RAPiD-Seg: Range-Aware Pointwise Distance Distribution Networks for 3D LiDAR Segmentation

Supplementary Materials

Li Li[1]   Hubert P. H. Shum[1]   Toby P. Breckon[1,2]

Department of {Computer Science[1] | Engineering[2]}, Durham University, UK

{li.li4, hubert.shum, toby.breckon}@durham.ac.uk

In this documentation, we supplement additional materials to support our findings, observations, and experimental results. Specifically, it is organized as follows:

- Sec. A supplements the analysis on RAPiD stability, including rotation stability and translation stability.
- Sec. B supplements the details on RAPiD features and implementations.
- Sec. C supplements additional quantitative results, including a full version of SoTA comparison.
- Sec. D attaches additional qualitative results, with the segmentation error map and magnification of regional details.
- Sec. E supplements the detailed descriptions of semantic classes for SemanticKITTI and nuScenes datasets.
- Sec. F acknowledges the public resources used during the course of this work.

A   The Analysis on RAPiD Invariance

In the main text, we mention that RAPiD exhibits rotation (Sec. A.1) invariance and translation invariance (Sec. A.2). Herein, we provide a detailed analysis and proof.

A.1   Rotation Invariance

Let the point cloud undergoes a rotation represented by a rotation matrix $\mathbf{R}$. The rotated position of a point $\boldsymbol{p}_j$ is $\mathbf{R}\boldsymbol{p}_j$. Since rotation affects only the spatial coordinates and not the reflectivity values, we only need to calculate the spatial coordinate components. The distance of spatial coordinate components between two rotated points is:

$$\|\mathbf{R}\boldsymbol{p}_j - \mathbf{R}\boldsymbol{p}_{j,l}\|_2. \tag{A1}$$

Using the property of rotation matrices ($\mathbf{R}^\top \mathbf{R} = \mathbf{I}$), this simplifies to:

$$\|\boldsymbol{p}_j - \boldsymbol{p}_{j,l}\|_2. \tag{A2}$$

This shows that the distances in RAPiD remain unchanged under rotation, implying rotation stability.
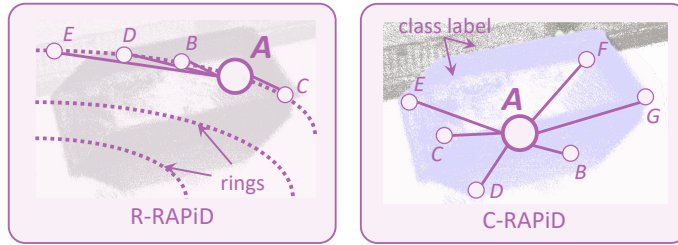
Fig. A1: Visual illustration of R-RAPiD and C-RAPiD.

A.2   Translation Invariance

Let the point cloud undergo a translation represented by a vector $\boldsymbol{t}$. Similarly to rotation, translation affects only the spatial coordinates. The reflectivity values are invariant under translation. The translated position of a point $\boldsymbol{p}_j$ is $\boldsymbol{p}_j + \boldsymbol{t}$. The distance between the two translated points is:

$$\|(\boldsymbol{p}_j + \boldsymbol{t}) - (\boldsymbol{p}_{j,l} + \boldsymbol{t})\|_2. \tag{A3}$$

Simplifying this, we get:

$$\|\boldsymbol{p}_j - \boldsymbol{p}_{j,l}\|_2. \tag{A4}$$

Again, the distances in RAPiD remain unchanged under translation, indicating translation stability.

B   More Details on RAPiD Features

In Sec. B.1, we supplement the details on point-wise RAPiD features as multi-neighboring-point stacked (MNPS) implementation to enhance local representation and noise robustness. We also include details in Sec. B.2 on range-specific parameterization to adapt RAPiD to LiDAR sparsity across different distances, ensuring optimal selection of k values while limiting maximum inter-point distances to address sparsity, particularly in distant ranges.

B.1   Multi-Neighboring-Point Stacked RAPiD

To facilitate the local representation of RAPiD and robustness against noise, we implement point-wise RAPiD features as the multi-neighboring-point stacked (MNPS) features. Specifically, for an anchor point $\boldsymbol{p}$, the k-point RAPiD features are implemented by computing the $(k-1)$-point RAPiD among the sub-pointcloud formed by the k nearest neighbors of the anchor point $\boldsymbol{p}$. This $(k-1)$-point RAPiD thus serves as the MNPS implementation for the current anchor point $\boldsymbol{p}$.

Intuitively, our implementation of MNPS resembles a sliding window mechanism, wherein each anchor point and its neighboring points are considered a small window. This window slides over anchor points to compute the RAPiD

features of the sub-windows. Consequently, our MNPS implementation inherits the advantages of sliding windows in extracting local features, flexibility (various window sizes and sliding steps), computational efficiency (reusing part of the results from the previous window), and noise reduction.

