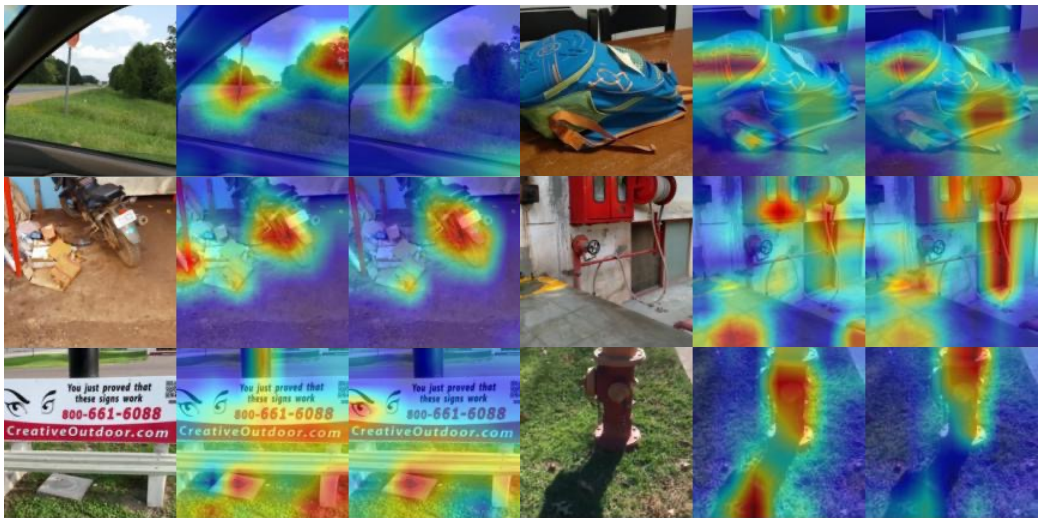
A Appendix

A.1 Effect of Background Augmentation

Reducing bias on the dataset is one of the fundamental problems in computer vision. A typical example of data bias in the object-centric dataset is that the background of images contains locational context. For instance, couch images are mostly captured in the living room, and motorcycle images are taken outdoors hence the roadway or garage appears in the background most of the time. Such a strong correlation between background information and the class labels makes the classification networks biased toward the background so that the networks make predictions by looking at the backgrounds rather than the foreground objects. Here, we demonstrate that our datasets, with additional augmentations, are more resistant to such bias. We use a popular analysis tool, Grad-CAM++ [1], to verify that the classification model trained with our dataset avoids overfitting on backgrounds. We use the official Grad-CAM++ implementation and feed test images. According to Figure a.1, the classification model trained on our dataset focuses on the object, whereas the classification model trained on the original CO3D focuses on the background for many outdoor scenes.



(a) Original (b) CAM–CO3D (c) CAM–ours (a) Original (b) CAM–CO3D (c) CAM–ours

Figure a.1: Visualization of class activation maps generated by GradCAM++. The left and right activation maps are generated from the ResNet18 trained with the original CO3D images and PeRFception–CO3D images, respectively.

We also conduct control experiments about the strength of background augmentation. In detail, we compare evaluated performance by changing the probability, *i.e,* 0%, 50%, and 100%, to apply the background augmentation for each iteration. In comparison to the results without background augmentation, the background augmentation with a probability of 50% increases the 2D classification accuracy; whereas the 100% probability of background augmentation does not improve the performance. In this ablation study, we determine the appropriate level of background augmentation assistance for improving 2D classification accuracy. Table a.1 reports the results of the control experiments.

We also visualize several examples of background augmented images on Figure a.4.

Table a.1: 2D classification accuracies (Acc@1/Acc@5) of the ResNet model trained either on CO3D or PeRFception–CO3D and evaluated either on CO3D or PeRFception–CO3D. * denotes the ImageNet[2] pretrained network. PeRF-CO3D is an abbreviation of PeRFception-CO3D. p stands for the probability to apply background augmentation.

