Spherical Mask: Coarse-to-Fine 3D Point Cloud Instance Segmentation with Spherical Representation
– Supplementary Material –

In this supplementary material, we provide following additional information:

- More Discussion on Advantages of Coarse-to-Fine Approach
- Details of Voting, Dynamic Convolution, MLP Heads, Mask Assembly and Training
- Comparison between RID and AABB
- Additional ablation results
- Additional results on S3DIS dataset
- Additional discussion on ScanNet results
- Additional discussion on Soft Margin Loss
- Implementation of $findSector(.)$ function
- Runtime Analysis
- Additional qualitative results

1. More Discussion on Advantages of Coarse-to-Fine Approach

Despite the two limitations, (1) False negative error propagation from the coarse detection and (2) Excessive instance size estimation of AABB, the main advantage of the coarse-to-fine approach comes with the coarse detection that circumvents the class-imbalance problem. For example, in kernel-based and Transformer-based methods, each binary mask has the size of an entire point cloud, and only foreground points of an instance must be classified as 1. As the dimensions of instances are usually much smaller than the dimensions of an entire point cloud, this results in a significant class imbalance problem, where almost all points must be classified as background. In ScanNetV2, the average ratio of foreground points in the point cloud is around 2.3%. Despite the learning-based methods, such as focal loss and dice loss, to deal with the class imbalance, it is arguably always advantageous to provide input with less class imbalance. The coarse-to-fine approaches sidestep this issue using coarse detection, where the prediction is simpler and does not require fine binary classification. The coarse detection just needs to be around the instance. The refinement stage then only needs to consider the points inside the coarse detection without most of the background points further away, significantly alleviating the class-imbalance problem. Based on this advantage of the coarse-to-fine approach, our method RID and RPM tries to cut the error propagation from the coarse detection by using the coarse detection as a soft reference rather than the hard reference in the existing works.

2. Details of Voting

Similar to [12, 14], we use two set-abstract layers with the following structures. The first set-abstraction layer has a ball-query radius of 0.2 and chooses 1024 points with farthest-point sampling (FPS). The maximum number of neighboring points each point considers is set to 32. The grouped features are the point locations($\mathbb{R}^{1024\times 3}$) and the corresponding features($\mathbb{R}^{1024\times D}$) from the encoder. Here, D is 32, as specified in the main paper. The grouped features are then fed into MLP, which consists of 3 blocks, each with Convolution, BatchNorm and ReLU. The output is followed by max-pooling to produce 64 channels of features from the 1024 sub-sampled points. We also add a residual connection using the initial features from the encoder to prevent the gradient vanishing problem.

The second set-abstraction layer has 0.4 and 256 for ball-query radius and number of points for FPS, respectively. Following the same procedure as the set-abstraction layer from the first sampling, the MLP layer outputs 32 channels of features for 256 sub-sampled points, which corresponds to F_2 in the main paper. Here, the output feature dimension of the set-abstraction is reduced to 32 to minimize the memory consumption during training. Empirically, we have not found meaningful differences in performance by setting the feature dimensions higher, such as 64 and 128.

3. Details of Dynamic Convolution

As mentioned in Implementation Detail(Section 4.3) of the main paper, we implement Dynamic Convolution for the per-point offset prediction(RPM) with two layers with 32 hidden dimensions. Similar to [5, 14], an MLP consisting of two blocks with Convolution, BatchNorm, and ReLU as one block generates weights $W \in \mathbb{R}^{K\times H}$ for each instance by taking vote features $F_2 \in \mathbb{R}^{K\times D}$ as input. Here, H is decided by the parameters such as hidden dimension and input feature dimension for dynamic convolution. W is then used to convolve F_2 and the predicted instance center $F_{\text{center}} \in \mathbb{R}^{K\times 3}$, as follows:

$$F_\delta = Conv\left([F_2, F_{\text{center}}]; W\right), \tag{1}$$

where $Conv$ is implemented as two convolutional layers which have hidden feature dimensions of 32 from W. Here, $[,]$ refers to concatenation.

4. Details MLP based Heads

There are four MLPs used as final prediction heads: (1) *CenterHead* (2) *RayHead* (3) Classification and (4) Confidence. All of them take $F_2 \in \mathbb{R}^{K\times D}$ as input.

Each MLP consists of two blocks of Convolution, BatchNorm and ReLU. For the third block, we only use the convolution as an inference output. The momentum is set to 0.1 for all BatchNorm. All MLPs for heads have the same hyper-parameters, except for the output dimensions. For example, classification head outputs scores for each class ($\mathbb{R}^{K\times N_{\text{class}}}$), and confidence head outputs confidence score($\mathbb{R}^{K\times 1}$). Here, N_{class} stands for the number of classes.

5. Details of Mask Assembly and Finding Misclassified Points

As mentioned in Section 3.3.2 of the main paper, the final mask is assembled by comparing if the addition of $p_s^{(i)}$ and $f_\delta^{(i)}$ is smaller than $f_{ray}^{(\tilde{i})}$, as in Equation 12. Here, the addition of $p_s^{(i)}$ and $f_\delta^{(i)}$ is performed by adding $f_\delta^{(i)}$ only to r coordinate of $p_s^{(i)} \in \mathbb{R}^{N_p\times 3}$ among (r, θ, φ) because θ and φ coordinates for each point remain constant within their respective sector as specified in Section 3.3.2(line 259-262). This operation is the same for finding the misclassified points in Equations 4, 5 and 8 in the main paper.