As shown in Fig. A1, we take the computation of 5-point R-RAPiD (left) and 7-point C-RAPiD (right) as an example. For 5-point R-RAPiD of anchor point A, let point B, C, $\cdots$, E be the nearest neighborhoods of A in their belonged ring (the local distances in Fig. A1 are magnified for better visualization). The MNPS implementation $\mathbf{M}_A$ of the 4-point R-RAPiD within a sub-pointcloud composed of B, $\cdots$, E is:

$$
\mathbf{M}_A = \begin{bmatrix}
\boldsymbol{\rho}_{A,B}, & \boldsymbol{\rho}_{A,C}, & \boldsymbol{\rho}_{A,D}, & \boldsymbol{\rho}_{A,E} \\
\boldsymbol{\rho}_{B,A}, & \boldsymbol{\rho}_{B,D}, & \boldsymbol{\rho}_{B,C}, & \boldsymbol{\rho}_{B,E} \\
\boldsymbol{\rho}_{C,A}, & \boldsymbol{\rho}_{C,B}, & \boldsymbol{\rho}_{C,D}, & \boldsymbol{\rho}_{C,E} \\
\boldsymbol{\rho}_{D,B}, & \boldsymbol{\rho}_{D,E}, & \boldsymbol{\rho}_{D,A}, & \boldsymbol{\rho}_{D,C} \\
\boldsymbol{\rho}_{E,D}, & \boldsymbol{\rho}_{E,B}, & \boldsymbol{\rho}_{E,A}, & \boldsymbol{\rho}_{E,C},
\end{bmatrix}
\tag{B5}
$$

where $\boldsymbol{\rho}$ is the 4D distance defined in Sec. 3 in the main text. The computation of C-RAPiD follows a similar approach, with the only difference being that the selection of neighboring points is confined within the semantic category to which the anchor point belongs.

B.2   Range and Sparsity Analysis

Preliminaries: The parameter k represents the number of neighboring points considered for each point in the cloud. For two point clouds with the same basic structure, a small k value (*e.g.*, $k = 5$) will result in a small inter-point distance. A larger k means that the local geometry of the point clouds must be similar over a larger radius.

Our RAPiD are adapted to LiDAR sparsity at different distances by using range-specific parameters k_{close}, k_{mid} and k_{far}, for close, mid, and far ranges, respectively. Herein, we supplement the hyper-parameter analysis of various ranges R and k.

As shown in Fig. B1 (right), $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are two adjacent points on the same LiDAR ring, with the LiDAR being at a range of R. According to the principles of trigonometric geometry, the inter-point distance between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is:

$$
\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|_2 = 2R\sin(\theta/2),
\tag{B6}
$$

where θ is the LiDAR angular resolution (horizontal/azimuth). For SemanticKITTI dataset, $\theta = 0.09\,\text{deg}$ [5]; for nuScenes dataset, $\theta = 0.1 \sim 0.4\,\text{deg}$ [3]. We aim for our k-point RAPiD to focus on local geometric structures; therefore, in areas where the point cloud is sparse, we seek to avoid excessively large inter-point distances that can result from an overly large k. For the k-point RAPiD in the extreme case, we constrain the possible maximum inter-point distance from exceeding a certain threshold $\delta_{\max}$:

$$
\delta_{\max} = (k-1)\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|_2 = 2(k-1)R\sin(\theta/2).
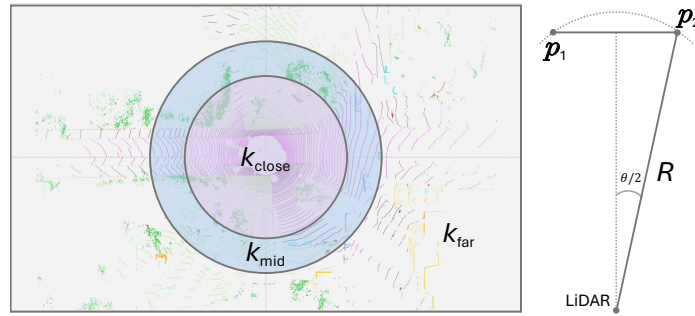\tag{B7}
$$

Fig. B1: The illustration of range R and k.

Experimentally, we select $\delta_{\max} = 0.25$, and subsequently determined the corresponding combinations of k and R via Eq. (B7). The proportional visualization in Fig. B1 (left) demonstrates that our range division is rational, effectively distinguishing various point cloud sparsities based on range R, thereby facilitating the selection of appropriate k.

