Electronic Engineering and Informatics
G. Izat Rashed (Ed.)

doi:10.3233/ATDE240088

Smaller Pillar Size and Better Shape Learning

Run YU, Mingzheng ZHANG[1] and Xuanpeng ZHU
ZTE Corporation, ZTE Plaza, Keji Road South, Hi-Tech Industrial Park, Nanshan District, Shenzhen, 518057, P.R.China

Abstract. BEV-based 3D object detectors have gained recognition for their rapid processing capabilities, making them well-suited for on-device applications. In this research paper, we introduce SS-Pillar, a novel and efficient method for 3D object detection that offers superior quality results. Compared with the previous methods, we introduce the SS module which is the auxiliary network based on fine-grained pillars to learn the shape of objects. We also design a shape complete module to address the issue of far-distance missing region affected by the occlusion and sparse point clouds. Our model achieves real-time performance (28.29FPS) on NVIDIA Tesla V100 GPU significantly and outperforms competitive baselines on KITTI and nuScenes datasets.

Keywords. Point cloud object detection, Fine-grained pillars, Auxiliary network, Anchor free

1. Introduction

Recent advancements in LIDAR-based 3D object detection have shown remarkable progress, primarily attributed to the successful application of deep neural networks (DNN) for point cloud representation learning. Nevertheless, there is an increasing demand to create a highly efficient 3D detector capable of real-time processing for autonomous vehicles.

Currently, one of the popular methods for on-device deployment is PointPillars [12]. In this method, the point cloud is first converted into pillars, and then they utilize PointNet [4] to learn the features of points within each pillar. Pointpillars only adapts the 2D convolutions, making it easy to deploy and having a significant advantages in speed. However, it lacks powerful pillar feature encoding, decimates much local fine-grained information, and impairs performance especially for small objects.

To address these limitations, we propose an efficient 3D detector from LIDAR point cloud called SS-Pillar. SS-Pillar is a pillar-based model consisting of pillar encoding, feature extraction, and an auxiliary network SS module. The SS module is designed for learning the fine-grained shapes of objects, including those in occlusion with a shape complete module. Additionally, we propose an efficient and compact backbone and center-based detector [18].

[1] Corresponding author: Mingzheng ZHANG, ZTE Corporation, ZTE Plaza, Keji Road South, Hi-Tech Industrial Park, e-mail: zhang.mingzheng@zte.com.cn

Overall, our proposed SS-Pillar model overcomes the limitations of existing methods by leveraging fine-grained object shape information while maintaining real-time speed and high accuracy, making it well-suited for autonomous vehicle applications.

2. Related Work

2.1. LIDAR-based 3D Object Detection

As we kown, point clouds are characterized by their irregular and sparse nature, necessitating encoding techniques (e.g., raw points, voxelization) prior to their input into a network. For instance, certain approaches utilize a mesh grid to convert point clouds into voxels, where features such as location, density, and intensity are concatenated in different voxels as distinct channels. Voxelized point clouds are often projected onto various views, such as the bird's-eye view (BEV), and processed using 2D convolutional networks [3], [10], [17]. Alternatively, they can be preserved in 3D coordinates and processed using sparse 3D convolutional networks [15]. Other methodologies employ raw point clouds as input for 3D detection or segmentation, utilizing multi-layer perceptrons (MLPs) and max pooling to address the disorderliness of point clouds, yielding satisfactory performance [4]. Moreover, certain approaches combine voxelization and raw points, leading to techniques like VFE (Voxel Feature Encoding), which significantly enhances the performance of LIDAR-based detectors [20].

However, the use of encoders can slow down the detection pipeline. In response, Pointpillar [12] proposes the encoding of 3D points as pillars instead of voxels. Consequently, the entire set of 3D points is transformed into a 2D pseudo image with channels equivalent to those of VFE [22].

