3D Spatial Recognition without Spatially Labeled 3D

Zhongzheng Ren[1,2*] Ishan Misra[1] Alexander G. Schwing[2] Rohit Girdhar[1]
[1]Facebook AI Research [2]University of Illinois at Urbana-Champaign
https://facebookresearch.github.io/WyPR

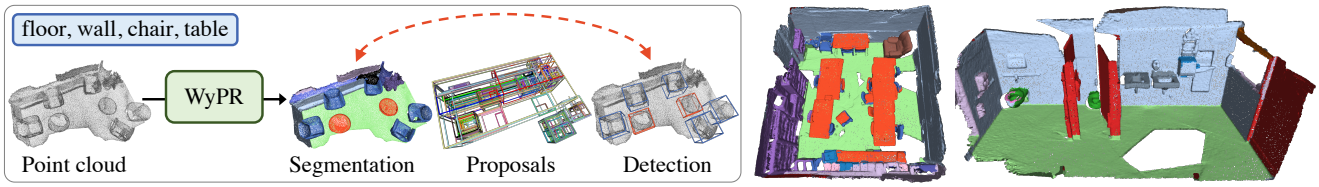

Figure 1: (Left) Our framework, WyPR, jointly learns semantic segmentation and object detection for point cloud data from only scene-level class tags. We find that encouraging consistency between the two tasks is key. (Right) Sample segmentation results from ScanNet val set, without seeing any point-level labels during training. Please refer to § 4.4 and Appendix F for more analysis and visualizations.

Abstract

*We introduce WyPR, a **W**eakly-supervised framework for **P**oint cloud **R**ecognition, requiring only scene-level class tags as supervision. WyPR jointly addresses three core 3D recognition tasks: point-level semantic segmentation, 3D proposal generation, and 3D object detection, coupling their predictions through self and cross-task consistency losses. We show that in conjunction with standard multiple-instance learning objectives, WyPR can detect and segment objects in point cloud data without access to any spatial labels at training time. We demonstrate its efficacy using the ScanNet and S3DIS datasets, outperforming prior state of the art on weakly-supervised segmentation by more than 6% mIoU. In addition, we set up the first benchmark for weakly-supervised 3D object detection on both datasets, where WyPR outperforms standard approaches and establishes strong baselines for future work.*

1. Introduction

Recognition (*i.e.*, segmentation and detection) of 3D objects is a key step towards scene understanding. With the recent development of consumer-level depth sensors (*e.g.*, LiDAR [13, 43]) and the advances of computer vision algorithms, 3D data collection has become more convenient and inexpensive. However, existing 3D recognition systems often fail to scale as they rely on strong supervision, such as point level semantic labels or 3D bounding boxes [9, 29, 32], which are time consuming to obtain. For example, while the popular large-scale indoor 3D dataset ScanNet [10] was collected by only 20 people, the annotation effort involved more than 500 annotators spending nearly 22.3 minutes per scan. Furthermore, due to the high annotation cost, existing 3D object detection datasets have limited themselves to a small number of object classes. This time consuming labeling process is a major bottleneck preventing the community from scaling 3D recognition.

Motivated by this observation, we study 3D weakly-supervised learning with only scene-level class tags available as supervision to train semantic segmentation and object detection models. Scene-level tags are very efficient to annotate, taking only a second or less for each object in the scene [36]. Hence, methods that rely on such supervision can be scaled more easily than those that rely on box-level supervision.

For this we develop the novel weakly-supervised framework called WyPR, shown in Fig. 1. Using just scene level tags, it jointly learns both segmentation of point cloud and detection of 3D boxes. Why should *joint* learning of segmentation and detection perform better than independently learning the two tasks? First, since these two tasks are related, joint training is mutually beneficial for representation learning. Second, these tasks naturally constrain each other, leading to effective self-supervised objectives that further improve performance. For example, the semantic labels of points within a bounding box should be consistent, and vice versa. Lastly, directly learning to regress to dimensions of 3D bounding boxes, as common in supervised ap-

*Work partly done during an internship at Facebook AI Research.

Methods	[46]	[56]	[59]	[51]	[53]	[33]	WyPR
Weak labels	2D boxes	2D inst seg	sparse label	2D sem seg	region & scene tags	scene tags	scene tags
Tasks	det	det	seg	seg	seg	det	det + seg
Dataset	indoor	outdoor	indoor & objects	indoor	indoor	outdoor	indoor

Table 1: Summary of closely related work in weakly-supervised 3D recognition. Compared to prior work, our proposed method (WyPR) uses the readily available scene tags, and jointly learns detection and segmentation in the more challenging indoor room setting.

proaches [28, 29, 39], is extremely challenging using weak labels. Learning weakly-supervised segmentation first permits a two-stage detection framework, where object proposals are generated bottom-up conditioned on segmentation prediction and further classified using a weakly-supervised detection algorithm.

To achieve this, WyPR operates on point cloud data of complex indoor scenes and combines a weakly-supervised semantic segmentation stage (§ 3.1) with a weakly-supervised object detection stage (§ 3.2). The latter takes as input the geometric representation of the input scene and a set of computed 3D proposals from **GSS**, our novel **G**eometric **S**elective **S**earch algorithm (§ 3.3). GSS uses local geometric structures (*e.g.*, planes) and the previously computed segmentation, for bottom-up proposal generation. Due to the uninformative nature of weak labels, weakly-supervised frameworks often suffer from noisy prediction and high variance. We address this by encouraging both cross-task and cross-transformation consistency through self-supervised objectives. We evaluate WyPR on standard 3D datasets, *i.e.*, ScanNet and S3DIS (§ 4), improving over prior work on weakly-supervised 3D segmentation by more than 6% mIoU, and establishing new benchmarks and strong baselines for weakly-supervised 3D detection.

Our contributions are as follows: 1) a novel point cloud framework to jointly learn weakly-supervised semantic segmentation and object detection, which significantly outperforms single task baselines; 2) an unsupervised 3D proposal generation algorithm, geometric selective search (GSS), for point cloud data; and 3) state-of-the-art results on weakly-supervised semantic segmentation, and benchmarks on weakly-supervised proposal generation and object detection.

2. Related work

3D datasets. Semantically labeled 3D data can be broadly classified into indoor [2, 5, 10, 41] and outdoor [7, 8, 14, 44] settings. ScanNet [10], a popular 3D detection and segmentation dataset, contains 20 classes labeled in 1500 scenes. While this dataset is large, it is small in comparison to 2D datasets, which reach tens of millions of images [21] and thousands of instance labels [17]. While the popularity of advanced 3D sensors [13, 43] could lead to a similar growth in 3D data, annotating that data would still be extremely time consuming. This underscores the need to develop weakly-supervised techniques for 3D recognition.

3D representations. 3D data is often represented via a point cloud, and processed using one of two main backbone architectures. The first [9, 15, 16, 37] projects points to intermediate volumetric grids, and then processes them using convolutional nets. These methods are efficient but suffer from information loss due to the discretization into voxels. The second operates directly on points [31, 32, 47, 52], processing them in parallel either using a pointwise MLP [31, 32], graph convolution [52], or point convolution [47]. Our method is compatible with either backbone architecture. We adopt PointNet++ [32] for experimentation.

3D tasks. Semantic segmentation [2, 6, 10], object detection [29, 30, 39, 42], and classification [57] are the standard recognition tasks defined on 3D data. For segmentation, the two most common tasks are point-level object parts segmentation [6] and scene object segmentation [2, 10], the latter of which we address in this work using weak supervision. For 3D object detection, standard techniques leverage either only a point cloud [29, 39, 61], or a point cloud together with the corresponding multi-view RGB images [18, 28, 30]. Unlike 2D where offline proposal generation methods [48, 64] are widely studied and generalize well to unseen datasets, 3D proposals generated from a point cloud are often trained in a supervised manner [20, 29, 39] and overfit to the training set. We propose an unsupervised 3D proposal generation algorithm GSS, which we further improve using weak supervision.

Weakly-supervised learning. Weak labels in the form of image-level class tags are widely studied in 2D tasks such as image localization [58, 63], semantic segmentation [27, 55], and object detection [3, 35, 45]. Prior work mostly formulates weakly-supervised learning as a multiple instance learning problem, where the target tasks are learned implicitly in a multi-label classification framework. Pipelined [40, 54] or end-to-end self-training modules [35, 45] have also been demonstrated to be beneficial.

Weakly-supervised learning in 3D. Compared to its 2D counterpart, weakly-supervised learning for 3D tasks is relatively unexplored. We summarize all relevant prior work in Tab. 1. For semantic segmentation, Wang *et al.* [51] leverage 2D segmentation as weak labels, Xu *et al.* [59] use a sparsely labeled point cloud, and Wei *et al.* [53] utilize both area and scene-level class tags during training. For

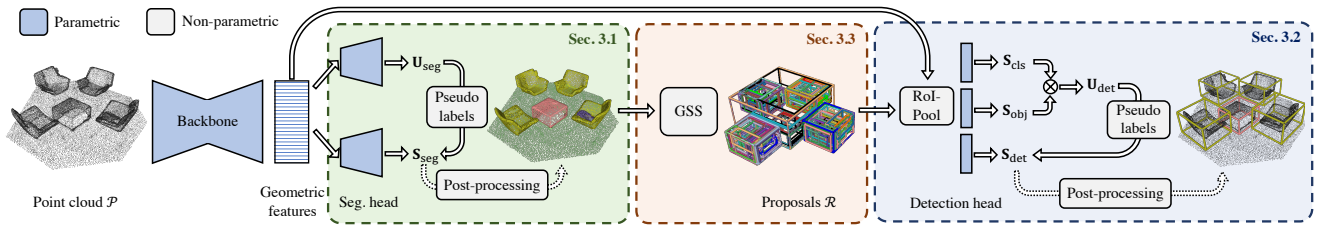

Figure 2: Approach Overview. A backbone network extracts geometric features which are used by the segmentation head to compute a point-level segmentation map. The segmentation map is passed into the 3D proposal generation module GSS, and the resulting proposals along with original features are used to detect 3D object instances. Through a series of self and cross-task consistency losses along with multiple-instance learning objectives, WyPR is trained end-to-end using only scene-level tags as supervision.

object detection, recent work uses small sets of labeled 3D data [24, 46, 62], 2D instance segmentation [56], and click annotation [24] as supervision. However, obtaining these labels is still time consuming. A closely related concurrent work [33] focuses on autonomous driving, building upon a small number of relatively easy objects (*e.g.*, car, pedestrian) while still using image data. In contrast, we focus on complex indoor scenes, exclusively relying on the 3D point cloud, *i.e.*, no images are required.

