

Divide and Conquer: 3D Point Cloud Instance Segmentation With Point-Wise Binarization

Weiguang Zhao¹, Yuyao Yan², Chaolong Yang¹, Jianan Ye², Xi Yang², Kaizhu Huang^{1*}

¹Duke Kunshan University ²Xi'an Jiaotong-Liverpool University

{weiguang.zhao, chaolong.yang, kaizhu.huang}@dukekunshan.edu.cn

{jianan.ye20}@student.xjtlu.edu.cn {yuyao.yan, xi.yang01}@xjtlu.edu.cn

Abstract

Instance segmentation on point clouds is crucially important for 3D scene understanding. Most SOTAs adopt distance clustering, which is typically effective but does not perform well in segmenting adjacent objects with the same semantic label (especially when they share neighboring points). Due to the uneven distribution of offset points, these existing methods can hardly cluster all instance points. To this end, we design a novel divide-and-conquer strategy named PBNet that binarizes each point and clusters them separately to segment instances. Our binary clustering divides offset instance points into two categories: high and low density points (HPs vs. LPs). Adjacent objects can be clearly separated by removing LPs, and then be completed and refined by assigning LPs via a neighbor voting method. To suppress potential over-segmentation, we propose to construct local scenes with the weight mask for each instance. As a plug-in, the proposed binary clustering can replace the traditional distance clustering and lead to consistent performance gains on many mainstream baselines. A series of experiments on ScanNetV2 and S3DIS datasets indicate the superiority of our model. In particular, PBNet ranks first on the ScanNetV2 official benchmark challenge, achieving the highest mAP. Code will be available publicly at <https://github.com/weiguangzhao/PBNet>.

1. Introduction

In this paper, we consider instance segmentation for 3D point clouds that aims to classify each point of 3D clouds as well as separating objects from each class. While a large body of successful algorithms have been developed for 2D images [28, 13, 2, 29], most of these methods are not par-

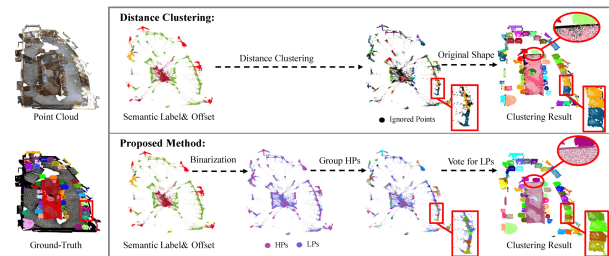


Figure 1. Distance clustering vs. the proposed PBNet. Points dropped in the distance clustering are called ignored points. Clustering based on HPs can better segment adjacent instances (highlighted in rectangular boxes), while the neighbor voting based on LPs can complete the instance (highlighted in ellipses).

ticularly effective for 3D point clouds due to the inherent irregularity and sparsity in 3D data [7, 11, 48].

In 3D point cloud segmentation, PointGroup [21] proposed a distance clustering framework to generate preliminary instance proposals. Although this framework is still being adopted by most SOTAs [14, 27, 3, 41, 45], it may usually have the following shortcomings: (1) distance clustering is limited to segment the adjacent objects with the same semantic label, especially when neighboring points are sticking together; (2) distance clustering only considers points within a distance threshold, which may generate incomplete instances.

To alleviate these drawbacks, we propose a novel proposal generation framework to better segment adjacent objects and complete instances. Inspired by DBSCAN algorithm [10], we divide foreground points into two categories: high and low density points (HPs vs. LPs), depending on the density of each point on the offset branch. As such, neighbor points between adjacent objects are binarized to LPs. Without the interference of neighbor points, grouping HPs can effectively separate adjacent objects. After that, we combine semantic prediction and neighbor voting

*Corresponding author

to assign LPs. In this way, PBNet completely clusters all predicted instance points and works much more reasonable than the traditional distance clustering. The advantages of our methods are illustrated in Fig. 1 where PBNet offers much better segmentation than the distance clustering. Notably, as shown in the experiments, by simply replacing the traditional distance clustering component, the proposed binary clustering strategy could also lead to significant performance gains on other mainstream baselines including PointGroup [21] and HAIS [3].

Furthermore, taking into account the effects of offset error and density threshold, some larger objects such as sofas and tables have a certain probability of being divided into multiple instances. We further propose to search surrounding instances for each instance to construct the corresponding local scene. By designing a concise strategy, we encode each instance in each local scene to generate the corresponding weight mask, thus offering the network with prior knowledge to better focus on the primary instance. Combining the global features and the local features, the final instance mask in the local scene will be predicted. Based on point-wise binarization and local scene, PBNet attains superior performance on both ScanNetV2 [5] and S3DIS [1] dataset. The contributions of our work are as follows:

- By dividing and conquering, we propose a novel clustering method based on binarized points to effectively segment adjacent objects and cluster all predicted instance points. It is appealing that by simply replacing the traditional distance clustering, our proposed binary clustering strategy can also lead to significant performance gains on many mainstream baselines.
- We propose to construct local scenes combined with global feature and weight mask to refine instances, which can suppress over-segmentation and further boost the performance substantially.
- Overall, we design a novel end-to-end 3D instance segmentation framework which significantly outperforms current SOTAs for 3D instance segmentation: our model ranks the first on mAP metric of the ScanNetV2 official benchmark challenge.

2. Related Work

2.1. Deep Learning on 3D Point Cloud

PointNet [34] pioneered the application of deep learning techniques to point cloud processing. Since then, deep learning has advanced significantly in a variety of 3D tasks, including 3D target detection, 3D semantic segmentation, 3D instance segmentation, 3D shape classification, and 3D reconstruction. Existing methods can be roughly divided into three categories: point-based, voxel-based, and multiview-based methods [11]. Point-based

methods [35, 43, 44, 36, 42] operate directly on the original points of the 3D point clouds without projection and volumetric operations. Volumetric-based methods [31, 37] convert the 3D point clouds into a 3D volume representation and then extract features using a sparse convolution network. Multiview-based methods [39, 6, 23, 17, 19] project 3D point clouds to multiple 2D planes in different directions to form multiple 2D images and then extract the features of these 2D images for feature fusion or analysis.

2.2. Instance Segmentation for 3D Point Cloud

Instance segmentation needs to separate each individual in the 3D scene, while semantic segmentation only needs to segment objects in the same category. The methods of 3D instance segmentation can be roughly divided into two categories: proposal-based and clustering-based. Proposal-based methods [16, 33, 46] are top-down approaches, which regress 3D bounding boxes to segment instances. GSPN [49] is an earlier proposal-based network. It abandons the traditional anchor-based method, and advocates learning what the target looks like before choosing the proposal region. 3D-BoNet [47] develops a novel multi-criteria loss to constrain bounding boxes. 3D-MPA [9] combines a sparse convolutional network with a graph convolutional network to refine proposals.

