# OcTr: Octree-based Transformer for 3D Object Detection
## *Supplementary Material*

This supplementary material provides more implementation details on OcTr in Sec. A, more experiments results in Sec. B and additional visualization in Sec. C.

## A. More Implementation Details

### A.1. Detailed Implementation

The voxel size in WOD and KITTI is set as [0.1m, 0.1m, 0.1875m] and [0.05m, 0.05m, 0.125m], respectively. The convolution patch embedding module outputs the feature map with a downsampling ratio of 4 and the dimension of the feature map is set as $64 \times 8 \times 376 \times 376$ in WOD and $64 \times 8 \times 400 \times 352$ in KITTI. There are two stacked octree Transformer layers, with two Octree Transformer Blocks (OTBs) in each layer. In the first layer, the pyramid height, the attention dimension, the number of heads, the dimension of heads, and the value of top$k$ are set to 4, 64, 2, 32 and 8, respectively. In the second layer, they are set to 3, 64, 2, 32 and 8. $\tau$ in Eq. (6) is set to 1 and $\Gamma$ in Eq. (10) is 10000 in practice. During the training procedure, we adopt the Adam optimizer with a batch size of 16, and the cosine annealing learning rate scheduler with an initial value of 0.01 for the two-stage model and 0.003 for the single-stage model. Other hyper-parameters in detection heads, data augmentation and post-processing are set the same as the default values in OpenPCDet [5].

Our code is implemented based on OpenPCDet [5]. All the experiments are conducted on 4 RTX 3090 GPUs except the ones on complexity analysis shown in Table 9.

### A.2. Detailed Architecture

The detailed architecture of OcTr is demonstrated in Fig. A. The convolutional patch embedding is composed of sparse convolutions with the kernel size of $3 \times 3 \times 3$, and $4\times$ downsampling is conducted on input feature maps. A regular sparse convolution layer is applied for downsampling between OTBs, and the successive height compression operation is replaced with a pixel-wise sub-manifold sparse convolution on BEV features.

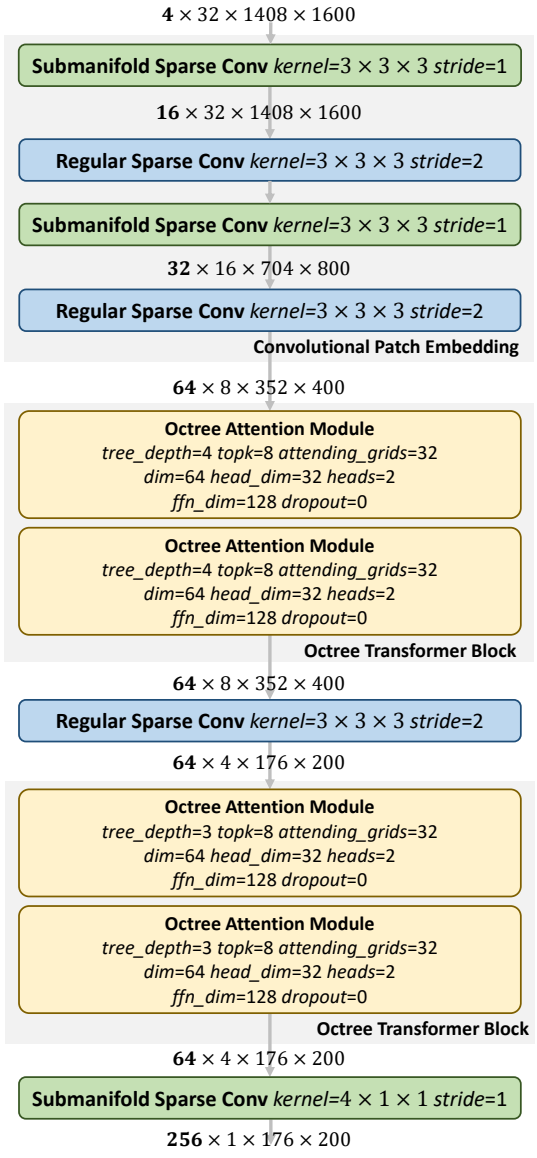


Figure A. Detailed architecture of the proposed OcTr network.

### A.3. Top$k$ Sampling

Top$k$ sampling is an important component in OcTr (in Sec. 3.3 of the main body). To generate the selected sparse

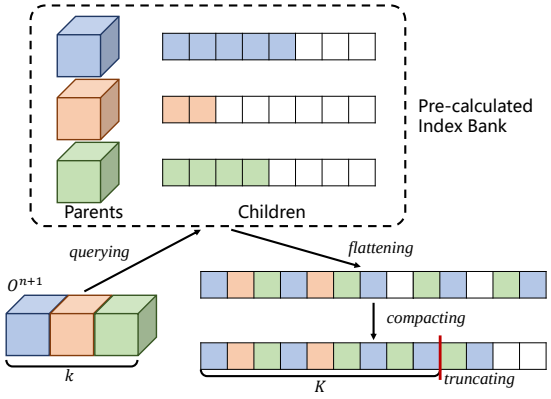


Figure B. Illustration of top$k$ sampling (the white/colored square denotes empty/non-empty grid, respectively).

