PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation

Li Jiang\textsuperscript{1}\ast  Hengshuang Zhao\textsuperscript{1}\ast  Shaoshuai Shi\textsuperscript{1}  Shu Liu\textsuperscript{2}  Chi-Wing Fu\textsuperscript{1}  Jiaya Jia\textsuperscript{1,2}
\textsuperscript{1}The Chinese University of Hong Kong  \textsuperscript{2}SmartMore
\{lijiang, hszhao, cwfu, leojia\}@cse.cuhk.edu.hk  ssshi@ee.cuhk.edu.hk  sliu@smartmore.com

Abstract

Instance segmentation is an important task for scene understanding. Compared to the fully-developed 2D, 3D instance segmentation for point clouds have much room to improve. In this paper, we present PointGroup, a new end-to-end bottom-up architecture, specifically focused on better grouping the points by exploring the void space between objects. We design a two-branch network to extract point features and predict semantic labels and offsets, for shifting each point towards its respective instance centroid. A clustering component is followed to utilize both the original and offset-shifted point coordinate sets, taking advantage of their complementary strength. Further, we formulate the ScoreNet to evaluate the candidate instances, followed by the Non-Maximum Suppression (NMS) to remove duplicates. We conduct extensive experiments on two challenging datasets, ScanNet v2 and S3DIS, on which our method achieves the highest performance, 63.6\% and 64.0\%, compared to 54.9\% and 54.4\% achieved by former best solutions in terms of mAP with IoU threshold 0.5.

1. Introduction

Instance segmentation is a fundamental and challenging task that requires to predict not only the semantic labels but also the instance IDs for every object in the scene. It has drawn much interest recently, given the potential applications for both outdoor and indoor environment regarding autonomous driving, robot navigation, to name a few.

Convolutional neural networks has boosted the performance of 2D instance segmentation \cite{10, 17, 29, 5}. However, given unordered and unstructured 3D point clouds, 2D methods cannot be directly extended to 3D points and make the latter remains very challenging \cite{49, 19, 53}. In this paper, we address the challenging 3D point cloud instance segmentation task by exploring the void space between 3D objects, along with the semantic information, to better segment individual objects.

\ast Equal Contribution.
Net v2, our performance on the test set is 63.6% in terms of mAP50, which is 8.7% higher than the former best solution [23]. For S3DIS, we accomplish 64.0% mAP50, 69.6% mPrec50, and 69.2% mRec50, outperforming all previous approaches by a large margin.

In summary, our contribution is threefold.

- We propose a bottom-up 3D instance segmentation framework, named PointGroup, to deal with the challenging 3D instance segmentation task.
- We propose a point clustering method based on dual coordinate sets, i.e., the original and shifted sets. Along with the new ScoreNet, object instances can be better segmented out.
- The proposed method achieves state-of-the-art results on various challenging datasets, demonstrating its effectiveness and generality.

2. Related Work

**Deep Learning in 3D Scenes**  2D image pixels are in regular grids, thus can be naturally processed by convolutional neural networks [24, 22, 42, 46, 18]. In contrast, 3D point clouds are unordered and scattered in 3D space, causing extra difficulty in point cloud scene understanding [37, 41].

Several approaches handle data irregularity. The Multi-Layer Perception (MLP)-style networks, e.g., PointNet [35, 37], directly apply MLP together with max-pooling to grab local and global structures in 3D. The learned features are then used for point cloud classification and segmentation. Other approaches [51, 48, 57, 52, 21] enhance feature learning on local regions by dynamic context aggregation and attention modules.

Besides working directly on the irregular input, several approaches transform the unordered point set to an ordered one to apply the convolution operations. PointCNN [26] learns the order transformation for points reweighting and permutation. Some other approaches [30, 43, 47, 39, 13, 7] align and voxelize point cloud to produce regular 3D ordered tensors for 3D convolution. Multi-view strategies [36, 44, 45] are also widely explored, where 3D point clouds are projected into 2D views for view-domain processing.

**2D Instance Segmentation**  Instance segmentation aims to find the foreground objects in a scene and mark each object instance with a unique label. Overall, there are two major lines. The first is detection- or top-down-based, which directly detects object instances. Early works [14, 15] use proposals from MCG [1] for feature extraction. Methods of [9, 10, 16] adopt pooled features for faster processing. Mask R-CNN [17] is widely known as an effective approach with the extra segmentation head in the detection framework, like Faster R-CNN [38]. Further works [29, 6, 5] enhance the feature learning for instance segmentation.

