End-to-end 3D Point Cloud Instance Segmentation without Detection

Haiyong Jiang[1,2], Feilong Yan[4], Jianfei Cai[2,3], Jianmin Zheng[2], Jun Xiao[1*]
[1]University of Chinese Academy of Science, [2]Nanyang Technological University,
[3]Monash University, [4]Huya Live

Abstract

3D instance segmentation plays a predominant role in environment perception of robotics and augmented reality. Many deep learning based methods have been presented recently for this task. These methods rely on either a detection branch to propose objects or a grouping step to assemble same-instance points. However, detection based methods do not ensure a consistent instance label for each point, while the grouping step requires parameter-tuning and is computationally expensive. In this paper, we introduce an assign-and-suppress network, dubbed as AS-Net, to enable end-to-end instance segmentation without detection and a separate step of grouping. The core idea is to frame instance segmentation as a candidate assignment problem. At first, a set of instance candidates are sampled. Then we propose an assignment module for candidate assignment and a suppression module to eliminate redundant candidates. A mapping between instance labels and instance candidates is further sought to construct an instance grouping loss for the network training. Experimental results demonstrate that our method is more effective and efficient than previous detection-free approaches.

1. Introduction

3D instance segmentation has wide applications spanning from 3D perception in autonomous systems to 3D reconstruction in augmented reality and virtual reality. For example, it is critical for an indoor robot to identify obstacles and targets in a scene so that it can interact with a specific object and move around the scene. Achieving this goal requires to distinguish different semantic labels as well as different instances with the same semantic label. Therefore, it is important to investigate the problem of 3D instance segmentation.

3D instance segmentation from point clouds is a very challenging task. While bearing the difficulties incurred by scattered data and an additional dimension, it also shares the same problems as its 2D counterpart. Firstly, instance labels are in a random order, which is quite different from semantic labels and makes it difficult to directly optimize instance labels in an end-to-end way. Secondly, the number of instances, which greatly impacts instance segmentation, is unknown during inference, thus posing additional challenges.

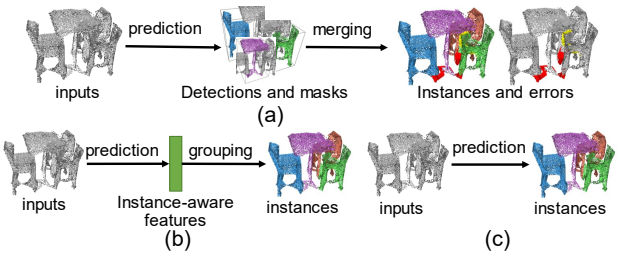

Figure 1. An illustration of different frameworks for 3D instance segmentation. (a) Detection-based framework, (b) detection-free framework, (c) ours. Note that there are inconsistent or missing labels when detection-based results are merged, as highlighted in red and yellow in (a).

Great progress has been made in 2D/3D instance segmentation [4, 7, 11, 12, 18, 22, 28, 29, 31]. In general, existing methods can be classified into two categories: detection-based and detection-free. Detection-based approaches [11, 12, 31] can well handle random instance labels and an irregular number of instances by using a detection branch to mask different objects. However, they cannot ensure consistent labeling for each point. For example, a point may get multiple instance labels or no label depending on how many segmented regions contain it, as illustrated in Fig. 1(a). On the other hand, detection-free approaches [18, 22, 28, 29] harness an additional grouping step to sidestep ordering and irregular number of instances, e.g. using mean-shift algorithm [22, 29], as depicted in Fig. 1(b). The additional group step usually relies on hyper-parameters setting like clustering bandwidth [21] for good performance. In addition, these methods often optimize a proxy objective instead of instance segmentation, e.g. respectively minimizing or maximizing the embedding feature distance between points of the same instance or two

*indicates the corresponding author. Email: haiyong.jiang1990@gmail
The work was carried out at Nanyang Technological University.

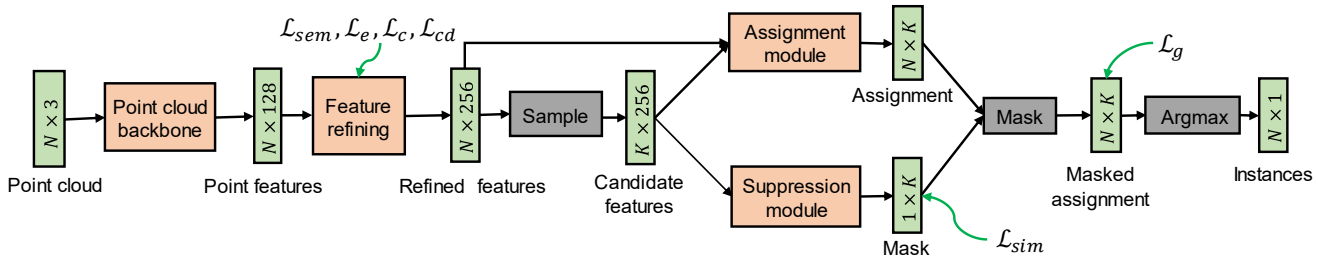

Figure 2. Pipeline of our method. Our method takes in a point cloud of N points, and learns point features with a point cloud feature extraction backbone. Point features are further refined by a refining module. Then a set of instance candidates representing different instances are sampled. Instance grouping then boils down to candidate assignment with an assignment module and redundant candidates are masked with a suppression module. Loss functions ($\mathcal{L}_{sem}, \mathcal{L}_e, \mathcal{L}_c, \mathcal{L}_{cd}, \mathcal{L}_g, \mathcal{L}_{sim}$) enforced at different positions are detailed in Sec. 3.5. K denotes the number of instance candidates. Boxes colored in and denote modules with and without learnable parameters, respectively.

different instances [4, 22, 28, 29]. Thus, there is a gap between the training objective and the final instance segmentation.

