Supplementary Materials

I. Dataset

Our dataset consists of two parts, real data and simulation data. we augment the 30k real frames with 100k simulation samples. In this supplementary, we provide details about i) the collection and labeling rules and the statistics of our real dataset; ii) the technical details for building our LiDAR simulator; iii) the experiments to shown the effectiveness and the preciseness of both the simulation and the real datasets.

A. Real Data

As shown in Table I, we labeled a total of 30k frames of real LiDAR point cloud frames. The data format is the same as that of KITTI and Apollo datasets. They are split into 20k training samples and 10k test samples for evaluation. There are totally about 500k objects in the training set and 260k foreground objects for test and evaluation. These objects are categorized into eight classes, including: large vehicles (bus, trunk *etc*.), car, pedestrian, motorcyclist, bicyclist, traffic cone, others (*e.g*. animals) and background.

The data was acquired using Apollo cars [1] with a Velodyne HDL-64E S3 laser scanner. The sensor is positioned on the top and center of the car. The range of the vertical scanning angle is $[-24.9^{\circ}, 2.0^{\circ}]$, and the scanning distance range is 120m. During the data acquisition, the scanner speed is set to 10Hz. Each point cloud frame contains about 120k points. The point clouds were also motion-compensated using a high-precision GPS/IMU installed on the vehicle.

To guarantee the data samples' diversity, we selected the data samples from more than 100 sequences (a total of more than 20 hours and 100 miles). We also enforced a minimum gap of 10m or 1s between data frames and dropped samples with less than two few objects.

Data was labeled manually on point cloud with the help and guidance of color images, first by setting the bounding box for each individual object, and second by removing false positives (mainly the points from the ground and the neighborhood objects) from the 3D bounding box.

B. Simulation Data

We augment the dataset with a large amount of simulation data. As shown in Table 1, simulation data can boost the training and improve the AP by $2 \sim 3.5\%$ compared with model trained only on real dataset.

We build our LiDAR simulation system [2] using real backgrounds and hundreds of foreground 3D object models. (All details are also available in our arXiv paper [2].) The backgrounds are collected from four cities and seven different roads/areas.

We capture real world background with a professional 3D scanner Riegl VMX-1HA which can provide the dense and high-quality point cloud from the real traffic scenarios. To obtain a clean background, moving objects (such as cars, pedestrians, bicyclists) can be removed by repeatedly scanning certain traffic scene several rounds (*e.g*., 5 rounds). While, to remove the static movable obstacles, we use a state-of-the-art semantic segmentation approach (Pointnet++ [5]) to get the initial semantic masks and then correct these wrong parts manually.

Then, synthetic models of various objects, such as vehicles and pedestrians, can be inserted to create various traffic scenes. The obstacle placement are based on the probability maps. We use a well-trained VoxelNet [6] to calculate the initial probability distributions of different kinds of objects in different regions of the scenes. With the guidance of these probability maps, we are able to place the objects as real as possible. We also augment the positions and poses to the neighbouring positions based on a Gaussian kernel on the probability maps.

Our simulation LiDAR renderer takes the real static backgrounds and real objects' distributions to generate new realistic LiDAR points, already annotated at the point level for synthetic objects. The domain gaps between simulation and real data can be significantly reduced by acquiring real-world information, such as the backgrounds and the types, numbers, locations, orientations and poses of the real objects.

We compare our simulation data and real data visually in Fig. 1. There is no visual differences between our simulation and real data samples. It's already very hard to differentiate the simulation samples from real samples by just human eyes. Also, in the experiments, as shown in Table II, the model trained only with our simulation dataset achieves a similar accuracy as the model trained with real data. This is especially true for larger objects (*e.g*. car), where the accuracies are very close (only 0.3% in AP).

TABLE I: Statistics (number of obejcts) of our LiDAR point cloud dataset.

