Learning Neural Duplex Radiance Fields for Real-Time View Synthesis Supplementary Material

Ziyu Wan[1] Christian Richardt[2] Aljaž Božič[2] Chao Li[2] Vijay Rengarajan[2]
Seonghyeon Nam[2] Xiaoyu Xiang[2] Tuotuo Li[2] Bo Zhu[2] Rakesh Ranjan[2] Jing Liao[1*]

[1]City University of Hong Kong [2]Meta Reality Labs

1. Overview

In this supplemental material, additional implementation details and experimental results are provided, including:

- More details about network architecture (Section 2);
- More details about shader implementation (Section 3);
- More ablation studies about threshold selection, the teacher model and the number of mesh layers. We also provide another perspective to visualize and understand the proposed method (Section 4);
- Video demo. Please refer to our project page.

2. Network Architecture

We present the details of the two versions of our shading CNN in Table 1. To generate high-frequency textures and view-dependent effects, we also leverage the positional encoding, which maps view direction and intersection locations into a higher-dimensional space. For the CUDA version, both the position and view encoding dimensions are set to 10. For the WebGL version, we set the dimension of view direction encoding to 5 and the others to 0. To ensure the CNN efficiency of the CUDA version, we use depthwise separable convolution [2] to avoid expensive computations. We also tried to implement depthwise separable convolution of WebGL version into the fragment shader but didn't attain effective speedup.

Table 1. **Detailed architecture of convolutional shading network.**

Version	Layer	Kernel size / stride	Channels	Non-Linearity
CUDA	2DConv	$3 \times 3/(1,1)$	$256 \rightarrow 256$	`ReLU`
	2DConv	$3 \times 3/(1,1)$	$256 \rightarrow 256$	`ReLU`
	2DConv	$1 \times 1/(1,1)$	$256 \rightarrow 3$	`Sigmoid`
WebGL	2DConv	$2 \times 2/(1,1)$	$55 \rightarrow 32$	`ReLU`
	2DConv	$2 \times 2/(1,1)$	$32 \rightarrow 3$	`Sigmoid`

3. Shader Implementation

Since the learnable features are attached on the mesh vertices, we directly generate the screen-space feature buffer using traditional hardware rasterization. More specifically, we render 4 RGBA buffers for each mesh: 2 for 8-channel features, 1 for ray-surface intersection positions, and 1 for view directions. To implement the convolution operator in the same framework, after optimizing the whole parameters using PyTorch, we import the 2-layer convolution weights into 2 RGBA `GL_TEXTURE_2D` with shape `out_channel` × `in_channel` thanks to the 2×2 spatial kernel. We employ two passes in total to generate view-dependent RGB frames, one pass for each convolutional layer. In each pass, we only consider the

* Corresponding author.

Table 2. **Quantitative comparisons on Synthetic-NeRF [3] Ficus data with different thresholds.** Our approach significantly outperforms single-surface based baselines.

	10^{-4}	5×10^{-4}	10^{-3}	5×10^{-3}	10^{-2}	**Ours**
PSNR ↑	25.50	26.91	27.52	28.01	27.05	**32.67**
SSIM ↑	0.921	0.936	0.941	0.944	0.935	**0.975**
LPIPS ↓	0.088	0.073	0.068	0.063	0.069	**0.024**

