# Back to Reality: Weakly-supervised 3D Object Detection with Shape-guided Label Enhancement

Xiuwei Xu[1,2], Yifan Wang[1], Yu Zheng[1,2], Yongming Rao[1,2], Jie Zhou[1,2], Jiwen Lu[1,2]*
[1]Department of Automation, Tsinghua University, China
[2]Beijing National Research Center for Information Science and Technology, China
{xxw21, yifan-wa21, zhengyu19}@mails.tsinghua.edu.cn; raoyongming95@gmail.com;
{jzhou, lujiwen}@tsinghua.edu.cn

## Abstract

*In this paper, we propose a weakly-supervised approach for 3D object detection, which makes it possible to train strong 3D detector with position-level annotations (i.e. annotations of object centers). In order to remedy the information loss from box annotations to centers, our method, namely Back to Reality (BR), makes use of synthetic 3D shapes to convert the weak labels into fully-annotated virtual scenes as stronger supervision, and in turn utilizes the perfect virtual labels to complement and refine the real labels. Specifically, we first assemble 3D shapes into physically reasonable virtual scenes according to the coarse scene layout extracted from position-level annotations. Then we go back to reality by applying a virtual-to-real domain adaptation method, which refine the weak labels and additionally supervise the training of detector with the virtual scenes. Furthermore, we propose a more challenging benckmark for indoor 3D object detection with more diversity in object sizes for better evaluation. With less than 5% of the labeling labor, we achieve comparable detection performance with some popular fully-supervised approaches on the widely used ScanNet dataset. Code is available at:* https://github.com/wyf-ACCEPT/BackToReality.

## 1. Introduction

3D object detection is a fundamental scene understanding problem, which aims to detect the 3D bounding boxes and semantic labels from a point cloud of 3D scene. Due to the irregular form of point clouds and complex contexts in 3D scenes, most existing 2D methods [33, 34, 52] cannot be directly applied to 3D object detection. Fortunately, with the development of deep learning techniques on point cloud understanding [29, 30], recent works [13, 22, 27, 37, 53] have proposed deep neural networks to directly detect objects from point clouds and achieved favorable performance.

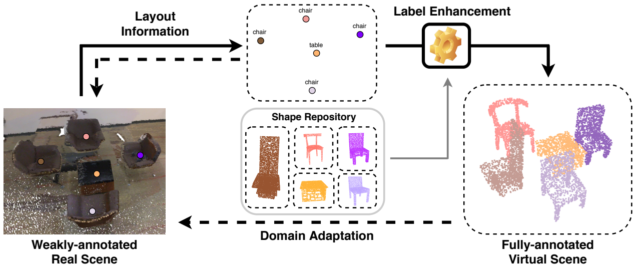


Figure 1. Demonstration of *BR*. We regard position-level annotations as the coarse layout of the scenes, which is utilized to generate virtual scenes from a 3D shape repository. Physical constraints are applied on the virtual scenes to remedy the information loss from box annotations to centers. Then a virtual-to-real domain adaptation method is presented to additionally supervise the real-scene 3D object detection with the virtual scenes. Dashed arrows indicate supervision for training.

Despite the success in deep learning methods of object detection on point cloud, massive amounts of labeled bounding boxes are required for training the detector. This issue significantly limits the applications of these methods, as labeling a precise 3D box takes more than 100s even by experienced annotator [38]. Therefore, 3D object detection methods using cheap labels are desirable for practical applications. Motivated by that, increasing attention has been paid to weakly-supervised 3D object detection methods, which can be divided into two categories according to the form of annotation: scene-level [35] and position-level [23, 24] with {class tag} and {object center, class tag} annotated for each object respectively. The two types of annotation only require less than 1% and 5% time for one instance compared to labeling a bounding box, as shown in Table 1. While scene-level annotation is more time-saving, it is hard for the detector to learn how to precisely locate each object in a scene due to the lack of position informa-

[*] Corresponding author.

Table 1. Annotating time and detection results of methods based on different types of annotation. The benchmark is detailed in Section 4. (BBox refers to bounding-box annotation. S-L and P-L indicate scene-level and position-level annotations respectively.)

| Annotation | BBox [22] | S-L [35] | P-L [23] | P-L(BR) |
|---|---|---|---|---|
| Time(s per object) | 110 | 1 | 5 | 5 |
| mAP@0.25(%) | 54.2 | <20 | 32.4 | 47.0 |

tion, and thus the performance is far from satisfactory [35]. Considering the time-accuracy tradeoff, position-level annotation is a more practical solution. However, previous position-level weakly-supervised 3D detection methods still require a number of precisely labeled boxes and can only cope with sparse outdoor scenes [23, 24]. Purely position-level weakly-supervised method for the complicated indoor detection task is still under exploration.

In this paper, we propose a shape-guided label enhancement approach named *Back to Reality* (BR) for weakly-supervised 3D object detection[1]. To reduce the labor cost, we only label the center of each object in 3D space and the labeling error of centers is allowed[2]. While largely reducing the workload of labeling, the information loss is non-negligible from box annotations to centers. To solve the problem, BR converts the weak labels into virtual scenes which contain much of the lost information, and in turn utilizes them to additionally supervise real-scene training, as shown in Figure 1. Our approach is based on two motivations: 1) in 3D vision, large-scale datasets of synthetic shapes are available. They contain rich geometry information, which can serve as strong prior to assist 3D object detection; 2) the position-level annotations are not only supervision for training, but they also provide coarse layout of the scene. Therefore, we assemble the 3D shapes into fully-annotated virtual scenes according to the coarse layout and apply physical constraints on them to remedy the information loss. Then a virtual-to-real domain adaptation method is presented to align the global features and object proposal features extracted by the detector between the real and virtual scenes. Moreover, our method can take advantage of the precise center labels in virtual scenes to correct the center error of position-level annotations. In this way the useful knowledge contained in virtual scenes is transferred back to reality. Experimental results on ScanNet [9] show the effectiveness of the proposed BR method.

## 2. Related Work

**3D Shape to Scene:** As it is much easier to obtain a large scale synthetic 3D shape dataset than a real scene dataset, utilizing the shapes to assist scene understanding is a promising idea. Existing approaches can be divided into two categories: supervised [4, 5, 8, 42] and unsupervised [10, 21, 26, 32, 44]. In terms of supervised methods, the synthetic shapes are usually used to complete the imperfect real scene scans. Given a set of CAD models and a real scan, a network is trained to predict how to place the CAD models in the scene and replace the partial and noisy real objects [4,5,8,42]. Human-annotated pairs of raw scans and object-aligned scans are used in the training process. As supervised methods need extra human labor, that may limit the full utilization of 3D shape datasets. Unsupervised methods are often used for data augmentation or dataset expansion. 3D CAD models are placed in a random manner following the basic physical constraints, in order to generate mixed reality scenes [10, 44] or virtual scenes [21, 26]. Recently, RandomRooms [32] proposes to use ShapeNet dataset for unsupervised pretraining of 3D detector. Our approach also utilizes 3D shapes to assist object detection in an unsupervised manner. Differently, we aim to make use of synthetic shapes to enhance the weak label and gain stronger supervision in position-level weakly-supervised detection task.

**3D Object Detection:** Early 3D object detection methods mainly include template-based methods [18,20,25] and sliding-window methods [39, 40]. Deep learning-based 3D detection frameworks for point clouds began to emerge thanks to PointNet/PointNet++ [29, 30]. However, [6, 7, 17, 28] rely on generating 2D proposals and then project them into the 3D space, which is hard to handle scenes with heavy occlusion. More recently, networks that directly consume point clouds have been proposed [13, 22, 27, 37, 53]. While the development of 3D object detection methods is rapid, the application is still restricted partially due to the limited labeled data. To reduce the labor of human annotation, weakly-supervised methods [23, 24, 31, 35], semi-supervised methods [43, 51] and unsupervised pretraining methods [14,32,47,49] have been proposed recently. However, pretraining methods rely on huge computing resources for training the networks in a contrastive learning manner. Semi-supervised methods follow the similar procedure as their 2D counterparts [41] and do not fully exploring the characteristics of 3d data. Therefore, we study weakly-supervised approach tailored for 3D object detection task.

## 3. Approach

Figure 2 illustrates our approach. Given real scenes with position-level annotations, we utilize 3D shapes to convert the weak labels into virtual scenes, which are utilized to provide additional supervision for the training of the detector. In this section, we first discuss our weakly-supervised setting and then demonstrate the two steps of BR.

[1] Label enhancement (LE) is a technique to recover label distributions from logical labels, as defined in [48]. Here we extend the concept of LE to denote the process of recovering the lost information for weak labels.

[2] We show the detailed labeling strategy in Section 3.1.