Sub sets	Vehicle		Cyclist		Pedestrian	Others		Total
	large vehicle	car	bicyclist	motorcyclist		traffic cone	others	
Test set	21k	150k	8k	14k	30k	6k	28k	260k
Training set	43k	300k	16k	29k	60k	13k	56k	520k
Simulation data	240k	1.5m	100k	170k	320k	75k	–	2.4m
Total	300k	2.0m	120k	210k	410k	90k	80k	3.2m

TABLE II: Evaluations (AP %) and comparisons of models trained on real data and simulation data.

dataset	car	large vehicle	pedestrian	bicyclist	motorcyclist	Traffic cone
simulation	86.1	69.5	50.1	35.1	44.1	52.1
real	86.4	71.4	60.7	43.5	52.3	60
sim+real	89	74.9	62.4	45.1	55	61.8

TABLE III: Speed comparisons of different models (TESLA V100 GPU).

models	SGPN [52]	VoxelNet [58]	our model (sparse feature)	our model (dense feature)
Elapsed time	1000ms	250ms	210ms	300ms

TABLE IV: AP^{70}/AP^{95} evaluations of common classes on our test set

Method + traning set	car	pedestrian	bicyclist
ours (seg.) + kitti [3] (bbox)	74.3/49.7	47.1/17.3	21.7/9.1
ours (seg.) + Nuscenes [4] (bbox)	80.3/59.9	57.5/21.1	27.2/14.3
ours (seg.) + ours (bbox)	86.9/63.7	60.3/24.4	31.2/15.2
ours (seg.) + ours (point label)	**90.1/78.7**	**69.1/41.8**	**39.9/29.0**

II. Experiments

In this supplementary, we provide more results of our model and the comparisons with the state-of-the-art methods (in Fig. 2 and 3).

We also compare the efficiency/speed of our method with other approaches in Table III. Our method can run at a speed of 3∼5 fps on a TESLA V100 GPU, which is similar to the state-of-the-art 3D detector [6] and three times faster than instance segmentation model SGPN [7]. The dense feature encoding in our model takes about 100ms, but the whole model can be further accelerated to about 200ms per frame by reducing K in KNN algorithm from 64 to 32 (with only 0.9% decrease in AP), which is even faster than spare features.

During the data labeling for instance segments, we find that 12-16% outliers are mixed into the ground-truth bounding boxes and 76% bounding boxes have outliers. This means a 100% accurate 3D detector only has about 90% accuracy for segmentation. Furthermore, any imperfection (errors of angle, size, *etc*.) of the detected bounding boxes will heavily reduce the precision of the segments. State-of-the-art detectors are compared in Table IV of the paper.

To prove the effectiveness and preciseness of our instance segmentation labels (both bounding boxes and point-wise labels). We compare different models trained on several existing LiDAR datasets. Since there is no point-wise labels in these datasets, we treat all the points in the bounding boxes as the instance segments. Results of the instance segmentation models trained with "bounding boxes" of different datasets are shown in Table IV. Evaluations are done on our test set which has the instance segment labels. Gaps are very large (14∼20% in AP^{95}). This shows that point-wise labels are necessary in training instance segmentation models.

References

[1] Baidu. Apollo dataset, http://data.apollo.auto/?locale=en-us&lang=en. 1

[2] J. Fang, F. Yan, T. Zhao, F. Zhang, D. Zhou, R. Yang, Y. Ma, and L. Wang. Simulating lidar point cloud for autonomous driving using real-world scenes and traffic flows. *arXiv preprint arXiv:1811.07112*, 2018. 1

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition*, 2012. 2

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 2

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 1

[6] 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 (CVPR)*, pages 4490–4499, 2018. 1, 2

[7] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.

