LiDAR-based Panoptic Segmentation via Dynamic Shifting Network — *Supplementary Material*

Fangzhou Hong[1,3] Hui Zhou[2] Xinge Zhu[3] Hongsheng Li[3,4] Ziwei Liu[1]✉
[1]S-Lab, Nanyang Technological University [2]Sensetime Research
[3]CUHK-SenseTime Joint Laboratory, The Chinese University of Hong Kong
[4]School of CST, Xidian University
{fangzhou001, ziwei.liu}@ntu.edu.sg zhouhui@sensetime.com
zx018@ie.cuhk.edu.hk hsli@ee.cuhk.edu.hk

In this supplementary material, we provide the following sections for a better understanding of the main paper. Firstly, the details of the heuristic clustering algorithms mentioned in the main paper are provided (Sec. 1). Then, we provide analyses of the differentiability of the dynamic shifting to give some insights into our design (Sec. 2). We provide further analysis of the dynamic shifting module and time consumption (Sec. 3). Moreover, we report the implementation details of DS-Net for the reproducibility (Sec. 4). Last but not least, the per-class results are reported (Sec. 5), and more visualization examples are displayed (Sec. 6).

1. Details of Heuristic Clustering Algorithms

Breadth First Search (BFS). BFS is simple but effective for indoor point clouds. For the points to be clustered, BFS first constructs a graph where edges connect point pairs that are closer than a given radius. Then each connected subgraph is considered a cluster. BFS has shown that it is capable of performing high-quality clustering for indoor point clouds [4] which are dense, even and complete. However, it does not apply to LiDAR point clouds. As discussed in the main paper, large density difference within and between clusters means that the fixed radius can not properly adapt to different clusters. Therefore, it is not a good idea to use BFS as the clustering algorithm for autonomous driving scenes.

DBSCAN [3] and HDBSCAN [1]. Both DBSCAN and HDBSCAN are density-based clustering algorithms which make them perform badly on LiDAR point clouds. Similar to BFS, DBSCAN also constructs a graph based on the mutual distances of the points. Although the new concept of *core point* is introduced to filter out noise points, the problem brought by the fixed radius also occurs in this algorithm. Moreover, the mechanism of noise points recognition also brings problems. For any points to be clustered, if there exists less than a certain number of points within a certain radius, the point is labeled as a noise point. However, the number and densities of points inside instances vary greatly, which makes it hard to determine the line between instances with little points and noise points.

HDBSCAN has a more complex rule of constructing graphs and is claimed to be more robust to density changing than DBSCAN. The *mutual reachable distance* replaced euclidean distance as the indicators of graph construction, which makes it more robust to density changes. Moreover, with the help of the cluster hierarchy, DBSCAN can automatically adapt to data with different distributions. However, by introducing the concept of *mutual reachable distance*, HDBSCAN intuitively assumes that the points with lower density are more likely to be *seas* (noise points) that separate *lands* (valid clusters), which is not the case in LiDAR point clouds where low-density point clouds can also be valid instances that are far away from the LiDAR sensor. Thus DBSCAN and HDBSCAN can not provide high-quality clustering results.

Mean Shift [2]. Mean Shift performs clustering in a very different way than the above three algorithms. Firstly, seeding points are sampled from the points to be clustered. Then, seeding points are iteratively shifted towards cluster centers in order to obtain centers of all clusters. The positions where seeding points are shifted to is calculated by applying the kernel function on corresponding points. In our implementation, we use the flat kernel which takes the mean of all points within a query ball as the result. The radius of the ball is denoted *bandwidth*. After several iterations, all the shifted points have converged and the cluster centers are extracted from the converged points. All the points to be clustered are assigned to the nearest cluster centers, which produces the final instance IDs. The advantage of Mean Shift is that the kernel function is not sensitive to density changes and robust to noise points, which makes it more suitable than density-based clustering algorithms.