2.2. Anchor-Based/Anchor-Free Detector

Various types of detectors exist, including anchor-based and anchor-free detectors [19]. The concept of anchors was introduced in Faster R-CNN [13] firstly, where 2D anchors were expanded into the 3D space by incorporating a z-axis value. Predefined boxes with $(\{x, y, z, l, w, h, \theta\})$ were employed for describing bounding boxes. However, the use of dense anchors results in a large number of potential objects, which in turn necessitates non-maximum suppression (NMS) to address the issue effectively. In contrast, anchor-free detectors frame the detection problem as a keypoint detection task. For instance, Chen proposed hotspots [5], while Yin, Zhou, et al. introduced CenterNet [8]. These networks derive features based on object parts, rendering the need for anchors unnecessary. There's no doubt single-stage detectors possess a simpler structure that is more suitable for real-time deployment. Cxonsequently, we present a single-stage, anchor-free 3D detector for our methodology.

3. Methods

This section provides a comprehensive overview of SS-Pillar, focusing on three key aspects: the SS Module, Shape Complete Module, and backbone and anchor-free detector. The architecture of SS-Pillar is visualized in Figure 1.

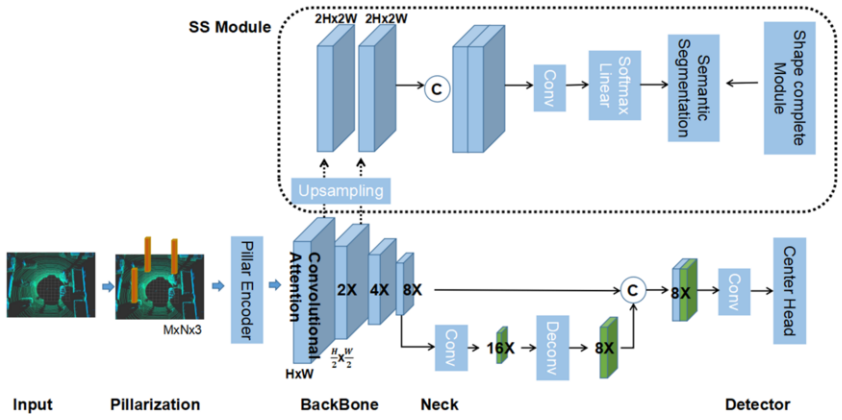

Figure 1. A framework of our pillar-based 3D detection (SS-Pillar) system and detailed structure of backbone and SS Module. The whole pipeline consists of pillarization, backbone, neck, detector and SS Module, where Shape complete Module completes the occupancy of object shapes in the regions affected by occlusion.

3.1. Preliminaries

To ensure fast pipeline processing on GPU, we adopt a pillar-based model, specifically Pointpillar as discussed in section 2.1, as our base model. Since the features of each pillar are derived from the points within it, and the density of points varies for pillars at different distances from the Lidar sensor, it is logical to partition the scene into a near sub-scene and a far sub-scene based on depth zones along the forward-axis. While both sub-scenes share the same voxel size, the number of points assigned to different pillars differs significantly. Due to the sparser nature of 3D point clouds in the far sub-scene, the corresponding pseudo-image comprises numerous empty features. Consequently, the model must learn distinct representations for objects of the same class due to the substantial differences in the point cloud.

To determine the effect of different sizes of scenes and point cloud sparsity on the performance and speed of pillar-based methods, we construct the experiments referred to [9]. For instance, we trained models on the KITTI dataset with pillar sizes $v_x = v_y = d$, with $d \in \{16,20,24,28\}$. To account for different pillar sizes, we trained two models for each size: one on objects present in the full scene and another on objects within the near-related subset. The pillar sizes trained in the near-related subset are specifically labeled as {16-near, 20-near, 24-near, 28-near}. We used Car, Pedestrian and Cyclist class combinations as in the original Pointpillar. As can be seen in the Figure 2, the experiments show that

- For hard level objects, the evaluation metric mAP increase with reducing the pillar size, or to put it in another way, it makes sense to detect distant small objects using the fine-grained pillar size.
- Compared with the full-scene models, models trained on near objects achieve similar or better performance. That is, the model learn objects' appearance representation nearly from near objects due to the sparsity of point clouds at large distances. Hence, how to exploit the far sub-scene information becomes a challenge.

