Instance Segmentation in 3D Scenes using Semantic Superpoint Tree Networks

Zhihao Liang[1,2], Zhihao Li[3], Songcen Xu[3], Mingkui Tan[1] and Kui Jia[1,4,5*]
[1]South China University of Technology, [2]DexForce Technology Co., Ltd.
[3]Noah's Ark Lab, Huawei Technologies, [4]Pazhou Laboratory, [5] Peng Cheng Laboratory
eezhihaoliang@mail.scut.edu.cn, {kuijia, mingkuitan}@scut.edu.cn,
{zhihao.li, xusongcen}@huawei.com

Abstract

Instance segmentation in 3D scenes is fundamental in many applications of scene understanding. It is yet challenging due to the compound factors of data irregularity and uncertainty in the numbers of instances. State-of-the-art methods largely rely on a general pipeline that first learns point-wise features discriminative at semantic and instance levels, followed by a separate step of point grouping for proposing object instances. While promising, they have the shortcomings that (1) the second step is not supervised by the main objective of instance segmentation, and (2) their point-wise feature learning and grouping are less effective to deal with data irregularities, possibly resulting in fragmented segmentations. To address these issues, we propose in this work an end-to-end solution of Semantic Superpoint Tree Network (SSTNet) for proposing object instances from scene points. Key in SSTNet is an intermediate, semantic superpoint tree (SST), which is constructed based on the learned semantic features of superpoints, and which will be traversed and split at intermediate tree nodes for proposals of object instances. We also design in SSTNet a refinement module, termed CliqueNet, to prune superpoints that may be wrongly grouped into instance proposals. Experiments on the benchmarks of ScanNet and S3DIS show the efficacy of our proposed method. At the time of submission, SSTNet ranks top on the ScanNet (V2) leaderboard, with 2% higher of mAP than the second best method. The source code in PyTorch is available at https://github.com/Gorilla-Lab-SCUT/SSTNet.

1. Introduction

The task of 3D instance segmentation is fundamental in many applications concerned with 3D scene understanding. Given an observed scene of point cloud reconstructed from depth cameras via multi-view fusion techniques [8, 14], the task is to both assign semantic labels of pre-defined object categories to individual scene points, and differentiate those belonging to different object instances. Learning to achieve 3D instance segmentation is challenging at least in the following aspects: (1) observed scene points are usually sparse and irregular, which poses difficulties for learning point-wise classification based on shape features of local (and possibly global) contexts around individual points; (2) the unknown number of object instances in a scene introduces additional uncertainties to the problem of learning point-instance associations that is already combinatorial; (3) even though point-wise classification and point-instance associations can be conducted, learning consistencies among spatially adjacent points are not guaranteed, which may cause fragmented segmentations, especially around object boundaries (cf. Fig. 1 for an illustration).

State-of-the-art methods [16, 37, 9], e.g., those ranking top on the ScanNet benchmark [7], tackle (some of) the above challenges with the following general pipeline. They first train networks to learn point-wise features that are discriminative at both the semantic and instance levels, followed by a separate step of point clustering that groups together those believed to be on same instances, using the learned point-wise features. While promising, they have the following shortcomings. Firstly, the second step of point clustering is independent of network training, whose results are thus not guaranteed by guiding towards the ground-truth groupings of object instances. Secondly, while superpoints [19] have been commonly used for semantic segmentation of 3D points [18, 4], when coming to instance segmentation, these state-of-the-art methods, except OccuSeg [15], choose to conduct both feature learning and grouping in a point-wise manner, which takes away their chance to leverage the geometric regularities established at the mid-level shape representation of superpoints.

To overcome these shortcomings, we are motivated to develop an end-to-end solution for proposing object instances from an observed scene of points. Considering that a superpoint represents a geometrically homogeneous neighborhood, we choose to work with superpoints pre-

*Correspondence to Kui Jia <kuijia@scut.edu.cn>

computed from the scene points, and the problem of instance segmentation boils down as learning a network that groups superpoints on same object instances. In this work, we design such a solution called *Semantic Superpoint Tree Network (SSTNet)*, as illustrated in Fig. 2. Similar to existing methods, SSTNet starts with a backbone that learns point-wise semantic and instance-level features; differently from them, SSTNet immediately aggregates these features as superpoint-wise ones efficiently via point-wise pooling. Key in SSTNet is an intermediate, semantic superpoint tree (SST), with the superpoints as its tree leaves. SST is constructed based on the pooled semantic (and instance-level) features of superpoints, and will be traversed and split by the subsequent SSTNet module of binary classification; starting from the root, a proposal of object instance is formed as the superpoints of a tree branch when non-splitting decision is made at the intermediate tree node that spans the branch (cf. Fig. 3 for an illustration). Our tree construction is highly efficient by choosing ways of feature inheritance from leaves to the root and pair-wise similarity metric, which support fast algorithms such as nearest-neighbor chain [35]. We note that erroneous assignments of superpoints to instances may occur when constructing and traversing the tree. To compensate, we design a subsequent refinement module termed CliqueNet, which converts each proposed branch as a graph clique and learns to prune some of the branch nodes. A ScoreNet [16] is finally used to evaluate the generated proposals, which gives instance segmentation results of our SSTNet.

Thorough experiments on the benchmark datasets of ScanNet [7] and S3DIS[1] show the efficacy of our proposed method. Notably, SSTNet outperforms all existing methods on the two benchmarks, and at the time of submission, it ranks top on the ScanNet (V2) leaderboard, with 2% higher of mAP than the second best method. We finally summarize our technical contributions as follows.

- We propose an end-to-end solution of *Semantic Superpoint Tree Network (SSTNet)* to directly propose and evaluate object instances from observed 3D scenes. By working with superpoints, our method enjoys the benefit of geometric regularity that supports consistent and sharp segmentations, especially at object boundaries.

- We choose a strategy of divisive grouping in SSTNet, which first builds the tree, followed by tree traversal for object proposal via node splitting. By constructing the tree with appropriate node merging and feature inheritance, our strategy is an order of magnitude faster than the alternative, agglomerative grouping, thus enabling efficient training and inference of SSTNet.

- Considering that erroneous assignments of superpoints to instances may occur when constructing and traversing the tree, we design a refinement module in SSTNet, termed CliqueNet, which converts each proposed branch as a graph clique and learns to prune some of the branch nodes. Experiments show its efficacy.

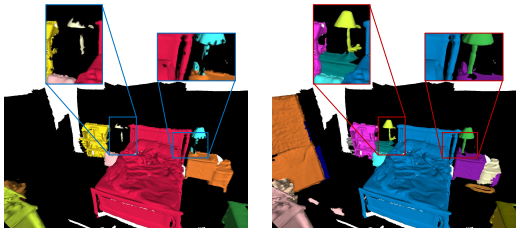

Figure 1. Visualization of example instance segmentation results from an existing, point-wise grouping method (PointGroup[16], left) and our SSTNet (right). Different colors represent segmented instances.

