Supplementary Material for Rotationally Equivariant 3D Object Detection

Hong-Xing Yu
Stanford University

Jiajun Wu
Stanford University

Li Yi
Tsinghua University, Shanghai Qi Zhi Institute

Abstract

In this supplementary document, we provide further analysis in Section 1, including analysis on using category-level pose estimation, the number of discretized orientation bins, ablation study on rotation equivariance suspension, analysis on object rotation augmentations, and comparisons of inference time and the number of parameters. We also include additional implementation details of the backbone architecture of VoteNet in Section 2. Our code can be found in our supplementary material.

1. Additional Experimental Analysis

In this section we provide additional experiments analysis for EON.

Comparison to using category-level pose estimation. A naive method to directly achieve object-level equivariance in 3D detection might be to use category-level pose estimation to refine the bounding box orientations. To do this, we attach a head to VoteNet to predict a NOCS [7] coordinate for each box proposal, and use the estimated pose to refine box orientation. Table 1 shows that category-level pose estimation can only marginally improve the baseline VoteNet. This is because our equivariant design is not only to refine orientations. More importantly, it learns better geometric object features and distinguishes objects from backgrounds to improve proposal generations.

Effects of finer orientation discretization. In our EON we predict the object-level orientations by discretizing the yaw orientation group into N bins and perform a 4-way classification for each seed feature orbit. To explore the effect of different numbers of bins, we study the mAP performance versus different values of N on ScanNetV2 and show the results in Figure 1. We can see that the performance boost of EON-VoteNet over the baseline VoteNet reaches the peak at $N = 4$, and saturates with finer discretization. Thus, we empirically set $N = 4$ to obtain full performance gain with minimal additional computation.

Method	mAP@0.25	mAP@0.5
VoteNet	50.4	28.3
VoteNet+NOCS [7]	50.7	28.5
EON-VoteNet (ours)	**56.7**	**36.5**

Table 1. Comparison to category-level pose estimation method on ScanNetV2 dataset with Scan2CAD detection labels.

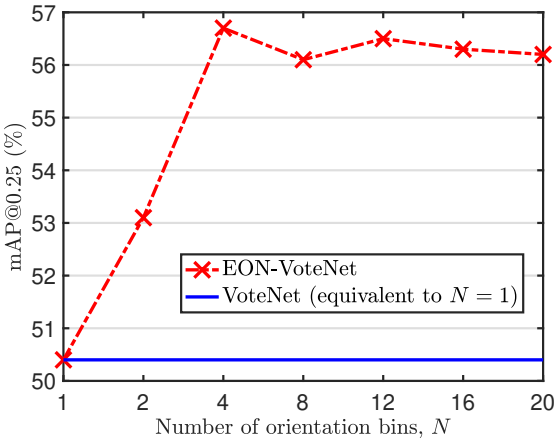
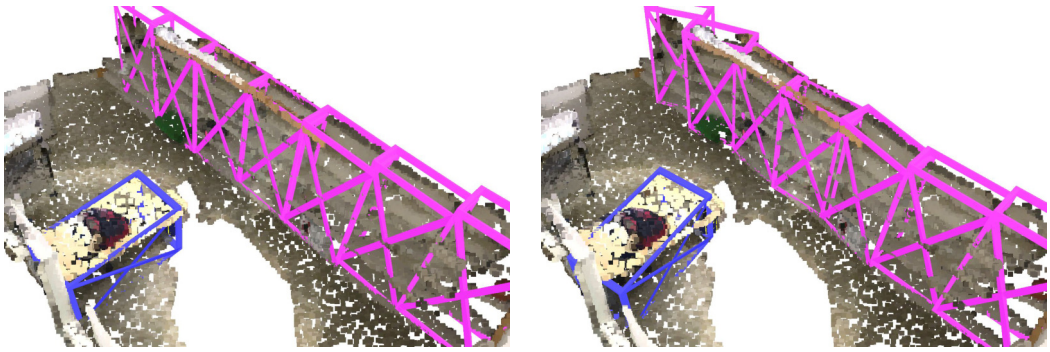


Figure 1. Performances versus different numbers of orientation bins. Results are evaluated on ScanNetV2 with Scan2CAD labels, measured by mAP@0.25. Note that EON-VoteNet is equivalent to VoteNet when $N = 1$.

Ablating rotation equivariance suspension. A core component in our model design is the rotation equivariance suspension on the region aggregation module for object-level rotation equivariance. Thus, we explore the importance of rotation equivariance suspension by replacing it with an invariant max-pooling which is typically adopted for extracting invariant features from equivariant features [1,2]. We refer to this baseline model as Invariant Object detection Networks, or ION. ION extracts the same feature orbits as EON in the seed feature extraction module. Instead of suspending the rotation equivariance, ION directly extracts invariant features from the feature orbits. We show a comparison of EON-VoteNet and ION-VoteNet in Table 2.



w/o ObjAug w/ ObjAug

Figure 2. An example of using object rotation data augmentation.

Method	Trash-bin	Display	Others	Bathtub	Chair	Cabinet	Bookshelf	Table	Sofa	Bed	mAP
ION-VoteNet	34.0	23.8	**20.0**	43.9	71.7	47.1	51.4	70.3	**69.9**	83.5	51.5
EON-VoteNet (ours)	**45.3**	**35.8**	16.4	**49.1**	**86.3**	**51.9**	**51.0**	**75.0**	68.7	**87.2**	**56.7**

Table 2. Ablation study on rotation equivariance suspension. We compare EON-VoteNet to ION-VoteNet on ScanNet V2 validation set using Scan2CAD detection labels. Performances are measure by AP_{25}.