In this paper, we propose a novel framework, called ASNet, which aims at providing an end-to-end solution for 3D instance segmentation without detection. The whole pipeline is shown in Fig. 2. Instead of detecting a large number of objects for different instances, we sample a small set of instance candidates as instance representatives. Instance segmentation then becomes a problem of assigning points to different candidates with an assignment module. Then a suppression module is proposed to mask redundant candidates so that the irregular number of instances can be tamed. At last, we introduce a mapping between instance labels and instance candidates to facilitate the direct optimization of instance grouping with random orders of instances. In Fig. 1, we illustrate the differences between our framework and the previous ones.

In summary, our contributions are:

- A 3D instance segmentation framework that first samples a set of instance candidates, then assigns points to different candidates with an assignment module and eliminates duplicate candidates with a suppression module.
- An algorithm mapping instance labels to instance candidates and facilitating an end-to-end training of instance segmentation.
- Extensive experiments that show our method achieves superior results at much faster running speed, compared to existing methods.

2. Related Work

2D Instance Segmentation. 2D instance segmentation aims at both semantic classification and pixel grouping of instance objects, pioneered by [6, 9]. Recently, advanced deep learning has greatly pushed forward the performance of instance segmentation. Previous works can be classified into two streams, namely detection-based methods and detection-free methods.

Detection-based methods either employ a sliding-and-segmentation process [7, 23] or a joint detection-and-segmentation process [5, 9, 10, 11]. The sliding-and-segmentation based approaches [7, 23] suffer from inaccurate boundary prediction and segmentation of overlapping objects. The joint detection-and-segmentation methods, e.g. MaskRCNN [11], can generate nice instance segmentation, but require a larger memory footprint because of the use of the additional detection process.

The other category of works obtains instances by grouping pixels according to predicted pixel information, e.g. basin energy for watershed transform [3], shape information [2, 14, 20], semantic information [2], and embedding features [4, 8, 16, 21], etc. However, most of these methods require multi-stage processing during inferences. For example, embedding based approaches [4, 16, 21] first infer embedding features, and then group pixels with a clustering algorithm. This inevitably introduces gaps between training and testing phases as a proxy objective rather than pixel grouping loss is optimized. Also, additional hyper-parameters, e.g. clustering bandwidth [4, 21] and parameters in conditional random field [2], also require careful tuning to ensure good performance. Though Neven et al. [21] proposed to directly estimate these hyper-parameters, i.e. clustering bandwidth in mean-shift algorithm, it can only attenuate but cannot eliminate the gap.

3D Instance Segmentation for Point Clouds. Considering the prevalent point cloud data and its wide applications in autonomous driving and scene reconstruction, great interests in 3D point cloud instance segmentation have been spurred in the vision community. 3D instance segmentation can be treated as an analogy of 2D instance segmentation on 3D data, aiming at object-level understanding of 3D scene.

However, noisy data and unstructured topology make this problem more difficult. Seminal work [28] investigated 3D instance segmentation by exploring pairwise similarity matrix. Pham et al. [22] proposed a multi-value conditional random field (MV-CRF) to enhance instance segmentation and semantic segmentation. Wang et al. [29] explored the dependency between semantic features and instance features, achieving the-state-of-the-art results. These methods are all based on point cloud representation and generally use a multi-stage process as 2D proposal-free instance segmentation [4, 16, 18, 21], thus inheriting their limitations like hyper-parameter tuning and optimization gaps.

Another category of 3D instance segmentation methods is based on detection. In particular, Hou et al. [12] jointly optimized 3D detection and 3D semantic instance segmentation by extending MaskRCNN [11] to multi-modal signals in RGBD scans. Yi et al. [31] proposed a Generative Shape Proposal Network (SGPN) to generate shapes from different seeds, and then instance segmentation was attained by estimating a bounding box and its segmentation. Yang et al. [30] presented an interesting work which directly predicts a fixed number of bounding boxes and then estimates an instance mask for each bounding box. However, detection-based methods may assign no label or inconsistent instance labels to a point because of a missing shot or overlapping in detection segments. In this work, we try to learn 3D instance segmentation in an end-to-end fashion without detection and without the additional step of grouping.

3D Point Cloud Analysis. The great performance of deep neural network on 2D image analysis has motivated researchers to apply it to 3D point clouds. Qi et al. firstly proposed PointNet [24] and PointNet++ [25] in order to handle random orders of point clouds and multi-scaling feature extraction. Recent works also investigated the spherical kernel [19], angular representation [15], and tangent projection [27] to learn feature representations from point clouds. Our work relies on a backbone network to extract point cloud features. Though we use PointNet and PointNet++ for our evaluation, it can be easily changed to other architectures.

3. Proposed Method

We aim at instance segmentation of 3D point clouds in an end-to-end fashion without detection. Our method takes a 3D point cloud $X = \{x_j\}_{j=1}^{N}$ as input and predicts instance label $Y^i = \{y_j^i\}_{j=1}^{N}$ for each point as illustrated in Fig. 2, where N is the number of input points, x_j denotes input features of each point, e.g. coordinates and color, and y_j^i is an instance label in an instance label set L^i.