However, Mean Shift is not perfect. The choice of pa-

rameters of the kernel function, which is the *bandwidth* in this case, is not trivial. The bandwidth controls the range that the kernel function is applied on. Small bandwidth would mislead the regressed centers of a single instance shifting to several different cluster centers and cause over-segmentation. On the contrary, large bandwidth would mislead regressed centers of neighboring instances shifting to one cluster center and result in under-segmentation. The performance of the classes with relatively small size drops with the bandwidth increasing and vice versa. Therefore, the fixed bandwidth can not handle large and small instances simultaneously. Besides, assigning each point to the cluster centers is not reasonable for that nearby instances may have different sizes which would lead to a different degree of dispersion of regressed centers. For example, edge points of a large instance may be farther to its center than that of nearby instances. Although Mean Shift is not as bad as density-based clustering algorithms, there remains a lot of room for improvement.

2. Gradient Calculation of Dynamic Shifting

In this chapter, we are going to show the gradients of one dynamic shifting iteration and that directly regressing bandwidth with flat kernel is not differentiable. The forward pass of dynamic shifting can be broken down to the following steps:

$$K_j = (XX^T \leq \delta_j) \tag{1}$$
$$D_j = K_j \mathbf{1} \tag{2}$$
$$S_j = D_j^{-1} K_j X \tag{3}$$
$$w_j = f(F, p_j) \tag{4}$$
$$W_j = w_j \mathbf{1}^{1\times 3} \tag{5}$$
$$Y_j = W_j \odot S_j \tag{6}$$
$$Y = \sum_{j=1}^{l} Y_j, \tag{7}$$

where X represents the seeding points, f is the combination of Softmax and MLP, p_j is the parameter of f, F is the backbone features, Y is the shifted seeding points, and $\odot$ denotes element-wise product. It is worth noting that S_j, which is defined by equation 1, 2 and 3, is constant and therefore does not have gradients. Assuming c is the loss, backpropagation gradients are calculated as follows.

$$\frac{\partial c}{\partial F} = \frac{\partial f(F, p_j)}{\partial F} \frac{\partial c}{\partial w_j} \tag{8}$$
$$\frac{\partial c}{\partial p_j} = \frac{\partial f(F, p_j)}{\partial p_j} \frac{\partial c}{\partial w_j} \tag{9}$$
$$\frac{\partial c}{\partial w_j} = \frac{\partial c}{\partial W_j} \mathbf{1}^{3\times 1} \tag{10}$$
$$\frac{\partial c}{\partial W_j} = S_j \odot \frac{\partial c}{\partial Y_j} \tag{11}$$
$$\frac{\partial c}{\partial Y_j} = \frac{\partial c}{\partial Y} \tag{12}$$

However, if the bandwidth δ is learnable, then equation 1 will turn to:

$$K = (XX^T \leq \delta), \tag{13}$$

which unfortunately is not differentiable. Therefore, in our ablation study, in order to further demonstrate that direct regression is not a good strategy, we adopt the Gaussian kernel which is formally defined as:

$$K = exp(-\frac{XX^T}{2\delta^2}), \tag{14}$$

where δ is the learnable bandwidth. With the Gaussian kernel, the direct regression version of dynamic shifting is now differentiable.

3. Further Analyses

Number of Iterations Settings. In the dynamic shifting module, other than bandwidth candidates, the hyper-parameter to tune is the number of iterations, which is essential for the final clustering quality because too few iterations would cause insufficient convergence while too many iterations would add unnecessary time and space complexity. As shown in Table 1, we experiment on several different iteration number settings. The best result is achieved when the number of iterations is set to 4 which is the counterpoint of sufficient convergence and efficiency.

Table 1: Experiments on the number of iteration. All results in [%].

Number of Iteration	PQ	PQ†	RQ	SQ	mIoU
1	57.0	62.6	67.4	77.3	63.5
2	57.6	63.1	67.8	77.5	**63.6**
3	**57.7**	63.3	**68.0**	**77.6**	63.4
4	**57.7**	**63.4**	**68.0**	**77.6**	63.5
5	57.5	63.2	67.7	**77.6**	63.4

Learned Bandwidths of Different Iterations. The average learned bandwidths of different iterations are shown in Fig. 1. As expected, as the iteration rounds grow, points of the same instance gather tighter which usually require smaller bandwidths. After four iterations, learned bandwidths of most classes have dropped to 0.2, which is the lowest they can get, meaning that four iterations are enough for *things* points to converge to cluster centers, which further validates the conclusion made in the last paragraph.