Train Dataset	CO3D		PeRF-CO3D (p=0.0)		PeRF-CO3D (p=0.5)		PeRF-CO3D (p=1.0)	
Test Dataset	CO3D	PeRF-CO3D	CO3D	PeRF-CO3D	CO3D	PeRF-CO3D	CO3D	PeRF-CO3D
Acc@1 ($\mu \pm \sigma$)								
ResNet18	84.74 ± 0.04	80.39 ± 0.19	82.56 ± 0.24	81.43 ± 0.14	83.15 ± 0.10	82.05 ± 0.24	81.93 ± 0.10	81.31 ± 0.12
ResNet34	86.18 ± 0.05	81.62 ± 0.10	83.98 ± 0.07	82.71 ± 0.17	84.62 ± 0.14	83.61 ± 0.04	83.40 ± 0.10	82.54 ± 0.22
ResNet50	86.83 ± 0.02	82.19 ± 0.09	84.12 ± 0.28	82.80 ± 0.37	84.83 ± 0.14	83.77 ± 0.08	83.60 ± 0.11	82.73 ± 0.04
ResNet101	87.42 ± 0.04	82.86 ± 0.06	84.97 ± 0.13	83.71 ± 0.13	85.95 ± 0.22	85.11 ± 0.23	85.01 ± 0.08	83.82 ± 0.05
ResNet152	87.75 ± 0.11	83.04 ± 0.11	85.67 ± 0.31	84.35 ± 0.19	86.40 ± 0.04	85.28 ± 0.02	85.11 ± 0.09	84.06 ± 0.09
ResNext50	86.84 ± 0.29	81.99 ± 0.14	84.21 ± 0.08	83.04 ± 0.04	85.25 ± 0.16	84.32 ± 0.18	84.06 ± 0.38	83.23 ± 0.28
ResNext101	87.73 ± 0.03	82.90 ± 0.07	85.22 ± 0.09	84.17 ± 0.15	86.37 ± 0.10	85.48 ± 0.06	85.46 ± 0.16	84.46 ± 0.11
WideResNet50	87.32 ± 0.05	82.45 ± 0.11	84.68 ± 0.15	83.64 ± 0.27	85.58 ± 0.10	84.68 ± 0.02	84.35 ± 0.12	83.46 ± 0.24
WideResNet101	87.80 ± 0.12	83.04 ± 0.10	85.36 ± 0.26	84.15 ± 0.19	86.32 ± 0.21	85.30 ± 0.11	85.45 ± 0.13	84.23 ± 0.23
ResNet18*	87.75 ± 0.18	82.70 ± 0.15	85.28 ± 0.14	84.09 ± 0.20	85.97 ± 0.16	84.97 ± 0.13	84.62 ± 0.02	83.62 ± 0.13
ResNet34*	88.83 ± 0.06	84.01 ± 0.12	86.60 ± 0.27	85.43 ± 0.22	87.19 ± 0.31	86.25 ± 0.19	86.09 ± 0.11	85.01 ± 0.09
ResNet50*	89.51 ± 0.21	84.76 ± 0.04	87.49 ± 0.11	86.60 ± 0.08	88.12 ± 0.08	87.30 ± 0.08	87.09 ± 0.12	86.25 ± 0.17
ResNet101*	90.21 ± 0.09	85.60 ± 0.16	88.39 ± 0.22	87.46 ± 0.17	89.00 ± 0.07	88.32 ± 0.13	88.28 ± 0.07	87.26 ± 0.01
ResNet152*	90.60 ± 0.10	86.26 ± 0.10	89.17 ± 0.15	88.19 ± 0.03	89.52 ± 0.20	88.73 ± 0.15	88.63 ± 0.20	87.59 ± 0.13
ResNext50*	89.21 ± 0.14	83.90 ± 0.28	87.28 ± 0.06	86.28 ± 0.14	87.82 ± 0.19	87.30 ± 0.06	86.81 ± 0.27	86.09 ± 0.25
ResNext101*	90.52 ± 0.07	85.91 ± 0.13	88.51 ± 0.19	87.82 ± 0.06	89.17 ± 0.07	88.51 ± 0.16	88.57 ± 0.02	87.60 ± 0.18
WideResNet50*	89.78 ± 0.06	85.13 ± 0.33	87.80 ± 0.18	86.88 ± 0.06	88.23 ± 0.10	87.75 ± 0.25	87.12 ± 0.11	86.50 ± 0.08
WideResNet101*	90.45 ± 0.10	85.97 ± 0.09	88.53 ± 0.04	87.59 ± 0.12	89.22 ± 0.13	88.39 ± 0.07	88.10 ± 0.11	87.20 ± 0.10
Acc@5 ($\mu \pm \sigma$)								
ResNet18	96.24 ± 0.17	94.19 ± 0.16	95.55 ± 0.18	95.00 ± 0.15	95.70 ± 0.11	95.37 ± 0.05	95.30 ± 0.08	94.85 ± 0.09
ResNet34	96.68 ± 0.11	94.62 ± 0.11	95.88 ± 0.07	95.37 ± 0.10	96.23 ± 0.03	95.89 ± 0.06	95.74 ± 0.11	95.29 ± 0.07
ResNet50	96.90 ± 0.07	94.67 ± 0.14	96.10 ± 0.02	95.51 ± 0.13	96.41 ± 0.11	95.99 ± 0.08	96.01 ± 0.07	95.48 ± 0.04
ResNet101	97.04 ± 0.03	94.78 ± 0.05	96.23 ± 0.07	95.58 ± 0.09	96.76 ± 0.10	96.32 ± 0.12	96.39 ± 0.03	95.74 ± 0.14
ResNet152	97.14 ± 0.04	94.94 ± 0.09	96.32 ± 0.09	95.73 ± 0.05	96.86 ± 0.06	96.39 ± 0.06	96.46 ± 0.07	95.92 ± 0.09
ResNext50	96.75 ± 0.12	94.54 ± 0.10	95.95 ± 0.15	95.32 ± 0.13	96.49 ± 0.10	95.92 ± 0.16	96.08 ± 0.12	95.47 ± 0.15
ResNext101	96.98 ± 0.07	94.71 ± 0.12	96.09 ± 0.11	95.54 ± 0.07	96.71 ± 0.01	96.26 ± 0.03	96.49 ± 0.04	95.80 ± 0.03
WideResNet50	96.84 ± 0.13	94.60 ± 0.10	96.05 ± 0.04	95.52 ± 0.05	96.44 ± 0.11	96.03 ± 0.03	96.10 ± 0.16	95.55 ± 0.01
WideResNet101	97.05 ± 0.01	94.91 ± 0.08	96.26 ± 0.07	95.70 ± 0.08	96.86 ± 0.11	96.31 ± 0.10	96.49 ± 0.06	95.89 ± 0.01
ResNet18*	97.13 ± 0.10	95.04 ± 0.16	96.39 ± 0.08	95.89 ± 0.03	96.73 ± 0.04	96.24 ± 0.09	96.19 ± 0.08	95.67 ± 0.10
ResNet34*	97.39 ± 0.14	95.28 ± 0.15	96.67 ± 0.08	96.18 ± 0.04	97.00 ± 0.13	96.50 ± 0.09	96.61 ± 0.05	96.02 ± 0.03
ResNet50*	97.60 ± 0.04	95.46 ± 0.02	96.84 ± 0.04	96.39 ± 0.07	97.11 ± 0.05	96.69 ± 0.08	96.95 ± 0.06	96.37 ± 0.13
ResNet101*	97.90 ± 0.03	95.86 ± 0.05	97.06 ± 0.19	96.74 ± 0.11	97.48 ± 0.02	97.13 ± 0.05	97.16 ± 0.04	96.57 ± 0.03
ResNet152*	97.97 ± 0.04	95.97 ± 0.04	97.32 ± 0.17	96.87 ± 0.13	97.66 ± 0.07	97.24 ± 0.08	97.22 ± 0.17	96.69 ± 0.09
ResNext50*	97.33 ± 0.09	94.99 ± 0.11	96.54 ± 0.09	96.04 ± 0.02	97.08 ± 0.14	96.66 ± 0.07	96.57 ± 0.12	96.03 ± 0.04
ResNext101*	97.67 ± 0.03	95.51 ± 0.06	96.90 ± 0.02	96.61 ± 0.03	97.32 ± 0.05	96.93 ± 0.07	97.07 ± 0.04	96.52 ± 0.05
WideResNet50*	97.60 ± 0.07	95.53 ± 0.17	96.75 ± 0.10	96.39 ± 0.08	97.13 ± 0.02	96.84 ± 0.09	96.72 ± 0.07	96.22 ± 0.05
WideResNet101*	97.76 ± 0.07	95.67 ± 0.04	97.01 ± 0.05	96.62 ± 0.07	97.48 ± 0.07	97.10 ± 0.06	96.93 ± 0.09	96.48 ± 0.06

A.2 Implementation Details

Here, we describe the thorough details of the data generation with Plenoxels [3] and the data augmentation methods proposed in our main paper. We also provide the architectural details of 3D semantic segmentation models trained on PeRFception-ScanNet dataset.

A.2.1 Data Generation

We use the official implementation of Plenoxels [3], which is implemented in PyTorch with custom CUDA kernels. With a single RTX 3090 GPU, it takes up to 30 minutes per scene.

PeRFception-CO3D. Plenoxels uses a different learning rate for each parameter. Following the default configuration for rendering Tanks and Temples [4] dataset, we use the learning rate 30 for density values and 10^{-2} for spherical harmonic coefficients. We skip foreground rendering for 1,000 steps at the beginning of training for stable training. For total variation losses, we set the weight λ as 5.0×10^{-5} for foreground densities, 5.0×10^{-3} for foreground spherical harmonics, 1.0×10^{-3} for background colors, and 1.0×10^{-3} for background densities. The background brightness is set to 0.5. We set the lambda value to 10^{-5} for the beta loss and 10^{-10} for the sparsity loss, which is 10 times larger than the default configuration. We use the 9-dimensional spherical harmonic coefficients

for each RGB channel. We prune voxels whose density values are below 1.28 during the upsampling step.

PeRFception-ScanNet. As we described in Section 4.2 in our main paper, we initialize the voxel grid with point clouds that are obtained by TSDF integration for smoother surface geometry. The unprojected depth maps are aggregated into TSDF volume using the ground truth camera parameters. Since the ScanNet depth maps often contain noisy observations, the resulting point cloud is contaminated with outlier points. To filter out the outliers, we discretize the point cloud into coarse voxels with discretization size of 5cm and perform connected component analysis to detect disconnected voxels which are likely to be outliers. The refined point cloud is then utilized as the initialization for Plenoxels. The voxel grid is initialized with 256^3 resolution and trained for 51,200 iterations. All the other hyperparameters are same with CO3D but removed background rendering since all the points of ScanNet datasets should be considered as foregrounds.

The input images are resized into 640*480 resolution. For all scenes, we exclude image frames that are severely affected by motion blurs. Detecting blurriness was done automatically by OpenCV blur detection algorithm and we filter out the 150 most blurry frames at most. We consider a scene defective when the number of frames is too low after filtering blur images.

A.2.2 Random Pose Selection

Plenoxels show great rendering quality when the camera pose is close to the camera poses in the train set. Unfortunately, it fails to reliably render frames when attempting to render extremely unobserved parts, which are significantly out of train-view coverage. Thus, selecting appropriate poses is crucial for photorealistic image generation. We propose an operation that selects an intermediate pose from two input poses. The operator $\mathrm{AA2R}$ converts the axis-angle representation, i.e., (v, θ) where v stands for the axis and θ stands for the angle, to the rotation matrix. $\mathrm{R2AA}$ is exactly the inverse operation of $\mathrm{AA2R}$. We compute the distance between the selected poses with distance functions D_R and D_t to choose a pair of poses that are close enough. We use the rotation threshold $\theta = \frac{1}{24}\pi$ and the translation threshold 0.5.

$$
\begin{aligned}
\mathrm{ReduceAngle}(\mathbf{R}, s) &= \mathrm{AA2R}(\mathrm{R2AA}(\mathbf{R})_v, s \cdot \mathrm{R2AA}(\mathbf{R})_\theta) \\
\mathrm{IntermediateRot}(\mathbf{R}_1, \mathbf{R}_2, s) &= \mathrm{ReduceAngle}(\mathbf{R}_2\mathbf{R}_1^{-1}, s)\mathbf{R}_1 \\
D_R(\mathbf{R}_1, \mathbf{R}_2) &= \arccos\left(\mathrm{Tr}(\mathbf{R}_2^{-1}\mathbf{R}_1)/2 - 1\right) \\
D_t(\mathbf{t}_1, \mathbf{t}_2) &= ||\mathbf{t}_2 - \mathbf{t}_1||_2^2
\end{aligned}
$$

The main concept of the Algo 1 is to select an intermediate camera pose between randomly selected camera poses in the train set. For intrinsic parameters and image shape, we randomly select intrinsic matrix among intrinsic matrices in train set and use the corresponding image shape.