The other line is segmentation- or bottom-up-based, where pixel-level semantic segmentation is performed followed by grouping of pixels to find object instances. Zhang et al. [56, 55] utilize MRF for local patch merging. Arnab and Torr [3] use CRF. Bai and Urtasun [4] combine the classical watershed transform and deep learning to produce energy maps to distinguish among individual instances. Liu et al. [28] employ a sequence of neural networks to construct objects from pixels.

**3D Instance Segmentation**  With available large-scale 3D labeled datasets [8, 2], instance segmentation of 3D point clouds becomes important. Similar to 2D cases, current 3D methods can also be grouped into two lines. Detection-based methods extract 3D bounding boxes, and inside each box, utilize a mask learning branch to predict the object mask. Yang et al. [53] present the 3D-BoNet that directly predicts 3D bounding boxes and point-level masks simultaneously per instance. Li et al. [54] propose GSPN, which takes an analysis-by-synthesis strategy to generate proposals for instance segmentation. Hou et al. [19] combine multi-view RGB input with 3D geometry to jointly infer object bounding boxes and corresponding instance masks in an end-to-end manner.

Contrarily, segmentation-based methods predict the semantic labels, and utilize point embedding to group points into object instances. Wang et al. [49] design SGPN by clustering points based on the semantic segmentation predicted by backbones such as PointNet++. Liu and Furukawa [27] predict both the semantic labels and affinity between adjacent voxels in different scales to group instances. Phm et al. [33] develop a multi-task learning framework with a multi-value CRF model to jointly reason over both the semantic and instance labels. Wang et al. [50] learn a semantic-aware point-level instance embedding to benefit learning of both the semantic and instance tasks. Lahoud et al. [23] introduce a multi-task learning strategy where points of the same instance are grouped closer and different clusters are more separated from each other.

Different from the above methods, we present a new approach named PointGroup to tackle the 3D instance segmentation task. Our proposed model mainly contains two parts – that is, (i) learning to group points into different clusters based on their semantic predictions in both the original coordinate space and shifted coordinate space, and (ii) ScoreNet to learn to predict the score for selecting proper clusters. The overall framework is differentiable. It can be jointly optimized and trained in an end-to-end manner.

3. Our Method

3.1. Architecture Overview

To obtain instance-level segmentation labels for 3D objects, we consider two problems. The first is to separate
the contents in the 3D space into individual objects, and the second is to determine the semantic label of each object. Unlike 2D images, there is no view-occlusion problem in the 3D scenes, and objects scattered in 3D are usually naturally separated by void space. Hence, we propose leveraging these characteristics of 3D objects to group 3D content into object instances according to the semantic information. Fig. 2 overviews the architecture of our approach, which has three main components, i.e., the backbone, the point clustering part, and ScoreNet.

The input to the backbone network (Fig. 2(a)) is a point set \( P \) of \( N \) points. Each point has a color \( f_i = (r_i, g_i, b_i) \) and 3D coordinates \( p_i = (x_i, y_i, z_i) \), where \( i \in \{1, \ldots, N\} \). The backbone extracts feature \( F_i \) for each point. We denote the output feature of the backbone as \( F = \{F_i\} \in \mathbb{R}^{N \times K} \), where \( K \) is the number of channels. We then feed \( F \) into two branches, one for semantic segmentation and the other for predicting a per-point offset vector to shift each point towards the centroid of its respective object instance. Let \( s_i \) and \( o_i = (\Delta x_i, \Delta y_i, \Delta z_i) \) denote the predicted semantic label and offset vector of point \( i \), respectively.

After obtaining the semantic labels, we begin to group points into instance clusters based on the void space between objects. In the point clustering part (Fig. 2(b)), we introduce a clustering method to group points that are close to each other into the same cluster, if they have the same semantic label. However, clustering directly based on the point coordinate set \( P = \{p_i\} \) may fail to separate same-category objects that are close to each other in the 3D space and mis-group them, for example, two pictures that hang side-by-side on the wall.

