Supplementary Material for CIRCLE: Convolutional Implicit Reconstruction and Completion for Large-scale Indoor Scene

Hao-Xiang Chen, Jiahui Huang, Tai-Jiang Mu*, and Shi-Min Hu

BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
{chx20,huang-jh18}@mails.tsinghua.edu.cn
{taijiang,shimin}@tsinghua.edu.cn

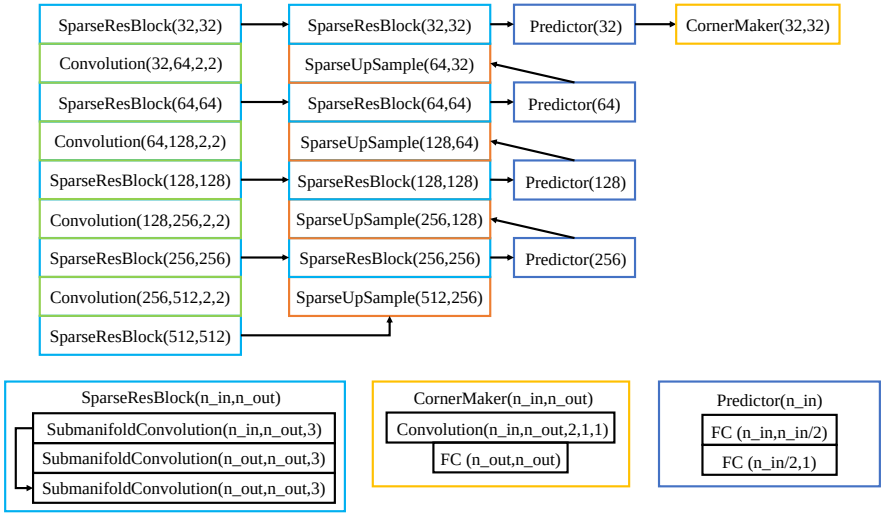

Fig. 1: **Detailed architecture of ϕ_U.** Notice that all Fully-Connected (FC) layers receive sparse features as input and no dense tensor is built throughout the graph, significantly reducing the memory consumption.

1 Network Architecture

For our point encoder ϕ_E, we use a shared MLP model, which contains 4 layers including the input and output layers. The output feature size is set to 32.

For our sparse U-Net ϕ_U, we illustrate it in Fig. 1. Convolution parameters are given in the format of (`n_in`, `n_out`, `kernel_size`, `stride`, `padding`), where the stride and padding are set to 1 and 0 respectively as the default values. All

* Corresponding author.

convolutional layers and fully-connected layers except for the output layer are followed by instance normalization and LeakyRelu layers.

For our SDF decoder ϕ_{D}, we use a small network which only contains 3 linear layers including the input and output layers and the channel sizes of the hidden layers are 64. Unlike other SDF decoders, the input latent vectors are not concatenated with the intermediate output of the network.

2 Differentiable Renderer

2.1 Derivation

In this section, we detail the procedure of implicit differentiation to obtain Eq.1 of the main paper. We denote the ray origin as $\boldsymbol{o}$, ray direction as $\boldsymbol{d}$, and the rendered depth as z, the hit point can be expressed as $\boldsymbol{p} = \boldsymbol{o} + z\boldsymbol{d}$. Compute the total derivative of $f(\boldsymbol{p}, \theta) = 0$ and we get:

$$\frac{\partial f}{\partial \boldsymbol{p}}\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{o}}\mathrm{d}\boldsymbol{o} + \frac{\partial f}{\partial \boldsymbol{p}}\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{d}}\mathrm{d}\boldsymbol{d} + \frac{\partial f}{\partial \boldsymbol{p}}\frac{\partial \boldsymbol{p}}{\partial z}\mathrm{d}z + \frac{\partial f}{\partial \theta}\mathrm{d}\theta = 0, \tag{1}$$

and according to $\boldsymbol{p} = \boldsymbol{o} + z\boldsymbol{d}$, we replace $\partial\boldsymbol{p}/\partial\boldsymbol{o}$, $\partial\boldsymbol{p}/\partial\boldsymbol{d}$ and $\partial\boldsymbol{p}/\partial z$ with 1, z and $\boldsymbol{d}$ respectively:

$$\frac{\partial f}{\partial \boldsymbol{p}}\mathrm{d}\boldsymbol{o} + \frac{\partial f}{\partial \boldsymbol{p}}z\mathrm{d}\boldsymbol{d} + \langle\frac{\partial f}{\partial \boldsymbol{p}}, \boldsymbol{d}\rangle\mathrm{d}z + \frac{\partial f}{\partial \theta}\mathrm{d}\theta = 0. \tag{2}$$

To compute $\partial z/\partial\theta$, we ignore $\mathrm{d}\boldsymbol{d}$ and $\mathrm{d}\boldsymbol{o}$:

$$\langle\frac{\partial f}{\partial \boldsymbol{p}}, \boldsymbol{d}\rangle\mathrm{d}z + \frac{\partial f}{\partial \theta}\mathrm{d}\theta = 0 \Rightarrow \frac{\partial z}{\partial \theta} = -\langle\frac{\partial f}{\partial \boldsymbol{p}}, \boldsymbol{d}\rangle^{-1}\frac{\partial f}{\partial \theta}. \tag{3}$$

Similarly, we can compute the partial derivatives for $\boldsymbol{o}$ and $\boldsymbol{d}$:

$$\frac{\partial z}{\partial \boldsymbol{o}} = -\langle\frac{\partial f}{\partial \boldsymbol{p}}, \boldsymbol{d}\rangle^{-1}\frac{\partial f}{\partial \boldsymbol{p}}, \quad \frac{\partial z}{\partial \boldsymbol{d}} = -z\langle\frac{\partial f}{\partial \boldsymbol{p}}, \boldsymbol{d}\rangle^{-1}\frac{\partial f}{\partial \boldsymbol{p}}. \tag{4}$$

In our implementation, to satisfy the above partial derivatives, we construct the forward equation as:

$$z = z_0 + \frac{f(\boldsymbol{p}_0, \theta_0) - f(\boldsymbol{p}, \theta)}{\langle \partial f/\partial \boldsymbol{p}|_{\boldsymbol{p}=\boldsymbol{p}_0}, \boldsymbol{d}_0\rangle}, \tag{5}$$

where $f(\boldsymbol{p}_0, \theta_0)$ means the SDF value provided by ϕ_{D}, and all the variables with subscript 0 are the constant values evaluated at the hit point.

2.2 More Results

We further demonstrate an alternative renderer using auto differentiation provided by the deep learning framework, *i.e.*Ours-AD, and show our differentiable

renderer is faster and more accurate than Ours-AD in Fig. 2 and Fig. 5. For Ours-AD, rendered depth is given by:

$$z = c + \sum_{i=0}^{N} f(\boldsymbol{p}_i, \theta), \tag{6}$$

where c is the depth of ray-voxel intersect point and $\boldsymbol{p}_i$ is the ith point in sphere tracking procedure.

When rendering 300,000 rays, 'Ours' takes about 1.3G GPU memory while 'Ours-AD' takes about 4.6G. It is because 'Ours-AD' stores all of the points $\boldsymbol{p}_i$ in the compute graph while 'Ours' only stores the hit point.

Fig. 2: **Speed comparison of two renderers.** We show that 'Ours' is faster than 'Ours-AD'. With the increasing number of rendered rays, rendering time grows slowly, implying that the performance bottleneck lies in the loop of the sphere tracing that is hard to be parallelized.

3 Sampling Rates and Noise Level

3.1 Sampling Rates

In the paper, we randomly sample 50% of frames to form the input incomplete scenes. To further demonstrate our model's capability of completing larger missing regions, we design more challenging tasks by decreasing the sampling rate to 25% and 10%. Shown in Fig. 3, the precision drops with lower sampling rates because the network is encouraged to complete more areas, leading to increased regions of inaccurate completions. In Fig. 4, the improvement of our recall over the original input scene's is more significant as the sampling rate decreases, showing that our model can effectively handle the challenging cases with more missing regions. Additional qualitative results are found in Fig. 6.