Multi-task learning. Multi-task learning [4] has been widely studied for various vision tasks [12, 19, 25, 34]. It is of particular importance for weakly-supervised [23, 36] or self-supervised 2D object detection [11, 34] as multi-tasking provides mutual regularization and hence better representation learning. For detection and segmentation, prior work has studied joint training with 2D data [23] or supervised 3D data [49]. In this paper, we develop a novel framework for learning both tasks under weak supervision.

3. WyPR

Our goal is to use weak supervision in the form of scene-level tags and learn a joint 3D segmentation and detection model, which we refer to as **WyPR**. Specifically, we assume availability of data $\mathcal{D} = \{(\mathcal{P}, \mathbf{y})\}$ of point cloud $\mathcal{P}$ and corresponding scene-level tags $\mathbf{y} \in \{0, 1\}^C$, which indicate absence or presence of the C object classes. $\mathcal{P}$ is a set of six-dimensional points $\mathbf{p} \in \mathcal{P}$, represented by their 3D location and RGB-color. Note, $\mathbf{y}$ only indicates existence of objects in the scene and does not contain any information about per-point semantic labels or object locations.

Approach overview. Fig. 2 provides an overview of our model which consists of three parametric modules: a backbone network, followed by a segmentation and a detection head. We first extract geometric features from the input point cloud using the backbone network. Specifically, we use the variant of PointNet++ [32] following VoteNet [29], which is an encoder-decoder network with skip connections. The features are then fed into the segmentation and detection modules. The segmentation module assigns each point from the input point cloud $\mathcal{P}$ to one of C classes. We use this segmentation output to generate 3D region proposals $\mathcal{R}$ that are likely to contain objects in the scene. Finally, the detection module classifies each proposal into either one of C classes or background (not an object) class, using the backbone features corresponding to that proposal.

Notation. We denote the output of the segmentation module as $\mathbf{S}_{\text{seg}} \in \mathbb{R}^{|\mathcal{P}| \times C}$, where the rows represent the score logits over the C classes for all points $\mathcal{P}$. The detection module produces a score matrix $\mathbf{S}_{\text{det}} \in \mathbb{R}^{|\mathcal{R}| \times (C+1)}$ over the C classes and background for all 3D proposals $\mathcal{R}$. For readability, we also use $\mathbf{p}, \mathbf{r}$ as indices into $\mathbf{S}_{\text{seg}}, \mathbf{S}_{\text{det}}$ in the following sections.

3.1. Weakly-supervised 3D semantic segmentation

The segmentation module consists of two identical heads that independently process the backbone features using a series of unit PointNet [31] and nearest neighbor upsampling layers (Fig. 2 green region). The output from these heads are two score matrices $\mathbf{U}_{\text{seg}}, \mathbf{S}_{\text{seg}} \in \mathbb{R}^{|\mathcal{P}| \times C}$ respectively, containing logits over C object classes for all points $\mathbf{p} \in \mathcal{P}$. The parameters of the backbone and the segmentation module are optimized to minimize a composed loss

$$\mathcal{L}_{\text{seg}} = \mathcal{L}_{\text{seg}}^{\text{MIL}} + \mathcal{L}_{\text{seg}}^{\text{SELF}} + \mathcal{L}_{\text{seg}}^{\text{CST}} + \mathcal{L}_{\text{d}\to\text{s}} + \mathcal{L}_{\text{smooth}}, \quad (1)$$

where $\mathcal{L}_{\text{seg}}^{\text{MIL}}$ denotes a multiple-instance learning (MIL) loss, $\mathcal{L}_{\text{seg}}^{\text{SELF}}$ denotes a self-training loss, $\mathcal{L}_{\text{seg}}^{\text{CST}}$ and $\mathcal{L}_{\text{d}\to\text{s}}$ represent consistency loss across geometric transformations and tasks respectively, and $\mathcal{L}_{\text{smooth}}$ is a smoothness regularization loss. We describe the individual loss terms next.

MIL loss. The multiple-instance learning loss [54, 55] encourages to learn the per-point semantic segmentation logits without access to point-level supervision. We first convert the per-point logits $\mathbf{U}_{\text{seg}}$ into a scene-level prediction ϕ via average pooling and a sigmoid normalization

$$\phi[c] = \text{sigmoid}\left(\frac{1}{|\mathcal{P}|}\sum_{\mathbf{p}\in\mathcal{P}} \mathbf{U}_{\text{seg}}[\mathbf{p}, c]\right). \quad (2)$$

The scene-level prediction ϕ is then supervised using the

Algorithm 1 Segmentation pseudo label generation

Input: class label $\mathbf{y}$, segmentation logits $\mathbf{U}_{\text{seg}}$, threshold p_1
Output: pseudo label $\hat{\mathbf{Y}}_{\text{seg}}$

$\hat{\mathbf{Y}}_{\text{seg}} = \mathbf{0}$ ▷ initialize to zero matrix
for each point $\mathbf{p} \in \mathcal{P}$ **do**
  $c = \text{argmax}(\mathbf{y} \odot \mathbf{U}_{\text{seg}}[\mathbf{p}, :])$ ▷ element-wise product
  $\hat{\mathbf{Y}}_{\text{seg}}[\mathbf{p}, c] = 1$
for ground-truth class c where $\mathbf{y}[c] = 1$ **do**
  $\mathcal{P}'[c] \leftarrow$ lowest p_1-th percentile of $\hat{\mathbf{Y}}_{\text{seg}}[:, c]$
  $\hat{\mathbf{Y}}_{\text{seg}}[\mathbf{p}, c] = 0 \; \forall \mathbf{p} \in \mathcal{P}'[c]$ ▷ ignore points with low score

Algorithm 2 Detection pseudo label generation

Input: class label $\mathbf{y}$, detection logits $\mathbf{U}_{\text{det}}$, proposals $\mathcal{R}$, threshold τ, p_2
Output: pseudo label $\hat{\mathbf{Y}}_{\text{det}}$

for ground-truth class c where $\mathbf{y}[c] = 1$ **do**
  $\hat{\mathbf{Y}}_{\text{det}} = \mathbf{0}$ ▷ initialize to zero matrix
  $\mathcal{R}'[c] \leftarrow$ top p_2-th percentile of $\mathbf{U}_{\text{det}}[:, c]$ ▷ $\mathcal{R}'[c]$ is descending
  $\mathcal{R}^*[c] \leftarrow \mathbf{r}'_1$ ▷ save 1st RoI (top-scoring) $\mathbf{r}'_1 \in \mathcal{R}'[c]$
  for $i \in \{2, \cdots, |\mathcal{R}'[c]|\}$ **do** ▷ start from the 2nd highest
    $\mathcal{R}^*[c] \leftarrow \mathbf{r}'_i$ **if** $\text{IoU}(\mathbf{r}'_i, \hat{\mathbf{r}}) < \tau \; \forall \hat{\mathbf{r}} \in \mathcal{R}^*[c]$
  $\hat{\mathbf{Y}}_{\text{det}}[\mathbf{r}, c] = 1 \; \forall \mathbf{r} \in \mathcal{R}^*[c]$

scene-level tags $\mathbf{y}$ using the binary cross-entropy loss

$$\mathcal{L}_{\text{seg}}^{\text{MIL}} = -\sum_{c=1}^{C} \mathbf{y}[c] \log \boldsymbol{\phi}[c] - (1 - \mathbf{y}[c]) \log(1 - \boldsymbol{\phi}[c]). \quad (3)$$

Self-training loss. Inspired by the success of self-training in weakly-supervised detection [35, 45, 50], we further incorporate a self-training loss. The previously computed segmentation logits $\mathbf{U}_{\text{seg}}$ are used to supervise the final segmentation logits $\mathbf{S}_{\text{seg}}$ via a cross-entropy loss

$$\mathcal{L}_{\text{seg}}^{\text{SELF}} = -\frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \sum_{c=1}^{C} \hat{\mathbf{Y}}_{\text{seg}}[\mathbf{p}, c] \log \boldsymbol{\psi}[\mathbf{p}, c], \quad (4)$$

where $\boldsymbol{\psi}[\mathbf{p}, c] = \text{softmax}(\mathbf{S}_{\text{seg}}[\mathbf{p}, c])$ denotes the probability of point $\mathbf{p}$ belonging to class c, and $\hat{\mathbf{Y}}_{\text{seg}}[\mathbf{p}, c] \in \{0, 1\}$ is the point-level pseudo class label inferred from score matrix $\mathbf{U}_{\text{seg}}$. We detail the process of computing the pseudo label in Alg. 1. Intuitively, the algorithm ignores noisy predictions in $\mathbf{U}_{\text{seg}}$, leading to robust self-supervision for $\mathbf{S}_{\text{seg}}$.

Cross-transformation consistency loss. In addition, we use $\mathcal{L}_{\text{seg}}^{\text{CST}}$ to encourage that the segmentation predictions are consistent across data augmentations $\mathcal{T}$. We obtain an augmented point cloud $\tilde{\mathcal{P}} = \mathcal{T}(\mathcal{P})$ by changing the original scene $\mathcal{P}$ via standard augmentations (see § 4 and Appendix C.1 for details). We predict the semantic segmentation $\tilde{\mathbf{S}}_{\text{seg}}$ on this transformed point cloud. The consistency loss is then formulated as

$$\mathcal{L}_{\text{seg}}^{\text{CST}} = \frac{1}{|\mathcal{P} \cap \tilde{\mathcal{P}}|} \sum_{\mathbf{p} \in \mathcal{P} \cap \tilde{\mathcal{P}}} D_{\text{KL}} \left(\boldsymbol{\psi}[\mathbf{p}, \cdot] \,||\, \tilde{\boldsymbol{\psi}}[\mathbf{p}, \cdot] \right), \quad (5)$$

where $\boldsymbol{\psi}[\mathbf{p}, c] = \text{softmax}(\mathbf{S}_{\text{seg}}[\mathbf{p}, c])$ and $\tilde{\boldsymbol{\psi}}[\mathbf{p}, c] = \text{softmax}(\tilde{\mathbf{S}}_{\text{seg}}[\mathbf{p}, c])$ are the probabilities of the point $\mathbf{p}$ belonging to class c, and D_{KL} is the KL divergence over C classes for points that are common across the transformation. This loss encourages the probability distributions for semantic segmentation of corresponding points within the point cloud $\mathcal{P}$ and $\tilde{\mathcal{P}}$ to match.