Thus, we use the learned offset \( o_i \) to shift point \( i \) towards its respective instance centroid and obtain the shifted coordinates \( q_i = p_i + o_i \in \mathbb{R}^3 \). For points belonging to the same object instance, different from \( p_i \), the shifted coordinates \( q_i \) cluster around the same centroid. So by clustering based on shifted coordinate set \( Q = \{q_i\} \), we separate nearby objects better, even though they have the same semantic labels.

However, for points near object boundary, the predicted offsets may not be accurate. So, our clustering algorithm employs “dual” point coordinate sets, i.e., the original coordinates \( P \) and the shifted coordinates \( Q \). We denote the clustering results \( C \) as the union of \( C^p = \{C_1^p, \ldots, C_{M_p}^p\} \) and \( C^q = \{C_1^q, \ldots, C_{M_q}^q\} \), which are the clusters discovered based on \( P \) and \( Q \), respectively. Here, \( M_p \) and \( M_q \) denote the number of clusters in \( C^p \) and \( C^q \), respectively, and \( M = M_p + M_q \) denotes the total.

Lastly, we construct the ScoreNet (Fig. 2(c)) to process the proposed point clusters \( C = C^p \cup C^q \) and produce a score per cluster proposal. NMS is then applied to these proposals with the scores to generate final instance prediction. In the following, we denote the instance predictions as \( G = \{G_1, \ldots, G_{M_{pred}}\} \subseteq C \) and the ground-truth instances as \( I = \{I_1, \ldots, I_{M_{gt}}\} \). Here, \( G_i \) and \( I_i \) are subsets of \( P \), while \( M_{pred} \) and \( M_{gt} \) denote the number of instances in \( G \) and \( I \), respectively. Also, we use \( N_i^p \) and \( N_i^q \) to represent the number of points in \( G_i \) and \( I_i \), respectively.

### 3.2. Backbone Network

We may use any point feature extraction network to serve as the backbone network (Fig. 2(a)). In our implementation, we voxelize the points and follow the procedure of [13] to construct a U-Net [25, 40] with Submanifold Sparse Convolution (SSC) and Sparse Convolution (SC). We then recover points from voxels to obtain the point-wise features \( F \) for subsequent processing. Afterwards, we construct two branches based on the point-wise features \( F \) to predict semantic label \( s_i \) and offset vector \( o_i \) for each point.
Semantic Segmentation Branch  We apply an MLP to F to produce semantic scores \( \mathbf{S} = \{s_{c_1}, ..., s_{c_N}\} \in \mathbb{R}^{N \times N_{\text{class}}} \) for the \( N \) points over the \( N_{\text{class}} \) classes, and regularize the results by a cross entropy loss \( L_{\text{sem}} \). The predicted semantic label \( s_i \) for point \( i \) is the class with the maximum score, i.e., \( s_i = \text{argmax}(s_{c_i}) \).

Offset Prediction Branch  The offset branch encodes \( F \) to produce \( N \) offset vectors \( \mathbf{O} = \{o_1, ..., o_N\} \in \mathbb{R}^{N \times 3} \) for the \( N \) points. For points belonging to the same instance, we constrain their learned offsets by an \( L_1 \) regression loss as

\[
L_{o_{x,y,z}} = \frac{1}{\sum_i m_i} \sum_i ||o_i - (\hat{c}_i - p_i)|| \cdot m_i,
\]

where \( m = \{m_1, ..., m_N\} \) is a binary mask. \( m_i = 1 \) if point \( i \) is on an instance and \( m_i = 0 \) otherwise. \( \hat{c}_i \) is the centroid of the instance that point \( i \) belongs to, i.e.,

\[
\hat{c}_i = \frac{1}{N_{g(i)}} \sum_{j \in I_{g(i)}} p_j,
\]

where \( g(i) \) maps point \( i \) to the index of its corresponding ground-truth instance, i.e., the instance that contains point \( i \). \( N_{g(i)} \) is the number of points in instance \( I_{g(i)} \).