Clustering-based methods dominate the benchmark challenge for this task, especially on ScanNetV2 [5] dataset. These methods predict point-wise distance offsets from instance center points and group points on this branch. PointGroup [21] takes point offset and distance clustering as the core of the algorithm. Many subsequent methods [14, 27, 3, 41, 45] are all based on the distance clustering algorithm. HAIS [3] aggregates instances according to the number of points and designs the mask loss to refine instances. SSTNet [26] utilizes superpoints to build a tree and aggregate the tree nodes to generate instances. SoftGroup [41] adopts soft semantic predictions to reduce the impact of semantic error. DKNet [45] utilizes MLP to predict point-wise confidence based on distance clustering. DKNet can improve the segmentation of adjacent objects, but it also introduces confidence error and still ignores some foreground points. In contrast, PBNet binarizing points by point-wise density is more concise and effective.

Most SOTAs adopt merging-based method to suppress over-segmentation. HAIS [3] makes rules based on the average number of points contained in each category and average sizes of that to aggregate instances. MaskGroup [53] sets an increasing distance threshold to merge instances iteratively. DKNet [45] learns the direct fusion relationship of each instance through the network to form a merging map, and utilize greedy algorithm to merge these instances. These methods are prone to under-segmentation while suppressing over-segmentation, and the instance edge

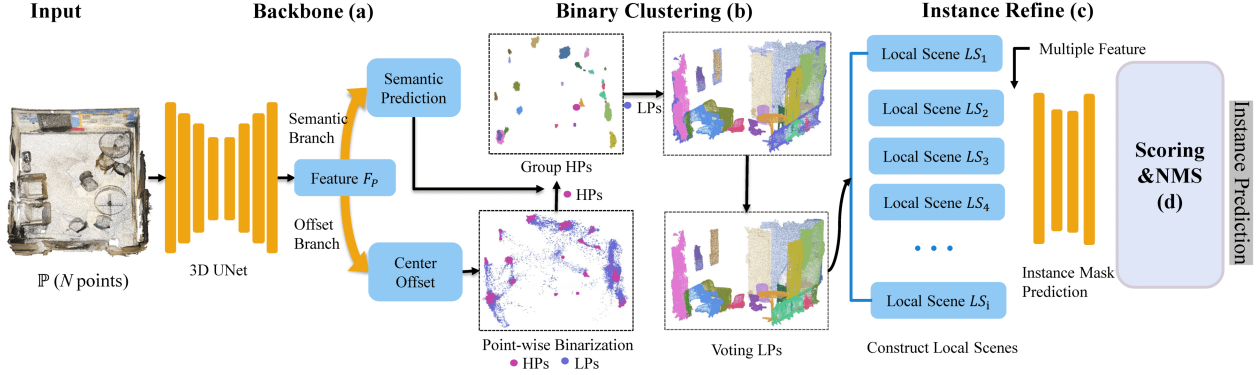


Figure 2. Network Architecture.

cannot be refined by directly merging instances. Inspired by Knet [52] and Mask-RCNN [13], we construct the local scenes for each instance and generate the weight mask for local scene to implied different instances. Different from the existing SOTAs, our methods is soft and combine global and local feature to refine instances.

The proposed PBNet can be deemed as one voxel-based and clustering-based method. Different from the existing SOTAs, we divide the points into two categories in the offset branch and process them separately. As shown in Fig. 1, the adjacent objects with the same semantic label can be separated based on HPs. Meanwhile, handling LPs can better complete instances. Then PBNet construct the local scenes for each instance to suppress over-segmentation softly. PBNet demonstrate its superiority to the other SOTAs.

3. Our Method

3.1. Architecture Overview

The overall network architecture of PBNet is depicted in Fig. 2. It consists of four main parts: Backbone (a), Binary Clustering (b), Instance Refine (c), and Scoring & NMS (d). First, traditional normal vectors are calculated on the faces¹ of the point cloud. We then feed the network with xyz , rgb , and normal vector features. 3D UNet [25, 38] and two FC layers are combined as a backbone to predict the point-wise semantic label and distance offset from the instance center. Then we calculate the density of each point on offset branch, and classify these points into two categories (HPs/LPs) by setting the density threshold θ_d . Combined with semantic prediction, HPs will be grouped to form preliminary instances. We convert the grouped HPs and ungrouped LPs in the offset coordinate system back to the original coordinate system. Furthermore, LPs will be assigned to the instances by neighbor voting algorithm. In order to suppress over-segmentation, we search surround-

¹The face is one base attribute of 3D items, often adopted in previous 3D instance segmentation works [26, 45].

ing instances for each instance to construct the corresponding local scene. The number of local scenes is the same as that of instances. Integrated with feature F_p , local scenes are utilized to refine each instance mask. Finally, we adopt ScoreNet [21] and Non-maximum suppression (NMS) to achieve the instance prediction.

3.2. Backbone

Same as many SOTAs [21, 3, 26, 41, 45], 3D UNet [25, 38] is used to extract features of each point in our implementation. The point cloud is converted into a voxel form before it is fed into 3D UNet. When the features are extracted by 3D UNet, the voxel form point cloud is then converted to the point format according to the index. The semantic and offset branches composed of multi-layer perceptrons (MLP) are utilized to predict semantic label and offset for each point. At this stage, the background points (wall, floor) in the offset branch are removed according to the prediction of semantic results.

Semantic Branch. The features of each point are fed into a 3-layer MLP to predict its semantic score of each class. The semantic scores are recorded as $\mathbf{S} \in [0, 1]^{N \times M}$, where N and M are the number of point and class, respectively. The class with the highest score will be the semantic label for points. We utilize the cross-entropy loss L_{sem} to regularize the semantic results.