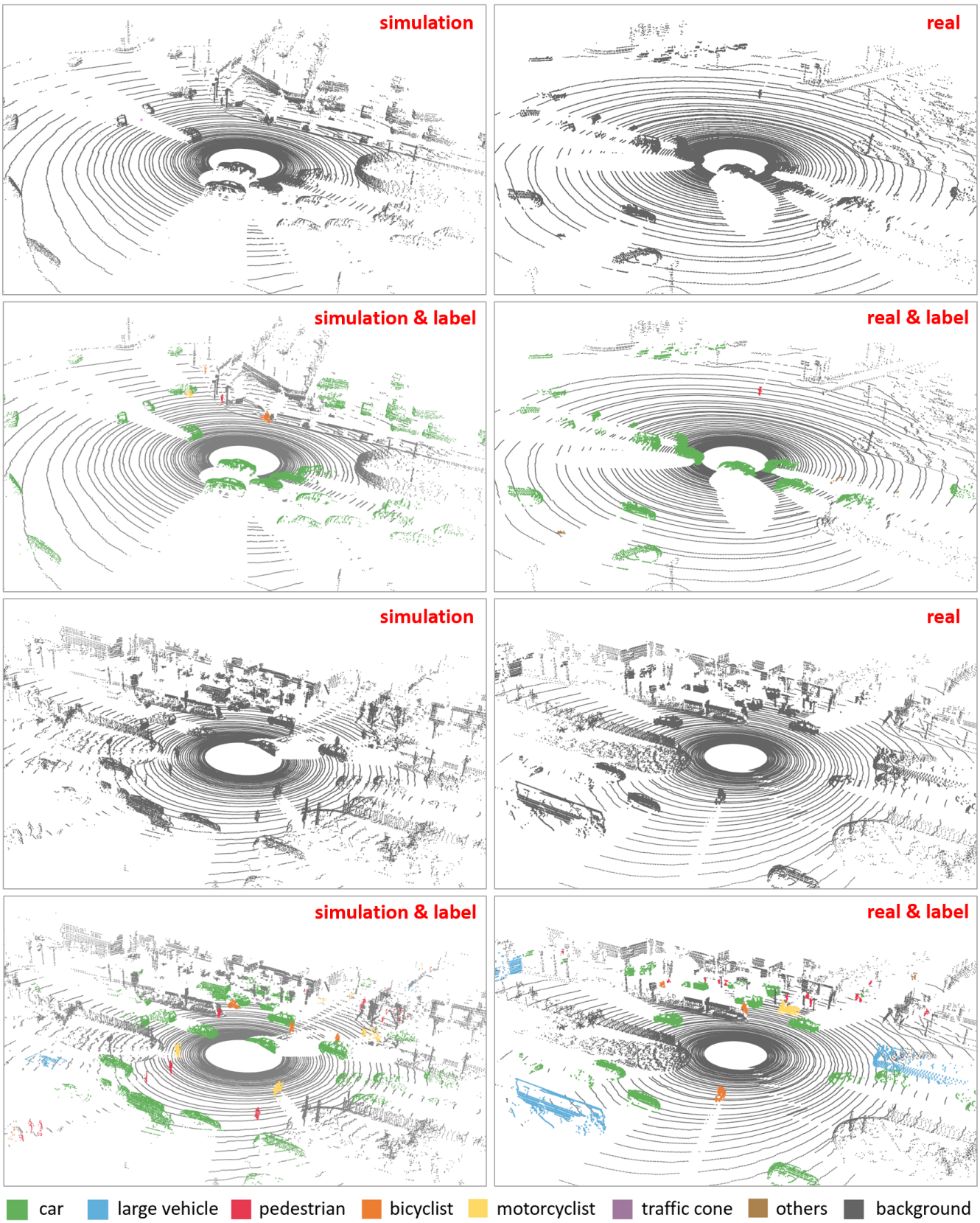

Fig. 1: Visual comparisons between simulation data and real data. ***Left:*** simulation samples and the corresponding point-wise labels. ***Right:*** real samples and the corresponding point-wise labels. The domain gaps between our simulation data and the real data are very small. There is no visual differences between them.