During the inference time, the same operation is applied to generate the binary masks for all K votes, $F_{mask} \in \mathbb{R}^{K\times N_p}$, using K point clouds in spherical coordinates, $P_s \in \mathbb{R}^{K\times N_P\times 3}$, centered to predicted instance centers $F_{\text{center}} \in \mathbb{R}^{K\times 3}$ in cartesian coordinates. The runtime analysis, including the finding of the index $\tilde{i}$ using $findSector(.)$, is illustrated in Section 13 of this supplementary material.

6. Details of Training

In Section 3.4 of the main paper, we illustrate how the loss for each vote is calculated. As mentioned in Implementation Detail(Section 4.3), this loss is calculated for the number of votes mapped to the ground-truth instance with the batch size 10. Therefore, the final loss L_{total} is the averaged value of:

$$L_{\text{total}} = \frac{1}{N_{\text{batch}} N_{\text{inst}} N_{\text{dupl}}} \sum_{b=1}^{N_{\text{batch}}} \sum_{k=1}^{N_{\text{inst}}^{(b)} N_{\text{dupl}}} L^{(b,k)}, \tag{2}$$

where N_{batch}, N_{inst}, and $N_{\text{inst}}^{(b)}$ stand for batch size, a total number of ground-truth instances in the batch, and a number of ground-truth instances in b_{th} point cloud in the batch, respectively. N_{dupl} is the number to duplicate ground-truth instances, and L is the loss for each vote mapped to a ground-truth instance. As mentioned in Section 3.4 and 4.3 of the main paper, N_{batch} and N_{dupl} are set to 10 and 4 respectively.

7. Comparison between RID and AABB

The common characteristic of existing works with the coarse-to-fine approach is based on AABB. As one of the problems mentioned in the main paper, the excessive instance size estimation by AABB brings a burden to the refinement stage, as a bigger size box could include more background points to be refined. Therefore, tighter coarse detection is always advantageous for the refinement step, improving the overall performance of the system. As mentioned in Section (3.3.1) of