A.2.3 Architectural Details

We train three variants of 3D ResUNets implemented with sparse convolutional layers [5] for semantic segmentation on PeRFception-ScanNet dataset. The layer-wise architectural details are depicted in Table a.2.

A.3 Analysis on Rate-Distortion Trade-off

We compare the reconstruction qualities by varying the resolution: 64, 128, 256, and 384. We follow the setup from our paper and measure the average memory footprint and error metrics (PSNR, SSIM, LPIPS) per scene on the randomly selected subset of our PeRFception-CO3D dataset. We report the evaluated scores in Table a.3. The model with 128 resolution has shown the best trade-off between memory and reconstruction quality.

We also explore several quantization methods and their optimal compression bit. We report the evaluated scores in Table a.4. "ours (**SH**)" applies our quantization on spherical harmonics coefficients

Algorithm 1 Random Pose Generation

Input
$P = \{\{\mathbf{R}_i, \mathbf{t}_i\}\}_{i=1,2,\cdots N}$: The set of poses in train split
θ: The rotation distance threshold
d: The translation distance threshold
Output
$\mathbf{p}_{\text{out}} \in \mathbb{R}^{4\times4}$: The output pose

s: A random value from $[0, 1]$
$j, k = \text{RandomIndex}(N), \text{RandomIndex}(N)$
while $D_R(\mathbf{R}_j, \mathbf{R}_k) \geq \theta$ and $D_t(\mathbf{t}_j, \mathbf{t}_k) \geq d$ **do**
  $j, k = \text{RandomIndex}(N), \text{RandomIndex}(N)$
end while
$\mathbf{R}_{jk} = \text{IntermediateRot}(\mathbf{R}_j, \mathbf{R}_k, s)$
$\mathbf{p}_{\text{out}}[:3, :3] = \mathbf{R}_{jk}$
$\mathbf{p}_{\text{out}}[:3, 3] = s\mathbf{t}_k + (1-s)\mathbf{t}_j$
$\mathbf{p}_{\text{out}}[3, 3] = 1$
return $\mathbf{p}_{\text{out}}$

Table a.2: Architectures of Res16UNet variants for semantic segmentation on PeRFception-ScanNet. We denote a convolution layer with its *kernel size*, *output channel size*, and *convolution stride size*. All convolution layers except for the last layer have a Batch Normalization and a ReLU layer after them. The layers with the tag "conv_tr" indicates the transposed convolution layers. We use a square bracket to denote a residual block, with the number of blocks stacked.

layer name	Res16UNet14A	Res16UNet18A	Res16UNet34C
init	3^3, 32, stride 1		
conv1	2^3, 32, stride 2		
block1	$\begin{bmatrix}3^3, 32, \text{stride } 1\\3^3, 32, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 32, \text{stride } 1\\3^3, 32, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 32, \text{stride } 1\\3^3, 32, \text{stride } 1\end{bmatrix} \times 2$
conv2	2^3, 32, stride 2		
block2	$\begin{bmatrix}3^3, 64, \text{stride } 1\\3^3, 64, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 64, \text{stride } 1\\3^3, 64, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 64, \text{stride } 1\\3^3, 64, \text{stride } 1\end{bmatrix} \times 3$
conv3	2^3, 64, stride 2		
block3	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 4$
conv4	2^2, 128, stride 2		
block4	$\begin{bmatrix}3^3, 256, \text{stride } 1\\3^3, 256, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 256, \text{stride } 1\\3^3, 256, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 256, \text{stride } 1\\3^3, 256, \text{stride } 1\end{bmatrix} \times 6$
conv4_tr	2^3, 128, stride 2		2^3, 256, stride 2
block5	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 256, \text{stride } 1\\3^3, 256, \text{stride } 1\end{bmatrix} \times 2$
conv5_tr	2^2, 128, stride 2		
block6	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 128, \text{stride } 1\\3^3, 128, \text{stride } 1\end{bmatrix} \times 2$
conv6_tr	2^3, 96, stride 2		
block7	$\begin{bmatrix}3^3, 96, \text{stride } 1\\3^3, 96, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 96, \text{stride } 1\\3^3, 96, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 96, \text{stride } 1\\3^3, 96, \text{stride } 1\end{bmatrix} \times 2$
conv7_tr	2^3, 96, stride 2		
block8	$\begin{bmatrix}3^3, 96, \text{stride } 1\\3^3, 96, \text{stride } 1\end{bmatrix} \times 1$	$\begin{bmatrix}3^3, 96, \text{stride } 1\\3^3, 96, \text{stride } 1\end{bmatrix} \times 2$	$\begin{bmatrix}3^3, 96, \text{stride } 1\\3^3, 96, \text{stride } 1\end{bmatrix} \times 2$
final	1^3, 20, stride 1		

Table a.3: Reconstruction quality and required memory for different voxel resolutions.

Resolution	Memory(MB)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
64	38.0	26.56	0.7464	0.4827
128	47.2	29.39	0.8081	0.4017
256	63.2	30.81	0.8551	0.3353
384	161.2	31.03	0.8619	0.3202

only; "ours (**SH+D**)" applies our quantization to both spherical harmonics coefficients and densities. "clip" clips the feature values to a heuristically searched interval. In detail, we have clipped values that are statistically found from 5 scenes. For spherical harmonics, we clipped values to [-0.1, 0.1] since 90% of values are located inside the selected interval. For density values, we clipped to [0, 1000] to remove negative densities. We additionally seek the optimal bit for each compression method. Original quantization method with 8 bit compression, i.e., ours(SH)-8bit, has shown the best performance and has reasonable memory requirement.

Table a.4: Analysis on the memory usage and reconstruction quality of different quantization methods and their quantization bits.

Quantization Method	Bit	Memory(MB)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
ours (**SH**)	16	114.5	30.65	0.8548	0.3363
ours (**SH**)	8	63.2	30.81	0.8551	0.3353
ours (**SH**)	4	37.4	27.48	0.7932	0.4512
ours (**SH**)	2	24.8	14.69	0.5325	0.6797
ours (**SH + D**)	8	60.8	30.53	0.8546	0.3369
ours (**SH + D**)	4	34.8	23.07	0.7238	0.5166
ours (**SH + D**)	2	21.8	14.65	0.5280	0.9981
clip	8	62.6	17.78	0.7227	0.4879
clip	4	37.4	17.72	0.6575	0.5502
clip	2	24.4	16.66	0.6476	0.5881

We additionally explore the effect of **progressive scaling**, i.e. upsampling and pruning step of Plenoxel [3] to progressively increase the resolution on memory footprint and rendering quality. We consider two progressive scaling methods: **weight-based scaling** and **density-based scaling**. According to Plenoxels [3], **weight-based scaling** applies threshold to the maximum weight $T_i(1 - exp(1 - \sigma_i\delta_i))$ of each voxel over all training rays, whereas **density-based scaling** directly prunes voxels by their density values. For more details, please refer to the supplementary materials of Plenoxels. For each method, we change the threshold of each scaling method to find a good adjustment between memory usage and rendering quality. We report the evaluated scores in Table a.5. We select the "density-based scaling" method as our progressive scaling method with a sigma threshold 20, which has shown the best trade-off between memory footprint and rendering quality.

Table a.5: Analysis on the progressive scaling methods with varying the pruning threshold.

Progressive Scaling Method	Threshold	Memory(MB)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Density	5	69.0	30.66	0.8554	0.3352
Density	10	66.4	30.79	0.8550	0.3354
Density (ours)	20	63.2	30.81	0.8551	0.3353
Density	100	49.8	30.47	0.8547	0.3537
Weight (Plenoxels - default)	1.28	75.8	30.81	0.8557	0.3333
Weight	2.56	73.0	30.71	0.8560	0.3335

Table a.6: PSNR of PeRFception-ScanNet dataset with varying resolutions. PCTL stands for *percentile*.