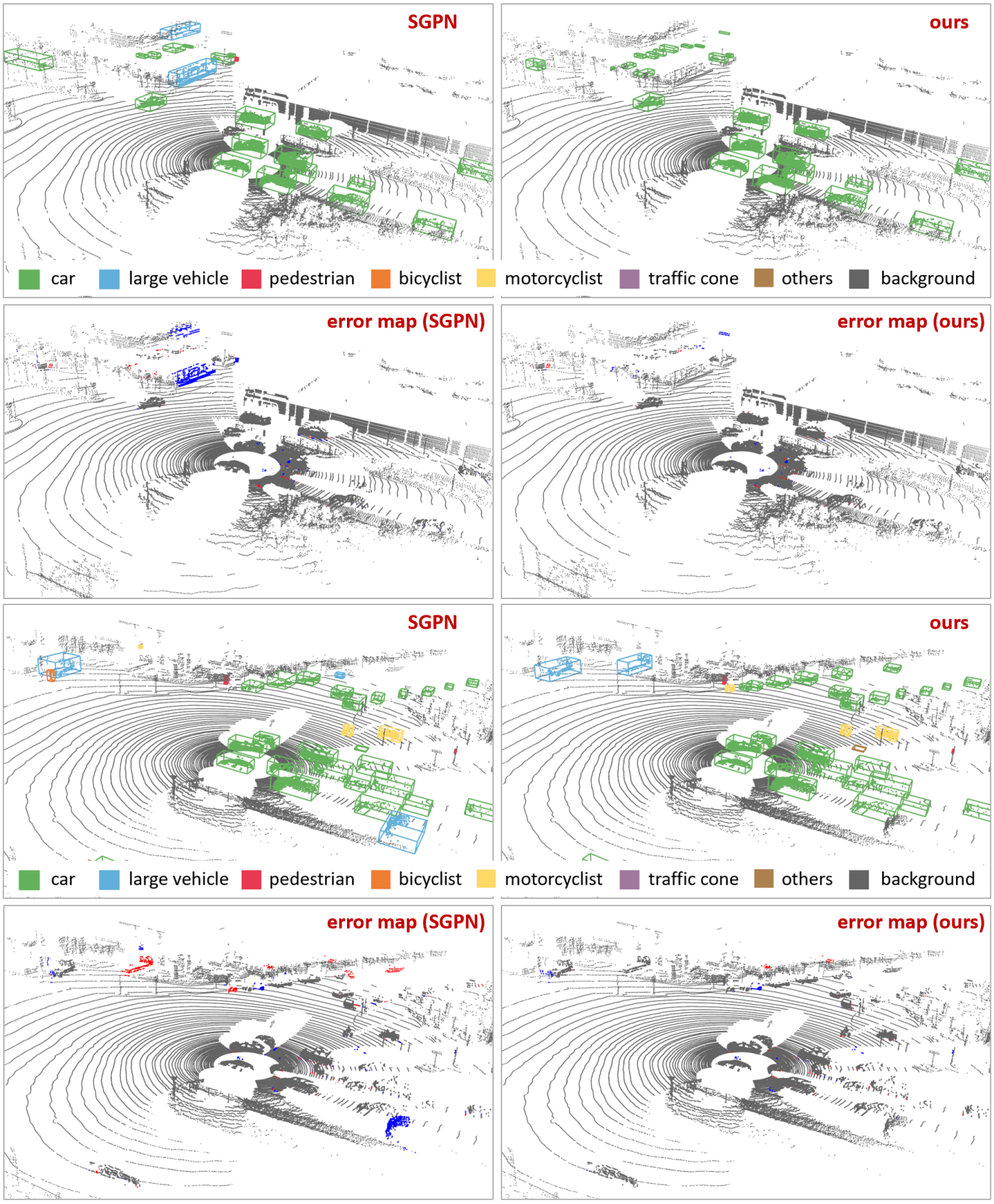

Fig. 2: Visual comparisons between SGPN [52] and our model in complex and challenging scenes. ***Left:*** results of SGPN [52] and the corresponding error maps. ***Right:*** results of our model and the corresponding error maps. Bounding boxes are estimated directly from the instance segments. In the error maps, ***red*** represents the ***false negatives (FN)***, ***blue*** means the ***false positives (FP)***. Many small objects are missed in SGPN. It also picks up many false positives.