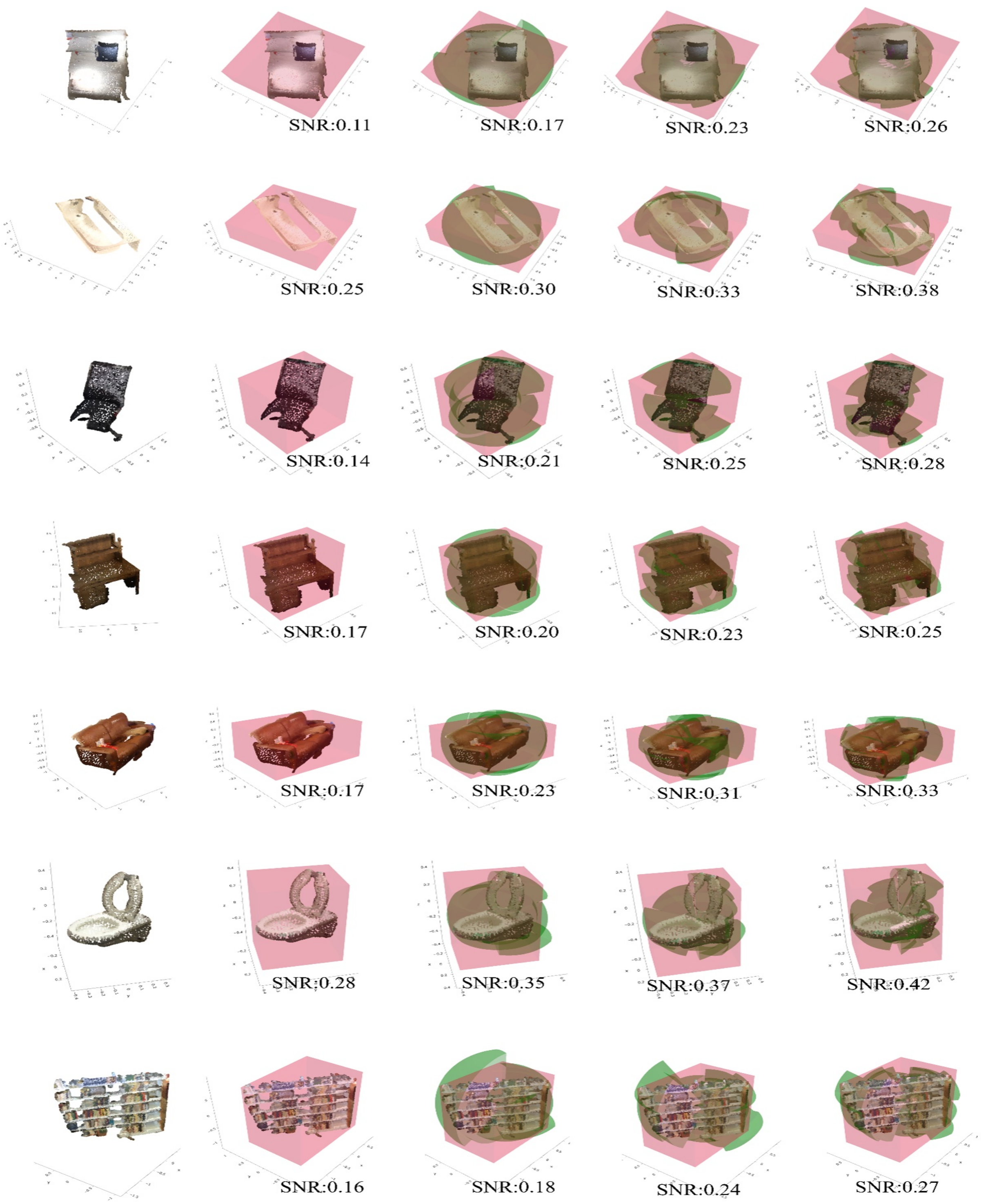


Figure 1. Visually comparing AABB and RID with different configurations of N_θ and N_φ for various instances, such as bed, bathtub, chair, desk, sofa, toilet, and bookshelf. A larger signal-to-noise ratio(SNR) means fewer background points inside the coarse detection, leading to higher precision for the refinement stage and thus improving the overall system performance.

the main paper, RID offers tighter boundaries for instances than AABB. To compare AABB and RID, we measure Signal-to-Noise Ratio for spaces occupied by foreground points compared to the spaces occupied by coarse detections. Specifically, we create a 3D grid, where each grid cell has $0.1m \times 0.1m \times 0.1m$ of space, and count two numbers: (1) N_{coarse}, the number of grid cells inside a coarse detection. (2) $N_{\text{foreground}}$, the number of grid cells occupied by foreground points. We then acquire SNR by dividing $N_{\text{foreground}}$ with N_{coarse}, so that the SNR effectively reflects how tight the boundary is. For example, the higher SNR indicates a tighter boundary, leading to better performance by making the refinement step simpler with fewer background points. Figure 1 shows the comparisons of AABB and RID with different N_θ and N_φ for various instances in ScanNetV2 dataset. As can be seen, depending on the type of instance, the SNR increases from minimum 12% to maximum 50% from AABB when using 3/3 for N_θ/N_φ(the second column), suggesting the advantage of RID over AABB. Increasing N_θ/N_φ from 3/3(column 2) to 5/5(column 3) leads to 22% higher SNR in average. The average SNR improves around 12% by increasing N_θ/N_φ from 5/5(column 3) to 7/7(column 4).

8. Additional Ablation

N_θ/N_φ	mAP	AP_{50}	AP_{25}
1/1	59.6	77.1	85.3
2/2	60.0	77.7	86.1
3/3	60.6	78.4	87.1
4/4	61.2	**80.0**	**89.3**
5/5	**62.3**	79.9	88.2
6/6	60.6	78.5	88.2

Table 1. Impact of N_θ and N_φ on ScanNetV2 validation set

N_θ/N_φ	mAP	AP_{50}	AP_{25}
2/8	61.1	78.2	86.9
4/6	61.9	79.5	88.5
5/5	62.3	80.0	89.3
6/4	**62.5**	**80.5**	**89.4**
8/2	61.7	78.9	88.1

Table 2. Impact of N_θ and N_φ on ScanNetV2 validation set

AABB	RID	L_{mc}	L_{sc}	mAP	AP_{50}	AP_{25}
✓				36.5	57.6	77.4
	✓			51.6	69.4	86.8
	✓	✓		58.5	77.6	87.5
	✓		✓	56.5	77.1	87.1
	✓	✓	✓	**62.3**	**79.9**	**88.2**

Table 3. Impact of each component on ScanNetV2 validation set

8.1. More Results on N_θ and N_φ

In this section, we report additional results of varying N_θ and N_φ. With the same setting as the ablation in the main paper, we fix the backbone and RPM with L_{mc} and L_{sc} and only change N_θ and N_φ. Table 1 shows the results by having the identical N_θ and N_φ and Table 2 shows additional results by varying N_θ and N_φ. As can be seen, decreasing or increasing N_θ and N_φ to an extreme, such as 2/8 and 8/2, results in a performance drop. This is expected, as too few sectors in one axis increases the complexity. Having 6/4 of N_θ and N_φ shows a slight improvement of 0.2 in mAP, suggesting that increasing the number of sectors in the horizontal axis is slightly more helpful than increasing the number of sectors in the vertical axis.

8.2. More Result with AABB

In this section, we report an additional quantitative result of using AABB. For this experiment, we fix the backbone and replace RID and RPM with an MLP that predicts AABB offsets to the mapped ground truth as AABB prediction in [1, 8, 17]. Table 3 shows the performance of only using AABB. Using only RID gains 15.1 higher mAP compared to AABB, suggesting that RID circumvents the excessive size estimation issues of AABB and offers tighter boundaries of instances.

9. Additional results on S3DIS dataset

Table 4 and Table 5 show the quantitative results on S3DIS Area 5 and 6 fold cross-validation from published results. In Area 5, our proposed method outperforms the second-best-performing method with margins of 2.8, 2.39, and 4.09 in mAP, AP_{50}, and mRec_{50}, improving the performance of SOTA 4.8%, 3.4%, and 5.6% respectively. For the 6-fold cross-validation, our proposed method shows the improvement of 3.2, 1.7, and 0.6 in mAP, AP_{50}, and mRec_{50}, pushing the performance of SOTA for 5.3%, 2.4%, and 0.7% respectively. Figure 6 shows the qualitative results on Area 5 of S3DIS.

