Ponder: Point Cloud Pre-training via Neural Rendering
Supplementary Material

1. Implementation Details

In this section, we give more implementation details of our **Ponder** model.

1.1. Pre-training Details

Network architecture. To process the extracted 3D feature volume, our approach utilizes a 3D U-Net. We adopt the standard implementation of 3D U-Net, which consists of four down-sampling stages with corresponding channels of 32, 64, 128, and 256, respectively. All convolution layers use a 3D kernel of size 3. To construct the neural rendering decoders, Ponder employs a five-layer MLP network as the SDF decoder and a three-layer MLP network as the RGB decoder.

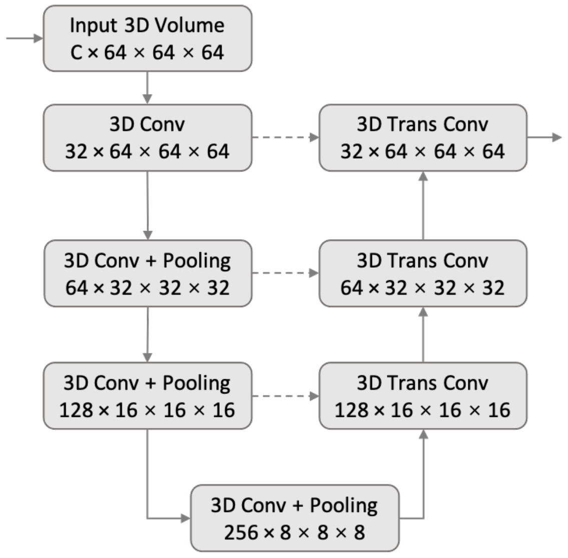

Figure 1. **3D U-Net architecture.**

3D feature volume. Given a point cloud $\mathcal{X}$, we first discretize the 3D space into a feature volume, $\mathcal{V}$, of resolution $H \times W \times D$. For each voxel center in $\mathcal{V}$, we then apply average pooling to aggregate features from surrounding points of $\mathcal{X}$. When there is no point near a voxel due to the sparsity of $\mathcal{X}$, that voxel remains empty. The point cloud $\mathcal{X}$ can be created from either single or multiple depth frames.

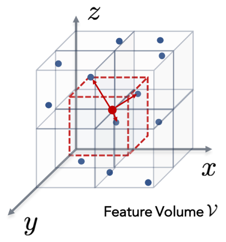

Figure 2. 3D feature volume construction.

In our experiments, we build a hierarchical feature volume $\mathcal{V}$ with a resolution of [16, 32, 64]. Building a 3D hierarchical feature volume has been wildly used for recovering detailed 3D geometry, e.g. [3, 2]. After processing the 3D feature volume with a 3D CNN, we use trilinear interpolation to get the feature of the query point $\mathbf{p}$, which is sampled along the casting ray and denoted as $\mathcal{V}(\mathbf{p})$. We use the drop-in replacement of *grid_sampler* from [10] to accelerate the training.

Ray sampling strategy. Similar to [8, 11], we sample twice for each rendering ray. First, we uniformly sample coarse points between the near bound z_n and far bound z_f. Then, we use importance sampling with the coarse probability estimation to sample fine points. Folowing [11], the coarse probability is calculated based on $\Phi_h(s)$. By this sampling strategy, our method can automatically determine sample locations and can collect more points near the surface, which makes the training process more efficient.

Back projection Here we give details of the back projection function π^{-1} to get point clouds from depth images. Let $\mathbf{K}$ be camera intrinsic parameters, $\xi = [\mathbf{R}|\mathbf{t}]$ be camera extrinsic parameters, where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation matrix. $\boldsymbol{X}_{uv}$ is the projected point location and $\boldsymbol{X}_w$ is the point location in the 3D world coordinate. Then, according to the pinhole camera model:

$$s\boldsymbol{X}_{uv} = \mathbf{K}(\mathbf{R}\boldsymbol{X}_w + \mathbf{t}), \tag{1}$$

where s is the depth value. After expanding the $\boldsymbol{X}_{uv}$ and $\boldsymbol{X}_w$:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}(\mathbf{R}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}). \tag{2}$$

Then, the 3D point location can be calculated as follows:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R}^{-1}(\mathbf{K}^{-1}s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \mathbf{t}) \tag{3}$$

The above Equation 3 is the back-projection equation π^{-1} used in this paper.