Offset Branch. Similar to the semantic branch, we adopt a 3-layer MLP to predict offset vector $\mathbf{o}_i = \{o_x^i, o_y^i, o_z^i\}$ of each point, where $i \in \{1, \dots, N\}$. Since $\hat{\mathbf{c}}_i = \{\hat{c}_x^i, \hat{c}_y^i, \hat{c}_z^i\}$ is the centroid of the instance that point i belongs to, L_1 regression loss is taken to constrain points with the same instance labels to learn offsets [33, 21]. The calculation formula of L_1 regression loss is as follows:

$$L_{o_dist} = \frac{1}{\sum_{i=1}^N} \sum_{i=1}^N \|\mathbf{o}_i - (\hat{\mathbf{c}}_i - \mathbf{p}_i)\|, \quad (1)$$

where $\mathbf{p}_i = \{p_x^i, p_y^i, p_z^i\}$ describes the 3D coordinate of

point i in the original point clouds. The calculation formula of $\hat{\mathbf{c}}_i$ is as follows:

$$\hat{\mathbf{c}}_i = \frac{1}{N_{map(i)}^I} \sum_{j \in I_{map(i)}} \mathbf{p}_j, \quad (2)$$

where $map(i)$ maps point i to the index of its corresponding ground-truth instance. $N_{map(i)}^I$ is the number of points in instance $I_{map(i)}$. In order to regress the precise offsets, we follow [24, 21] to adopt direction loss L_{o-dir} :

$$L_{o-dir} = -\frac{1}{\sum_{i=1}^N} \sum_{i=1}^N \frac{\Delta \mathbf{p}_i}{\|\Delta \mathbf{p}_i\|_2} \cdot \frac{\hat{\mathbf{c}}_i - \mathbf{p}_i}{\|\hat{\mathbf{c}}_i - \mathbf{p}_i\|_2}. \quad (3)$$

This loss reinforces each point to move towards the correct direction by constraining the angle between the predicted offset vector and the ground-truth vector.

3.3. Binary Clustering

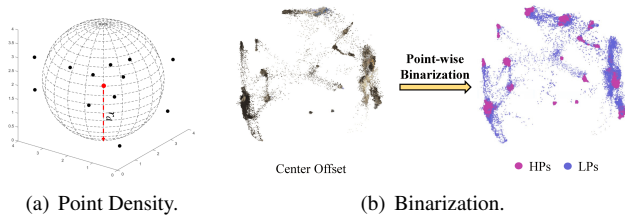


Figure 3. Point-wise Binarization.

3.3.1 Point-wise Binarization

We deploy point-wise density to conduct binarization. Its calculation process is shown in Fig. 3(a). For each point, we draw a sphere of radius r_d . The number of points in the ball is used to reflect the density. Exactly, the density of the point p_i can be defined as the quantity of points within a sphere centred on the point p_i with radius r_d . For example, in Fig. 3(a), the value reflecting the density of the red points is given as 7. According to this method, we calculate the density value of every instance point on the offset branch. With the density of points, these points can be divided into two categories: HPs and LPs. If the densities of points are greater than the threshold θ_d , these points are classified as HPs, while the remaining points will be classified as LPs.

3.3.2 Grouping HPs

We utilize semantic prediction and develop one modified variant of DBSCAN [10] to group HPs. Specifically, we extend the traditional unsupervised DBSCAN to a weakly-supervised version by feeding semantic labels to guide clustering. With the weakly supervised information, PBNNet can lead to much accurate clustering. Meanwhile, considering

that the number of HPs is often huge, we further take binary search, and CUDA to speed up the clustering process. As a result, the time complexity can be substantially reduced from $\mathcal{O}(N_h^2)$ to $\mathcal{O}(N_h \log(N_h)/(K_c * T))$, where N_h is the number of HPs, K_c is semantic category number, T is thread number of CUDA. Overall, our HP grouping method is both accurate and fast.

3.3.3 Voting LPs

LPs are also critical to instance segmentation, which can lead to more complete and refined instances. We combine LPs and grouped HPs, and change them back to the original shape according to the index. As shown in Fig. 4, we find that all LPs are almost edge points. To this end, we develop neighbor voting [51] to determine which instance these LPs belong to. Different from the previous algorithm, we introduce the mean size of each category r_m (which can be estimated from training data) and predict the semantic label to assist judgment. For each noise point, we select the HPs which share the same semantic label as LPs in the r_m range. Then we count which instance these HPs belong to, and take the instance that contains the most HPs as the attribution of this noise point. There might be also an extreme case, i.e., there are no HPs with the same semantics around the noise. In this case, we put aside the semantic label and directly exploit the nearest neighbor voting method [40] to determine the attribution of the noise point. We repeat this operation until each noise point is classified. The time complexity of voting LPs is $\mathcal{O}(N_h * N_l/(K_c * T))$, where N_l is the number of LPs.

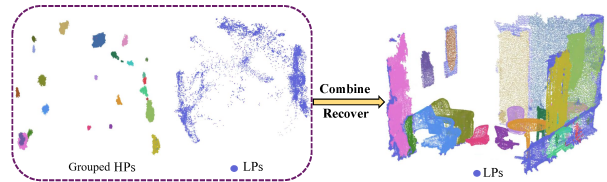


Figure 4. Combination and Recovery

3.4. Instance Refining

3.4.1 Local Scene Construction

Some objects with larger sizes and asymmetric shapes are easily over-segmented, such as the class of sofa as seen in Fig. 5(a). In the 2D domain, KNet [52] proposes that an object corresponds to an image mask. Due to the sheer size of the 3D scene, this method is difficult to be applied directly. Inspired by KNet, we propose to search the nearest K instances (secondary instances) for each instance (primary instance). One local scene corresponds to one primary instance. To differentiate the primary and secondary instances in each scene, we define a concise formula to gen-

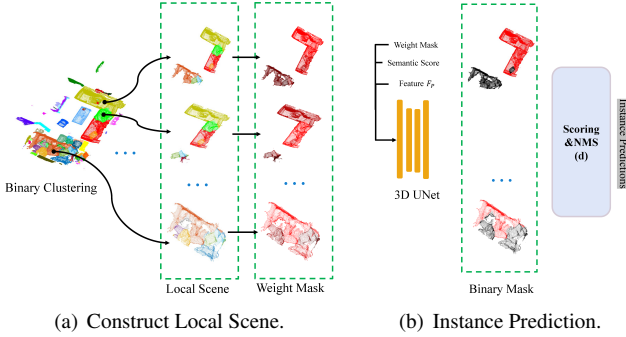


Figure 5. Instance Prediction.

erate the weight mask. The calculation formula of weight masks \mathbb{W} is as follows:

$$\mathbb{W}_i = (\text{Min}(K, K_s - 1) - i) / (\text{Min}(K, K_s - 1)), \quad (4)$$

where \mathbb{W}_i is the weight mask of the i th closest secondary instance to the primary instance. $\text{Min}(\cdot)$ is the function that takes the minimum value. K_s is the number of instances contained in the current semantic scene.