10. Additional discussion on ScanNet results

Class-level Performance - Why better? In ScanNetV2, we found a pattern of certain instances being located very close($<$ 20 cm) to each other, which can impose challenges in sharply segregating boundaries. For example, the top 3 categories found to have the same type of instances around the most are *Picture*, *Desk*, and *Bookshelf*. On average, a picture hangs with 2.05 pictures on a wall, a desk has 1.07 adjacent desks, usually in an office scene, and a bookshelf also has 1.29 bookshelves very close, usually in a library scene.

To further understand this, we plotted mAP conditioned on the number of close instances belonging to the same categories in Figure 2. The figure shows the drop in performance as the number of adjacent instances increases for all the methods. We think this is why we observe the performance drop in Table 1 of the main paper across all the methods. However, our method

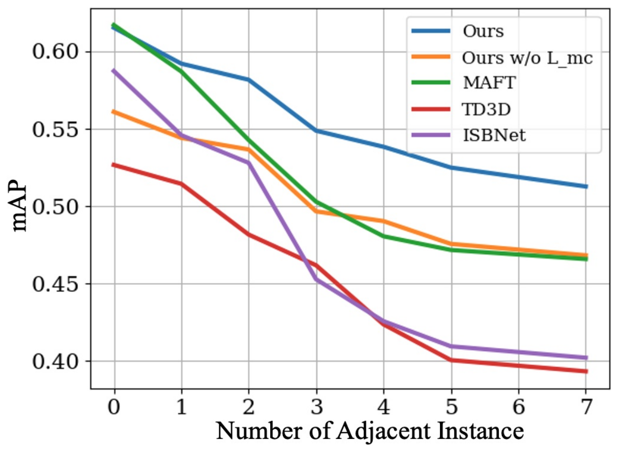


Figure 2. Comparing mAPs of methods for instances closely located to the same class of instances.

performs better as L_{mc} explicitly considers misclassified points around the edges of instances in RPM without considering immense background points as other methods.

Class-level Performance - Why worse? The performance drop of our method was observed in classes *Bed*, *Counter*, and *Cabinet* when compared with transformer-based SOTA methods, MAFT, and QueryFormer. These objects are usually very big and require larger receptive field architectures like transformers. This observation was explored in [10]. We use sparse convolutions [9] in our encoder, and the architectural design with transformers is left for future work.

We expect similar reasoning (e.g., *Wall* has 1.04 adjacent walls) as mentioned above to be valid for S3DIS as well. However, the classes such as *Ceiling* and *Floor* being massive and frequent in all the scans can introduce a huge data imbalance between classes, which was not the case for ScanNet. We are not sure how this would affect the above reasoning, so we left it for future work.

11. Additional Discussion on Soft Margin Loss with Tanh

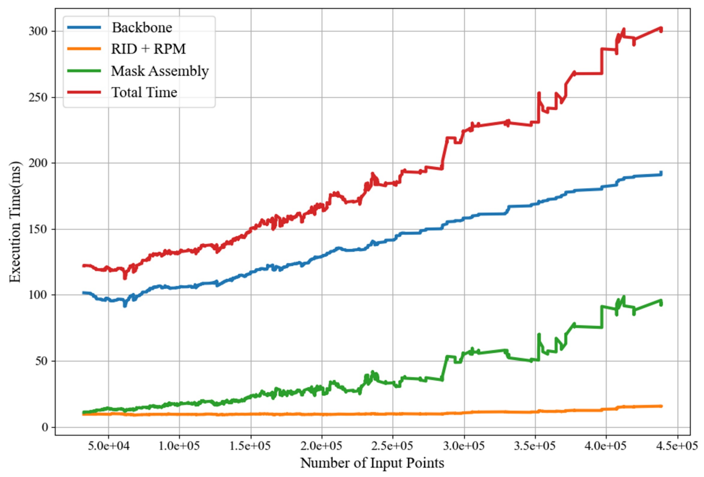
During our initial experiments, we found that using the original Soft Margin loss without tanh was found to diverge approximately around 80 epochs. However, the proposed use of tanh stabilized the training and resulted in overall better results. When tested with the best-performing model(w/o tanh), it resulted in 57.7, 78.0, and 87.9 in mAP, AP_{50}, and AP_{25}, respectively on ScanNet, which is 4.6 lower in mAP compared to our final model.

Method	mAP	AP50	$mPrec_{50}$	$mRec_{50}$
GSPN [18]	-	-	36.0	28.7
PointGroup [7]	-	5.78	61.9	62.1
HAIS [2]	-	-	71.1	65.0
SSTNet [10]	42.7	59.3	65.6	64.2
SoftGroup [15]	51.6	66.1	73.6	66.6
Mask3D [13]	56.5	69.3	68.7	70.7
TD3D [8]	52.1	67.2	-	-
RPGN [3]	-	-	64.0	63.0
PointInst3D [6]	-	-	<u>73.1</u>	65.2
DKNet [16]	-	-	70.8	65.3
ISBNet [14]	56.3	67.5	70.5	72.0
PBNet [19]	53.5	66.4	**74.9**	65.4
QueryFormer [11]	<u>57.7</u>	<u>69.9</u>	70.5	<u>72.2</u>
MAFT [9]	-	69.1	-	-
Ours	**60.5**	**72.3**	71.3	**76.3**

