0.7. The IoU threshold is the same for both BEV and 3D evaluations. Notably, the KITTI dataset is split into easy, moderate, and hard according to degrees of occlusions. The detection performance is officially ranked on the average precision of moderate results.

As demonstrated in Tables 1 and 2, the proposed method outperforms the selected state-of-the-art methods in both 3D and BEV detection performance. Compared with methods that use RGB images and LiDAR point clouds as input, the proposed method achieves higher average precision with only LiDAR point clouds, thus producing less temporal and spatial overhead. Compared with LiDAR-only methods, the proposed method achieves better results across all splits except for the hard split in 3D detection performance. In this split, the proposed method is slightly inferior to the PointPillars. Actually, the proposed method is an improved version of PointPillars. A detailed ablation study is performed in the next section.

4.7. Time-Cost Evaluations. The time cost of the proposed method is about 26 ms for each forward propagation on our computing platform. That is to say, it takes about 26 ms for once detection. Table 3 presents the comparisons of time costs between the proposed method and some state-of-the-art methods by using the same computing platform. As shown in the table, methods using LIDAR data only runs faster than methods using LIDAR and image data. Among these single mode-based methods, SECOND needs more time since it extracts 3D points from voxels but not the pillars used by PointPillars and our method. Our method improves PointPillars by introducing scale-aware feature extraction and two attention mechanisms. Hence, ours takes more time than PointPillars for once forward propagation.

FIGURE 6: Illustrations of the selected pseudo-images from different resolutions. (a) is the source image and (b) is the corresponding BEV image. (c), (d), and (e) are pseudo-images from high, middle, and low resolutions, respectively. (f), (g), and (h) are selected zoom-in regions that contain two car objects which are labeled with red rectangles.

FIGURE 7: Illustrations of feature maps with (a) and (b) without spatial-attention operations. In each subfigure, correct activations belong to car objects being labeled with red ellipses while wrong activations caused by complex backgrounds are labeled with yellow ellipse (best viewed in color).

FIGURE 8: Qualitative results of the proposed method on the KITTI validation split. For each sample, the lower part is the image labeled with the predicted 3D bounding box and the upper part is a representative view of the corresponding point clouds (best viewed when zoomed-in).

FIGURE 9: Failure samples of the proposed method on the KITTI validation split. For each sample, the lower part is the image labeled with the predicted 3D bounding box and the upper part is a representative view of the corresponding point clouds (best viewed when zoomed-in).

TABLE 1: 3D detection performance: average precision (AP) for car objects in easy, moderate, and hard splits of the KITTI validation set.

Method	Modality	Easy	Mod	Hard
MV3D [4]	RGB + LiDAR	71.29	62.68	56.56
F-PointNet [16]	RGB + LiDAR	83.76	70.92	63.65
AVOD-FPN [15]	RGB + LiDAR	84.41	74.44	68.65
VoxelNet [5]	LiDAR	81.97	65.46	62.85
SECOND [14]	LiDAR	87.43	76.48	69.10
PointPillars [6]	LiDAR	85.43	75.90	69.59
Ours	LiDAR	88.38	78.28	69.52

TABLE 2: BEV detection performance: average precision (AP) for car objects in easy, moderate, and hard splits of the KITTI validation set.

Method	Modality	Easy	Mod	Hard
MV3D [4]	RGB + LiDAR	86.55	78.10	76.67
F-PointNet [16]	RGB + LiDAR	88.16	84.02	76.44
AVOD-FPN [15]	RGB + LiDAR	84.41	74.44	68.65
VoxelNet [5]	LiDAR	89.60	84.81	78.57
SECOND [14]	LiDAR	89.96	87.07	79.66
PointPillars [6]	LiDAR	89.79	86.60	84.17
Ours	LiDAR	91.18	88.37	85.32

TABLE 3: Comparisons of time costs between different methods.

Method	Modality	Time cost (ms)
F-PointNet [16]	LIDAR+ image	152
AVOD-FPN [15]	LIDAR+ image	95
SECOND [14]	LiDAR	47
PointPillars [6]	LiDAR	19
Ours	LiDAR	26

However, as shown in Tables 1 and 2, the higher time cost compared with PointPillars is a trade-off between accuracy improvements.

The frequency of LIDAR is usually 10 Hz or 20 Hz; that is to say, the scan cycle time is 50 ms or 100 ms, which is much longer than our time cost. Meanwhile, in the urban environment, the vehicle speed is generally 60 km/h, about 16.67 m/s, and the driving distance of the vehicle is about 0.43 m among our time cost. Combined with the above two points, it can be inferred that the time cost can meet the requirements of real-time 3D detection.