Cross-task consistency loss. We further employ a cross-task regularization term $\mathcal{L}_{\text{d}\to\text{s}}$. It uses the detection results to refine the segmentation prediction. Intuitively, all points within a confident bounding box prediction should have the same semantic label. Assume we have access to a set of confident bounding boxes $\mathbf{r} \in \mathcal{R}^*$ and their corresponding predicted score matrix $\mathbf{S}_{\text{det}} \in \mathbb{R}^{|\mathcal{R}^*| \times (C+1)}$. Using this information, we encourage consistency via a cross entropy loss on the point-level predictions, with the box-level prediction as a soft target

$$\mathcal{L}_{\text{d}\to\text{s}} = -\frac{1}{|\mathcal{R}^*|} \sum_{\mathbf{r} \in \mathcal{R}^*} \frac{1}{|\mathcal{P}^r|} \sum_{\mathbf{p} \in \mathcal{P}^{\mathbf{r}}} \sum_{c=1}^{C} \boldsymbol{\xi}[\mathbf{r}, c] \log \boldsymbol{\psi}[\mathbf{p}, c], \quad (6)$$

where $\boldsymbol{\psi}[\mathbf{p}, c]$ is the point probability from Eq. (4), $\boldsymbol{\xi}[\mathbf{r}, c] = \text{softmax}(\mathbf{S}_{\text{det}}[\mathbf{r}, c])$ denotes the probability of proposal $\mathbf{r}$ belonging to object class c, and $\mathcal{P}^{\mathbf{r}}$ denotes the set of points within proposal $\mathbf{r}$. In practice, the confident bounding boxes $\mathcal{R}^*$ are obtained from Alg. 2, discussed later in § 3.2.

Smoothness regularization. Finally, we compute $\mathcal{L}_{\text{smooth}}$ to encourage local smoothness. We first detect a set of planes $\mathcal{G}$ from input point cloud $\mathcal{P}$ using an unsupervised off-the-shelf shape detection algorithm [22] detailed in Appendix B. We then compute

$$\mathcal{L}_{\text{smooth}} = -\sum_{i=1}^{|\mathcal{G}|} \frac{1}{|\mathcal{G}[i]|} \sum_{\mathbf{p} \in \mathcal{G}[i]} \sum_{c=1}^{C} \bar{\boldsymbol{\psi}}[c] \log \boldsymbol{\psi}[\mathbf{p}, c], \quad (7)$$

where $\bar{\boldsymbol{\psi}}[c] = \frac{\sum_{\mathbf{p} \in \mathcal{G}[i]} \boldsymbol{\psi}[\mathbf{p}, c]}{|\mathcal{G}[i]|}$ is the mean probability of all the points which lie inside plane $\mathcal{G}[i]$ for class c.

3.2. Weakly-supervised 3D object detection

Our object detection module assumes access to a set of 3D region proposals $\mathcal{R}$ (discussed in § 3.3) and uses the backbone features to classify the proposals into one of the C object classes or background (Fig. 2 blue region **AS:** not the same blue as in the figure). Each region of interest (RoI) $\mathbf{r} \in \mathbb{R}^6$ is represented by a six-dimensional vector denoting its center location and its width, height and length. We extract RoI features by averaging the backbone features of all the points within each proposal. Inspired by prior 2D literature [3], we use three separate linear layers to extract classification logits $\mathbf{S}_{\text{cls}} \in \mathbb{R}^{|\mathcal{R}| \times (C+1)}$, objectness logits $\mathbf{S}_{\text{obj}} \in \mathbb{R}^{|\mathcal{R}| \times (C+1)}$, and final detection logits

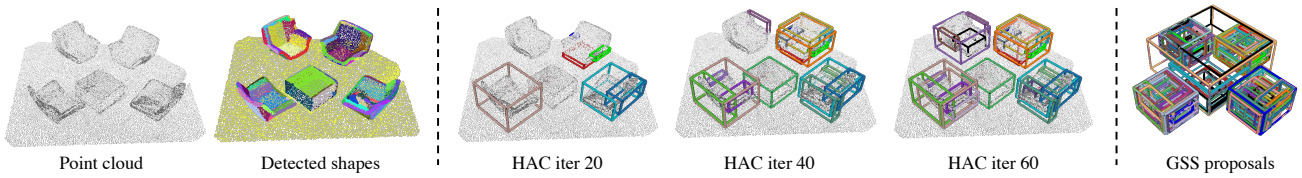

Figure 3: Geometric Selective Search (GSS). Our algorithm takes as input the point cloud and detected planes (left column). It then hierarchically groups the neighboring planes into sub-regions and generates 3D proposals for the combined regions (middle column). We run the algorithm multiple times with different grouping criteria to encourage high recall of final output proposals (right column).

$\mathbf{S}_{\text{det}} \in \mathbb{R}^{|\mathcal{R}|\times(C+1)}$ from the RoI features. As in [3], we normalize $\mathbf{S}_{\text{cls}}$ using a softmax function over rows to obtain the probability over object classes for each proposal. Similarly, we normalize $\mathbf{S}_{\text{obj}}$ over columns to obtain a probability over proposals for each class. Intuitively, $\mathbf{S}_{\text{cls}}[\mathbf{r}, c]$ represents the probability of region $\mathbf{r}$ being classified as class c, and $\mathbf{S}_{\text{obj}}[\mathbf{r}, c]$ is the probability of detecting region $\mathbf{r}$ for class c. We aggregate the evidence from both matrices via element-wise multiplication to obtain the score matrix $\mathbf{U}_{\text{det}} = \mathbf{S}_{\text{cls}} \odot \mathbf{S}_{\text{obj}}$. Similar to the self-training discussed earlier for segmentation, we infer pseudo-labels from $\mathbf{U}_{\text{det}}$ to supervise the final detection logits $\mathbf{S}_{\text{det}}$. We learn the backbone and the detection module using the loss

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{det}}^{\text{MIL}} + \mathcal{L}_{\text{det}}^{\text{SELF}} + \mathcal{L}_{\text{det}}^{\text{CST}}, \tag{8}$$

where $\mathcal{L}_{\text{det}}^{\text{MIL}}$ is a MIL objective for detection, $\mathcal{L}_{\text{det}}^{\text{SELF}}$ is a self-training loss, and $\mathcal{L}_{\text{det}}^{\text{CST}}$ is the cross-transformation consistency loss. All the terms are described next.

MIL loss. Similar to the segmentation head, the multiple instance learning (MIL) loss for detection is

$$\mathcal{L}_{\text{det}}^{\text{MIL}} = -\sum_{c=1}^{C+1} \mathbf{y}[c] \log \boldsymbol{\mu}[c] - (1-\mathbf{y}[c]) \log(1-\boldsymbol{\mu}[c]), \tag{9}$$

where $\boldsymbol{\mu}[c] = \sum_{\mathbf{r}\in\mathcal{R}} \mathbf{U}_{\text{det}}[\mathbf{r}, c]$ is the row-sum of the score matrix $\mathbf{U}_{\text{det}}$ for class c. This sum-pooling operation aggregates RoI scores into a scene-level score vector $\boldsymbol{\mu}$, which is used for multi-label scene classification.

Self-training loss. As done before for segmentation, we incorporate a self-training loss for detection as well. The final detection logits $\mathbf{S}_{\text{det}}$ are supervised by $\mathbf{U}_{\text{det}}$ via

$$\mathcal{L}_{\text{det}}^{\text{SELF}} = -\frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} \sum_{c=1}^{C+1} \hat{\mathbf{Y}}_{\text{det}}[\mathbf{r}, c] \log \boldsymbol{\xi}[\mathbf{r}, c], \tag{10}$$

where $\boldsymbol{\xi}[\mathbf{r}, c] = \text{softmax}(\mathbf{S}_{\text{det}}[\mathbf{r}, c])$ denotes the probability of proposal $\mathbf{r}$ belonging to object class c, and $\hat{\mathbf{Y}}_{\text{det}}[\mathbf{r}, c] \in \{0, 1\}$ is the RoI pseudo class label inferred from score matrix $\mathbf{U}_{\text{det}}$. The pseudo label $\hat{\mathbf{Y}}_{\text{det}}$ is computed using Alg. 2. Conceptually, this algorithm selects a set of confident yet diverse predictions as the pseudo labels for self-training.

Cross-transformation consistency loss. Following the consistency loss for semantic segmentation (Eq. (5)), we encourage detection predictions to be consistent under transformation $\mathcal{T}$ via

$$\mathcal{L}_{\text{det}}^{\text{CST}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} D_{\text{KL}} \left(\boldsymbol{\xi}[\mathbf{r}, \cdot] \;||\; \tilde{\boldsymbol{\xi}}[\mathcal{T}(\mathbf{r}), \cdot] \right), \tag{11}$$

where $\boldsymbol{\xi}[\mathbf{r}, c]$ refers to the RoI probability introduced in Eq. (10), $\tilde{\boldsymbol{\xi}}[\mathcal{T}(\mathbf{r}), c]$ denotes the RoI probability obtained from the transformed input $\tilde{\mathcal{P}} = \mathcal{T}(\mathcal{P})$ and proposal $\mathcal{T}(\mathbf{r})$ via the same backbone and detection module.

3.3. Geometric Selective Search (GSS)

The detection module uses a proposal set $\mathcal{R}$ as input. In weakly-supervised learning, proposals are necessary because it is not possible to mimic supervised methods that directly predict 3D bounding box parameters (*e.g.*, size and location). The key observation which inspires our novel 3D proposal generation algorithm is that most indoor objects are rigid and mainly consist of basic geometric structures (*e.g.*, planes, cylinders, spheres). We thus devise a bottom-up solution termed Geometric Selective Search (GSS), first detecting basic geometric shapes which are then grouped hierarchically to form 3D proposals.

Given an input point cloud with unoriented normals, we adopt a region-growing-based method [22, 26] for detecting primitive shapes (*e.g.*, planes) as shown in Fig. 3 left. We choose region-growing over the popular RANSAC-based methods [38] because 1) it is deterministic; 2) it performs better in the presence of large scenes with fine-grained details. We then apply hierarchical agglomerative clustering (HAC) to iteratively group the detected shapes into sub-regions. In each HAC iteration, we compute the similarity score s between all spatially overlapping sub-regions and group the two most similar regions. We iterate until no neighbors can be found or only one region is left. Every time we generate a new region, we also compute the axis-aligned bounding boxes of the new region and add it into the proposal pool. We illustrate the process of growing the proposal pool during HAC in Fig. 3 (middle columns).