Training Time. The **Ponder** model is pre-trained with 8 NVIDIA A100 GPUs for 96 hours.

1.2. Transfer Learning Details

3D scene reconstruction. ConvONet [9] reconstructs scene geometry from the point cloud input. It follows a two-step manner, which first encodes the point cloud into a 3D feature volume or multiple feature planes, then decodes the occupancy probability for each query point. To evaluate the transfer learning capability of our point cloud encoder, we conduct an experiment where we replace the point cloud encoder of ConvONet directly with our pretrained encoder, without any additional modifications. We choose the highest performing configuration of ConvONet as the baseline setting, which uses a 3D feature volume with a resolution of 64. For the training of ConvONet, we follow the same training setting as the released code[1].

Image synthesis from point clouds. Point-NeRF [12] renders images from neural point cloud representation. It first generates neural point clouds from multi-view images, then uses point-based volume rendering to synthesize images. To transfer the learned network weight to the Point-NeRF pipeline, we 1) replace the 2D image feature backbone with a pre-trained point cloud encoder to get the neural point cloud features, 2) replace the color decoder by a pre-trained color decoder, 3) keep the other Point-NeRF module untouched. Since a large amount of point cloud is hard to be directly processed by the point cloud encoder, we down-sample the point cloud to 1%, which will decrease the rendering quality but help reduce the GPU memory requirements. We report the PSNR results of the unmasked region as the evaluation metric, which is directly adopted from the original codebase[2]. For training Point-NeRF, we follow the same setting as Point-NeRF.

2. Supplementary Experiments

2.1. Transfer Learning

Label Efficiency Training. We also do experiments to show the performance of our method with limited labeling for the downstream task. Specifically, we test the label efficiency training on the 3D object detection task for ScanNet. Following the same setting with IAE[13], we use 20%, 40%, 60%, and 80% of ground truth annotations. The results are shown in Figure 3. We show constantly improved results over training from scratch, especially when only 20% of the data is available.

Color information for downstream tasks. Different from previous works, since our pre-training model uses a colored point cloud as the input, we also use color information for the downstream tasks. Results are shown in Table 2. Using color as an additional point feature can help the VoteNet baseline achieve better performance on the SUN RGB-D dataset, but get little improvement on the ScanNet dataset. This shows that directly concatenating point positions and colors as point features shows limited robustness to application scenarios. By leveraging the proposed **Ponder** pre-training method, the network is well initialized to handle the point position and color features, and achieve better detection accuracy.

[1]https://github.com/autonomousvision/convolutional_occupancy_networks
[2]https://github.com/Xharlie/pointnerf

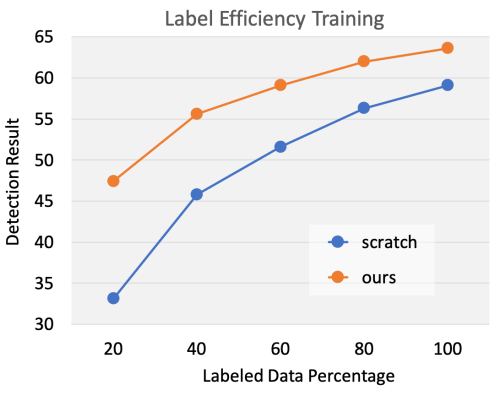

Figure 3. **Label efficiency training.** We show the 3d object detection experiment results using limited downstream data. Our pretrained model is capable of achieving better performance than training from scratch using the same percentage of data or requires fewer data to get the same detection accuracy.

Ablation study of different loss terms. The ablation study of different loss terms is shown in Tab. 1, which demonstrates the effectiveness of each loss term.