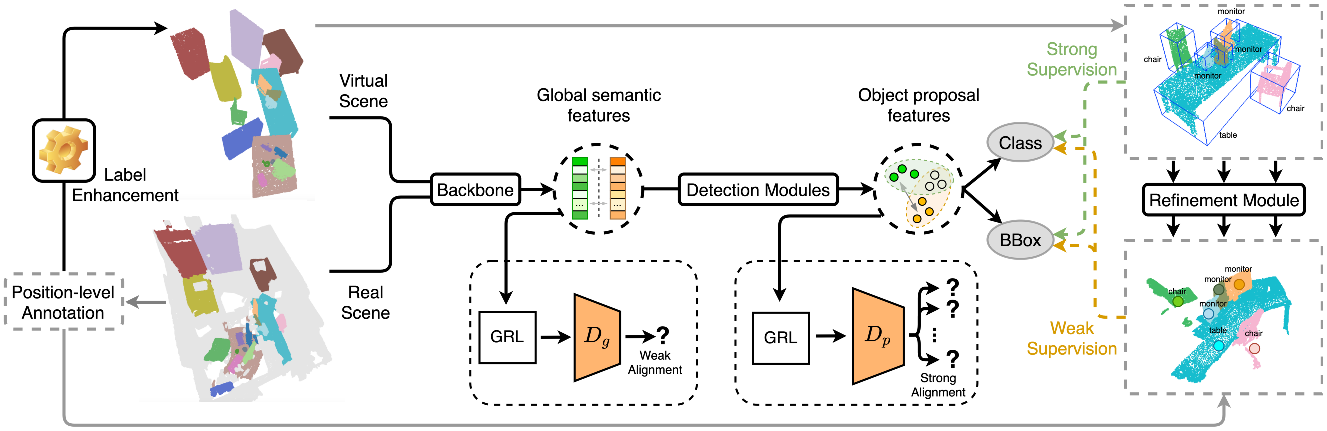


Figure 2. The framework of our *BR* approach. Given real scenes with position-level annotations, we first enhance the weak labels to get fully-annotated virtual scenes. Then the real scenes and virtual scenes are fed into the detector, trained with weakly-supervised and fully-supervised detection loss respectively. During training we use the precise object centers in virtual scenes to refine the imprecise centers in real scenes. Strong-weak adversarial domain adaptation method is utilized to align the distributions of features from both domains. The global discriminator outputs judgments for each scene, and the proposal discriminator outputs judgments for each object proposal. (Here GRL refers to gradient reversal layer; $D_g$ and $D_p$ stand for the global and proposal discriminators respectively.)

### 3.1. Position-level Annotation

As choosing a point in 3D space is hard to operate, we divide the labeling process into two steps: firstly we label the center of an object in a proper 2D view of the scene, and calculate the line that goes through this center and the focus point of the camera according to the camera parameters of the 2D view. Secondly we choose a point on the line to determine the object's center in 3D space. This strategy requires less than 5s to label an instance, and the labeling error can be controlled within 10% of the instance size.

When the 3D scene is scanned, in many cases we can acquire **mesh** data. We assume the meshes are available in our input. Nevertheless, case where we only have point cloud data is also considered in our approach and experiments.

### 3.2. Shape-guided Label Enhancement

While position-level annotation requires far less labeling time, its information loss is severe, which is manifested in two aspects: 1) the information of objects' sizes is lost; 2) the object centers are imprecise. In spite of this, position-level annotations can provide a coarse layout of the scenes. By assembling synthetic 3D shapes according to the layout, we are able to enhance the weak labels and generate accurately-annotated virtual scenes where sizes are available and centers are precise. Our label enhancement method is two-step: 1) first we calculate some basic properties of 3D shapes; 2) then we place these shapes to generate physically reasonable virtual scenes from the labels. For simplicity, we move some implementation details to supplementary[3].

**Definition of Shape Properties:** Given a synthetic 3D shape, which is represented as $O \in R^{N\times3}$, we assume it is axis-aligned and normalized into a unit sphere. The length, width and height of $O$ is defined as $l$, $w$ and $h$. Then we divide the categories of shapes into three classes: supporter, stander and supportee. Supporters and standers are objects that can only be supported by ground, with the difference that standers are not likely to support other things. Other categories are supportees.

Then if a shape belongs to supporter, three properties are calculated: minimum-area enclosing rectangle ($MER^*$), supporting surface height ($SSH^*$) and compactness of the supporter surface ($CSS^*$). The $MER$ is computed in XY plane, which is the minimum rectangle enclosing all the points of the shape. The $SSH$ is the height of the highest surface on which other objects can stand. The $CSS$ is a boolean value, indicating whether the supporting surface can be approximated by the $MER$.

**Virtual Scene Generation:** We utilize a three-stage approach to construct the virtual scenes, which is equivalent to generate the position of each shape stage by stage: 1) we first refine the coarse layout provided by position-level annotations and generate the initial positions; 2) then we generate gravity-aware positions by restoring the supporting relationships between objects; 3) lastly we generate collision-aware positions to make the virtual scenes physically reasonable. The pipeline is shown in Figure 3.

To generate *initial positions*, we need to recover a more precise layout from the geometric information of the scenes. Given a scene in mesh format, we first oversegment the meshes using a normal-based graph cut method [11, 15]. The result is a segment graph, where the nodes indicating segments and the edges denoting adjacency relations. Then for horizontal* segments whose area* is larger than $A_{min}$ and height* is larger than $H_{min}$, we iteratively merge

[3] we use * to indicate that the exact definition is in supplementary.

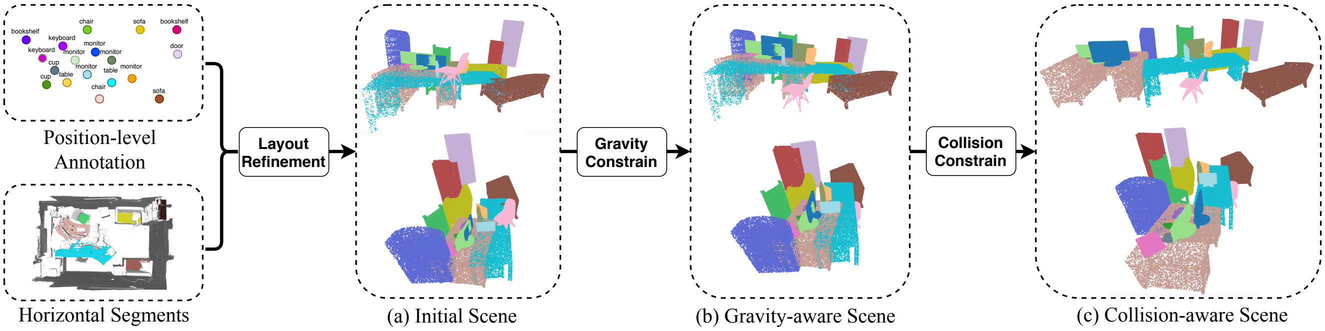


Figure 3. The pipeline of our three-stage virtual scene generation method. We first extract horizontal segments from the mesh data and use them to refine the coarse layout provided by position-level annotations. Then synthetic 3D shapes are placed in virtual scenes according to the new layout to construct initial virtual scenes. After that we apply gravity and collision constraints on the virtual scenes to restore the lost physical relationships between objects and make the scenes more realistic.

their neighbors into them if the height difference between the horizontal segment and the neighbor segment is smaller than $\Delta_h$. Once merged, the segments are considered as a whole and the height of the new merged segment is set to be same as the original horizontal segments. After merging, each horizontal segment is represented by its $MER$. If only one supporter's center falls in a $MER$, we assign this $MER$ to the supporter. When the centers of multiple supporters fall in the same $MER$, we perform K-means clustering of the horizontal segment according to these centers and calculate $MER$ for each supporter respectively.

Then we place the 3D shapes of corresponding categories on the centers given by position-level annotations and utilize the horizontal segments to refine the layout. The initial positions of the shapes are represented by a dictionary, whose key is the instance index and value is a list:

$$[(x,y,z),(s_x,s_y,s_z),O,\theta,S,M,H] \tag{1}$$

where the instance index is a integer ranging from 1 to the number of objects in the scene. $(x, y, z)$ denotes the center coordinates. $(s_x, s_y, s_z)$ indicates the scales in three dimensions. $\theta$ is the rotation angle of the shape. $S$ tells whether the shape is a supporter. $M$ and $H$ indicate the $MER$ and $SSH$ of supporter. They are set to None when $S$ is false. If the shape has been assigned a horizontal segment, we use the $MER$ of that segment to initialize the above parameters. That is, we choose a supporter whose $CSS$ is True and make the $MER$ of this supporter overlap with the horizontal segment. Otherwise we conduct random initialization. If only point cloud data is available, we simply perform random initialization and the following stages are the same.