Reso	Mem (GB)	Avg. PSNR (dB)	50th PCTL	75th PCTL	90th PCTL	95th PCTL
128	28.7	23.20 $\pm$3.69	23.59	26.14	27.94	29.12
256	43.8	23.01 $\pm$3.96	23.38	26.10	28.02	29.20
512	113.8	22.94 $\pm$4.26	23.22	26.35	28.34	29.49

We also analyze the *resolution vs. rendering quality* trade-off on our PeRFception-ScanNet dataset. To measure the trade-off rates, we train Plenoxels with lower (128) and higher (512) resolutions than the default configuration (256) on a randomly selected subset of ScanNet scenes.

As reported in Table a.6 and Figure a.2, there is no direct correlation between the resolution and the average rendering quality on ScanNet reconstruction. This could be attributed to various non-trivial factors. We visualized the histogram of PSNR distribution on the ScanNet dataset in Figure a.2. The X-axis represents the PSNR score, and Y-axis represents the percentage of scenes. Note that the PSNR distribution of the higher resolution Plenoxel reconstructions exhibits fat-tailed distribution, whereas the lower resolution reconstructions show the long-tailed distribution. We speculate that this is due to the fact that higher resolution reconstruction results in each voxel learning spherical harmonics parameters from fewer rays. Thus, errors in camera parameters or motion blur would result in larger errors for smaller voxels as parameters are learned from fewer rays. Thus, the rendering quality increases for the scenes with accurate camera poses and less motion blur, while it decreases for noisy scenes. We conjecture that higher resolution reconstruction would yield better performance if we have high-resolution images with accurate camera poses.

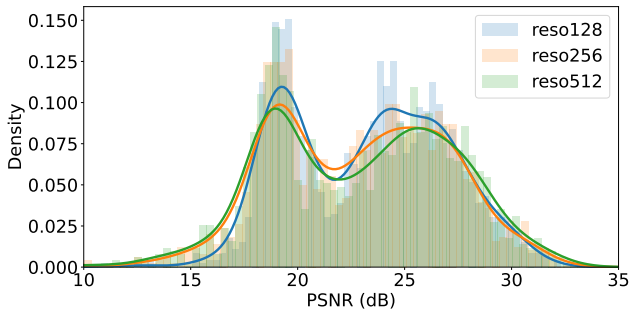


Figure a.2: Distributions of PSNR values of PeRFception-ScanNet dataset with varying resolutions.

A.4 Dataset Statistics

PeRFception-CO3D dataset includes 18,618 object-centric scenes that cover all of the original CO3D dataset scenes, only except those with incorrect camera poses. It contains a fair quantity of 51 different object-class labeled scenes; see Figure a.3.

A.5 Camera Manipulation

Camera manipulation has been one of the inaccessible augmentation techniques on conventional image datasets. In contrast, our implicit data can be rendered from many continuous viewpoints and with arbitrary camera parameters (*e.g.*, intrinsics, extrinsics, and radial distortions), allowing us to generate images with non-standard but realistic camera-level augmentations. We have not adopted this augmentation skill while training 2D classification networks since most images do not have distortions. We perform camera intrinsics augmentation and camera distortion augmentation and Figure a.4 visualizes several examples of such camera manipulations.

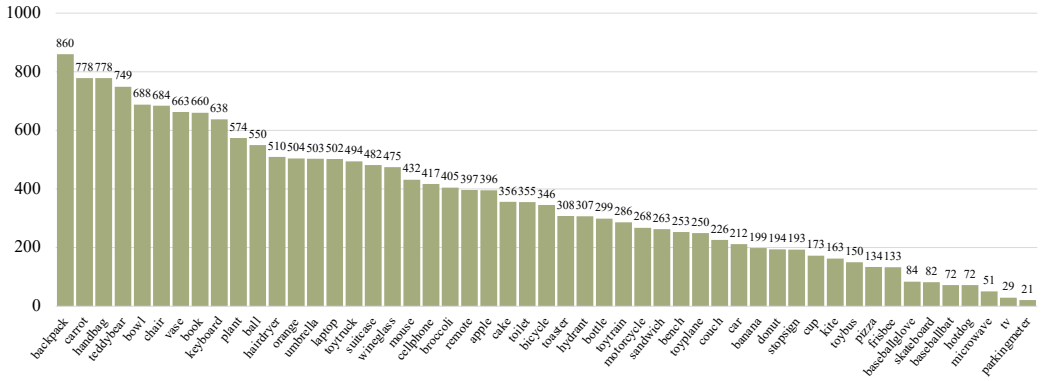


Figure a.3: Statistics of PeRFception-CO3D dataset scene per object-class. The y-axis visualizes the number of scenes for each class.

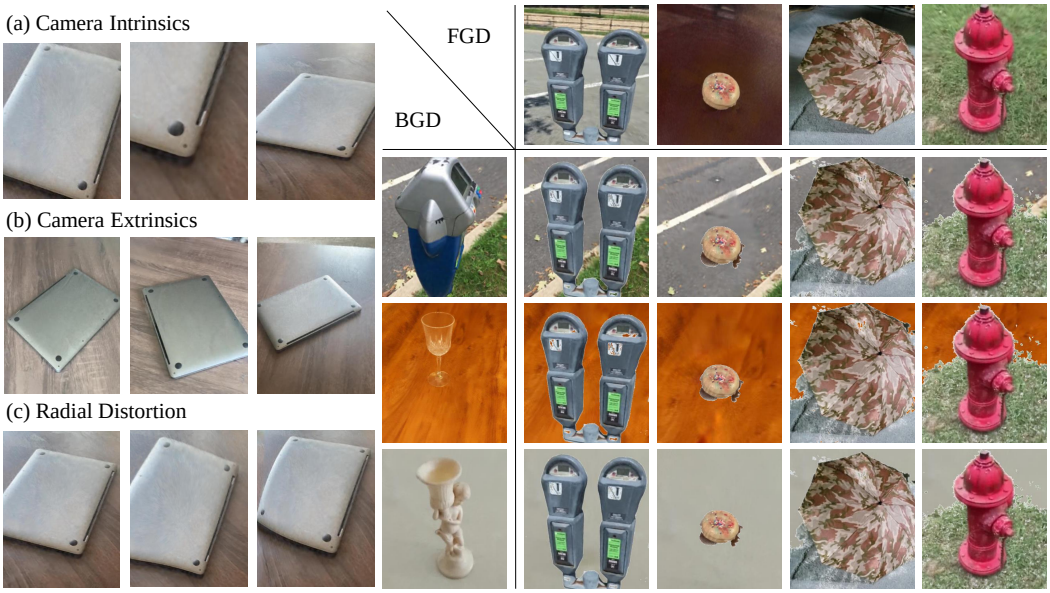


Figure a.4: Examples of camera manipulation and background augmentations. FGD and BGD are source images for foreground and background respectively.

A.6 Quantitative Results

In this section, we provide additional quantitative results that are not included in the main paper.

Classwise Rendering Quality. We provide classwise PSNR, SSIM, and LPIPS scores of PeRFception-CO3D dataset in Table a.7.

2D and 3D Perception Tasks. Table a.1 reports the results of the 2D classification experiments. In Table a.8, we report the quantitative classification results on PeRFception-CO3D dataset. In Table a.11, we report the scenewise semantic segmentation results on PeRFception-ScanNet dataset. For all experiments, we perform three experiments with different random seed values and report the mean and standard deviation of evaluation metrics.