3.4.2 Instance Mask Prediction

Each local scene is fed to the 3D UNet to refine the mask of primary instances. Weight mask, semantic score and feature F_p are concatenated to be the input feature. Weight masks can provide prior knowledge to direct the network to focus more on the primary instance. The semantic score has been verified in the literature as one effective idea for instance segmentation [26]. Feature F_p is extracted by the backbone while the whole 3D scene is given as the input. The ground-truth mask of the local scene is defined as a binary mask, where ground-truth primary instance mask is 1 and others are 0. Then we adopt the binary cross-entropy to calculate the mask prediction loss L_{s_mask} :

$$L_{s_mask} = - \frac{1}{\sum_{i=1}^{\hat{N}^l} \hat{N}_i} \cdot \sum_{i=1}^{\hat{N}^l} \sum_{j=1}^{\hat{N}_i} (y_j^i \log(\hat{y}_j^i) + (1 - y_j^i) \log(1 - \hat{y}_j^i)), \quad (5)$$

where \hat{N}^l is the number of local scenes. \hat{N}_i denotes the number of points within the i th local scene and y_j^i describes the ground-truth score of the j th points of the i th local scene. Moreover dice loss L_{dice} is also applied, following DKNNet [45]. The calculation formula of L_{dice} is as follows:

$$L_{dice} = \frac{1}{\hat{N}^l} \sum_{i=1}^{\hat{N}^l} \left(\left(1 - 2 \frac{M_i^p \cdot M_i^g}{|M_i^p| + |M_i^g|} \right) \right), \quad (6)$$

where M_i^p and M_i^g are predicted mask and ground-truth masks for the i th local scene, respectively.

3.5. Scoring & NMS

Due to the over-segmentation, the primary instances may correspond to the same ground-truth instance after refinement. NMS is introduced to filter refined primary instances. Following [3], we adopt ScoreNet [18, 20, 25] to evaluate all instances and score them. ScoreNet consists of a lightweight 3D UNet and fully connected layers. For instance scores, we exploit a soft label SC to supervise the predicted instance score \hat{SC} . Same as [21], the binary cross-entropy is used to calculate the instance score loss:

$$L_{s_ins} = - \frac{1}{N_{ins}} \sum_{i=1}^{N_{ins}} (SC_i \log(\hat{SC}_i) + (1 - SC_i) \log(1 - \hat{SC}_i)), \quad (7)$$

where N_{ins} is the number of the predicted instances. We take the score as the confidence for each instance and utilize NMS to get the final instance result.

3.6. Multi-Task Training

Our model can be trained in an end-to-end manner, even if it has multiple different tasks. The total loss of our network can be written as:

$$L_{all} = L_{sem} + L_{o_dist} + L_{o_dir} + L_{s_mask} + L_{dice} + L_{s_ins} \quad (8)$$

All loss weights are set to 1.0, which works well as empirically verified in the experiments. Since L_{s_mask} , L_{dice} and L_{s_ins} are affected by semantic and offset results, we do not add these losses until 128 epochs.

4. Experiments

4.1. Experiment Setting

Datasets. ScanNetV2 [5], one most challenging 3D dataset, includes 1,201 training samples, 312 validation samples, and 100 test samples where 20 semantic classes and 18 instance classes are labeled. Following most similar work in instance segmentation, classes including wall and floor are removed. The color for instance segmentation is random because the number of instances for each sample is flexible. We compare the results on the validation as well as test set, which come from the official evaluation website.

S3DIS [1] dataset includes 271 scenes within 6 areas. In these scenes, a total of 13 semantic classes are labeled. We utilize all the classes for instance evaluation and report the results on area 5, while the remaining areas are used for training. As points of the S3DIS scene is much more than ScanNetV2, we randomly sample points before each cropping by following the previous methods [21, 26].

| Method | mAp | bathroom | bed | bookshelf | cabinet | chair | counter | curtain | desk | door | furniture | picture | refrigerator | s.curtain | sink | sofa | table | toilet | window |
|------------------|-------------|-------------|-------------|-------------|---------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|-------------|
| SGPN [42] | 4.9 | 2.3 | 13.4 | 3.1 | 1.3 | 14.4 | 0.6 | 0.8 | 0.0 | 2.8 | 1.7 | 0.3 | 0.9 | 0.0 | 2.1 | 12.2 | 9.5 | 17.5 | 5.4 |
| GSPN [49] | 15.8 | 35.6 | 17.3 | 11.3 | 14.0 | 35.9 | 1.2 | 2.3 | 3.9 | 13.4 | 12.3 | 0.8 | 8.9 | 14.9 | 11.7 | 22.1 | 12.8 | 56.3 | 9.4 |
| 3D-Bonet [47] | 25.3 | 51.9 | 32.4 | 25.1 | 13.7 | 34.5 | 3.1 | 41.9 | 6.9 | 16.2 | 13.1 | 5.2 | 20.2 | 33.8 | 14.7 | 30.1 | 30.3 | 65.1 | 17.8 |
| 3D-MPA [9] | 35.5 | 45.7 | 48.4 | 29.9 | 27.7 | 59.1 | 4.7 | 33.2 | 21.2 | 21.7 | 27.8 | 19.3 | 41.3 | 41.0 | 19.5 | 57.4 | 35.2 | 84.9 | 21.3 |
| PointGroup [21] | 40.7 | 63.9 | 49.6 | 41.5 | 24.3 | 64.5 | 2.1 | 57.0 | 11.4 | 21.1 | 35.9 | 21.7 | 42.8 | 66.0 | 25.6 | 56.2 | 34.1 | 86.0 | 29.1 |
| OCCuSeg [12] | 48.6 | 80.2 | 53.6 | 42.8 | 36.9 | 70.2 | 20.5 | 33.1 | 30.1 | 37.9 | 47.4 | 32.7 | 43.7 | 86.2 | 48.5 | 60.1 | 39.4 | 84.6 | 27.3 |
| Dyco3d [14] | 39.5 | 64.2 | 51.8 | 44.7 | 25.9 | 66.6 | 5.0 | 25.1 | 16.6 | 23.1 | 36.2 | 23.2 | 33.1 | 53.5 | 22.9 | 58.7 | 43.8 | 85.0 | 31.7 |
| PE [50] | 39.6 | 66.7 | 46.7 | 44.6 | 24.3 | 62.4 | 2.2 | 57.7 | 10.6 | 21.9 | 34.0 | 23.9 | 48.7 | 47.5 | 22.5 | 54.1 | 35.0 | 81.8 | 27.3 |
| SSTNet [26] | 50.6 | 73.8 | 54.9 | 49.7 | 31.6 | 69.3 | 17.8 | 37.7 | 19.8 | 33.0 | 46.3 | 57.6 | 51.5 | 85.7 | 49.4 | 63.7 | 45.7 | 94.3 | 29.0 |
| HAIS [3] | 45.7 | 70.4 | 56.1 | 45.7 | 36.4 | 67.3 | 4.6 | 54.7 | 19.4 | 30.8 | 42.6 | 28.8 | 45.4 | 71.1 | 26.2 | 56.3 | 43.4 | 88.9 | 34.4 |
| MaskGroup [53] | 43.4 | 77.8 | 51.6 | 47.1 | 33.0 | 65.8 | 2.9 | 52.6 | 24.9 | 25.6 | 40.0 | 30.9 | 38.4 | 29.6 | 36.8 | 57.5 | 42.5 | 87.7 | 36.2 |
| SoftGroup [41] | 50.4 | 66.7 | 57.9 | 37.2 | 38.1 | 69.4 | 7.2 | 67.7 | 30.3 | 38.7 | 53.1 | 31.9 | 58.2 | 75.4 | 31.8 | 64.3 | 49.2 | 90.7 | 38.8 |
| RPGN [8] | 42.8 | 63.0 | 50.8 | 36.7 | 24.9 | 65.8 | 1.6 | 67.3 | 13.1 | 23.4 | 38.3 | 27.0 | 43.4 | 74.8 | 27.4 | 60.9 | 40.6 | 84.2 | 26.7 |
| PointInst3D [15] | 43.8 | 81.5 | 50.7 | 33.8 | 35.5 | 70.3 | 8.9 | 39.0 | 20.8 | 31.3 | 37.3 | 28.8 | 40.1 | 66.6 | 24.2 | 55.3 | 44.2 | 91.3 | 29.3 |
| DKNet [45] | 53.2 | 81.5 | 62.4 | 51.7 | 37.7 | 74.9 | 10.7 | 50.9 | 30.4 | 43.7 | 47.5 | 58.1 | 53.9 | 77.5 | 33.9 | 64.0 | 50.6 | 90.1 | 38.5 |
| Ours | 57.3 | 92.6 | 57.5 | 61.9 | 47.2 | 73.6 | 23.9 | 48.7 | 38.3 | 45.9 | 50.6 | 53.3 | 58.5 | 76.7 | 40.4 | 71.7 | 55.9 | 96.9 | 38.1 |

