Scalable SoftGroup for 3D Instance Segmentation on Point Clouds

Thang Vu, *Member, IEEE*, Kookhoi Kim, *Member, IEEE*, Thanh Nguyen, *Member, IEEE*, Tung M. Luu, *Member, IEEE*, Junyeong Kim, *Member, IEEE*, and Chang D. Yoo, *Senior Member, IEEE*

***Abstract*—This paper considers a network referred to as SoftGroup for accurate and scalable 3D instance segmentation. Existing state-of-the-art methods produce hard semantic predictions followed by grouping instance segmentation results. Unfortunately, errors stemming from hard decisions propagate into the grouping, resulting in poor overlap between predicted instances and ground truth and substantial false positives. To address the abovementioned problems, SoftGroup allows each point to be associated with multiple classes to mitigate the uncertainty stemming from semantic prediction. It also suppresses false positive instances by learning to categorize them as background. Regarding scalability, the existing fast methods require computational time on the order of tens of seconds on large-scale scenes, which is unsatisfactory and far from applicable for real-time. Our finding is that the k-Nearest Neighbor (k-NN) module, which serves as the prerequisite of grouping, introduces a computational bottleneck. SoftGroup is extended to resolve this computational bottleneck, referred to as SoftGroup++. The proposed SoftGroup++ reduces time complexity with octree k-NN and reduces search space with class-aware pyramid scaling and late devoxelization. Experimental results on various indoor and outdoor datasets demonstrate the efficacy and generality of the proposed SoftGroup and SoftGroup++. Their performances surpass the best-performing baseline by a large margin ($6\% \sim 16\%$) in terms of AP_{50}. On datasets with large-scale scenes, SoftGroup++ achieves a $6\times$ speed boost on average compared to SoftGroup. Furthermore, SoftGroup can be extended to perform object detection and panoptic segmentation with nontrivial improvements over existing methods.**

***Index Terms*—Point clouds, point grouping, octree grouping, instance segmentation, object detection, panoptic segmentation.**

I. Introduction

With the rapid evolution of 3D sensors and the widespread availability of large-scale 3D datasets, there has been a notable surge in interest towards achieving a deeper understanding of 3D scenes. Instance segmentation on point clouds is a 3D perception task that serves as the foundation for a wide range of applications such as autonomous driving, virtual reality, and robot navigation. Instance segmentation processes the point clouds to output a category and an instance mask for each detected object.

This paper proposes a network referred to as SoftGroup for accurate and scalable point cloud instance segmentation. To attain high accuracy, existing state-of-the-art methods [1], [2], [3] consider 3D instance segmentation as a bottom-up pipeline. These methods first predict the point-wise semantic labels and center offset vectors and then group points of the same predicted labels with small geometric distances into instances. These grouping algorithms are performed based on hard semantic predictions, where a point is associated with a single class. Unfortunately, objects frequently exhibit local ambiguity, and semantic segmentation tends to be noisy, often resulting in varied predictions for different parts of the same object. Consequently, employing hard semantic predictions for the purpose of instance grouping gives rise to two challenges: (1) the low overlap between the predicted instances and the ground truth, and (2) the introduction of additional false-positive instances stemming from incorrect semantic delineations. Fig. 1 shows a visualization example. Here, in the semantic prediction results, some parts of `cabinet` are wrongly predicted as `other furniture`. When hard semantic predictions are used to perform grouping, the semantic prediction error is propagated to instance prediction. As a result, the predicted `cabinet` instance has low overlap with the ground truth, and the `other furniture` instance is a false positive. The proposed SoftGroup overcomes these problems by considering soft semantic scores to perform grouping instead of hard one-hot semantic predictions. The intuition of SoftGroup is illustrated in Fig. 2. Our observation is that the object parts with incorrect semantic predictions still have reasonable scores for the true semantic class. SoftGroup relies on a score threshold instead of the maximum argument value in determining which category the object belongs to. Grouping on the soft semantic scores produces a more accurate instance of the true semantic class. The instance with incorrect semantic prediction will be suppressed by learning to categorize it as background. To this end, we treat an instance proposal (i.e., output of grouping) as either a positive or negative sample depending on its maximum Intersection over Union (IoU) with the ground truth. This is followed by a top-down refinement stage to refine the positive

Manuscript received 21 October 2022; revised 21 August 2023; accepted 11 October 2023. Date of publication 20 October 2023; date of current version 6 March 2024. This work was supported in part by the Institute for Information Communications Technology Planning Evaluation (IITP) grant funded by the Korea Government (MSIT) under Grant 2021-0-01381, in part by the Development of Causal AI through Video Understanding, in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by Korea government (MSIT) under Grant 2019-0-01371, and in part by the Development of brain-inspired AI with human-like intelligence. Recommended for acceptance by E. Kalogerakis. *(Corresponding author: Chang D. Yoo.)*

Thang Vu, Kookhoi Kim, Thanh Nguyen, Tung M. Luu, and Chang D. Yoo are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: thangvubk@kaist.ac.kr; rlarnrghlapz@kaist.ac.kr; thanhnguyen@kaist.ac.kr; tungluu2203@kaist.ac.kr; cd_yoo@kaist.ac.kr).

Junyeong Kim is with the Department of AI, Chung-Ang University, Seoul 06974, South Korea (e-mail: junyeongkim@cau.ac.kr).

Digital Object Identifier 10.1109/TPAMI.2023.3326189

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

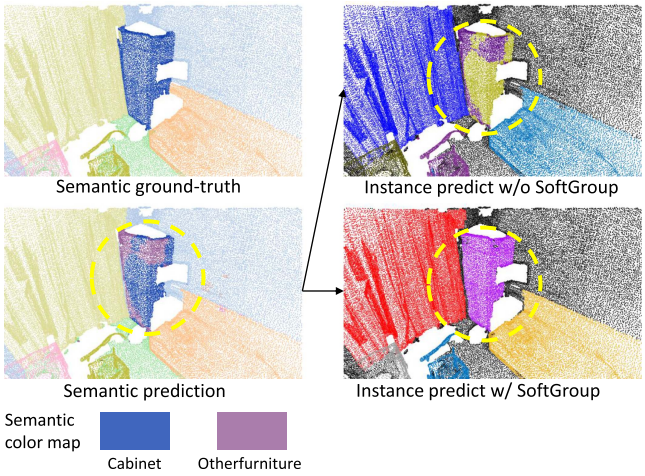


Fig. 1. *Instance segmentation with and without SoftGroup from the same semantic prediction results.* The last row shows the palette for semantic predictions only. Instance predictions are illustrated by different random colors for different objects. In the semantic prediction results, some regions of `cabinet` are wrongly predicted as `other furniture`. Without SoftGroup, these errors are propagated to instance prediction. SoftGroup addresses this problem and produces more accurate instance masks.

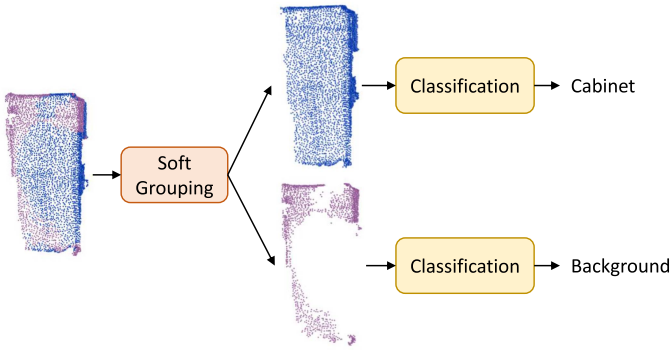
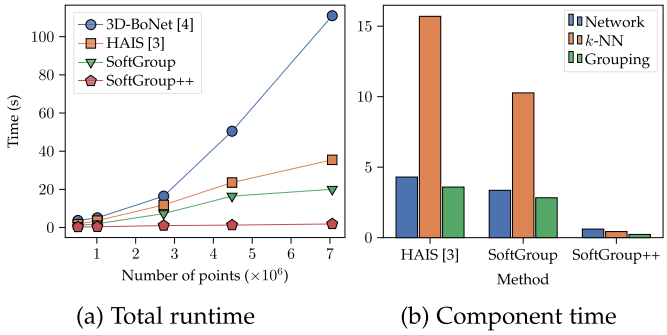


Fig. 2. The `cabinet` in Fig. 1 is extracted to illustrate the high-level pipeline of our method. The soft grouping module is based on soft semantic scores to output a more accurate instance (the upper one). The classifier processes each instance and suppresses the instance from wrong semantic prediction (the lower one).

samples and suppress the negative samples. As shown in Fig. 1, SoftGroup can produce accurate instance masks from imperfect semantic prediction.

SoftGroup is further extended for scalability to maintain fast inference speed on large-scale scenes. This advanced architecture will henceforth be referred to as SoftGroup++. Fig. 3(a) shows that the runtimes of existing methods grow quickly as the number of points increases. For instance, existing methods require ∼20 s to ∼50 s to process a scene with ∼4.5 M points. The processing time of each network component for this scene is further analyzed, as provided in Fig. 3(b). The results reveal that k-NN is the computational bottleneck, leading to the quick growth of inference time with regard to input size in HAIS [3] and SoftGroup. These methods perform vanilla k-NN that requires measuring pair-wise distances of all points leading to time complexity of $\mathcal{O}(n^2)$ which is not scalable.

To enable fast inference on large-scale scenes, SoftGroup++ introduces an inference algorithm with low time complexity



(a) Total runtime (b) Component time

Fig. 3. (a) *Total runtime.* The runtimes of existing methods significantly increase as the number of points increases. (b) *Component time.* We measure the component time of processing a large scene of ∼4.5 M points. The measurement exposes k-NN as a computational bottleneck.

and search space. SoftGroup++ performs octree k-NN instead of vanilla k-NN to reduce time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. To effectively parallelize octree k-NN on GPU, a strategy to unfold the recursive structure of octree is presented such that tree traversal can be performed based on a simple arithmetic procedure. To reduce search space, SoftGroup++ relies on class-aware pyramid scaling and late devoxelization, which respectively perform adaptive downsampling on the backbone output features and delay the conversion from voxels to points until the end of the model. Fig. 3(a) shows that SoftGroup++ still maintains a fast inference speed as input size increases. The computational bottleneck of k-NN is also addressed, as presented in Fig. 3(b).