2D Classification Anlaysis. We conduct additional 2D classification experiments using the pretrained Vision Transformer (ViT) [6] models to show that our PeRFception-CO3D dataset improves the classification performance. Similar to our ResNet 2D classification experiments, we train ViT models on

either CO3D images or PeRFception renderings and test on the CO3D test split and the PeRFception-CO3D test split. As reported in the Table a.9, the ViT models trained on the PeRFception-CO3D train set achieve higher classification accuracy than those trained on the CO3D train set. We conjecture that the domain gap for ViT models is much smaller, and the data augmentations on PeRFception made the ViT models to be more robust in real images.

	apple	backp	ball	banana	bbb	bbg	bench	bicy	book	bottle	bowl	brocc	cake
PSNR	29.56	28.99	29.06	30.62	29.21	30.03	27.01	27.67	28.88	29,85	31.83	29.67	29.02
SSIM	0.8767	0.8702	0.8661	0.8951	0.8691	0.8898	0.7883	0.8246	0.8746	0.8671	0.8717	0.8584	0.8695
LPIPS	0.3263	0.3349	0.3614	0.2591	0.3566	0.3347	0.3471	0.3416	0.3453	0.3110	0.3135	0.3479	0.3452

	car	carrot	cellp	chair	couch	cup	donut	frisbee	hairdryer	handbag	hotdog	hydrant	kbd
PSNR	23.63	28.78	29.33	28.32	29.31	28.83	28.78	30.06	29.12	28.39	28.95	26.12	29.74
SSIM	0.7259	0.8573	0.8603	0.8524	0.8727	0.8628	0.8695	0.8720	0.8706	0.8622	0.8738	0.7428	0.8789
LPIPS	0.4003	0.3609	0.3511	0.3475	0.3358	0.3602	0.3484	0.3386	0.3351	0.3367	0.3639	0.3416	0.3301

	kite	laptop	micro	motor	mouse	orange	park	pizza	plant	remote	sandwc	skb	stop
PSNR	29.04	27.99	27.75	25.61	29.04	28.55	24.69	29.00	28.63	29.23	29.00	28.57	23.48
SSIM	0.8693	0.8535	0.8506	0.7772	0.8687	0.8521	0.7420	0.8687	0.8452	0.8667	0.8738	0.8458	0.7234
LPIPS	0.3293	0.3530	0.3650	0.3641	0.3486	0.3469	0.3835	0.3214	0.3436	0.3668	0.3459	0.3536	0.4121

	suit	teddy	toast	toil	tbus	tplane	ttrain	ttruck	tv	umb	vase	wine	**overall**
PSNR	28.87	29.39	28.05	32.61	28.15	29.30	28.42	28.63	27.69	28.62	28.47	28.13	**28.82**
SSIM	0.8609	0.8713	0.8564	0.9297	0.8549	0.8703	0.8438	0.8627	0.8709	0.8504	0.8536	0.8420	**0.8564**
LPIPS	0.3526	0.3345	0.3466	0.2841	0.3579	0.3537	0.3765	0.3546	0.3865	0.3601	0.3570	0.3820	**0.3451**

Table a.7: Classwise rendering quality of PeRFception-CO3D dataset.

Table a.8: Score tables for 3D classification networks. We repeat each experiment for 3 times and report the mean (μ) and standard deviation (σ). **D** denotes density and **SH** denotes spherical harmonic coefficients.

Acc@1 ($\mu \pm \sigma$)				
Input Feature	None	**D**	**SH**	**SH + D**
3DResNet14	59.36 ± 0.30	72.44 ± 0.29	71.87 ± 0.61	72.92 ± 0.42
3DResNet18	63.85 ± 0.33	75.18 ± 0.70	75.58 ± 0.37	75.72 ± 0.25
3DResNet34	64.55 ± 0.84	76.38 ± 0.34	76.51 ± 0.61	76.50 ± 0.03
3DResNet50	65.25 ± 0.75	76.42 ± 0.19	77.59 ± 0.17	77.53 ± 0.27
3DResNet101	66.21 ± 0.93	77.27 ± 0.61	78.04 ± 0.58	77.19 ± 0.89
Acc@5 ($\mu \pm \sigma$)				
Input Feature	None	**D**	**SH**	**SH + D**
3DResNet14	81.30 ± 0.87	90.04 ± 0.13	89.60 ± 0.36	90.83 ± 0.03
3DResNet18	84.47 ± 0.54	91.10 ± 0.24	90.98 ± 0.40	91.54 ± 0.21
3DResNet34	84.26 ± 0.47	91.79 ± 0.61	91.79 ± 0.49	91.98 ± 0.08
3DResNet50	85.71 ± 0.64	92.91 ± 0.24	92.73 ± 0.21	92.93 ± 0.54
3DResNet101	86.50 ± 0.09	92.68 ± 0.34	93.24 ± 0.49	93.09 ± 0.21

Table a.9: 2D classification accuracies (Acc@1) of the ImageNet [2] pretrained ViT model trained either on CO3D or PeRFception–CO3D and evaluated either on CO3D or PeRFception–CO3D. PeRF-CO3D is an abbreviation of PeRFception-CO3D. p stands for the probability to apply background augmentation.

Train Dataset	CO3D		PeRF-CO3D (p=0.5)	
Test Dataset	CO3D	PeRF-CO3D	CO3D	PeRF-CO3D
ViT/S-16	87.61	83.80	87.64	86.54
ViT/L-16	88.30	84.90	88.65	87.59

Table a.10: Evaluated semantic segmentation performance on PeRFception-ScanNet dataset without (left) and with (right) spherical harmonic coefficients as input feature. **D** denotes density and **SH** denotes spherical harmonic coefficients.

	mIoU (%)				mAcc (%)			
Input Feature	None	**D**	**SH**	**SH + D**	None	**D**	**SH**	**SH + D**
3DRes16UNet14A [5]	65.83	65.88	66.01	**66.41**	74.41	74.19	74.49	**75.19**
3DRes16UNet18A [5]	66.78	67.67	68.15	**68.25**	75.45	76.54	76.91	**77.46**
3DRes16UNet34C [5]	67.82	68.79	68.42	**69.17**	76.31	77.28	77.10	**77.85**
FPT (5cm) [7]	67.16	67.99	67.42	**68.35**	74.57	**77.35**	76.61	77.22

A.7 Qualitative Results

We provide additional qualitative results introduced in the main paper. Figure a.5 illustrates several examples of PeRFception-CO3D with accurate geometric information and photorealistic rendering quality. Figure a.6, a.7 and a.8 compare visual reconstruction ability of CO3D and PeRFception-CO3D. Figure a.9, a.10, and a.11 visualize rendered novel views and their corresponding error maps. In Figure a.12, we provide the qualitative results of rendered novel views and their corresponding error maps on PeRFception-ScanNet dataset. Figure a.13 visualizes reconstructed sparse voxels of PeRFception-ScanNet with predicted semantic labels. We have attached an additional video to show our photorealistic rendering.

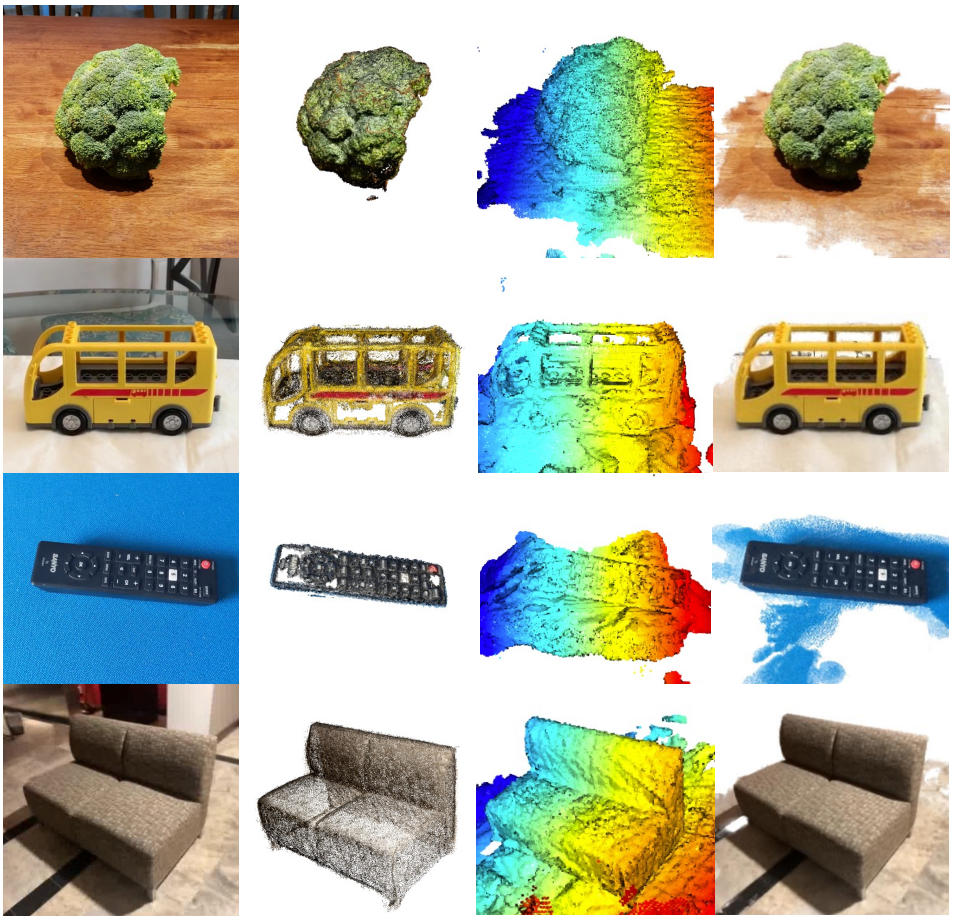
Table a.11: Scenewise statistics of semantic segmentation networks on the PeRFception-ScanNet validation split. All experiments are performed with three different random seed values and the mean and standard deviation are reported.