Next we traverse the initial positions to generate *gravity-aware positions*. In this process we only need to change $z$ and $SSH$ in the position dictionary. For supporters and standers, we directly align their bottoms with the ground (i.e. the XY plane). For a supportee, if its $(x, y)$ fall in any supporter's $MER$, we assign it to the nearest supporter and align its bottoms with the supporting surface. Otherwise, it is aligned to the ground.

After that we move the shapes to acquire *collision-aware positions*. This stage only $x$ and $y$ in the position dictionary will be changed. First we move the objects on the ground, the supported ones on which will move together if there are. Then for each supporter, we move its supportees until there is no overlap. Note that the three generation stages can not only make the virtual scenes more realistic, but also weaken the impact of imprecise center labels. Thus the virtual scene generation method is robust to labeling errors.

Finally, we convert the collision-aware positions to point clouds with proper density. As larger surfaces are more likely to be captured by the sensor, we use the maximum of $(ls_x)(ws_y)$, $(ws_y)(hs_z)$ and $(ls_x)(hs_z)$ to approximate the surface area of shapes. Then the number of points for each object is set proportional to their surface areas using uniform sampling, the largest one remaining $N$ points.

### 3.3. Virtual2Real Domain Adaptation

Although the label enhancement approach is able to generate physically reasonable fully-annotated virtual scenes, there is still a huge domain gap between them and the real scenes (e.g. backgrounds like walls are missed in the virtual scenes). Therefore, we need to mining useful knowledge in the perfect virtual labels to make up for the information loss of position-level annotations, rather than just relying on the virtual scenes.

We refer to the virtual scenes and real scenes as source domain and target domain respectively. A virtual-to-real adversarial domain adaptation method is utilized to solve the above problem, whose overall objective is:

$$\begin{aligned} \max_{D}\min_{O} J &= L_{sup}(O) - L_{adv}(O,D) \\ &= (L_1+L_2+L_3)-(L_4+L_5) \end{aligned} \tag{2}$$

where $O$ refers to the object detection network (detector) and $D$ indicates the discriminators used for adversarial feature alignment. $L_{sup}$ aims to minimize the differences between the predicted bounding boxes and the annotations,

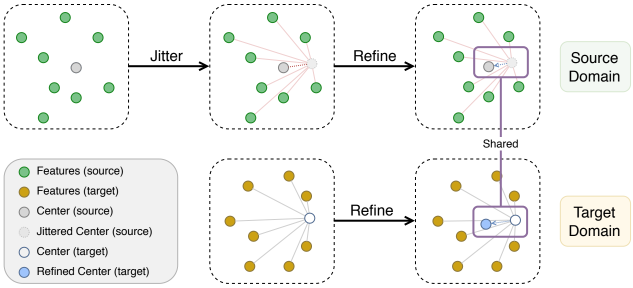


Figure 4. Demonstration of our center refinement method. We first jitter the center labels in source domain, and utilize a PointNet-like module to predict the center offset from the local graph of the jittered centers. This module can be directly utilized to predict the center error in target domain as the global semantic features from the two domains have been aligned.

which can be further divided into the loss for center refinement module ($L_1$), fully-supervised detection loss on source domain ($L_2$) and weakly-supervised detection loss on target domain ($L_3$). The objective of $L_{adv}$ is to align the features from source domain and target domain, which aims to utilize the knowledge learned from source domain to assist object detection in target domain. $L_{adv}$ can be divided into global feature alignment loss ($L_4$) and proposal feature alignment loss ($L_5$). Below we will explain these loss functions and our network in detail.

Firstly we elaborate on $L_{sup}(O)$. As shown in Figure 2, we divide the detector into three blocks: a backbone which extracts global semantic features from the scene, a detection module which generates object proposals from the semantic features, and a prediction head which predicts the semantic label and bounding box from each object proposal feature.

During training, we jointly refine the imprecise center labels in target domain and supervise the predictions of the detector. As shown in Figure 4, we jitter the center labels in source domain by adding noise within 10% of the objects' sizes to imitate the labeling error in target domain. Then for each jittered center, we query its $k$ nearest neighbors in 3D euclidean space from the global semantic features to construct a local graph, and predict the center offset through a PointNet-like module:

$$p(c) = \text{MLP}_2\left\{\max_{i\in N(c)}\{\text{MLP}_1[f_i; c_i - c]\}\right\} \qquad (3)$$

where $p$ denotes the PointNet-like module, $c$ indicates the jittered center label, $N(c)$ is the index set of the $k$ nearest neighbors of $c$, $f_i$ is the global semantic feature, whose coordinate is $c_i$, and *max* refers to the channel-wise max-pooling. We set $\boldsymbol{L_1}$ as the mean square error between the ground-truth center offset and $p(c)$. Then for fully-supervised training, the detection loss $\boldsymbol{L_2}$ is the same as the loss utilized in the original method. For weakly-supervised training, we utilize $p$ to predict the center error in target domain and acquire refined center labels. We set $\boldsymbol{L_3}$ as a simpler version of $L_2$ which ignores the supervision for box sizes. More details about $L_3$ can be found in supplementary.

Secondly we analyze $L_{adv}(O, D)$. We conduct feature alignment in an adversarial manner: the discriminator predicts which domain the features belong to, and the detector aims to generate features that are hard to discriminate. The sign of gradients is flipped by a gradient reversal layer [12].

As the virtual scenes and real scenes are processed by the same network, we hope $L_3$ helps the network learn how to locate each object in real scenes, and $L_2$ compensates for the information loss of centers and sizes. However, due to the domain gap, $L_2$ will introduce domain-specific knowledge of the virtual scenes, which impairs the influence of $L_3$. Besides, the center refinement module is trained only on source domain, which may not perform well on target domain. Therefore, we align the global semantic features and object proposal features with $L_4$ and $L_5$ respectively. Inspired by [36], the features are aligned with different intensities at different stages. For global semantic features, we use a PointNet to predict the domain label. Focal loss [19, 36] is utilized to apply weak alignment:

$$\boldsymbol{L_4} = -\sum_{i=1}^{B}(1-p_i)^{\gamma} log(p_i),\ \gamma > 1 \qquad (4)$$

where $B$ is the batch size, and $p_i$ refers to the probability of the global discriminator's predictions on the corresponding domain. Features with high $p$ is easy to judge, which means they are domain-specific features and forcing invariance to them can hurt performance. So a small weight is used to reduce their impact on training. For object proposal features, they will be directly taken to predict the properties for bounding boxes. As the properties are domain-invariant and have real physical meaning, we strongly align this stage of features using an objectness weighted L2 loss:

$$\boldsymbol{L_5} = \sum_{i=1}^{B}\sum_{j=1}^{N} s_{ij}(1-p_{ij})^2 \qquad (5)$$

where $B$ is the batch size, $N$ is the number of proposals, $s_{ij}$ refers to the objectness label and $p_{ij}$ is the probability of the proposal discriminator's predictions on the corresponding domain. We detail the architectures of center refinement module and discriminators in supplementary.

## 4. Experiment

In this section, we conduct experiments to show the effectiveness of our BR approach. We first describe the datasets and experimental settings. Then we evaluate the generated virtual scenes and report the detection results of our method. We also design experiment to show the robustness of our virtual scene generation method and demonstrate the practicality of our approach. Finally we de-

Table 2. Number of objects in each category in the training set and validation set of ScanNet, and average number of points of objects in each category in the real scenes and the virtual scenes.

| | Property | Bathtub | Bed | Bench | Bookshelf | Bottle | Chair | Cup | Curtain | Desk | Door | Dresser | Keyboard | Lamp | Laptop | Monitor | Nightstand | Plant | Sofa | Stool | Table | Toilet | Wardrobe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # train | Object Number | 113 | 308 | 58 | 786 | 234 | 4357 | 132 | 408 | 551 | 2028 | 174 | 193 | 376 | 86 | 574 | 190 | 293 | 406 | 315 | 1526 | 201 | 98 |
| # validate | | 31 | 81 | 21 | 234 | 41 | 1368 | 34 | 95 | 127 | 467 | 43 | 53 | 83 | 25 | 191 | 34 | 50 | 97 | 51 | 407 | 58 | 19 |
| # real | Point Number | 2941 | 3905 | 1015 | 2679 | 101 | 726 | 66 | 2919 | 1525 | 1110 | 1274 | 74 | 272 | 173 | 370 | 700 | 792 | 2718 | 525 | 1282 | 1445 | 2762 |
| # virtual | | 6891 | 8683 | 4097 | 6258 | 162 | 2135 | 91 | 5495 | 5004 | 6048 | 2703 | 480 | 609 | 343 | 939 | 1088 | 1249 | 7250 | 1391 | 5421 | 3716 | 6105 |