Table 1. mAP on ScanNetV2 Hidden Test Set.

Evaluation Metric. Following the ScanNetV2 official benchmark challenge, we report the mean average precision AP (mAP) at overlap 0.25 (AP_{25}), overlap 0.5 (AP_{50}), and over overlaps in the range [0.5:0.95:0.05] (AP) for ScanNetV2 dataset. Moreover, SoftGroup [41] and DKNet [45] also report the Box AP_{50} and AP_{25} results, which are commonly used in 3D object detection. For fair comparison, we follow them to report these metrics. Finally, we take the performance of mAP , AP_{50} , mean precision ($mPrec_{50}$) and mean recall ($mRec_{50}$) as the metric for S3DIS dataset, same as SOTAs.

Implementation Details. We conduct training with two RTX3090 cards for 512 epochs. The batch size of training is set to 4. We adopt Adam [22] as the optimizer. The initial learning rate is set to 0.001 which decays with the cosine anneal schedule [30]. We set the voxel size to 0.02 by following pioneer methods [21, 3, 14]. For hyperparameters of density clustering, we tune r_d , θ_d empirically as 0.04, 30 respectively. The secondary instance number K is empirically set to 7. The 3D UNet of backbone is MinkowskiNet34C [4], while the 3D UNet in mask prediction and ScoreNet are both MinkowskiNet14A [4]. Data enhancements such as rotation, elastic distortion [38], color jittering, mixing [32] are adopted following the previous work [21, 32, 41]. Following SOTAs [45, 12, 26], a graph-based post-processing is utilized to smooth labels.

4.2. Comparison to SOTAs

Result on ScanNetV2. Tab. 1 shows the mAP results of PBNet and SOTAs on the hidden test set of ScanNetV2 benchmark. PBNet ranks the first on mAP metric of ScanNetV2 3D instance segmentation challenge, on January 2023. Specifically, PBNet achieves the best performance in 10 out of 18 classes. Following previous work [45, 41], we also report the mask segmentation and the detection box

results on ScanNetV2 validation set in Tab. 2. For the mask segmentation, PBNet again shows relative 5.4%, and 4.3% improvements on mAP and AP_{50} respectively. On the other hand, our method also gets the best results on Box AP_{50} for the detection task.

Result on S3DIS. Following SOTAs, we report the results of Area 5 and 6-fold cross-validation on the S3DIS dataset in Tab. 3. For the 6-fold cross-validation, we report the average results. As observed, our approach is still ahead of the other methods on the major metrics mAP and AP_{50} . When evaluated on Area 5, our method shows the best result on three over four metrics, i.e., mAP , AP_{50} , and $mPre_{50}$. As for the metric of $mRec_{50}$, our method is inferior to Softgroup [41] but still competitive, which ranks the second among all the methods. In the results of 6-fold cross-validation, our model attains the best mAP and AP_{50} , and it ranks the second and third respectively on $mRec_{50}$ and $mPre_{50}$. In short, our model demonstrates the overall best performance on both Area 5 and 6-fold cross-validation.

Qualitative Comparison. We also provide qualitative comparisons based on ScanNetV2 (see Fig. 6). Clearly, our method exhibits visually better performance than the other SOTAs. More visualization results on both ScanNetV2 and S3DIS are provided in the supplementary.

| | Segmentation | | | Detection | |
|------------------|--------------|-------------|-------------|---------------|---------------|
| | mAP | AP_{50} | AP_{25} | Box AP_{50} | Box AP_{25} |
| VoteNet [33] | - | - | - | 33.5 | 58.6 |
| 3D-MPA [9] | 35.3 | 59.1 | 72.4 | 49.2 | 64.2 |
| PointGroup [21] | 34.8 | 56.9 | 71.3 | 48.9 | 61.5 |
| Dyco3D [14] | 40.6 | 61.0 | - | 39.5 | 64.1 |
| HAIS [3] | 43.5 | 64.1 | 75.6 | 53.1 | 64.3 |
| SSTNet [26] | 50.0 | 64.7 | 73.9 | 52.7 | 62.5 |
| SoftGroup [41] | 46.0 | 67.6 | 78.9 | 59.4 | 71.6 |
| PointInst3D [15] | 45.6 | 63.7 | - | 51.0 | - |
| DKNet [45] | 51.5 | 67.0 | 77.0 | 59.0 | 67.4 |
| Ours | 54.3 | 70.5 | 78.9 | 60.1 | 69.3 |

Table 2. Quantitative Comparison on ScanNetV2 Validation Set.