Losses	AP_{50} ↑	AP_{25} ↑
L	**41.0**	63.6
- L_c	40.9	**64.2**
- L_d	40.5	63.4
- L_e	40.9	63.3
- L_e - L_f	40.7	63.1
- L_e - L_f - L_s	40.5	63.2

Table 1. **Ablation study for loss terms** *3D detection AP_{25}* and AP_{50} on ScanNet.

More comparisons on 3D detection. More detection accuracy comparisons are given in Table 2. Even using an inferior backbone, our **Ponder** model is able to achieve similar detection accuracy with 2 in ScanNet and better accuracy in SUN RGB-D.

3D semantic segmentation with point-based approaches. Tab. 3 shows our additional experiments with the point-based approach **Ponder**+DGCNN.

Ablation study of different pre-training epochs. Tab. 4 shows that longer pre-training epochs lead to better performance in downstream tasks.

Method	Detection Model	Pre-training Type	Pre-training Data	Pre-training Epochs	ScanNet AP_{50} ↑	ScanNet AP_{25} ↑	SUN RGB-D AP_{50} ↑	SUN RGB-D AP_{25} ↑
VoteNet*	VoteNet*	-	-	-	37.6	60.0	33.3	58.4
DPCo[6]	VoteNet*	Contrast	Depth	120	**41.5**	**64.2**	35.6	59.8
IPCo[6]	VoteNet*	Contrast	Color & Depth	120	40.9	63.9	35.5	60.2
VoteNet (w color)	VoteNet	-	-	-	33.4	58.8	34.3	58.3
Ponder	VoteNet	Rendering	Depth	100	40.9	**64.2**	36.1	60.3
Ponder	VoteNet	Rendering	Color & Depth	100	41.0	63.6	**36.6**	**61.0**

Table 2. **3D object detection** *AP_{25} and AP_{50}* on ScanNet and SUN RGB-D. * means a different but stronger version of VoteNet.

Method	OA↑	mIoU↑
DGCNN	84.1	56.1
Jigsaw	84.4	56.6
OcCo	85.1	58.5
IAE	85.9	60.7
Ponder	**86.2**	**61.1**

Table 3. **3D semantic segmentation** *OA and mIoU* on S3DIS dataset `with DGCNN model`.

Epochs	AP_{50} ↑	AP_{25} ↑
20	38.7	62.0
40	39.4	62.8
60	40.0	62.7
80	40.4	63.1
100	**41.0**	**63.6**

Table 4. **Ablation study for pre-training epochs.** *3D detection AP_{25}* and *AP_{50}* on ScanNet.

2.2. More qualitative examples

As mentioned in the paper, the pre-trained **Ponder** model can be directly used for surface reconstruction and image synthesis tasks. We give more application examples in Figure 4 and Figure 5. The results show that even though the input is sparse point clouds from complex scenes, our method is able to recover high-fidelity meshes and recover realistic color and depth images.

3. Multi-Camera 3D Object Detection

To further verify the effect of utilizing rendering in self-supervised learning, we conduct exploratory experiments on the multi-camera 3D object detection task, which employs multiview images as input data.

3.1. Experimental Setup

Dataset. The nuScenes dataset [1] is a popular benchmark for autonomous driving that includes data collected from six cameras, one LiDAR, and five radars. With 1000 scenarios, the dataset is split into three sets of 700, 150, and 150 scenes for training, validation, and testing, respectively. The evaluation metrics used for 3D object detection in the nuScenes dataset incorporate the commonly used mean average precision (mAP) and a novel nuScenes detection score (NDS).

Implementation Details. For the downstream task, we adopt the latest state-of-the-art method, i.e., UVTR [7], as our baseline. Specifically, we use ResNet50-DCN [5, 4] as the image backbone, which is initialized with the pre-trained weights (i.e., the weights of ResNet-50 Caffe model) from MMDetection[3]. To construct the 3D feature volume, we first project predefined 3D voxels to multi-view images through transformation matrices. Then, the voxel features are interpolated from the image features via the projected pixel locations. The resolution of the predefined 3D volume is $[128, 128, 5]$. The model is trained with the AdamW optimizer with an initial learning rate of $2e^{-4}$ for 24 epochs.