Both SoftGroup and SoftGroup++ are conceptually simple and easy to implement. For instance segmentation task, they outperform the previous state-of-the-art method by a large margin, ranging from 6% to 16% in terms of AP_{50} on different indoor and outdoor datasets. SoftGroup++ shows scalability advantages on S3DIS with large-scale scenes with 6× inference speed up. The versatility of SoftGroup is also demonstrated by the extension to object detection and panoptic segmentation with nontrivial improvements over existing methods.

Differences From Our Conference Paper: This manuscript is a significant extension of our conference version, which was previously published in [5]. In this updated version, we further investigate the scalability of recent 3D instance segmentation methods and reveal the computational bottleneck of k-NN. To efficiently process large-scale scenes, we propose SoftGroup++ which aims to reduce time complexity and search space. Octree k-NN replaces vanilla k-NN to reduce the time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. Class-aware pyramid scaling and late devoxelization reduce the search space and runtime of intermediate network components. Our proposed method is extensively benchmarked on various tasks and datasets demonstrating its versatility and generality. The source code and trained models are available at https://github.com/thangvubk/SoftGroup.

II. Related Work

Deep Learning on 3D Point Clouds: Point cloud representation is a common data format for 3D scene understanding.

To process point clouds, early methods [6], [7], [8], [9] extract hand-crafted features based on statistical properties of points. Recent deep learning methods learn to extract features from points. PointNet-based methods [10], [11] propose to process points through shared Multi-Layer Perceptron (MLP) and then aggregate regional and global features from symmetric functions, such as max-pooling. Convolution methods are actively explored for point cloud processing. Continuous convolution methods [12], [13], [14], [15] learn the kernels which are associated with the spatial distribution of local points. Discrete convolution methods [16], [17], [18], [19], [20], [21] learn the kernels which are regular grids obtaining from point quantization. Transformers [22], [23], [24] and graph-based methods [25], [26], [27] are also proposed to address the data irregularity of point clouds.

Proposal-Based Instance Segmentation: Proposal-based methods consider a top-down strategy that generates region proposals and then segments the object within each proposal. Existing proposal-based methods for 3D point clouds are highly influenced by the success of Mask-R CNN [28] for 2D images. To handle data irregularity of point clouds, Li et al. [29] propose GSPN, which takes an analysis-by-synthesis strategy to generate high-objectness 3D proposals, which are refined by a region-based PointNet. Hou et al. [30] present 3DSIS that combines multi-view RGB input with 3D geometry to predict bounding boxes and instance masks. Yang et al. [4] propose 3D-BoNet which directly outputs a set of bounding boxes without anchor generation and non-maximum suppression, then segments the object by a pointwise binary classifier. Liu et al. [31] present GICN to approximate the instance center of each object as a Gaussian distribution, which is sampled to get object candidates and then produce bounding boxes and instance masks.

Grouping-Based Instance Segmentation: Grouping-based methods rely on a bottom-up pipeline that produces per-point predictions (such as semantic maps, and geometric shifts, or latent features) then groups points into instances. Wang et al. [32] propose SGPN to construct a feature similarity matrix for all points and then group points of similar features into instances. Pham et al. [33] present JSIS3D that incorporates the semantic and instance labels by a multi-value conditional random field model and jointly optimizes the labels to obtain object instances. Lahoud et al. [34] propose MTML to learn feature and directional embedding, then perform mean-shift clustering on the feature embedding to generate object segments, which are scored according to their direction feature consistency. Han et al. [35] introduce OccuSeg which performs graph-based clustering guided by object occupancy signal for more accurate segmentation outputs. Zhang et al. [36] consider a probabilistic approach that represents each point as a tri-variate normal distribution followed by a clustering step to obtain object instances. Jiang et al. [1] propose PointGroup to segment objects on original and offset-shifted point sets, relying on a simple yet effective algorithm that groups nearby points of the same label and expands the group progressively. Chen et al. [3] extend PointGroup and propose HAIS that further absorbs surrounding fragments of instances and then refines the instances based on intra-instance prediction. Liang et al. [2] SSTNet to construct a tree network from pre-computed superpoints then traverse the tree and split nodes to get object instances.

Panoptic Segmentation: Since most of the instance segmentation methods rely on semantic segmentation to produce instance results, these methods can be naturally extended to panoptic segmentation. Zhou et al. [37] propose Panoptic-PolarNet to perform semantic segmentation and class-agnostic instance clustering on polar Bird's Eye View (BEV) representation for efficient panoptic segmentation. Gasperini et al. [38] introduce Panoster which is a learning-based solution object clustering, enabling direct derivation of panoptic results from semantic and instance predictions without post-processing. Hong et al. [39] present DS-Net with Dynamic Shift module for object clustering, followed by consensus-driven fusion module to incorporate semantic and instance results. Li et al. [40] propose Panoptic-PHNet with different improvements including fused 2D-3D backbone, transformer-based offset branch, and center grouping module for more accurate panoptic segmentation.

Efficient Point Cloud Processing: Due to data irregularity, efficient point cloud processing is crucial for various tasks, ranging from recognition to compression, and reconstruction. Riegler et al. [21] propose an octree-based convolutional network that hierarchically partitions the space to focus on relevant regions with low memory allocation and computation. Fu et al. [41] presents an octree-based attention network for point cloud compression that extends the receptive field of context and exploits features from sibling nodes and their ancestors to model the dependency of nodes in large-scale context. Xu et al. [42] introduce a Grid Graph Convolutional Network (Grid-GCN) that utilizes the advantages of volumetric models and point-based models to achieve efficient data structuring and efficient computation at the same time. Rosu et al. [43] propose LatticeNet for 3D semantic segmentation that embeds point features into a sparse permutohedral lattice for fast convolutions while keeping a low memory footprint. Lombardi et al. [44] also utilize a permutohedral lattice on hierarchical point cloud features for efficient point cloud reconstruction. Park et al. [24] proposes Fast Point Transformer with a lightweight self-attention layer and hashing-based architecture.

The proposed SoftGroup and SoftGroup++ harness the benefits of both proposal-based and grouping-based approaches synergistically. They are constructed as a two-stage pipeline, where the bottom-up stage generates high-quality object proposals by grouping on soft semantic scores, and then the top-down stage process each proposal to refine positive samples and suppress negative ones. To efficiently process large-scale scenes, SoftGroup++ proposes to reduce time complexity and search space with octree k-NN, class-aware pyramid scaling, and late devoxelization.

III. SoftGroup for Accurate Point Cloud Instance Segmentation

This subsection presents SoftGroup with a focus on accuracy. The overall architecture of SoftGroup is depicted in Fig. 4, which is divided into two stages. In the bottom-up grouping stage, the

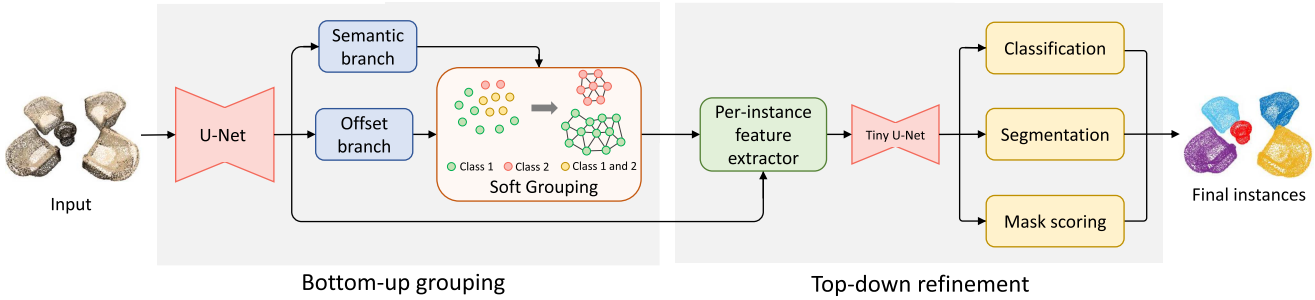


Fig. 4. *The architecture of the proposed method consists of bottom-up grouping and top-down refinement stages.* From the input point clouds, the U-Net backbone extracts the point features. Then semantic and offset branches predict the semantic scores and offset vectors, followed by a soft grouping module to generate instance proposal. The soft grouping module addresses the problem of ambiguous semantic predictions, denoted by yellow points by assigning them to possible classes based on semantic scores for instance proposal generation. The feature extractor layer extracts backbone features from instance proposals. The features for each proposal are fed into a tiny U-Net followed by the classification, segmentation, and mask scoring branches to get the final instances.

point-wise prediction network takes point clouds as the input and produces point-wise semantic labels and offset vectors. The soft grouping module processes these outputs to produce preliminary instance proposals. In the top-down refinement stage, based on the proposals, the corresponding features from the backbone are extracted and used to predict classes, instance masks, and mask scores as the final results.

A. Point-Wise Prediction Network

The input of the point-wise prediction network is a set of N points, each of which is represented by its coordinate and color. The point set is voxelized to convert points into sparse volumetric grids, which are used as the input of a U-Net style backbone [45] to obtain point features. The Submanifold Sparse Convolution [18] is adopted to implement the U-Net for 3D point clouds. The employed backbone architecture is identical to that of HAIS [3]. From the output backbone features, the inverse mapping of the input voxelization step is applied to obtain point features. Then, two branches are constructed to produce the point-wise semantic scores and offset vectors.

Semantic Branch: A semantic branch is constructed from a two-layer MLP, which learns to output semantic scores $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N\} \in \mathbb{R}^{N \times N_{\text{class}}}$ for N points over N_{class} classes. Different from existing methods [1], [3], we directly perform grouping on semantic scores without converting them into one-hot semantic predictions.

Offset Branch: In parallel with the semantic branch, we apply a two-layer MLP to learn the offset vectors $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_N\} \in \mathbb{R}^{N \times 3}$, which represents the vector from each point to the geometric center of the instance the point belongs. Based on the learned offset vectors, we shift the points to the center of the corresponding instance to perform grouping more effectively.