Different from previous two-stage detection-free methods [22, 28, 29] and detection-based methods [12], we formulate this problem as a one-stage process without an additional grouping step or detection. A point cloud is firstly processed with a point cloud backbone to extract semantic features, centroid-aware features, and instance-aware features by exploring the supervisions from semantic labels, instance centroid, and instance labels, where the centroid of an instance is obtained by calculating the mean coordinate of all points in an instance. These features are then concatenated as refined features $F^r \in \mathbb{R}^{N\times 256}$. Instance segmentation is achieved by first sampling K instance candidates $L^c = \{1, 2, ..., K\}$ and then predicting the score that each point belongs to an instance candidate, dubbed as candidate assignments $W \in \mathbb{R}^{N\times K}$. Each row in W contains the scores that a point is assigned to each of the K candidates. Redundant candidates are further eliminated according to a mask $M \in \{0, 1\}^{1\times K}$ yielded by a suppression network, with each dimension specifying if a candidate is masked / removed. The final grouping label for each point is estimated by taking the one with the maximal score among all unmasked candidates.

3.1. Feature Learning for Point clouds

Instance segmentation is relevant to both semantic information and geometric information. For example, objects with different semantic labels can be easily identified as different instances. But semantic information cannot be used to distinguish objects at the same category. In this case, geometric information, e.g. object bounding boxes and object centroids, can assist instance identification. In fact, this kind of information is also widely exploited to guide 2D instance segmentation [3, 11]. In our method, we jointly learn the centroid-aware and semantic-aware information for 3D

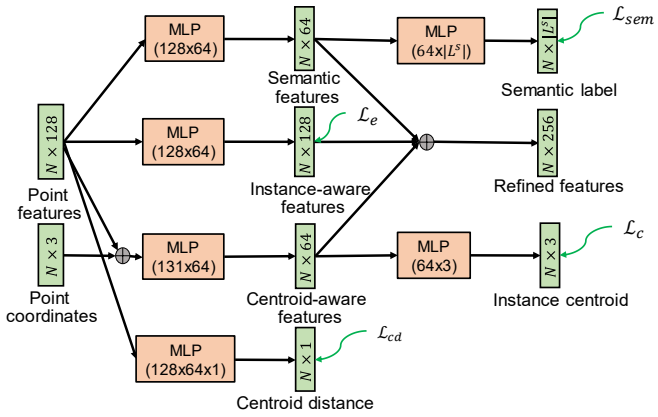

Figure 3. Feature refinement module. Point features are refined by incorporating semantic features, centroid-aware features, and instance-aware features. We also predict centroid distance between the predicted instance centroid and its ground truth counterpart. Note that $\oplus$ denotes feature concatenation, while $|L^s|$ counts the number of semantic labels. Supervision losses for different features are marked with $\curvearrowleft$.

instance segmentation.

We firstly extract point features with an existing point cloud backbone, e.g. PointNet [24], as shown in Fig. 2. Then point features are fed to four branches to jointly learn semantic features (the top branch), instance-aware features (the second branch), centroid-aware features (the third branch), and centroid distance between the centroid prediction and ground truth (the bottom branch) with four separated multi-layer perceptron (MLP) as illustrated in Fig. 3. Semantic features are learned by enforcing a semantic loss $\mathcal{L}_{sem}$, which can help distinguish objects with different semantic labels. Instance-aware features are learned by minimizing an embedding loss $\mathcal{L}_e$ such that features of points in a same instance should be close. Centroid-aware features are trained by optimizing the distance $\mathcal{L}_c$ between predicted instance centroids and their ground truth. These instance centroids help differentiate objects according to their positions. Distance $\mathcal{L}_c$ is further estimated by minimizing a centroid distance loss $\mathcal{L}_{cd}$, which can act as a regularizer. Then we concatenate features from the top three branches as refined features F^r. All loss functions are detailed in Sec. 3.5.

3.2. Instance Candidate Sampling

In our method, we sample a set of instance candidates L^c as instance representatives, which are used to group points. However, it is difficult to determine instance candidates without knowing instance segmentation. An intuition is to generate plenty enough instance candidates covering all instances and different candidates should have inter-candidate distance as large as possible. We empolyed furthest point sampling proposed in PointNet++ [25] for this task. Basically, furthest point sampling calculates the distance between the sampled points and each of the other points, and adds the point with the largest distance to the sampling set. This process iterates until enough candidates are sampled. In our experiment, we use Euclidean distance between instance-aware features of points as point distance.

3.3. Candidate Suppression

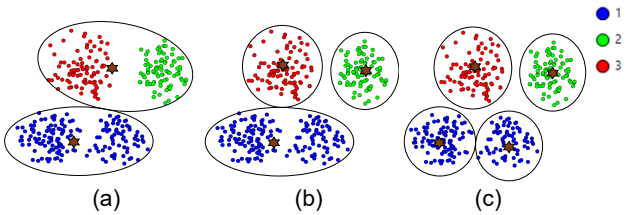

Figure 4. Influence of the number of instance candidates. Stars mark different candidates, and black circles give their member points. (a) A candidate is missing, (b) the correct number of candidates is sampled, (c) a redundant candidate is generated.

The number of sampled candidates has great impacts on the performance as shown in Fig. 4. For example, missing a candidate reduces the number of instances, thus decreasing the recall (see Fig. 4 (a)). In contrast, redundant candidates deteriorate the grouping accuracy, as they nibble points of other candidates, see blue points in Fig. 4 (c).