2. Related Works

In this section, we briefly review the literature of 3D segmentation, focusing on those relevant to elements of our proposed method.

3D Semantic Segmentation Establishing geometric regularities is essential to realize semantic segmentation for an irregular point cloud. Recent methods used projection[20], voxelization[12, 5] or local aggregation[26, 34] to perform brief regularization, while the subsequent semantic learning task is still challenging. Instead, superpoint-based[19, 18] methods aggregated the geometrically homogeneous points as superpoints to establish a certain degree of geometric regularities. Furthermore, superpoints become the mid-level shape representation to boild down the problem of instance segmentation as grouping the superpoints that belong to the same instance.

3D Instance Segmentation Considering bottom-up methods, which cluster results based on semantic segmentation. [38, 33, 17] heuristically[6, 3] clustered instance masks based on discriminative instance-level features[2]. Intuitively, PointGroup[16] utilized the adjacency of instance-wise coordinates. The above clustering results relied on the boundary conditions due to the lack of explicit boundary supervision. To address this issue, SSTNet combines the bottom-up clustering strategy with top-down traversal to realize end-to-end learning proposal generation.

Image Segmentation for Object Proposals To overcome the complexity caused by sliding windows[11, 28], segmentation-based[32, 27, 31] methods treat 2D detection as image segmentation, where the candidates are hypothesized from hierarchical image segmentation using an agglomeration manner. Furthermore, SSTNet involves a greedy agglomeration strategy and employs a learning splitting classifier to get rid of dependence on the times of agglomeration and generate precise mask results.

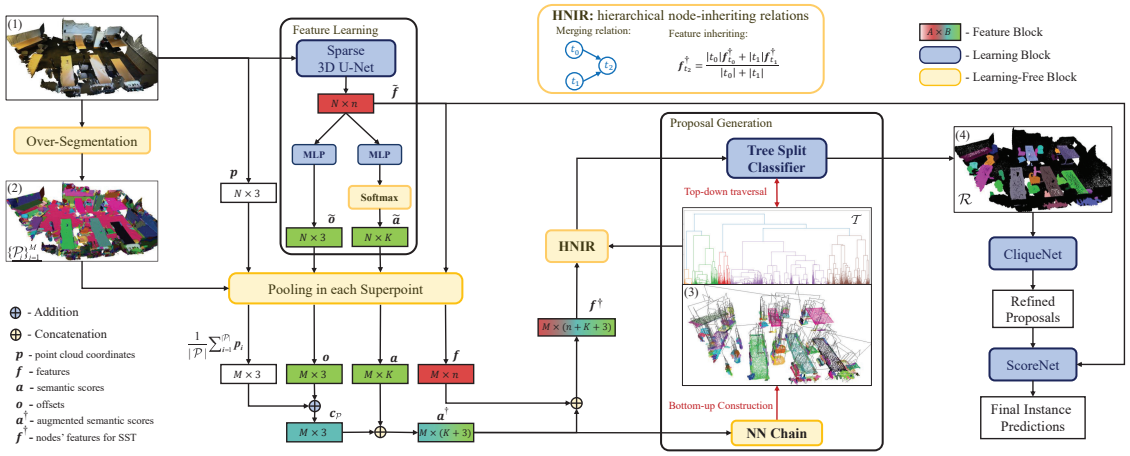

Figure 2. Overview of our proposed Semantic Superpoint Tree Network (SSTNet). Please refer to the main text for details of the individual modules. N is the number of scene points, M is the number of superpoints, K is the number of categories, and n is the dimension of output features from the backbone. $\tilde{\boldsymbol{f}}, \tilde{\boldsymbol{a}}, \tilde{\boldsymbol{o}}$ denote the point-wise features, semantic scores and offsets respectively. (1) input scene, (2) generated superpoint set $\{\mathcal{P}_i\}_{i=1}^{M}$, (3) foreground superpoints and Semantic Superpoint Tree(SST)$\mathcal{T}$, (4) generated proposals $\mathcal{R}$ after tree traversal and splitting. Nearest-neighbor chain (NN Chain) is the algorithm we use for efficiently constructing the tree.

3. Overview

Assume an input point set $\mathcal{I} = \{\boldsymbol{p} \in \mathbb{R}^3\}$ of reconstructed 3D scene from depth cameras via multi-view fusion techniques (e.g., SLAM [8, 14]), which contains an unknown number of object instances of K categories. The task is to segment out those points in $\mathcal{I}$ that define each of such instances, and is challenging as analyzed in Section 1. To alleviate the difficulty, we choose to establish a certain degree of geometric regularities for points in $\mathcal{I}$, by leveraging a mid-level shape representation called superpoints [19, 18] — for an input $\mathcal{I}$, superpoints define geometrically homogeneous neighborhoods of its local points, and are usually computed by over-segmenting $\mathcal{I}$ using graph partition. [1] With the set of superpoints $\{\mathcal{P}\}$ pre-computed from $\mathcal{I}$, the problem of instance segmentation boils down as grouping together spatially close $\{\mathcal{P}\}$ that belong to a same object instance and assigning them a semantic label. This is technically a clustering/grouping problem in the 3D space where superpoints live; given spatial compactness for superpoints on a same instance, it is natural to consider hierarchical clustering/grouping to achieve the goal. The strategy resembles those used for object proposals in 2D images via hierarchical image segmentation [32, 29, 13]. To implement the above idea, we propose in this work an end-to-end, hierarchical segmentation network that is trained to semantically group superpoints of a scene as object instances of pre-defined categories. The key in our network is an intermediate, semantic superpoint tree (SST); it is constructed based on the learned semantic features in the preceding network module, and will be traversed and split in the subsequent network module; we thus term our proposed method as Semantic Superpoint Tree Network (SSTNet). Fig. 2 gives the illustration.