IoU	D	SH	bath	bed	bksf	cab	chair	cntr	curt	desk	door	floor
14A			88.7±0.8	79.6±0.3	63.7±1.3	60.8±1.0	86.5±0.5	52.0±1.5	59.3±1.7	62.9±1.4	43.9±0.6	94.8±0.0
14A	✓		86.6±0.5	77.1±1.0	64.1±0.9	59.4±0.5	86.9±0.5	48.5±5.6	53.9±1.7	63.1±1.1	54.5±1.1	94.6±0.1
14A		✓	88.3±1.1	79.1±1.2	64.5±0.4	60.8±0.4	86.6±0.4	51.1±0.8	49.9±2.0	65.7±0.7	50.8±0.9	94.8±0.0
14A	✓	✓	87.5±0.2	78.4±0.4	63.1±1.1	59.4±0.7	86.8±0.4	53.0±3.1	53.0±1.9	63.7±0.3	54.9±1.3	94.7±0.0
18A			89.3±0.4	80.2±0.7	65.3±1.1	59.6±0.8	87.2±0.7	50.8±2.0	58.1±0.7	63.5±0.5	45.0±1.3	94.9±0.1
18A	✓		86.6±0.3	79.1±0.7	68.1±2.1	60.4±1.1	85.6±0.8	51.9±0.8	55.7±1.1	64.9±0.6	57.1±0.7	94.6±0.2
18A		✓	87.6±0.9	79.2±1.1	66.5±1.0	61.6±0.8	87.7±0.3	53.5±1.7	51.1±3.2	63.6±0.7	55.7±0.3	94.9±0.0
18A	✓	✓	86.8±0.5	78.1±1.1	68.1±0.5	62.3±0.8	87.5±0.3	53.0±0.4	54.9±2.7	61.4±0.7	58.9±0.1	94.9±0.1
34C			88.6±0.4	79.0±0.8	65.2±1.4	61.3±1.0	87.5±0.3	53.7±0.9	62.1±2.7	63.0±0.3	48.7±1.1	94.9±0.0
34C	✓		87.8±0.9	78.3±0.8	69.1±2.3	62.8±1.3	88.1±0.3	54.2±4.3	58.0±2.2	64.9±0.7	58.2±1.2	94.9±0.0
34C		✓	89.3±0.6	80.7±0.9	66.7±0.1	63.2±1.6	88.5±0.5	56.9±3.2	58.7±2.2	65.1±2.2	55.3±0.7	94.9±0.0
34C	✓	✓	88.0±0.7	80.1±1.1	68.2±1.8	62.6±0.9	87.9±0.3	55.7±1.8	61.7±1.2	64.9±0.2	58.7±0.9	94.9±0.1

IoU	D	SH	othr	pic	ref	show	sink	sofa	tab	toil	wall	wind
14A			47.0±0.6	10.9±0.7	57.3±1.0	62.0±0.5	75.0±0.3	79.7±1.2	65.9±0.8	92.2±0.9	80.8±0.1	55.7±0.2
14A	✓		46.6±1.2	21.1±1.0	59.0±2.3	55.3±1.4	73.0±0.5	79.9±1.5	67.2±1.6	92.3±0.8	82.7±0.3	58.0±0.7
14A		✓	47.1±1.5	16.9±2.2	60.1±1.0	65.4±0.8	74.1±0.8	77.8±1.3	68.0±0.5	91.8±0.9	81.9±0.6	57.8±0.9
14A	✓	✓	45.0±2.1	21.5±0.8	56.8±2.1	57.6±1.7	73.6±0.8	79.6±1.3	67.3±0.6	92.6±0.1	82.7±0.4	59.1±1.4
18A			47.8±0.5	13.4±0.1	60.2±0.5	68.9±1.7	75.5±0.7	80.0±1.6	65.4±1.4	93.3±1.4	81.4±0.2	58.9±0.5
18A	✓		48.3±0.2	21.4±0.6	61.0±0.7	63.5±1.6	71.6±2.2	79.8±1.5	66.9±1.8	92.0±1.4	83.5±0.0	60.9±1.3
18A		✓	49.6±1.5	21.3±0.8	60.6±2.4	71.0±2.4	73.9±0.6	80.2±0.6	66.2±0.8	93.7±0.0	83.5±0.1	61.5±0.9
18A	✓	✓	49.7±0.5	22.2±0.2	60.0±0.7	62.6±0.9	74.0±0.2	81.5±1.5	66.3±0.3	92.7±0.6	84.0±0.3	62.4±0.7
34C			48.4±0.8	13.9±0.3	61.5±1.1	71.0±0.8	75.2±0.2	82.2±0.8	65.6±0.3	93.4±0.5	81.8±0.3	60.0±1.8
34C	✓		46.3±0.6	23.6±1.9	63.6±1.2	66.1±0.3	73.3±0.5	81.8±1.0	67.1±0.5	91.9±0.5	84.0±0.1	61.6±0.9
34C		✓	49.2±0.6	21.5±0.9	64.1±3.5	72.8±0.9	73.7±1.1	80.9±1.1	66.4±1.1	92.5±0.7	83.5±0.0	59.4±1.0
34C	✓	✓	48.4±1.1	22.1±1.5	63.1±2.0	67.5±1.2	73.4±1.1	79.9±0.6	65.0±1.0	92.3±1.2	84.5±0.1	63.1±1.5

Acc	D	SH	bath	bed	bksf	cab	chair	cntr	curt	desk	door	floor
14A			91.9±1.2	85.4±0.4	79.4±0.9	75.2±0.2	92.0±0.7	61.8±1.3	69.1±0.6	80.1±2.3	53.2±0.7	98.4±0.0
14A	✓		89.5±0.5	81.9±1.3	80.0±0.9	74.4±1.4	91.7±0.4	57.4±8.7	61.5±0.6	80.0±3.1	63.7±1.7	98.2±0.1
14A		✓	91.7±1.3	84.3±0.8	78.2±1.4	75.6±0.6	91.0±0.3	63.4±0.5	61.5±1.4	81.2±1.2	60.9±1.5	98.2±0.1
14A	✓	✓	91.4±0.2	83.5±0.3	78.3±2.8	74.7±1.6	91.1±0.2	63.3±3.7	62.8±1.0	80.7±0.5	65.1±1.7	98.3±0.0
18A			92.8±0.2	86.3±0.2	79.6±2.0	76.4±0.9	92.4±0.6	62.5±2.8	72.2±0.9	81.5±1.9	54.2±2.8	98.1±0.1
18A	✓		89.7±0.3	83.1±0.4	80.7±0.4	71.9±4.5	89.0±2.3	61.5±0.4	66.2±0.1	82.3±1.4	69.3±0.9	98.3±0.1
18A		✓	91.4±0.7	84.2±0.4	79.9±0.8	76.6±1.5	92.5±0.3	64.9±3.2	62.7±1.8	83.6±1.0	66.2±0.5	98.2±0.0
18A	✓	✓	90.3±1.4	82.4±1.0	81.4±0.3	78.6±1.0	92.4±0.3	62.6±0.2	68.1±3.1	81.1±2.3	71.7±0.5	98.2±0.0
34C			92.8±1.1	85.1±0.5	80.2±1.9	76.2±2.0	93.0±0.5	63.8±0.8	72.6±2.2	81.1±1.6	58.7±1.0	98.2±0.0
34C	✓		91.0±1.1	82.7±1.3	83.0±1.0	77.1±1.2	92.2±0.2	63.4±5.0	69.7±3.3	81.9±1.5	69.0±2.0	98.3±0.0
34C		✓	92.8±0.8	85.6±0.4	80.3±2.9	78.0±1.4	92.9±0.4	68.3±3.1	69.0±1.3	83.7±0.6	66.7±1.5	98.3±0.0
34C	✓	✓	91.5±1.1	84.5±0.9	83.2±1.8	76.9±0.9	92.3±0.2	66.7±2.0	70.9±1.7	82.6±0.2	68.9±1.2	98.3±0.0