The cross-entropy loss and ℓ_1 regression loss are used to train the semantic and offset branches, respectively.

$$L_{\text{semantic}} = \frac{1}{N} \sum_{i=1}^{N} \text{CE}(\boldsymbol{s}_i, s_i^*), \tag{1}$$

$$L_{\text{offset}} = \frac{1}{\sum_{i=1}^{N} \mathbb{1}_{\{\boldsymbol{p}_i\}}} \sum_{i=1}^{N} \mathbb{1}_{\{\boldsymbol{p}_i\}} \|\boldsymbol{o}_i - \boldsymbol{o}_i^*\|_1, \tag{2}$$

where s^* is the semantic label, $\boldsymbol{o}^*$ is offset label representing the vector from a point to the geometric center of the instance that the point belongs to (analogous to [1], [2], [3]), and $\mathbb{1}_{\{\boldsymbol{p}_i\}}$ is the indicator function indicating whether the point $\boldsymbol{p}_i$ belongs to any instance.

B. Soft Grouping

The soft grouping module receives the semantic scores and offset vectors as the input and produces instance proposals. First, the offset vectors are used to shift points toward the corresponding instance centers. To perform grouping using the semantic scores, we define a score threshold τ to determine which semantic classes a point belongs to, allowing the point to be associated with multiple classes. Given semantic scores $\boldsymbol{S} \in \mathbb{R}^{N \times N_{\text{class}}}$, we iterate through N_{class} classes, and at each class index we slice a point subset of the whole scene that has the score (w.r.t. the class index) higher than the threshold τ. We follow [1], [3] to perform grouping on each point subset. Since all points in each subset belong to the same class, we simply traverse all the points in the subset and create the links between points having a geometric distance smaller than a grouping radius r to get the instance proposals. For each iteration, the grouping is performed on a point subset of the whole scan, ensuring fast inference. The overall instance proposals are the union of the proposals from all subsets.

We note that existing proposal-based methods [4], [30], [31] commonly consider bounding boxes as object proposals then perform segmentation within each proposal. Intuitively, the bounding box that largely overlaps the instance should have the center close to the object center. However, generating high-quality bounding box proposals in 3D point clouds is challenging since the point only exists on object surfaces. Instead, SoftGroup relies on point-level proposals which are more accurate and naturally inherit the scattered property of point clouds.

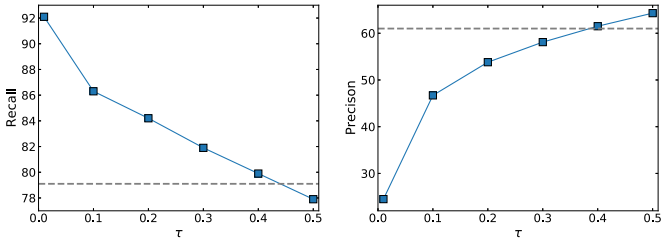


Fig. 5. *The recall and precision of semantic prediction with varying score threshold* τ. The dashed lines denote the recall and precision with hard semantic prediction.

Since the quality of instance proposals from grouping highly depends on the quality of semantic segmentation, we quantitatively analyze the impact of τ on the recall and precision of semantic predictions. The recall and precision for class j are defined as follows.

$$\text{recall}_j = \sum_{i=1}^{N} \frac{(s_{ij} > \tau) \wedge (s_i^* = j)}{s_i^* = j},$$

$$\text{precision}_j = \sum_{i=1}^{N} \frac{(s_{ij} > \tau) \wedge (s_i^* = j)}{s_{ij} > \tau}. \tag{3}$$

Fig. 5 shows the recall and precision (averaged over classes) with the varying score thresholds τ compared with those of hard semantic prediction. With hard semantic prediction, the recall is 79.1%, indicating more than 20% amount of points over classes are not covered by the predictions. When using the score threshold, the recall increases as the score threshold decreases. However, the small score threshold also leads to low precision. We propose a top-down refinement stage to mitigate the low-precision problems. The precision can be interpreted as the relation between foreground and background points of object instances. We set the threshold to 0.2 with precision near 50%, leading to the ratio between foreground and background points for the ensuring stage being balanced.

C. Top-Down Refinement

The top-down refinement stage classifies and refines the instance proposals from the bottom-up grouping stage. A feature extractor layer processes each proposal to extract its corresponding backbone features. The extracted features are fed into a tiny U-Net network (a U-Net style network with a small number of layers) before predicting classification scores, instance masks, and mask scores at the ensuing branches.

Classification Branch: The classification branch starts with a global average pooling layer to aggregate the feature of all points in the instance, followed by a MLP to predict the classification scores $\boldsymbol{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\} \in \mathbb{R}^{K \times (N_{\text{class}}+1)}$, where K is the number of instances and $N_{\text{class}} + 1$ indicates N_{class} foreground classes with an extra background. We directly derive the object category and classification confidence score from the output of the classification branch.

We note that existing grouping-based methods typically derive the object category from semantic predictions. However, instances may come from objects with noisy semantic predictions. The proposed method directly uses the output of the classification branch as the instance class. The classification branch aggregates all point features of the instance and classifies the instance with a single label, leading to more reliable predictions.

Segmentation Branch: As shown in Section III-B, the instance proposals contain both foreground and background points, we construct a segmentation branch to predict an instance mask within each proposal. The segmentation branch is a point-wise MLP of two layers that output an instance mask $\boldsymbol{m}_k$ for each instance k.

Mask Scoring Branch: The mask scoring branch shares the same structure as the classification branch. This branch outputs the mask scores $\boldsymbol{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K\} \in \mathbb{R}^{K \times N_{\text{class}}}$, which estimate the IoU of a predicted mask with the ground truth. The mask score is combined with the classification score by multiplication to get the final confidence score.

Learning Targets: Training the top-down refinement branches requires the target labels for each branch. To this end, we follow the logic in existing 2D object detection and segmentation methods [28], [46]. We treat all instance proposals having IoU with a ground-truth instance higher than 50% as the positive samples and the rest as negatives. Every positive sample is assigned to a ground-truth instance with the highest IoU. The classification target of a positive sample is the categorical label of the corresponding ground-truth instance. The segmentation and mask scoring branches are trained on positive samples only. The mask target of a positive sample is the mask of the assigned ground-truth instance. The mask score target is the IoU between the predicted mask and the ground truth. The training loss of these branches is the combination of cross-entropy, binary cross-entropy, and ℓ_2 regression losses, following [28], [47].

$$L_{\text{class}} = \frac{1}{K} \sum_{k=1}^{K} \text{CE}(\boldsymbol{c}_k, c_k^*), \tag{4}$$

$$L_{\text{mask}} = \frac{1}{\sum_{k=1}^{K} \mathbb{1}_{\{\boldsymbol{m}_k\}}} \sum_{k=1}^{K} \mathbb{1}_{\{\boldsymbol{m}_k\}} \text{BCE}(\boldsymbol{m}_k, \boldsymbol{m}_k^*), \tag{5}$$

$$L_{\text{mask_score}} = \frac{1}{\sum_{k=1}^{K} \mathbb{1}_{\{\boldsymbol{e}_k\}}} \sum_{k=1}^{K} \mathbb{1}_{\{\boldsymbol{e}_k\}} \|\boldsymbol{e}_k - \boldsymbol{e}_k^*\|_2. \tag{6}$$

Here, c^*, $\boldsymbol{m}^*$, $\boldsymbol{e}^*$ are the classification, segmentation, and mask scoring targets, respectively. K is the total number of proposals and $\mathbb{1}_{\{\cdot\}}$ indicates whether the proposal is a positive sample.

D. Multi-Task Learning

The whole network can be trained in an end-to-end manner using a multi-task loss.

$$L = L_{\text{semantic}} + L_{\text{offset}} + L_{\text{class}} + L_{\text{mask}} + L_{\text{mask_score}}, \tag{7}$$

where L_{semantic} and L_{offset} are the semantic and offset losses defined at Section III-A while L_{class}, L_{mask} and $L_{\text{mask_score}}$ are the classification, segmentation and mask score losses defined at Section III-C.

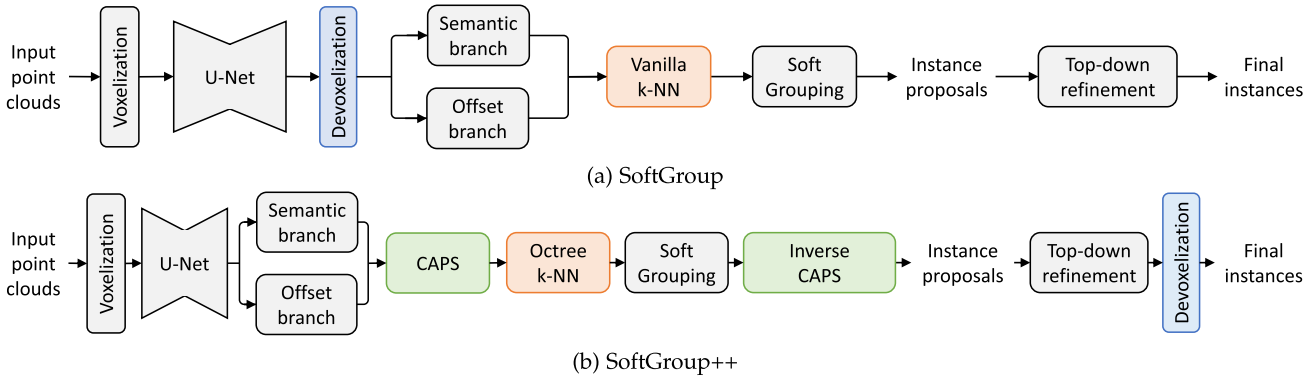


Fig. 6. *Architecture comparison between SoftGroup and SoftGroup++.* To highlight the differences between them, devoxelization and vanilla k-NN of SoftGroup are shown. SoftGroup++ replaces vanilla k-NN with octree k-NN, introduces class-aware pyramid scaling (CAPS), and delay voxelization until the end of the network.

IV. SoftGroup++ for Scalable Point Cloud Instance Segmentation

For fast inference on large-scale scenes, SoftGroup++ extends SoftGroup with two major improvements: time complexity and search space reduction. The overall architecture of SoftGroup++ in comparison with SoftGroup is illustrated in Fig. 6. To highlight the differences between these two architectures, relevant components for comparison of SoftGroup (i.e., devoxelization and vanilla k-NN) are shown. SoftGroup++ replaces vanilla k-NN with octree k-NN to reduce time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. Additionally, class-aware pyramid scaling (CAPS) and late devoxelization are proposed to reduce search space. These improving components are parameter-free, and thus inference on SoftGroup and SoftGroup++ can be performed from the same trained models. Importantly, since each component can receive either point or voxel input, the presentation in the next subsections assumes point input is used for simplicity.

