Supplementary Material for 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

This supplementary material 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.*

1. Approach Details

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

1.1. 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 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:

$$(x, y, l, w, \theta) = \text{minAreaRect}(1000 * S) \tag{1}$$

$$MER = (x, y, \frac{l}{1000}, \frac{w}{1000}, \theta) \tag{2}$$

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:

$$A = \frac{\text{contourArea}(\text{convexHull}(1000 * P))}{1000000} \tag{3}$$

*Corresponding author.

Table 1. The class-specific detection results (mAP@0.5) of different weakly-supervised methods on ScanNetV2 validation set. (FSB is the fully-supervised baseline. † 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 [7]	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 † [6]	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 [5]	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† [6]	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

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:

$$CSS = \begin{cases} True, & A_1 + A_2 > 0.9 * l * w \\ False, & otherwise \end{cases} \tag{4}$$

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 (3) 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.

1.2. 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:

$$L_3 = L_f + L_i,\ L_f = L_s + L_o + L_c \tag{5}$$

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:

$$L_c = \sum_i max(||C_{gi} - C_i||_2 - \lambda S_{gi}, 0) \tag{6}$$

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 [7], 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 [5], 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 [8] 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 the features of the neighbors, which is followed by a shared MLP ($MLP(256, 128)$[1]) 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

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

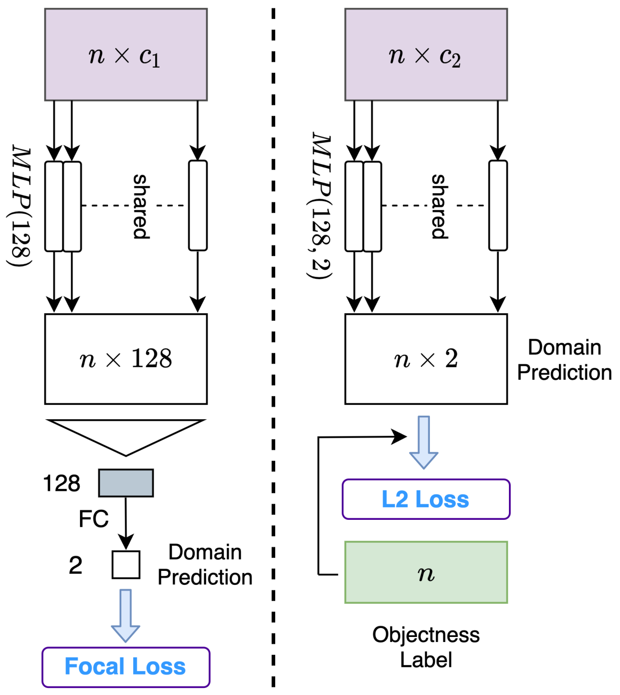

Figure 1. Architecture of the global and proposal discriminators. (Global on the left, proposal on the right.)

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 1.

2. Implementation Detail of WS3D

In this section, we show how we implement WS3D [6] to adapt to indoor 3D object detection task.

2.1. 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.

2.2. 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.

2.3. 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.

3. Augmentation Strategy

As the number of scenes which contain small objects[2] 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 [4] 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 densities of these added small objects as mentioned in the virtual scene generation method.

Apart from small objects, we also consider the scarce objects[3], as the number of them is relatively small and thus the detector is not sufficiently trained on these categories.

[2]Small objects are {bottle, cup, keyboard}.
[3]Scarce objects are {bathtub, bench, dresser, laptop, wardrobe}.

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:

$$Corr(s,c) = \sum_{i=1}^{22} l_{s_i}(v_{c_i} - r) \tag{7}$$

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:

$$v_{c_i} = \begin{cases} \frac{Num(i,Index(c))}{Num(Index(c))}, & i \neq Index(c) \\ 0, & i = Index(c) \end{cases} \tag{8}$$

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.

4. More Detection Results

We show 3D object detection results (mAP@0.5) of different weakly-supervised methods on ScanNetV2 [3] validation set in Table 1.

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.

References

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

[2] Opencv. [EB/OL]. https://opencv.org/. 1

[3] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828—-5839, 2017. 4

[4] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*, 2019. 3

[5] Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong. Group-free 3d object detection via transformers. *arXiv preprint arXiv:2104.00678*, 2021. 2

[6] Q. Meng, W. Wang, T. Zhou, J. Shen, L. V. Gool, and D. Dai. Weakly supervised 3d object detection from lidar point cloud. In *ECCV*, pages 515–531, 2020. 2, 3

[7] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, pages 9277–9286, 2019. 2

[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 2