4.8. Ablation Study. In this section, we conducted several ablation experiments to evaluate the effectiveness of different components of the proposed SAPN. The average precisions

TABLE 4: Average precisions of BEV detection on the KITTI validation split. E_{AP} , M_{AP} , and H_{AP} represent average precisions of easy, moderate, and hard splits, respectively.

Multi-resolution	Spatial-attention	SE-attention	E_{AP}	M_{AP}	H_{AP}
✓	×	×	89.98	87.02	84.01
×	✓	×	90.02	87.68	84.08
×	×	✓	89.67	86.88	83.31
✓	✓	×	91.03	87.91	84.91
✓	×	✓	90.55	87.55	83.55
×	✓	✓	90.32	88.02	84.12
✓	✓	✓	91.18	88.37	85.32

TABLE 5: Average precisions of 3D detection on the KITTI validation split. E_{AP} , M_{AP} , and H_{AP} represent average precisions of easy, moderate, and hard splits, respectively.

Multi-resolution	Spatial-attention	SE-attention	E_{AP}	M_{AP}	H_{AP}
✓	×	×	86.12	76.12	68.56
×	✓	×	86.47	77.22	68.86
×	×	✓	86.01	76.03	68.13
✓	✓	×	87.86	77.86	69.86
✓	×	✓	86.34	76.88	68.78
×	✓	✓	86.65	77.98	69.12
✓	✓	✓	88.38	78.28	69.52

of different combinations of three components, namely, multiresolution, spatial-attention, and SE-attention, are calculated from the perspective of 3D and BEV detections. All experiments are performed on the train split and evaluated on the validation split. As shown in Table 4, the proposed SAPN with all three components outperforms all other combinations or single component from the perspective of BEV detection. Among all single components, the spatial-attention module achieves the best performance and even slightly outperforms the combination of multi-resolution and SE-attention modules in M_{AP} and H_{AP} . From the perspective of 3D detection as demonstrated in Table 5, the proposed SAPN with all three components almost outperforms other combinations, except for H_{AP} . H_{AP} of the combination of multiresolution and spatial-attention is slightly higher than H_{AP} of all three components. The spatial-attention module still achieves the best performance among the three single components.

5. Conclusion and Discussion

In this work, we proposed SAPN, which is a one-stage object detection approach in 3D point clouds. In the first step, we extracted multiresolution pillar-level features from the point clouds, which made the detection approach sensitive to objects having dense or sparse points. In the backbone step, we built three branches to perform multiscale feature engineering and to refine the feature maps with a spatial-attention mechanism. In the final step, the SE-attention was introduced to reweight the fused features that will be fed into

the detection head, which performs 3D object detection in a multitask learning manner. Experiments on the challenging KITTI benchmark demonstrate the effectiveness of the SAPN by achieving similar or better performance compared with several state-of-the-art LiDAR-based 3D detection methods. We also conclude that the proposed strategies can be easily embedded into other LiDAR-based 3D detection approaches with slight modifications.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Xiang Song and Biao Yang performed methodology. Weiqin Zhan, Xiaoyu Che, and Huilin Jiang provided software. Xiang Song, Weiqin Zhan, and Biao Yang performed validation. Xiang Song performed investigation. Xiang Song wrote original draft. Xiang Song, Xiaoyu Che, and Xiang Song performed review and editing. Xiang Song, Biao Yang, and Huilin Jiang were responsible for funding acquisition. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (Grant no. 61801227); the Qing Lan Project of Jiangsu (Grant no. QLGC-2020); the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (Grants nos. 18KJB413007, 18KJB520003, and 19KJD420002); the fund of the Jiangsu SINO-ISRAEL Industrial Technology Research Institute (Grant no. JSII-TRI202007); the Changzhou Application Foundation Research Project (Grant no. CJ20200083); the Jiangsu Province Industry-University-Research Cooperation Project (Grant no. BY2019078); the Open Project of Key Laboratory of Ministry of Public Security for Road Traffic Safety (Grant no. 2019ZDSYSKFKT06); the Key Laboratory for New Technology Application of Road Conveyance of Jiangsu Province (Grant no. BM20082061708).

References

- [1] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: fast object detection in 3D point clouds using efficient convolutional neural networks," in *2017 Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Singapore, pp. 1355–1361, September 2017.
- [2] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," 2016, <http://arxiv.org/abs/1608.07916>.