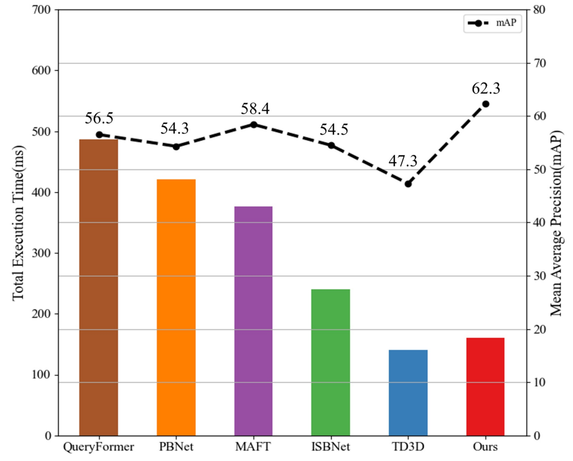
Table 4. Quantitative 3D instance segmentation results on S3DIS Area 5. The best results are in bold, and the second best ones are in <u>underlined</u>.

Method	mAP	AP50	$mPrec_{50}$	$mRec_{50}$
GSPN [18]	-	54.4	38.2	31.2
3D-BoNet [17]	-	-	65.6	47.7
PointGroup [7]	-	64.0	69.6	69.2
OccuSeg [4]	-	-	72.8	60.3
HAIS [2]	-	-	73.2	69.4
SSTNet [10]	54.1	67.8	73.5	73.4
PointInst3D [6]	-	-	76.4	74.0
DKNet [16]	-	-	75.3	71.1
ISBNet [14]	<u>60.8</u>	70.5	77.5	<u>77.1</u>
PBNet [19]	59.5	<u>70.6</u>	**80.1**	72.9
Ours	**64.0**	**72.3**	<u>78.1</u>	**77.7**

Table 5. Quantitative 3D instance segmentation results on S3DIS 6-fold cross-validation. The best results are in bold, and the second best ones are in <u>underlined</u>.



(a)



(b)

Figure 3. Runtime analysis on ScanNetV2 validation set. (a) The runtime of each component in Spherical Mask with respect to the number of input points using a NVIDIA A10 GPU. (b) Comparing the average execution time and the performance in mAP with recent works. On average, TD3D, ISBNet, MAFT, and *Ours* use 3947, 3264, 3723, and 3738 MiB on GPU for ScanNet, respectively.

12. Implementation of $findSector(.)$

The function $findSector(.)$ takes centered points $p_s \in \mathbb{R}^{N_p \times 3}$ in spherical coordinates and returns the index of the sectors that the points belong to as mentioned in Section 3.3.2 (line 296-300) and Equations (4), (5), (6), (8) and (12). Alg. 1 shows the PyTorch implementation of $findSector$. p_s for each vote and batch(during training) are stacked and fed into $findSector$ at once for the efficient computation using GPU.

Algorithm 1 $findSector$ in PyTorch

```
    Input: $p_s$, $N_\theta$, $N_\varphi$
    Output: Indices of sectors that each element of $p_s$ belong to
 1: Function FINDSECTOR($p_s$,$N_\theta$,$N_\varphi$)
 2:     $\mathrm{u}_\theta,\ \mathrm{u}_\varphi = 360/N_\theta,\ 180/N_\varphi$        ▷ Horizontal and vertical unit angles for each spherical sector in degree
 3:     $p_r, p_\theta, p_\varphi = p_s[:,0], p_s[:,1], p_s[:,2]$        ▷ Basis of spherical coordinates
 4:     $\theta_{\text{angles}} =$ torch.arange(0,360, int($\mathrm{u}_\theta$))        ▷ Finding the smallest positive value by subtraction
 5:     $\varphi_{\text{angles}} =$ torch.arange(0,180, int($\mathrm{u}_\varphi$))
 6:     $\text{diff}_\theta = p_\theta$[:,:,None].repeat(1,1,len($\theta_{\text{angles}}$)) - $\theta_{\text{angles}}$[None,None,:]
 7:     $\text{diff}_\varphi = p_\varphi$[:,:,None].repeat(1,1,len($\varphi_{\text{angles}}$)) - $\varphi_{\text{angles}}$[None,None,:]
 8:     $\text{diff}_\theta[\text{diff}_\theta < 0] = \mathrm{u}_\theta + 1$        ▷ Preventing negative values from being the smallest values
 9:     $\text{diff}_\varphi[\text{diff}_\varphi < 0] = \mathrm{u}_\varphi + 1$
10:     $\text{diff}_\theta = \text{diff}_\theta/\mathrm{u}_\theta$
11:     $\text{diff}_\varphi = \text{diff}_\varphi/\mathrm{u}_\varphi$
12:     $\theta_{\text{sector}}$ =torch.argmin($\text{diff}_\theta$,2)        ▷ The index with the smallest positive value is the sector that each point belongs
13:     $\varphi_{\text{sector}}$ =torch.argmin($\text{diff}_\varphi$,2)
14:     return $\varphi_{\text{sector}} * N_\theta + \theta_{\text{sector}}$        ▷ Sector indices
```