The above mechanism looks similar to the vote generation strategy in VoteNet [34]. However, rather than regressing the bounding boxes based on the votes of a few subsampled seed points, we predict an offset vector per point to gather the instance points around a common instance centroid, in order to better cluster relevant points into the same instance. Also, we observe that the distances from points to their instance centroids usually have small values (0 to 1m). Fig. 3b gives the statistical analysis on the distribution of such distances in the ScanNet dataset. Considering diverse object sizes of different categories, we find it is hard for the network to regress precise offsets, particularly for boundary points of large-size objects, since these points are relatively far from the instance centroids. To address this issue, we formulate a direction loss to constrain the direction of predicted offset vectors. We follow [23] to define the loss as a means of minus cosine similarities, i.e.,

\[
L_{o_{x,y,z}} = -\frac{1}{\sum_i m_i} \sum_i \frac{o_i}{||a_i||_2} \cdot \frac{\hat{c}_i - p_i}{||\hat{c}_i - p_i||_2} \cdot m_i.
\]

Such loss is irrelevant to the offset vector norm and ensures that the points move towards their instance centroids.

3.3. Clustering Algorithm

Given the predicted semantic labels and offset vectors, we are ready to group the input points into instances. To this end, we introduce a simple and yet effective clustering algorithm. It is detailed in Algorithm 1.

**Algorithm 1** Clustering algorithm. \( N \) is the number of points. \( M \) is the number of clusters found by the algorithm.

**Input:** clustering radius \( r \);
- cluster point number threshold \( N_{\theta} \);
- coordinates \( X = \{x_1, x_2, ..., x_N\} \in \mathbb{R}^{N \times 3} \); and
- semantic labels \( S = \{s_1, ..., s_N\} \in \mathbb{R}^N \).

**Output:** clusters \( C = \{C_1, ..., C_M\} \).

1. initialize an array \( v \) (visited) of length \( N \) with all zeros
2. initialize an empty cluster set \( C \)
3. for \( i = 1 \) to \( N \) do
4.  if \( s_i \) is a stuff class (e.g., wall) then
5.    \( v_i = 1 \)
6.  for \( i = 1 \) to \( N \) do
7.    if \( v_i == 0 \) then
8.      initialize an empty queue \( Q \)
9.      initialize an empty cluster \( C \)
10.     \( v_i = 1 \); \( Q\).enqueue(\( i \)); add \( i \) to \( C \)
11.    while \( Q \) is not empty do
12.      \( k = Q\).enqueue()
13.      for \( j \in [1, N] \) with \( ||x_j - x_k||_2 < r \) do
14.        if \( s_j == s_k \) and \( v_j == 0 \) then
15.          \( v_j = 1 \); \( Q\).enqueue(\( j \)); add \( j \) to \( C \)
16.      if number of points in \( C > N_{\theta} \) then
17.        add \( C \) to \( C \)
18. return \( C \)

The core step of our algorithm is that for point \( i \), we get points within the ball of radius \( r \) centered at \( x_i \) (the coordinate of point \( i \)) and group points with the same semantic labels as point \( i \) into the same cluster. Here, \( r \) serves as a spatial constraint in the clustering, so that two intra-category objects at a distance larger than \( r \) are not grouped. Here, we use the breadth-first search to group points of the same instance into a cluster. In our implementation, for points in the scene, neighboring points within an \( r \)-sphere can be found in parallel in advance of the clustering to boost speed.

As presented in Sec. 3.1, we apply the clustering algorithm separately on the “dual” set, i.e., the original coordinate set \( \mathbf{P} \) and the shifted set \( \mathbf{Q} \), to produce cluster sets \( \mathbf{C}_{\mathbf{P}} \) and \( \mathbf{C}_{\mathbf{Q}} \). Clustering on \( \mathbf{P} \) may mis-group nearby objects of the same class, while clustering on \( \mathbf{Q} \) does not have this problem but may fail to handle the boundary points of large objects. We collectively employ \( \mathbf{P} \) and \( \mathbf{Q} \) to find candidate clusters due to their complementary properties. Analysis on the clustering performance of using \( \mathbf{P} \) alone, \( \mathbf{Q} \) alone, or both \( \mathbf{P} \) and \( \mathbf{Q} \) is presented in Sec. 4.2.2.

3.4. ScoreNet

The input to ScoreNet is the set of candidate clusters \( \mathbf{C} = \{C_1, ..., C_M\} \), where \( M \) denotes the total number of candidate clusters, and \( C_i \) denotes the \( i \)-th cluster. Also, we use \( N_i \) to represent the number of points in \( C_i \). The goal of
ScoreNet is to predict a score for each cluster to indicate the quality of the associated cluster proposal, so that we could precisely reserve the better clusters in NMS and thus combine strength of $C^p$ and $C^q$.