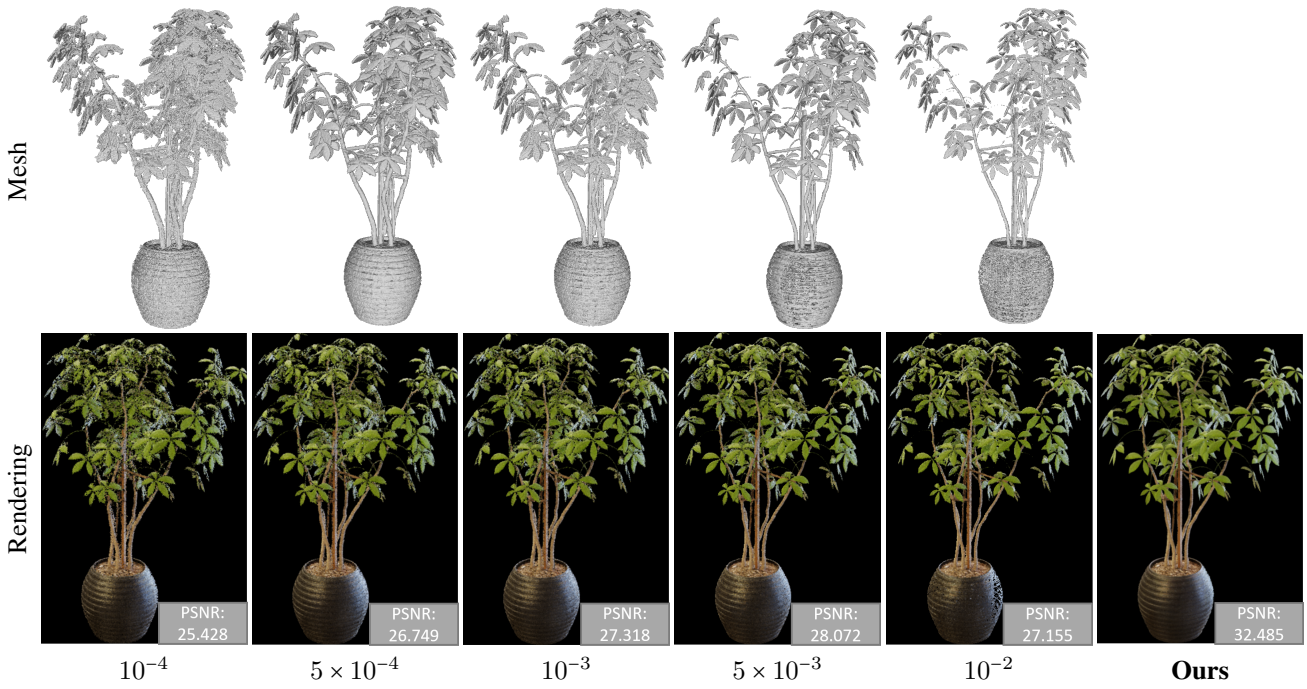

Figure 1. **Qualitative comparisons on Synthetic-NeRF [3] Ficus data with different thresholds.** The corresponding mesh is also visualized in the top row. We observe that it is difficult to employ a single maching cubes surface to faithfully represent a NeRF.

radiance information of a local receptive field, which can be efficiently queried through `texelFetch`. To minimize the memory footprint, we implement `sin/cos` positional encoding in the fragment shader of the first convolution layer rather than in the rasterization step. We additionally tried to put the forward propagation of all CNN layers in one fragment shader, but found the efficiency to be poor. There are also some well-known existing web-based deep learning frameworks like `TF.js`, but we found they could not efficiently support the high-resolution synthesis.

4. Additional Ablation Studies

Results of Single Surface using Different Thresholds. In the main paper, we have conducted ablation studies on the effectiveness of duplex radiance fields by using a single extracted mesh with a threshold 10^{-4}. Here another interesting question is, what will be the best performance if we adjust the threshold to get a better surface? Will this surpass the performance of neural duplex radiance fields? To answer these questions, we conducted both quantitative and qualitative experiments on NeRF-Synthetic Ficus data [3]. As shown in Table 2 and Figure 1, carefully fine-tuning the threshold will lead to some performance improvements, but the rendering quality is still not acceptable due to the inaccurate geometry. In contrast, through learning the aggregation of radiance features on two different coarse geometry surfaces, our method achieves significantly better novel-view synthesis quality than all these variations. Please note the isolated parts of extracted mesh will be automatically removed with respect to their diameters.

Results using Other Teacher Models. To prove the generalization ability of our method, we also conduct experiments on learning neural duplex radiance fields from other NeRF models. More specifically, we train Instant-NGP [4] on the

Table 3. **Quantitative comparisons using different teacher models.** On Synthetic-NeRF [3] Chair data, we show our method have generalization capabilities to different NeRF models.

	TensoRF [1]	Ours-TensoRF	Instant-NGP [4]	Ours-Instant-NGP
PSNR ↑	**34.73**	34.08	34.30	34.17
SSIM ↑	0.981	**0.983**	0.979	0.982
LPIPS ↓	0.013	0.013	**0.010**	0.011

Figure 2. **Qualitative comparisons on Synthetic-NeRF [3] Chair data using different teacher models.**