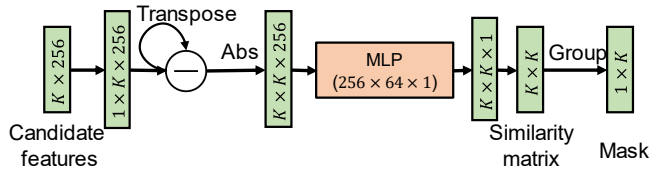

Figure 5. The suppression module. Symbol ⊖ denotes subtraction of features.

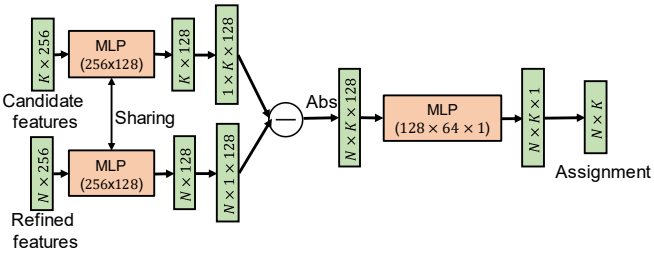

Figure 6. The assignment module.

In our method, we set the candidate number K to a large enough number so that it can cover all instances in most times. Unfortunately, this will greatly increase redundant candidates. To circumvent candidate redundancy, we introduce a suppression module as shown in Fig. 5 to predict a candidate mask M. Suppression module firstly calculates absolute differences between feature F^r_j, F^r_k of any two candidates $j, k \in L^c$, then feeds the result to a two-layer MLP ($128 \times 64 \times 1$) to estimate a similarity matrix $S \in \mathbb{R}^{K \times K}$ as shown in Fig. 5. The similarity matrix S is then binarized to $0, 1$, with 1 denoting two candidates are from a same instance.

Grouping candidates from a same instance can be seen as a problem to find all connected components of a graph if we treat similarity matrix S as an adjacent graph representation between any two candidates. Power of a matrix can be used to solve this problem. The intuition is that a $k-$th power of a connection matrix represents the connected components with less than k hops. If k is large enough, power of a matrix will find all connected components as explained in [26]. In our experiment, we compute a $32-$th power of similarity matrix S, and each row of the result matrix denotes the connectivity of a candidate to the others. For each group of candidates, we keep one candidate and mask the others as redundant candidates. Candidate grouping is non-differential, and the suppression module is learned by minimizing a similarity loss in Sec. 3.5.

3.4. Instance Assignment

We formulate instance grouping as an assignment of points X to different instance candidates L^c. In this paper, we propose a dedicated assignment module to learn the assignment as shown in Fig. 6. The assignment network firstly encodes features with a one-layer MLP (256×128), then takes the absolute differences between candidate features and point features, finally estimates assignment scores W with a two-layer MLP ($128 \times 64 \times 1$). Then assignment scores are masked with predicted mask M in Fig. 5 by $W - \alpha(1 - M)$ with broadcasting, where α is set to a large value to eliminate redundant candidates. Final instance segmentation can be obtained by labeling each point with the candidate of the highest score.

3.5. Objective Functions

The whole network is learned by optimizing an objective combing semantic loss $\mathcal{L}_{sem}$, instance centroid loss $\mathcal{L}_c$, centroid distance loss $\mathcal{L}_{cd}$, embedding loss $\mathcal{L}_e$, instance grouping loss $\mathcal{L}_g$, and candidate similarity loss $\mathcal{L}_{sim}$:

$$\begin{aligned}\mathcal{L} = \quad & w_{sem} \cdot \mathcal{L}_{sem} + w_c \cdot \mathcal{L}_c + w_{cd} \cdot \mathcal{L}_{cd} + w_e \cdot \mathcal{L}_e \\ & + w_g \cdot \mathcal{L}_g + w_{sim} \cdot \mathcal{L}_{sim},\end{aligned} \tag{1}$$

where w_* denotes balancing weights for different loss terms. Figs. 2 and 3 show where these losses are enforced during training.

Semantic loss. Semantic loss $\mathcal{L}_{sem}$ calculates the cross entropy loss between predicted semantic labels and the ground truth labels.

Centroid loss. Instance centroid loss $\mathcal{L}_c$ is defined as the distance between predicted instance centroid z_j and the ground truth centroid z_j^*:

$$\mathcal{L}_c = \sum_{j=1}^{N} \|z_j - z_j^*\|_2. \tag{2}$$

Centroid distance loss. Centroid distance loss $\mathcal{L}_{cd}$ is computed as the following:

$$\mathcal{L}_{cd} = \sum_{j=1}^{N} \|d_j - \|z_j - z_j^*\|_2\|_2, \tag{3}$$

where d_j is the predicted centroid distance.

Similarity loss. The suppression module is learned with a similarity loss $\mathcal{L}_{sim}$ enforced on similarity matrix S. Basically, this loss uses a binary cross entropy to measure if two candidates are from the same instance. The ground truth is easy to attain as instance labels are given.

Instance grouping loss. Given predicted candidate assignment and the ground truth instance label of each point, candidates and instances may be in different orders and have different numbers (see Fig. 7 (a) for an example). Because instance labels only specify which points are in a same group and have no specific meaning, the order of instances can be random. This randomness makes it tough to directly optimize instance segmentation accuracy.