Due to the varying sizes of k-point RAPiD features resulting from different values of k, we input them into an RAE to encode into fixed-sized RAPiD embeddings.
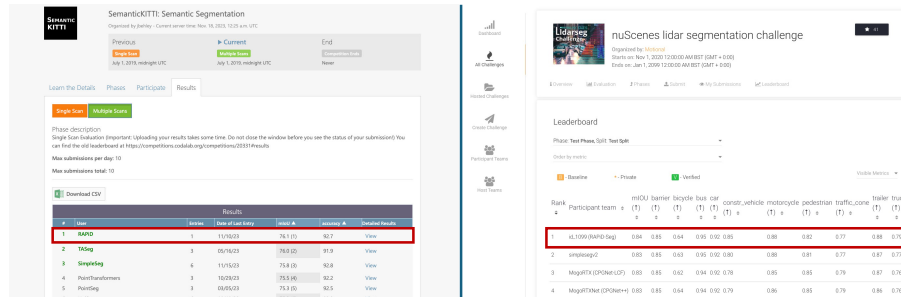
C   More Quantitative Results

Fig. C1: Official leaderboard screenshots. Our method is termed as `RAPiD` and `id_1099` (`RAPiD-Seg`) on SemanticKITTI (left) and nuScenes (right) datasets, respectively.

Comparing with SoTA Methods: In Tab. C1, we showcase the performance of our RAPiD-Seg LiDAR segmentation method on SemanticKITTI *test* set in comparison with a full list of published leading contemporary SoTA approaches to demonstrate its superior efficacy (the full version of Tab. 1 in the main text). Fig. C1 shows the official leaderboard screenshots of all SoTA methods on the SemanticKITTI and nuScenes datasets, upon which we rank first.

Table C1: Quantitative results of RAPiD-Seg and SoTA segmentation methods on SemanticKITTI [2] *test* set; **Best**/<u>2nd best</u> highlighted.

Method	mIoU	car	bicy	moto	truc	o.veh	ped	b.list	m.list	road	park	walk	o.gro	build	fenc	veg	trun	terr	pole	sign
AMVNet [9]	65.3	96.2	59.9	54.2	48.8	45.7	71.0	65.7	11.0	90.1	71.0	75.8	32.4	92.4	69.1	85.6	71.7	69.6	62.7	67.2
JS3C-Net [14]	66.0	95.8	59.3	52.9	54.3	46.0	69.5	65.4	39.9	88.9	61.9	72.1	31.9	92.5	70.8	84.5	69.8	67.9	60.7	68.7
SPVNAS [12]	66.4	97.3	51.5	50.8	59.8	58.8	65.7	65.2	43.7	90.2	67.6	75.2	16.9	91.3	65.9	86.1	73.4	71.0	64.2	66.9
Cylinder3D [17]	68.9	97.1	67.6	63.8	50.8	58.5	73.7	69.2	48.0	92.2	65.0	77.0	32.3	90.7	66.5	85.6	72.5	69.8	62.4	66.2
AF2S3Net [4]	69.7	94.5	65.4	**86.8**	39.2	41.1	**80.7**	80.4	<u>74.3</u>	91.3	68.8	72.5	**53.5**	87.9	63.2	70.2	68.5	53.7	61.5	<u>71.0</u>
RPVNet [13]	70.3	97.6	68.4	68.7	44.2	61.1	75.9	74.4	73.4	**93.4**	70.3	**80.7**	33.3	<u>93.5</u>	72.1	86.5	<u>75.1</u>	71.7	64.8	61.4
SDSeg3D [8]	70.4	97.4	58.7	54.2	54.9	65.2	70.2	74.4	52.2	90.9	69.4	76.7	41.9	93.2	71.1	86.1	74.3	71.1	65.4	70.6
GASN [16]	70.7	96.9	65.8	58.0	59.3	61.0	<u>80.4</u>	**82.7**	46.3	89.8	66.2	74.6	30.1	92.3	69.6	<u>87.3</u>	73.0	72.5	66.1	**71.6**
PVKD [6]	71.2	97.0	67.9	69.3	53.5	60.2	75.1	73.5	50.5	91.8	70.9	77.5	41.0	92.4	66.5	73.8	71.9	64.9	65.8	
2DPASS [15]	72.9	97.0	63.6	63.4	61.1	61.5	77.9	<u>81.3</u>	74.1	89.7	67.4	74.7	40.0	<u>93.5</u>	**72.9**	86.2	73.9	71.0	65.0	70.4
PCSeg [10]	72.9	97.5	51.2	67.6	58.6	68.6	78.3	80.9	**75.6**	92.5	71.5	78.3	36.9	93.1	71.4	85.4	73.6	69.9	66.1	68.7
RangeFormer [7]	73.3	96.7	69.4	73.7	59.9	66.2	78.1	75.9	58.1	92.4	73.0	78.8	42.4	92.3	70.1	86.6	73.3	<u>72.8</u>	<u>66.4</u>	66.6
UniSeg [11]	<u>75.2</u>	**97.9**	**71.9**	75.2	<u>63.6</u>	<u>74.1</u>	78.9	74.8	60.6	<u>92.6</u>	<u>74.0</u>	<u>79.5</u>	<u>46.1</u>	93.4	<u>72.7</u>	**87.5**	**76.3**	**73.1**	**68.3**	68.5
RAPiD-Seg (Ours)	**76.1**	<u>97.7</u>	<u>71.1</u>	<u>76.2</u>	**72.5**	**80.7**	79.9	79.1	59.8	91.8	**78.2**	78.6	46.0	**93.6**	72.1	86.9	74.6	72.3	65.9	68.5