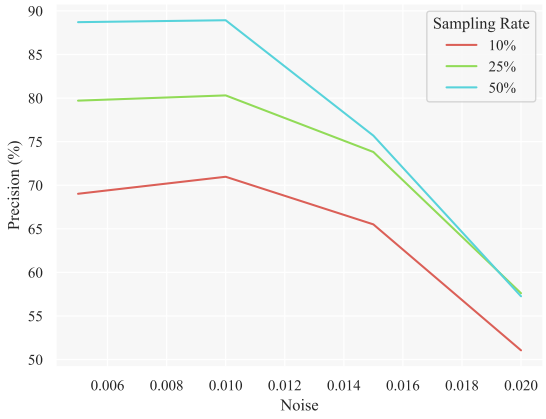

Fig. 3: **Precision under different sampling rates and noise level.**

3.2 Noise Level

We control the artificial noise and see how the noise influences our method in Fig. 3. We choose different standard deviations of noise in test time and fix it to 0.01 while training. Our method works well when the standard deviation is smaller than 0.01. However, when the standard deviation is greater than 0.015, our method starts to fail. For real world noise, we provide more qualitative results on ScanNetv2 dataset in Fig. 7.

Fig. 4: **Recall under different sampling rates.**The left sub-figure shows the mean recall of input scenes and reconstructed scenes, and the right one show the the improvement of our recall over the original input under different sampling rates.