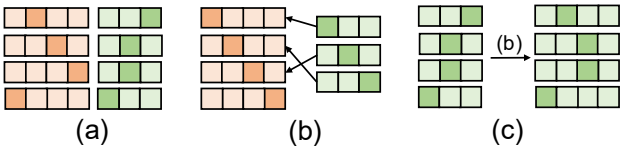

Figure 7. A matching example between candidates and instance labels. (a) Predicted candidate assignment (left) and instance labels (right) for a list of points, (b) the optimal mapping from instance labels (right) to candidates (left), (c) instance labels (left) in (a) is remapped according to (b). Candidates and instance labels are denoted by the index with the densest color.

To tackle this problem, we propose to map instance labels L^i to candidates L^c. An example is illustrated in Fig. 7. We firstly calculate the optimal mapping, namely $\{1 \rightarrow 1, 2 \rightarrow 3, 3 \rightarrow 2\}$, as shown in Fig. 7(b). Then the original instance labels $\{3, 2, 2, 1\}$ are mapped to $\{2, 3, 3, 1\}$ in Fig. 7(c) so that remapped instance labels and candidates will have a consistent order. The optimal mapping is obtained by minimizing the cost of the optimal matches:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{|L^i|} \sum_{k=1}^{|L^c|} b_{j,k} \cdot cost(j,k), \\ \text{s.t.} \quad & \sum_{j=1}^{|L^i|} b_{j,k} = 1, \forall k = 1, ..., |L^c|, \\ & \sum_{k=1}^{|L^c|} b_{j,k} <= 1, \forall j = 1, ..., |L^i|, \end{aligned} \tag{4}$$

where $b_{j,k}$ is a binary variable denoting if j and k are a match, $|\cdot|$ counts the number of instances or candidates, and $cost(j,k)$ measures how well a match is. The first constraint ensures each instance label is assigned to a candidate, while the second constraint guarantees at most one candidate is matched to instance labels. In our case, the matching should maximize instance segmentation accuracy, thus the cost is defined as one minus the intersection over union between predicted candidate assignments and instance labels:

$$cost(j,k) = 1.0 - \frac{\sum_{m=1}^{N} \mathbb{1}(y_m^c = l_j^c) \wedge \mathbb{1}(y_m^i = l_k^i)}{\sum_{m=1}^{N} \mathbb{1}(y_m^c = l_j^c) \vee \mathbb{1}(y_m^i = l_k^i)}, \tag{5}$$

where y_m^c is the predicted candidate assignment of point m, y_m^i denotes the ground truth instance label that point belongs to, and $\mathbb{1}(\cdot)$ tests if a given value is true.

This assignment problem can be solved by the Hungarian algorithm [17] or integer programming. In our implementation, we use the linear assignment solver in SciPy. As the number of instances is small for each input (no more than 50 instances), the problem can be solved efficiently.

After obtaining the mapping between instance labels and instance candidates, we can optimize instance segmentation by minimizing the cross entropy between the predicted probability and the optimally assigned labels.

Embedding loss. Though our method can be learned with the proposed instance grouping loss, instance-aware features are still required to ensure good candidates can be sampled in the candidate sampling step. This is because the sampling step is non-differential, therefore instance grouping loss cannot be propagated back to guide the embedding learning. Following [22, 29], the embedding loss $\mathcal{L}_e$ is defined as follows:

$$\mathcal{L}_e = \mathcal{L}_{pull} + \mathcal{L}_{push} + w_{reg} \cdot \mathcal{L}_{reg}, \tag{6}$$

where $\mathcal{L}_{pull}$ is a pulling loss, $\mathcal{L}_{push}$ is a pushing loss, and $\mathcal{L}_{reg}$ is a regularization term. Pulling loss tries to minimize distances between instance-aware feature f_j^i and the mean feature of its owner instance m_k, while pushing loss maximizes inter-instance distance of the mean feature m_k, m_o of two different instances:

$$\mathcal{L}_{pull} = \frac{1}{|L^i|}\sum_{k=1}^{|L^i|}\frac{1}{N_k}\sum_{j=1}^{N_k}\max\left(0, \|m_k - f_j^i\|_2 - \delta_1\right)^2, \tag{7}$$

$$\mathcal{L}_{push} = \frac{1}{|L^i|(|L^i|-1)}\sum_{k=1}^{|L^i|}\sum_{o=1,o\neq k}^{K}\max\left(0, \delta_2 - \|m_k - m_o\|_2\right)^2, \tag{8}$$

where N_k is the number of points in instance k, $\delta_1 = 0.5, \delta_2 = 1.5$ are two margins to clip the loss. Loss $\mathcal{L}_{reg}$ helps restrict instance-aware features to be finite by encouraging small values:

$$\mathcal{L}_{reg} = \frac{1}{|L^i|}\sum_{k=1}^{|L^i|}\|m_k\|_2. \tag{9}$$

4. Experiments

4.1. Datasets and Evaluation Metrics

Datasets. We conduct the evaluation on Stanford 3D Indoor Semantic Dataset (S3DIS) [1], which is widely used in 3D instance segmentation [22, 28, 29]. S3DIS contains 3D scans collected in 6 areas. Following the standard data split, area 5 is used for testing and other areas are used for training. We also evaluate our method on SceneNN [13], which is an indoor scene dataset scanned at room scale. We follow data splits of JSIS [22] for training and testing. As the registered mesh contains many outliers, we clean them up by removing outlier instances with less than 200 points.