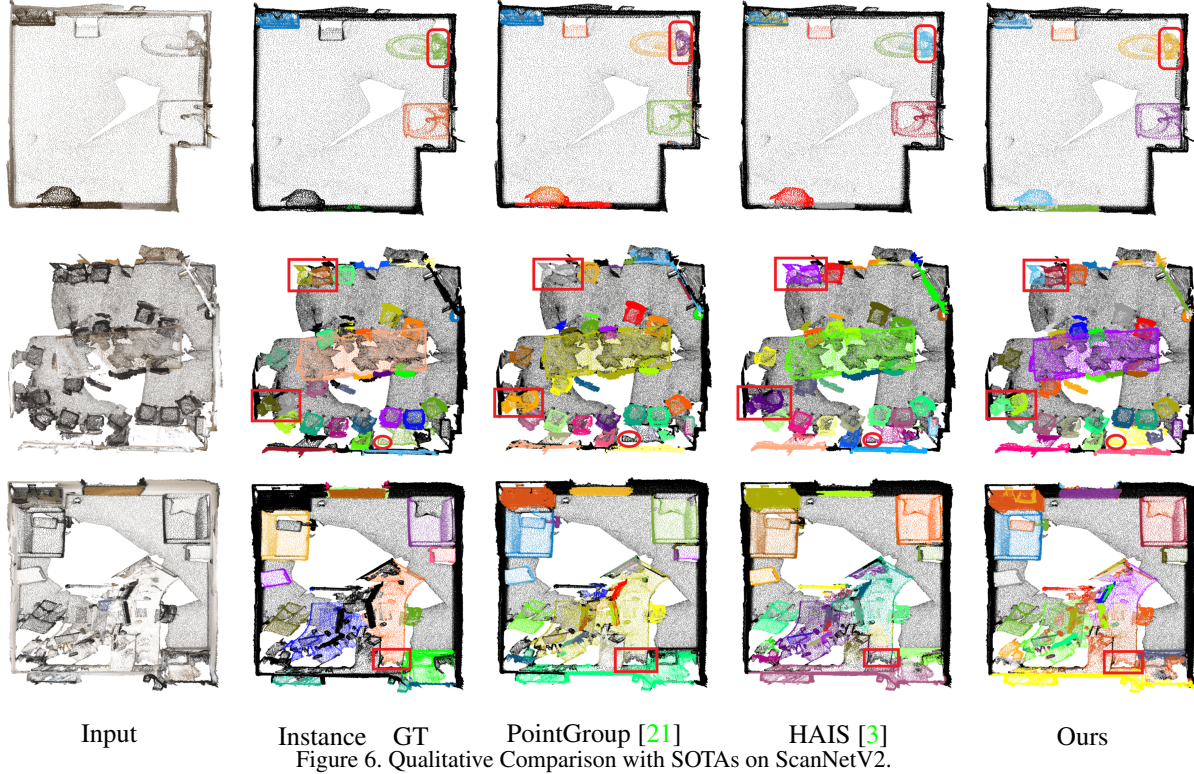


Figure 6. Qualitative Comparison with SOTAs on ScanNetV2.

| | mAP | AP_{50} | $mPrec_{50}$ | $mRec_{50}$ |
|-------------------------------|-------------|-------------|--------------|-------------|
| SGPN [†] [42] | - | - | 36.0 | 28.7 |
| Dyco3D [†] [14] | - | - | 64.3 | 64.2 |
| PointGroup [†] [21] | - | 57.8 | 61.9 | 62.1 |
| HAIS [†] [3] | - | - | 71.1 | 65.0 |
| SSTNet [†] [26] | 42.7 | 59.3 | 65.5 | 64.2 |
| MaskGroup [†] [53] | - | 65.0 | 62.9 | 64.7 |
| SoftGroup [†] [41] | 51.6 | 66.1 | 73.6 | 66.6 |
| RPGN [†] [8] | - | - | 64.0 | 63.0 |
| PointInst3D [†] [15] | - | - | 73.1 | 65.2 |
| DKNNet [†] [45] | - | - | 70.8 | 65.3 |
| Ours[†] | 53.5 | 66.4 | 74.9 | 65.4 |
| SGPN [‡] [42] | - | - | 38.2 | 31.2 |
| PointGroup [‡] [21] | - | 64.0 | 69.6 | 69.2 |
| HAIS [‡] [3] | - | - | 73.2 | 69.4 |
| SSTNet [‡] [26] | 54.1 | 67.8 | 73.5 | 73.4 |
| MaskGroup [‡] [53] | - | 69.9 | 66.6 | 69.6 |
| SoftGroup [‡] [41] | 54.4 | 68.9 | 75.3 | 69.8 |
| RPGN [‡] [8] | - | - | 84.5 | 70.5 |
| PointInst3D [‡] [15] | - | - | 76.4 | 74.0 |
| DKNNet [‡] [45] | - | - | 75.3 | 71.1 |
| Ours[‡] | 59.5 | 70.6 | 80.1 | 72.9 |

Table 3. Quantitative Comparison on S3DIS. [†] and [‡] indicate respectively the results on Area 5 and 6-fold cross-validation.

4.3. Ablation Study and Analysis

To verify the effectiveness of our method, in this section, we conduct ablation experiments and parameter sensitivity analysis on the ScanNetV2 validation set. First, we verify two main modules in the network architecture (see Fig. 2): Binary Clustering (b) and Instance Refine (c). As shown in

Tab. 4, PBNet achieves significant improvements compared to the baseline. Comparing distance clustering, our binary clustering attains improvements on all three metric: mAP , AP_{50} and AP_{25} . Meanwhile, our refinement method based on local scenes also plays a vital role in improving performance. Remarkably, it manages to improve 4.1% w.r.t. mAP when the refinement is applied on top of binary clustering, which clearly demonstrates its effectiveness.

Ablation on Binary Clustering. We conduct further analysis on the effectiveness of binary clustering. Specifically, binary clustering includes Group HPs and Voting LPs. Tab. 5 analyzes the effectiveness of each part. In the part without LPs, we take LPs as background points. Evidently, a combination of both parts could lead to the best results. Qualitative comparison between with(w) and without(w/o) voting LPs in Fig. 7 also validates this point.