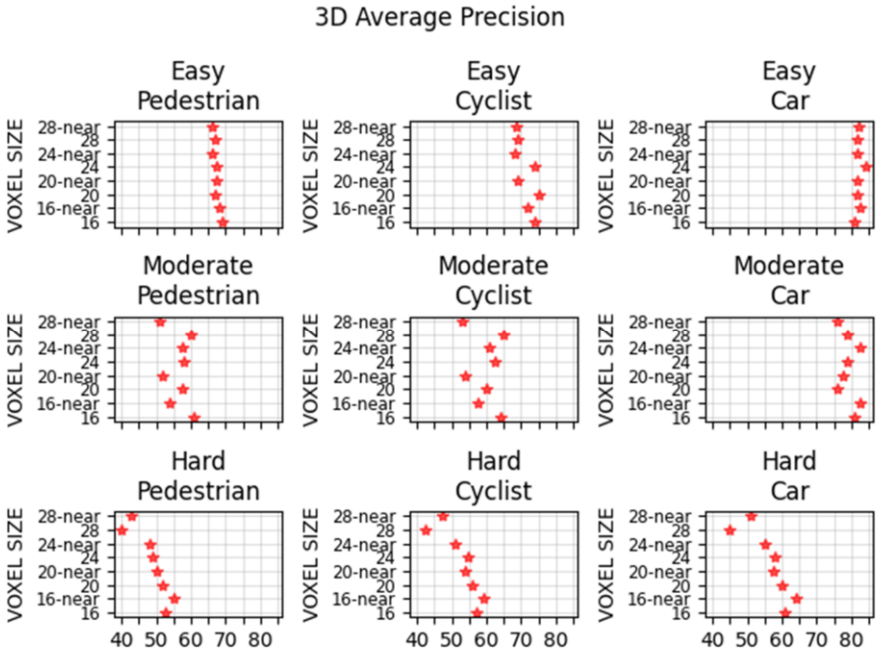

Figure 2. illustrates the performance evaluation of Pointpillar models trained on both the full-scene and a near-related subset, considering different classes and levels. The suffix "-near" indicates that the models were trained specifically on the near-related subset, and their evaluation is also conducted on this subset.

3.2. SS Module

This subsection mainly describe the detailed design of the SS module for being aware of the shapes of objects using the fine-grained pillar size.

3.2.1.Motivation

The improved detection of small and large-distance objects is evident in Figure 2 when using a fine-grained pillar size. Our approach avoids the need for additional inference time and complex deployment by incorporating an auxiliary network inspired by SA-SSD [7]. This auxiliary network leverages semantic segmentation to learn the fine-grained pillar structures from intermediate features obtained during various stages of down-sampling. To achieve this objective, it is necessary to convert the CNN features into representations that capture the fine-grained details of the pillars.

3.2.2.Feature representation

In Figure 3, it can be observed that the feature maps from the initial two layers primarily capture the shape and structure of objects, while the last two layers focus on object textures. To leverage this observation, we transform the feature maps from the first two layers of the backbone into a real-world fine-grained pillar representation based on the current stage's quantization step. For feature propagation, we employ bilinear interpolation. Furthermore, we introduce a pillar-based semantic segmentation task to guide the backbone in learning more distinguishing patterns within object classes.

Specifically, we utilize a sigmoid function in the segmentation branch to predict the probability of each center point of the pillars denoted as π. To optimize this task, we apply a focal loss [11],

$$L_{SS} = \frac{1}{N}\sum -\alpha(1-\hat{p}_i)^{\gamma}\log(\hat{p}_i)$$

where p_i is multiplied by the one hot label and α and γ are the hyper-parameters and we use the empirical values 0.25 and 2.