In order to pick which two regions $\mathbf{n}_i, \mathbf{n}_j$ to group, HAC uses a similarity score

$$s(\mathbf{n}_i, \mathbf{n}_j) = w_1 s_{\text{size}} + w_2 s_{\text{volume}} + w_3 s_{\text{fill}} + w_4 s_{\text{seg}}, \tag{12}$$

where $w_i \in \{0, 1\} \;\; \forall i \in \{1, \cdots, 4\}$ are binary indicators. s_{size} and $s_{\text{volume}} \in [0, 1]$ measure size and volume

compatibility and encourage small regions to merge early; $s_{\text{fill}} \in [0, 1]$ measures how well two regions are aligned. Besides similarities of low-level cues, we also measure high-level semantic similarities by incorporating segmentation similarity $s_{\text{seg}} \in [0, 1]$. This score is the histogram intersection of the normalized C-dimensional class histogram of two regions' points. The class labels of these points are computed from $\mathbf{S}_{\text{seg}}$ using the inference procedure described in § 4. Please see Appendix A for the exact formulation of the above five metrics. During training, as the segmentation module improves, s_{seg} increasingly prefers grouping regions which correspond to the same object. A similar idea to compute proposals from segmentations has also been widely adopted in the 2D case [1, 48]. In practice, we find that multiple runs of HAC with different w_i values, results in a more diverse set of proposals as each run uses a different weighted similarity measure. We provide the values of w_i for different runs in Appendix A.

GSS can be made **completely unsupervised** by removing the segmentation term $s_{\text{seg}}(\mathbf{n}_i, \mathbf{n}_j)$ from Eq. (13). This variant is also valuable as the proposals can be precomputed offline and are of decent quality (verified in § 4). These proposals are independent of any specific supervision and can benefit various downstream unsupervised or weakly-supervised 3D recognition tasks, akin to Selective Search [48] or Edge Boxes [64] in 2D. This is distinct from existing 3D proposal techniques that either use 2D image cues [33] or full bounding box supervision [28, 29].

4. Experiments

We empirically evaluate WyPR on two standard 3D benchmarks. We first provide the key implementation details (more details in Appendix C) and describe the baseline methods we compare to (§ 4.1). We then present the quantitative results (§ 4.2 and 4.3), ablate our design choices and present qualitative results (§ 4.4).

Input. Our network takes as input a fixed-size point cloud, where 40K points are randomly sub-sampled from the original scan. In addition to using color (RGB) and coordinates (XYZ) as input features, following [29], we include surface normal and a height feature of each point.

Augmentation. We augment the input point cloud at two places in our framework: (1) data augmentation at the input, and (2) to compute the consistency loss in Eq. (5) and Eq. (11). In practice, we find it beneficial to apply different geometric transformations for the above two purposes. To augment the input, we follow [29] and use random sub-sampling of 40K points, random flipping in both horizontal and vertical directions, and random rotation of $[-5, 5]$ degrees around the upright-axis. To compute the consistency loss, we use random flipping, point jittering, random rotation with an angle uniform in $[0, 30]$ degrees around the upright-axis, random scaling by a factor from $[0.8, 1.2]$, and point dropout ($p = 0.1$). Finally, we also find that jittering the point cloud is crucial to obtain good proposals for noisy point clouds (analyzed in § 4.4).

Methods	evaluation split	mIoU
Weakly-supervised methods		
PCAM [53]	train	22.1
MPRM [53]	train	24.4
WyPR	train	**30.7**
MIL-seg	val	20.7
WyPR	val	29.6
WyPR+prior	val	**31.1**
WyPR	test	24.0
Supervised methods		
VoteNet [29]	test	55.7
SparseConvNet [9]	test	73.6

Table 2: 3D semantic segmentation on ScanNet. WyPR outperforms standard baselines and existing state-of-the-art [53] by a margin. We also report fully supervised methods for reference.

Network architecture. *(1) Backbone.* We use PointNet++ [32] as the backbone model to compute the point cloud features. The model has 4 set abstraction (SA) layers and 2 feature propagation (FP) layers. The four SA layers sub-sample the point cloud to 2048, 1024, 512 and 256 points using a receptive radius of 0.2, 0.4, 0.8 and 1.2 meters respectively. The two FP layers up-sample the last SA layer's output back from 256 to 1024 points. The final output has (256+3) dimensions (feature + 3D coordinates). *(2) Segmentation module.* This module is implemented as two FP layers which upsample the backbone features (1024 points) to the input size (40K points), and a two layer MLP (implemented as two 1×1 convolutional layers) which convert the features into per-point classification logits. *(3) Detection module.* This module has 3 fully-connected layers, computing the classification $\mathbf{S}_{\text{cls}}$, objectness $\mathbf{S}_{\text{obj}}$, and final classification logits $\mathbf{S}_{\text{det}}$ respectively, as described in § 3.2.

Training. We train the entire network end-to-end from scratch with an Adam optimizer for 200 epochs. We use 8 GPUs with a batch size of 32. The initial leaning rate is 0.003 and is decayed by $10\times$ at epoch $\{120, 160, 180\}$.

Inference. *(1) Segmentation.* We generate the segmentation mask from the predicted logits ($\mathbf{S}_{\text{seg}}$) by taking the class with highest score for each point. We then post-process the output for smoothness by using the detected planes (as in Eq. (7)), and assign each point in the plane to the most frequently occurring class. *(2) Detection.* Following [29], we post-process the final output probability, softmax($\mathbf{S}_{\text{det}}$), by thresholding to drop predictions with score < 0.01, and class-wise non-maximum suppression (NMS) with IoU threshold 0.25.

Dataset. We use the ScanNet [10] and S3DIS [2] datasets

Methods	Proposal #boxes	MABO	AR	Detection mAP
Unsupervised methods				
Qin *et al*. [33]	1k	0.092	23.6	-
GSS	≤256	0.321	73.4	-
GSS	≤1k	**0.378**	**86.2**	-
Weakly-supervised methods				
MIL-det (unsup. GSS)	≤1k	0.378	86.2	9.6
WyPR	≤1k	0.409	89.3	18.3
WyPR+prior	≤1k	**0.427**	**90.5**	**19.7**
Supervised methods				
F-PointNet [30]	-	-	-	10.8
GSPN [60]	-	-	-	17.7
3DSIS [18]	-	-	-	40.2
VoteNet [29]	256	0.436	84.7	58.6
VoteNet [29]	1k	0.450	88.1	55.3

Table 3: 3D object detection on ScanNet. Unsupervised GSS outperforms concurrent work [33] by a large margin. In the weakly-supervised setting, WyPR outperforms standard baselines and even some fully supervised approaches [30, 60].

to evaluate our method. ScanNet contains 1.2K training and 300 validation examples of hundreds of different rooms, annotated with 20 semantic categories. We extract ground truth bounding boxes from instance segmentation masks following [29]. To demonstrate the generalizability of our method, we futher evaluate on S3DIS, which contains 6 floors of 3 different buildings and 13 objects classes. We use the fold #1 split following prior work [2, 9], where area 5 is used for testing and the rest for training.

Evaluation. We report mean intersection over union (mIoU) across all classes for semantic segmentation, mean average precision (mAP) across all classes at IoU 0.25 for object detection, and average recall (AR) and mean average best overlap (MABO) across all classes for proposal generation. Please see [10, 29, 48] for more on these metrics.

4.1. Baselines

Besides comparing to the few existing 3D weakly-supervised learning methods, we build the following baselines, using standard weakly-supervised learning techniques:

MIL-seg: Single task segmentation trained with Eq. (3).

MIL-det: Single task object detection, which uses the unsupervised GSS proposals and is trained with Eq. (9).

WyPR: Our full model trained with Eq. (1) and Eq. (8).

WyPR+prior: We compute per-class mean shapes using external synthetic datasets [6, 57], and use those to reject proposals and pseudo labels in the WyPR detection module that do not satisfy the prior. We also use a floor height prior for segmentation. Please see Appendix D for details.

Methods	Segmentation mIoU	Proposal MABO	AR	Detection mAP
Weakly-supervised methods				
MIL-seg	17.6	-	-	-
MIL-det (unsup. GSS)	-	0.412	84.9	15.1
WyPR	**22.3**	**0.441**	**88.3**	**19.3**
Supervised methods				
PointNet++ [31]	41.1	-	-	-
SparseConvNet [9]	62.4	-	-	-
Armeni *et al*. [2]	-	-	-	49.9

Table 4: Generalizing to S3DIS. WyPR seamlessly generalizes to S3DIS, and outperforms standard baselines for both weakly-supervised segmentation and detection.

Removed	Seg. losses $\mathcal{L}_{\text{seg}}^{\text{SELF}}$	$\mathcal{L}_{\text{seg}}^{\text{CST}}$	$\mathcal{L}_{\text{d}\to\text{s}}$	$\mathcal{L}_{\text{smooth}}$	Det. losses $\mathcal{L}_{\text{det}}^{\text{SELF}}$	$\mathcal{L}_{\text{det}}^{\text{CST}}$	Seg. mIoU	Det. mAP
Self-training		✓	✓	✓		✓	22.1	13.2
Cross-transformation cst.	✓		✓	✓	✓		28.2	16.9
Cross-task consistency	✓	✓		✓	✓	✓	26.7	17.4
Local smoothness	✓	✓	✓		✓	✓	27.3	17.8
WyPR	✓	✓	✓	✓	✓	✓	29.6	18.3

Table 5: Ablation study of losses. We remove one set of losses at a time. All models are trained with $\mathcal{L}_{\text{seg}}^{\text{MIL}}$ and $\mathcal{L}_{\text{det}}^{\text{MIL}}$.

4.2. Quantitative results on ScanNet

Semantic Segmentation. Apart from the above baselines we compare WyPR to recent approaches, PCAM [53] and MPRM [53]. PCAM can be interpreted as **MIL-seg** with a KPConv [47] backbone, and MPRM adds multiple additional self-attention modules to PCAM. Since prior work reports results on the training set only, we compare against their results on the training set in Tab. 2 (top 3 rows). WyPR outperforms both methods (PCAM and MPRM) by a significant margin (+8.6% / +6.3%). Since the main difference between prior work and our method is our joint detection-segmentation framework, these results show the effectiveness of joint-training. When comparing against our baselines on the validation set (Tab. 2 middle) our joint model outperforms the single-task baseline (MIL-seg) by 8.9%. We observe a large performance gap when comparing against state-of-the-art fully supervised models (bottom two rows). One possible solution to minimize the gap is to utilize an external object prior (*e.g.*, shape) from readily-available synthetic data, which improves results by +1.5%.