Evaluation. Our method mainly aims at instance segmentation of 3D point clouds. The widely used metrics for instance segmentation are mean class precision (mPrec) and mean class recall (mRec) with an intersection over union (IoU) larger than 0.5 between the prediction and the ground truth. We also report the mean class coverage (mCov) and mean class weighted coverage (mwCov) following [29, 32]. Coverage measures the instance-wise IoU between the ground truth and its matching predictions. Given a list of ground truth regions $\mathcal{G}$ and prediction regions $\mathcal{P}$ of a specific category, mCov and mwCov are calculated as:

$$mCov(\mathcal{G}, \mathcal{P}) = \frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}}\max_{p\in\mathcal{P}} IoU(g, p), \tag{10}$$

$$\begin{aligned} mwCov(\mathcal{G}, \mathcal{P}) &= \sum_{g\in\mathcal{G}} w_g \max_{p\in\mathcal{P}} IoU(g, p), \\ w_g &= \frac{|g|}{\sum_{g'\in\mathcal{G}}|g'|}, \end{aligned} \tag{11}$$

where $|\cdot|$ counts the number of points in a list, and $IoU(\cdot,\cdot)$ calculates IoU between two point sets.

Implementation. We implemented the algorithm in PyTorch. The network was trained with an Adam optimizer with an initial learning rate of 0.002. During training and testing, we segmented the scene into point clouds with 4096 points following SGPN [28]. Our algorithm predicts instance segmentation for each point cloud and then merges them with the BlockMerging algorithm proposed in SGPN as the final result.

4.2. Comparisons to the-state-of-the-art

Existing methods. We compare our method with the-state-of-the-art methods in 3D instance segmentation, including SGPN [28], ASIS [29], and JSIS [22]. Results of these methods are generated with their released codes. As the network backbone has great influences on final results, we evaluate these methods with a same backbone, i.e. PointNet [24]. Wang et. al [29] only released the code with a PointNet++ [25] backbone. To enable a fair comparison, we implement ASIS with a PointNet backbone. Results of our method with a PointNet++ backbone are also reported to facilitate a comparison with the state-of-the-art results of ASIS. For the evaluation on S3DIS dataset, we assess the performance with released models of JSIS (PN) and ASIS (PN2), and the finetuned SGPN (PN) model, where PN and PN2 denote PointNet and PointNet++, respectively. For the evaluation on other datasets and ASIS (PN) model, we train the network from scratch with the training dataset.

Results on S3DIS dataset. In our experiments, we found 3D instance segmentation sensitive to different samplings of a point cloud. In Fig. 8, we compare the performance of different methods and their variances on 6 sampling of point clouds in the testing dataset. Our method has consistently better results on the average of all metrics. Though results of all methods oscillate, ours has smaller variances in both mCov and mwCov metrics. To eliminate the influences of

Table 1. Comparison of mCov metric on S3DIS dataset. #inst counts the number of instances of a category.

		ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
#inst	2346	76	68	343	3	74	52	127	154	258	11	217	42	921
SGPN (PN)	0.439	0.805	0.863	0.533	**0.025**	0.035	**0.693**	**0.615**	**0.492**	0.462	**0.202**	0.288	0.365	0.322
JSIS (PN)	0.394	0.827	0.836	0.534	0.000	0.029	0.491	0.101	0.479	0.714	0.150	**0.456**	0.082	0.421
ASIS (PN)	0.422	0.864	**0.883**	0.603	0.000	0.036	0.619	0.130	0.461	**0.649**	0.099	0.389	**0.403**	**0.355**
Ours (PN)	**0.444**	**0.869**	0.872	**0.654**	0.000	**0.083**	0.606	0.332	0.454	0.635	0.136	0.406	0.400	0.327
ASIS (PN2)	0.446	**0.869**	**0.883**	0.605	0.000	0.019	0.602	0.100	0.467	0.680	0.231	0.407	0.593	0.341
Ours (PN2)	**0.496**	0.860	0.863	**0.695**	0.000	**0.075**	**0.624**	**0.144**	**0.541**	**0.768**	**0.326**	**0.518**	**0.697**	**0.392**

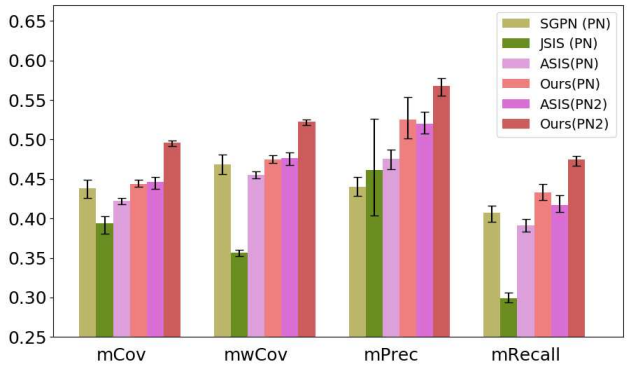

Figure 8. Quantitative comparison on S3DIS dataset. The height of each color bar shows the average of each metric, and the range of a black bar line denotes the difference between the maximal value and the minimal value of each metric.

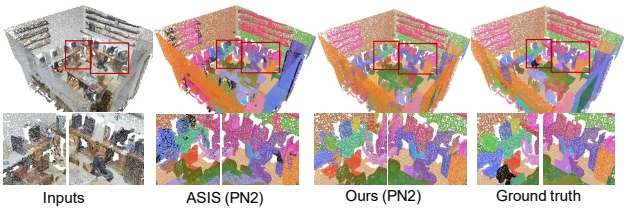

Figure 9. Qualitative comparison with ASIS (PN2). Colors help mark different instances. Notice how cluttered objects in the highlighted regions are distinguished.

these variances, average results of 6 samplings from a same input are reported. In Tab. 1, we compare with the state-of-the-art methods on mCov metric. Results demonstrate our method is comparable to other methods when using a PointNet backbone, and can achieve the best results on $10/13$ categories if a PointNet++ backbone is used. Results on the other metrics are provided in the supplemental. Qualitative results are demonstrated in Fig. 9 and Fig. 10. We match predicted instances with their ground truth by our assginment algorithm in Sec. 3.5 so that same instances will have the same color. We can achieve better instance segmentation with both PointNet backbone and PointNet++ backbone, especially in cluttering regions, e.g. nearby chairs, long tables and a bunch of objects in the highlighted regions in Fig. 9 and Fig. 10.