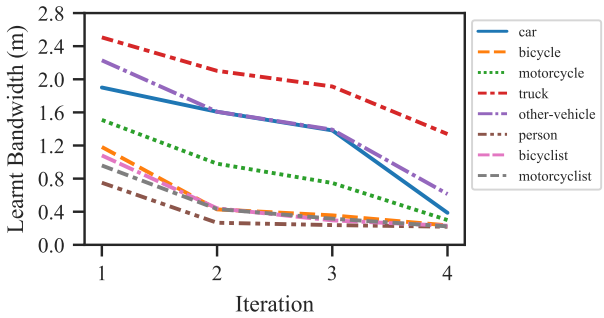

Figure 1: **Relationship Between Iterations and the Learned Bandwidths.** With number of iteration increases, the learned bandwidth decreases. At the 4th iteration, the learned bandwidths of most classes drop near the lower limit.

Inference Time Analysis. Inference time of different modules of DS-Net on nuScenes is reported in Table 2. The dynamic shifting module consumes 33.1 ms per frame, which is attributed to down sampling and gpu-accelerated matrix operations. After kernel operations, the seeding points are mostly converged. Therefore, the final cluster step is not time consuming. Compared to Mean Shifting, which takes 190.3 ms, the proposed dynamic shifting module is efficient and performs better.

Table 2: Inference Time Analysis [ms].

Module	Voxelize	Cylinder	Sem	Ins	DS	Fusion	Other	All
Time (ms)	35.3	106.7	5.3	44.4	33.1	3.3	36.7	264.8

Ablation on Backbone Choices. To demonstrate that the dynamic shifting module can apply to different backbones, we report the performance of a rectangular convolution version of plain backbone and DS-Net in Table 3.

Table 3: Results of replacing cylinder convolutions with rectangular ones on SemanticKITTI validation set [%].

Name	PQ	PQ†	RQ	SQ	mIoU
Rec Plain Backbone	53.2	58.8	64.1	72.6	61.2
Rec DS-Net	**55.5**	**61.0**	**65.7**	**74.2**	**61.4**

4. Implementation Details

Backbone. For both datasets, each input point is represented as a 4 dimension vector including XYZ coordinates and the intensity. The backbone voxelizes a single frame to $480 \times 360 \times 32$ voxels under the cylindrical coordinate system. For that we should not use the information of bounding boxes in this segmentation task, the ground truth center of each instance is approximated by the center of its tight box that parallel to axes which makes a better approximation than the mass centers of the incomplete point clouds. The bandwidth of the Mean Shift used in our backbone method is set to 1.2. Adam solver is utilized to optimize the network. The minimum number of points in a valid instance is set to 50 for SemanticKITTI and 5 for nuScenes.

Dynamic Shifting Module. The number of the FPS downsampled points in the dynamic shifting module is set to 10000. The final heuristic clustering algorithm used in the dynamic shifting module is Mean Shift with 0.65 bandwidth for SemanticKITTI and BFS with 1.2 radius for nuScenes. Bandwidth candidates are set to 0.2, 1.7 and 3.2 for both datasets. The number of Iterations is set to 4 for both datasets. We train the network with the learning rate of 0.002, epoch of 50 and batch size of 4 on four Geforce GTX 1080Ti. Each batch consumes around 7GB of GPU memory. The whole traning process takes about 3-4 days. The dynamic shifting module only takes 3-5 hours to train on top of a pretrained backbone.

5. Per-class Evaluation Results

Detailed per-class PQ, RQ and SQ results are presented in table 4, 5 and 6 respectively. All the results are reported on the held-out test set of SemanticKITTI. "$*$" denotes the unpublished method which is in 2nd place on the public benchmark of SemanticKITTI (accessed on 2020-11-16). Compared to semantic segmentation + detection baseline methods, our DS-Net has huge advantages in *things* classes in terms of PQ, RQ and SQ. With the help of the dynamic shifting module, our DS-Net surpasses the backbone (with fusion module) in all classes in all three metrics, which demonstrates the effectiveness of the novel dynamic shifting module. Moreover, our DS-Net shows superiority in most classes compared with 2nd place method "Polarnet_seg". Fig. 2 gives the screenshot of the public leaderboard of SemanticKITTI at 2020-11-16 and our DS-Net achieves 1st place.