D   More Qualitative Results

Segmentation Error Map: We present qualitative comparisons with PCSeg [11] and ground truth through error maps in Fig. F1 (SemanticKITTI) and Fig. F2 (nuScenes). The visualization underscores the superior performance of our method, marked by significantly reduced segmentation errors in each analyzed frame.

Magnification of Regional Details: We present a visualization of magnified regional details in Fig. F3 to showcase segmentation details and performance at a long range. The magnified details indicate that our method performs well in segmenting various classes such as other grounds, parking areas, sidewalks, vegetation, *etc*.

RAPiD Embedding Visualizations: We present the visualization of the RAPiD embeddings in Fig. F4. Specifically, we initially train the RAE, following which the pointwise RAPiD features are processed through the RAE to yield the RAPiD embeddings. Given that these RAPiD embeddings belong to a high-dimensional space (exceeding three dimensions), we employ Principal Component Analysis (PCA, [1]) to reduce the dimensionality of the RAPiD embeddings to a 3D representation, thereby facilitating visualization. The results show that our RAPiD features are stable and distinctive among semantic categories. Moreover, embeddings of the same surface material/semantic category/object exhibit consistency across different viewpoints or ranges. This further enhances the performance of our semantic segmentation network RAPiD-Seg, which integrates RAPiD features, achieving commendable segmentation results.

E   The Description on Semantic Classes

We supplement the semantic categories in two datasets with ⬤ visualization colors, **full names**, `abbreviation names`, as well as detailed descriptions [1].

E.1   SemanticKITTI Dataset

There are a total of 19 classes chosen for training and evaluation by merging classes with similar motion statuses and discarding sparsely represented ones.

---

[1] All descriptions are referenced from the official documentations in SemanticKITTI point labeler and nuScenes devkit.

1. ● **car**: This includes cars, jeeps, SUVs, and vans with a continuous body shape (*i.e.*, the driver cabin and cargo compartment are one).
2. ● **bicycle** (`bicy`): Includes bicycles without the cyclist or possibly other passengers. The cyclist and passengers receive the label cyclist.
3. ● **motorcycle** (`moto`): This includes motorcycles and mopeds without the driver or other passengers. Both driver and passengers receive the label motorcyclist.
4. ● **truck** (`truc`): This includes trucks, vans with a body that is separate from the driver cabin, pickup trucks, as well as their attached trailers.
5. ● **other vehicle** (`o.veh`): Caravans, Trailers, and fallback category for vehicles not explicitly defined otherwise in meta category level vehicle.
6. ● **person** (`ped`): Persons moving by their own legs, sitting, or any unusual pose, but not meant to drive a vehicle.
7. ● **bicyclist** (`b.list`): Humans driving a bicycle.
8. ● **motorcyclist** (`m.list`): Persons riding a motorcycle.
9. ● **road**: Paved pathways primarily designed for the movement of vehicles, particularly automobiles, trucks, buses, and motorcycles.
10. ● **parking** (`park`): Areas where vehicles can be parked and left.
11. ● **sidewalk** (`walk`): Areas used mainly by pedestrians, and bicycles, but not meant for driving.
12. ● **other ground** (`o.gro`): Other areas that are not used by pedestrians or meant for driving.
13. ● **building** (`build`): Building walls, doors, *etc*.
14. ● **fence** (`fenc`): fences, small walls, crash barriers, *etc*.
15. ● **vegetation** (`veg`): Trees, and other forms of vertical growing vegetation.
16. ● **trunk** (`trun`): Tree trunks.
17. ● **terrain** (`terr`): Grass and all other types of horizontal spreading vegetation, including soil.
18. ● **pole**: Thin and elongated, typically vertically oriented poles, *e.g.*, sing or traffic signs.
19. ● **traffic sign** (`sign`): Traffic signs without pole.

E.2   nuScenes Datatset

There are a total of 16 classes for LiDAR semantic segmentation, following the amalgamation of akin classes and the removal of rare ones.

1. ● **barrier** (`barr`): Any metal, concrete, or water barrier temporarily placed in the scene in order to re-direct vehicle or pedestrian traffic. In particular, includes barriers used in construction zones. If there are multiple barriers either connected or just placed next to each other, they should be annotated separately.
2. ● **bicycle** (`bicy`): Human or electric powered 2-wheeled vehicle designed to travel at lower speeds either on road surface, sidewalks, or bicycle paths.
3. ● **bus**: Buses designed to carry more than 10 people.