- [3] B. Yang, W. Luo, and R. Urtasun, "Pixor: real-time 3D object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7652–7660, Salt Lake City, UT, USA, June 2018.
- [4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, Honolulu, HI, USA, July 2017.
- [5] Y. Zhou and O. Tuzel, "Voxelnet: end-to-end learning for point cloud based 3D object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, Salt Lake City, UT, USA, June 2018.
- [6] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12 697–712 705, Long Beach, CA, USA, July 2019.
- [7] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, Honolulu, HI, USA, July 2017.
- [8] S. Shi, X. Wang, and H. Li, "Pointnet: 3D object proposal generation and detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–779, Seoul, South Korea, October 2019.
- [9] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2156, Las Vegas, NV, USA, June 2016.
- [10] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3D object proposals using stereo imagery for accurate object class detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1259–1272, 2017.
- [11] A. Broggi, S. Cattani, M. Patander, M. Sabbatelli, and P. Zani, "A full-3D voxel-based dynamic obstacle detection for urban scenario using stereo vision," in *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, IEEE, Hague, The Netherlands, pp. 71–76, October 2013.
- [12] M. Simon, S. Milz, K. Amende, and H. Gross, "Complex-yolo: real-time 3D object detection on point clouds," <http://arxiv.org/abs/1803.06199>.
- [13] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Ipod: intensive point-based object detector for point cloud," 2018, <http://arxiv.org/abs/1812.05276>.
- [14] Y. Yan, Y. Mao, and B. Li, "Second: sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.
- [15] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "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)*, IEEE, Madrid, Spain, pp. 1–8, October 2018.
- [16] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 918–927, Salt Lake City, UT, USA, June 2018.
- [17] Z. Wang and K. Jia, "Frustum convnet: sliding frustums to aggregate local point-wise features for amodal 3D object detection," 2019, <http://arxiv.org/abs/1903.01864>.
- [18] K. Shin, Y. P. Kwon, M. Tomizuka, and Roarnet, "A robust 3D object detection based on region approximation refinement," in *Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Paris, France, pp. 2510–2515, June 2019.
- [19] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: deep sensor fusion for 3D bounding box estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 244–253, Salt Lake City, UT, USA, June 2018.
- [20] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proceedings of the European Conference on Computer Vision, ECCV*, Munich, Germany, pp. 641–656, September 2018.
- [21] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7345–7353, Long Beach, CA, USA, June 2019.
- [22] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A general pipeline for 3D detection of vehicles," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Brisbane, Australia, pp. 3194–3200, May 2018.
- [23] H. Su, V. Jampani, D. Sun et al., "Splatnet: sparse lattice networks for point cloud processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2530–2539, Salt Lake City, UT, USA, June 2018.
- [24] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3D segmentation of point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2626–2635, Salt Lake City, UT, USA, June 2018.
- [25] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, Salt Lake City, UT, USA, June 2018.
- [26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the Advances in neural information processing systems*, pp. 5099–5108, Long Beach, CA, USA, December 2017.
- [27] J. Li, B. M. Chen, and G. Hee Lee, "So-net: selforganizing network for point cloud analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9397–9406, Salt Lake City, UT, USA, June 2018.
- [28] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," 2018, <http://arxiv.org/abs/1801.07829>.
- [29] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3D point clouds using geo-cnn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 998–1008, Long Beach, CA, USA, June 2019.
- [30] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8895–8904, Long Beach, CA, USA, June 2019.
- [31] H. Lei, N. Akhtar, and A. Mian, "Octree guided cnn with spherical kernels for 3D point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9631–9640, Long Beach, CA, USA, June 2019.
- [32] H. Thomas, C. R. Qi, J.-E. Deschaud et al., "Flexible and deformable convolution for point clouds," 2019, <http://arxiv.org/abs/1904.08889>.

- [33] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 87–102, Munich, Germany, September 2018.
- [34] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional shape context net for point cloud recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4606–4615, Salt Lake City, UT, USA, June 2018.
- [35] R. Klokov and V. Lempitsky, "Escape from cells: deep kd-networks for the recognition of 3D point cloud models," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 863–872, Venice, Italy, October 2017.
- [36] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4548–4557, Salt Lake City, UT, USA, June 2018.
- [37] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–66, Munich, Germany, September 2018.
- [38] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, <http://arxiv.org/abs/1502.03167>.
- [39] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, Haifa, Israel, June 2010.
- [40] W. Liu, D. Anguelov, D. Erhan et al., "Ssd: single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, Springer, Amsterdam, The Netherlands, pp. 21–37, October 2016.
- [41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [42] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, Salt Lake City, UT, USA, June 2018.
- [43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, Venice, Italy, October 2017.
- [44] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Washington, DC, USA, pp. 3354–3361, June 2012.