For pre-training, our model shares a similar architecture as the baseline, except that the point cloud is additionally used to supervise the rendered depth. As our goal is to pre-train the 2D backbone, the point cloud is not used as input to construct the 3D feature volume, which is different from the process of Ponder in the main text.

3.2. Main Results

Method	mAP↑	NDS↑
UVTR[7]	28.69	35.79
Ours	**30.10** (+1.41)	**36.31** (+0.52)

Table 5. Performance comparisons on the nuScenes val set.

The Effect of Pre-training. Table 5 shows that our method could yield up to 1.41% mAP and 0.52% NDS gains compared with the baseline, demonstrating the effectiveness of our pre-training method. The consistent improvement in both indoor and outdoor scenarios validates the robustness of our approach.

Visualization. Figure 6 provides some qualitative results of the reconstructed image and depth map, which only takes the image as input during inference. Our approach has the capability to estimate the depth of small objects, such as cars at a distance. This quality in the pre-training process encodes intricate and continuous geometric representations, which can benefit many downstream tasks. In Figure 7, we present 3D detection results in camera space and BEV (Bird's Eye View) space. Our model can predict accurate bounding boxes for nearby objects and also shows the capability of detecting objects from far distances.

[3]https://github.com/open-mmlab/mmdetection

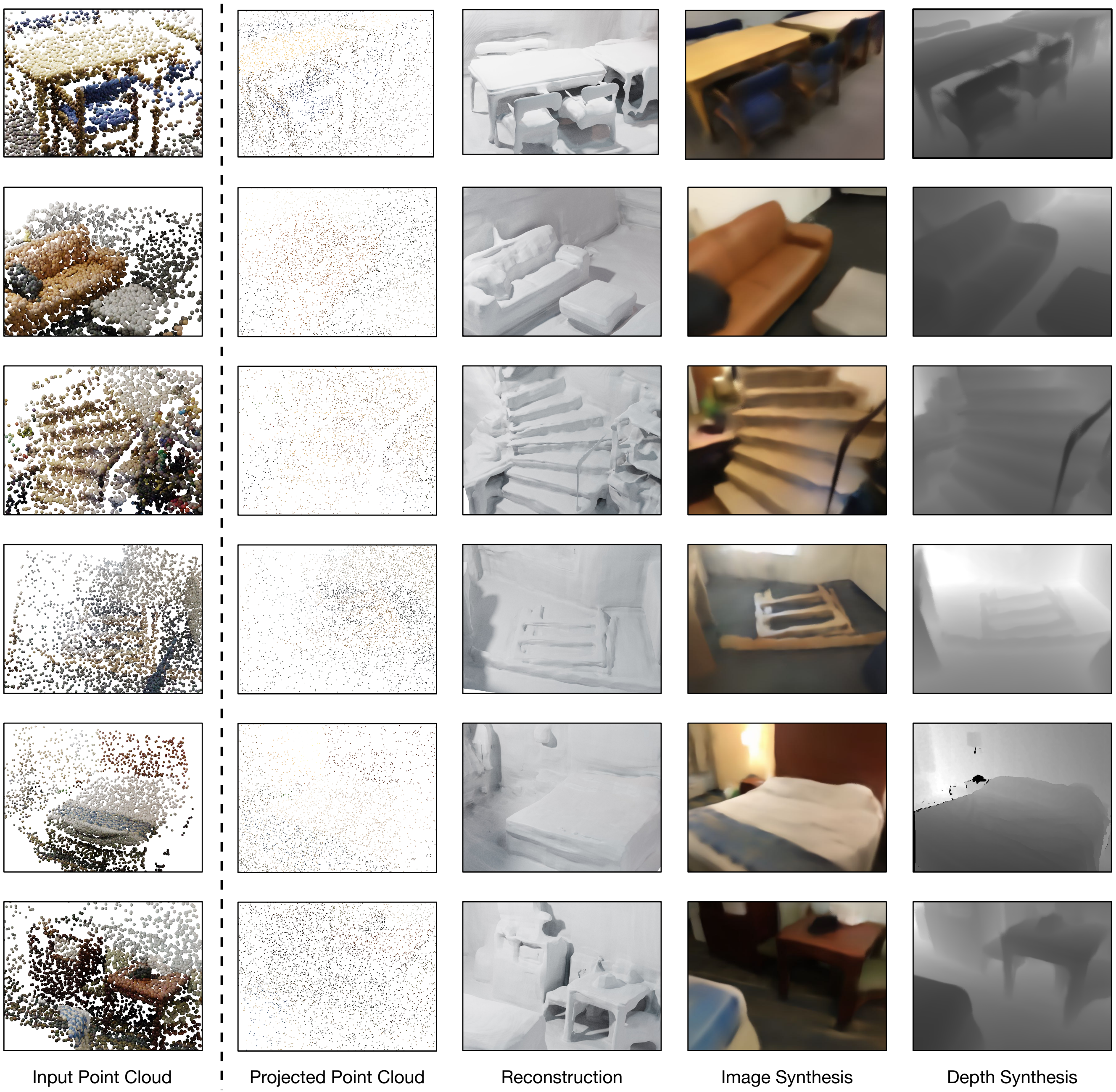

Figure 4. **More results of application examples of Ponder** on the ScanNet validation set (part 1). The input point clouds are represented by large spheres for improved clarity. The projected point clouds illustrate the actual sparsity of the point data.

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In CVPR, 2020. 3

[2] Zhang Chen, Yinda Zhang, Kyle Genova, Sean Fanello, Sofien Bouaziz, Christian Häne, Ruofei Du, Cem Keskin, Thomas Funkhouser, and Danhang Tang. Multiresolution deep implicit functions for 3d shape representation. In ICCV, 2021. 1

[3] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In CVPR, 2020. 1

[4] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In ICCV, 2017. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016. 3

[6] Lanxiao Li and Michael Heizmann. A closer look at invariances in self-supervised pre-training for 3d vision. In ECCV,