4. ● **car**: Vehicle designed primarily for personal use, *e.g.* sedans, hatch-backs, wagons, vans, mini-vans, SUVs, and jeeps.

5. ● **construction vehicle** (`const`): Vehicles primarily designed for construction. Typically very slow-moving or stationary. Cranes and extremities of construction vehicles are only included in annotations if they interfere with traffic. Trucks used for hauling rocks or building materials are considered trucks rather than construction vehicles.

6. ● **motorcycle** (`motor`): Gasoline or electric powered 2-wheeled vehicle designed to move rapidly (at the speed of standard cars) on the road surface. This category includes all motorcycles, vespas, and scooters. It also includes light 3-wheel vehicles, often with a light plastic roof and open on the sides, that tend to be common in Asia.

7. ● **pedestrian** (`ped`): All types of pedestrians moving around the cityscape.

8. ● **traffic cone** (`cone`): All types of traffic cones.

9. ● **trailer** (`trail`): Any vehicle trailer, both for trucks, cars, and motorcycles (regardless of whether currently being towed or not). Trailers hauled after a semi-tractor should be labeled as `trail`.

10. ● **truck**: Vehicles primarily designed to haul cargo including pick-ups, lorries, trucks, and semi-tractors.

11. ● **driveable surface** (`driv`): All paved or unpaved surfaces that a car can drive on with no concern of traffic rules.

12. ● **other flat** (`other`): All other forms of horizontal ground-level structures that do not belong to any of driveable surface, curb, sidewalk, and terrain. Includes elevated parts of traffic islands, delimiters, rail tracks, stairs with at most 3 steps, and larger bodies of water (lakes, rivers).

13. ● **sidewalk** (`walk`): Sidewalk, pedestrian walkways, bike paths, *etc.* Part of the ground designated for pedestrians or cyclists. Sidewalks do not have to be next to a road.

14. ● **terrain** (`terr`): Natural horizontal surfaces such as ground level horizontal vegetation (≤ 20 cm tall), grass, rolling hills, soil, sand and gravel.

15. ● **manmade** (`made`): Includes man-made structures but not limited to: buildings, walls, guard rails, fences, poles, drainages, hydrants, flags, banners, street signs, electric circuit boxes, traffic lights, parking meters and stairs with more than 3 steps.

16. ● **vegetation** (`veg`): Any vegetation in the frame that is higher than the ground, including bushes, plants, potted plants, trees, *etc.* Only tall grass (≥ 20 cm) is part of this.

E.3   Excluded Semantic Classes

Certain categories (● unlabeled, ● outlier, ● other structure, ● other object, *etc.*), despite being annotated in the dataset ground truth, are excluded from the evaluations and tests. This exclusion is attributed to the inherent noise associated with LiDAR data or other related factors.

F   Public Resources Used

We acknowledge the use of the following public resources, during the course of this work:

- nuScenes[2] ....CC BY-NC-SA 4.0
- nuScenes-devkit[3] ....Apache License 2.0
- SemanticKITTI[4] .... CC BY-NC-SA 4.0
- SemanticKITTI-API[5] ....MIT License
- PCSeg[6] .... Apache License 2.0
- Pointcept[7] ....MIT License
- MinkowskiEngine[8] ....MIT License
- Cylinder3D[9] .... Apache License 2.0
- VoxSeT[10] ....MIT License
- LiM3D[11] .... Apache License 2.0
- SpConv[12] .... Apache License 2.0
- Average-Minimum-Distance[13] ....CC BY-NC-SA 4.0
- PyTorch-Lightning[14] .... Apache License 2.0

(**NOT THE END**; *visualization images follow*)