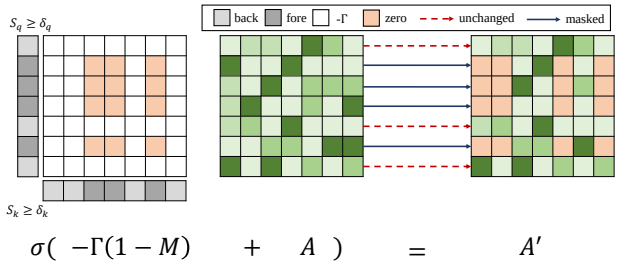


Figure C. In SAM, the attention scores of background grids maintain unchanged, while those of foreground ones are masked.

octants for subdivision, we first record the indices between the child and parent octants as a pre-calculated index bank. Fig. B depicts the entire procedure. In level $n$, by querying about top$k$ parent octants and the pre-calculated index, we densify the sampling outputs $\vec{K} \in \mathbb{R}^{B\times m_{n+1}\times 8\cdot k\times d}$ and flatten the tensors of the key/value in an $8 \cdot k \rightarrow [k, 8]$ manner. We then compact the tensor and truncate the top $K$ children, resulting in $\vec{K} \in \mathbb{R}^{B\times m_{n+1}\times K\times d}$. By using the pre-defined index bank, we broadcast the sampled tensors to align the features in layer $n$, generating a tensor with the shape of $\mathbb{R}^{B\times m_n\times K\times d}$.

According to statistics, the downsampling ratio in feature pyramid construction is fixed as 3.2, *i.e.* $m_n/m_{n+1} \approx 3.2$. Empirically, to adequately query, we set $K = 4 \times k$ in our implementation.

### A.4. Semantic Attention Mask

Following Eq. (10) in the main body, we further show the details of SAM in Fig. C. To obtain a mask for inferior foreground grids, we define a boolean tensor $\mathcal{M}_q = \mathbb{I}_{S_q\geq\delta_q}$, where $S_q$ is calculated by the mean scatter function [1]. Similarly, we have $\mathcal{M}_k = \mathbb{I}_{S_k\geq\delta_k}$ and obtain the boolean semantic mask on the attention matrices, which is measured by $\mathcal{M} = \mathcal{M}_q \cdot \mathcal{M}_k$. Though segmentation scores indicate the significance of grids, they suffer from inaccurate predictions. Considering that the mask $\mathcal{M}$ tends to suppress the attention scores of the background grids to 0 and thus deteriorate representations, we simply maintain the attention scores of the background unchanged. The hyper-parameters $\delta_q$ and $\delta_k$ are set to 0.05 and 0.2, respectively.

[1] https://pytorch-scatter.readthedocs.io/

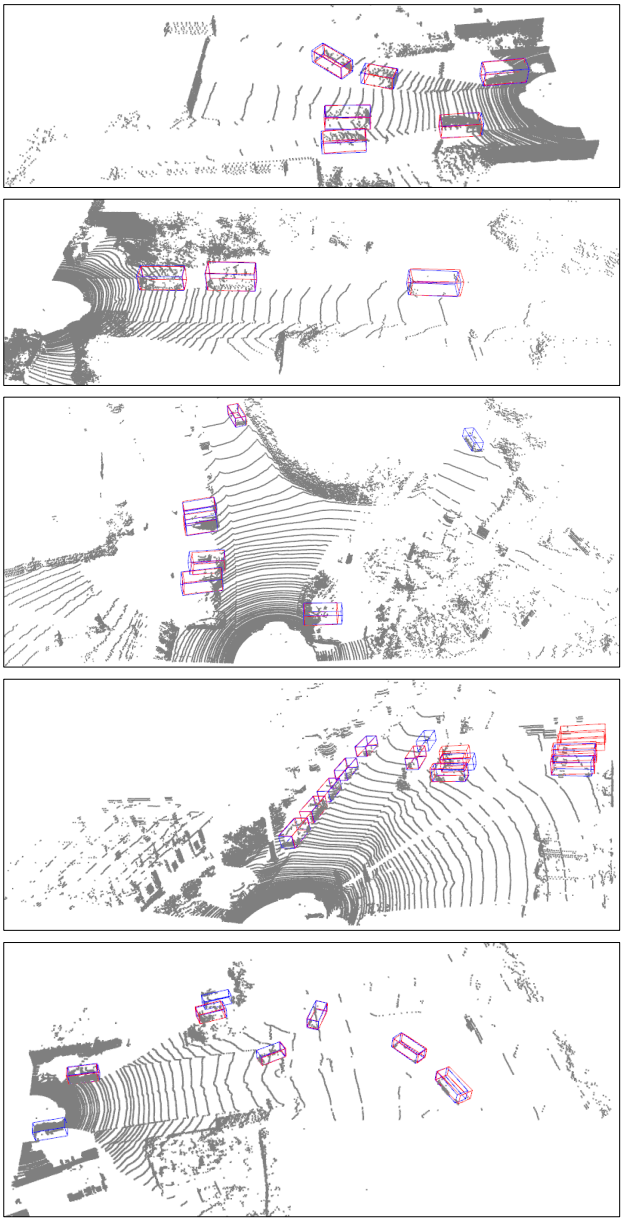

Figure D. Visualization on the KITTI *val* set. The blue/red bounding boxes indicate the predicted/ground-truth results, respectively.