Table 3. The class-specific detection results (mAP@0.25) of different weakly-supervised methods on ScanNetV2 validation set. (FSB is the fully-supervised baseline. $^{\dagger}$ indicates the method requires a small proportion of bounding boxes to refine the prediction. Other methods only use position-level annotations as supervision. We set best scores in bold, runner-ups underlined.)

| | Setting | batht. | bed | bench | bsf. | bot. | chair | cup | curt. | desk | door | dres. | keyb. | lamp | lapt. | monit. | n.s. | plant | sofa | stool | table | toil. | ward. | mAP@0.25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VoteNet | FSB [27] | 66.8 | 86.2 | 24.4 | **55.6** | 0.0 | 88.3 | 0.0 | 48.5 | 62.8 | 45.8 | 24.1 | 0.1 | 47.2 | 5.2 | 62.1 | 73.2 | 13.4 | 88.7 | 35.1 | 62.6 | 94.6 | 7.8 | 45.1 |
| | WSB | 21.9 | 46.9 | 0.3 | 2.3 | 0.0 | 53.7 | 0.0 | 0.9 | 32.1 | 1.0 | 6.6 | 0.1 | 0.2 | 0.1 | 1.8 | 53.6 | 0.1 | 57.0 | 4.6 | 6.4 | 19.7 | 0.0 | 14.1 |
| | WS3D$^{\dagger}$ [23] | 22.0 | 58.5 | 10.3 | 5.8 | 0.0 | 60.4 | 0.0 | 4.1 | 26.7 | 3.2 | 1.6 | 0.0 | 14.0 | 0.6 | 18.6 | 46.3 | 0.4 | 32.7 | 11.8 | 23.5 | 65.0 | 0.0 | 18.4 |
| | WSBP$_P$ | 43.2 | 58.0 | 2.4 | 16.1 | 0.0 | 75.1 | 0.7 | 7.9 | 54.2 | 6.4 | 7.1 | 2.3 | 35.2 | 18.4 | 12.8 | 64.0 | 4.4 | 68.5 | 20.2 | 22.0 | 71.6 | 5.2 | 27.1 |
| | WSBP$_M$ | 45.0 | 49.6 | 5.5 | 18.5 | 0.0 | 62.7 | 2.9 | 11.4 | 49.6 | 6.9 | 2.5 | 1.0 | 30.0 | 7.6 | 21.4 | 64.8 | 7.3 | 79.6 | 23.1 | 35.2 | 80.9 | 2.2 | 27.6 |
| | BR$_P$(Ours) | 51.2 | 73.0 | 16.4 | 27.1 | 0.1 | 70.3 | 0.0 | 8.3 | 44.5 | 7.3 | 16.0 | 1.5 | 40.2 | 7.7 | 42.1 | 50.8 | 7.4 | 67.1 | 10.7 | 39.0 | 88.4 | 18.1 | 31.2 |
| | BR$_M$(Ours) | 57.1 | 80.4 | 14.3 | 31.7 | 0.0 | 77.4 | 0.0 | 13.2 | 49.7 | 11.3 | 14.8 | 1.0 | 43.5 | 6.0 | 56.5 | 65.0 | 10.6 | 80.2 | 26.9 | 44.2 | 91.4 | 6.5 | 35.5 |
| GroupFree3D | FSB [22] | **86.2** | 87.5 | 16.3 | 49.6 | 0.6 | **92.5** | 0.0 | **70.9** | **78.5** | **53.5** | **56.0** | 6.4 | **68.2** | 11.5 | **81.5** | **88.5** | 15.2 | 88.2 | **45.6** | **65.0** | **99.7** | **31.2** | **54.2** |
| | WSB | 75.0 | 75.7 | 4.3 | 17.2 | 0.0 | 81.4 | 0.0 | 3.5 | 34.0 | 4.7 | 3.2 | 2.1 | 46.6 | 3.3 | 45.8 | 52.8 | 8.3 | 71.0 | 15.7 | 18.1 | 90.8 | 0.7 | 29.7 |
| | WS3D$^{\dagger}$ [23] | 71.9 | 78.3 | 0.9 | 20.2 | 0.8 | 79.2 | 1.0 | 2.9 | 47.6 | 7.7 | 10.6 | 19.2 | 41.6 | 13.5 | 65.6 | 41.2 | 0.8 | 74.6 | 17.7 | 26.3 | 88.9 | 1.7 | 32.4 |
| | WSBP$_P$ | 71.9 | 77.1 | 7.7 | 25.2 | **3.0** | 80.6 | 0.4 | 3.2 | 50.1 | 10.5 | 36.3 | 17.0 | 52.9 | **30.3** | 59.9 | 63.8 | 9.6 | 78.2 | 28.4 | 25.3 | 93.3 | 14.4 | 38.2 |
| | WSBP$_M$ | 81.8 | 82.6 | 0.0 | 35.0 | 0.0 | 77.5 | 0.4 | 27.1 | 38.4 | 7.6 | 22.3 | 9.7 | 44.3 | 24.4 | 65.4 | 76.5 | 5.5 | 62.4 | 34.7 | 28.7 | **99.7** | 5.4 | 37.7 |
| | BR$_P$(Ours) | 72.3 | 73.5 | **45.8** | 27.7 | 0.0 | 77.2 | **8.2** | 30.8 | 35.0 | 17.8 | 51.7 | 0.3 | 64.2 | 25.0 | 63.5 | 66.6 | 23.8 | 86.7 | 33.9 | 37.6 | 98.3 | 5.2 | 43.0 |
| | BR$_M$(Ours) | 85.3 | **90.9** | 8.8 | 34.3 | 1.9 | 80.0 | 7.7 | 24.7 | 58.0 | 20.8 | 45.4 | **31.3** | 64.4 | 25.8 | 67.5 | 76.7 | **27.3** | **91.4** | 43.3 | 46.7 | 94.8 | 8.3 | 47.1 |

sign several ablation studies to verify our scene generation method and domain adaptation method in detail.

## 4.1. Experiments Setup

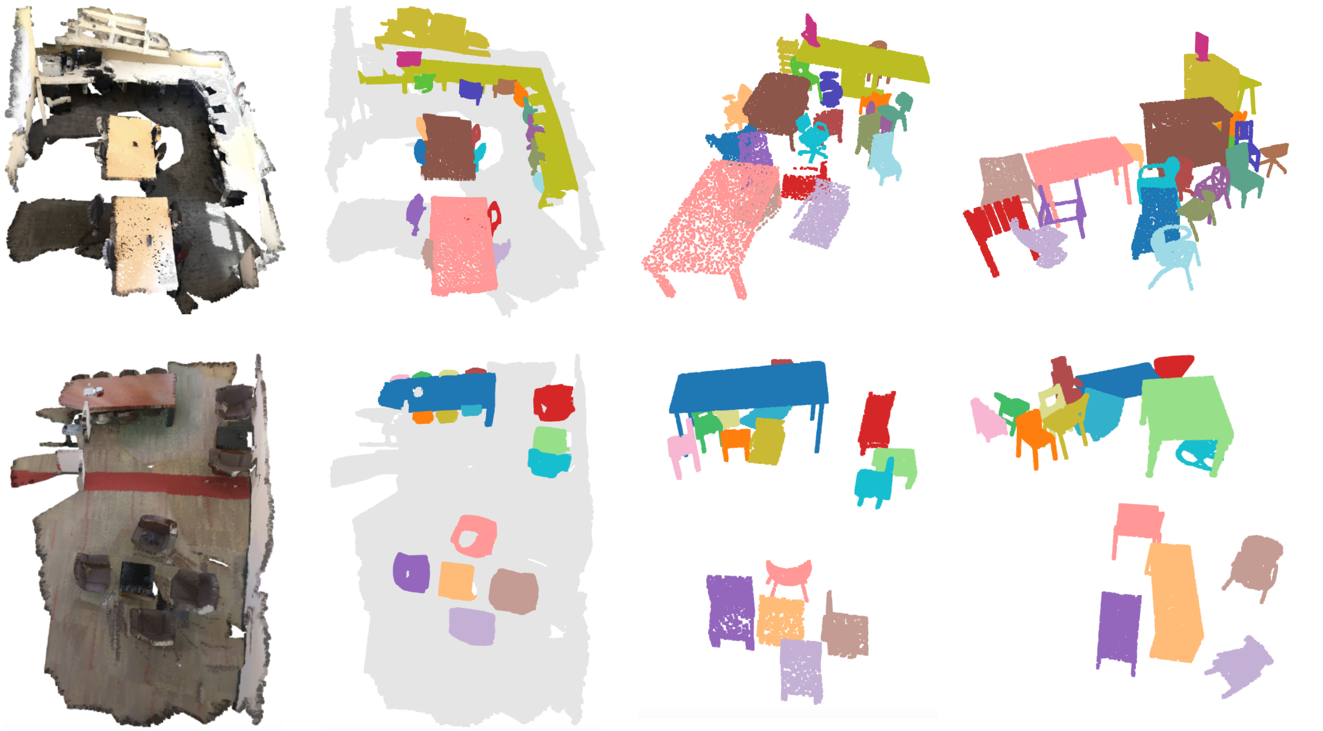
**Datasets:** We choose ModelNet40 [45] as the dataset of synthetic 3D shapes. ModelNet40 contains 12,311 synthetic CAD models from 40 categories, split into 9,843 for training and 2,468 for testing. We perform experiments on ScanNet [9] dataset. ScanNet is a richly annotated dataset of indoor scenes with 1201 training scenes and 312 validation scenes. For each object appeared in the scenes, ScanNet officially provides its corresponding class in ModelNet40. Therefore we choose 22 categories of ModelNet40 which have more than 50 objects in ScanNet training set and 20 in validation set, and report detection performance on them. Since ScanNet does not provide human-labeled bounding boxes, we predict axis-aligned bounding boxes and evaluate the prediction on validation set as in [22,27,46,50]. We name this benchmark ScanNet-md40.