---

[2] https://www.nuscenes.org/nuscenes.
[3] https://github.com/nutonomy/nuscenes-devkit.
[4] http://semantic-kitti.org.
[5] https://github.com/PRBonn/semantic-kitti-api.
[6] https://github.com/PJLab-ADG/PCSeg.
[7] https://github.com/Pointcept/Pointcept.
[8] https://github.com/NVIDIA/MinkowskiEngine.
[9] https://github.com/xinge008/Cylinder3D.
[10] https://github.com/skyhehe123/VoxSeT.
[11] https://github.com/l1997i/lim3d.
[12] https://github.com/traveller59/spconv.
[13] https://github.com/dwiddo/average-minimum-distance.
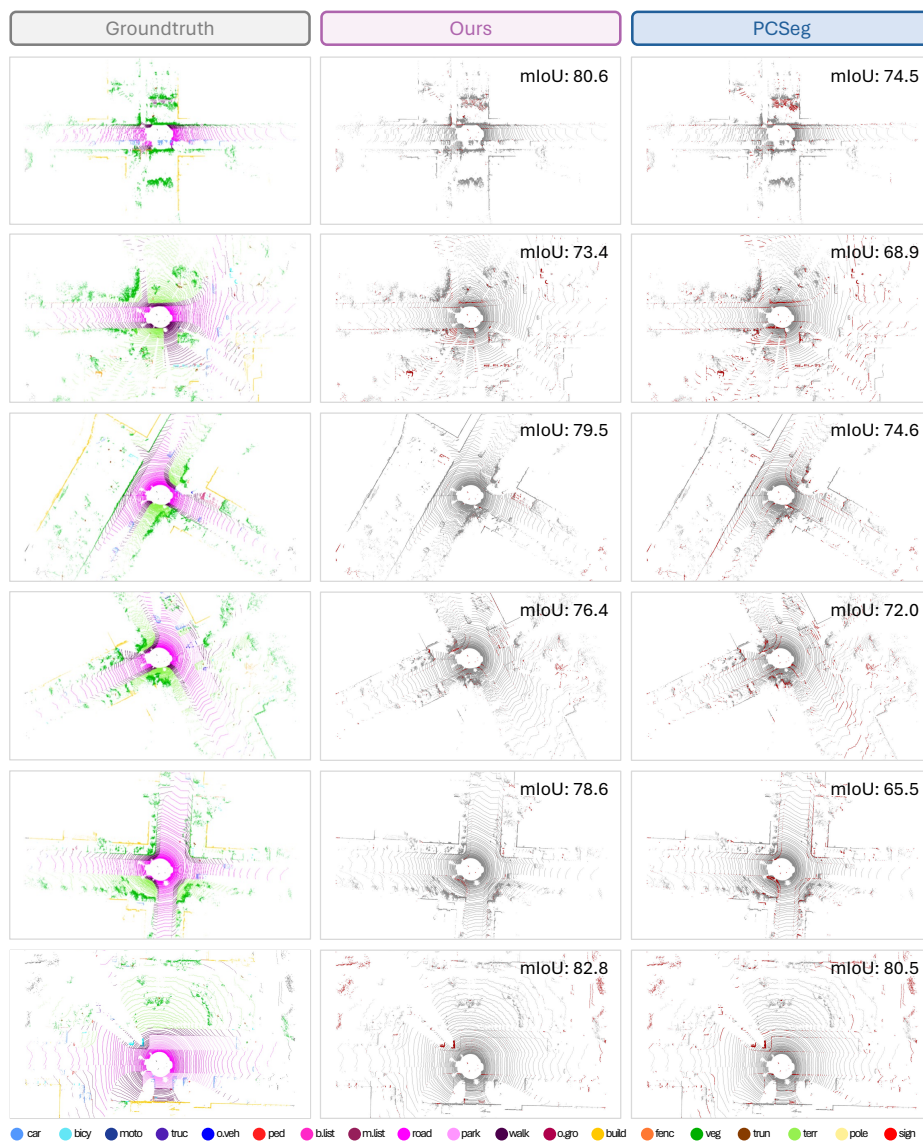[14] https://github.com/Lightning-AI/lightning.