Object Detection. To the best of our knowledge, no prior work has explored weakly-supervised 3D object detection using scene-level tags. We compare against our baseline methods in Tab. 3 (middle rows). Our model significantly outperforms the single-task baseline (MIL-det) by 8.7% mAP, and achieves competitive results compared to even some fully supervised methods (F-PointNet [30] and GSPN [60], numbers borrowed from [29]). However, the performance gap is large when compared to the state-of-

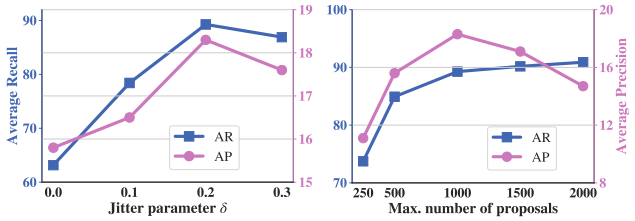

Figure 4: Effect of jittering and #proposals. Jittering the point cloud before proposal generation results in a $>2\%$ gain in AP. The performance varies gracefully with #proposals, and we find 1000 proposals to have the right balance for high precision and recall.

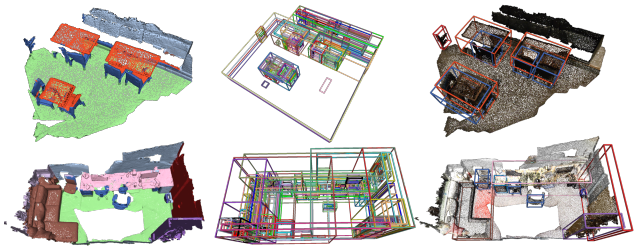

Figure 5: Qualitative results on ScanNet. WyPR+prior is able to segment, generate proposals and detect objects without having ever seen any spatial annotations.

the-art fully supervised methods. Similar to segmentation, the performance of our model can be further improved by incorporating an external object prior (+1.4%).

Proposal Generation. GSS can be made unsupervised by relying only on low-level shape and color cues, *i.e.*, removing s_{seg} from Eq. (13) (§ 3.3). We compare the unsupervised GSS to a concurrent unsupervised 3D proposal approach by Qin *et al*. [33]. We adapt their method, originally designed for outdoor environments, to indoor scenes by replacing their front-view projection to a Y-Z plane projection. For a fair comparison we use 1000 proposals and report results in Tab. 3 (top rows). Unsupervised GSS outperforms [33] by a large margin, and obtains recall values comparable to even supervised approaches. The complete GSS, including the weakly-supervised similarity s_{seg}, further improves over the unsupervised baseline (+3.1% AR/+0.031 MABO), and outperforms supervised methods on recall (+1.2%), indicating the importance of joint training.

4.3. Generalizing to S3DIS

We train WyPR on S3DIS following the settings of § 4.2. Since there is no prior weakly-supervised work on this dataset, we compare against our baselines from § 4.1. The results are summarized in Tab. 4, where WyPR outperforms both single-task baselines with gains of 4.7% mIoU for segmentation, 3.4% AR for proposal generation, and 4.2% mAP for detection. These results also demonstrate that our design choices are not specific to ScanNet and generalize to different 3D datasets.

4.4. Analysis

Which loss terms matter? In Tab. 5 we analyze the relative contribution of the loss terms in Eq. (1) and (8). We find self-training to be the most critical. Removing $\mathcal{L}_{\text{seg}}^{\text{SELF}}$ and $\mathcal{L}_{\text{det}}^{\text{SELF}}$ leads to a significant drop in both metrics: -7.5% mIoU and -5.1% mAP. This is consistent with observations in prior work on weak-supervision [34, 54]. Next, we find enforcing consistency between detection and segmentation tasks to add large gains, especially for segmentation: 2.9% mIoU. Enforcing consistency across transformations is particularly important for detection, leading to a 1.4% mAP gain. Finally, encouraging smoothness over primitive structures improves both metrics by 1.7% mIOU and 0.5% mAP.

Jittering for proposal generation. We observe that scanned point clouds are often imperfect, with large holes in objects due to occlusions, clutter or sensor artifacts. This makes it challenging for GSS to correctly group parts. To overcome this, we jitter the points in 3D space using a random multiplier within range $[1-\delta, 1+\delta]$ and decide the neighboring regions based on the jittered points. This simple technique counts spatially close but non-overlapping regions as neighbors, and greatly improves GSS results. We show the impact of δ in Fig. 4 (left).

Number of proposals. We randomly sample at most 250, 500, 1000, 1500, 2000 regions from the same set of computed proposals and report the recall and detection mAP in Fig. 4 (right). Using fewer proposals hurts both the recall and precision since the model misses many relevant objects. In contrast, a large number of proposals increases recall but hurts precision, presumably because too many proposals increase the false positive rate of the detection module. We find 1000 proposals to be a good balance between precision and recall, and use this number for all our experiments.

Qualitative results. Fig. 5 shows a few representative examples of our model's predictions on ScanNet. As can be seen, input point clouds are quite challenging, with large amounts of clutter and sensor imperfections. Nevertheless, our model is able to recognize objects such as chairs, tables, and sofa with good accuracy. Please see Appendix F for more results, analysis and failure modes.

5. Conclusion

We propose WyPR, a novel framework for joint 3D semantic segmentation and object detection, trained using only scene-level class tags as supervision. It leverages a novel unsupervised 3D proposal generation approach (GSS) along with natural constraints between the segmentation and detection tasks. Through extensive experimentation on standard datasets we show WyPR outperforms single task baselines and prior state-of-the-art methods on both tasks.

Acknowledgements. This work is supported in part by

NSF under Grant #1718221, 2008387 and MRI #1725729, NIFA award 2020-67021-32799.

References

[1] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 6

[2] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv:1702.01105*, 2017. 2, 6, 7

[3] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016. 2, 4, 5

[4] Rich Caruana. Multitask learning. *Machine Learning*, 1997. 3

[5] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. 2

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012*, 2015. 2, 7, 14

[7] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019. 2

[8] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016. 2

[9] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 1, 2, 6, 7

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1, 2, 6, 7

[11] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017. 3

[12] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 3

[13] Cat Franklin. Apple unveils new ipad pro with breakthrough lidar scanner and brings trackpad support to ipados. https://www.apple.com/, 2020. 1, 2

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013. 2

[15] Rohit Girdhar, David Ford Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 2

[16] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[17] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 2

[18] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *CVPR*, 2019. 2, 7

[19] Iasonas Kokkinos. Ubernet: Training a 'universal' convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017. 3

[20] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018. 2

[21] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*, 2018. 2

[22] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *IJCV*, 2012. 4, 5, 11, 13

[23] Xiaoyan Li, Meina Kan, Shiguang Shan, and Xilin Chen. Weakly supervised object detection with segmentation collaboration. In *CVPR*, 2019. 3

[24] Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Luc Van Gool, and Dengxin Dai. Weakly supervised 3d object detection from lidar point cloud. In *ECCV*, 2020. 3

[25] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016. 3

[26] Sven Oesau, Yannick Verdie, Clément Jamin, Pierre Alliez, Florent Lafarge, Simon Giraudot, Thien Hoang, and Dmitry Anisimov. Shape detection. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.1 edition, 2020. 5, 13

[27] Deepak Pathak, Philipp Krähenbühl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*, 2015. 2

[28] Charles R. Qi, Xinlei Chen, Or Litany, and Leonidas J. Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *CVPR*, 2020. 2, 6

[29] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 1, 2, 3, 6, 7, 13, 14

[30] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018. 2, 7

[31] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2, 3, 7

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1, 2, 3, 6

[33] Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3d object detection from point clouds. In *ACM MM*, 2020. 2, 3, 6, 7, 8

[34] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *CVPR*, 2018. 3, 8

[35] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Yong Jae Lee, Alexander G. Schwing, and Jan Kautz. Instance-aware, context-focused, and memory-efficient weakly supervised object detection. In *CVPR*, 2020. 2, 4

[36] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Alexander G. Schwing, and Jan Kautz. UFO^2: A unified framework towards omni-supervised object detection. In *ECCV*, 2020. 1, 3

[37] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 2

[38] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, 2007. 5, 13

[39] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 2

[40] Krishna Kumar Singh, Fanyi Xiao, and Yong Jae Lee. Track and transfer: Watching videos to simulate strong human supervision for weakly-supervised object detection. In *CVPR*, 2016. 2

[41] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 2

[42] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *CVPR*, 2016. 2

[43] Scott Stein. Lidar on the iphone 12 pro. https://www.cnet.com/, 2020. 1, 2

[44] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2

[45] Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance classifier refinement. In *CVPR*, 2017. 2, 4

[46] Yew Siang Tang and Gim Hee Lee. Transferable semi-supervised 3d object detection from rgb-d data. In *CVPR*, 2019. 2, 3

[47] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 2, 7

[48] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M.Smeulders. Selective search for object recognition. *IJCV*, 2013. 2, 6, 7, 11

[49] Ozan Unal, Luc Van Gool, and Dengxin Dai. Improving point cloud semantic segmentation by learning 3d object proposal generation. *arXiv:2009.10569*, 2020. 3

[50] Fang Wan, Chang Liu, Wei Ke, Xiangyang Ji, Jianbin Jiao, and Qixiang Ye. C-mil: Continuation multiple instance learning for weakly supervised object detection. In *CVPR*, 2019. 4

[51] Haiyan Wang, Xuejian Rong, Liang Yang, Shuihua Wang, and Yingli Tian. Towards weakly supervised semantic segmentation in 3d graph-structured point clouds of wild scenes. In *BMVC*, 2019. 2

[52] 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*, 2019. 2

[53] Jiacheng Wei, Guosheng Lin, Kim-Hui Yap, Tzu-Yi Hung, and Lihua Xie. Multi-path region mining for weakly supervised 3d semantic segmentation on point clouds. In *CVPR*, 2020. 2, 6, 7, 14, 15

[54] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *CVPR*, 2017. 2, 3, 8

[55] Yunchao Wei, Huaxin Xiao, Honghui Shi, Zequn Jie, Jiashi Feng, and Thomas S Huang. Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation. In *CVPR*, 2018. 2, 3

[56] Benjamin Wilson, Zsolt Kira, and James Hays. 3d for free: Crossmodal transfer learning using hd maps. *arXiv:2008.10592*, 2020. 2, 3

[57] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2, 7, 14

[58] Jia Xu, Alexander Schwing, and Raquel Urtasun. Tell Me What You See and I will Show You Where It Is. In *CVPR*, 2014. 2

[59] Xun Xu and Gim Hee Lee. Weakly supervised semantic point cloud segmentation: Towards 10x fewer labels. In *CVPR*, 2020. 2

[60] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *CVPR*, 2019. 7

[61] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *ECCV*, 2020. 2

[62] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. Sess: Self-ensembling semi-supervised 3d object detection. In *CVPR*, 2020. 3

[63] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 2

[64] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 2, 6

Appendix

In this section, we provide: (1) algorithm details and ablation studies of Geometric Selective Search (Appendix A); (2) introduction of the shape detection algorithm (Appendix B); (3) additional implementation details (Appendix C); (4) details of the integration of an external object prior (Appendix D); and (5) per-class segmentation results; (6) additional qualitative results (Appendix F).