A. Time Complexity Reduction

Octree k-NN: In recent grouping-based instance segmentation methods [1], [3], k-NN constructs the point adjacency matrix, which serves as the prerequisite for grouping. To cope with varying point density, a radius constraint r is added such that the distance from a valid neighbor to the query point should be less than r. Existing methods adopt vanilla k-NN algorithm, where pair-wise distance needs to be evaluated on the whole point set, and thus the time complexity of this algorithm w.r.t. the number of points is $\mathcal{O}(n^2)$. Since this quadratic time complexity is not scalable, we propose octree k-NN with time complexity of $\mathcal{O}(n \log n)$.

Constructing Octree: Octree is a data structure that partitions the 3D space by recursively subdividing it into eight octants. Given a set of points, we first derive its tight axis-aligned bounding box. Then we recursively divide the 3D box into eight child boxes (octants). To balance the construction and traversal time, we limit the number of tree levels M to a small value (e.g., 3) and store points in the last tree level (leaf nodes).

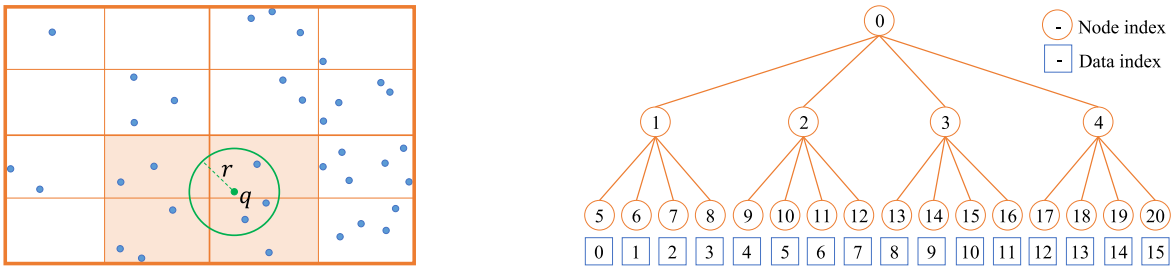
k-Nearest Neighbor Search on Octree: Given query point q and the constructed octree, we are ready to perform k-NN of the query q with radius search r. The details are presented in Algorithm 1. The core idea of the algorithm is to find a small point subset near the query and then perform vanilla k-NN on the subset as opposed to the whole set. Starting from the root node, the algorithm recursively traverses through the tree. If the box associated with the current node intersects the sphere $S(q, r)$, there exist octants of the current node that intersects the sphere $S(q, r)$. These octants are enqueued and then checked in next iterations. The procedure is repeated until the leaf nodes. A point list P is used to store all the points associated with the leaf nodes having intersections with the sphere $S(q, r)$. Fig. 7(a) illustrates the results of the algorithm on a two-level tree. For a neat presentation, we consider a quadtree which is the 2D version of the octree. The query sphere has intersections with 4 boxes of leaf nodes, of which the points are taken to perform k-NN with the query.

To construct the adjacency matrix of all points, the algorithm is performed on each point of the point set, which requires GPU parallelization for a speed boost. One of the main challenges for parallelization is that the implementation of recursive tree traversal on CUDA kernel is nontrivial. Inspired by [21], [48], we present a simple strategy to unfold and traverse the tree using direct indexing, such that the algorithm is performed on each CUDA kernel for each point effectively. To this end, when constructing the tree, we index the tree nodes in breadth-first order, as shown in Fig. 7(b), given a parent node index i, we can access its child nodes directly by simple arithmetic:

$$\text{ch}_j(i) = i \times 2^d + j \text{ for } j = 1..2^d \tag{8}$$

where $\text{ch}_j(i)$ is index of the j-th child node and d is dimension (d is 3 in octree or 2 in quadtree). Differ from [21], [48], we construct a full tree such that it is not required to encode the tree structure. The data index to access the points associated with the leaf node i can be derived as:

$$\text{data}(i) = i - \sum_{m=0}^{M-1} 2^{md} \tag{9}$$



(a) Quadtree k-NN (b) Node index and data index in breadth-first order

Fig. 7. *Visualization of quadtree k-NN and breadth-first indices:* For neat presentation, we consider a quadtree of 2D data. (a) *Illustration of 2-level quadtree k-NN on query q with search radius r.* There are 4 leaf nodes (highlighted in orange background) that intersects query circle. Quadtree k-NN only measures distance from query to the points in these leaf nodes instead of the whole point set. *(b) Illustration of node and data indices in breadth-first order.* From breadth-first order, the recursive structure of the tree can be unfolded where child nodes and data indices can be retrieved by simple arithmetic given in (8) and (9) such that the tree-based search algorithm can be parallelized effectively on GPU.

Algorithm 1: Octree k-NN.

Input: q: point query, r: search radius, $S(q, r)$: sphere with center q and radius r, $root$: root node of constructed octree, each node is associated with corresponding boxes, points, and octants.
Output: k nearest neighbors of query q with radius r.

```
Initialize an empty point list P
Initialize an empty node queue Q
Q.enqueue(root)
while Q is not empty do
  node = Q.dequeue()
  if node.box ∩ S(q, r) ≠ ∅ then
    if node is not leaf then
      for octant in node.octants do
        Q.enqueue(octant)
      end for
    else
      P.append(node.points)
    end if
  end if
end while
Perform k-NN of query q with radius r on point set P
```

Since node and data indices are derived via simple arithmetic, we can unfold the recursive structure and parallelize the octree k-NN such that each CUDA kernel performs the algorithm for a query.

B. Search Space Reduction

Class-Aware Pyramid Scaling: Class-Aware Pyramid Scaling (CAPS) is proposed to reduce search space for k-NN and soft grouping module. Here, the search space is referred to as the number of input points used in k-NN. In point cloud processing, common search space reduction methods are point sampling (e.g., random sampling, furthest point sampling) and voxelization. In this work, voxelization is adopted since point-level results can be simply attained using the inverse mapping of voxelization. It is important to note that the voxelization step of CAPS is independent of input voxelization at the beginning of the network. Before delving into the proposed CAPS, we present a naive scaling strategy for search space reduction. In particular, the naive scaling strategy directly voxelizes semantic and offset predictions of the whole scene with a single voxel size. The scores/features of points within a voxel are averaged to get corresponding voxel scores/features. This strategy exhibits two major limitations: (1) the scores of a class may interfere with the scores of other classes due to score averaging in voxelization and (2) it is difficult to choose a fixed voxel size for downscaling the whole scene since different object classes may have different object sizes. As ablated in our experiments (Section V-D), this strategy severely worsens the prediction accuracy.

The limitations of the naive scaling strategy are addressed in the proposed CAPS. To avoid score interference between classes, the proposed CAPS is class-aware such that downscaling is performed on a point subset for a given class. To cope with the variation in object sizes of different classes, CAPS adaptively selects downscaling levels such that the object with more points has a higher downscaling level. The details of CAPS are presented in Algorithm 2. The algorithm iterates through C classes and extract semantic $\boldsymbol{S}_i$ and offset $\boldsymbol{O}_i$ subsets for i-th class. The pyramid level l is computed by comparing the number of points in $\boldsymbol{S}_i$ with predefined thresholds $\boldsymbol{t}$. Then, $\boldsymbol{S}_i$ and $\boldsymbol{O}_i$ are downscaled with a voxel size of $l \times V$, where V is the base voxel size. The downscaled semantic and offset predictions of all classes are aggregated and used as the input for k-NN and soft grouping. Since top-down refinement stage requires instance proposals of the same scale, a module referred to as inverse CAPS is applied, where inverse mapping of the voxelization in CAPS is used to perform inverse scaling.

Late Devoxelization: In voxel-based networks, it is common that the number of voxels is much less than the number of points. Table I illustrates the ratio between voxels and points in different datasets. The number of voxels is computed with the voxel size according to Table II. Existing methods [1], [2], [3] typically perform early devoxelization, where the conversion from voxels to points is performed right after the U-Net backbone. It is noted that all the points within a voxel share the same values after devoxelization. Early devoxelization lead to the repetition of

Algorithm 2: Class-Aware Pyramid Scaling.

Input: L: number of pyramid levels, V: base voxel size, C: number of classes, $\boldsymbol{t} = [t_1, \ldots, t_L]$: increasing order thresholds to determine pyramid levels, $\boldsymbol{S}$: semantic predictions, $\boldsymbol{O}$: offset predictions.
Output: $\boldsymbol{S}'$: semantic predictions after CAPS, $\boldsymbol{O}'$: offset predictions after CAPS.
1: Initialize empty sets $\boldsymbol{S}' = \{\}$ and $\boldsymbol{O}' = \{\}$
2: **for** $i = 1$ **to** C **do**
3: Extract semantic and offset prediction subsets for i-th class $\boldsymbol{S}_i \subset \boldsymbol{S}$ and $\boldsymbol{O}_i \subset \boldsymbol{O}$
4: Get number of voxels in the subset $|\boldsymbol{S_i}|$
5: Compute pyramid level: $l = \text{argmin}_{j \in \{1,\ldots,L\}}\{t_j > |\boldsymbol{S_i}|\}$
6: Voxelize $\boldsymbol{S}_i$ and $\boldsymbol{O}_i$ with voxel size being $l \times V$ which results in $\boldsymbol{S}'_i$ and $\boldsymbol{O}'_i$
7: Aggregate downscaled results: $\boldsymbol{S}' = \boldsymbol{S}' \cup \boldsymbol{S}'_i$, $\boldsymbol{O}' = \boldsymbol{O}' \cup \boldsymbol{O}'_i$
8: **end for**

TABLE I
RATIO BETWEEN #VOXELS AND #POINTS

ScanNet v2	S3DIS	STLS3D	SemanticKITTI
0.69	0.27	0.92	0.72

TABLE II
VOXEL SIZE AND GROUPING BANDWIDTH (IN METERS) FOR DIFFERENT DATASETS

Hyper-parameter	ScanNet v2	S3DIS	STLS3D	SemanticKITTI
Voxel size	0.02	0.02	0.33	0.05
Grouping bandwidth	0.04	0.04	0.90	0.10

computation on the same input, and thus the ensuing modules are operated at a high computational cost. To address this problem, this work proposes to delay the conversion from voxels to points until the end of the network, which is referred to as late devoxelization. Late devoxelization eliminates computation repetition and reduces input size for intermediate network components, leading to a speed boost. The effectiveness of late devoxelization can be anticipated by the voxel-to-point ratio.