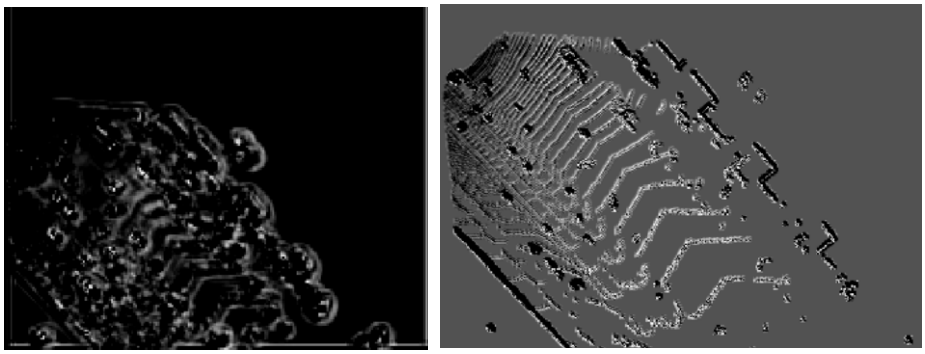

Figure 3. Feature maps from different CNN down-sampling layers.

3.3. Shape complete module: learning shapes in occlusion

3.3.1.Motivation

As we analyzed before, the SS Module is designed for being aware of the shapes of objects. However, due to signal miss and occlusion the model can't learn knowledge from the shape of far-distance objects. To solve such a problem, we design a shape-complete module to help the backbone learn the complement of the target objects.

3.3.2.Evaluation the completed object shapes

Given the target object A and the source object B covering most parts of A, we give a cost E to evaluate the similarity of A and B,

$$E(A,B) = \sum_{x\in A}\min_{y\in B}\| x-y \| + \frac{\alpha}{N(B/A)} - \beta IoU(D_A, D_B)$$

where D_A means the 2D bounding box of A and N(B/A) means the pillar numbers belonging to B except A.

To approximate the complete shape of A , we select the top source objects B with the best score evaluated by the cost E .

3.3.3.Create target labels

We use the labeled target to get the fine-grained pillars belonging to the objects. For instance, we mirror the pillars against the orientational center axis of the bounding box for cars dyed blue and cyclists dyed green illustrated in Figure 4.

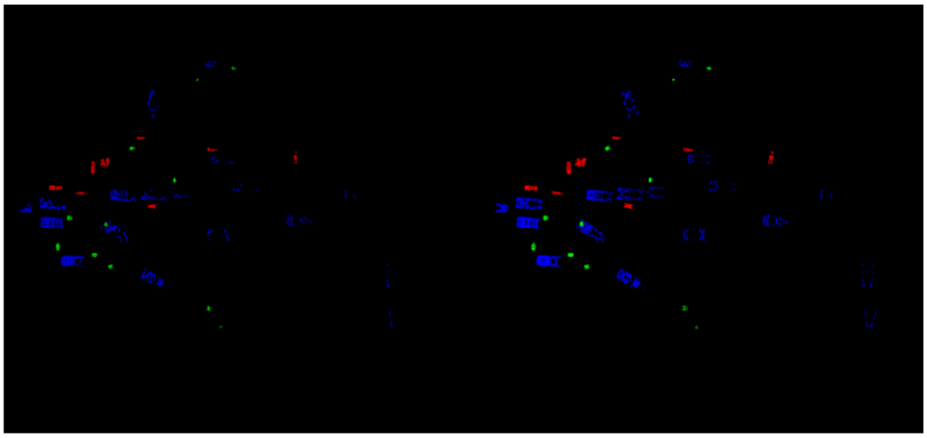

Figure 4. The process to approximate the complete object shapes for car, cyclists and pedestrian on KITTI under the bird eye's view.

3.4. Design for backbone and center-based head

3.4.1.Backbone

We have observed that the lack of robust feature encoding in previous pillar-based methods is the primary factor contributing to their suboptimal performance. Inspired by CBAM [16], we add the convolutional attention module before encoder the pillar feature map F, resulting in $F' = M_S \otimes M_C \otimes F$, where

$$M_S(F) = \sigma(MLP(Avgpool(F)) + MLP(Maxpool(F)))$$

$$M_C(F) = \sigma(conv([Avgpool(F); Maxpool(F)]))$$

In addition, we devised an encoder to facilitate hierarchical deep pillar feature extraction, along with a neck module for effective multi-scale feature fusion. Inspired by RepVGG [6] and Fastpillar [21], we exploit RepVGG block (i.e. a stack of 3×3 convolutional layers) to simplify the model structure and reduce the inference latency with excellent feature representation ability.