Method	VoteNet	EON-VoteNet
Trained w/ ObjAug	49.7	53.6
Trained w/o ObjAug	**50.4**	**56.7**

Table 3. Evaluation for object rotation data augmentation (ObjAug). Tested on ScanNetV2 with Scan2CAD labels, measured by mAP@0.25.

Method	Inference time (ms)	#Parameters
VoteNet	85.3	0.96M
EON-VoteNet	92.8	1.10M

Table 4. Comparison on inference time (using a single input point cloud with 40K points as input) tested on a single TitanRTX, and numbers of parameters.

From Table 2 we see that EON-VoteNet outperforms ION-VoteNet significantly, despite that the latter uses the same equivariant computation for seed features. This comparison suggests the importance of maintaining object-level equivariance throughout the detector.

Effect of object rotation data augmentation. A natural alternative idea to approach object rotation equivariance is to use object rotation data augmentation. That is, for each oriented box in a training scene, we randomly rotate the points inside the box w.r.t. its box central gravity axis by $[-30, 30]$ degrees. We also correspondingly rotate the box orientation label and vote labels inside the box. We denote this augmentation strategy as "ObjAug". We show results of VoteNet and EON-VoteNet when using ObjAug in Table 3.

From Table 3 we can observe that ObjAug hurts both methods' performances. A possible reason is that the bounding boxes cannot perfectly enclose an object, so that they either include background segments or leave some object surface points unchanged (for example, some surface points of the desk is not rotated in Figure 2). This leads to corruptions to the point cloud data. Another possible reason is that ObjAug changes the scene layouts to an unnatural configuration (for example, all bookshelves have different orientations as shown in Figure 2), so that they deviate from common room layouts present in the test scenes.

Inference time and numbers of parameters. Finally, we report the inference time and the numbers of parameters in Table 4 for VoteNet and EON-VoteNet. All models are tested on a single Nvidia TitanRTX with batch size 1. We can see that our EON does not add significant cost in inference time and model capacity.

2. Additional Implementation Details

In our experiments, we apply our EON to VoteNet [3] and PointRCNN [5]. While both VoteNet and PointRCNN originally use PointNet++ [4] as their backbones, we replace it with KPConv [6] for VoteNet to test our EON design on different backbones. We show the detailed architecture of the KPConv-based backbone in Table 5 for VoteNet and Table 6 for EON-VoteNet.

Layer name	Output shape	Radius (meter)	Kernel dimension	Note
SA1	2048×32	0.2	15×4×32	
SA2	1024×32	0.4	15×32×32	
SA3	512×64	0.8	15×32×64	Skip to FP4
SA4	256×64	1.2	15×64×64	Skip to FP3
SA5	128×64	1.5	15×64×64	Skip to FP2
SA6	64×128	1.8	15×64×128	
FP1	128×64	1.5	15×192×64	
FP2	256×64	1.2	15×128×64	
FP3	512×64	0.8	15×128×64	
FP4	1024×64	0.4	15×96×64	

Table 5. Architecture of KPConv-based [6] backbone used for VoteNet [3]. Following PointNet++ [4] layer definition, "SA" in the layer name stands for set abstraction, and "FP" stands for feature propagation. As in PointNet++ [4], SA layers use farthest point sampling for abstraction downsampling, FP layers use 3-NN for upsampling, and every layer uses ball queries for local grouping. The format for output shape is #points × output_dimension. The format for kernel dimension is #kernels × input_dimension × output_dimension. Each layer is followed by a batchnorm and a ReLU activation.

Layer name	Output shape	Radius (meter)	Kernel dimension	Note
SA1	2048×32×4	0.2	15×4×32	
SA2	1024×32×4	0.4	15×32×32	
SA3	512×64×4	0.8	15×32×64	Skip to FP4
SA4	256×64×4	1.2	15×64×64	Skip to FP3
SA5	128×64×4	1.5	15×64×64	Skip to FP2
SA6	64×128×4	1.8	15×64×128	
FP1	128×64×4	1.5	15×192×64	
FP2	256×64×4	1.2	15×128×64	
FP3	512×64×4	0.8	15×128×64	
FP4	1024×64×4	0.4	15×96×64	

Table 6. Architecture of KPConv-based [6] equivariant backbone used for EON-VoteNet. Following PointNet++ [4] layer definition, "SA" in the layer name stands for set abstraction, and "FP" stands for feature propagation. As in PointNet++ [4], SA layers use farthest point sampling for abstraction downsampling, FP layers use 3-NN for upsampling, and every layer uses ball queries for local grouping. The format for output shape is #points × output_dimension × #orientation. The format for kernel dimension is #kernels × input_dimension × output_dimension. Each layer is followed by a batchnorm and a ReLU activation, and a 1D convolution with kernel size 3 on the orientation dimension for orientation channel communication [1].

References

[1] Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3d point cloud analysis. In *CVPR*, 2021. 1, 3

[2] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector neurons: A general framework for so(3)-equivariant networks. *ICCV*, 2021. 1

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

[4] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv:1706.02413*, 2017. 2, 3

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

[6] 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 *CVPR*, 2019. 2, 3

[7] 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*, 2019. 1