Compared to the 18-category setting in previous works [22, 27, 46], our ScanNet-md40 benchmark is actually more challenging. Apart from the categories of big objects (e.g. desk and bathtub), we also aim to detect relatively small objects, such as laptop, keyboard and monitor. We think our benchmark can better evaluate the performance of both detectors and weakly-supervised learning methods.

**Compared Methods:** To illustrate the effect of our BR approach, the popular VoteNet [27] and state-of-the-art GroupFree3D [22] are chosen as our detectors. We compare BR with the following settings: 1) FSB: fully-supervised baseline, which serves as the upper bound of weakly-supervised methods; 2) WSB: weakly-supervised baseline, which trains the detector on real scenes by using $L_3$ only; 3) WS3D: another position-level weakly-supervised approach proposed in [23], which makes use of a number of precisely annotated bounding boxes; 4) WSBP: WSB pretrained on the virtual scenes. For settings which require the virtual scenes, we conduct experiments on two versions of virtual scenes (from points/meshes), which are distinguished by subscripts $M$ and $P$ respectively.

**Implementation Details:** We set $N = 10000$, $A_{min} = 0.1m^2$, $H_{min} = 0.1m$, $\Delta_h = 0.02m$, $k = 16$ and $\gamma = 3$. During training, as real scenes are more complicated, the converging of $L_3$ is much slower than $L_2$. Therefore we multiple $L_2$ by 0.1 to slow down the training on virtual scenes and stabilize the process of feature alignment. To better train our center refinement module, the global semantic features should not change rapidly. Therefore we first train BR without $L_1$ until convergence, and then use the whole loss function to fine-tune the network. For GroupFree3D which has several decoders and each one outputs a stage of proposal features, we conduct feature alignment only for the last stage.

Different from previous works [22,27], in our setting we need to detect small objects, such as bottle, cup and keyboard. As it is difficult for the network to extract high-quality features of these objects, we utilize an augmentation strategy to alleviate the problem, which is similar to [16].



(a) Real Scenes (b) Cared Objects (c) Mesh-version Virtual Scenes (d) Point-version Virtual Scenes

Figure 5. The qualitative visualization results of our virtual scene generation. In (b), (c) and (d), the same color indicates the same object. Gray points are floors, walls and objects that we do not care. It can be seen that the virtual scenes preserve the coarse scene context and the supporting relationships between objects.

Please refer to supplementary for more details.

### 4.2. Results and Analysis

**Virtual Scene Evaluation:** We first evaluate the statistics of the generated virtual scenes by calculating the average number of points of objects in each category in real scenes and virtual scenes. As the input point clouds are downsampled to a given number before fed into the network, we only care about the ratio of average point numbers of objects in each category as the numbers can be controlled by the downsampling scale. We demonstrate the results in Table 2. It shows that the ratio in our virtual scenes is similar with that in the real scenes, which indicates the statistics of the virtual scenes are reasonable.

We also show qualitative visualizations to demonstrate our scene generation method in Figure 5. The virtual scenes generated with mesh information are named as mesh-version virtual scenes. Otherwise they are named as point-version virtual scenes. It is shown that the mesh-version virtual scenes can largely preserve the layout of the real scenes, and the point-version ones successfully combine the individual 3D shapes in a meaningful way.

**3D Object Detection Results:** As shown in Table 7, with position-level annotations only, WSB reduces the detection accuracy by a large margin in terms of mAP@0.25 compared to FSB. That's mainly because WSB fails to learn the ability of predicting precise centers and sizes of bounding boxes according to the scene context. WS3D makes use of some box annotations and achieve better performance. However, as it is specially designed for outdoor 3D object detection, WS3D is still far from satisfactory when coping with the complicated indoor scenes. With pretraining on the virtual scenes, WSBP has more than 8% improvement over the WSB. That shows the ability of predicting precise bounding boxes learned in the source domain has been successfully transferred to the target domain. With our domain adaptation method to conduct better transferring, the improvement over the WSB is boosted to a higher level. The above results shows each step in BR is necessary: the virtual scenes are helpful to boost the detection performance, and the domain adaptation method can further explore the potential of the virtual scenes. Interestingly, as the virtual scenes become more realistic (from point-version to mesh-version), the performance of BR improves a lot while WSBP has little change, which indicates that layout may not be that important in pretraining as in domain adaptation.

In terms of class-specific results, on some categories the mAP@0.25 of the $\mathrm{BR}_M$ (for GroupFree3D) is even the highest among all the methods including the FSB. However, all methods fail to precisely detect cup and bottle, which

Table 4. The detection results (mAP@0.25) of BR under different error rate for center labeling on ScanNet. We adopt GroupFree3D as the detector and utilize mesh-version virtual scenes for BR.

| Method | Error Rate | | | | |
|---|---|---|---|---|---|
| | 10% | 20% | 30% | 40% | 50% |
| WSB | 29.7 | 26.8 | 25.0 | 22.3 | 19.7 |
| $\text{BR}_M$(Ours) | **47.1** | **46.0** | **43.9** | **43.1** | **41.2** |

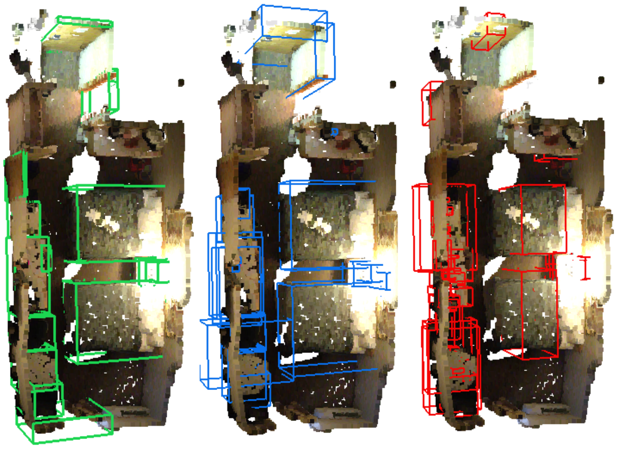


Figure 6. Visual Results on ScanNet. We compare BR and WSB with the ground-truth bounding boxes.

shows current 3D detectors still face huge challenges in small object detection. More detection results (mAP@0.5) can be found in supplementary.

**Robustness for Labeling Error:** In our labeling strategy, the center error is within 10%, which we define as the error rate, of the object's size. To show the robustness of our approach, we gradually increase this rate from 10% to 50% by randomly jittering the centers according to the box sizes, and report the detection results of WSB and $\text{BR}_M$ (for GroupFree3D) in terms of mAP@0.25. As shown in Table 4, with the increasing of error rate, the performance of BR degrades more slowly than WSB. Even if the error rate is 50%, which allows us to label the centers in a more time-saving strategy, BR can still achieve satisfactory results (higher than 0.41 in terms of mAP@0.25).

**Visualization Results:** We visualize the detection results of WSB and $\text{BR}_M$ (for GroupFree3D) on ScanNet. As shown in Figure 6, BR can produce more accurate detection results with less false positives. The visual results further confirm the effectiveness of the proposed method.

## 4.3. Ablation Study

We further design ablation experiments to study the influences of each scene generation step and each domain adaptation loss to the performance of our BR approach. In this section, we adopt VoteNet as the detector and use point-version virtual scenes for universality.