A. Geometric Selective Search (GSS)

As introduced in the main paper § 3.3, the goal of GSS is to capture all possible object locations in 3D space. We formulate a bottom-up algorithm where the key idea is to utilize the geometric and semantic cues for guiding 3D proposal generation.

A.1. Approach

Given an input point cloud with unoriented normals, we first detect primitive shapes using a region growing based method [22] as detailed in Appendix B. It outputs a set of detected planes with assigned points, *i.e.*, each point is assigned to at most one plane or none.

We then apply hierarchical agglomerative clustering (HAC) to generate the candidate bounding boxes from the detected planes. We first initialize a region set with the detected planes, and then compute the similarity score s between all neighboring regions in the set. Two regions are neighboring if the corresponding convex hull of them overlap. To overcome the artifacts of the point cloud, we randomly jitter the points of each region before computing their convex hull. This technique greatly improves the results in practice as verified in Fig. 4. Once the neighboring relationships and similarity scores are computed, the two most similar regions are grouped into a new region. We then generate an axis-aligned 3D box for the new region as a proposal. New similarity scores are calculated between the resulting region and its neighbors. HAC is repeated until no neighbors can be found or only a single region remains. We provide the detailed pseudo-code in Alg. 3.

In order to pick which two regions $\mathbf{n}_i, \mathbf{n}_j$ to group, we use the similarity score $s(\mathbf{n}_i, \mathbf{n}_j) =$

$$\begin{aligned} &w_1 s_{\text{color}}(\mathbf{n}_i, \mathbf{n}_j) + w_2 s_{\text{size}}(\mathbf{n}_i, \mathbf{n}_j) + \\ &w_3 s_{\text{volume}}(\mathbf{n}_i, \mathbf{n}_j) + w_4 s_{\text{fill}}(\mathbf{n}_i, \mathbf{n}_j) + w_5 s_{\text{seg}}(\mathbf{n}_i, \mathbf{n}_j), \end{aligned} \tag{13}$$

where $w_i \in \{0, 1\} \ \ \forall i \in \{1, \cdots, 5\}$ are binary indicators. Binary weights are used over continuous values to encourage more diverse outputs following [48]. $s_{\text{color}} \in [0, 1]$ measures the color similarity; s_{size} and $s_{\text{volume}} \in [0, 1]$ measure size and volume compatibility and encourage small regions to merge early; $s_{\text{fill}} \in [0, 1]$ measures how well two regions are aligned; and $s_{\text{seg}}(\mathbf{n}_i, \mathbf{n}_j) \in [0, 1]$ measures high-level semantic similarities. We detail each metric next.

Algorithm 3 Geometric Selective Search (GSS)

Input: point cloud $\mathcal{P}$
Output: 3D proposal set $\mathcal{R}$
Detect shapes from $\mathcal{P} \rightarrow$ initial regions $\mathcal{N} = \{\mathbf{n}_1, \mathbf{n}_2, \cdots\}$
Initialize similarity set $\mathcal{S} = \emptyset$, proposal set $\mathcal{R} = \emptyset$
for each neighboring region pair $(\mathbf{n}_i, \mathbf{n}_j)$ **do**
  $\mathcal{S} = \mathcal{S} \cup s(\mathbf{n}_i, \mathbf{n}_j)$ ▷ compute and store similarities
while $\mathcal{S} \neq \emptyset$ **do** ▷ HAC
  Get the most similar pair $s(\mathbf{n}_i, \mathbf{n}_j) = \max(\mathcal{S})$
  Remove similarities regarding $\mathbf{n}_i$: $\mathcal{S} = \mathcal{S} \setminus s(\mathbf{n}_i, *)$
  Remove similarities regarding $\mathbf{n}_j$: $\mathcal{S} = \mathcal{S} \setminus s(\mathbf{n}_j, *)$
  Update region set $\mathcal{N} = \mathcal{N} \setminus \mathbf{n}_i, \mathcal{N} = \mathcal{N} \setminus \mathbf{n}_j$
  Merge and generate new region $\mathbf{n}_k = \mathbf{n}_i \cup \mathbf{n}_j$
  Compute similarity of $\mathbf{n}_k$ and its neighbors in $\mathcal{N}$: $\mathcal{S} = \mathcal{S} \cup \{s(\mathbf{n}_k, \mathbf{n}') : \text{neighbor}(\mathbf{n}_k, \mathbf{n}') = \text{True} \ \forall \mathbf{n}' \in \mathcal{N}\}$
  Add new region to $\mathcal{N} = \mathcal{N} \cup \mathbf{n}_k$
  Generate 3D proposal $\mathcal{R} = \mathcal{R} \cup \text{AxisAlignedBox}(\mathbf{n}_k)$

Color similarity s_{color}. Color is an informative low-level cue to guide the plane grouping process. For each region, we first compute the L1-normalized color histogram following [48]. The similarity score is computed as the histogram intersection:

$$s_{\text{color}}(\mathbf{n}_i, \mathbf{n}_j) = \sum_k \min(b_i^k, b_j^k), \tag{14}$$

where b_i^k, b_j^k are the k-th bin in the color histograms of $\mathbf{n}_i$ and $\mathbf{n}_j$ respectively. Following [48], we use 25 bins for each HSV color channel and 75 in total for one histogram.

Size similarity s_{size} and volume similarity s_{volume}. These two metrics encourage small regions to merge early. This strategy is desirable as it guarantees a bottom-up grouping of parts of different objects at multiple locations in 3D space. It encourages diverse 3D proposals and prevents a single region from absorbing all other regions gradually. We compute size similarity

$$s_{\text{size}}(\mathbf{n}_i, \mathbf{n}_j) = 1 - \frac{\text{size}(\mathbf{n}_i) + \text{size}(\mathbf{n}_i)}{\text{size}(\mathcal{P})}, \tag{15}$$

where $\text{size}(\mathbf{n}_i), \text{size}(\mathbf{n}_j), \text{size}(\mathcal{P})$ are the size of the axis-aligned bounding boxes of region $\mathbf{n}_i, \mathbf{n}_j$ and the whole point cloud. Similarly, volume similarity is defined as:

$$s_{\text{volume}}(\mathbf{n}_i, \mathbf{n}_j) = 1 - \frac{\text{volume}(\mathbf{n}_i) + \text{volume}(\mathbf{n}_i)}{\text{volume}(\mathcal{P})}, \tag{16}$$

where $\text{volume}(\mathbf{n}_i), \text{volume}(\mathbf{n}_j), \text{volume}(\mathcal{P})$ are the volume of the water-tight convex hull of region $\mathbf{n}_i, \mathbf{n}_j$ and the whole point cloud.

Alignment score s_{fill}. This score measures how well two regions fit into each other and encourage merged regions to be cohesive. Essentially, if one region is contained in the other one, they should be merged first to avoid any holes. Meanwhile, a low score means the two regions don't fit very

Class	cabinet	bed	chair	sofa	table	door	window	shelf	picture	counter	desk	curtain	fridge	sc*	toilet	sink	bathtub	other	mean
Unsupervised GSS																			
ABO	0.402	0.414	0.419	0.462	0.432	0.327	0.349	0.469	0.121	0.286	0.365	0.342	0.469	0.421	0.415	0.355	0.325	0.432	0.378
Recall	86.0	97.5	90.4	99.0	91.1	67.0	86.9	100.0	26.1	75.0	92.1	91.0	98.2	96.4	94.8	91.8	77.4	90.9	86.2
GSS																			
ABO	0.449	0.471	0.441	0.437	0.464	0.379	0.388	0.446	0.136	0.366	0.381	0.399	0.501	0.478	0.409	0.365	0.400	0.453	0.409
Recall	90.6	98.8	91.7	98.9	93.7	75.2	89.7	100.0	27.9	88.5	94.5	97.0	96.5	100.0	94.8	92.9	83.8	92.3	89.3

Table 6: Per-class results of GSS proposals. GSS achieves more than 80% recall rate for all classes except picture (27.9%) and door (75.2%), where the plan detection algorithm often fails to differentiate these two objects from the surrounding wall. Here sc* refers to the 'shower curtain' class.

well, and they may form a strange region. **AS:** what's a 'strange region'? We compute the alignment score:

$$s_{\text{fill}}(\mathbf{n}_i, \mathbf{n}_j) = 1 - \frac{\text{size}(\mathbf{n}_i \cup \mathbf{n}_j) - \text{size}(\mathbf{n}_i) - \text{size}(\mathbf{n}_i)}{\text{size}(\mathcal{P})}, \tag{17}$$

where $\mathbf{n}_i \cup \mathbf{n}_j$ means the union of two regions, and the other numbers are identical to the ones used for the computation of s_{color}.

Semantic similarity s_{seg}. The above four metrics are mainly low-level geometric cues. GSS can also utilize high-level semantic information, *i.e.*, weakly-supervised segmentation prediction. For each region, we first infer the segmentation mask from $\mathbf{S}_{\text{seg}}$ using the inference procedure described in § 4. We then take the most likely class assignment for each point in the region and compute an L1-normalized histogram over classes for that region. The similarity score is computed as the histogram intersection:

$$s_{\text{seg}}(\mathbf{n}_i, \mathbf{n}_j) = \sum_{c=1}^{C} \min(b_i^c, b_j^c), \tag{18}$$

where b_i^c, b_j^c are the bin of class c in the class histograms.

Post-processing. To remove the redundant proposals, we use several post-processing steps: (1) the proposals are first filtered by a 3D NMS module with an IoU threshold of 0.75; (2) we then remove the largest bounding boxes after NMS as it covers the whole scene rather than certain objects due to the bottom-up nature of HAC; (3) we keep at most 1000 proposals through random sampling.

Diversification strategies. Since a single strategy usually overfits, we adopt multiple strategies to encourage a diverse set of proposals, which will eventually lead to a better coverage of all objects in 3D space. Specifically, we first create a set of complementary strategies, and ensemble their results afterwards. Highly-overlapping redundant proposals are removed though an NMS with IoU threshold of 0.75 and we still keep at most 1000 proposals through random sampling after ensembeling.