More specifically, SSTNet starts with a **backbone** that learns point-wise feature $\tilde{\boldsymbol{f}} \in \mathbb{R}^n$ for each $\boldsymbol{p} \in \mathcal{I}$, which is then fed into a subsequent module of **semantic scoring** to output semantic score $\tilde{\boldsymbol{a}} \in [0,1]^K$ and offset $\tilde{\boldsymbol{o}} \in \mathbb{R}^3$, where $\tilde{\boldsymbol{a}}$ is a K-dimensional probability vector representing soft label prediction of the point $\boldsymbol{p}$, and $\tilde{\boldsymbol{o}}$ is a predicted coordinate offset relative to center of the instance to which $\boldsymbol{p}$ belongs. In parallel with this module we apply over-segmentation to $\mathcal{I}$ to have the superpoints $\{\mathcal{P}\}$; note that this is applied only once during network training. The point-wise $\{\tilde{\boldsymbol{f}}\}$, $\{\tilde{\boldsymbol{a}}\}$, and $\{\tilde{\boldsymbol{o}}\}$ are aggregated, via average pooling, inside each superpoint to form the superpoint feature $\boldsymbol{f} \in \mathbb{R}^n$, score $\boldsymbol{a} \in [0,1]^K$, and offset $\boldsymbol{o} \in \mathbb{R}^3$. Assume that a collection of superpoints are obtained from $\mathcal{I}$, we use the thus obtained $\{\boldsymbol{f}\}$, $\{\boldsymbol{a}\}$, and $\{\boldsymbol{o}\}$ for use in subsequent modules of the network. To achieve efficient training of SSTNet, we choose divisive grouping (i.e., in a top-down manner) after **construction of semantic superpoint tree** $\mathcal{T}$, instead of agglomerative grouping commonly used in hierarchical image segmentation [29, 32], which means that the whole tree $\mathcal{T}$ is first constructed whose leaf nodes represent individual superpoints. We then design a module of **tree traversal and splitting** that learns to hierarchically split the tree nodes; starting from the root, a proposal of object instance is formed as a tree branch when non-splitting decision is made at an intermediate tree node. We note that erroneous assignments between superpoints and object instances may occur during both stages of tree con-

[1] By using superpoints, we rely on the assumption that individual superpoints would not be across object boundaries; while this is not guaranteed, it is the cost that we would like to trade for the benefit of geometric regularities which the superpoints bring to the original, irregular point set $\mathcal{I}$.

struction and tree traversal and splitting. [25, 24] demonstrated the refinement can achieve higher accuracy for mesh reconstruction. Inspired by them, we design a subsequent refinement module termed **CliqueNet** to compensate for some of these errors. This module converts each proposal branch as a graph clique and learns to prune some of the branch nodes. We finally use a ScoreNet [16] to evaluate the generated proposals, which gives instance segmentation results of our SSTNet. The whole network is trained in an end-to-end manner, which, to the best of our knowledge, is the first one for the task of 3D instance segmentation on point set. The intermediate SST construction is highly efficient, whose computational complexity and running time are given in Section 4.2. Section 4 also presents individual modules of the network and compares with alternative designs.

4. Individual Modules of the Proposed Network

4.1. Backbone and Semantic Scoring

Assume that the input $\mathcal{I}$ contains N points. Given $\{\boldsymbol{p}_i \in \mathcal{I}\}_{i=1}^N$, we use a 3D convolutional backbone of U-Net style [30] to learn the point-wise features $\{\tilde{\boldsymbol{f}}_i \in \mathbb{R}^n\}_{i=1}^N$, whose layers are implemented as submanifold sparse convolution (SSC) or sparse convolution (SC) [12]. We give layer specifics in the supplementary material.

We obtain the semantic scoring $\{\tilde{\boldsymbol{a}}_i \in [0,1]^K\}_{i=1}^N$ and offset prediction $\{\tilde{\boldsymbol{o}}_i \in \mathbb{R}^3\}_{i=1}^N$ from $\{\tilde{\boldsymbol{f}}_i\}_{i=1}^N$, by employing two multi-layer perceptrons (MLPs) respectively. Let $\{\tilde{\boldsymbol{a}}_i^*\}_{i=1}^N$ denote the ground-truth semantic labels of the N points in the form of K-dimensional, one-hot vector. We use the following loss to train the MLP for semantic scoring

$$L_{\text{semantic}} = -\frac{1}{N}\sum_{i=1}^{N} \texttt{CE}(\tilde{\boldsymbol{a}}_i, \tilde{\boldsymbol{a}}_i^*) + 1 - \frac{2\sum_{i=1}^N \tilde{\boldsymbol{a}}_i^\top \tilde{\boldsymbol{a}}_i^*}{\sum_{i=1}^N \tilde{\boldsymbol{a}}_i^\top \tilde{\boldsymbol{a}}_i + \sum_{i=1}^N \tilde{\boldsymbol{a}}_i^{*\top} \tilde{\boldsymbol{a}}_i^*}, \quad (1)$$

where $\texttt{CE}(\cdot,\cdot)$ denotes the cross-entropy loss, and the remaining terms in (1) define a dice loss that alleviates the imbalance among the K categories [22]. Let $\boldsymbol{c}_{\boldsymbol{p}}^*$ denote the geometric center of the object instance to which any $\boldsymbol{p} \in \mathcal{I}$ belongs. We use the following loss to train the MLP for offset prediction

$$L_{\text{offset}} = \frac{1}{N'}\sum_{i=1}^{N} \|\tilde{\boldsymbol{o}}_i - (\boldsymbol{c}_{\boldsymbol{p}_i}^* - \boldsymbol{p}_i)\|_2 \cdot \mathbb{I}(\boldsymbol{p}_i) - \frac{1}{N'}\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{o}}_i^\top}{\|\tilde{\boldsymbol{o}}_i\|_2} \cdot \frac{\boldsymbol{c}_{\boldsymbol{p}_i}^* - \boldsymbol{p}_i}{\|\boldsymbol{c}_{\boldsymbol{p}_i}^* - \boldsymbol{p}_i\|_2} \cdot \mathbb{I}(\boldsymbol{p}_i), \quad (2)$$

where $\mathbb{I}(\boldsymbol{p}) \in \{0,1\}$ is an indicator function telling whether the point $\boldsymbol{p}$ belongs to any object instance, and $N' = \sum_{i=1}^N \mathbb{I}(\boldsymbol{p}_i)$ counts the number of such points. We give specifics of the two MLPs in the supplementary material.

4.2. Construction of Semantic Superpoint Tree

As stated in the preceding section, our construction of SST $\mathcal{T}$ is based on superpoints $\{\mathcal{P}\}$ pre-computed from the input $\mathcal{I}$; without loss of generality, we assume M ones are computed from $\mathcal{I}$. Features $\{\boldsymbol{f}_i \in \mathbb{R}^n\}_{i=1}^M$, semantic scores $\{\boldsymbol{a}_i \in [0,1]^K\}_{i=1}^M$, and offsets $\{\boldsymbol{o}_i \in \mathbb{R}^3\}_{i=1}^M$ at the superpoint level are obtained simply via average pooling over those point-wise ones inside each of superpoints $\{\mathcal{P}_i\}_{i=1}^M$.