Table 5. The detection results (mAP@0.25) of BR with virtual scenes at different generation stages on ScanNet. Here the detector is VoteNet and the virtual scenes are point-version.

| Gravity Constrain | Collision Constrain | Density Control | mAP@0.25 |
|---|---|---|---|
| | | | 26.3 |
| ✓ | | | 27.2 |
| ✓ | ✓ | | 28.5 |
| ✓ | ✓ | ✓ | **31.2** |

Table 6. The detection results (mAP@0.25) of BR with different domain adaptation modules on ScanNet. Here the detector is VoteNet and the virtual scenes are point-version.

| Global Alignment | Proposal Alignment | Center Refinement | mAP@0.25 |
|---|---|---|---|
| | | | 24.2 |
| ✓ | | | 28.7 |
| | ✓ | | 27.4 |
| ✓ | ✓ | | 30.2 |
| ✓ | ✓ | ✓ | **31.2** |

In Table 5, we illustrate that in our virtual scene generation pipeline, the physical constraints and density control are effective. As the virtual scenes become more realistic, the performance of our BR approach is getting better.

As shown in Table 6, we show the effect of each domain adaptation module and the center refine module. It can be seen that with global alignment or object proposal alignment, the detection performance can be boosted by 3.5% and 2.2% respectively. By combining the two kinds of feature alignments, we are able achieve higher detection accuracy. Then after applying the center refinement method, the performance is further boosted by 1.0%.

## 4.4. Limitation

Due to the limited number of categories in ModelNet40, we selectively evaluate the performance of BR on 22 classes. However, as online repositories of user-generated 3D shapes, such as the 3D Warehouse repository [3], contain 3D shapes in almost any category, BR can be easily extended to 3D object detection on more classes once these online synthetic shapes are organized into a standard dataset. Therefore, ideally we can leverage a larger synthetic 3D shape dataset, which covers all objects that may appear in indoor scenes. This dataset can promote more researches on 3D scene understanding with synthetic shapes, which we leave for future work.

## 5. Conclusion

In this paper, we have proposed a new label enhancement approach, namely Back to Reality (BR), for 3D object detection trained using only object centers and class tags as supervision. To fully explore the information contained in the position-level annotations, we regard them as the coarse layout of scenes, which is utilized to assemble 3D shapes into fully-annotated virtual scenes. We apply physical constraints on the generated virtual scenes to make sure the relationship between objects is reasonable. Then in order to make use of the virtual scenes to remedy the information loss from box annotations to centers, we present a virtual-to-real domain adaptation method, which transfers the useful knowledge learned from the virtual scenes to real-scene 3D object detection. Experimental results on ScanNet dataset show the effectiveness of our BR approach.

## Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62125603, and Grant U1813218, and in part by a grant from the Beijing Academy of Artificial Intelligence (BAAI).

## References

[1] Open3d: A modern library for 3d data processing. [EB/OL]. http://www.open3d.org/. 11

[2] Opencv. [EB/OL]. https://opencv.org/. 10

[3] Trimble 3d warehouse. [EB/OL]. http://3dwarehouse.sketchup.com/. 8

[4] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *CVPR*, pages 2614–2623, 2019. 2

[5] Armen Avetisyan, Angela Dai, and Matthias Nießner. End-to-end cad model retrieval and 9dof alignment in 3d scans. In *ICCV*, pages 2551–2560, 2019. 2

[6] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, pages 2147–2156, 2016. 2

[7] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, pages 1907–1915, 2017. 2

[8] Manuel Dahnert, Angela Dai, Leonidas J Guibas, and Matthias Niessner. Joint embedding of 3d scan and cad objects. In *ICCV*, pages 8749–8758, 2019. 2

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828—-5839, 2017. 2, 6, 13

[10] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 2

[11] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 3

[12] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015. 5

[13] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *CVPR*, pages 4421–4430, 2019. 1, 2

[14] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *CVPR*, pages 15587–15597, 2021. 2

[15] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3d scenes via shape analysis. In *ICRA*, pages 2088–2095, 2013. 3

[16] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*, 2019. 6, 12

[17] Jean Lahoud and Bernard Ghanem. 2d-driven 3d object detection in rgb-d images. In *ICCV*, pages 4622–4630, 2017. 2

[18] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *CGF*, volume 34, pages 435–446, 2015. 2

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 5

[20] Or Litany, Tal Remez, Daniel Freedman, Lior Shapira, Alex Bronstein, and Ran Gal. Asist: automatic semantically invariant scene transformation. *CVIU*, 157:284–299, 2017. 2

[21] Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, pages 529–537, 2019. 2

[22] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. *arXiv preprint arXiv:2104.00678*, 2021. 1, 2, 6, 11, 13

[23] 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*, pages 515–531, 2020. 1, 2, 6, 12, 13

[24] Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Yunde Jia, and Luc Van Gool. Towards a weakly supervised framework for 3d point cloud object detection and annotation. *TPAMI*, 2021. 1, 2

[25] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *TOG*, 31(6):1–10, 2012. 2

[26] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, pages 523–540, 2020. 2

[27] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, pages 9277–9286, 2019. 1, 2, 6, 11, 13

[28] 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*, pages 918–927, 2018. 2

[29] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 1, 2

[30] 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*, pages 5099–5108, 2017. 1, 2, 11

[31] Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3d object detection from point clouds. In *ACM MM*, pages 4144–4152, 2020. 2

[32] Yongming Rao, Benlin Liu, Yi Wei, Jiwen Lu, Cho-Jui Hsieh, and Jie Zhou. Randomrooms: Unsupervised pre-training from synthetic shapes and randomized layouts for 3d object detection. In *ICCV*, pages 3283–3292, 2021. 2

[33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 1

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *TPAMI*, 39(6):1137–1149, 2016. 1

[35] Zhongzheng Ren, Ishan Misra, Alexander G Schwing, and Rohit Girdhar. 3d spatial recognition without spatially labeled 3d. In *CVPR*, pages 13204–13213, 2021. 1, 2

[36] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection. In *CVPR*, pages 6956–6965, 2019. 5

[37] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, pages 770–779, 2019. 1, 2

[38] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576, 2015. 1

[39] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, pages 634–651, 2014. 2

[40] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *CVPR*, pages 808–816, 2016. 2

[41] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017. 2

[42] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-aware 3d model embedding and retrieval. In *ECCV*, pages 397–413, 2020. 2

[43] He Wang, Yezhen Cong, Or Litany, Yue Gao, and Leonidas J Guibas. 3dioumatch: Leveraging iou prediction for semi-supervised 3d object detection. In *CVPR*, pages 14615–14624, 2021. 2

[44] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*, pages 2642–2651, 2019. 2

[45] 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*, pages 1912–1920, 2015. 6

[46] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. Mlcvnet: Multi-level context votenet for 3d object detection. In *CVPR*, pages 10447–10456, 2020. 6

[47] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *ECCV*, pages 574–591, 2020. 2

[48] Ning Xu, Yun-Peng Liu, and Xin Geng. Label enhancement for label distribution learning. *TKDE*, 2019. 2

[49] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. *arXiv preprint arXiv:2101.02691*, 2021. 2

[50] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *ECCV*, pages 311–329, 2020. 6

[51] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. Sess: Self-ensembling semi-supervised 3d object detection. In *CVPR*, pages 11079–11087, 2020. 2

[52] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1

[53] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, pages 4490–4499, 2018. 1, 2

# Supplementary Material

## A. overview

This supplementary material[4] is organized as follows:

- Section 1 details the Approach section in the main paper.
- Section 2 shows the implementation detail of WS3D.
- Section 3 details our augmentation strategy for small objects during training.
- Section 4 shows more experimental results.

## B. Approach Details

In this section, we show the details in our approach, which is divided into shape-guided **label enhancement** and virtual2real **domain adaptation**.

### B.1. Label Enhancement

We show the exact definitions of some concepts appeared in Section 3.2 of the main paper as below.

**Shape Properties:** The $MER$ is computed in XY plane, which is the minimum rectangle enclosing all the points of the object template. The $SSH$ is the height of the largest surface on which other objects can stand. The $CSS$ is a boolean value, indicating whether the supporting surface is similar with the $MER$ (i.e. we can use the $MER$ to approximate the supporting surface if $CSS$ is true).

In order to calculate $MER$, we use the OpenCV [2] toolbox to calculate the $MER$ of 2D point set. As OpenCV cannot be directly utilized to process point clouds, we first

[4] We include our code in the folder "BackToReality". Please refer to the README file for more details.

project the object templates to XY plane to acquire 2D point sets. Then we calculate the $MER$ of a point set $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ as below:

$$(x, y, l, w, \theta) = \text{minAreaRect}(1000 * S) \tag{6}$$

$$MER = (x, y, \frac{l}{1000}, \frac{w}{1000}, \theta) \tag{7}$$

where minAreaRect is a function in OpenCV, which takes integer 2D point set as input and returns a rectangle, and rectangle is represented by a quintuple $(x, y, length, width, \theta)$, which indicates the center coordinate, length, width and rotation angle of a rectangle. $1000 * S$ means that we multiply all the coordinates in $S$ by 1000 and then convert the coordinates from float to integer, which can reduce the rounding error.