13. Runtime Analysis

In this section, we report a runtime analysis of Spherical Mask. Figure 3 (a) shows the execution speed of each module with respect to the number of input points on ScanNetV2. Here, Mask Assembly includes the $findSector$ function and the binary mask generation using the inequality operator, as in Equation (12) of the main paper. As can be seen, the execution time gradually increases as the number of input points grows. The backbone takes the most time among the three due to the Furthest Point Sampling(FPS). The execution time of Mask Assembly also increases more rapidly compared to RID and

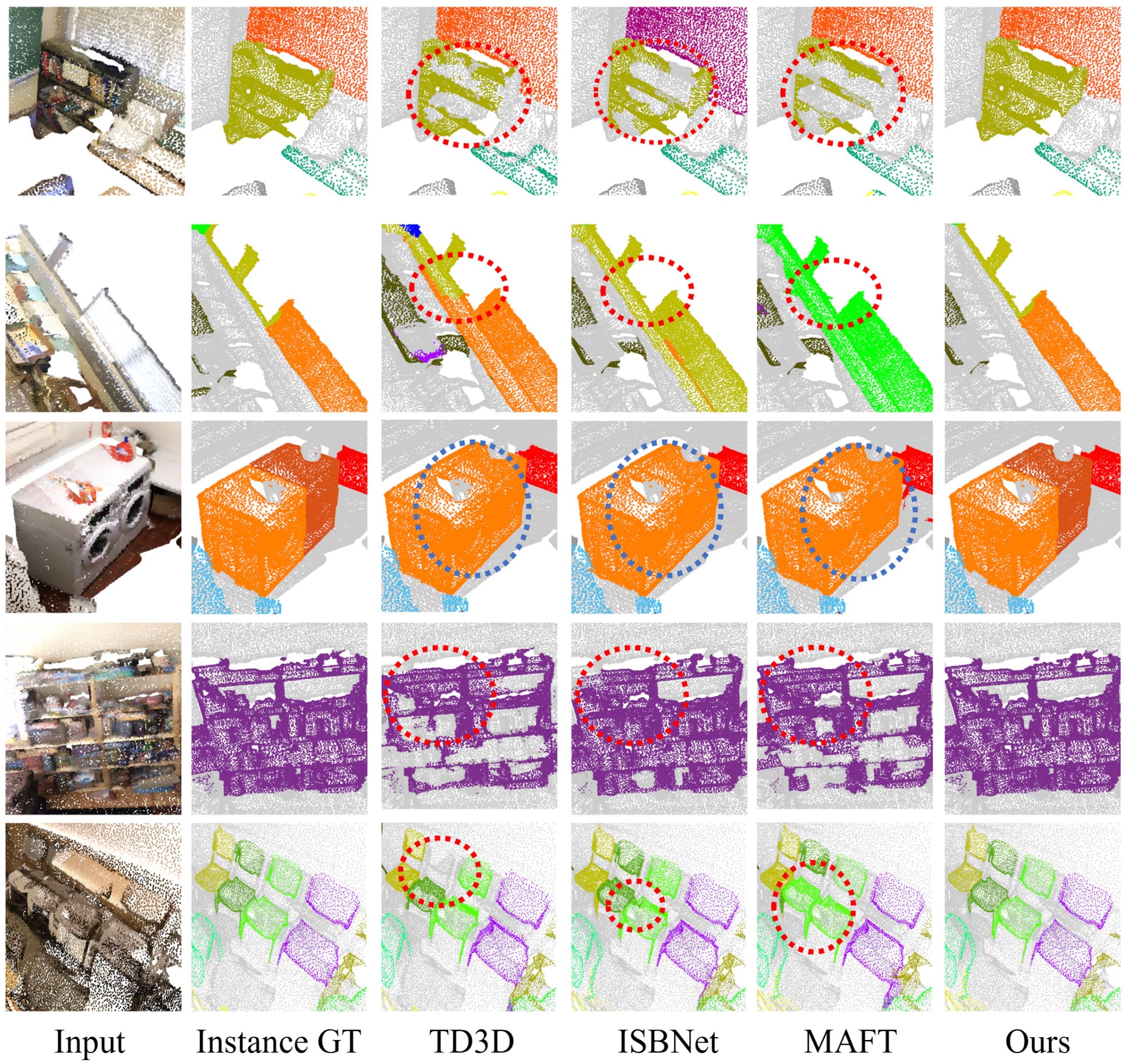


Figure 4. Qualitative results on ScanNetV2 validation set.

RPM modules, as it includes more operations. On average, Spherical Mask segments a point cloud in 167ms.

Figure 3 (b) compares the existing methods and Spherical Mask in terms of execution time and mAP on ScanNetV2 validation set. Spherical Mask achieves the second fastest execution speed while showing the strongest performance in mAP.

14. Additional Qualitative Results

In this section, we show additional qualitative results on ScanNetV2, S3DIS, and STPLS3D datasets. For all the qualitative results of previous works, we only include the results that reproduce the reported results on their papers based on the published source codes for each dataset at the time of submission. We additionally include the results of TD3D[8], which we do not include in the main paper, to highlight only the best-performing methods. The details of each qualitative figure are as follows.

Figure 4 shows the qualitative results of TD3D[8], ISBNet[14], MAFT[9], and our proposed Spherical Mask on ScanNetV2 validation set. For S3DIS, Figure 6 shows the results of TD3D[8], ISBNet[14] and our proposed method on Area 5. Similar to S3DIS, Figure 5 visually compares the results of TD3D[8], ISBNet[14], and our proposed method on STPLS3D.