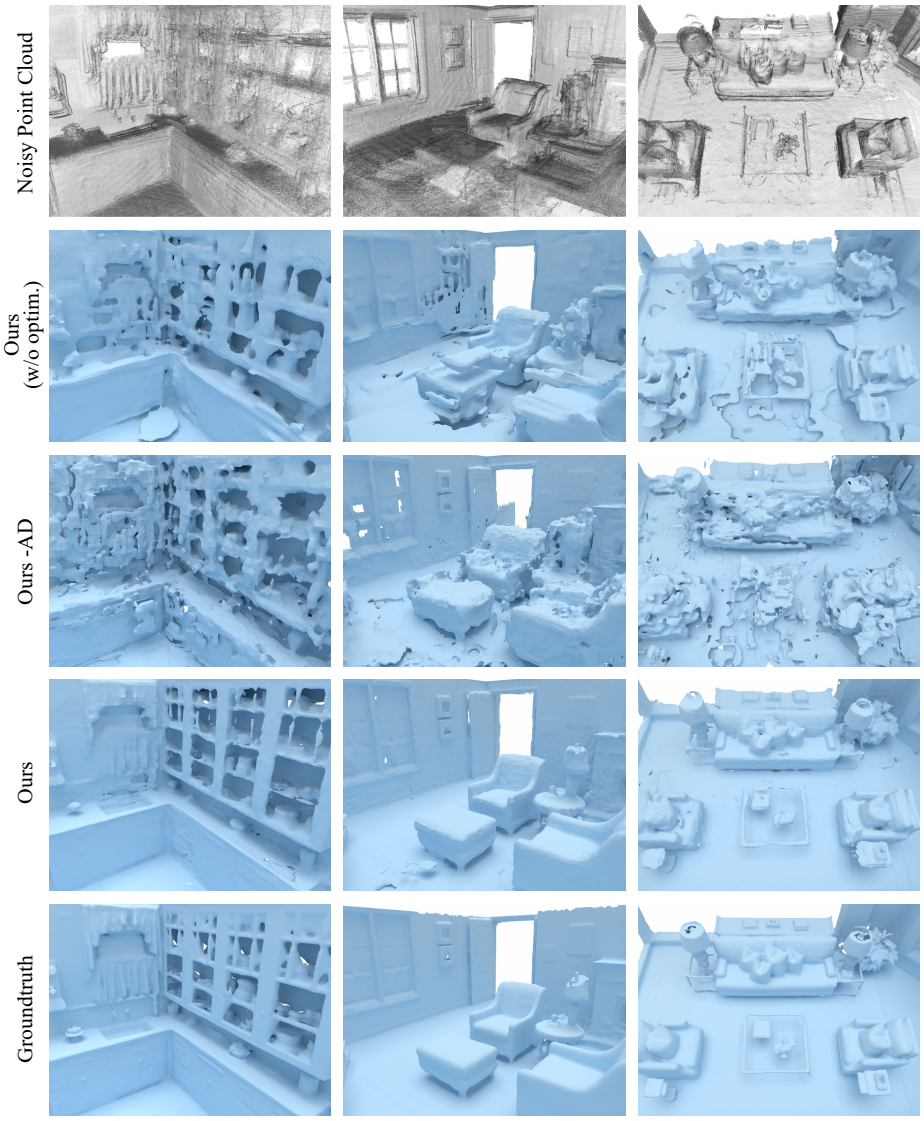

Fig. 5: **Qualitative results of differentiable renderer.** Our approach optimize geometry and poses jointly and generate fine-detailed mesh. Using implicit differentiation, our renderer provides more accurate gradient for poses than Ours-AD.