Acc	D	SH	othr	pic	ref	show	sink	sofa	tab	toil	wall	wind
14A			54.2±0.7	11.8±0.5	64.6±1.1	70.0±0.9	82.6±0.7	90.0±0.5	76.9±0.4	95.2±0.4	94.4±0.1	64.9±0.5
14A	✓		54.2±1.9	24.4±1.5	66.4±2.8	62.7±1.2	81.4±1.7	90.0±0.7	77.2±0.8	94.4±1.2	95.4±0.3	70.0±2.2
14A		✓	55.2±2.8	19.4±2.5	66.0±1.3	73.0±0.8	80.5±1.1	89.5±0.7	78.8±1.3	94.1±1.2	94.8±0.1	68.2±1.0
14A	✓	✓	52.4±2.1	25.4±0.7	67.7±0.8	64.7±1.3	81.3±0.6	90.0±0.1	78.0±0.6	94.3±0.3	95.1±0.3	68.4±1.9
18A			55.4±1.3	15.1±0.2	66.2±0.5	75.9±1.9	83.3±1.8	89.7±0.7	75.9±2.3	95.8±0.6	94.2±0.3	67.9±0.2
18A	✓		56.2±0.1	25.0±1.7	69.8±1.5	71.6±3.9	84.1±0.6	90.3±0.3	79.0±1.0	96.9±1.9	95.4±0.7	70.7±0.6
18A		✓	56.5±1.6	24.0±0.8	68.2±0.6	77.4±2.8	81.3±1.3	89.3±1.6	77.4±1.4	95.2±0.1	94.9±0.0	71.8±0.3
18A	✓	✓	56.2±0.9	27.8±0.6	69.7±1.7	68.6±0.8	82.0±1.2	90.0±0.9	76.6±2.1	95.1±0.2	94.7±0.1	71.4±0.6
34C			55.9±1.6	15.2±0.1	69.5±1.3	78.1±0.5	83.2±1.1	90.5±0.5	77.2±0.5	96.0±0.2	94.4±0.3	67.8±3.1
34C	✓		52.7±1.1	28.7±2.0	72.7±0.3	72.4±0.3	82.4±0.7	90.6±0.2	79.5±0.7	95.6±0.5	95.3±0.2	71.6±1.2
34C		✓	55.7±1.3	25.2±1.0	70.7±2.6	79.6±0.6	81.1±2.4	89.5±0.4	78.1±2.8	95.3±0.2	94.5±0.3	71.4±1.7
34C	✓	✓	54.7±2.0	27.6±1.9	72.8±1.1	73.9±1.3	81.7±1.9	91.0±0.6	76.5±1.2	95.4±0.3	95.3±0.1	74.1±1.4

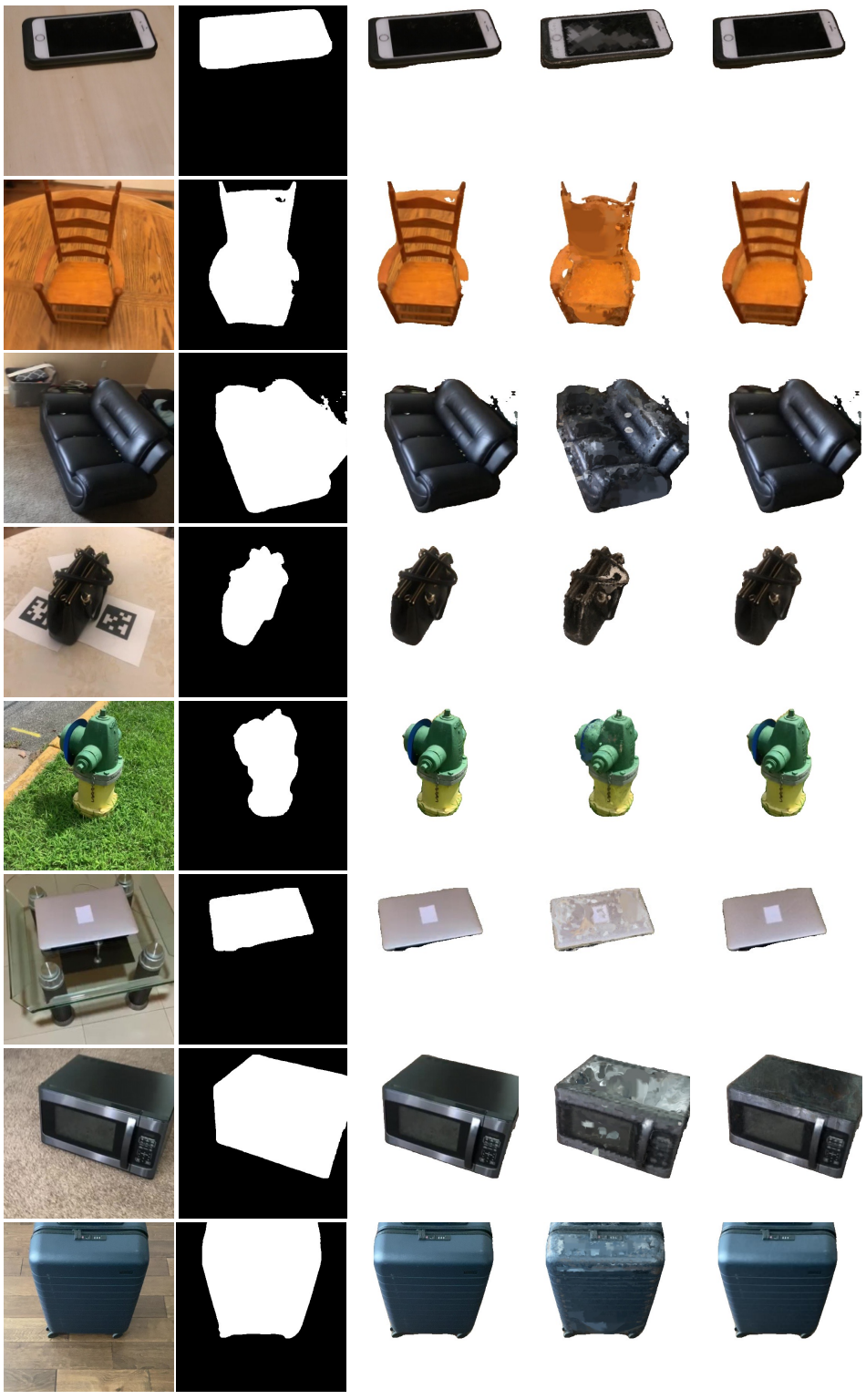


(a) Source (b) CO3D - pcd (c) Ours (shape) (d) Ours (color)

Figure a.5: Visualization of a few example data of original CO3D and PeRFception-CO3D.

(a) GT (b) Mask (c) Masked GT (d) CO3D (e) Ours

Figure a.6: Comparing visual reconstruction quality of original CO3D and PeRFception-CO3D.



(a) GT (b) Mask (c) Masked GT (d) CO3D (e) Ours

Figure a.7: Comparing visual reconstruction quality of original Co3D and PeRFception-Co3D.

(a) GT (b) Mask (c) Masked GT (d) CO3D (e) Ours

Figure a.8: Comparing visual reconstruction quality of original Co3D and PeRFception-Co3D.