V. EXPERIMENTS

A. Experimental Settings

Datasets: The experiments are conducted on various indoor and outdoor datasets, including S3DIS [49], ScanNet v2 [50], STPLS3D [51], SemanticKITTI [52]. The ScanNet dataset contains 1613 scans which are divided into training, validation, and testing sets of 1201, 312, 100 scans, respectively. Instance segmentation is evaluated on 18 object classes. Following existing methods, the benchmarked results are reported on the hidden test split. The ablation study is conducted on the validation set.

The S3DIS dataset contains 3D scans of 6 areas with 271 scenes in total. The dataset consists of 13 classes for instance segmentation evaluation. Following existing methods, two settings are used to evaluate the instance segmentation results: testing on Area 5 and 6-fold cross-validation.

The STPLS3D dataset is recently proposed for outdoor 3D instance segmentation with more than 16 km^2 of landscapes. The dataset is annotated with 14 instance segmentation classes. The point clouds are cropped into non-overlapped blocks of 250 m^2, resulting in nearly 1 million points each scene. We follow [51] to divide the data to train and test splits.

The SemanticKITTI dataset is proposed for panoptic segmentation, which unifies instance segmentation (for thing classes) and semantic segmentation (for stuff classes). The dataset consists of 22 sequences with 43552 frames. The training and validation are performed on sequences 00-10, and testing is conducted on sequences 11-21. Panoptic segmentation is evaluated on 11 thing and 8 stuff classes.

Evaluation Metrics: The evaluation metric is the standard average precision. Here, AP_{50} and AP_{25} denote the scores with IoU thresholds of 50% and 25%, respectively. Likewise, AP denotes the averaged scores with IoU threshold from 50% to 95% with a step size of 5%. Additionally, the S3DIS is also evaluated using mean precision (mPrec), and mean recall (mRec), which are adopted in previous methods. For panpotic segmentation, the panoptic quality (PQ) [53] is used as the default evaluation metric. PQ captures both recognition quality (RQ) and segmentation quality (PQ). PQ^{th} and PQ^{st} indicate the scores on thing and stuff classes separately. In addition, $PQ^{\dagger}$ is also reported as suggested in [54]. Runtime is measured on a single RTX 8000.

Implementation Details: The implementation details follow those of existing methods [1], [3]. The model is implemented using PyTorch deep learning framework [60] and trained by Adam optimizer [61] with a batch size of 4. The learning rate is initialized to 0.001 and scheduled by a cosine annealing [62]. The score threshold for soft grouping τ is set to 0.2. Note that the voxel size and grouping radius r should be adapted for different datasets. We following [1], [51] to set these hyper-parameters, as summarized in Table II. Extra hyper-parameters of SoftGroup++ are set as follows. CAPS uses 3 levels with the threshold to determine pyramid level being $\boldsymbol{t} = [10^5, 10^6, +\infty]$. Based voxel size of CAPS is set to the input voxel size. The number of octree levels is set to 3. At training time, the scenes are randomly cropped to at most 250 k points. At inference, the whole scene is fed into the network without cropping.

Especially, S3DIS has large-scale scenes of high-point density, SoftGroup and SoftGroup++ are set up differently. In particular, SoftGroup follows [1] to randomly subsample the scenes at a ratio of 1/4. At inference, the scene is divided into four parts before feeding into the model, and then the

TABLE III
3D INSTANCE SEGMENTATION RESULTS ON SCANNET V2 HIDDEN TEST SET IN TERMS OF AP_{50} SCORES

Method	AP_{50}	bathtub	bed	bookshe.	cabinet	chair	counter	curtain	desk	door	other	picture	fridge	s. curtain	sink	sofa	table	toilet	window
SGPN [32]	14.3	20.8	39.0	16.9	6.5	27.5	2.9	6.9	0.0	8.7	4.3	1.4	2.7	0.0	11.2	35.1	16.8	43.8	13.8
GSPN [29]	30.6	50.0	40.5	31.1	34.8	58.9	5.4	6.8	12.6	28.3	29.0	2.8	21.9	21.4	33.1	39.6	27.5	82.1	24.5
3D-SIS [30]	38.2	**100.0**	43.2	24.5	19.0	57.7	1.3	26.3	3.3	32.0	24.0	7.5	42.2	85.7	11.7	69.9	27.1	88.3	23.5
MASC [55]	44.7	52.8	55.5	38.1	38.2	63.3	0.2	50.9	26.0	36.1	43.2	32.7	45.1	57.1	36.7	63.9	38.6	98.0	27.6
PanopticFusion [56]	47.8	66.7	71.2	59.5	25.9	55.0	0.0	61.3	17.5	25.0	43.4	43.7	41.1	85.7	48.5	59.1	26.7	94.4	35.9
3D-Bonet [4]	48.8	**100.0**	67.2	59.0	30.1	48.4	9.8	62.0	30.6	34.1	25.9	12.5	43.4	79.6	40.2	49.9	51.3	90.9	43.9
MTML [34]	54.9	**100.0**	80.7	58.8	32.7	64.7	0.4	81.5	18.0	41.8	36.4	18.2	44.5	**100.0**	44.2	68.8	57.1	**100.0**	39.6
3D-MPA [57]	61.1	**100.0**	83.3	76.5	52.6	75.6	13.6	58.8	47.0	43.8	43.2	35.8	65.0	85.7	42.9	76.5	55.7	**100.0**	43.0
Dyco3D [58]	64.1	**100.0**	84.1	89.3	53.1	80.2	11.5	58.8	44.8	43.8	53.7	43.0	55.0	85.7	53.4	76.4	65.7	98.7	56.8
PE [36]	64.5	**100.0**	77.3	79.8	53.8	78.6	8.8	79.9	35.0	43.5	54.7	54.5	64.6	93.3	56.2	76.1	55.6	99.7	50.1
PointGroup [1]	63.6	**100.0**	76.5	62.4	50.5	79.7	11.6	69.6	38.4	44.1	55.9	47.6	59.6	**100.0**	66.6	75.6	55.6	99.7	51.3
GICN [31]	63.8	**100.0**	**89.5**	80.0	48.0	67.6	14.4	73.7	35.4	44.7	40.0	36.5	70.0	**100.0**	56.9	83.6	59.9	**100.0**	47.3
OccuSeg [35]	67.2	**100.0**	75.8	68.2	57.6	84.2	**47.7**	50.4	52.4	56.7	58.5	45.1	55.7	**100.0**	75.1	79.7	56.3	**100.0**	46.7
SSTNet [2]	69.8	**100.0**	69.7	88.8	55.6	80.3	38.7	62.6	41.7	55.6	58.5	**70.2**	60.0	**100.0**	**82.4**	72.0	69.2	**100.0**	50.9
HAIS [3]	69.9	**100.0**	84.9	82.0	67.5	80.8	27.9	75.7	46.5	51.7	59.6	55.9	60.0	**100.0**	65.4	76.7	67.6	99.4	56.0
SoftGroup (ours)	76.1	**100.0**	80.8	84.5	**71.6**	86.2	24.3	82.4	65.5	**62.0**	73.4	69.9	79.1	98.1	71.6	84.4	76.9	**100.0**	59.4
SoftGroup++ (ours)	**76.9**	**100.0**	80.3	**93.7**	68.4	**86.5**	21.3	**87.0**	**66.4**	57.1	**75.8**	**70.2**	**80.7**	**100.0**	65.3	**90.2**	**79.2**	**100.0**	**62.6**

features of these parts are merged right after the U-Net backbone. Meanwhile, since SoftGroup++ is efficient, the whole scene is used without subsampling. This implementation detail ensures consistency between training and testing for the network, thus bringing extra AP improvements of SoftGroup++ on S3DIS dataset.

B. Benchmarking Results

1) Accuracy Analysis: ScanNet v2: Table III shows the results of our method and recent state-of-the-art on the hidden test set of ScanNet v2 benchmark. SoftGroup achieves AP_{50} of 76.1%, surpassing the existing best method (HAIS) by a significant margin of 6.2%. SoftGroup++ slightly improves the performance to 76.9%. Regarding class-wise scores, SoftGroup++ achieves the best performance in 13 out of 18 classes.

S3DIS: Table IV summarizes the results on Area 5 and 6-fold cross-validation of S3DIS dataset. On both Area 5 and cross-validation evaluations, the proposed SoftGroup and SoftGroup++ achieve higher overall performance compared to existing methods. Notably, on Area 5 evaluation, SoftGroup achieves AP/AP_{50} of 51.6/66.1(%), which is 8.9/6.8(%) improvement compared to SSTNet. SoftGroup++ further improves the overall results with 1.7% and 2.4% higher than SoftGroup on Area 5 and cross-validation settings, respectively.

STPLS3D: We report instance segmentation results on STPLS3D dataset in Table V. STPLS3D is an outdoor dataset with larger sparsity and object size variation compared to ScanNet and S3DIS. SoftGroup achieves AP/AP_{50}/AP_{25} of 47.3/63.1/71.4(%) which is 12.2/16.4/21.1(%) improvement compared to HAIS. SoftGroup++ performs comparably to SoftGroup with AP/AP_{50}/AP_{25} of 46.5/62.9/71.8(%). The significant performance on different indoor and outdoor datasets demonstrates the superiority and generalization of the proposed method.

TABLE IV
3D INSTANCE SEGMENTATION RESULTS ON S3DIS DATASET

Method	AP	AP_{50}	$mPrec_{50}$	$mRec_{50}$
SGPN† [32]	-	-	36	28.7
ASIS† [59]	-	-	55.3	42.4
PointGroup† [1]	-	57.8	61.9	62.1
SSTNet† [2]	42.7	59.3	65.5	64.2
HAIS† [3]	-	-	71.1	65.0
SoftGroup† (ours)	**51.6**	66.1	73.6	66.6
SoftGroup++† (ours)	50.9	**67.8**	**73.8**	**67.6**
SGPN‡ [32]	-	-	38.2	31.2
ASIS‡ [59]	-	-	63.6	47.5
3D-BoNet‡ [4]	-	-	65.6	47.7
PointGroup‡ [1]	-	64.0	69.6	69.2
SSTNet‡ [2]	54.1	67.8	73.5	73.4
HAIS‡ [3]	-	-	73.2	69.4
SoftGroup‡ (ours)	54.4	68.9	75.3	69.8
SoftGroup++‡ (ours)	**56.6**	**71.3**	**75.9**	**74.4**

Methods marked with † are evaluated on area 5, and method marked with ‡ are evaluated on 6-fold cross-validation.