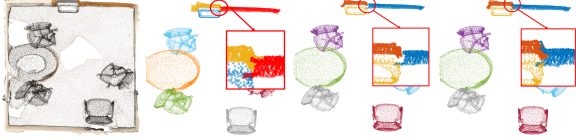
In addition, we examine if our binary clustering idea could work as a plug-in to improve other mainstream baselines. To this end, we take the baselines PointGroup [21] and HAIS [3] as two typical examples where we simply

| Distance Clustering | Binary Clustering | Instance Refine | mAP | AP_{50} | AP_{25} |
|---------------------|-------------------|-----------------|-------------|-------------|-------------|
| ✓ | | | 48.9 | 66.9 | 77.9 |
| | ✓ | | 50.4 | 68.3 | 78.6 |
| | ✓ | ✓ | 54.3 | 70.5 | 78.9 |

Table 4. Ablation Study on Network Module.

| Group HPs | Vote LPs | mAP | AP_{50} | AP_{25} |
|-----------|----------|-------------|-------------|-------------|
| ✓ | | 52.9 | 68.7 | 78.4 |
| ✓ | ✓ | 54.3 | 70.5 | 78.9 |

Table 5. Ablation Study on Binary Clustering.



Input Instance GT w/o Voting w Voting

Figure 7. Ablation Study on Voting LPs. Red boxes highlight the difference between results with (w) and without (w/o) voting LPs.

replace the traditional distance clustering with our binary clustering. To be fair, we directly use their published pre-trained model for validation. We report the results in Tab. 6. As clearly observed, our binary clustering leads to substantial improvements as opposed to distance clustering, verifying the advantages of our proposed method.

| Baseline Model | mAP | AP_{50} | AP_{25} |
|-----------------|-------------|-------------|-------------|
| PointGroup [21] | 35.5(+2.0%) | 58.4(+2.6%) | 72.3(+1.4%) |
| H AIS [3] | 44.7(+2.8%) | 65.7(+2.5%) | 76.0(+0.5%) |

Table 6. Our binary clustering leads to consistent improvement by simply replacing distance clustering on various baselines.

Ablation on Instance Refine. In Tab. 7, we report the ablation experiment results of instance refine. Notably, the proposed mask loss boosts both the mAP and AP_{50} metric. Combined with the local scene mechanism, instance refine increases the performance w.r.t. all three metrics. Particularly, it improves by a relative 7.7% and 3.2% on mAP and AP_{50} against the baseline respectively.

| Baseline | Local Scene | Mask Loss | mAP | AP_{50} | AP_{25} |
|----------|-------------|-----------|-------------|-------------|-------------|
| ✓ | | | 50.4 | 68.3 | 78.6 |
| | | ✓ | 53.0 | 68.8 | 78.5 |
| | ✓ | ✓ | 54.3 | 70.5 | 78.9 |

Table 7. Ablation Study on Instance Refine.

4.4. Efficiency

We examine the efficiency of our PBNet in this section. A single RTX 3090 is adopted to conduct this experiment on the ScanNetV2 validation set. In detail, we report the average inference time for each component of our network architecture in Tab. 8. The baseline includes backbone, 3D convolution, MLP, and data conversion.

| Baseline | Group HPs | Vote LPs | Local Scene | Post-process | Infer. Time(ms) |
|----------|-----------|----------|-------------|--------------|-----------------|
| ✓ | | | | | 190.8 |
| ✓ | ✓ | | | | 322.9 |
| ✓ | ✓ | ✓ | | | 339.5 |
| ✓ | ✓ | ✓ | ✓ | | 402.0 |
| ✓ | ✓ | ✓ | ✓ | ✓ | 420.8 |

Table 8. Average Inference Time(per scene).

As shown in Tab. 9, our PBNet takes an average of 420ms for each 3D scene inference on a single RTX 3090, which is still efficient in practice. Furthermore, H AIS [3] is currently the fastest inference method for 3D instance segmentation. In contrast, our PBNet only introduces limited latency (150-250 ms) but achieves a significant mAP improvement. Compared with another fast model DKNet [45], our method is slightly slower with a limited latency (about 63 ms). Overall, our algorithm is still reasonably efficient though it is slower than H AIS and DKNet. Given the significant mAP improvement, we believe it is a worthwhile trade-off and we will leave the exploration of speeding up our algorithm as future work.

| Methods | mAP | AP_{50} | AP_{25} | Infer. Time(ms) |
|------------|-------------|-------------|-------------|-----------------|
| H AIS [3] | 43.5 | 64.1 | 75.6 | 206.0 |
| DKNet [45] | 51.5 | 67.0 | 77.0 | 357.5 |
| Ours | 54.3 | 70.5 | 78.9 | 420.8 |

Table 9. Average Inference Time Comparison (per scene).

Parameter Analysis. The clustering-based methods all contain fine-tuning parameters. For example, DKNet [45] includes three parameters: r_d , α , and T_θ , where α is the formula coefficient and T_θ is the normalized centroid score threshold; RPGN [8] has five parameters. In comparison, our method needs three parameters: r_d , d_θ , K , where r_d and d_θ are used in binary clustering, and K is for constructing local scenes. In Fig. 8, we conduct parameter sensitivity analysis on the ScanNetV2 validation set.

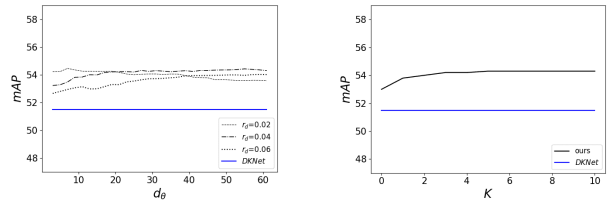


Figure 8. Parameter Sensitivity Analysis.

Specifically, we plot mAP by setting r_d to 0.02, 0.04 and 0.06, and continuously increasing the density threshold d_θ . Obviously, when r_d is 0.04, mAP appears stable especially when d_θ is greater than 20. In our experiments, we thus set r_d and d_θ to 0.04 and 30 respectively. We also evaluate mAP vs. K . When K is greater than 6, mAP stabilizes and does not change. Hence, we set K to 6. Overall, the number of hyperparameters is three in our method, which is parred to or fewer than that of SOTAs. All these parameters appear less sensitive as observed empirically.

5. Conclusion

We propose a novel divide and conquer strategy for 3D point cloud instance segmentation with point-wise binarization. Termed as PBNet, our end-to-end network makes a first attempt to divide offset instance points into

two categories: high and low density points (HPs vs. LPs). While HPs can be leveraged to separate adjacent objects confidently, LPs can help complete and refine instances via a novel neighbor voting scheme. We have developed a local scene mechanism to refine instances and suppress over-segmentation. Extensive experiments on benchmark ScanNetV2 and S3DIS datasets have shown that our model can overall beat the existing best models. In the future, we will explore how to speed up our algorithm.