Given the predicted $\{\boldsymbol{f}_i, \boldsymbol{a}_i, \boldsymbol{o}_i\}_{i=1}^M$ for $\{\mathcal{P}_i\}_{i=1}^M$, a tree can grow greedily [23], starting from merging the leaf nodes of superpoints (cf. Fig. 3 for an illustration). To define the linkage criteria, there exist many choices of similarity metric between any pair of $\mathcal{P}_i$ and $\mathcal{P}_j$. In this work, we choose semantic score and offset prediction over the triple $\{\boldsymbol{f}, \boldsymbol{a}, \boldsymbol{o}\}$ to define the metric. Specifically, for a superpoint $\mathcal{P}$, we first compute the predicted geometric center of a (possible) object instance to which it may belong as $\boldsymbol{c}_{\mathcal{P}} = \boldsymbol{o} + \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|} \boldsymbol{p}_i$, and then concatenate $\boldsymbol{a}^\dagger = [\boldsymbol{a}; \boldsymbol{c}_{\mathcal{P}}] \in \mathbb{R}^{K+3}$ [2]. We use the augmented $\boldsymbol{a}_i^\dagger$ and $\boldsymbol{a}_j^\dagger$ to represent $\mathcal{P}_i$ and $\mathcal{P}_j$, and compute the Euclidean distance $\|\boldsymbol{a}_i^\dagger - \boldsymbol{a}_j^\dagger\|$ as the linkage criterion that determines the ordering of pair-wise superpoint merging. Merging two superpoints $\mathcal{P}_i$ and $\mathcal{P}_j$ results in an intermediate tree node, denoted as $t \in \mathcal{T}$. We compute semantic score of t, via weighted averaging, as

$$\boldsymbol{a}_t = w_i \boldsymbol{a}_i + w_j \boldsymbol{a}_j, \quad (3)$$

where the weights w_i and w_j are proportional to the respective sizes of $\mathcal{P}_i$ and $\mathcal{P}_j$, i.e., $w_i = |\mathcal{P}_i|/(|\mathcal{P}_i| + |\mathcal{P}_j|)$ and $w_j = |\mathcal{P}_j|/(|\mathcal{P}_i| + |\mathcal{P}_j|)$. Offset prediction of t is computed similarly as $\boldsymbol{o}_t = w_i \boldsymbol{o}_i + w_j \boldsymbol{o}_j$. We then compute the augmented $\boldsymbol{a}_t^\dagger$ from the obtained $\boldsymbol{a}_t$ and $\boldsymbol{o}_t$. Note that we also compute the feature $\boldsymbol{f}_t = w_i \boldsymbol{f}_i + w_j \boldsymbol{f}_j$ for the node t, which will be used in the subsequent module of proposal generation via tree traversal and splitting. Given the augmented $\boldsymbol{a}_t^\dagger$ for any $t \in \mathcal{T}$ and the pair-wise similarity metric based on Euclidean distance, the tree can be constructed hierarchically, as illustrated in Fig. 3, whose depth ranges between $\log_2^M$ and $M-1$. For clarity, we write the M leaf nodes as $\{t_{\mathcal{P}_i} \in \mathcal{T}\}_{i=1}^M$ and any root or intermediate one as $t \in \mathcal{T}$.

Our use of the augmented semantic score $\boldsymbol{a}^\dagger = [\boldsymbol{a}; \boldsymbol{c}_{\mathcal{P}}]$ to represent each $t_{\mathcal{P}}$ (and t) is based on the argument that for any pair of $\mathcal{P}_i$ and $\mathcal{P}_j$ on a same instance, both their se-

[2] Considering the domain difference of $\boldsymbol{a} \in [0,1]^K$ and $\boldsymbol{c}_{\mathcal{P}} \in \mathbb{R}^3$, we ever try weighted concatenation such as $\boldsymbol{a}^\dagger = [\alpha\boldsymbol{a}; \beta\boldsymbol{c}_{\mathcal{P}}]$, where α and β are hyper-parameters. We end with the empirical setting of $\alpha = \beta = 1$, which gives good results in practice.

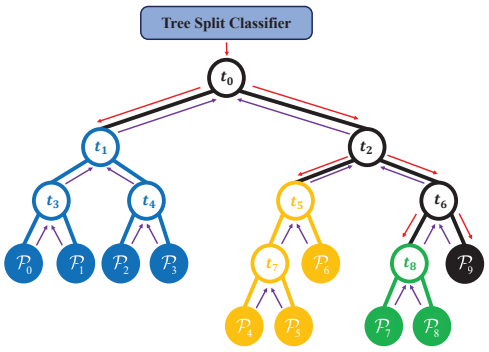

Figure 3. Illustration on the construction and traversal of semantic superpoint tree (SST). $\rightarrow$ represents the bottom-up construction process; $\rightarrow$ represents the top-down traversal process.

mantic scores and instance centers are expected to be consistent. Empirical results in Section 5.1 show that it gives better performance than alternative choices do, which verifies the hypothesis. We thus term the constructed $\mathcal{T}$ as semantic superpoint tree.

Remarks Given the M superpoints, the hierarchical tree construction described above has a complexity of $\mathcal{O}(M^3)$. Due to the linear feature inheritance (3) and the use of Euclidean distance as the similarity metric, construction of $\mathcal{T}$ can be made highly efficient by employing the fast algorithm of nearest-neighbor chain [23], which results in a same $\mathcal{T}$ at a complexity of $\mathcal{O}(M^2)$. On a machine running at 13 Hz, it takes $\sim$ 75 milliseconds per construction (e.g., scenes of ScanNet [7]), thus supporting online SST construction per iteration of network training.

4.3. Proposal Generation via Tree Traversal and Splitting

Given the constructed SST $\mathcal{T}$, our proposed SSTNet generates proposals of object instance by learning a binary classifier that traverses and splits nodes of $\mathcal{T}$. For any root or intermediate node t, denote its two child nodes as $s_1 \in \mathcal{T}$ and $s_2 \in \mathcal{T}$. Each t in fact defines a tree branch, denoted as $\mathcal{B}_t$, that contains leaf nodes of superpoints. As stated in Section 4.2, feature $\boldsymbol{f}_t$ and augmented score $\boldsymbol{a}_t^\dagger$ associated with each t have been hierarchically inherited from its contained superpoints. We use the concatenated $\boldsymbol{f}_t^\dagger = [\boldsymbol{f}_t; \boldsymbol{a}_t^\dagger] \in \mathbb{R}^{n+K+3}$ as feature of node t.