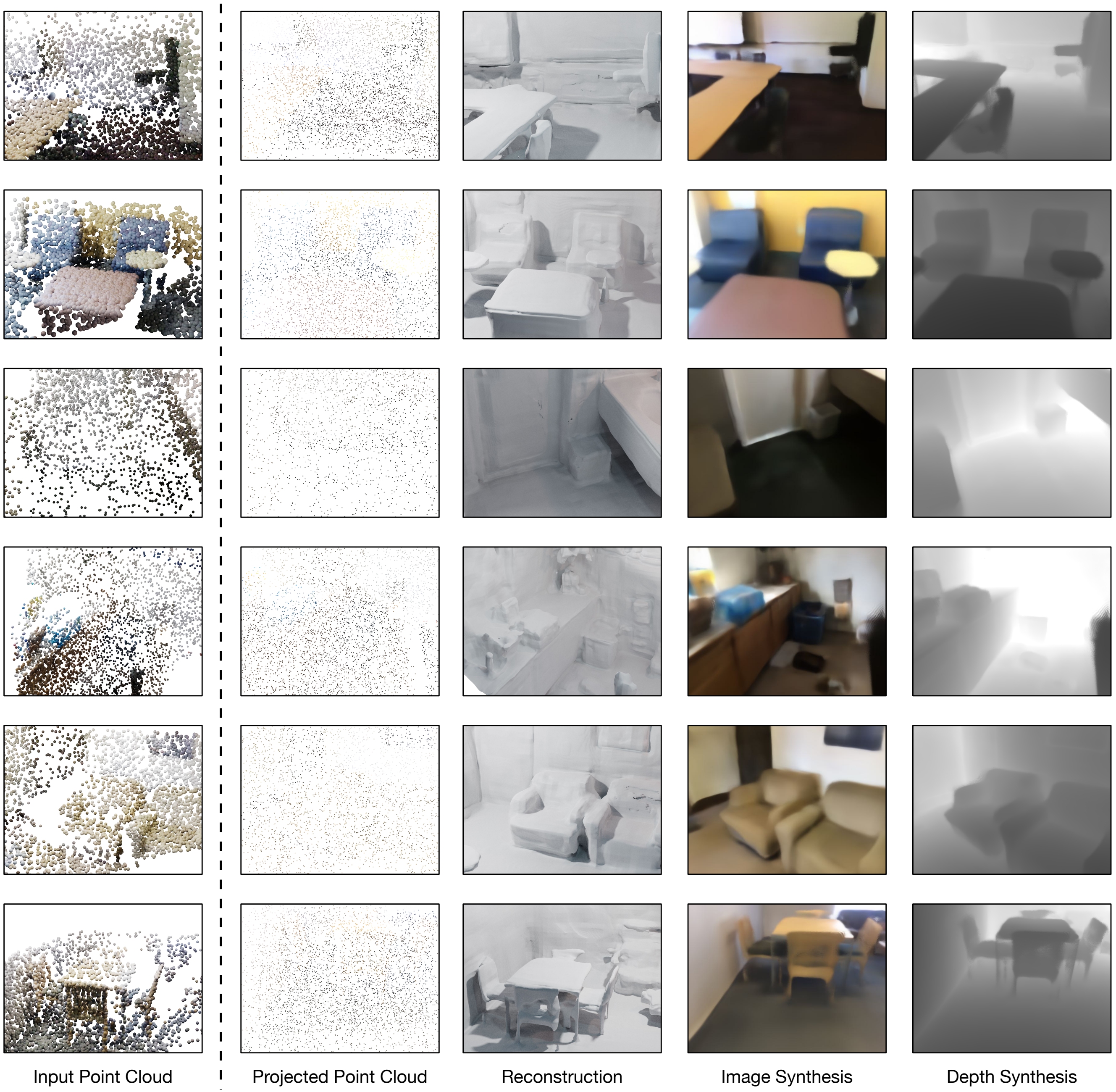

Figure 5. **More results of application examples of Ponder** on the ScanNet validation set (part 2). The input point clouds are represented by large spheres for improved clarity. The projected point clouds illustrate the actual sparsity of the point data.

2022. 3

[7] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. 2022. 3

[8] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1), 2021. 1

[9] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In ECCV. Springer, 2020. 2