To start, for each cluster, we gather the point features from $F \in \mathbb{R}^{N \times K}$ (the features extracted by the backbone) and form $F_{C_i} = \{F_{h(C_i,1)}, ..., F_{h(C_i,N)}\}$ for cluster $C_i$, where $h$ maps the point index in $C_i$ to corresponding point index in $F$. Similarly, we express the coordinates for points in $C_i$ as $P_{C_i} = \{p_{h(C_i,1)}, ..., p_{h(C_i,N)}\}$.

To better aggregate the cluster information, we take $F_{C_i}$ and $P_{C_i}$ as the initial features and coordinates, and voxelize the clusters the same way as we do at the beginning of the backbone network. The feature for each voxel is average-pooled from the initial features of points in that voxel. We then feed them into a small U-Net with SSC and SC to further encode the features. A cluster-aware max-pooling is then followed to produce a single cluster feature vector $f_{C_i} \in \mathbb{R}^{1 \times K_c}$ per cluster. The final cluster scores $S_c = \{s_1^c, ..., s_M^c\} \in \mathbb{R}^M$ are obtained as

$$S_c = \text{Sigmoid}(\text{MLP}(F_{C_i})), \quad (4)$$

where $F_{C_i} = \{f_{C_i,1}, ..., f_{C_i,M}\} \in \mathbb{R}^{M \times K_c}$. The structure of ScoreNet is illustrated in Fig. 3a.

Inspired by [25, 20], to reflect the quality of clusters in the scores, we use a soft label to replace a binary 0/1 label to supervise the predicted cluster score as

$$s_i^c = \begin{cases} 0 & \text{iou}_i < \theta_l \\ 1 & \text{iou}_i > \theta_h \\ \frac{1}{\frac{1}{\text{iou}} - \frac{1}{\theta_l}} \cdot (\text{iou}_i - \theta_l) & \text{otherwise} \end{cases}, \quad (5)$$

where $\theta_l$ and $\theta_h$ are empirically set to 0.25 and 0.75 respectively in our implementation, and $\text{iou}_i$ is the largest Intersection over Union (IoU) between cluster $C_i$ and ground-truth instances as

$$\text{iou}_i = \max \left( \{\text{IoU}(C_i, I_j) \mid I_j \in \mathbb{I} \} \right). \quad (6)$$

We then use the binary cross-entropy loss as our score loss, which is formulated as

$$L_{c, s} = -\frac{1}{M} \sum_{i=1}^{M} (\hat{s}_i^c \log(s_i^c) + (1 - \hat{s}_i^c) \log(1 - s_i^c)). \quad (7)$$

3.5. Network Training and Inference

Training We train the whole framework in an end-to-end manner with the total loss as

$$L = L_{sem} + L_{o, dir} + L_{o, reg} + L_{c, s}. \quad (8)$$

Inference In the inference process, we perform NMS on clusters $C$ with predicted scores $S_c$ to obtain the final instance predictions $G \subseteq C$. The IoU threshold is empirically set as 0.3. Since we cluster based on the semantic information, the semantic label of a cluster is exactly the category that the cluster points belong to.

4. Experiments

Our proposed PointGroup architecture is effective for instance segmentation of 3D point clouds. To demonstrate its effectiveness, we conduct extensive experiments on two challenging point cloud datasets, ScanNet v2 [8] and S3DIS [2]. On both of them, we achieve state-of-the-art performance on the 3D instance segmentation task.

4.1. Experimental Setting

Datasets The ScanNet v2 [8] dataset contains 1,613 scans with 3D object instance annotations. The dataset is split into training, validation, and testing sets, each with 1,201, 312, and 100 scans, respectively. 18 object categories are used for instance segmentation evaluation. For ablation studies, we train on the training set and report results on the validation set. To compare with other approaches, we train on the training set and report results on the testing set.

The S3DIS [2] dataset has 3D scans across six areas with 271 scenes in total. Each point is assigned one label out of 13 semantic classes. All the 13 classes are used in instance segmentation.
evaluation. Overall, we evaluate our model under two settings: (i) Area 5 is adopted for testing, whereas all the others are used for training; and (ii) six-fold cross validation that each area is treated as the testing set once.