Fig. F1: Qualitative comparisons with PCSeg [10] and groundtruth through error maps on SemanticKITTI [2] *validation* set. To highlight the differences, the **correct** / **incorrect** predictions are painted in **gray** / **dark red**, respectively. Each scene is visualized from the ego-vehicle LiDAR bird's eye view (BEV) and covers a region of 50m by 30m. Best viewed in colors.
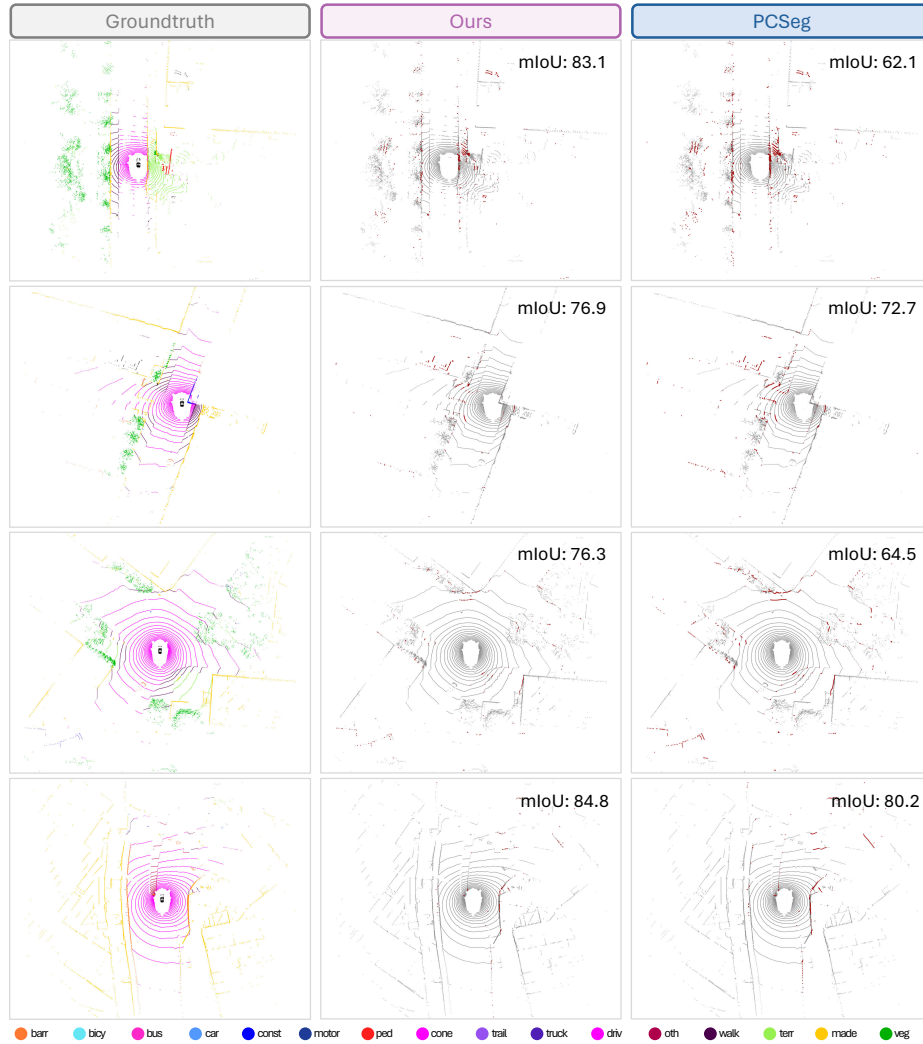
Fig. F2: Qualitative comparisons with PCSeg [10] and groundtruth through error maps on nuScenes [3] *validation* set. To highlight the differences, the **correct** / **incorrect** predictions are painted in **gray** / **dark red**, respectively. Each scene is visualized from the ego-vehicle LiDAR bird's eye view (BEV) and covers a region of 50m by 40m. Best viewed in colors.

Fig. F3: Magnification of regional details: comparing with PCSeg [10] and groundtruth on SemanticKITTI [2] *validation* set. To highlight the differences, areas of improvement are highlighted in green, and areas of underperformance in red. Best viewed in colors.
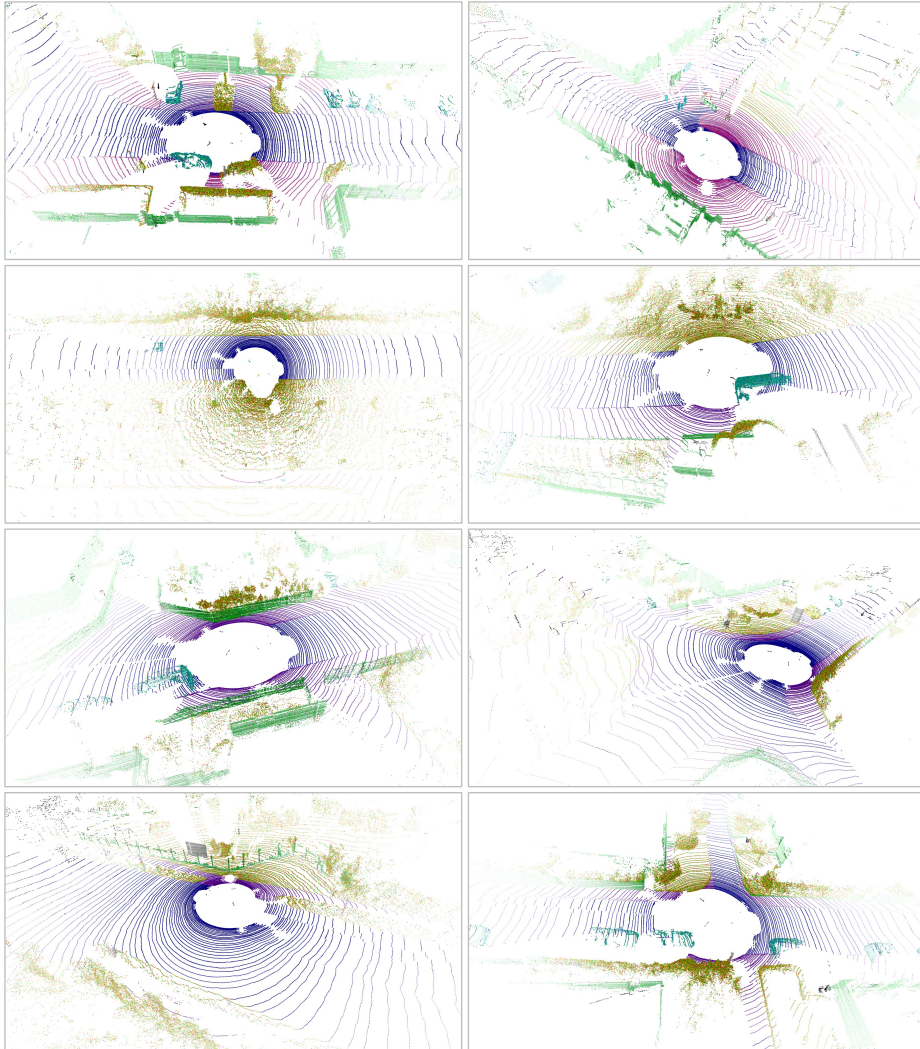
Fig. F4: Our method learns a high-dimensional RAPiD latent representation for capturing the localized geometric structure of neighboring points. We apply PCA [1] to reduce the latent dimension to 3 and plot as RGB. Different colors represent various RAPiD 3D representations. Best viewed in colors.