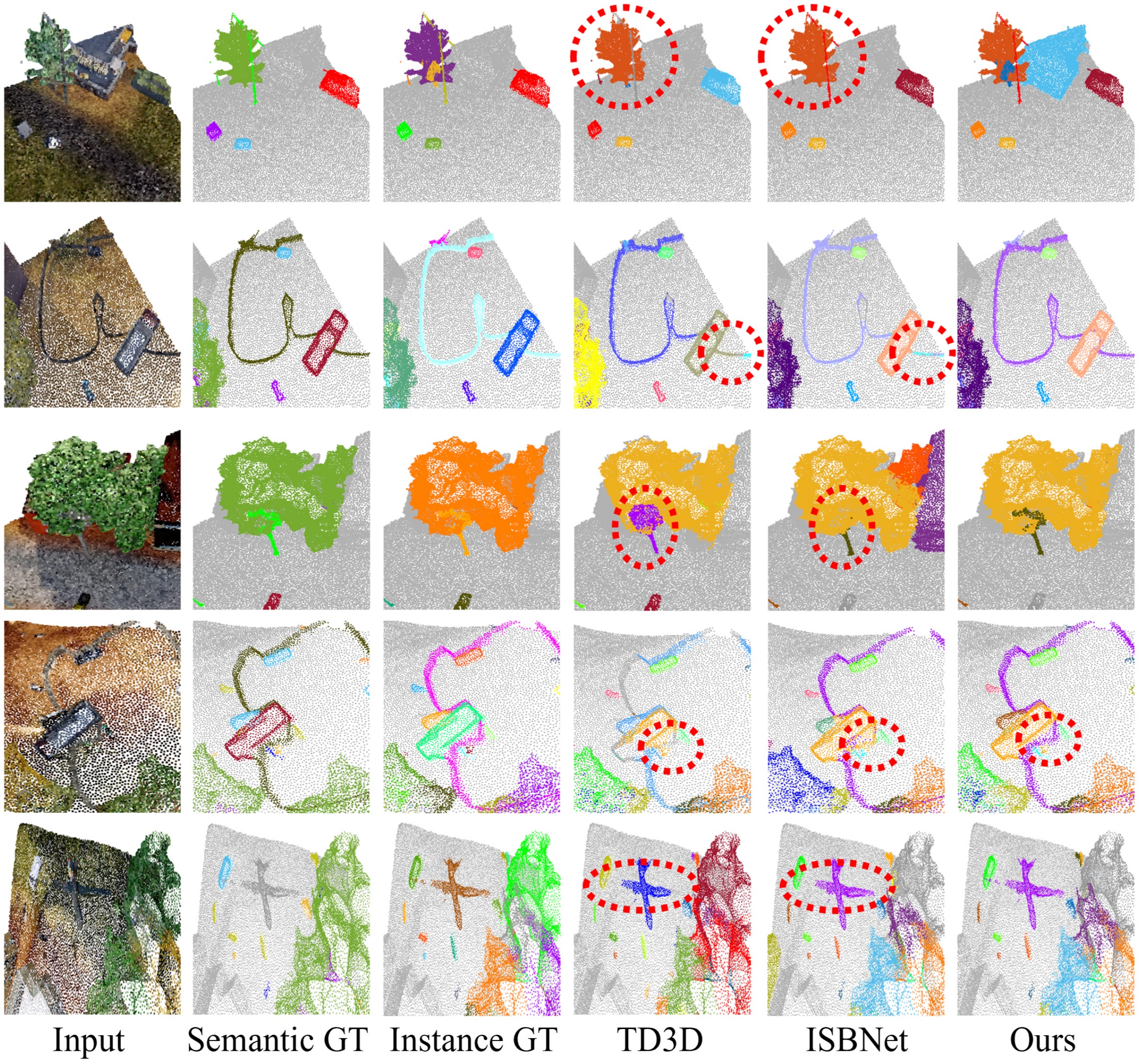


Figure 5. Qualitative results on STPLS3D.