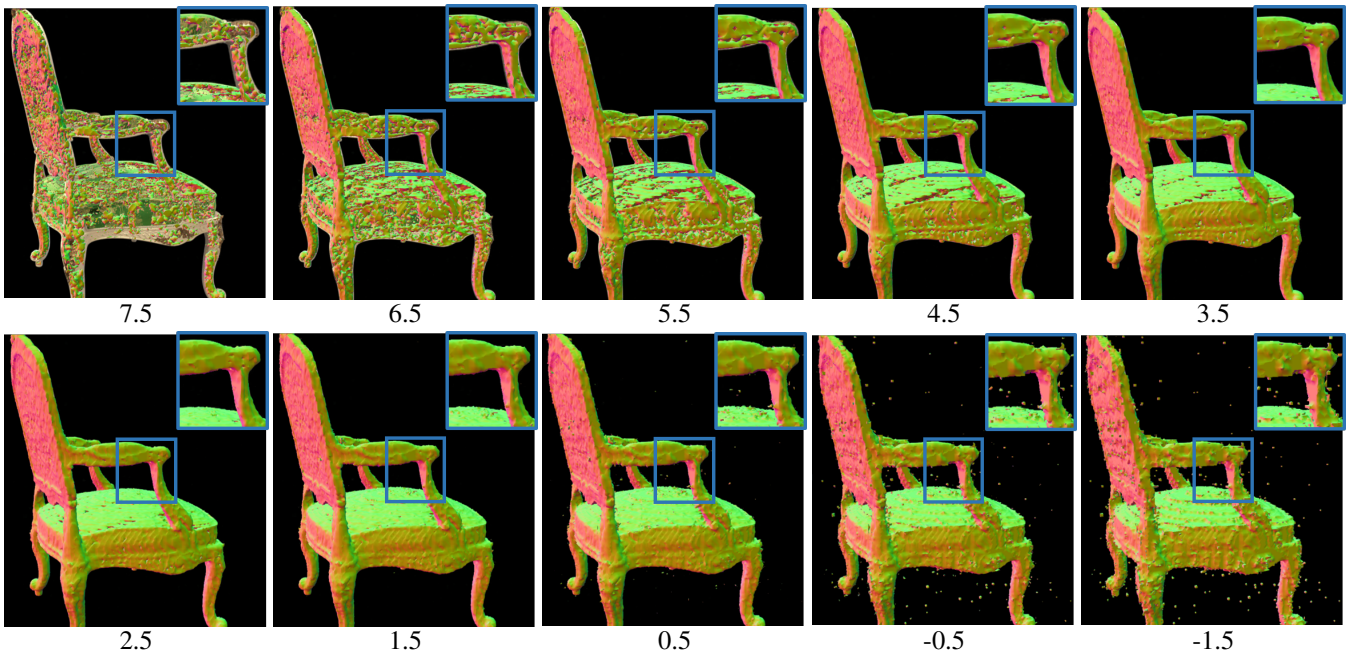

Figure 3. **Simultaneous visualization of extracted geometry normal map and corresponding rendering frame.**

NeRF-Synthetic Chair scene [3] from scratch, extract the two geometry proxies using marching cubes with thresholds -1.5 and 5.5 (refer to Figure 3 for geometry visualization). To ensure the fairness of experiments, we leverage the same sampled camera poses for distillation view synthesis as those used in the main paper, and the same configurations for optimizing neural duplex radiance fields. We present both quantitative results and qualitative results in Table 3 and Figure 2, which effectively demonstrate the generalizations of the neural duplex radiance fields.

Another Perspective to Understand Duplex Radiance Fields. To better understand the mechanism of the proposed neural duplex radiance field and how it works, we simultaneously visualize the geometry and the rendering frame under different thresholds from a pretrained Instant-NGP model [4] in Figure 3. We have several important observations: 1) Larger thresholds will lead to an under-estimated mesh, which cannot cover the ground-truth surface, but include abundant geometric details. 2)

Table 4. **Comparisons with different mesh layer settings.** Time: Rasterization overheads of the CUDA version.

Threshold	A[1]	B[1]	C[1]	**Ours**[2]	D[3]	E[4]	F[4]
5×10^{-5}							✓
1×10^{-4}		✓		✓	✓	✓	✓
5×10^{-4}						✓	
1×10^{-3}	✓				✓		
5×10^{-3}						✓	
1×10^{-2}			✓	✓	✓	✓	✓
5×10^{-2}							✓
PSNR ↑	27.52	25.50	27.05	32.67	32.84	33.03	32.90
SSIM ↑	0.941	0.921	0.935	0.975	0.976	0.977	0.976
LPIPS ↓	0.068	0.088	0.069	0.024	0.022	0.021	0.022
Time (ms) ↓	2.41	2.59	2.03	4.64	7.23	9.64	8.85

Smaller thresholds will lead to an over-estimated mesh with noise, but it can effectively wrap up the underlying true surface and contains fewer holes. 3) The distribution of the geometries is *spatially heterogeneous*. For example, the triangle surfaces extracted from threshold in the range $[2.5, 6.5]$ are not concurrently inside or outside of the scenes. In one mesh, some regions may be covered and others may not. Hence, based on these observations, it will generally be unlikely that a single surface extracted from a NeRF can represent the 3D scene without sacrificing rendering quality. In contrast, our proposed neural duplex radiance field can effectively resolve these issues through learning the combination of two-layer duplex meshes while achieving high-quality rendering.

Will more mesh layers lead to better results? As shown in Table 4, increasing the mesh layers can further increase rendering quality, but the gain is **minor** compared to the improvement from a single mesh ('A'/'B'/'C') to our two-layer duplex mesh ('Ours'). Meanwhile, increasing the mesh layers will result in linear growth of memory footprint and rasterization time, since the hardware rasterization cannot run in parallel for multiple meshes, which will negatively impact the real-time application. Hence, our duplex radiance field provides the optimal trade-off between quality and speed.

References

[1] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *ECCV*, 2022.

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1251–1258, 2017.

[3] 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. In *ECCV*, 2020.

[4] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–15, 2022.