Denote the binary classifier to be learned as $\phi : \mathbb{R}^{n+K+3} \times \mathbb{R}^{n+K+3} \in (0, 1)$. Starting from the root node, we maintain a queue of tree traversal in a breadth-first manner. Let $\mathcal{Q}$ and $\mathcal{R}$ be two empty sets, and push the root into the queue $\mathcal{Q}$. A node t is to be split once $\phi(\boldsymbol{f}_{s_1}^\dagger, \boldsymbol{f}_{s_2}^\dagger) < 0.5$, i.e., the two child nodes of t are believed to belong to different object instances; we then push s_1 and s_2 into the queue $\mathcal{Q}$. Conversely, when $\phi(\boldsymbol{f}_{s_1}^\dagger, \boldsymbol{f}_{s_2}^\dagger) \geq 0.5$, we consider all superpoints contained in the tree branch $\mathcal{B}_t$ as a proposal of object instance, and push t into $\mathcal{R}$; we stop traversing the intermediate nodes contained in $\mathcal{B}_t$. Note that we have established an index table of the hierarchical node-inheriting relations when constructing $\mathcal{T}$, which supports efficient retrieval of both intermediate and leaf nodes/superpoints contained in any branch $\mathcal{B}_t$. All proposals of object instance would be obtained in $\mathcal{R}$ when the queue $\mathcal{Q}$ becomes empty. Algorithm 1 gives pseudo code of the above procedure.

In this work, we implement the classifier ϕ as an MLP, whose details are given in the supplementary material. To train ϕ, we define the instance-level, ground-truth labels for nodes of the tree as follows. Assume that a training scene $\mathcal{I}$ contains J object instances, which may belong to some of the K categories. For any superpoint $\mathcal{P}$ (i.e, a leaf node $t_\mathcal{P}$), we assign its instance-level, soft label $\boldsymbol{q}_\mathcal{P}^* \in [0, 1]^J$ according to what proportions its contained points belong to (some of) the J instances. The soft label $\boldsymbol{q}_t^* \in [0, 1]^J$ for any intermediate or root t is again hierarchically inherited, via weighted averaging, from those of superpoints, similar to the inheritance of features. Given that s_1 and s_2 are the two child nodes of t in $\mathcal{T}$, we use the following loss symmetric to them to train ϕ

$$L_{\text{splitting}} = \mathbb{E}_{t \in \mathcal{T}/\{t_{\mathcal{P}_i}\}_{i=1}^M} \frac{1}{2} [\texttt{BCE}(\phi(\boldsymbol{f}_{s_1}^\dagger, \boldsymbol{f}_{s_2}^\dagger), \boldsymbol{q}_{s_1}^{*\top} \boldsymbol{q}_{s_2}^*) + \texttt{BCE}(\phi(\boldsymbol{f}_{s_2}^\dagger, \boldsymbol{f}_{s_1}^\dagger), \boldsymbol{q}_{s_1}^{*\top} \boldsymbol{q}_{s_2}^*)], \quad (4)$$

where $\texttt{BCE}(\cdot, \cdot)$ denotes a binary cross-entropy loss, and $\boldsymbol{q}_{s_1}^{*\top} \boldsymbol{q}_{s_2}^* \in [0, 1]$ indicates, in a soft manner, whether the two child nodes belong to a same instance.

Remarks In the proposed SSTNet, we choose to first build the tree, as described in Section 4.2, and then learn to traverse and split tree nodes to generate instance proposals; in other words, we choose a strategy of divisive grouping, instead of an agglomerative one commonly used in hierarchical image segmentation [32, 29, 13]. Our motivation for such a design is mostly computational: by using nearest-neighbor chain [23], our tree construction has a complexity of $\mathcal{O}(M^2)$, and the tree traversal to propose all the branches of object instances has a complexity of $\mathcal{O}(M)$, giving rise to an overall complexity of $\mathcal{O}(M^2 + M)$; in contrast, learning to generate proposals in an agglomerative manner has an order-of-magnitude higher complexity of $\mathcal{O}(M^3)$.

4.4. CliqueNet for Refinement of Proposals

We note that in the forward pass of SSTNet, once a superpoint $\mathcal{P}$ truly on an object instance is constructed into a wrong branch $\mathcal{B}_t$ of SST $\mathcal{T}$, e.g., $\mathcal{B}_t$ corresponding to the background or a different instance, the mistake cannot be corrected. Nevertheless, when any branch $\mathcal{B}_t$ is proposed as an object instance, we have the chance to improve its score evaluation (cf. Section 4.5) by pruning its contained superpoints that may belong to other instances or the background.

Consider a proposed branch $\mathcal{B}_t$ consisting of M_t leaf nodes of superpoints. A straightforward way to implement the pruning is to concatenate feature representation

Algorithm 1 Pseudo code of proposal generation via tree traversal and splitting

Input: tree $\mathcal{T}$, node features $\{\boldsymbol{f}_i^\dagger\}_{i=1}^{|\mathcal{T}|}$, classifier ϕ;

initialize $\mathcal{R} = \emptyset$ to store proposals, and queue $\mathcal{Q} = \emptyset$;
push the root of $\mathcal{T}$ into $\mathcal{Q}$;
while - $\mathcal{Q}$.isempty() **do**
  $t = \mathcal{Q}$.dequeue()
  if - t.isleaf() **then**
    $\{s_1, s_2\} = t$.getchild()
    $\boldsymbol{f}_{s_1}^\dagger = s_1$.getfeature()
    $\boldsymbol{f}_{s_2}^\dagger = s_2$.getfeature()
    if $\phi(\boldsymbol{f}_{s_1}^\dagger, \boldsymbol{f}_{s_2}^\dagger) \geq 0.5$ **then**
      push t into $\mathcal{R}$, and $\mathcal{B}_t = t$.getbranch()
    else
      $\mathcal{Q}$.enqueue(s_1, s_2)
    end if
  end if
end while
return $\mathcal{R}$;

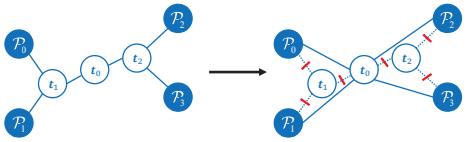

Figure 4. Illustration on conversion of a tree branch as a (graph) clique.