**Evaluation Metrics** We use the widely-adopted evaluation metric – mean average precision (mAP). Specifically, AP$_{25}$ and AP$_{50}$ denote the AP scores with IoU threshold set to 25% and 50%, respectively. Also, AP averages the scores with IoU threshold set from 50% to 95%, with a step size of 5%. Besides, approaches of [50, 53] reported performance of mean precision (mPrec) and mean recall (mRec) on S3DIS, we also include these results for comparison.

**Implementation Details** We set the voxel size as 0.02m. In the clustering part, we set the clustering radius $r$ as 0.03m and the minimum cluster point number $N_0$ as 50. In the training process, we use the Adam solver with a base learning rate of 0.001. For each scene in the dataset, we set the maximum number of points as 250k, due to GPU memory limit. If the scene has more than 250k points, we randomly crop part of the scene and gradually adjust the crop size, according to the number of points in the cropped area. During the testing process, we feed the whole scene into the network without cropping.

Specifically, scenes in S3DIS have high point density. Some scenes are even with millions of points. Hence, for each S3DIS scene, we randomly sub-sample ~1/4 points before each cropping.

### 4.2. Evaluation on ScanNet

#### 4.2.1 Benchmark Results

We first report performance of our PointGroup model on the testing set of ScanNet v2, as listed in Table 1. PointGroup accomplishes the highest AP$_{50}$ score of 63.6%, outperforming all previous methods by a large margin. Compared with the former best solution [23], which obtains 54.9% AP$_{50}$ score, our result is 8.7% higher (absolute) and 15.8% better (relative). For detailed results on each category, PointGroup ranks the 1st place in 13 out of 18 classes in total.

#### 4.2.2 Ablation Studies

We conduct ablation studies on the ScanNet validation set to analyze the design and parameter choice in our PointGroup.

**Clustering based on Different Coordinate Sets** Table 2 shows the comparison using the original coordinates P alone, the shifted coordinates Q alone, and both P and Q in the clustering. Clustering on points with P alone may mis-group two close objects with the same semantic label into the same instance. Hence, for categories, in which two objects are likely to be very close to each other (e.g., chairs and pictures), clustering on P alone does not perform well. Clustering on Q solves the problem in part by gathering instance points around the instance centroids and enlarging the space between clusters. However, due to inaccuracy in offset prediction, especially for boundary points of large objects (e.g., curtains and counters), clustering on Q alone does not perform perfectly.

Fig. 4 shows the qualitative results with models trained with clusters from different coordinate sets – (i) P only, (ii) Q only, and (iii) both P and Q. We could observe that the problem in (i) is the mistakenly grouped pictures on the wall in one cluster. The case of (ii) successfully separates the pictures into individual instances. Nevertheless, it suffers from inaccuracy around the object boundary areas. The case of (iii) takes strength of both (i) and (ii). Clustering on dual point sets (both P and Q) along with the precise scores from ScoreNet to select the final instance clusters, we combine the advantages of clustering on P and on Q to attain the best performance.

**Ablation on the Clustering Radius $r$** We use different values of $r$ in the clustering algorithm. The performance varies as shown in Table 3. A small $r$ is sensitive to point density. The scan for an object may have inconsistent point density in different parts. Clustering with such an $r$ may not be able to grow in low-density parts. On the contrary, a large $r$ increases the risk of grouping two nearby same-class objects into one. We empirically set $r$ to 0.03 (meter).

#### Table 1: 3D instance segmentation results on ScanNet v2 testing set with AP$_{50}$ scores. Our proposed PointGroup approach yields the highest average AP$_{50}$, outperforming all-state-of-the-art methods by a large margin. All numbers are from the ScanNet benchmark on 15/11/2019.