6. More Visualization Results of the DS-Net

We further show the qualitative comparison of our backbone and DS-Net. As shown in Fig. 3, 4 and 5, a total of 9 LiDAR frames from the validation set of SemanticKITTI are taken out for visualization. The left columns are the visualization of the results of our bare backbone with the consensus-driven fusion module. The middle columns show the results of the DS-Net and the right columns are the ground truth. For each frame, a region of interest (as framed in red) is zoomed in to show that our DS-Net is capable of correctly handling instances with different sizes and densities while the backbone method tends to either over-segment or under-segment in these complex cases. Please note that all the *stuff* classes points are colored following the official definition while the colors of *things* instances are randomly picked. Since the order of the predicted instance IDs is different from that of the ground truth, the colors of the *things* instances cannot correspond in predictions and the ground truth.

Table 4: Detailed per-class PQ results on the test set of SemanticKITTI.

Method	PQ	car	truck	bicycle	motorcycle	other-vehicle	person	bicyclist	motorcyclist	road	sidewalk	parking	other ground	building	vegetation	trunk	terrain	fence	pole	traffic sign
KPConv[8] + PointPillars[5]	44.5	72.5	17.2	9.2	30.8	19.6	29.9	59.4	22.8	84.6	60.1	34.1	**8.8**	80.7	77.6	53.9	42.2	49.0	46.2	46.8
RangeNet++[6] + PointPillars[5]	37.1	66.9	6.7	3.1	16.2	8.8	14.6	31.8	13.5	**90.6**	63.2	**41.3**	6.7	79.2	71.2	34.6	37.4	38.2	32.8	47.4
KPConv[8] + PV-RCNN[7]	50.2	84.5	21.9	9.9	34.2	25.6	51.1	67.9	43.8	84.9	**63.6**	37.1	8.4	83.7	78.3	57.5	42.3	**51.1**	51.0	57.4
PolarNet_seg*	53.3	88.7	**31.7**	34.6	**50.9**	**39.1**	57.5	68.7	45.1	88.1	59.7	40.5	1.0	**85.7**	77.7	53.2	39.7	44.8	48.6	57.6
Backbone with Fusion	53.1	90.6	15.8	44.2	46.9	28.5	63.1	67.7	47.6	88.2	59.4	29.5	3.0	82.5	79.0	56.6	42.3	48.1	53.2	63.6
DS-Net	**55.9**	**91.2**	28.8	**45.4**	47.2	34.6	**63.6**	**71.1**	**58.5**	89.1	61.2	32.3	4.0	83.2	**79.6**	**58.3**	**43.4**	50.0	**55.2**	**65.3**

Table 5: Detailed per-class RQ results on the test set of SemanticKITTI.

Method	RQ	car	truck	bicycle	motorcycle	other-vehicle	person	bicyclist	motorcyclist	road	sidewalk	parking	other ground	building	vegetation	trunk	terrain	fence	pole	traffic sign
KPConv[8] + PV-RCNN[7]	61.4	94.5	26.9	14.5	43.6	32.0	70.0	76.8	52.6	91.5	**78.7**	47.7	**11.4**	90.2	94.1	76.4	56.5	**66.8**	68.8	74.1
PolarNet_seg*	64.2	96.2	**36.0**	48.5	**58.6**	**43.0**	66.2	77.1	50.3	96.3	74.8	**54.2**	1.7	**92.1**	94.1	72.6	53.9	61.1	66.0	76.3
Backbone with Fusion	64.5	97.4	20.5	61.1	57.2	33.2	74.0	75.6	57.8	96.2	75.1	39.2	5.0	88.8	95.5	75.8	57.1	63.8	72.2	80.4
DS-Net	**66.7**	**97.5**	32.4	**62.2**	56.3	38.9	**74.3**	**78.4**	**62.7**	**96.8**	76.7	42.6	6.4	89.3	**95.7**	**77.5**	**58.3**	65.5	**74.0**	**81.9**

Table 6: Detailed per-class SQ results on the test set of SemanticKITTI.