$\boldsymbol{f}_t^\dagger$ at node t with each $\boldsymbol{f}_\mathcal{P}^\dagger$ of $\{\boldsymbol{f}_{\mathcal{P}_i}^\dagger\}_{i=1}^{M_t}$, and to learn a binary classifier that decides whether the superpoint $\mathcal{P}$ should be removed. This, however, involves only the pairwise relation between $\boldsymbol{f}_t^\dagger$ and each $\boldsymbol{f}_\mathcal{P}^\dagger$, and we empirically find that it is less effective to prune erroneously assigned superpoints. In this work, we propose a more effective scheme, termed CliqueNet, that determines which superpoints to remove by learning the feature interactions among $\{\boldsymbol{f}_t^\dagger, \boldsymbol{f}_{\mathcal{P}_1}^\dagger, \ldots, \boldsymbol{f}_{\mathcal{P}_{M_t}}^\dagger\}$. Specifically, given the proposed branch $\mathcal{B}_t$ as shown in Fig. 4, we first connect the node t directly with individual leaf nodes/superpoints, which forms a clique $\mathcal{C}$ when thinking of the whole SST $\mathcal{T}$ as a graph — we note that the cliques formed for different proposed branches are independent with each other, i.e., they are not on a same graph. An adjacency matrix $\boldsymbol{A}_\mathcal{C} \in \{0,1\}^{(M_t+1)\times(M_t+1)}$ can be computed that specifies node connections of the clique. Let $\bar{\boldsymbol{A}}_\mathcal{C} = \boldsymbol{A}_\mathcal{C} + \boldsymbol{I}$, where $\boldsymbol{I}$ is an identity matrix, and write features of clique nodes compactly as $\boldsymbol{F}_\mathcal{C}^\dagger = [\boldsymbol{f}_t^\dagger, \boldsymbol{f}_{\mathcal{P}_1}^\dagger, \ldots, \boldsymbol{f}_{\mathcal{P}_{M_t}}^\dagger] \in \mathbb{R}^{(n+K+3)\times(M_t+1)}$. Denote the CliqueNet as a function ψ, the first layer of ψ computes

$$\texttt{ReLU}(\bar{\boldsymbol{D}}_\mathcal{C}^{-1/2}\bar{\boldsymbol{A}}_\mathcal{C}\bar{\boldsymbol{D}}_\mathcal{C}^{-1/2}\boldsymbol{F}_\mathcal{C}^\dagger\boldsymbol{W}_\psi^1), \tag{5}$$

where $\bar{\boldsymbol{D}}_\mathcal{C}$ is the diagonal degree matrix of $\bar{\boldsymbol{A}}_\mathcal{C}$, and $\boldsymbol{W}_\psi^1$ denotes weight matrix of the first layer of ψ. In this work, we use a three-layer CliqueNet whose specifics are given in the supplementary material.

CliqueNet outputs scores $\psi(\boldsymbol{F}_\mathcal{C}^\dagger, \boldsymbol{A}_\mathcal{C}) \in (0,1)^{M_t+1}$ defined respectively for the $M_t + 1$ nodes in $\mathcal{C}$. To train ψ, we impose supervision on each node pair of t and $\mathcal{P}_i$, $i \in \{1, \ldots, M_t\}$, giving rise to

$$L_{\text{refining}} = \frac{1}{M_t}\sum_{i=1}^{M_t}\texttt{BCE}(\psi(\boldsymbol{F}_\mathcal{C}^\dagger, \boldsymbol{A}_\mathcal{C}), \boldsymbol{q}_t^{*\top}\boldsymbol{q}_{\mathcal{P}_i}^*), \tag{6}$$

where the instance-level, soft labels $\boldsymbol{q}_t^* \in [0,1]^J$ and $\boldsymbol{q}_\mathcal{P}^* \in [0,1]^J$ are defined in Section 4.3.

4.5. Proposal Evaluation

Denote a proposed branch of object instance, after pruning some superpoints by CliqueNet, as $\mathcal{B}_t^-$, and assume that it contains N_t^- raw points. Recall that their point-wise features have been computed by the backbone of SSTNet. We write these features compactly as $\widetilde{\boldsymbol{F}}_{\mathcal{B}_t^-} = [\tilde{\boldsymbol{f}}_1, \ldots, \tilde{\boldsymbol{f}}_{N_t^-}] \in \mathbb{R}^{n\times N_t^-}$. We follow [16] and use a ScoreNet, denoted as ω, to evaluate the proposal. The ScoreNet is simply a miniature of U-Net; one may refer to [16] for the network details. Depending on the intersection-over-union (IoU) value with the ground-truth instances in the scene $\mathcal{I}$, we define label of the proposal as $v_t^* \in [0,1]$ (cf. the supplementary material for details of setting the v_t^* value), and train the ScoreNet with the following loss

$$L_{\text{evaluation}} = \frac{1}{|\mathcal{R}|}\sum_{t\in\mathcal{R}}\texttt{BCE}(\omega(\widetilde{\boldsymbol{F}}_{\mathcal{B}_t^-}), v_t^*), \tag{7}$$

where $|\mathcal{R}|$ is the number of proposals generated by our SSTNet (cf. Algorithm 1).

4.6. Training and Inference

We write our overall objective for training SSTNet as

$$L_{\text{SSTNet}} = L_{\text{semantic}} + L_{\text{offset}} + L_{\text{splitting}} + L_{\text{refining}} + L_{\text{evaluation}}. \tag{8}$$

Note that SSTNet is trained in a greedy, module-wise manner, which means that the individual loss terms applied to their respective modules are sequentially invoked into the overall loss (8). Although the tree $\mathcal{T}$ needs to be constructed in every forward pass of SSTNet, it is highly efficient as indicated by the complexity and practical running time given in preceding sections. The complexity of tree traversal for instance proposals is linear w.r.t. the number of superpoints; furthermore, once a proposal is formed at an intermediate tree node, it is not necessary to traverse all the descendant nodes. The inference is simply a same procedure as a forward pass of SSTNet training. Given the non-overlapping nature of our proposed object instances, post-processing steps such as non-maximum suppression are not necessary.

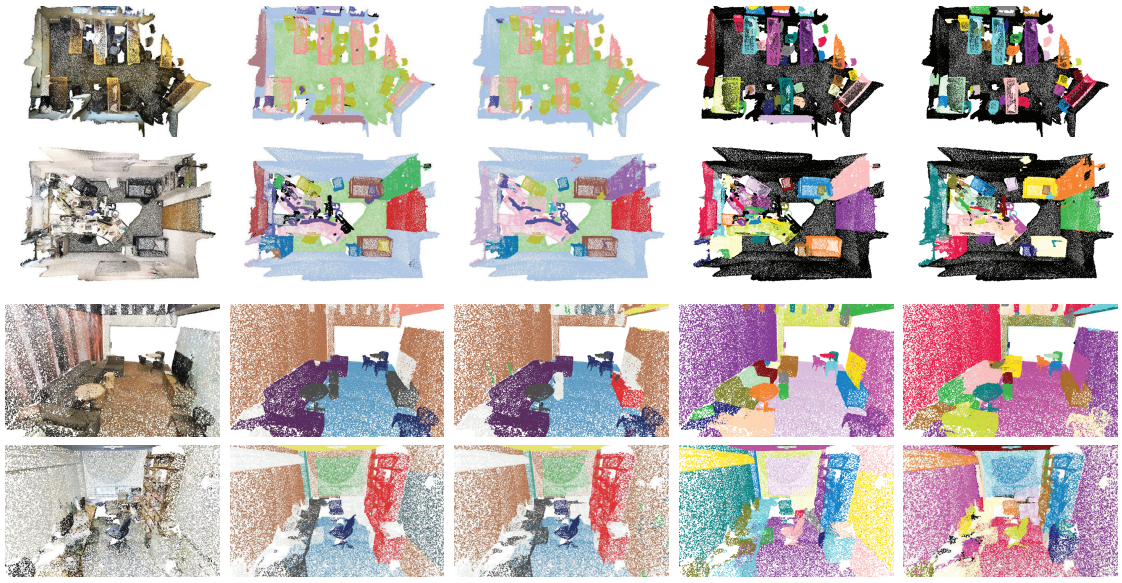

Figure 5. Visualization of the semantic and instance segmentation results on the validation set of ScanNet v2 (top) and S3DIS (bottom).