Metric	Avg. # boxes	MABO	AR
Single run			
SZ	382.9	0.351	84.1
C	252.0	0.316	70.7
V	366.8	0.367	84.4
F	330.2	0.398	81.8
SG	350.7	0.362	83.9
SZ+C	295.0	0.361	79.0
SZ+V	373.4	0.366	84.5
SZ+SG	369.2	0.353	85.1
V+F	373.3	0.384	85.7
V+SG	385.5	0.362	83.8
SZ+C+F	297.6	0.361	78.6
SZ+V+SG	377.5	0.391	**86.4**
V+F+SG	381.6	0.380	84.9
SZ+C+V+F	320.4	0.379	81.9
SZ+V+F+SG	369.1	0.387	86.1
Ensembeling			
C, V+F, SZ+V	712.0	0.378	86.2
C, V+F, SZ+V+SG	742.9	0.409	89.3

Table 7: GSS results using various similarity metrics. SZ, C, V, F, and SG represent s_{size}, s_{color}, s_{volume}, s_{fill}, and s_{seg} respectively.

A.2. Experiments

In this sub-section we evaluate the proposal quality of GSS and validate the corresponding design choices. We evaluate on the ScanNet validation set and report the two popular metrics: average recall (AR) and mean average best overlap (MABO) across all classes. In addition, we also report the average number of boxes of each scene.

We first examine each similarity metric and their combinations in Tab. 7. We first evaluate each single similarity and report their results in the top 5 rows, where we find size, volume, and segmentation metric to work much better than color and fill similarity. Tab. 7 also reports the results of different combined metrics. Combining multiple similarity metrics often yields better results than using each single similarity. The best result is achieved using the combination

of size, volume, and segmentation similarities.

In practice, we find that ensembling the results of multiple runs using different similarity metrics further improves the results as shown in Tab. 7 bottom. We provide the results of an unsupervised version (C, V+F, SZ+V) and the complete version (C, V+F, SZ+V+SG). Comparing these two methods, we find that introducing segmentation similarity is beneficial.

Lastly, we show per-class average best overlap (ABO) and recall rate in Tab. 6. We find that GSS achieves high recall rate ($> 80\%$) for all classes except picture (27.9%) and door (75.2%). This is likely due to the fact that these two objects are often embedded in the wall and hard to differentiate.

A.3. Qualitative results

Fig. 3 illustrates several representative examples of the generated proposals on ScanNet. From left to right, we show the input point cloud, the detected shapes, GSS computed proposals, and the ground-truth boxes. We show all the GSS computed proposals in the top 3 rows where we observe that the computed proposals are mainly around each object in the scene. In the bottom four rows, we show the best overlapping proposals with ground-truth bounding boxes. GSS generates proposals with great recall, and generalizes well to various object classes and complex scenes.

B. Shape detection

In this paper, we detect geometric shapes for two reasons: to be used in the local smoothness loss for segmentation (Eq. (7)), and as input to the GSS algorithm (Appendix A). As introduced in the main paper § 3.3, we adopt a region-growing algorithm [22, 26] for detecting primitive shapes (*e.g.*, planes). The basic idea is to iteratively detect shapes by growing regions from seed points. Specifically, we first choose a seed point and find its neighbors in the point cloud. These neighbors are added to the region if they satisfy the region requirements (*e.g.*, on the same plane), and hence the region grows. We then repeat the procedure for all the points in the region until no neighbor points meet the requirements. In the latter case we start a new region. Region-growing out-performs the popular RANSAC-based methods [38] because 1) it is deterministic; 2) it performs better in the presence of large scenes with fine-grained details; 3) it has higher shape detection recall. Even though it runs slower, we use it as a pre-processing step which won't influence the training speed.

In practice, we use the efficient implementation of The Computational Geometry Algorithms Library (CGAL) [26]. We set the search space to be the 12 nearest neighbors, the maximum distance from the furthest point to a plane to be 12, the maximum accepted angle between a point's normal and the normal of a plane to be 20 degree, and the minimum region size to be 50 points. We refer the reader to CGAL documents [26] for more details.

Representative visualization of the detected planes are provided in Fig. 6 second column from left. The algorithm detects big planes (*e.g.*, floor, table top, wall) with great accuracy and doesn't over segment these regions into small pieces. This is particularly useful for WyPR as the local smoothness loss will enforce the segmentation module to predict consistently within these shapes. For complex objects (*e.g.*, curtain, chair, and bookshelf), this algorithm segments the object regions into small shapes. Such primitive shapes will be used during the proposal generation algorithm GSS to infer the 3D bounding boxes of all objects in the scene.

C. Additional implementation details

In this section, we provide additional implementation details.

C.1. Geometric transformations

We apply geometric transformations in two places: 1) as data-augmentation; 2) for computing cross-transformation consistency losses (Eq. (5) and Eq. (11)) for both tasks.

To augment the input, we first randomly sub-sample 40,000 points as input in each training iteration. We then randomly flip the points in both horizontal and vertical directions with probability 0.5, and randomly rotate them around the upright-axis with $[-5, 5]$ degree. Note that after data augmentation, we only get one point cloud $\mathcal{P}$ as input.

To compute the consistency losses, we further transform the input point cloud using random flipping of both horizontal and vertical directions with probability 0.5, larger random rotation of $[0, 30]$ degrees around the upright-axis, random scaling by a factor within $[0.8, 1.2]$, and point dropout ($p = 0.1$). We denote the resulting point cloud as $\widetilde{\mathcal{P}}$, which will be used when computing $\mathcal{L}_{\text{seg}}^{\text{CST}}$ and $\mathcal{L}_{\text{det}}^{\text{CST}}$.

C.2. Backbone

We adopt a PointNet++ network as backbone, which has four set abstraction (SA) layers and two feature propagation (FP) layers. For a fair comparison we use the same backbone network as Qi *et al*. [29]. The input to the backbone is a fix-sized point cloud where we randomly sample 40,000 points from the original scans. The outputs of the backbone network are geometric representations of 1024 points with dimension 3+256 (XYZ+feature dimension).

C.3. Segmentation module

The segmentation module contains two feature propagation (FP) layers which upsample the geometric representations of 1024 points to 2048 and then 40,000 points with

metric	cabinet	bed	chair	sofa	table	door	shelf	desk	curtain	fridge	toilet	sink	bathtub
$\mu_{l:w}$	4.64	1.58	1.29	1.94	1.65	5.74	3.17	1.92	5.78	1.68	1.55	1.29	1.93
$\sigma_{l:w}$	5.81	0.45	0.53	0.54	1.02	3.78	2.07	0.91	3.58	1.16	0.39	0.26	0.42
$\mu_{l:h}$	1.49	2.12	1.16	2.36	3.04	0.61	1.22	2.28	1.40	0.65	1.08	2.14	3.18
$\sigma_{l:h}$	1.01	0.95	0.98	0.57	3.72	0.69	1.11	1.65	1.34	0.19	0.56	0.89	1.67

Table 8: Prior statistics of each class.

the same dimension (3+256) as before. We then use a two-layer MLP with dimension $[256, C]$ as the classifier where C represents the number of classes. The segmentation module outputs a dense semantic prediction for each point in the point cloud.

C.4. Detection module

The detection module first applies a RoI pooling by average-pooling the features of all points within each RoI. The computed RoI features are then fed into three fully-connected layers to get the classification $\mathbf{S}_{\text{cls}}$, objectness $\mathbf{S}_{\text{obj}}$, and final classification logits $\mathbf{S}_{\text{det}}$ respectively.

C.5. Losses

For computing the smoothness regularization $\mathcal{L}_{\text{smooth}}$ in Eq. (7), enumerating all the detected planes in each training iteration is time-consuming and not necessary. We thus randomly sample 10 planes in each iteration, as we find 10 to be the sweet spot balancing training speed and performance. For computing the self-training losses $\mathcal{L}_{\text{seg}}^{\text{SELF}}$ and $\mathcal{L}_{\text{det}}^{\text{SELF}}$, we set the threshold p_1 in Alg. 1 to be 0.1, and p_2 in Alg. 2 to be 0.15. The threshold τ in Alg. 2 is set to 0.25.

D. External prior

WyPR can be further improved by integrating external object priors as shown in Tab. 2 and Tab. 3. We mainly introduce two types of priors as they can be easily computed from external synthetic datasets [6, 57]: the shape prior and the location prior.

For the shape prior, we compute the mean aspect ratio between an object's 3D bounding box length to height ($\mu_{l:h}^c$), and length to width ($\mu_{l:w}^c$) for class $c \in \{1, \cdots, C\}$. Since objects can be of arbitrary pose in 3D space, we set length and width to measure the longer and shorter edge in the XY plane. We also compute the corresponding standard deviations $\sigma_{l:h}^c$ and $\sigma_{l:d}^c$. To use it, we reject proposals whose aspect ratios don't fall within the range $[\mu_{l:h}^c - 2\sigma_{l:h}^c, \mu_{l:h}^c + 2\sigma_{l:h}^c]$ and $[\mu_{l:w}^c - 2\sigma_{l:w}^c, \mu_{l:w}^c + 2\sigma_{l:w}^c]$ for any class $c \in \{1, \cdots, C\}$. We also reject pseudo bounding boxes of ground-truth class c ($R^*[c]$ in Alg. 2) whose aspect ratios don't fall in $[\mu_{l:h}^c - 2\sigma_{l:h}^c, \mu_{l:h}^c + 2\sigma_{l:h}^c]$ and $[\mu_{l:w}^c - 2\sigma_{l:w}^c, \mu_{l:w}^c + 2\sigma_{l:w}^c]$. The computed statistics of each class are shown in Tab. 8. There are certain classes that are missing from the external synthetic datasets [6, 57] such as shower curtain, window, counter, and picture. For these classes, we use the prior of other objects with similar shapes as a replacement. For example, we use the prior of curtain for shower curtain, table for counter, door for window and picture.

The location prior is only applied to the floor class. This prior is of vital importance as floor appears in almost every scene. It becomes a hard class for semantic segmentation as the MIL loss rarely sees any negative examples. Besides, a great portion of points in each scene belongs to the floor. We estimate the floor height as the 1% percentile of all points' heights following Qi *et al*. [29]. We force all the points below floor height to be floor. All the points above this height cannot be floor.

E. Per-class segmentation results

In Tab. 9, we report the per-class IoU on ScanNet. These results are consistent with Tab. 2 in main paper. Compared to prior methods PCAM [53] and MPRM [53], WyPR significantly out-performs them, and greatly improves the performance of some hard classes such as door, counter, and fridge.