[10] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. arXiv preprint arXiv:2206.14735, 2022. 1

[11] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689, 2021. 1

[12] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In CVPR, 2022. 2

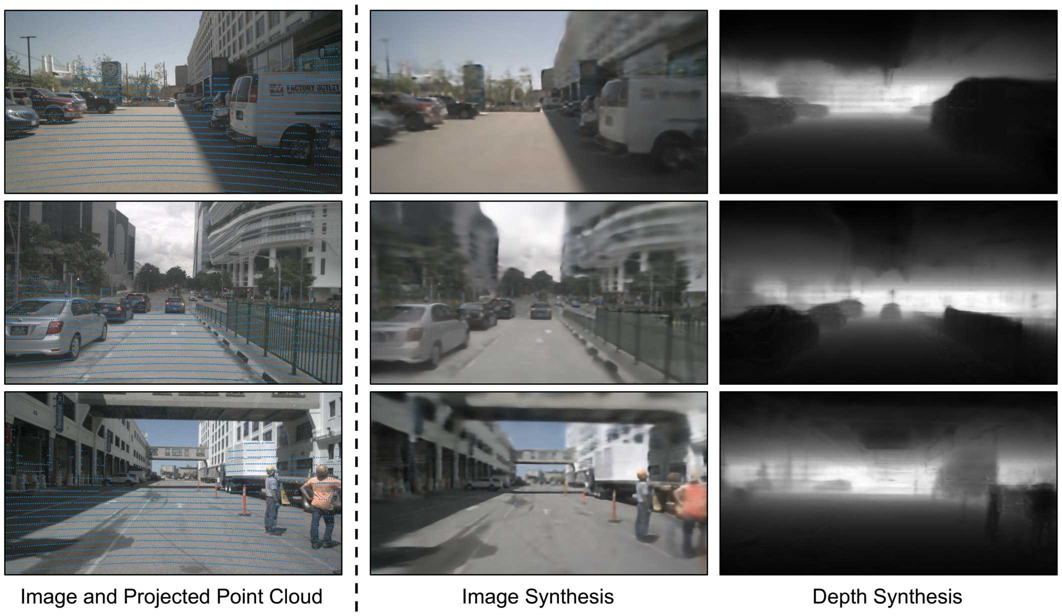

Figure 6. The predicted image and depth map on the nuScenes dataset. Left to right: image and projected point clouds, image predictions, and depth predictions.

