Supplementary Material
HVPR: Hybrid Voxel-Point Representation for Single-stage 3D Object Detection

Jongyoun Noh Sanghoon Lee Bumsub Ham*
School of Electrical and Electronic Engineering, Yonsei University

1. Implementation details

Point-based and voxel-based features. For each 3D scene, we subsample 16,384 point clouds, and use them as our inputs. We exploit PointNet++ [4], consisting of set abstraction (SA) and feature propagation (FP) layers, to extract point-based features. Specifically, we use two SA layers to downsample the number of point clouds from 16,384 to 4,096 and 1,024, respectively. We then obtain the point-based features of size 64×1 for each input point cloud after applying two FP layers. To extract voxel-based features, we use a multi-layer perceptron (MLP) consisting of two hidden layers, whose channel sizes are 32 and 64, respectively.

Backbone network. Our backbone network consists of three convolutional blocks, where each block extracts a feature map of a specific scale. See Fig. 2 in the main paper (Backbone Network with AMFM). Each block has three 2D convolutional, BatchNorm [2] and ReLU layers. For the first layer of each block, we exploit a convolution with stride 2 in order to downsample input feature maps. The sizes of output channels for feature maps are 128, 256, and 512, respectively, for each block, and the filter size is set to 3 for all layers.

AMFM. We use a MLP with two hidden layers with channel sizes of 16 and 32, respectively, to obtain voxel-wise representations of the 3D scale features. We adopt a 3×3 convolutional layer to produce a spatial attention map in AMFM. We upsample scale-aware features with a transposed convolution, such that they have the same spatial resolution and channel size as the largest one. See Fig. 2 in the main paper (Backbone Network with AMFM).

Detection head. We use two fully connected layers with 384 channels to localize and classify objects.

Settings for the pedestrian class. For the pedestrian class, we train our network for 200 epochs, with a learning rate of 2e-4 and a weight decay of 1e-4. The learning rate is decayed by a factor of 0.8 every 15 epochs. Batch size is set to 1 for each GPU. We apply a global translation, where a translation factor is drawn from $\mathcal{N}(0, 0.2)$, in addition to the data augmentation techniques as for the car class. The parameters of our model for the pedestrian class are the same with the ones for the car class except for the number of prototypes ($K = 20$ for car, $K = 10$ for pedestrian). For other parameters, we use the same setting as in [3] as follows: Assuming that the dimension of a 3D scene (W, H, L) is within a range of $[(0, 48), (-20, 20), (-2.5, 0.5)]$ meters. We set the size of anchors to $0.6 \times 0.8 \times 1.73$. We choose the anchors with the IoU scores larger than 0.5 as positive boxes, while those lower than 0.35 are used as negative ones. We use $(0.16, 0.16, 4)$ as the size of a voxel, $v_W \times v_H \times v_L$. The number of point clouds within each voxel N_{vox} and the size of augmented point clouds D are set to 100 and 9, respectively.

*Corresponding author.

Table 1: Runtime analysis for each step of our model.

Step	Time
Data preprocessing	1.2 ms
Voxel encoder	1.5 ms
Pseudo image w/ memory	3.2 ms
Backbone w/ AMFM	9.9 ms
Head and post-processing	11.9 ms

2. More results

Runtime analysis. The average runtime of our full model for the car class is 27.7 milliseconds with an Nvidia 2080Ti GPU. The detailed runtime for each step is shown in Table 1. We can see that most computation time is spent for the backbone network and the post processing. Estimating a pseudo image just takes 4.7 milliseconds.

AMFM. We visualize in Fig. 1 activations of multi-scale and scale-aware features from the first and the second blocks in a backbone network. Note that the scale-aware features are obtained by applying AMFM to multi-scale features. We can see that scale-aware features highly activate on decisive regions, compared to multi-scale ones. More specifically, the scale-aware features from the first block,

which are of high-resolution, activate on small or distant objects, while suppressing the features near a LiDAR sensor. The features from the second block, on the other hand, are of low-resolution. They attend more to regions near the sensor typically containing large objects. This suggests that AMFM allows our network to consider complex scale variations for object localization.