Method	SQ	car	truck	bicycle	motorcycle	other-vehicle	person	bicyclist	motorcyclist	road	sidewalk	parking	other ground	building	vegetation	trunk	terrain	fence	pole	traffic sign
KPConv[8] + PV-RCNN[7]	80.0	89.4	81.3	68.1	78.4	79.8	73.0	88.4	83.3	**92.9**	**80.8**	**77.8**	**73.4**	92.8	**83.2**	**75.2**	**75.0**	**76.5**	74.2	77.5
PolarNet_seg*	81.1	92.2	88.0	71.4	**86.9**	**90.8**	**87.0**	89.1	89.8	91.4	79.9	74.8	55.4	93.1	82.6	73.3	73.8	73.2	73.6	75.5
Backbone with Fusion	80.3	93.0	77.0	72.3	81.9	85.8	85.3	89.5	82.5	91.8	79.1	75.1	60.2	92.9	82.7	74.7	74.0	75.5	73.6	79.1
DS-Net	**82.3**	**93.6**	**88.9**	**73.0**	83.8	89.0	85.6	**90.7**	**93.3**	92.0	79.8	75.8	61.4	**93.2**	**83.2**	**75.2**	74.4	76.3	**74.5**	**79.7**

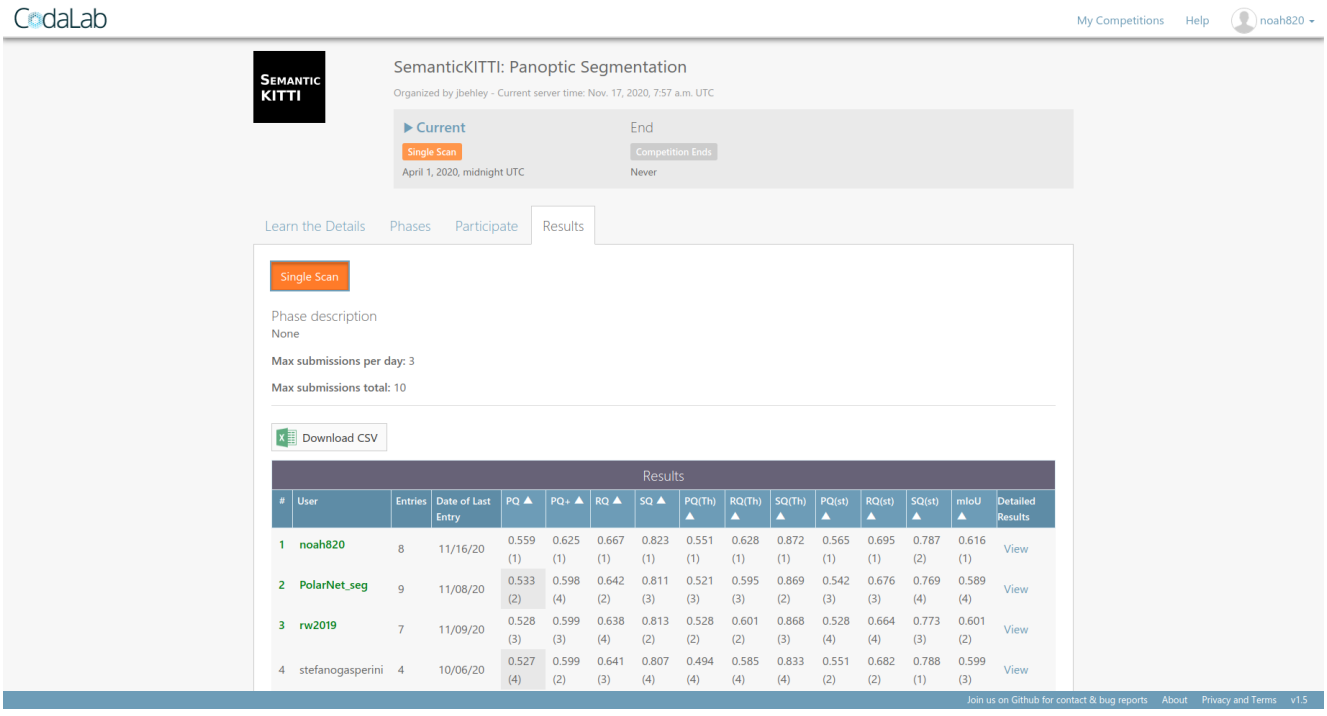

Figure 2: Screenshot of the public leaderboard (https://competitions.codalab.org/competitions/24025) of SemanticKITTI at 2020-11-16. Our method achieves 1st place.