F. Additional qualitative results

In Fig. 7, we show the qualitative comparison between ground-truth labels and our (WyPR+prior) prediction. In each row we show the results of both tasks for one scene. We find that WyPR segments and detects certain classes (table in row (a, f), chair in rows (a, b, f), sofa in row (b), bookshelf in row (c, f)) with great accuracy. WyPR also learns to recognize some uncommon objects of the dataset such as toilet and sink in row (d). Moreover, we observe that predicted segmentation mask and bounding boxes are highly consistent, which reflects the effectiveness of the joint-training framework.

Common failure cases for WyPR are partially observed objects (row (b): the window on the left side), ambiguous objects (row (a): picture and wall; row (b, f): sofa and left-most chair). When multiple objects of the same classes are spatially close, WyPR often cannot differentiate them and only predicts one big boxes covering everything (row (a): tow chair on the left side).

Methods	eval.	wall	floor	cabinet	bed	chair	sofa	table	door	window	shelf	picture	counter	desk	curtain	fridge	sc*	toilet	sink	bathtub	other	mIoU
PCAM [53]	train	54.9	48.3	14.1	34.7	32.9	45.3	26.1	0.6	3.3	46.5	0.6	6.0	7.4	26.9	0.0	6.1	22.3	8.2	52.0	6.1	22.1
MPRM [53]	train	47.3	41.1	10.4	43.2	25.2	43.1	21.5	9.8	12.3	45.0	9.0	13.9	21.1	40.9	1.8	29.4	14.3	9.2	39.9	10.0	24.4
WyPR	train	**59.3**	31.5	6.4	**58.3**	31.6	**47.5**	18.3	**17.9**	**36.7**	34.1	6.2	**36.1**	**24.3**	**67.2**	**8.7**	**38.0**	17.9	**28.9**	35.9	8.2	**30.7**
MIL-seg	val	36.4	36.1	**13.5**	37.9	25.1	31.4	9.6	18.3	19.8	33.1	**7.9**	20.3	21.7	32.5	6.4	14.0	7.9	14.7	19.4	8.5	20.7
WyPR	val	**58.1**	33.9	5.6	**56.6**	29.1	**45.5**	19.3	15.2	**34.2**	**33.7**	6.8	**33.3**	22.1	65.6	6.6	**36.3**	18.6	24.5	**39.8**	6.6	29.6
WyPR+prior	val	52.0	**77.1**	6.6	54.3	**35.2**	40.9	**29.6**	9.3	28.7	33.3	4.8	26.6	**27.9**	**69.4**	**8.1**	27.9	**24.1**	**25.4**	32.3	**8.7**	**31.1**

Table 9: 3D semantic segmentation on ScanNet. WyPR outperforms standard baselines and existing state-of-the-art [53] by a margin. Here sc* refers to the 'shower curtain' class.

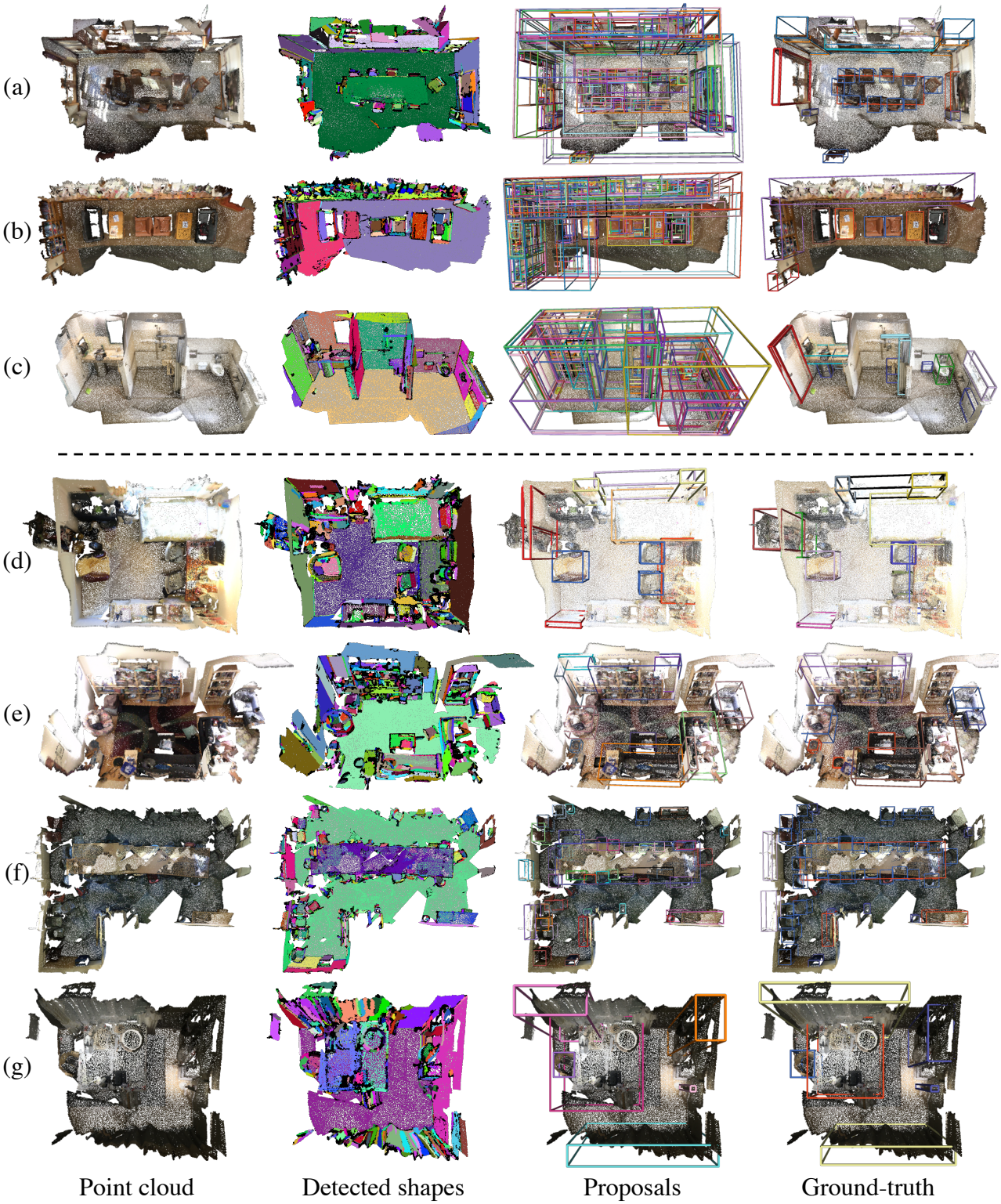

Figure 6: Visualization of the computed proposals. Top three rows show all the computed 3D proposals, from which we observe that the proposals are mainly around object areas. The bottom four rows show the proposals which best overlap with ground-truth boxes. GSS generates 3D proposals with great recall for various objects in complex scenes.

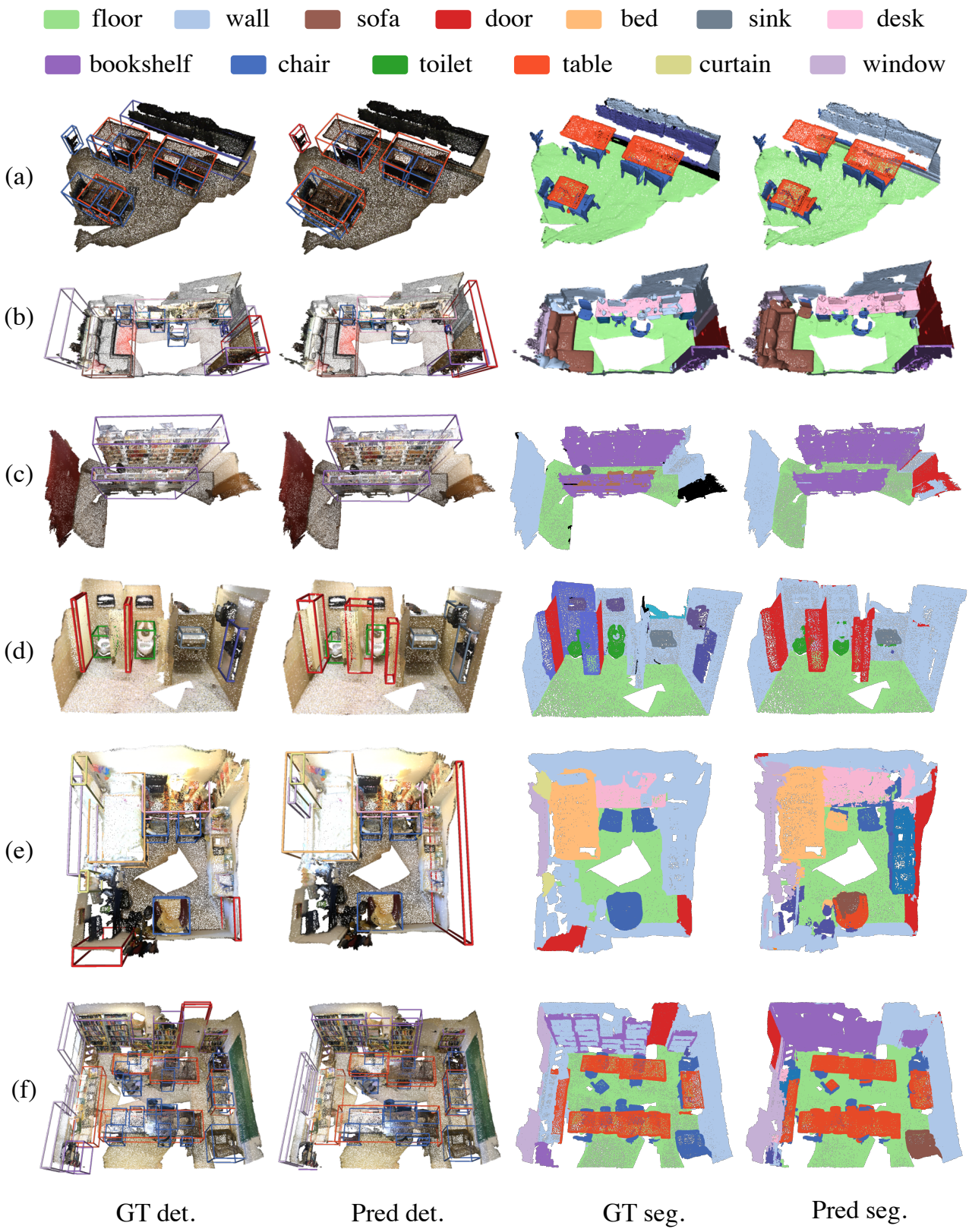

Figure 7: Additional qualitative results. We show the qualitative comparison between ground-truth labels and our (WyPR+prior) predictions. We show both detection and segmentation results for the same scene.