Qualitative results. We visualize in Fig. 2 detection results on the validation split of KITTI [1]. We can see that our model localizes small and/or occluded objects with sparse point clouds well. This indicates our hybrid 3D representation and scale-aware features are effective to localize hard examples and robust to complex scale variations. We also show failure examples in the last row of Fig. 2. Our model misses the heavily occluded objects, *e.g.*, in the left and middle images, some of which are not visible even in the RGB image. It also does not localize the object captured with little or no point clouds as shown in the right image.

References

[1] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 2, 4

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1

[3] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1

[4] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1

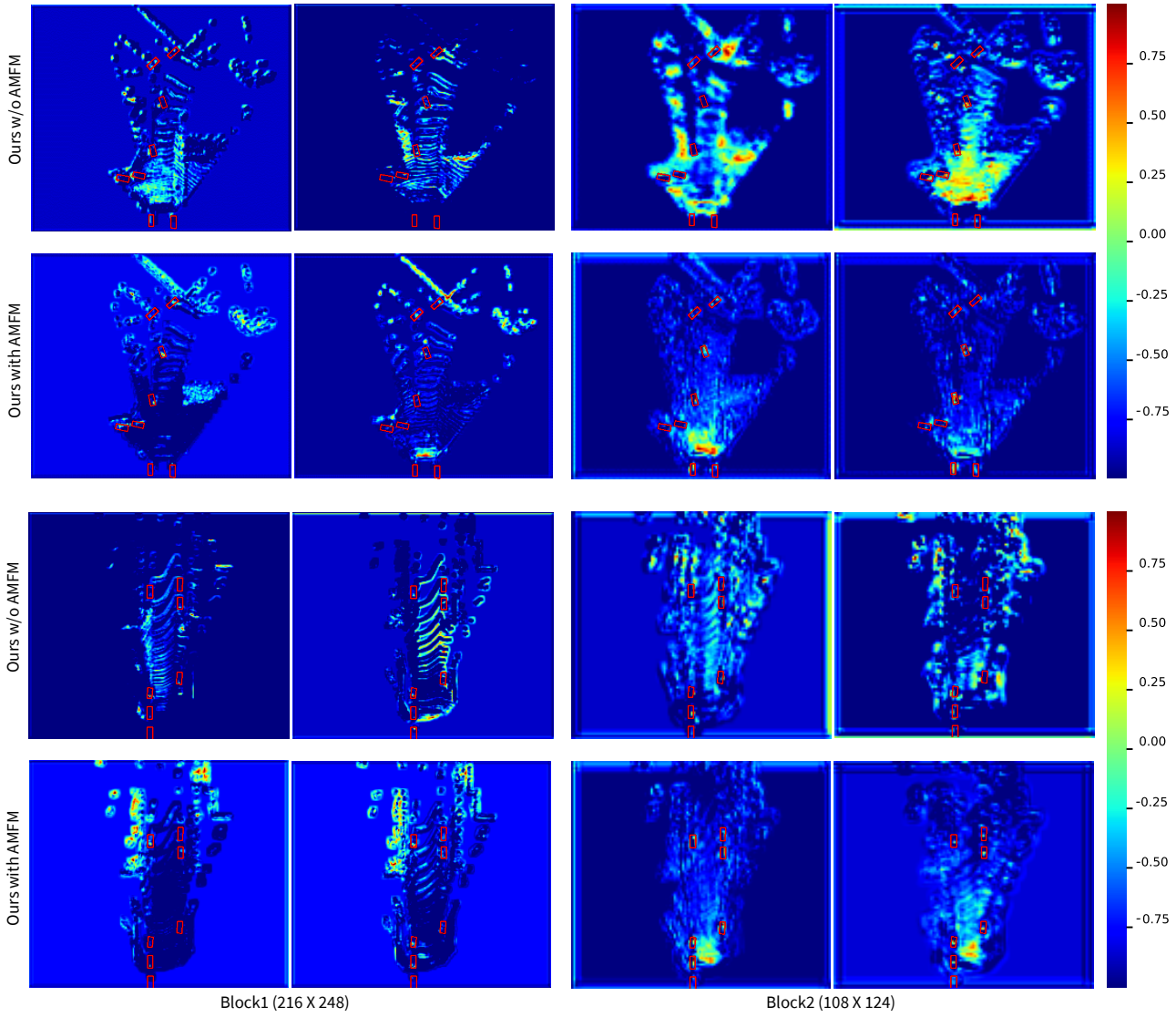

Figure 1: Qualitative comparison of multi-scale and scale-aware features from the first and the second blocks in the backbone network. We visualize feature activations w.r.t input point clouds. The numbers in parentheses indicate a spatial resolution of a feature map, and the red boxes show ground-truth objects. See text for details.