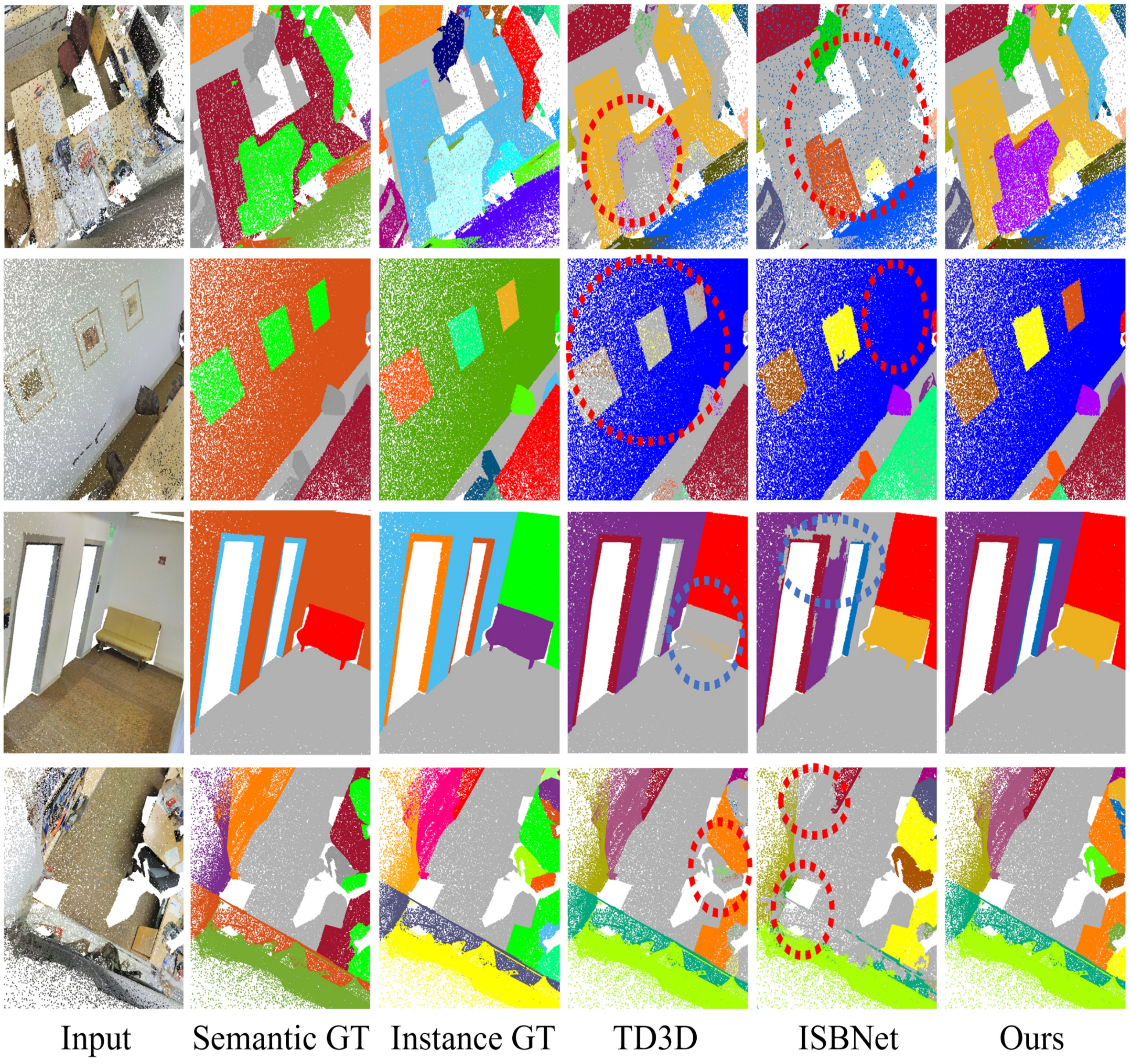


Figure 6. Qualitative results on S3DIS Area 5.

References

[1] BS-EN-ISO. Basic human body measurements for technological design: Worldwide and regional design ranges for use in product standards. *British Standards Institute*, 7250-3, 2015. 4

[2] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical aggregation for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15467–15476, 2021. 5

[3] Shichao Dong, Guosheng Lin, and Tzu-Yi Hung. Learning regional purity for instance segmentation on 3d point clouds. In *European Conference on Computer Vision*, pages 56–72. Springer, 2022. 5

[4] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2940–2949, 2020. 5

[5] Tong He, Chunhua Shen, and Anton Van Den Hengel. Dyco3d: Robust instance segmentation of 3d point clouds through dynamic convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 354–363, 2021. 2

[6] Tong He, Wei Yin, Chunhua Shen, and Anton van den Hengel. Pointinst3d: Segmenting 3d instances by points. In *European Conference on Computer Vision*, pages 286–302. Springer, 2022. 5

[7] 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. 5

[8] Maksim Kolodiazhnyi, Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Top-down beats bottom-up in 3d instance segmentation. *arXiv preprint arXiv:2302.02871*, 2023. 4, 5, 7

[9] Xin Lai, Yuhui Yuan, Ruihang Chu, Yukang Chen, Han Hu, and Jiaya Jia. Mask-attention-free transformer for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3693–3703, 2023. 5, 7

[10] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2783–2792, 2021. 5

[11] Jiahao Lu, Jiacheng Deng, Chuxin Wang, Jianfeng He, and Tianzhu Zhang. Query refinement transformer for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18516–18526, 2023. 5

[12] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. 1

[13] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d for 3d semantic instance segmentation. *arXiv preprint arXiv:2210.03105*, 2022. 5

[14] Khoi Nguyen Tuan Duc Ngo, Binh-Son Hua. Isbnet: a 3d point cloud instance segmentation network with instance-aware sampling and box-aware dynamic convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 5, 7

[15] Thang Vu, Kookhoi Kim, Tung M Luu, Thanh Nguyen, and Chang D Yoo. Softgroup for 3d instance segmentation on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2708–2717, 2022. 5

[16] Yizheng Wu, Min Shi, Shuaiyuan Du, Hao Lu, Zhiguo Cao, and Weicai Zhong. 3d instances as 1d kernels. In *European Conference on Computer Vision*, pages 235–252. Springer, 2022. 5

[17] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. *Advances in neural information processing systems*, 32, 2019. 4, 5

[18] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019. 5

[19] Weiguang Zhao, Yuyao Yan, Chaolong Yang, Jianan Ye, Xi Yang, and Kaizhu Huang. Divide and conquer: 3d point cloud instance segmentation with point-wise binarization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 562–571, 2023. 5