3.4.2.Center Head

Here we commonly use the center-based detect head CenterNet with little modification.

Unlike the normal CenterNet, for each pixel (x, y) in the pseudo image, we define the value of it in the heatmap as follows,

$$M_{x,y} = \begin{cases} 1, if\ d = 0 \\ 0.8, if\ d = 1 \\ \frac{1}{d}, else \end{cases}$$

where d represents the distance between the center of the bounding box and the coincident pixel in the pseudo image. A prediction $M_{x,y} = 1$ denotes the presence of an object center, while $M_{x,y} = 0$ indicates that the pillar represents the background. Following the approach presented in [8], we employ the modified focal loss for training the heatmap loss L_{hm}.

In accordance with the experiments conducted in [14], the optimization direction remains consistent even in the presence of significant orientation deviations. To address this, we introduce an Orientation-Decoupled IoU-related regression loss, referred to as OD-IoU, which decouples the orientation and extends the IoU regression loss.

3.4.3. Overall Loss Function

The total loss function is given as follows,

$$L_{total} = L_{SS} + L_{ori} + L_{hm} + L_{others}$$

where L_{SS} and L_{hm} are focal loss while L_{ori} and L_{others} are L_1 loss.

4. Experiments

This section begins with an introduction to the KITTI dataset utilized in our study, along with a detailed description of the experiment settings. Subsequently, we present the performance analysis conducted on the KITTI [1] validation dataset, followed by a presentation of preliminary results obtained from the nuScenes [2] validation dataset.

4.1. Dataset and experiments setting

The KITTI object detection dataset comprises a total of 7,481 training samples and 7,518 test samples. In our experiments, we divided the official dataset into 3,712 samples for training purposes, while the remaining 3,769 samples were allocated for validation. To ensure consistency with Pointpillars, we set the detection range, pillar size, and output channels of the pillar encoder similarly for KITTI . In the backbone architecture, all convolutional layers employ a kernel size of 3. The resulting shapes of the backbone and neck outputs are depicted in Figure 1. Each head consists of two convolutional layers: the first layer has a kernel size of 3 and 32 channels, while the second layer has a kernel size of 1, with different channel numbers assigned to different heads. During inference, we utilize circle Non-Maximum Suppression (NMS) to suppress overlapping detections. The model is trained for a total of 100 epochs.

For consistency, we have set the maximum number of objects, maximum points per voxel, maximum voxel number per frame, and voxel size to 500, 10, 160,000, and [0.2m, 0.2m, 8m], respectively.

4.2. Evaluation on the KITTI and nuScenes validation dataset

As the experiment setting before, compared with the original Pointpillar our model SS-Pillar can achieve better performance especially on Cyclist without a complex NMS, which is illustrated in Table 1.

Table 1. The KITTI validation dataset car, pedestrian, cyclist detection performance under the BBOX.

Method	Car BBOX IoU=0.7			Ped. BBOX IoU=0.5			Cyc. BBOX IoU=0.5		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Pointpillars	90.5317	88.5366	87.6965	64.8495	62.2438	58.7925	66.1198	56.0745	51.9256
SS-Pillar(with SSD head)	90.4778	89.0245	88.0708	72.6371	69.9601	67.3834	83.1102	71.2423	69.5357
SS-Pillar(no SS module)	89.8610	86.0672	85.3331	61.6167	54.5474	53.8150	76.0883	63.7429	59.0650
SS-Pillar(no shape complete)	89.8323	87.9542	85.5084	62.7041	58.6146	54.3333	80.5186	66.5156	62.3297
SS-Pillar	89.9190	88.2230	86.6229	62.9086	59.5958	54.4637	**81.4422**	**65.5512**	**62.9305**

As shown in Table 2, we use point clouds following the convention on the benchmark to measure the speed of SS-Pillar on KITTI test set. Compared with Pointpillar, SS-Pillar achieves 28.285 FPS on a single Tesla V100 GPU, including 7.204ms for pillar encoding, 18.38ms for network and 9.77ms for post-processing.