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg AP$_{50}$</th>
<th>bedroom</th>
<th>bookcase</th>
<th>chair</th>
<th>curtain</th>
<th>desk</th>
<th>door</th>
<th>drawer</th>
<th>fridge</th>
<th>cabinet</th>
<th>book</th>
<th>table</th>
<th>table</th>
<th>table</th>
<th>mirror</th>
<th>bed</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
<th>mirror</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScoreNet [9]</td>
<td>0.143</td>
<td>0.208</td>
<td>0.390</td>
<td>0.169</td>
<td>0.065</td>
<td>0.275</td>
<td>0.028</td>
<td>0.069</td>
<td>0.000</td>
<td>0.087</td>
<td>0.041</td>
<td>0.014</td>
<td>0.027</td>
<td>0.007</td>
<td>0.112</td>
<td>0.351</td>
<td>0.168</td>
<td>0.438</td>
<td>0.138</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D-VEVIS [11]</td>
<td>0.248</td>
<td>0.667</td>
<td>0.566</td>
<td>0.076</td>
<td>0.035</td>
<td>0.394</td>
<td>0.027</td>
<td>0.035</td>
<td>0.098</td>
<td>0.099</td>
<td>0.030</td>
<td>0.025</td>
<td>0.098</td>
<td>0.037</td>
<td>0.126</td>
<td>0.604</td>
<td>0.181</td>
<td>0.854</td>
<td>0.171</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R-PointNet [54]</td>
<td>0.506</td>
<td>0.500</td>
<td>0.405</td>
<td>0.311</td>
<td>0.348</td>
<td>0.589</td>
<td>0.054</td>
<td>0.068</td>
<td>0.126</td>
<td>0.283</td>
<td>0.290</td>
<td>0.028</td>
<td>0.219</td>
<td>0.214</td>
<td>0.331</td>
<td>0.396</td>
<td>0.275</td>
<td>0.821</td>
<td>0.245</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPC [13]</td>
<td>0.555</td>
<td>0.500</td>
<td>0.517</td>
<td>0.467</td>
<td>0.228</td>
<td>0.422</td>
<td>0.411</td>
<td>0.405</td>
<td>0.111</td>
<td>0.205</td>
<td>0.241</td>
<td>0.075</td>
<td>0.233</td>
<td>0.306</td>
<td>0.445</td>
<td>0.439</td>
<td>0.457</td>
<td>0.974</td>
<td>0.235</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D-SIS [19]</td>
<td>0.582</td>
<td>1.000</td>
<td>0.432</td>
<td>0.245</td>
<td>0.190</td>
<td>0.577</td>
<td>0.013</td>
<td>0.263</td>
<td>0.033</td>
<td>0.320</td>
<td>0.240</td>
<td>0.075</td>
<td>0.233</td>
<td>0.306</td>
<td>0.445</td>
<td>0.439</td>
<td>0.457</td>
<td>0.974</td>
<td>0.235</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAS [27]</td>
<td>0.447</td>
<td>0.528</td>
<td>0.555</td>
<td>0.381</td>
<td>0.382</td>
<td>0.633</td>
<td>0.002</td>
<td>0.000</td>
<td>0.613</td>
<td>0.175</td>
<td>0.250</td>
<td>0.434</td>
<td>0.437</td>
<td>0.411</td>
<td>0.857</td>
<td>0.405</td>
<td>0.591</td>
<td>0.267</td>
<td>0.944</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PanopticFusion [32]</td>
<td>0.478</td>
<td>0.667</td>
<td>0.712</td>
<td>0.595</td>
<td>0.259</td>
<td>0.550</td>
<td>0.000</td>
<td>0.613</td>
<td>0.175</td>
<td>0.250</td>
<td>0.434</td>
<td>0.437</td>
<td>0.411</td>
<td>0.857</td>
<td>0.405</td>
<td>0.591</td>
<td>0.267</td>
<td>0.944</td>
<td>0.335</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D-BoNet [53]</td>
<td>0.488</td>
<td>1.000</td>
<td>0.672</td>
<td>0.590</td>
<td>0.301</td>
<td>0.484</td>
<td>0.098</td>
<td>0.620</td>
<td>0.806</td>
<td>0.341</td>
<td>0.259</td>
<td>0.123</td>
<td>0.434</td>
<td>0.796</td>
<td>0.402</td>
<td>0.499</td>
<td>0.513</td>
<td>0.909</td>
<td>0.439</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTML [23]</td>
<td>0.549</td>
<td>1.000</td>
<td>0.807</td>
<td>0.588</td>
<td>0.327</td>
<td>0.647</td>
<td>0.004</td>
<td>0.815</td>
<td>0.180</td>
<td>0.418</td>
<td>0.364</td>
<td>0.182</td>
<td>0.445</td>
<td>1.000</td>
<td>0.442</td>
<td>0.688</td>
<td>0.571</td>
<td>1.000</td>
<td>0.396</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PointGroup (Ours)</td>
<td>0.636</td>
<td>1.000</td>
<td>0.765</td>
<td>0.624</td>
<td>0.505</td>
<td>0.797</td>
<td>0.116</td>
<td>0.696</td>
<td>0.384</td>
<td>0.441</td>
<td>0.559</td>
<td>0.476</td>
<td>0.596</td>
<td>1.000</td>
<td>0.666</td>
<td>0.756</td>
<td>0.556</td>
<td>0.997</td>
<td>0.513</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4872
Table 2: Ablation results using different coordinate sets on the ScanNet v2 validation set. Adopting both the original and shifted coordinates for clustering yields the best 3D instance segmentation performance.