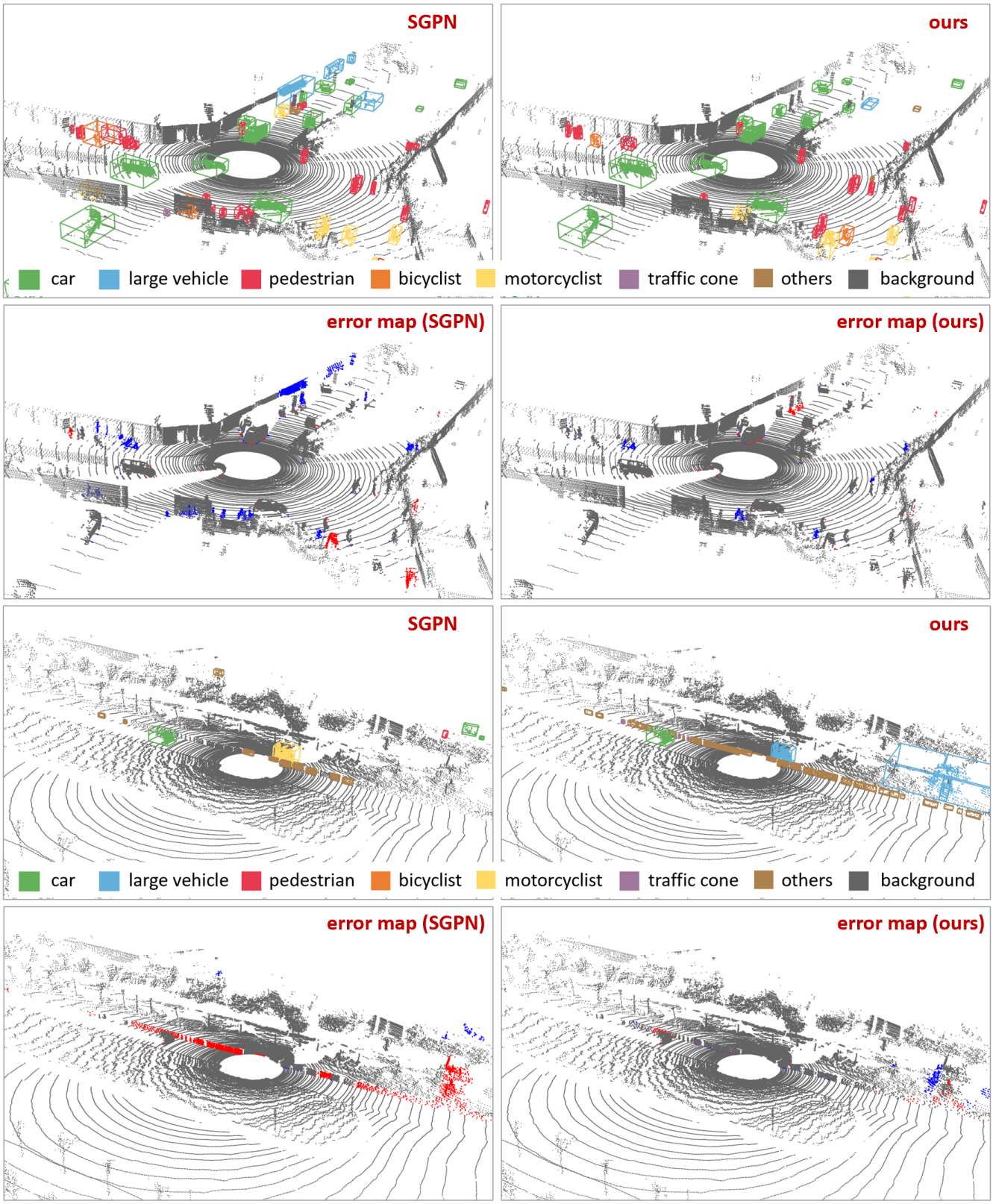

Fig. 3: Visual comparisons between SGPN [52] and our model in complex and challenging scenes. ***Left:*** results of SGPN [52] and the corresponding error maps. ***Right:*** results of our model and the corresponding error maps. Bounding boxes are estimated directly from the instance segments. In the error maps, ***red*** represents the ***false negatives (FN)***, ***blue*** means the ***false positives (FP)***. Many small objects are missed in SGPN. It also picks up many false positives.