TABLE V
3D INSTANCE SEGMENTATION RESULTS ON STPLS3D

Method	AP	AP_{50}	AP_{25}
PointGroup [1]	23.3	38.5	48.6
HAIS [3]	35.1	46.7	52.8
SoftGroup (ours)	**47.3**	**63.1**	71.4
SoftGroup++ (ours)	46.5	62.9	**71.8**

2) Runtime Analysis: Table VI reports the runtime comparison on different dataset. For a fair comparison, the reported runtime is measured on the same RTX 8000 GPU model. Since the number of points for a scan varies, we report both maximum and average runtime. SoftGroup shows slight improvement compared to HAIS. SoftGroup++ achieves the fastest inference

TABLE VI
RUNTIME COMPARISON ON A SINGLE RTX 8000

Method	S3DIS	ScanNet v2	STPLS3D
PointGroup [1]	-	1.16 (0.30)	-
SSTNet [2]	-	1.56 (0.29)	-
HAIS [3]	23.59 (3.87)	0.85 (0.27)	3.88 (2.83)
SoftGroup (ours)	16.46 (2.20)	0.84 (0.20)	3.42 (2.33)
SoftGroup++ (ours)	**1.27 (0.38)**	**0.41 (0.14)**	**2.24 (1.30)**

Numbers w/o and w/ parenthesizes are maximum and average runtime per scan, respectively.

TABLE VII
COMPONENT-WISE ANALYSIS ON SCANNET V2 VALIDATION SET

Baseline	Soft grouping	Top-down refinement	AP	AP_{50}	AP_{25}
✓			39.5	61.1	75.5
	✓		41.6	63.8	**79.2**
		✓	44.3	65.4	78.1
	✓	✓	**46.0**	**67.6**	78.9
Overall improvement			**+6.5**	**+6.5**	**+3.4**

Our model achieves significant improvement over the baseline.

TABLE VIII
ABLATION EXPERIMENTS ON VARYING SCORE THRESHOLD τ FOR SOFT GROUPING. "NONE" DENOTES THE THRESHOLD IS NOT USED, AND THE HARD SEMANTIC PREDICTION IS USED FOR GROUPING

τ	AP	AP_{50}	AP_{25}
None	44.3	65.4	78.1
0.01	40.1	58.5	69.2
0.1	45.3	66.5	78.5
0.2	**46.0**	**67.6**	**78.9**
0.3	45.2	66.8	78.5
0.4	44.7	46.1	78.3
0.5	43.9	64.8	77.7

speed in all measures. Notably, on S3DIS which consists of large-scale scenes, SoftGroup++ requires a maximum runtime of 1.27 s, which is significantly faster than SoftGroup of 16.46 s. On average, SoftGroup++ is 6× faster than SoftGroup on S3DIS.

C. SoftGroup Ablation Study

SoftGroup Component-Wise Analysis: We provide experimental results of SoftGroup when different components are omitted. The considered baseline is a model with hard grouping and the confidence scores of output instances are ranked by a ScoreNet branch [1], [2]. Table VII shows the ablation results. The baseline achieves 39.5/61.1/75.5(%) in terms of AP/AP_{50}/AP_{25}. Significant improvement is obtained by either applying soft grouping or top-down refinement. Combining these two components achieves the best overall performance AP/AP_{50}/AP_{25} of 46.0/67.6/78.9(%), which is significantly higher than the baseline by 6.5/6.5/3.4(%).

*Score Threshold for Soft Grouping:*Table VIII shows the experimental results with varying score thresholds for soft grouping. The baseline is with τ being "None", indicating the threshold is deactivated and the hard predicted label is used for grouping. The baseline achieves AP/AP_{50}/AP_{25} of 44.3/65.4/78.1(%). When τ is too high or too low the performance is even worse than the baseline. The best performance is obtained at τ of 0.2, which confirms our analysis at the Section III-B, where the number of positive and negative samples are balanced.

TABLE IX
THE IMPACT OF EACH BRANCH IN TOP-DOWN REFINEMENT ON SCANNET V2 VALIDATION SET

Class	Mask	Mask score	AP	AP_{50}	AP_{25}
✓			41.1	64.6	**79.7**
✓	✓		45.7	**68.4**	79.5
✓	✓	✓	**46.0**	67.6	78.9

Top-Down Refinement: We further provide the ablation results on the top-down refinement, on Table IX. With only the classification branch, our method achieves AP/AP_{50}/AP_{25} of 41.1/64.6/79.7(%). When mask branch and mask scoring branch are in turn applied, the performance tends to improve on the higher IoU threshold regions. Combining all branches yields the performance AP/AP_{50}/AP_{25} of 46.0/67.6/78.9(%).

Qualitative Analysis: Fig. 8 shows the visualization on ScanNet v2 dataset. Without SoftGroup, the semantic prediction errors are propagated to instance segmentation predictions (highlighted by dashed boxes). In contrast, SoftGroup effectively corrects the semantic prediction errors and thus generates more accurate instance masks.

D. SoftGroup++ Ablation Study

SoftGroup++ Component-Wise Analysis: The ablation study on SoftGroup++ is conducted on S3DIS since it consists of large-scale scenes. We report the experimental results when different components are omitted in Table X. Without proposed components, the runtime of the baseline model is 30.32 s with k-NN being the computational bottleneck of 23.32 s. When octree k-NN, CAPS, and late devoxelization are individually applied, the runtime of the above bottleneck significantly to 2.82 s, 1.32 s, and 0.96 s, respectively. When all proposed components are applied, the model achieves the lowest latency of 1.28 s, which is nearly 24× faster than the baseline.

Octree k-NN: Based on the two last rows of Table X, octree seems only slightly speeds up k-NN from 0.50 s to 0.44 s. The reason for the slight improvement is that the search space is reduced more significantly on S3DIS by CAPS and late devoxelization (implied by the voxel-to-point ratio in Table I), and thus the runtime difference of $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^2)$ algorithms is not very significant. However, the amount of search space reduction is non-deterministic, highly depending on the point density of different datasets. Octree k-NN is beneficial in reducing the time complexity of the framework: it solves the root cause of the problem by making sure that each component in the framework has a time complexity of less than $\mathcal{O}(n^2)$. For example, in a more challenging case, we can see the effectiveness of octree k-NN more clearly. We report the runtime of the whole scene (without cropping) of STPLS3D dataset with ∼4 M points in

TABLE X
ABLATION STUDY ON EACH COMPONENT OF THE PROPOSED METHOD ON THE LARGEST SCENE OF S3DIS AREA 5 WITH ~4.5 M POINTS

Baseline	Octree	CAPS	Late devoxel.	Point-wise	k-NN	Grouping	Top-down	Total
✓				0.13	23.32	4.54	2.33	30.32
✓	✓			0.13	2.82	4.54	2.36	9.85
✓		✓		0.13	1.32	0.11	2.38	3.94
✓			✓	0.13	0.96	0.45	0.51	2.05
✓	✓	✓		0.13	1.28	0.11	2.35	3.87
✓		✓	✓	0.13	0.50	0.23	0.48	1.34
✓	✓	✓	✓	0.13	0.44	0.23	0.48	**1.28**

The module-wise and total runtimes (in seconds) are measured on RTX 8000.

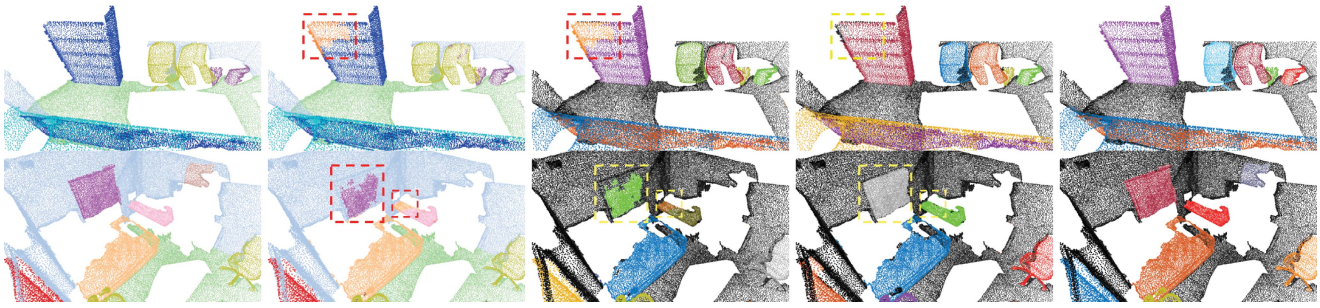


Fig. 8. *Qualitative results on ScanNet v2 validation set.* Instance prediction without SoftGroup output low-quality instance mask at the region of wrong semantic prediction (highlighted by dashed boxes). SoftGroup produces more accurate instance masks at these regions.