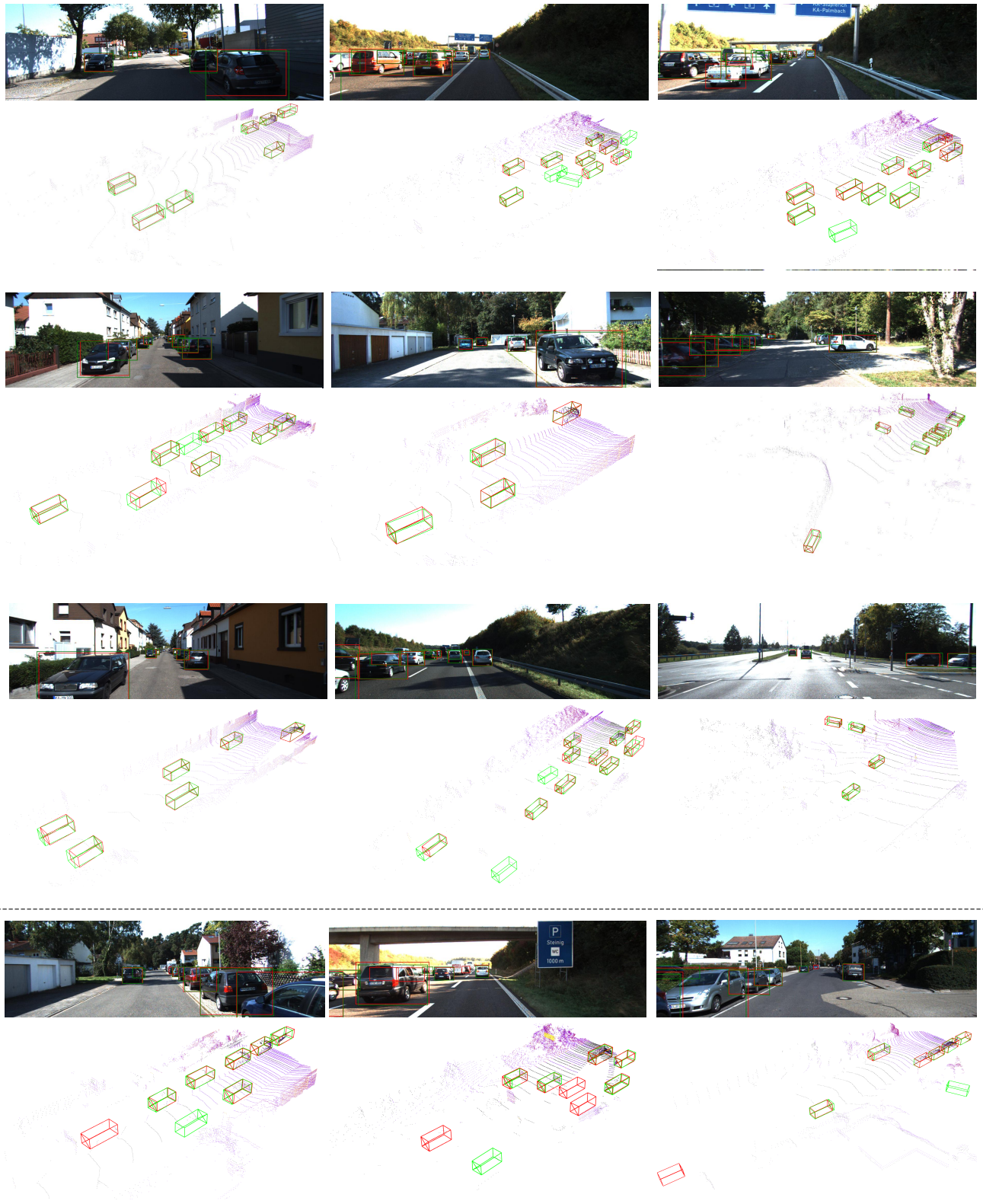

Figure 2: Qualitative results on the validation split of KITTI [1]. Our predictions and ground-truth bounding boxes are shown in green and red, respectively. We also show 2D bounding boxes projected from 3D detection results. The last row shows some failure cases. Best viewed in color.