Results on SceneNN dataset. We also evaluate our method on SceneNN dataset. Results are reported in Tab. 2 on 10 selected categories of NYU-40 labels (including wall, floor, bed, chair, table, door, desk, fridge, television, and prop). All methods perform badly because reconstructed scenes in SceneNN dataset have many outliers and flying shapes. Also, the dataset is quite small.

Table 2. Comparison results on SceneNN dataset.

Methods	mCov	mwCov	mPrec	mRecall
JSIS (PN)	0.095	0.112	0.029	0.039
ASIS (PN)	0.123	0.133	**0.080**	**0.095**
Ours (PN)	**0.124**	**0.134**	**0.080**	0.066

Table 3. Ablation study on different modules.

	w/o cen.	w/o sem.	w/o emb.	w/o c.d.	w/o sup.	Ours (PN)
mCov	0.430	0.435	0.400	0.432	0.350	**0.444**
mwCov	0.463	0.467	0.432	0.466	0.383	**0.475**
mPrec	0.491	0.495	0.458	0.504	0.392	**0.526**
mRecall	0.405	0.420	0.371	0.425	0.319	**0.433**

4.3. Ablation Study

To evaluate the effectiveness of different modules, we investigate the influence of centroid-aware features (w/o cen.), semantic features (w/o sem.), instance-aware features (w/o emb.), and centroid distance branch (w/o c.d.) by removing the corresponding branches from feature refining module. We also study how important the suppression module is by removing the masking process and the similarity loss (w/o sup.).

In Tab. 3, we show results on alternative designs. We can see fusion of different kinds of features boosts the performance, especially instance-aware features. This is because the candidate sampling step relies on instance-aware features to get reliable candidates. Though the centroid branching does not contribute to refining feature F^r, it still has impacts on the results. This may credit to the regularization of multi-task learning. Suppression module is also important as it helps eliminate redundant candidates as discussed in Fig. 4. In Fig. 11, influences of the selection of K is evaluated, and our method is robust to different choices of K during inference. This is attributed to the learned suppression module. In default, we use the maximal number of instances in all inputs of the training dataset as K.

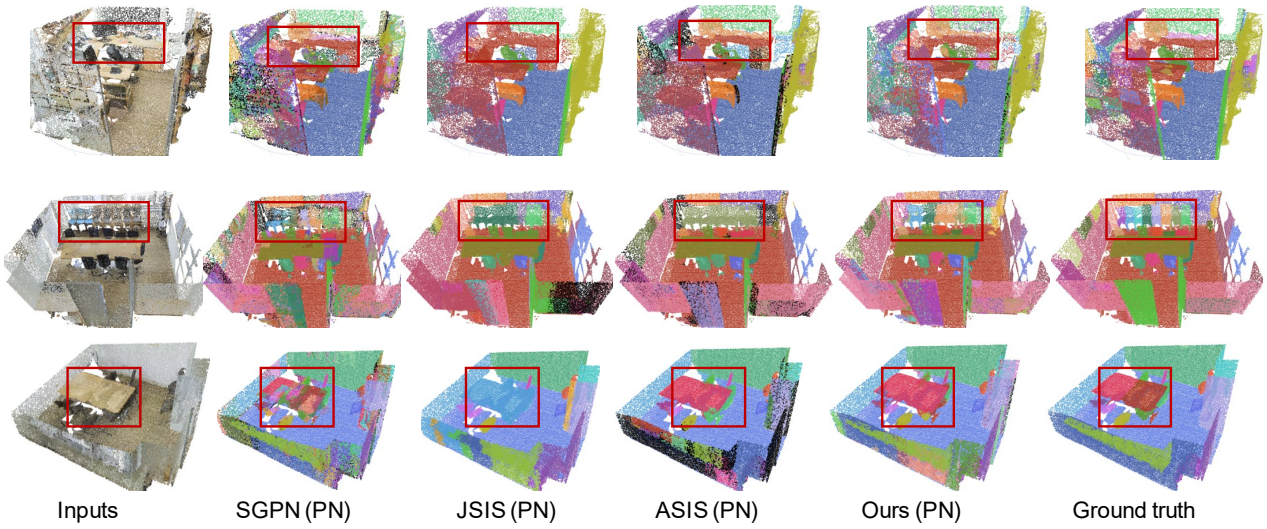

Figure 10. Instance segmentation results on S3DIS testing dataset. Notice how our method is able to distinguish different instances of the same categories, e.g. chairs.

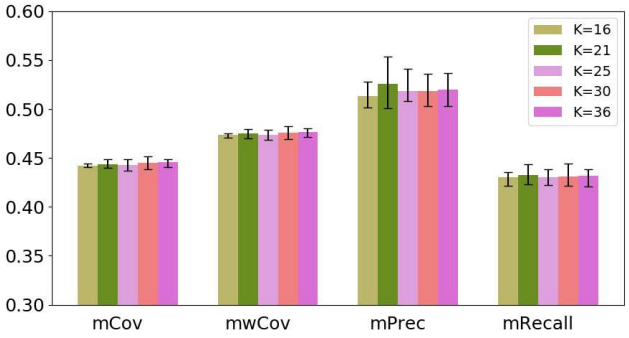

Figure 11. Influences of the selection of K during inference.