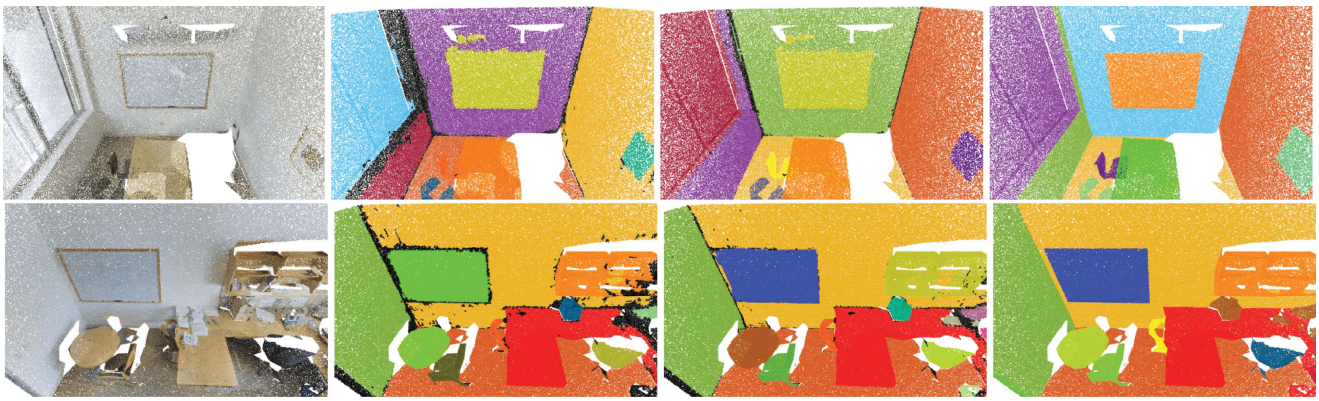


Fig. 9. *Qualitative comparison between naive scaling and the proposed CAPS.* Naive scaling misdetects the instance masks around object edges, illustrated in black points. The proposed CAPS produces more accurate instance masks.

TABLE XI
ABLATION W/ AND W/O OCTREE K-NN ON (UNCROPPED) STPLS3D DATASET

w/ Octree	Point-wise network	k-NN	Grouping	Top-down
N	0.25	1.48	2.98	1.53
Y	0.25	0.93	2.96	1.58

TABLE XII
ABLATION STUDY OF CAPS ON S3DIS AREA 5

Approach	AP	AP_{50}	AP_{25}	Time (s)
Naive scaling	47.0	64.7	73.7	0.44
CAPS	**50.9**	**67.8**	**76.9**	**0.39**

Table XI. The results indicate that the runtime of k-NN is more significantly reduced from 1.48 s to 0.93 s.

Effectiveness of CAPS: We compare the proposed CAPS with the naive scaling strategy in Table XII. While running slightly faster, CAPS achieves AP/AP_{50}/AP_{25} of 50.9/67.8/76.9(%) which is 3.9/3.1/3.2(%) improvement compared to naive scaling. The qualitative comparison between the two strategies is shown in Fig. 9. As expected, naive scaling misclassifies the points at

TABLE XIII
SEMANTIC SEGMENTATION RESULTS ON SCANNET V2 VALIDATION SET WITH BACK FUSION

Method	mIoU	bathtub	bed	bookshe.	cabinet	chair	counter	curtain	desk	door	other	picture	fridge	s. curtain	sink	sofa	table	toilet	window
w/o fusion	69.8	64.5	**80.3**	89.2	79.5	74.5	63.3	65.2	76.0	29.8	**65.3**	67.5	71.9	59.6	70.0	92.2	**63.9**	85.2	58.6
w/ fusion	**70.5**	**64.8**	80.3	**89.5**	**80.5**	**74.8**	**63.9**	**66.0**	**76.9**	**30.7**	64.5	**68.3**	**73.4**	**61.8**	**72.4**	**93.1**	63.5	**85.2**	**59.3**
Improvement	**0.7**	**0.3**	**0.0**	**0.3**	**1.0**	**0.3**	**0.6**	**0.8**	**0.9**	**0.9**	-0.8	**0.8**	**1.5**	**2.2**	**2.4**	**0.9**	-0.4	**0.0**	**0.7**

TABLE XIV
ABLATION STUDY WITH VARYING POINT DENSITY ON S3DIS AREA 5

Density	AP	AP_{50}	AP_{25}
1x	**50.9**	**67.8**	**76.9**
0.5x	50.6	67.7	76.4
0.25x	49.2	64.5	73.8
0.125x	46.0	62.1	72.5

TABLE XV
SOFTGROUP COMPLEMENTS SSTNET FOR BETTER OVERALL PERFORMANCE ON SCANNET VALIDATION SET

Method	AP	AP_{50}	AP_{25}
SoftGroup	46.0	67.6	**78.9**
SSTNet [2]	49.4	64.9	74.4
SSTNet [2] + SoftGroup	**51.4**	**67.7**	78.2

TABLE XVI
INSTANCE SEGMENTATION AND OBJECT DETECTION RESULTS ON SCANNET V2 VALIDATION SET

Method	Box AP_{50}	Box AP_{25}
F-PointNet [63]	10.8	19.8
GSPN [29]	17.7	30.6
3D-SIS [30]	22.5	40.2
VoteNet [64]	33.5	58.6
3D-MPA [57]	49.2	64.2
PointGroup [1]	48.9	61.5
SSTNet [2]	52.7	62.5
HAIS [3]	53.1	64.3
SoftGroup (ours)	59.4	71.6
SoftGroup++ (ours)	**59.6**	**71.7**

Our method achieves better results on both mask and box AP.

the object edges as the background (illustrated in black points), while the proposed CAPS produces more accurate instances.

Ablation Study on Varying Point Density: The generalization of the proposed method is demonstrated on various indoor and outdoor datasets with different point densities. We further study the performance of our method with the same dataset and varying point density. To this end, we subsample the original scenes of S3DIS with factors of 0.5x, 0.25x, and 0.125x. Table XIV reports the experiment results of SoftGroup++ with varying point density. Compared to the results from the original scenes (indicated by 1x), the performance on 0.5x scenes is approximately on par. As anticipated, as the point density decreases, there is a corresponding decline in performance. Grouping adjacent points becomes more challenging with a reduced point density. Fig. 10 provides some visual examples to illustrate this point. Despite the low point density, our method generates reasonably accurate instance masks.

E. Generalization

Complement to Superpoint-Based Instance Segmentation: Some existing methods, such as STTNet [2], exploit connectivity information among neighboring points to construct superpoint for more accurate instance segmentation. The concept behind SoftGroup is to prevent the error propagation from semantic segmentation to instance segmentation, and we demonstrate that this notion synergistically complements the superpoint-based approach. Table XV reports the experimental results of applying SoftGroup on SSTNet. SoftGroup brings substantial performance gain on SSTNet in terms of AP/AP_{50}/AP_{25} from 49.4/64.9/74.4(%) to 51.4/67.7/78.2(%). Compared to original SoftGroup, the combination of SoftGroup and SSTNet achieves significant AP improvement of 5.4%.

Semantic Segmentation With Back Fusion: We improve the semantic segmentation performance by fusing back the instance masks to semantic maps. To handle overlapping, the instances masks are pasted onto the semantic maps in ascending order of their confident scores. Table XIII reports the ablation results of back fusion. Overall, back fusion achieves mIoU improvement of 0.7 points on ScanNet v2 validation set. Notably, back fusion improves the performance of classes with close semantic meaning, such as `cabinet` versus `fridge`, `curtain` versus `shower curtain`.

Generalization to Object Detection: To obtain object detection results, we follow the approach in [57] to extract a tight axis-aligned bounding box from the predicted instance masks. Table XVI report the object detection results on ScanNet v2 validation. The proposed SoftGroup and SoftGroup++ significantly outperform the other methods in terms of bounding box AP. SoftGroup++ achieves the best overall performance with AP_{50}/AP_{25} of 59.6/71.7(%).

Generalization to Panoptic Segmentation: Since the proposed method produces both semantic and instance segmentation results, it naturally generalizes to panoptic segmentation. Table XVII reports the panoptic segmentation results on SemanticKITTI dataset. The proposed SoftGroup and SoftGroup++ achieve competitive performance. Panoptic-PHNet attains the best overall results with PQ of 61.5.



Fig. 10. *Visualization examples of instance segmentation results with varying point density.* Instances are denoted by different random colors.

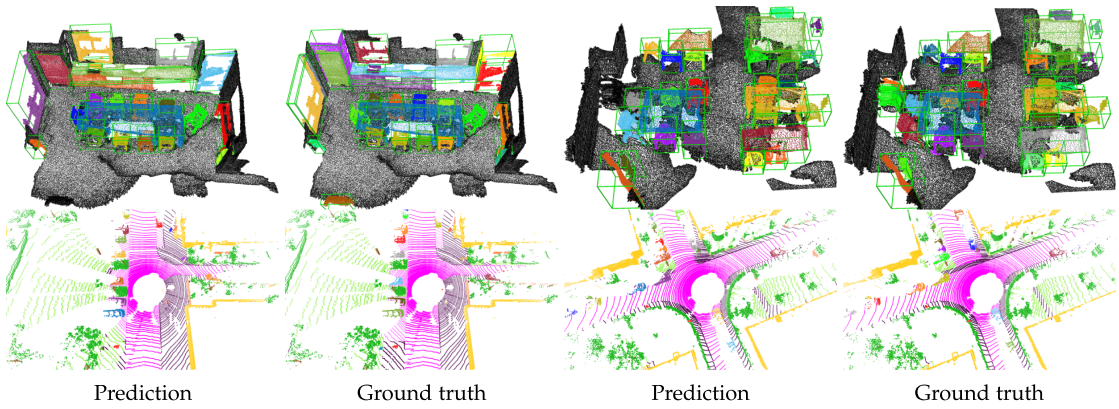


Fig. 11. Visualization examples of object detection on ScanNet v2 (two first rows) and panoptic segmentation on SemanticKITTI (two latter rows).

TABLE XVII
LIDAR-BASED PANOPTIC SEGMENTATION RESULTS ON SEMANTICKITTI TEST SET

Method	PQ	PQ†	RQ	SQ	PQ^{th}	RQ^{th}	SQ^{th}	PQ^{st}	RQ^{st}	SQ^{st}	mIoU
RangeNet++ [65] + PointPillars [66]	37.1	45.9	47.0	75.9	20.2	25.2	75.2	49.3	62.8	76.5	52.4
KPConv [15] + PV-RCNN [67]	50.2	57.5	61.4	80.0	43.2	51.4	80.2	55.9	68.7	79.9	62.8
LPASD [68]	38.0	47.0	48.2	76.5	25.6	31.8	76.8	47.1	60.1	76.2	50.9
PanosterK [38]	52.7	59.9	64.1	80.7	49.4	58.5	83.3	55.1	68.2	78.8	59.9
4D-PLS [69]	50.3	57.8	61.0	81.6	-	-	-	-	-	-	61.3
Panoptic-PolarNet [37]	54.1	60.7	65.0	81.4	53.3	60.6	87.2	54.8	68.1	77.2	59.5
DS-Net [39]	55.9	62.5	66.7	82.3	55.1	62.8	87.2	56.5	69.5	78.7	61.6
Panoptic-PHNet [40]	**61.5**	**67.9**	**72.1**	**84.8**	**63.8**	**70.4**	**90.7**	**59.9**	**73.3**	**80.5**	**66.0**
SoftGroup (ours)	57.0	63.9	68.1	82.5	56.5	65.2	85.4	57.3	70.2	80.4	61.7
SoftGroup++ (ours)	57.2	64.2	68.2	82.7	57.1	65.6	85.8	57.3	70.2	80.4	63.0

Panoptic-PHNet employs fused 2D-3D backbone, transformer-based offset branch, and center grouping module, which leads to significant performance gain. We posit that the SoftGroup concept is not only orthogonal to the advancements seen in Panoptic-PHNet, but it also complements them. This suggests potential for additional performance improvements. However, the lack of publicly released implementation details for Panoptic-PHNet makes the reproduction of the reported results a significant challenge. Thus, we defer a more thorough exploration of this potential performance enhancement to future research.