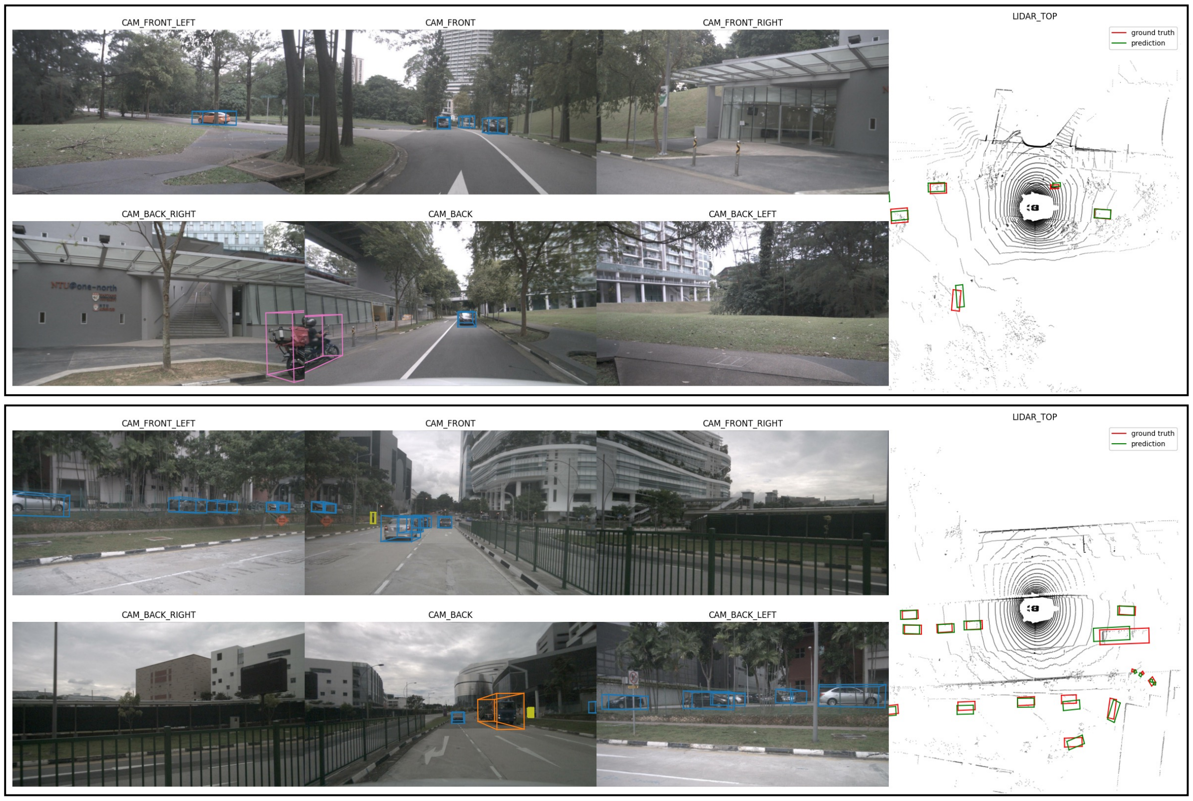

Figure 7. Qualitative results of multi-camera 3D object detection on the nuScenes dataset. We visualize the point cloud to better evaluate the quality of predicted bounding boxes.

[13] Siming Yan, Zhenpei Yang, Haoxiang Li, Li Guan, Hao Kang, Gang Hua, and Qixing Huang. Implicit autoencoder for point cloud self-supervised representation learning. arXiv preprint arXiv:2201.00785, 2022. 2