References

1. Abdi, H., Williams, L.J.: Principal component analysis. WIREs Comput. Stat. **2**(4), 433–459 (2010). https://doi.org/10.1002/wics.101 5, 12
2. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In: Int. Conf. Comput. Vis. pp. 9296–9306 (2019). https://doi.org/10.1109/ICCV.2019.00939 5, 9, 11
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: Nuscenes: A multimodal dataset for autonomous driving. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 11618–11628 (2020). https://doi.org/10.1109/CVPR42600.2020.01164 3, 10
4. Cheng, R., Razani, R., Taghavi, E., Li, E., Liu, B.: (AF)2-S3Net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 12547–12556 (2021) 5
5. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013). https://doi.org/10.1177/0278364913491297 3
6. Hou, Y., Zhu, X., Ma, Y., Loy, C.C., Li, Y.: Point-to-Voxel Knowledge Distillation for LiDAR Semantic Segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. (2022) 5
7. Kong, L., Liu, Y., Chen, R., Ma, Y., Zhu, X., Li, Y., Hou, Y., Qiao, Y., Liu, Z.: Rethinking Range View Representation for LiDAR Segmentation. In: Int. Conf. Comput. Vis. (2023) 5
8. Li, J., Dai, H., Ding, Y.: Self-distillation for robust LiDAR semantic segmentation in autonomous driving. In: Eur. Conf. Comput. Vis. pp. 659–676 (2022) 5
9. Liong, V.E., Nguyen, T.N.T., Widjaja, S., Sharma, D., Chong, Z.J.: AMVNet: Assertion-based Multi-View Fusion Network for LiDAR Semantic Segmentation (2020). https://doi.org/10.48550/arXiv.2012.04934 5
10. Liu, Y., Bai, Y., Chen, R., Hou, Y., Shi, B., Li, Y., Kong, L.: PCSeg: An Open Source Point Cloud Segmentation Codebase. https://github.com/PJLab-ADG/PCSeg (2023) 5, 9, 10, 11
11. Liu, Y., Chen, R., Li, X., Kong, L., Yang, Y., Xia, Z., Bai, Y., Zhu, X., Ma, Y., Li, Y., Qiao, Y., Hou, Y.: UniSeg: A Unified Multi-Modal LiDAR Segmentation Network and the OpenPCSeg Codebase. In: Int. Conf. Comput. Vis. pp. 21662–21673 (2023) 5
12. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In: Eur. Conf. Comput. Vis. (2020) 5
13. Xu, J., Zhang, R., Dou, J., Zhu, Y., Sun, J., Pu, S.: RPVNet: A deep and efficient range-point-voxel fusion network for LiDAR point cloud segmentation. In: Int. Conf. Comput. Vis. pp. 16024–16033 (2021) 5
14. Yan, X., Gao, J., Li, J., Zhang, R., Li, Z., Huang, R., Cui, S.: Sparse single sweep LiDAR point cloud segmentation via learning contextual shape priors from scene completion. In: Conf. AAAI Artif. Intell. vol. 35, pp. 3101–3109 (2021) 5
15. Yan, X., Gao, J., Zheng, C., Zheng, C., Zhang, R., Cui, S., Li, Z.: 2DPASS: 2D Priors Assisted Semantic Segmentation on LiDAR Point Clouds. In: Eur. Conf. Comput. Vis. (2022) 5
16. Ye, M., Wan, R., Xu, S., Cao, T., Chen, Q.: Efficient Point Cloud Segmentation with Geometry-Aware Sparse Networks. In: Avidan, S., Brostow, G., Cissé, M.,

Farinella, G.M., Hassner, T. (eds.) Eur. Conf. Comput. Vis. pp. 196–212. Springer (2022). https://doi.org/10.1007/978-3-031-19842-7_12 5

17. Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. (2021) 5