5. Experiments

Datasets We conduct experiments using the benchmark datasets of ScanNet (V2) [7] and S3DIS[1]. ScanNet has 1201 training, 312 validation, and 100 test scenes that contain object instances of 18 categories. Surface normals are also provided for each scene. We do analysis and ablation studies on its validation set, and submit our results to the hidden test set. S3DIS contains 6 large-scale indoor scenes with 13 object classes, we evaluate our model in the following aspects: (1) Area-5 is treated as the testing, while residuals are used for training, and (2) 6-fold cross validation that each area is treated as the testing once.

Implementation Details For each input scene, we concatenate the RGB values and point coordinates as the point-wise inputs of SSTNet. The network is trained using AdamW optimizer [21], with an initial learning rate of 1e-3 and weight decay of 1e-4; learning rates follow a polynomial learning rate policy. We set the batch size as 4. We pre-process scenes of S3DIS dataset by sub-sampling its points at a rate of 1/4. We employ a graph-based segmentation method [10] to generate superpoints for ScanNet scenes. For S3DIS, each scene is represented by colored point cloud and we employ SPP + SPG [19, 18] to generate its superpoints. Module and layer specifics of SSTNet are given in the supplementary material.

Evaluation Metrics Following the official ScanNet (V2) evaluation protocol, we report mean Average Precisions (mAPs) at different thresholds of IoU as the evaluation metric to compare different methods. The mAP@25 and mAP@50 denote the average precision scores with IoU thresholds respectively set to 25% and 50%, and the mAP averages the scores with IoU thresholds set from 50% to 95%, with a step size of 5%.

5.1. Ablation Studies and Analyses

We first conduct ablation studies to evaluate the efficacy of individual components proposed in SSTNet. These studies are conducted on the ScanNet (V2) dataset [7].

Analysis on Features for SST Construction The quality of SST depends on what features are used when contructing the tree. In this work, for a superpoint $\mathcal{P}$, we choose semantic score $\boldsymbol{a}$ and predicted instance center $\boldsymbol{c}_{\mathcal{P}}$ over the triple $\{\boldsymbol{f}, \boldsymbol{a}, \boldsymbol{c}_{\mathcal{P}}\}$, where $\boldsymbol{c}_{\mathcal{P}}$ is computed from the offset prediction $\boldsymbol{o}$ (cf. Section 4.2), and form the augmented $\boldsymbol{a}^{\dagger} = [\boldsymbol{a}; \boldsymbol{c}_{\mathcal{P}}]$ for SST construction. Results in Table 3 verify our argument that for any pair of superpoints on a same instance, their semantic scores and instance centers are expected to be consistent, while their superpoint-wise features are not necessarily to be similar.

Efficacy of Proposal Generation via Tree Traversal and Split Learning To verify the efficacy of our main proposal generation scheme via SST, we compare with two alternatives. The first alternative conducts the same traversal of SST but replaces the node-splitting classifier ϕ with a simple thresholding scheme, which we term as *SST-Thresholding*; to determine whether an intermediate node t is to be split into its child nodes s_1 and s_2, it thresholds the Euclidean distance $\|\boldsymbol{a}^{\dagger}_{s_1} - \boldsymbol{a}^{\dagger}_{s_2}\|_2$ where we optimally tune the thresholds for its best performance [3]. For

[3]We also try thresholding of the Euclidean distance $\|\boldsymbol{f}^{\dagger}_{s_1} - \boldsymbol{f}^{\dagger}_{s_2}\|_2$, where $\boldsymbol{f}^{\dagger}_{s_1} = [\boldsymbol{f}_{s_1}; \boldsymbol{a}^{\dagger}_{s_1}]$ and $\boldsymbol{f}^{\dagger}_{s_2}$ is computed similarly. It empirically gives even worse performance.

Method	**AP**	bath	bed	bkshf	cab	chair	cntr	curt	desk	door	ofurn	pic	fridg	showr	sink	sofa	table	toilet	wind
3D-MPA[9]	35.5	45.7	48.4	29.9	27.7	59.1	4.7	33.2	21.2	21.7	27.8	19.3	41.3	41.0	19.5	57.4	35.2	84.9	21.3
SSEN[38]	38.4	**85.2**	49.4	19.2	22.6	64.8	2.2	39.8	29.9	27.7	31.7	23.1	19.4	51.4	19.6	58.6	44.4	84.3	18.4
PE[37]	39.6	66.7	46.7	44.6	24.3	62.4	2.2	**57.7**	10.6	21.9	34.0	23.9	48.7	47.5	22.5	54.1	35.0	81.8	27.3
PointGroup[16]	40.7	63.9	49.6	41.5	24.3	64.5	2.1	57.0	11.4	21.1	35.9	21.7	42.8	66.0	25.6	56.2	34.1	86.0	**29.1**
OccuSeg[15]	48.6	80.2	53.6	42.8	**36.9**	**70.2**	**20.5**	33.1	**30.1**	**37.9**	**47.4**	32.7	43.7	**86.2**	48.5	60.1	39.4	84.6	27.3
Our SSTNet	**50.6**	73.8	**54.9**	**49.7**	31.6	69.3	17.8	37.7	19.8	33.0	46.3	**57.6**	**51.5**	85.7	**49.4**	**63.7**	**45.7**	**94.3**	29.0

Table 1. 3D instance segmentation on ScanNet (V2) benchmark (hidden testing set). Results of SSTNet are obtained by submitting onto the testing server the model trained on the ScanNet training set on January 4th, 2021.

Method	mAP	AP@50	AP@25
3D-MPA[9]	35.5	61.1	73.7
SSEN[38]	38.4	57.5	72.4
PE[37]	39.6	64.5	77.6
PointGroup[16]	40.7	63.6	77.8
OccuSeg[15]	48.6	67.2	74.2
Our SSTNet	**50.6**	**69.8**	**78.9**

Table 2. 3D instance segmentation on ScanNet (V2) benchmark (hidden testing set). Results of SSTNet are obtained by submitting onto the testing server the model trained on the ScanNet training set on January 4th, 2021.