Fig. 11 illustrates the visualization of object detection and panoptic segmentation outputs. Our method produces plausible predictions under different scenarios. In summary, quantitative and qualitative results on different tasks and datasets demonstrate the versatility and generality of the proposed method.

VI. CONCLUSION

We present SoftGroup and its extended SoftGroup++ for accurate and scalable instance segmentation on 3D point clouds. SoftGroup performs grouping on soft semantic scores to address the problem stemming from hard grouping on locally ambiguous objects. Then a top-down refinement stage is constructed to refine the positive samples and suppress the negatives. To efficiently process large-scale scenes, SoftGroup++ is introduced for low time complexity and search space. Octree k-NN replaces vanilla k-NN to reduce the time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. Class-aware pyramid scaling and late devoxelization reduce the search space and runtime of intermediate components. Extensive experiments on various datasets demonstrate the superiority and generality of our method.

REFERENCES

[1] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "PointGroup: Dual-set point grouping for 3D instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4866–4875.

[2] Z. Liang, Z. Li, S. Xu, M. Tan, and K. Jia, "Instance segmentation in 3D scenes using semantic superpoint tree networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 2763–2772.

[3] S. Chen, J. Fang, Q. Zhang, W. Liu, and X. Wang, "Hierarchical aggregation for 3D instance segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 15447–15456.

[4] B. Yang et al., "Learning object bounding boxes for 3D instance segmentation on point clouds," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 6737–6746.

[5] T. Vu, K. Kim, T. M. Luu, X. T. Nguyen, and C. D. Yoo, "SoftGroup for 3D instance segmentation on point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 2698–2707.

[6] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2011, pp. 1626–1633.

[7] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 3212–3217.

[8] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 3384–3391.

[9] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1704–1711.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv: 1706.02413*.

[12] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 90–105.

[13] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8887–8896.

[14] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9613–9622.

[15] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6410–6419.

[16] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 984–993.

[17] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on x-transformed points," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 828–838.

[18] B. Graham, M. Engelcke, and L. Van Der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9224–9232.

[19] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3070–3079.

[20] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 922–928.

[21] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6620–6629.

[22] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3744–3753.

[23] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 16239–16248.

[24] C. Park, Y. Jeong, M. Cho, and J. Park, "Fast point transformer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16928–16937.

[25] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 29–38.

[26] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4548–4557.

[27] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, 2019, Art. no. 146.

[28] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[29] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, "GSPN: Generative shape proposal network for 3D instance segmentation in point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3942–3951.

[30] J. Hou, A. Dai, and M. Nießner, "3D-SIS: 3D semantic instance segmentation of RGB-D scans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4416–4425.

[31] S.-H. Liu, S.-Y. Yu, S.-C. Wu, H.-T. Chen, and T.-L. Liu, "Learning Gaussian instance segmentation in point clouds," 2020, *arXiv: 2007.09860*.

[32] W. Wang, R. Yu, Q. Huang, and U. Neumann, "SGPN: Similarity group proposal network for 3D point cloud instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2569–2578.

[33] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung, "JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8819–8828.

[34] J. Lahoud, B. Ghanem, M. Pollefeys, and M. R. Oswald, "3D instance segmentation via multi-task metric learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9255–9265.

[35] L. Han, T. Zheng, L. Xu, and L. Fang, "OccuSeg: Occupancy-aware 3D instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2937–2946.

[36] B. Zhang and P. Wonka, "Point cloud instance segmentation using probabilistic embeddings," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8879–8888.

[37] Z. Zhou, Y. Zhang, and H. Foroosh, "Panoptic-PolarNet: Proposal-free LiDAR point cloud panoptic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13189–13198.

[38] S. Gasperini, M.-A. N. Mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari, "Panoster: End-to-end panoptic segmentation of LiDAR point clouds," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3216–3223, Apr. 2021.

[39] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu, "LiDAR-based panoptic segmentation via dynamic shifting network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13085–13094.

[40] J. Li, X. He, Y. Wen, Y. Gao, X. Cheng, and D. Zhang, "Panoptic-PHNet: Towards real-time and high-precision LiDAR panoptic segmentation via clustering pseudo heatmap," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11799–11808.

[41] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "OctAttention: Octree-based large-scale contexts model for point cloud compression," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 625–633.

[42] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, and U. Neumann, "Grid-GCN for fast and scalable point cloud learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5660–5669.

[43] R. A. Rosu, P. Schütt, J. Quenzel, and S. Behnke, "LatticeNet: Fast point cloud segmentation using permutohedral lattices," in *Proc. Robot. Sci. Syst.*, 2020.

[44] S. Lombardi, M. R. Oswald, and M. Pollefeys, "Scalable point cloud-based reconstruction with local implicit functions," in *Proc. Int. Conf. 3D Vis.*, 2020, pp. 997–1007.

[45] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Interv.*, 2015, pp. 234–241.

[46] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[47] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring R-CNN," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6402–6411.

[48] A. Miller, V. Jain, and J. L. Mundy, "Real-time rendering and dynamic updating of 3-D volumetric data," in *Proc. Workshop Gen. Purpose Process. Graph. Process. Units*, 2011, Art. no. 8.

[49] I. Armeni et al., "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1534–1543.

[50] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2432–2443.

[51] M. Chen et al., "STPLS3D: A large-scale synthetic and real aerial photogrammetry 3D point cloud dataset," 2022, *arXiv:2203.09065*.
[52] J. Behley et al., "Semantickitti: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9296–9306.
[53] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9396–9405.
[54] L. Porzi, S. R. Bulo, A. Colovic, and P. Kontschieder, "Seamless scene segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8269–8278.
[55] C. Liu and Y. Furukawa, "MASC: Multi-scale affinity with sparse convolution for 3D instance segmentation," 2019, *arXiv: 1902.04478*.
[56] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "PanopticFusion: Online volumetric semantic mapping at the level of stuff and things," 2019, *arXiv: 1903.01177*.
[57] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner, "3D-MPA: Multi-proposal aggregation for 3D semantic instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9028–9037.
[58] T. He, C. Shen, and A. van denHengel, "DyCo3D: Robust instance segmentation of 3D point clouds through dynamic convolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 354–363.
[59] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia, "Associatively segmenting instances and semantics in point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4091–4100.
[60] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017.
[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[62] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. Int. Conf. Learn. Representations*, 2017.
[63] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 918–927.
[64] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9276–9285.
[65] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4213–4220.
[66] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12689–12697.
[67] S. Shi et al., "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10526–10535.
[68] A. Milioto, J. Behley, C. McCool, and C. Stachniss, "LiDAR panoptic segmentation for autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 8505–8512.
[69] M. Aygun et al., "4D panoptic LiDAR segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5523–5533.



Thang Vu (Member, IEEE) received the BSc degree in electronic and telecommunication engineering from Hanoi University of Science and Technology, in 2016, and the MSc degree in electrical engineering from Korea Advanced Institute of Science and Technology, in 2019. He is currently working toward the PhD degree with the Korea Advanced Institute of Science and Technology. His research interests include object detection and instance segmentation on 2D/3D data, sensor fusion, and various recognition problems.



Kookhoi Kim (Member, IEEE) received the BSc degree in electronic engineering from Hanyang University, in 2020, and the MSc degree in electrical engineering from Korea Advanced Institute of Science and Technology, in 2022. His research interests include machine learning and deep learning for computer vision.



Thanh Nguyen (Member, IEEE) received the BSc degree in electronic and automation engineering from the Ho Chi Minh City University of Science and Technology, in 2015. He is currently working toward the MSc and PhD degrees with Korea Advanced Institute of Science and Technology. His research interests include machine learning, deep learning, and reinforcement learning.



Tung M. Luu (Member, IEEE) received the BSc degree in electronic and telecommunication engineering from Hanoi University of Science and Technology, in 2017, and the MSc degree in electrical engineering from Korea Advanced Institute of Science and Technology, in 2020. He is currently working toward the PhD degree with Korea Advanced Institute of Science and Technology. His research interests include machine learning, deep learning, and reinforcement learning.



Junyeong Kim (Member, IEEE) received the BS, MS, and PhD degrees from KAIST, Republic of Korea, in 2021. He was a post-doctoral researcher associate with the Artificial Intelligence and Machine Learning Laboratory, School of Electrical Engineering, KAIST, Republic of Korea. He currently works as an assistant professor with the Department of AI, Chung-Ang University, Republic of Korea. His research interests include visual-language reasoning, visual question answering, and various video-based problem.



Chang D. Yoo (Senior Member, IEEE) received the BS degree in engineering and applied science from California Institute of Technology, the MS degree in electrical engineering from Cornell University, and the PhD degree in electrical engineering from the Massachusetts Institute of Technology. From 1997 to 1999, he was a senior researcher with Korea Telecom (KT). Since 1999, he has been on the faculty with the Korea Advanced Institute of Science and Technology (KAIST) in the School of Electrical Engineering and an adjunct professor with the Department of Computer Science. He served as Dean of the Office of Special Projects and Dean of the Office of International Relations, respectively. He serves and has served as a Technical Committee Member of IEEE Machine Learning Signal Processing from 2011–2013 and 2023–Now. He also served as Associate Editor of various IEEE Transactions including IEEE TASLP and IEEE TIFS.