Table 2. The time-consuming and overall speed of each part of our SS-Pillar models on a single Tesla V100 GPU.

Method	FPS	Pillar(ms)	Network(ms)	Post-process(ms)	Overall
Pointpillar	14.643	7.204	7.785	53.5	68.488
SS-Pillar(SSD head)	10.786	7.204	47.74	37.77	92.714
SS-Pillar	**28.285**	7.204	18.38	**9.77**	35.354

4.3. Ablation studies

4.3.1.Number of the upsampling backbone layers

Effect of numbers of upsampling layers is shown in Table 3. Each row corresponds to an auxiliary network with different number of upsamping layers.

Table 3. Effect of numbers of upsampling layers.

Layers	Car AP	Pred. AP	Cyc. AP
0	89.86	61.62	76.1
2(SS-Pillar)	89.92	62.91	81.44
4	89.89	62.29	78.19

4.3.2.Importance of SS module

As shown in Figure 5, the SS module is able to guide the feature maps from the backbone leaning the shape of objects. Moreover, without extra cost at the inference stage, the SS module has a better performance than original version increasing almost 10 points of mAP.

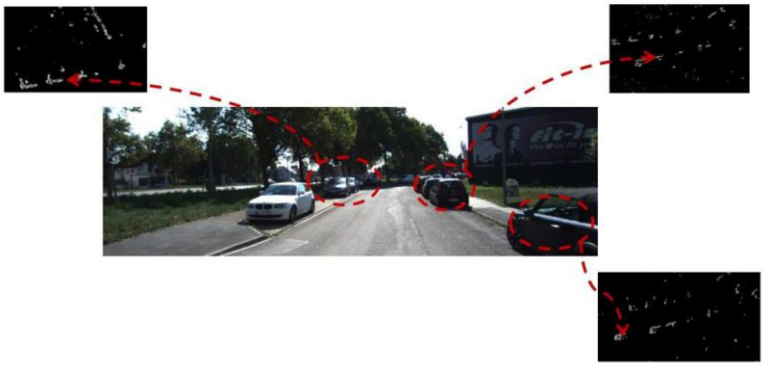

Figure 5. Visualization of the first layer feature map guiding by SS module on KITTI test set. The SS module is able to separate and learn the shape of objects.

5. Conclusion

In this study, we studied the pillar-based 3D object detectors and introduced a novel detector called SS-Pillar. This enables the features learned in the backbone network to have an improved understanding of object shapes in the Bird's Eye View (BEV) perspective. Additionally, we propose a shape completion module to address the issue of missing regions in the far-distance. Furthermore, we have developed a pillar encoder backbone and a widely used center-based detection head [18]. Experimental results on the KITTI dataset demonstrate the efficiency and superior performance of our method, which also exhibits faster processing speed for on-drive applications compared to Pointpillar.

References

[1] Geiger, A., Lenz, P., Urtasun, R., Are we ready for autonomous driving? the kitti vision benchmark suite, in 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence. https://ieeexplore.ieee.org/document/6248074

[2] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, Q., nuscenes: A multimodal dataset for autonomous driving, in proceedings of the IEEE/CVF conference on computer version and pattern recognition, 2020, pp. 11621–11631. https://arxiv.org/abs/1903.11027

[3] Chen, X. Z., Ma, H. M., Wan, J., Li, B., Xia, T., Multi-view 3d object detection network for autonomous driving, in CVPR, 2017. https://arxiv.org/abs/1611.07759

[4] Qi, C. R., Su, H., Mo, K. C., Guibas, L. J., Pointnet: Deep learning on point sets for 3d classification and segmentation, in CVPR, 2017. https://arxiv.org/abs/1612.00593