Acknowledgement. This work was in part supported by the Jiangsu Science and Technology Programme (Natural Science Foundation of Jiangsu Province) under No.BE2020006-4, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under No.22KJB520039, and the National Natural Science Foundation of China under No.62206225.

References

- [1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, pages 1534–1543, 2016. [2](#), [5](#)
- [2] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiao-xiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, pages 4974–4983, 2019. [1](#)
- [3] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical aggregation for 3d instance segmentation. In *ICCV*, pages 15467–15476, 2021. [1](#), [2](#), [3](#), [5](#), [6](#), [7](#), [8](#)
- [4] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, pages 8958–8966, 2019. [6](#)
- [5] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. [2](#), [5](#)
- [6] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *ECCV*, pages 452–468, 2018. [2](#)
- [7] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM TOG*, 36(4):1, 2017. [1](#)
- [8] Shichao Dong, Guosheng Lin, and Tzu-Yi Hung. Learning regional purity for instance segmentation on 3d point clouds. In *ECCV*, pages 56–72. Springer, 2022. [6](#), [7](#), [8](#)
- [9] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *CVPR*, pages 9031–9040, 2020. [2](#), [6](#)
- [10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. Density-based spatial clustering of applications with noise. In *KDD*, volume 240, page 6, 1996. [1](#), [4](#)
- [11] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *PAMI*, 43(12):4338–4364, 2020. [1](#), [2](#)
- [12] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *CVPR*, pages 2940–2949, 2020. [6](#)
- [13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. [1](#), [3](#)
- [14] Tong He, Chunhua Shen, and Anton van den Hengel. Dyc3d: Robust instance segmentation of 3d point clouds through dynamic convolution. In *CVPR*, pages 354–363, 2021. [1](#), [2](#), [6](#), [7](#)
- [15] Tong He, Wei Yin, Chunhua Shen, and Anton van den Hengel. Pointinst3d: Segmenting 3d instances by points. In *ECCV*, pages 286–302. Springer, 2022. [6](#), [7](#)
- [16] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *CVPR*, pages 4421–4430, 2019. [2](#)
- [17] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimensional scene understanding. In *CVPR*, pages 14373–14382, 2021. [2](#)
- [18] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, pages 6409–6418, 2019. [5](#)
- [19] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NeurIPS*, pages 2017–2025, 2015. [2](#)
- [20] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yunying Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, pages 784–799, 2018. [5](#)
- [21] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *CVPR*, pages 4867–4876, 2020. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [7](#), [8](#)
- [22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, page n.pag, 2015. [6](#)
- [23] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3f semantic segmentation. In *ECCV*, pages 518–535, 2020. [2](#)
- [24] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *ICCV*, pages 9256–9266, 2019. [4](#)
- [25] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xinggang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *CVPR*, pages 1019–1028, 2019. [3](#), [5](#)
- [26] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks. In *CVPR*, pages 2783–2792, 2021. [2](#), [3](#), [5](#), [6](#), [7](#)
- [27] Huayao Liu, Ruiping Liu, Kailun Yang, Jiaming Zhang, Kunyu Peng, and Rainer Stiefelhagen. Hida: Towards holistic indoor understanding for the visually impaired via semantic instance segmentation with a wearable solid-state lidar sensor. In *ICCV*, pages 1780–1790, 2021. [1](#), [2](#)

- [28] Shikun Liu, Andrew J Davison, and Edward Johns. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015. 1
- [29] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018. 1
- [30] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, page n.pag, 2016. 6
- [31] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *PR*, pages 922–928, 2015. 2
- [32] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3d: Out-of-context data augmentation for 3d scenes. In *3DV*, pages 116–125, 2021. 6
- [33] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, pages 9277–9286, 2019. 2, 3, 6
- [34] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 2
- [35] 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*, pages 5099–5108, 2017. 2
- [36] Dario Rethage, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari. Fully-convolutional point networks for large-scale point clouds. In *ECCV*, pages 596–611, 2018. 2
- [37] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, pages 3577–3586, 2017. 2
- [38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 3, 6
- [39] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, pages 945–953, 2015. 2
- [40] R Talavera-Llames, Rubén Pérez-Chacón, A Troncoso, and Francisco Martínez-Álvarez. Big data time series forecasting based on nearest neighbours distributed computing with spark. *Knowledge-Based Systems*, 161:12–25, 2018. 4
- [41] Thang Vu, Kookhoi Kim, Tung M. Luu, Xuan Thanh Nguyen, and Chang D. Yoo. Softgroup for 3d instance segmentation on 3d point clouds. In *CVPR*, 2022. 1, 2, 3, 6, 7
- [42] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *CVPR*, pages 2569–2578, 2018. 2, 6, 7
- [43] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5):1–12, 2019. 2
- [44] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, pages 9621–9630, 2019. 2
- [45] Yizheng Wu, Min Shi, Shuaiyuan Du, Hao Lu, Zhiguo Cao, and Weicai Zhong. 3d instances as 1d kernels. In *ECCV*, pages 235–252, 2022. 1, 2, 3, 5, 6, 7, 8
- [46] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. Mlcvnet: Multi-level context votenet for 3d object detection. In *CVPR*, pages 10447–10456, 2020. 2
- [47] 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. In *NeurIPS*, pages 6737–6746, 2019. 2, 6
- [48] Chaolong Yang, Yuyao Yan, Weiguang Zhao, Jianan Ye, Xi Yang, Amir Hussain, and Kaizhu Huang. Towards deeper and better multi-view feature fusion for 3d semantic segmentation. *arXiv preprint arXiv:2212.06682*, 2022. 1
- [49] 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 *CVPR*, pages 3947–3956, 2019. 2, 6
- [50] Biao Zhang and Peter Wonka. Point cloud instance segmentation using probabilistic embeddings. In *CVPR*, pages 8883–8892, 2021. 6
- [51] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *PR*, 40(7):2038–2048, 2007. 4
- [52] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. In *NeurIPS*, pages 10326–10338, 2021. 3, 4
- [53] Min Zhong, Xinghao Chen, Xiaokang Chen, Gang Zeng, and Yunhe Wang. Maskgroup: Hierarchical point grouping and masking for 3d instance segmentation. In *ICME*, pages 1–6, 2022. 2, 6, 7