To compute $SSH$, we first utilize Open3D [1] to get the normals of each point from point cloud. Then if the normal of a point is almost vertical (i.e. the normal's length along Z-axis is greater than 0.88), we record the coordinate of this point. After traversing all the points, we have recorded a list of coordinates. We sort the list according to the Z coordinate in ascending order, and the list of sorted Z coordinate is named as $l_z$. Then get a slice of $l_z$ from index $\lfloor\frac{4}{5}len_z\rfloor$ to $\lfloor\frac{9}{10}len_z\rfloor$, where $len_z$ denotes the length of $l_z$. $SSH$ can be calculated by averaging this slice. Note that this algorithm suppose the supporter has a large supporting surface on its top, and it can tolerate 10% points higher than this surface.

To calculate $CSS$, we collect points which satisfy $SSH - \frac{1}{10}h < z < SSH + \frac{1}{10}h$ from the given object template, where $h$ is the height of this object template. Then we project these points to XY plane and name them supporter points $P_S$. If $P_S$ can almost fill the $MER$, the $CSS$ is set to be $True$. To analyze the compactness, we use K-means algorithm to divide $P_S$ into 2 clusters: $P_{S1}$ and $P_{S2}$. Then we calculate the area of convex hull of $P_{S1}$ and $P_{S2}$. The area is computed by using OpenCV:

$$A = \frac{\text{contourArea}(\text{convexHull}(1000 * P))}{1000000} \tag{8}$$

where contourArea and convexHull are functions in OpenCV, $P$ is a 2D point set and $A$ is the area of $P$. The areas for $P_{S1}$ and $P_{S2}$ are $A_1$ and $A_2$ respectively. So we can compute $CSS$ as below:

$$CSS = \begin{cases} True, & A_1 + A_2 > 0.9 * l * w \\ False, & otherwise \end{cases} \tag{9}$$

where $l$ and $w$ are the length and width of the $MER$ of this object template.

**Segment Properties:** Next we provide the definitions of horizontal segment, the area of segment and the height of segment.

For a segment, we define $z$ as the Z coordinate of all the points on it. Then if $|maximum(z) - median(z)| < 0.2$ or $|minimum(z) - median(z)| < 0.2$, we consider this segment is horizontal. To calculate the area of segment, we directly utilize (8) and take all points on the segment as input (ignore the Z coordinates of points). To compute the height of a segment, we follow the same procedure as computing $SSH$: we first calculate the normals and pick out points with normals that are almost vertical, and then we pick out the Z coordinates of these points and acquire a list $l_z$. The segment's height is defined as the mean of $l_z$.

## B.2. Domain Adaptation

We first provide detailed definition of $L_3$. Then we show the architectures of our center refinement module and the two discriminators.

For weakly-supervised training, as only objects' centers and semantic classes are available, we set $L_3$ as a simpler version of $L_2$:

$$L_3 = L_f + L_i,\ L_f = L_s + L_o + L_c \tag{10}$$

$L_f$ is used to supervise the final prediction, where $L_s$ and $L_o$ are the cross entropy losses for semantic labels and objectness scores, and $L_c$ is defined as:

$$L_c = \sum_i max(||C_{gi} - C_i||_2 - \lambda S_{gi}, 0) \tag{11}$$

which denotes the hinge loss for centers. $C_i$ is the $i$-th predicted center, $C_{gi}$ is the nearest ground-truth center to $C_i$, and $S_{gi}$ indicates the average size for the semantic class of this object. We set $\lambda = 0.05$ to approximate the labeling error of centers. For $L_i$, we only make use of the center coordinates to weakly supervise the intermediate process of training. For example, in VoteNet [27], the detection module predicts votes from the semantic features and aggregate them to generate object proposals, in which voting coordinates are the intermediate variables need to be supervised. Here we utilize the Chamfer Distance between the voting coordinates and the ground-truth center coordinates to supervise the voting. In GroupFree3D [22], the detection module utilize KPS to sample the semantic features and generate initial object proposals, where the sampled points require supervision. Originally the KPS operation requires us to sample the nearest k points to the object center from the point cloud belong to this object. However, we weaken this requirement and sample the nearest k points without any constraints.

For the center refinement module, we adopt the Set Abstraction (SA) layer [30] to extract feature from the local KNN graph. Then a MLP is utilized to predict center offset from the feature. The SA layer first concatenates the relative coordinates between the center and its neighbors to

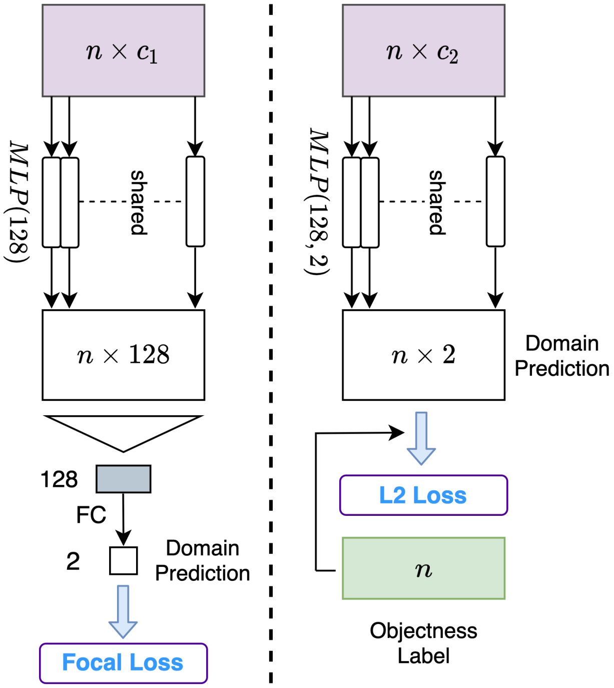


Figure 7. Architecture of the global and proposal discriminators. (Global on the left, proposal on the right.)

the features of the neighbors, which is followed by a shared MLP ($MLP(256, 128)$[5]) and a channel-wise max-pooling layer. The pooled feature contains the local information of the center, which is then concatenated with the one-hot vector of the center's semantic class (we name the feature after concatenation as center feature). We utilize another MLP ($MLP(64, 3)$) to predict the center offset from the center feature. For the global and proposal discriminators, we show their architectures in Figure 7.

## C. Implementation Detail of WS3D

In this section, we show how we implement WS3D [23] to adapt to indoor 3D object detection task.

### C.1. Introduction of WS3D

Here is a simple summary of WS3D: The authors annotate the object centers in the bird's eye view (BEV) maps, which takes 2.5s per object. Then they utilize a two-stage approach to detect a specific category of objects (the author focus on **Car** in their paper), which can be divided into proposal and refinement stages. At the proposal stage, WS3D creates cylindrical proposals from the labeled centers, whose radius and height are fixed since the sizes of cars are close. Therefore the probability of a car being wrapped in a cylindrical proposal is high. Then a network (Net1) is trained to generate proposals from a point cloud scene. At the refinement stage, another network (Net2) is trained to take in the cylindrical proposal and output the bounding box of the car contained in the proposal, where around 3% well-labeled instances are used for supervision.

### C.2. Proposal Stage

Since the indoor scenes in ScanNetV2 are more complicated, the size and height of each object is different, even for objects in the same class. Therefore we annotate the object centers in 3D space rather than in the BEV map, which is the same labeling strategy with us and takes 5s per object, to provide stronger supervision for WS3D. Instead of using a simple fixed-size cylinder as the proposal, we utilize a cuboid instead, whose size (length, width and height) is 1.5 times the average size of the object's category. In this way we are able to generate a more reasonable proposal.

During this stage, we can adopt different detectors as Net1. Net1 is trained with position-level annotations and used to predict the centers and semantic labels of objects (we adopt VoteNet and GroupFree3D as Net1 in our experiments). Then we generate cuboid proposals from the predicted centers and classes.

### C.3. Refinement Stage

We find 3% well-labeled bounding boxes are not enough to train the Net2, as there 22 categories in our benchmark and the size of each object is very different, so we use around 15% bounding boxes instead. The proposals generated from the previous stage are post-processed by a 3D NMS module with an IoU threshold of 0.25, and then refined into precise bounding boxes by Net2.

We adopt a PointNet++-like module as Net2, whose input is the point cloud inside the cuboid proposal and output is the refined center coordinate, box size and box orientation.

## D. Augmentation Strategy

As the number of scenes which contain small objects[6] and the probability of small objects being sampled are relatively smaller than others, it is difficult for the detector to learn how to locate small objects in complex scenes. Therefore we utilize an augmentation strategy similar to [16] to handle the problem.