Superpoint feature	Semantic score	Instance center	mAP	AP@50	AP@25
✓			40.1	55.3	66.2
	✓		43.5	59.8	72.2
		✓	47.3	61.6	71.4
✓	✓	✓	48.9	63.6	72.9
	✓	✓	**49.4**	**64.3**	**74.0**

Table 3. Analysis on the features used for SST construction. Experiments are conducted on the validation set of ScanNet (V2) [7]. Refer to Section 4.2 for how the three types of features are computed.

the second alternative, instead of relying on SST construction, given the M superpoints with its augmented semantic scores $\{\boldsymbol{a}_i^\dagger\}_{i=1}^M$, we first build a K-nearest-neighbor graph based on pair-wise Euclidean distances, and then train a classifier to decide whether some graph edges should be disconnected; the resulting, disconnected graph cliques are proposed as object instances; we term this alternative as *Superpoint Graph*, which can be interpreted as a flattened version of learning to propose object proposals. Table 4 shows that SST-thresholding performs the best at the low-precision metric of mAP@25, suggesting our construction of SST is indeed useful for generation of object proposals. On the averaged metric of mAP, SSTNet greatly outperforms the two alternatives.

Efficacy of the CliqueNet Refinement Ablation study on the efficacy of CliqueNet is presented in Table 5, which shows that pruning superpoints from proposed tree branches is effective at high-precision regimes of mAP metrics.

5.2. Results on the ScanNet Benchmark

We train SSTNet on the training set of ScanNet (V2) and submit our model onto the testing sever. Table 1 shows that on the leaderboard of ScanNet (V2) test set, SSTNet outperforms all existing methods. Results at the metrics of AP@25 and AP@50 are reported in Table 2.

Method	mAP	AP@50	AP@25
SST-Thresholding	46.3	62.6	**74.7**
Superpoint Graph	44.4	60.6	69.5
Our SSTNet	**49.4**	**64.3**	74.0

Table 4. Analyses on the efficacy of our proposal generation via traversal and node-splitting learning of semantic superpoint tree. Experiments are conducted on the validation set of ScanNet (V2) [7]. Reer to the main text for how the two alternatives are designed.

CliqueNet Refining	mAP	AP@50	AP@25
	49.4	64.3	**74.0**
✓	**50.0**	**64.7**	73.9

Table 5. Ablation study on the efficacy of CliqueNet for proposal refinement. Experiments are conducted on the validation set of ScanNet (V2) [7].

Method	mAP	AP@50	mPrec	mRec
ASIS[33]	-	-	55.3	42.4
PointGroup[16]	-	57.8	61.9	62.1
Our SSTNet	**42.7**	**59.3**	**65.5**	**64.2**
ASIS†[33]	-	-	63.6	47.5
3D-BoNet†[36]	-	-	65.6	47.6
OccuSeg†[15]	-	-	72.8	60.3
PointGroup†[16]	-	64.0	69.6	69.2
Our SSTNet†	**54.1**	**67.8**	**73.5**	**73.4**

Table 6. Results of instance segmentation on the S3DIS validation set. Methods without the † marks are evaluated on Area-5; methods marked with † are evaluated on 6-fold cross validation.

5.3. Results on S3DIS

Following the protocols used in previous methods, we employ the Area-5 and 6-fold cross validation, and use the mAP/AP@50/mean precision (mPrec)/mean recall(mRec) with IoU threshold 0.5 to evaluate SSTNet on the S3DIS dataset. One may refer to [33] for precise definitions of these metrics. Table 6 shows that SSTNet outperforms all exist methods, confirming the generalizable advantage of our proposed method.

Acknowledgement This work was partially supported by the Guangdong R&D key project of China (No.: 2019B010155001), the National Natural Science Foundation of China (No.: 61771201), and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No.: 2017ZT07X183).

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 2, 7

[2] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function, 2017. 2

[3] Ricardo Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. volume 7819, pages 160–172, 04 2013. 2

[4] Mingmei Cheng, Le Hui, Jian Xie, Jin an Yang, and Hui Kong. Cascaded non-local neural network for point cloud semantic segmentation. *arXiv preprint arXiv:2007.15488*, 2020. 1

[5] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 2

[6] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002. 2

[7] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 1, 2, 5, 7, 8

[8] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics 2017 (TOG)*, 2017. 1, 3

[9] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9031–9040, 2020. 1, 8

[10] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004. 7

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 2

[12] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 4

[13] Silvio Jamil F Guimarães, Jean Cousty, Yukiko Kenmochi, and Laurent Najman. A hierarchical image segmentation algorithm based on an observation scale. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 116–125. Springer, 2012. 3, 5

[14] L. Han and Lu Fang. Flashfusion: Real-time globally consistent dense 3d reconstruction using cpu computing. In *Robotics: Science and Systems*, 2018. 1, 3

[15] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2940–2949, 2020. 1, 8

[16] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 4, 6, 8

[17] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9256–9266, 2019. 2

[18] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7440–7449, 2019. 1, 2, 3, 7

[19] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 1, 2, 3, 7

[20] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017. 2

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 7

[22] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016. 4

[23] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011. 4, 5

[24] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4

[25] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9964–9973, 2019. 4

[26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R.

Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2

[27] Pekka Rantalankila, Juho Kannala, and Esa Rahtu. Generating object segmentation proposals using global and local search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 2

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 91–99, Cambridge, MA, USA, 2015. MIT Press. 2

[29] Zhile Ren and Gregory Shakhnarovich. Image segmentation by cascaded region agglomeration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 3, 5

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 4

[31] Chaoyang Wang, Long Zhao, Shuang Liang, Liqing Zhang, Jinyuan Jia, and Yichen Wei. Object proposal by multi-branch hierarchical segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[32] Huiqun Wang, Di Huang, Kui Jia, and Yunhong Wang. Hierarchical image segmentation ensemble for objectness in rgb-d images. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):93–103, 2019. 2, 3, 5

[33] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4096–4105, 2019. 2, 8

[34] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 2

[35] P WILLETT. Multidimensional clustering algorithms-murtagh, f, 1987. 2

[36] 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*, pages 6737–6746, 2019. 8

[37] Biao Zhang and Peter Wonka. Point cloud instance segmentation using probabilistic embeddings. *arXiv preprint arXiv:1912.00145*, 2019. 1, 8

[38] Dongsu Zhang, Junha Chun, Sang Cha, and Young Min Kim. Spatial semantic embedding network: Fast 3d instance segmentation with deep metric learning. In *arXiv preprint arXiv:2007.03169*, 2020. 2, 8