| Model | Vehicle (L1) mAP/mAPH | Vehicle (L2) mAP/mAPH | Pedes. (L1) mAP/mAPH | Pedes. (L2) mAP/mAPH | Cyclist (L1) mAP/mAPH | Cyclist (L2) mAP/mAPH |
|---|---|---|---|---|---|---|
| CF (1 frame) [7] | 75.2/74.7 | 70.2/69.7 | 78.6/73.0 | 73.6/68.3 | 72.3/71.3 | 69.8/68.8 |
| CF (8 frames) [7] | 78.8/78.3 | **74.3/73.8** | 82.1/**79.3** | **77.8/75.0** | 75.2/74.4 | 73.2/72.3 |
| FSD [1] | 79.2/78.8 | 70.5/70.1 | **82.6**/77.3 | 73.9/69.1 | **77.1/76.0** | **74.4/73.3** |
| Graph-RCNN [6] | **80.8/80.3** | 72.6/72.1 | 82.4/76.6 | 74.4/69.0 | 75.3/74.2 | 72.5/71.5 |
| OcTr | 79.2/78.7 | 70.8/70.4 | 82.2/76.3 | 74.0/68.5 | 73.9/72.8 | 71.1/69.2 |

Table A. Performance on WOD *validation* with 100% training data.

| Model | Vehicle (L1) mAP/mAPH | Vehicle (L2) mAP/mAPH | Pedes. (L1) mAP/mAPH | Pedes. (L2) mAP/mAPH | Cyclist (L1) mAP/mAPH | Cyclist (L2) mAP/mAPH |
|---|---|---|---|---|---|---|
| SECOND [50] | 76.2/75.7 | 68.3/67.8 | 68.6/55.3 | 62.8/50.5 | 62.4/56.6 | 60.1/54.6 |
| Ours (SECOND) | **77.9/77.4** | **70.2/69.7** | **71.5/61.1** | **65.7/56.1** | **70.7/69.3** | **68.1/66.8** |
| PV-RCNN++ [37] | 81.6/81.2 | 73.9/73.5 | 80.4/75.0 | 74.1/69.0 | 71.9/70.8 | 69.3/68.2 |
| Ours (PV-RCNN++) | **81.7/81.4** | **74.0/73.6** | **81.2/75.2** | **75.0/69.3** | **73.0/71.8** | **70.4/69.4** |

Table B. Performance on WOD *test* with 100% training data.

## B. More Experiments Results

We add experiments with 100% training data and compare OcTr with Graph-RCNN [6], FSD [1] and CenterFormer (CF) [7]. Note that PVRCNN++ in Table 1 is the same as PVRCNN++ (center). As in Table A, Graph-RCNN and CF (8 frames) achieve higher results, but either with multi-modal or multi-frame data for prediction. When using single frames, OcTr clearly outperforms CF. As for FSD, the performance of OcTr is comparable or even better than that of FSD on vehicle and pedestrian, but is moderately lower on cyclist. However, FSD builds a strong detection head, which tends to be complementary to our OcTr backbone. We believe that OcTr can be further promoted by combining FSD.

We also conduct experiments on Waymo *test* in Table B using the representative single-stage SECOND and two-stage PVRCNN++, and the results confirm the effectiveness of our method. It should be noted that the common testing tricks, *e.g.* TTA and WBF, are not applied.

## C. More Visualization Results

We additionally visualize some detection results by using the proposed OcTr network on KITTI [2] and WOD [4] in Fig. D and Fig. E, respectively. We use the two-stage detector PVRCNN++ [3] as our baseline model and only predict cars on KITTI. As displayed, we can observe that OcTr delivers accurate localization and classification for distant and sparse samples, even in crowded scenes.

## References


[1] Lue Fan, Wang Feng, Wang Naiyan, and Zhang Zhaoxiang. Fully sparse 3d object detection. In *NIPS*, 2022. 3

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. 3

[3] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *IEEE TPAMI*, 2021. 3

[4] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2446–2454, 2020. 3

[5] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 1

[6] Honghui Yang, Liu Zili, Wu Xiaopei, Wang Wenxiao, Qian Wei, He Xiaofei, and Cai Deng. Graph r-cnn: Towards accurate 3d object detection with semantic-decorated local graph. In *ECCV*, 2022. 3

[7] Zixiang Zhou, Zhao Xiangchen, Wang Yu, Wang Panqu, and Hassan Foroosh. Centerformer: Center-based transformer for 3d object detection. In *ECCV*, 2022. 3

Figure E. Visualization on the WOD *validation* set in crowded scenes. The blue/red bounding boxes indicate the predicted/ground-truth results, respectively.