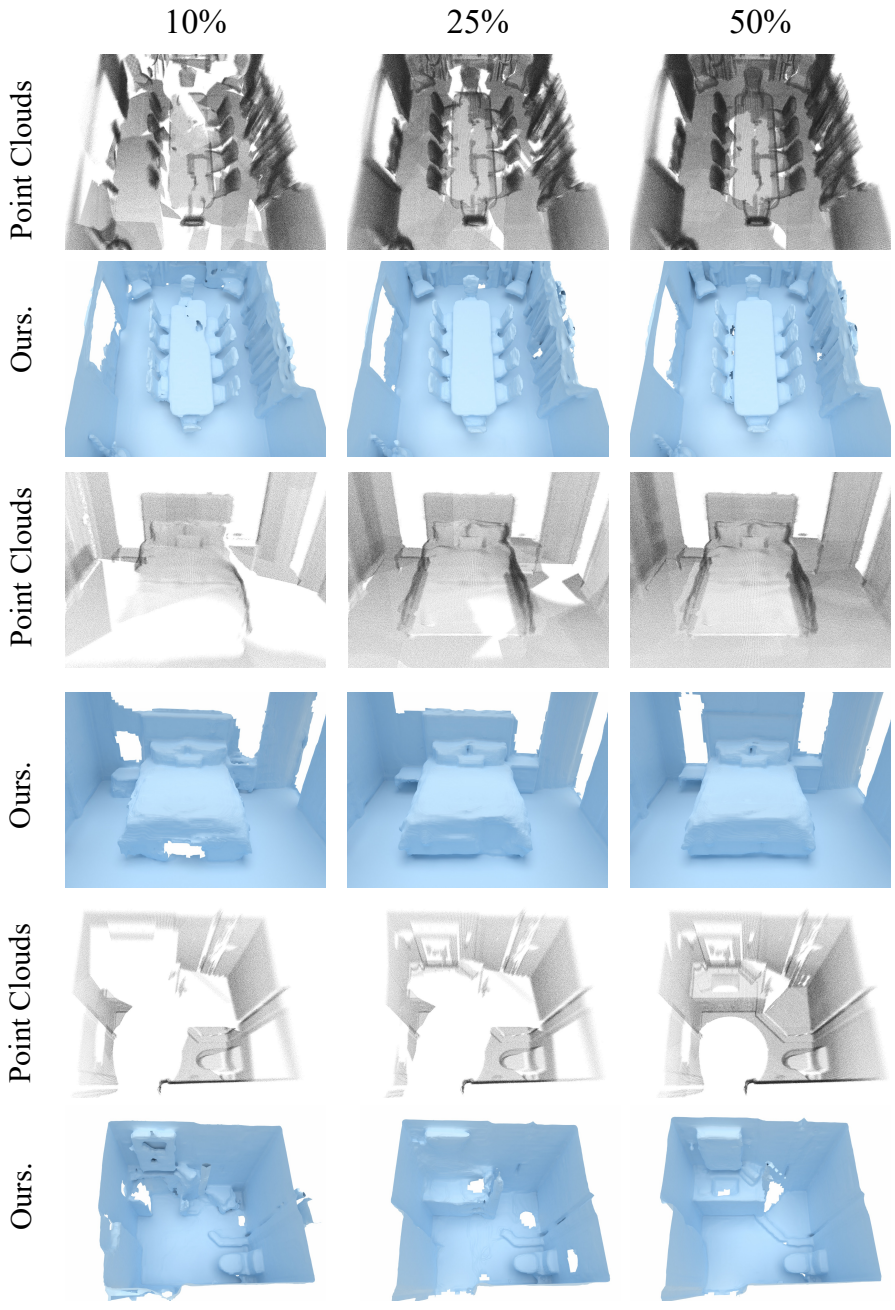

Fig. 6: **Qualitative result under different sampling rates.**

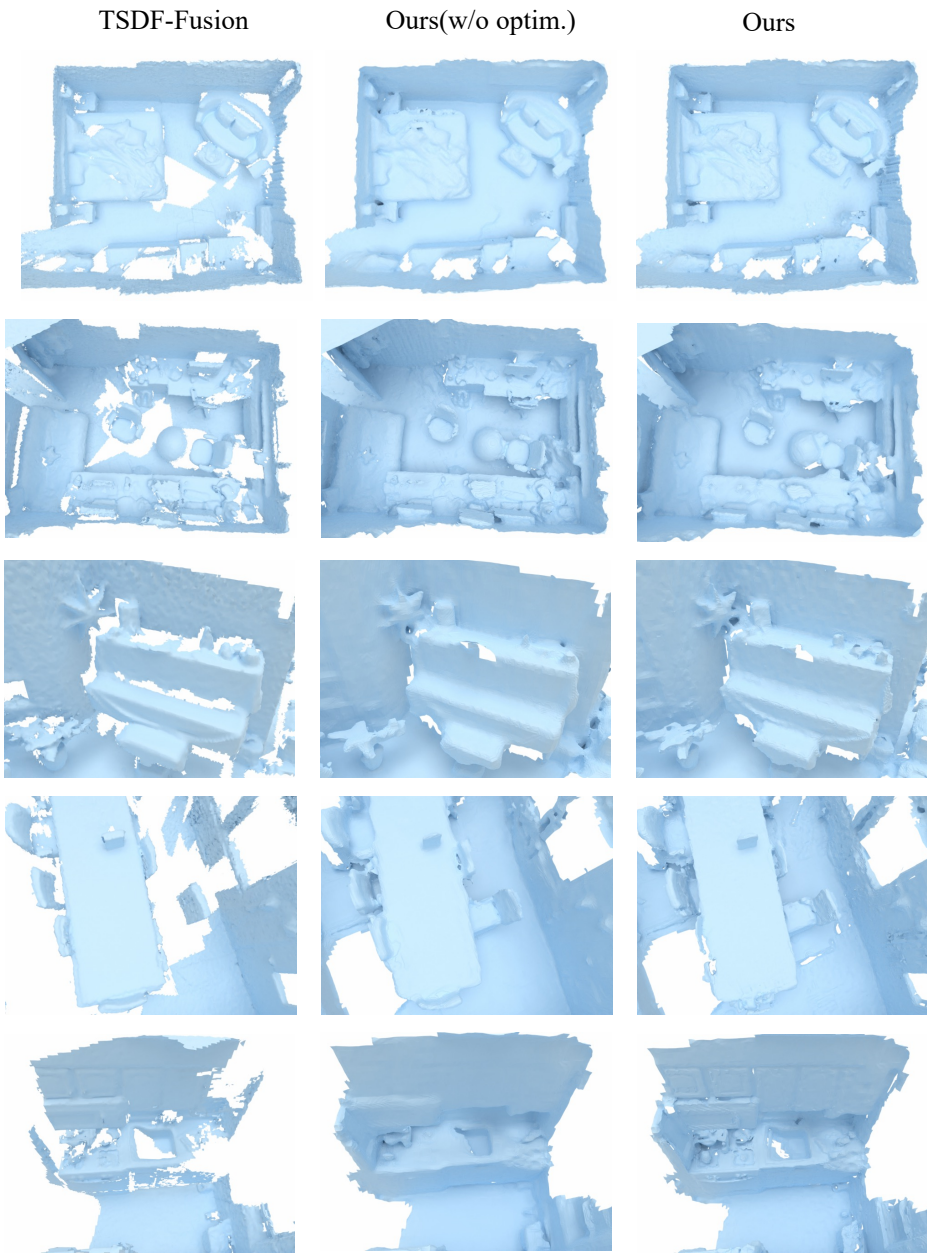

Fig. 7: **Qualitative results using the ScanNetv2 dataset.**Our differentiable renderer fixes inaccurate poses and significantly sharpens the geometric features.

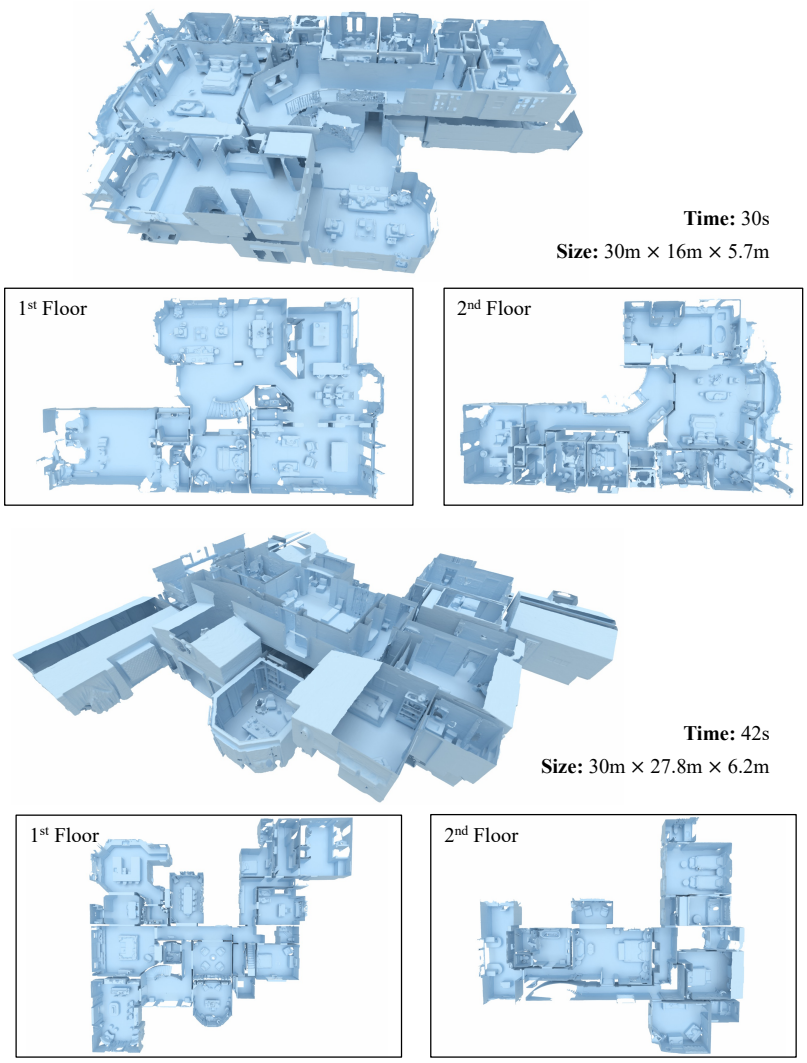

Fig. 8: **Results on large scenes of Matterport3D.** We show two building-scale reconstructions from our method, with a single feed-forward pass. The sizes of buildings and the inference time are given on the right side of the figure and the subfigures in the bordered boxes show each floor.

4 Reconstruction of Large Scenes

As illustrated in Fig. 8, our method has the ability to reconstruct large scenes using a single feed-forward pass with a small run-time memory usage thanks to the sparse structure.