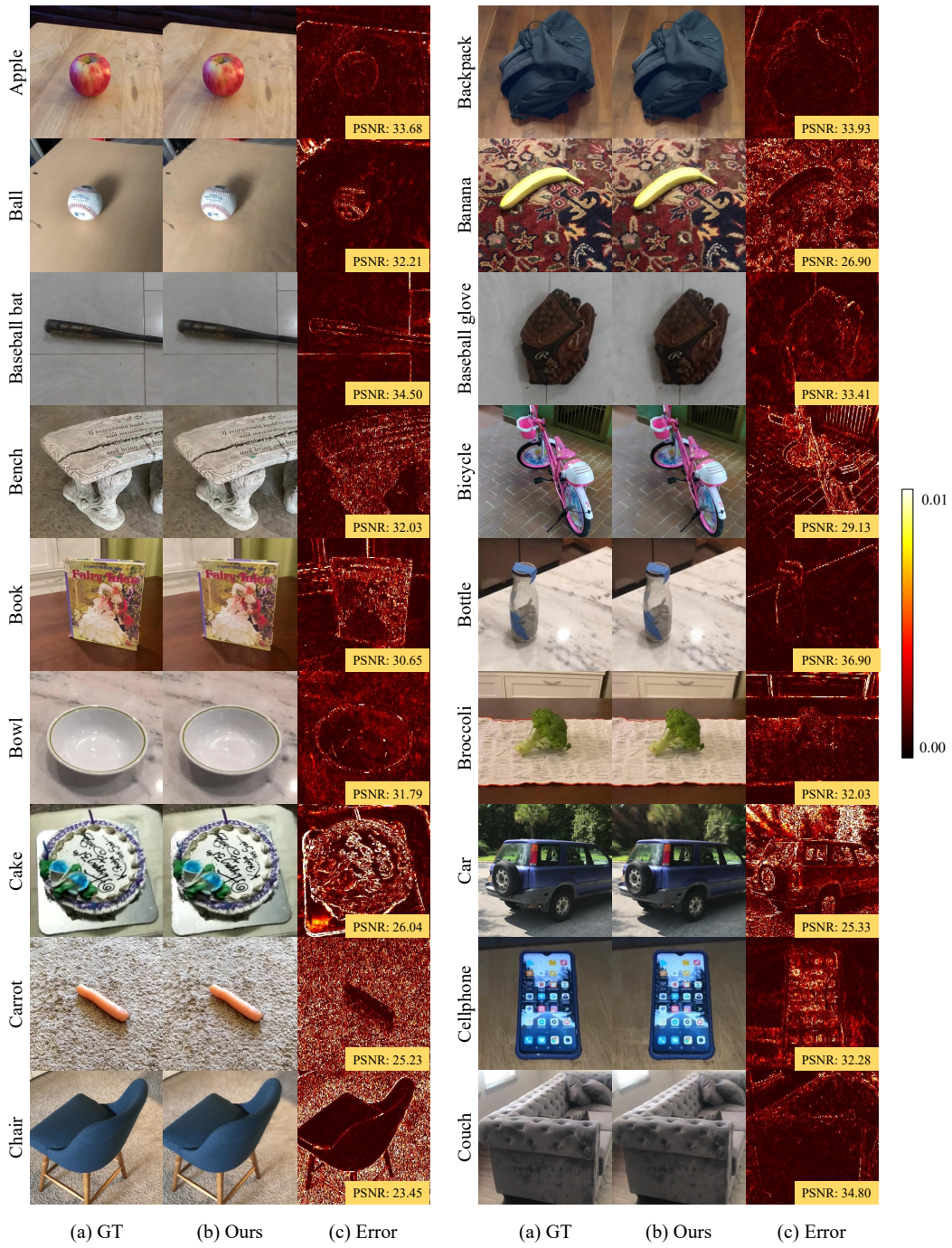


Figure a.9: Rendered class-wise novel views of PeRFception-CO3D. The number in error maps denote the estimated PSNR.

Figure a.10: Rendered class-wise novel views of PeRFception-CO3D. The number in error maps denote the estimated PSNR.

Figure a.11: Rendered class-wise novel views of PeRFception-CO3D. The number in error maps denote the estimated PSNR.

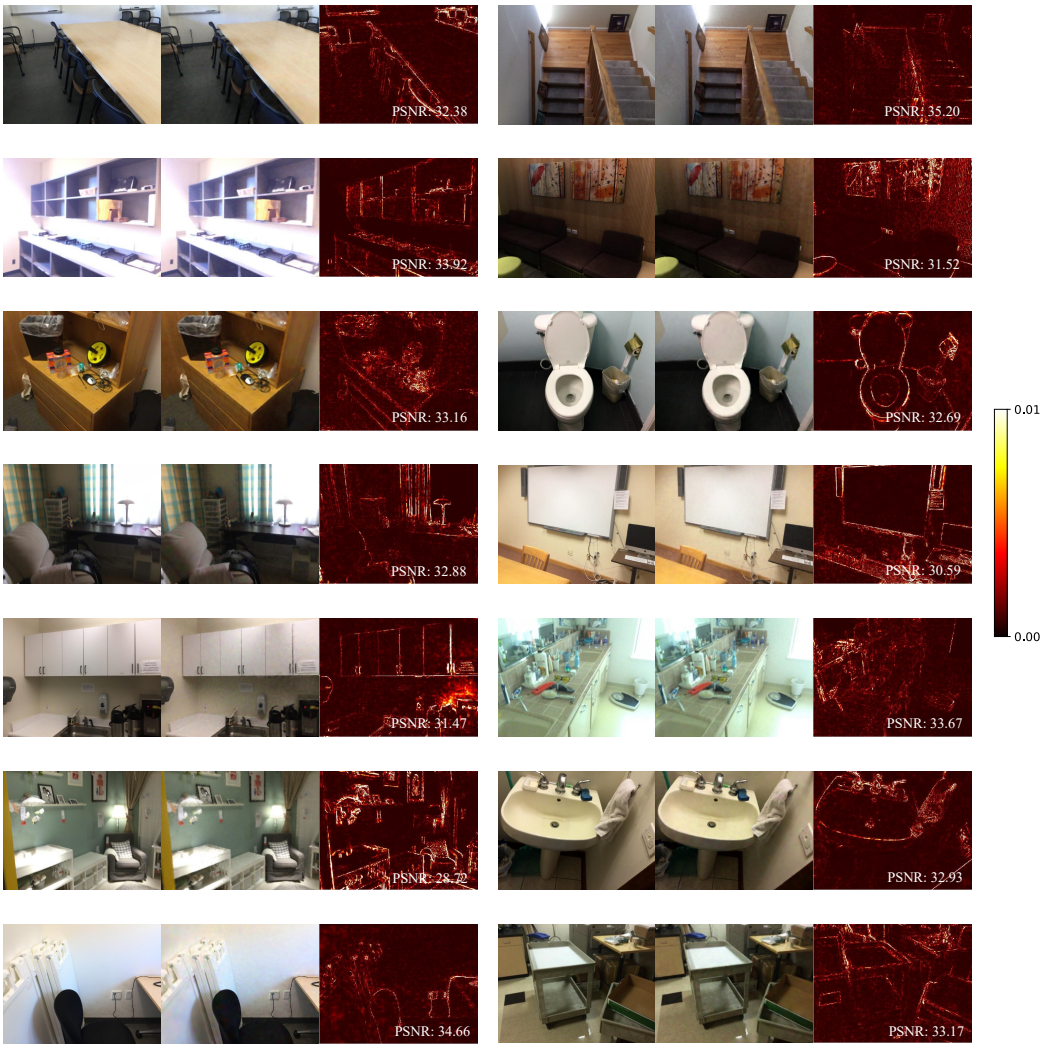


Figure a.12: Rendered class-wise novel views of PeRFception-ScanNet. The number in error maps denote the estimated PSNR

Figure a.13: Qualitative results of semantic segmentation on PeRFception-ScanNet dataset. (1st, 3rd columns) Ground truth point cloud with ground truth semantic labels, (2nd, 4th columns) Reconstructed sparse voxels with predicted semantic labels

References

[1] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021.

[4] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[5] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast Point Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.