[5] Chen, Q., Sun, L., Wang, Z. X., Jai, K., Yuille, A., Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots, in ECCV, 2020. https://arxiv.org/abs/1912.12791

[6] Ding,X. H., Zhang, X. Y., Ma, N. N., Han, J. G., Ding, G. G., Sun, J., Repvgg: Making vgg-style convnets great again, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13733–13742, 2021. 3, 5. https://arxiv.org/abs/2101.03697

[7] He, C. H., Zeng, H., Huang, J. Q., Hua, X. H., Zhang, L., Structure Aware Single-Stage 3D Object Detection From Point Cloud, in CVPR, 2020. https://ieeexplore.ieee.org/document/9157660
[8] Law, H., Deng, J., Cornernet: Detecting objects as paired keypoints, in ECCV, 2018. https://arxiv.org/abs/1808.01244
[9] Oleksiienko, I., Iosifidis, A., Analysis of voxel-based 3D object detection methods efficiency for real-time embedded systems, in Proceedings of International Conference on Emerging Techniques in Computational Intelligence (ICETCI), 2021, pp. 59-64. https://arxiv.org/abs/2105.10316
[10] Ku, J., Mozifian, M., Lee,J., Harakeh, A., Waslander, S. L., Joint 3d proposal generation and object detection from view aggregation, in IROS, 2018. https://arxiv.org/abs/1712.02294
[11] Lin, T. Y., Goyal, P., Girshick, R., He, K., Dollár, P., Focal loss for dense object detection, in proceedings of the IEEE international conference on computer version, 2017:2980-2988. https://arxiv.org/abs/1708.02002
[12] Lang, A. H., Vora, S., Caesar, H., Zhou, L. B., Yang, J., Beijbom, O., Pointpillars: Fast encoders for object detection from point clouds, in CVPR, 2019. https://arxiv.org/abs/1812.05784
[13] Ren, S. Q., He, K. M., Girshick, R., Sun, J., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Network, in TPMAI, 2016. https://arxiv.org/abs/1506.01497
[14] Shi, G. S., Li, R. F., Ma, C., PillarNet: High-Performance Pillar-based 3D Object Detection, in proceddings ECCV, 2022. https://arxiv.org/abs/2205.07403
[15] Song, S. R., Xiao, J. X., Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images, in CVPR,2016. https://arxiv.org/abs/1511.02300
[16] Woo, S., Park, J., Lee, J. Y., In So Kweon, CBAM: Convolutional Block Attention Module, in Proceedings of European Conference on Computer Vision, 2016: 21-37. https://arxiv.org/abs/1807.06521
[17] Yang, B., Luo, W. J., Urtasun, R., Pixor: Realtime 3d object detection from point clouds, in CVPR, 2018. https://arxiv.org/abs/1902.06326
[18] Yin, T. W., Zhou, X. Y., Krähenbühl, P., Center-Based 3D Object Detection and Tracking, in CVPR, 2021. https://arxiv.org/abs/2006.11275
[19] Zhang, S. F., Chi, C., Yao, Y. Q., Lei, Z., Li, S. Z., Bridging the Gap Between Anchor-based and Anchor-free Detection via Adaptive Training Sample Selection, in CVPR, 2020. https://arxiv.org/abs/1912.02424
[20] Zhou, Y., Tuzel, O., Voxelnet: End-to-end learning for point cloud based 3d object detection, in CVPR, 2018. https://arxiv.org/abs/1711.06396
[21] Zhou, S. F., Tian, Z., Chu, X. X., Zhang, X. Y., Zhang, B., Lu, X. B., Feng, C. J., Jie, Z. Q., Chiang, P. Y., Ma, L., FastPillars: A Deployment-friendly Pillar-based 3D Detector, arXiv preprint arXiv:2302.02367, 2023. https://arxiv.org/abs/2302.02367
[22] Zhou, X. Y., Wang, D. Q., and Krähenbühl, P., Objects as points, arXiv preprint arXiv:1904.07850, 2019. https://arxiv.org/abs/1904.07850