Table 4. Comparison on inference time. The time is measured and averaged on S3DIS testing dataset (area 5).

Methods	Network (ms)	Grouping (ms)	Overall (ms)
SGPN (PN)	47.0	8774.3	8821.3
ASIS (PN)	160.7	241.6	402.3
JSIS (PN)	14.5	5048.2	5062.7
Ours (PN)	34.3	0.0	34.3

4.4. Time Analysis

We compare the computation time in Tab. 4. The time is measured on a computer with a nVidia GTX 1080 GPU and an Intel i7-6850K CPU on the testing set of S3DIS. Note that data loading time is not counted. We measure the grouping time costed on grouping merging for SGPN, mean-shift clustering for ASIS, and both mean-shift clustering and multi-value CRF label refinement for JSIS. We can see the grouping stage is the most computational part in these methods. In contrast, our method can be executed in an end-to-end way, and the network inference runs quite fast. Therefore our method only took $34ms$ to process a point cloud with 4096 points. In Tab. 4, our algorithm is about 257x faster than SGPN, 11x faster than ASIS, and 147x faster than JSIS. All methods are evaluated with a PointNet backbone. Our computational boosting is owing to the end-to-end instance assignment design.

4.5. Limitations and Discussions

Our method uses furthest point sampling to generate instance candidates. This process is non-differential and highly depends on instance-aware features. If poor candidates are sampled, it will make it more difficult for redundancy removal and instance assignment. In the future, we will incorporate the estimated centroid distance to guide the candidate sampling.

5. Conclusion

We present an end-to-end approach for 3D instance segmentation. Different from the detection-free methods and detection-based methods, we directly generate a set of instance candidates, and cast instance grouping as a candidate assignment problem for each point. Redundant candidates are further masked with a suppression module. Experimental results on S3DIS dataset and SceneNN dataset demonstrate the efficiency and effectiveness of our method. Our framework is not restricted to 3D point cloud, and can be easily extended to handle 2D instance segmentation as well.

Acknowledgments. This research is supported by MoE Tier-2 Grants (2016-T2-2-065, 2017-T2-1-076) of Singapore and NTU Data Science and Artificial Intelligence Research Center (DSAIR) (No. M4082285). It is also partially supported by Monash FIT Start-up Grant and NSFC (No. 61802362).

References

[1] Iro Armeni, Ozan Sener, Amir Roshan Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1534–1543, 2016. 6

[2] Anurag Arnab and Philip H. S. Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 879–888, 2017. 2

[3] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2858–2866, 2017. 2, 3

[4] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *CoRR*, abs/1708.02551, 2017. 1, 2, 3

[5] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pages 534–549, 2016. 2

[6] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3992–4000, 2015. 2

[7] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3150–3158, 2016. 1, 2

[8] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. Semantic instance segmentation via deep metric learning. *CoRR*, abs/1703.10277, 2017. 2

[9] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII*, pages 297–312, 2014. 2

[10] Zeeshan Hayder, Xuming He, and Mathieu Salzmann. Boundary-aware instance segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 587–595, 2017. 2

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988, 2017. 1, 2, 3

[12] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of RGB-D scans. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4421–4430, 2019. 1, 3

[13] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. SceneNN: A scene meshes dataset with annotations. In *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*, pages 92–101, 2016. 6

[14] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: From edges to instances with multicut. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 7322–7331, 2017. 2

[15] Artem Komarichev, Zichun Zhong, and Jing Hua. A-CNN: annularly convolutional neural networks on point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 7421–7430, 2019. 3

[16] Shu Kong and Charless C. Fowlkes. Recurrent pixel embedding for instance grouping. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9018–9028, 2018. 2, 3

[17] H. W. Kuhn. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258, December 1956. 5

[18] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R. Oswald. 3d instance segmentation via multi-task metric learning. *CoRR*, abs/1906.08650, 2019. 1, 3

[19] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided CNN with spherical kernels for 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 9631–9640, 2019. 3

[20] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. SGN: sequential grouping networks for instance segmentation. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 3516–3524, 2017. 2

[21] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8837–8845, 2019. 1, 2, 3

[22] Quang-Hieu Pham, Duc Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. JSIS3D: joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8827–8836, 2019. 1, 2, 3, 6

[23] Pedro H. O. Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1990–1998, 2015. 2

[24] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. pages 77–85, 2017. 3, 4, 6

[25] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5105–5114, 2017. 3, 4, 6

[26] Steven Skiena. *Implementing discrete mathematics - combinatorics and graph theory with Mathematica*. Addison-Wesley, 1990. 4

[27] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3887–3896, 2018. 3

[28] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: similarity group proposal network for 3d point cloud instance segmentation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2569–2578, 2018. 1, 2, 3, 6

[29] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4096–4105, 2019. 1, 2, 3, 6

[30] 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 *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 6737–6746, 2019. 3

[31] 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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3947–3956, 2019. 1, 3

[32] Wei Zhuo, Mathieu Salzmann, Xuming He, and Miaomiao Liu. Indoor scene parsing with instance segmentation, semantic labeling and support relationship inference. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6269–6277, 2017. 6