Table 3: Ablation results for clustering with different radii on the ScanNet v2 validation set.

Table 4: Inference time (ms). BB denotes backbone + two branches; BQ denotes ballquary; subscripts p and q denote clustering on P and Q respectively; CL denotes our clustering algorithm; and SCN denotes ScoreNet.

Table 5: Instance segmentation results on the S3DIS validation set.

4.2.3 Runtime Analysis

Our method takes a whole scene as input per pass. Its runtime depends on the number of points and scene complexity. For runtime analysis, we sampled four scenes randomly from the ScanNet v2 validation set and tested them 100 times on a Titan Xp GPU to get an average runtime per scene. Table 4 reports the runtime breakdown. Clustering on Q (shifted) usually takes more time than clustering on P (original), as shifted points could have more neighbors.

4.3. Evaluation on S3DIS

We also evaluate our proposed PointGroup model on the S3DIS dataset. Apart from adopting AP\(_{50}\) as an evaluation metric, we also report mPrec\(_{50}\) and mRec\(_{50}\) on Q (shifted) as a more comprehensive measure.
Figure 5: Visualization of the semantic and instance segmentation results on ScanNet v2 (top) and S3DIS (bottom). For instance predictions, different colors represent separate instances, and the semantic results indicate the categories of instances.

metric, we also include the mPrec$_{50}$ and mRec$_{50}$ results in Table 5, where we use a score threshold of 0.2 to remove some low-confidence clusters.

PointGroup reaches the highest performance in terms of all three evaluation metrics. For results on Area 5, PointGroup gets 57.8% on AP$_{50}$, 61.9% on mPrec$_{50}$ and 62.1% on mRec$_{50}$. The mPrec$_{50}$ and mRec$_{50}$ are 6.6 and 19.7 points higher than ASIS [50], respectively. For the results on 6-fold cross validation, PointGroup is 9.6 points higher than SGPN [49] regarding AP$_{50}$, which is a big margin. The mPrec$_{50}$ and mRec$_{50}$ scores are 4 and 21.6 points higher than the second-best solution [53].

The large improvement of PointGroup over the former best approaches across different challenging datasets demonstrate its effectiveness and generality. Several visual illustrations of PointGroup over these two datasets are included in Fig. 5. We observe that the proposed approach well captures the 3D geometry information and obtains precise instance segmentation masks.

5. Conclusion

We have proposed PointGroup for 3D instance segmentation, with a specific focus of better grouping points by exploring the in-between space and point semantic labels among the object instances. Considering the situation that two intra-category objects may be very close to each other, we design a two-branch network to respectively learn a per-point semantic label and a per-point offset vector for moving each point towards its respective instance centroid. We then cluster points based on both the original point coordinates and the offset-shifted point coordinates. It combines the complementary strength of the two coordinate sets to optimize point grouping precision. Further, we introduced the ScoreNet to learn to evaluate the generated candidate clusters, followed by the NMS to avoid duplicates before we output the final predicted instances. PointGroup accomplished the best ever results.

In our future work, we plan to further introduce a progressive refinement module to relieve the semantic inaccuracy problem that affects the instance grouping and explore the possibility of incorporating weakly- or self-supervision techniques to further boost the performance.

Acknowledgments This project is supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK 14201717).
References

[36] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and