During trianing, we oversample the virtual scenes which contain small objects twice in each epoch. We further copy-paste small objects to the oversampled virtual scenes: for each small object, we copy it with a probability of 0.75 and paste it randomly in the scene (the pasted center must be in the axis-aligned bounding box of the whole scene). Then we apply gravity and collision contraints and control the

[5] Numbers in bracket are output layer sizes. Batchnorm is used for all layers with ReLU except for the final prediction layer in $MLP(64, 3)$.

[6] Small objects are {bottle, cup, keyboard}.

Table 7. The class-specific detection results (mAP@0.5) of different weakly-supervised methods on ScanNetV2 validation set. (FSB is the fully-supervised baseline. $^\dagger$ indicates the method requires a small proportion of bounding boxes to refine the prediction. Other methods only use position-level annotations as supervision.)

| | Setting | batht. | bed | bench | bsf. | bot. | chair | cup | curt. | desk | door | dres. | keyb. | lamp | lapt. | monit. | n.s. | plant | sofa | stool | table | toil. | ward. | mAP@0.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VoteNet | FSB [27] | 69.8 | 76.9 | 6.7 | 26.0 | 0.0 | 67.6 | 0.0 | 10.2 | 30.0 | 13.3 | 21.1 | 0.0 | 15.5 | 0.0 | 19.6 | 47.9 | 3.1 | 70.4 | 10.1 | 38.9 | 85.0 | 2.7 | 28.0 |
| | WSB | 0.0 | 11.5 | 0.0 | 0.0 | 0.0 | 1.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.6 | 0.0 | 0.2 | 0.7 | 0.0 | 0.2 | 0.0 | 0.1 | 2.5 | 0.0 | 0.8 |
| | WS3D $^\dagger$ [23] | 0.0 | 22.7 | 0.0 | 0.0 | 0.0 | 12.2 | 0.1 | 0.0 | 0.3 | 0.0 | 0.0 | 0.0 | 1.1 | 0.0 | 1.3 | 11.3 | 0.0 | 0.1 | 0.2 | 1.4 | 16.4 | 0.0 | 3.1 |
| | WSBP$_P$ | 0.0 | 3.7 | 0.0 | 0.1 | 0.0 | 28.4 | 0.0 | 0.0 | 1.1 | 0.5 | 0.0 | 0.0 | 1.2 | 0.0 | 0.0 | 26.1 | 0.0 | 0.9 | 4.4 | 0.8 | 7.6 | 0.6 | 3.4 |
| | WSBP$_M$ | 12.3 | 1.3 | 0.0 | 0.3 | 0.0 | 16.5 | 0.0 | 0.0 | 4.1 | 0.1 | 0.0 | 0.4 | 5.9 | 0.0 | 0.1 | 26.9 | 0.4 | 3.3 | 5.4 | 0.8 | 4.8 | 0.1 | 3.8 |
| | BR$_P$(Ours) | 36.8 | 15.2 | 1.2 | 6.9 | 0.0 | 42.7 | 0.0 | 0.0 | 4.4 | 1.3 | 2.1 | 0.0 | 9.0 | 0.0 | 2.7 | 31.4 | 1.3 | 14.4 | 4.1 | 8.3 | 51.6 | 0.0 | 10.6 |
| | BR$_M$(Ours) | 9.6 | 59.2 | 0.2 | 12.8 | 0.0 | 37.9 | 0.0 | 0.0 | 22.1 | 1.0 | 6.2 | 0.0 | 10.6 | 0.0 | 2.1 | 44.6 | 2.7 | 33.0 | 2.0 | 25.3 | 57.0 | 0.1 | 14.8 |
| GroupFree3D | FSB [22] | 75.7 | 75.6 | 4.5 | 28.4 | 0.0 | 75.3 | 0.0 | 20.3 | 47.4 | 24.7 | 29.5 | 0.3 | 20.4 | 0.0 | 37.5 | 61.4 | 3.7 | 74.6 | 37.1 | 51.1 | 96.2 | 11.7 | 35.2 |
| | WSB | 1.9 | 24.7 | 0.0 | 0.1 | 0.0 | 31.2 | 0.0 | 0.0 | 0.1 | 0.1 | 0.0 | 0.0 | 6.5 | 0.0 | 2.1 | 1.5 | 0.1 | 2.6 | 2.0 | 0.5 | 54.3 | 0.0 | 5.8 |
| | WS3D$^\dagger$ [23] | 3.8 | 25.7 | 0.0 | 0.1 | 0.0 | 36.4 | 0.0 | 0.0 | 2.1 | 0.0 | 0.3 | 0.3 | 10.2 | 0.0 | 7.5 | 16.4 | 0.2 | 2.7 | 4.5 | 0.4 | 68.3 | 0.0 | 8.1 |
| | WSBP$_P$ | 1.9 | 5.2 | 0.0 | 1.3 | 0.0 | 31.8 | 0.0 | 11.3 | 1.1 | 0.1 | 0.0 | 0.0 | 18.7 | 4.4 | 1.0 | 48.1 | 1.3 | 1.3 | 1.3 | 0.6 | 62.0 | 1.8 | 8.3 |
| | WSBP$_M$ | 4.9 | 16.7 | 0.0 | 0.5 | 0.0 | 34.1 | 0.0 | 0.1 | 5.6 | 0.2 | 0.5 | 0.1 | 9.0 | 4.6 | 8.9 | 48.5 | 0.9 | 9.9 | 12.3 | 3.4 | 51.9 | 0.0 | 9.6 |
| | BR$_P$(Ours) | 83.6 | 79.1 | 0.0 | 10.8 | 0.0 | 53.5 | 0.0 | 0.0 | 0.0 | 1.6 | 3.7 | 0.0 | 19.6 | 50.0 | 6.5 | 60.0 | 16.7 | 21.1 | 5.7 | 14.6 | 90.1 | 0.0 | 23.5 |
| | BR$_M$(Ours) | 83.3 | 65.0 | 0.0 | 4.1 | 0.0 | 56.2 | 0.0 | 0.5 | 11.8 | 2.1 | 16.7 | 1.2 | 23.8 | 12.5 | 16.0 | 80.0 | 17.5 | 42.2 | 28.6 | 28.0 | 99.2 | 0.0 | 26.8 |

densities of these added small objects as mentioned in the virtual scene generation method.

Apart from small objects, we also consider the scarce objects[7], as the number of them is relatively small and thus the detector is not sufficiently trained on these categories. We add the scarce objects to the oversampled virtual scenes to expand the number of them. We first decide how many objects of each scarce category we should add according to Table 2 in the main paper, where we set 40, 70, 15, 55 and 50 for bathtub, bench, dresser, laptop and wardrobe respectively. Then we choose scenes which are suitable for adding these objects by calculating the value of correlation between scenes and scarce categories as below:

$$Corr(s,c) = \sum_{i=1}^{22} l_{s_i}(v_{c_i} - r) \tag{12}$$

where $s$ indicates a scene and $c$ denotes a scarce category. $l_s$ is a 22-dimensional boolean vector where $l_{s_i}$ indicates whether there is an object of the $i$-th category in $s$. $v_c$ is a 22-dimensional vector which indicates the correlation between $c$ and other categories:

$$v_{c_i} = \begin{cases} \frac{Num(i,Index(c))}{Num(Index(c))}, & i \neq Index(c) \\ 0, & i = Index(c) \end{cases} \tag{13}$$

where $Num(...)$ is a function, whose input is a set of indexs of category and output is the number of scenes which contain objects in all the input categories. The larger $v_{c_i}$, the stronger the correlation between $c$ and the $i$-th category. As we hope the highly correlated scenes for $c$ do not contain too many categories with low $v_{c_i}$, we introduce a penalty term $r$ to reduce the value of $Corr(s,c)$ when there are a large number of categories weakly correlated to $c$ in $s$. We set $r = 0.25$ in our experiments.

## E. More Detection Results

We show 3D object detection results (mAP@0.5) of different weakly-supervised methods on ScanNetV2 [9] validation set in Table 7.

Consistent with the results on mAP@0.25, our BR approach achieves the best performance among all the weakly-supervised approaches. Under a more strict metric, the performances of most weakly-supervised approaches fail to surpass 10% in terms of mAP@0.5, that shows it is really hard to precisely detect the objects in a complicated indoor scene for a detector trained with only position-level annotations. However, the performance of BR$_M$ (for GroupFree3D) still achieves 26.8% in terms of mAP@0.5, which is comparable to the performance of fully-supervised VoteNet.

We also find our BR approach works better on GroupFree3D than on VoteNet (the gap between FSB and BR is smaller). This may be due to the features extracted by stronger detector has better generalization ability and thus our virtual2real domain adaptation method can transfer more useful knowledge contained in the virtual scenes to real-scene training.

[7] Scarce objects are {bathtub, bench, dresser, laptop, wardrobe}.