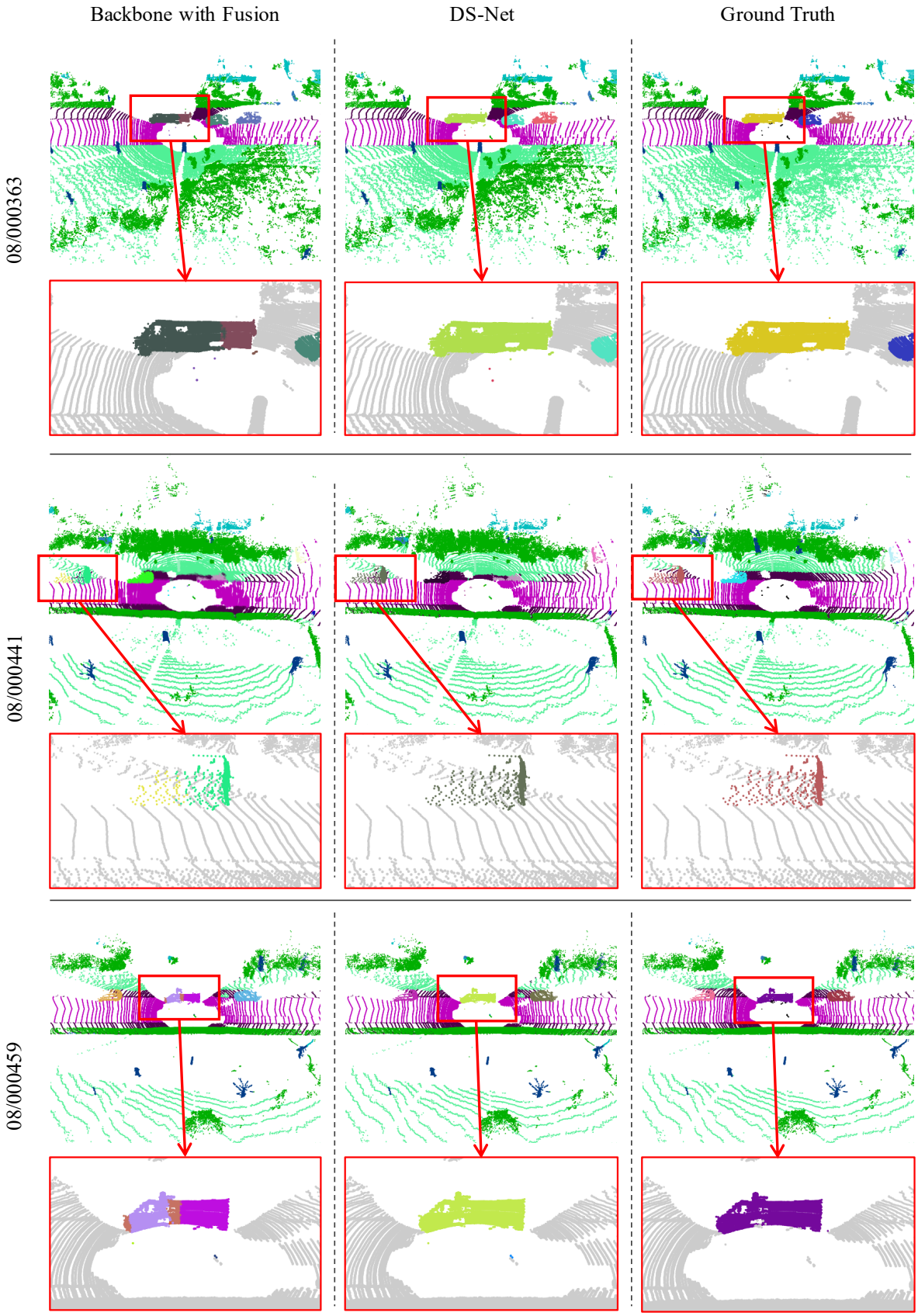

Figure 3: **Qualitative Comparison of the Backbone and DS-Net (1).**

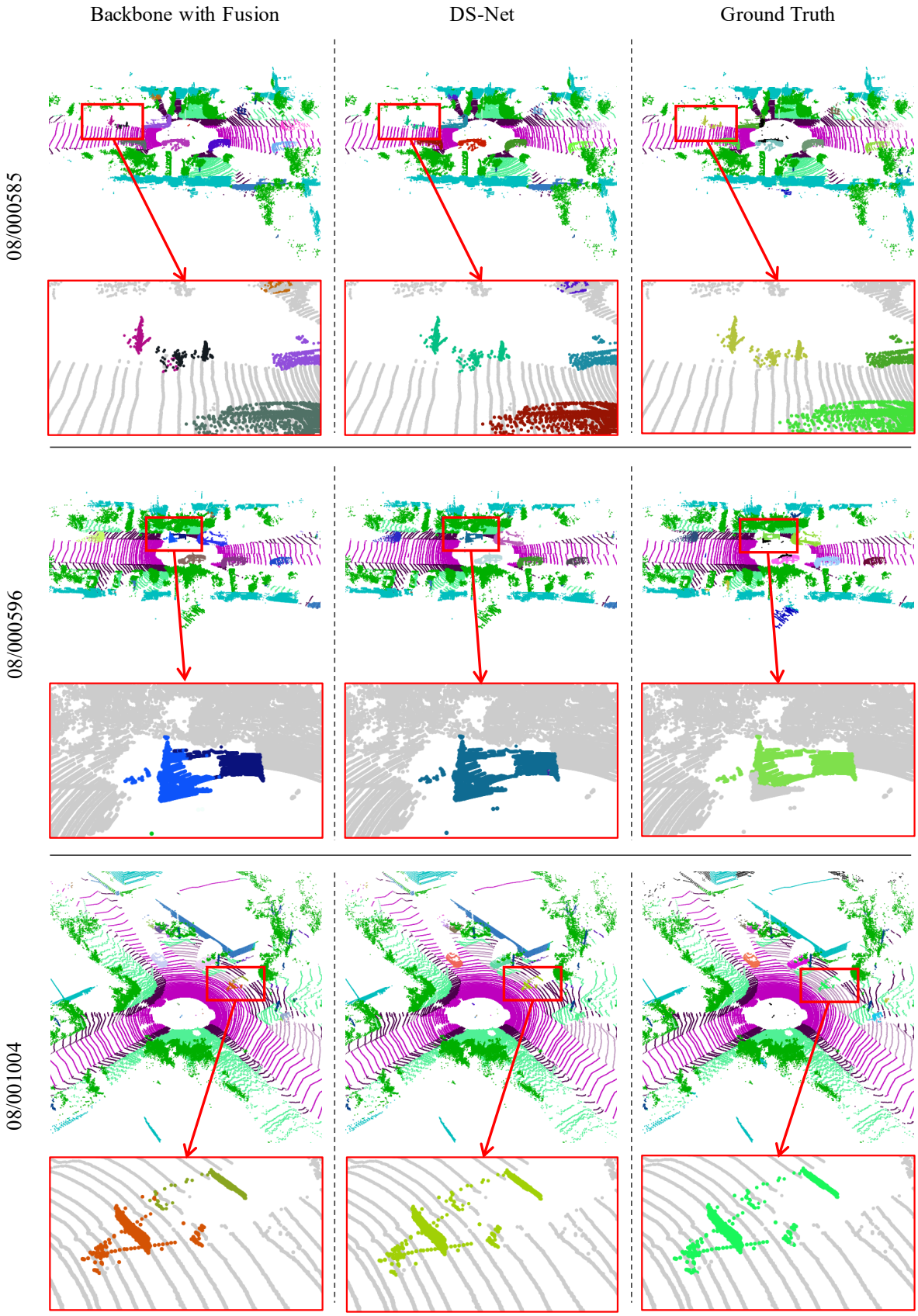

Figure 4: **Qualitative Comparison of the Backbone and DS-Net (2).**

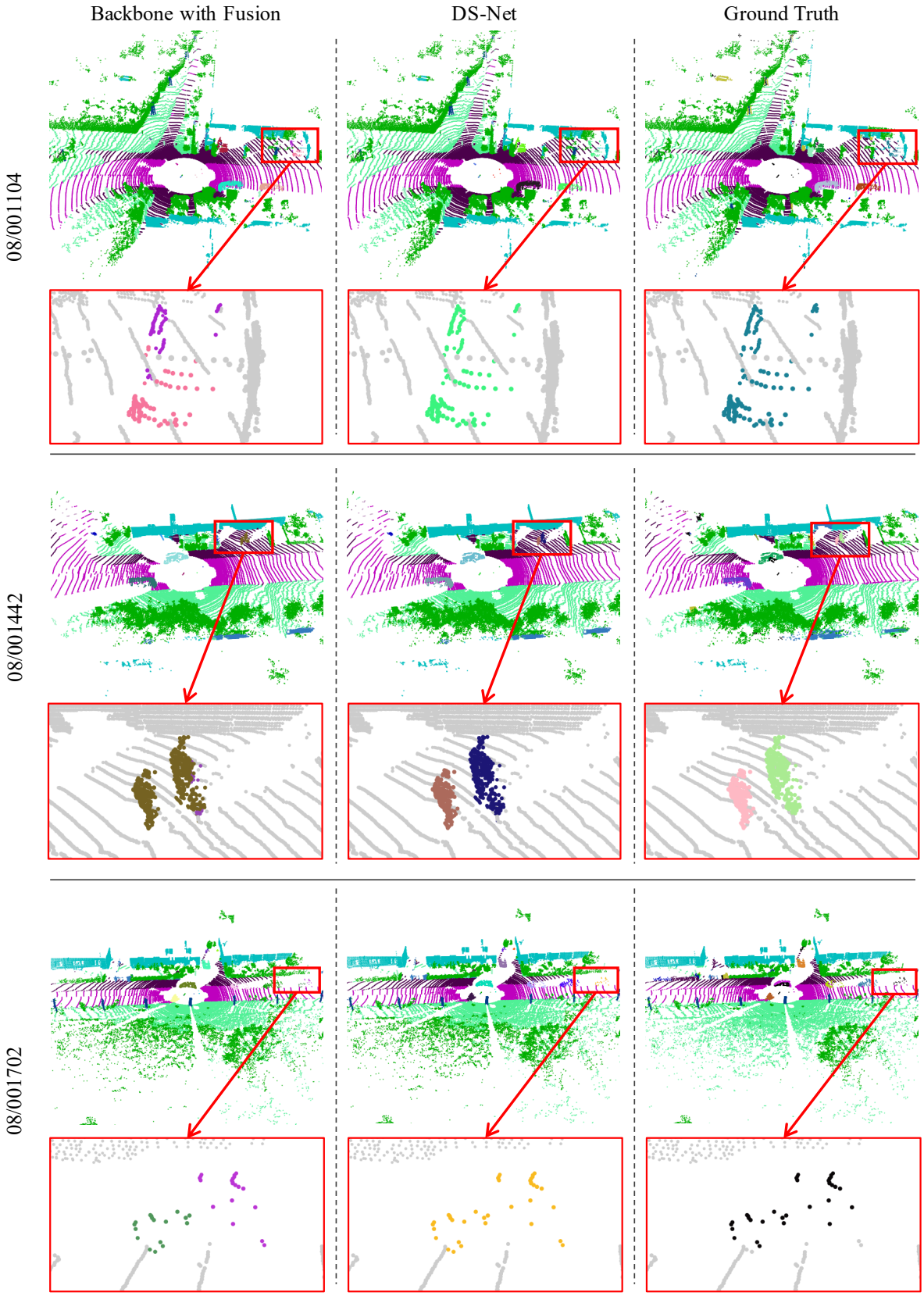

Figure 5: **Qualitative Comparison of the Backbone and DS-Net (3).**

References

[1] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013. 1

[2] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002. 1

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. 1

[4] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020. 1

[5] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 4

[6] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019. 4

[7] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 4

[8] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. 4