Baking Neural Radiance Fields for Real-Time View Synthesis Supplement

Peter Hedman Pratul P. Srinivasan Ben Mildenhall Jonathan T. Barron Paul Debevec
Google Research

1. WebGL Implementation Details

Our web renderer is implemented in WebGL using the THREE.js library. To conserve memory bandwidth, we load the 3D texture atlas as three separate 8-bit 3D textures: one for alpha, one for RGB and one for features. We load the indirection grid as a low resolution 8-bit 3D texture.

During ray marching, we first query the intersection grid at the current location along the ray. If this value indicates that the macroblock is empty, we use a ray-box intersection test to skip ahead to the next macroblock along the ray.

For non-empty macroblocks, we first query the alpha texture using nearest neighbor interpolation. If alpha is zero, the current voxel within the macroblock contains empty space, and we do not fetch any additional information. If alpha is non-zero, we use trilinear interpolation to fetch the high-resolution alpha, colors and features at that voxel. This reduces the bandwidth requirement from 64 bytes per sample to 1 byte per sample for rays that are traversing empty space inside each occupied macroblock.

We implement the view-dependence MLP as simple nested for-loops in a GLSL shader. We load the network weights as 32-bit floating point textures and hard-code the network biases directly into the shader. Interestingly, we found that reducing precision lower than 32 bits did not improve the rendering performance noticeably. For added efficiency, we only evaluate the view-dependence MLP for pixels that have non-zero accumulated alpha.

2. Performance Measurement

We measure performance using the Chrome browser running on a 2019 MacBook Pro Laptop equipped with an 85 watt AMD Radeon Pro 5500M GPU (8GB of GPU RAM).

For accurate performance measurements, we make sure the laptop is connected to the charger, close all other applications on the laptop, and restart our browser to disable frame-rate limiting from vertical synchronization:

```
--args --disable-gpu-vsync \
--disable-frame-rate-limit
```

In our results, we report the average frame time for rendering a 150-frame camera animation orbiting the scene (or rotating within the camera plane for forward-facing scenes). Our test-time renderings use the same image resolutions and camera intrinsics as the input training images:

- 39° field-of-view at 800×800 for Synthetic 360° scenes,
- 53° field-of-view at 1006×756 for Real Forward-Facing, and
- 53° at field-of-view 990×773 for Real 360° scenes.

	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑	MB ↓
1000^3	30.38	0.950	0.050	84.06	86.7
750^3	29.94	0.947	0.054	89.82	44.7
500^3	28.93	0.939	0.064	101.78	17.8

Table 1: Voxel grid resolution ablation using SNeRG (PNG) on Synthetic 360° Scenes.

3. Additional Experiment Details

3.1. Experiments with Changing 3D Resolution

Table 1 demonstrates that our method is able to achieve even higher rendering speeds and lower storage costs by baking the 3D grids at a lower resolution, at the expense of a slight decrease in rendering quality.

3.2. Real 360° Scenes

We evaluate our method on the two real 360° scenes provided by the original NeRF paper (*Flowers* and *Pine Cone*) and two new scenes that we have captured ourselves (*Toy Car* and *Spheres*). All four datasets contain 100-200 images where the camera orbits around an object. Note that the *Spheres* scene contains glossy objects that are hard to model using diffuse geometry alone.

Tables 2, 3, and 4 demonstrate that our method is able to maintain rendering quality close to the trained NeRF models while rendering about 30 frames per second (Table 5). Table 8 studies the impact of using different image and

	Mean	*Spheres*	*Toy Car*	*Flowers*	*Pine Cone*
JAXNeRF+	24.56	23.56	26.13	25.22	23.33
JAXNeRF+ Deferred	24.31	23.44	26.16	24.81	22.81
SNeRG (PNG)	23.97	23.16	26.17	24.43	22.14
JAXNeRF+ Diffuse	24.02	23.26	26.17	24.23	22.42

Table 2: PSNR ↑, Real 360° scenes.

	Mean	*Spheres*	*Toy Car*	*Flowers*	*Pine Cone*
JAXNeRF+	0.703	0.573	0.792	0.765	0.681
JAXNeRF+ Deferred	0.693	0.563	0.787	0.754	0.666
SNeRG (PNG)	0.662	0.521	0.788	0.746	0.595
JAXNeRF+ Diffuse	0.681	0.565	0.788	0.728	0.645

Table 3: SSIM ↑, Real 360° scenes.

	Mean	*Spheres*	*Toy Car*	*Flowers*	*Pine Cone*
JAXNeRF+	0.248	0.311	0.195	0.227	0.257
JAXNeRF+ Deferred	0.261	0.320	0.209	0.244	0.271
SNeRG (PNG)	0.293	0.357	0.209	0.272	0.336
JAXNeRF+ Diffuse	0.260	0.306	0.200	0.257	0.277

Table 4: LPIPS ↓, Real 360° scenes.

video compression algorithms for these datasets, and shows that we are able to store these scenes using about 50 MB.

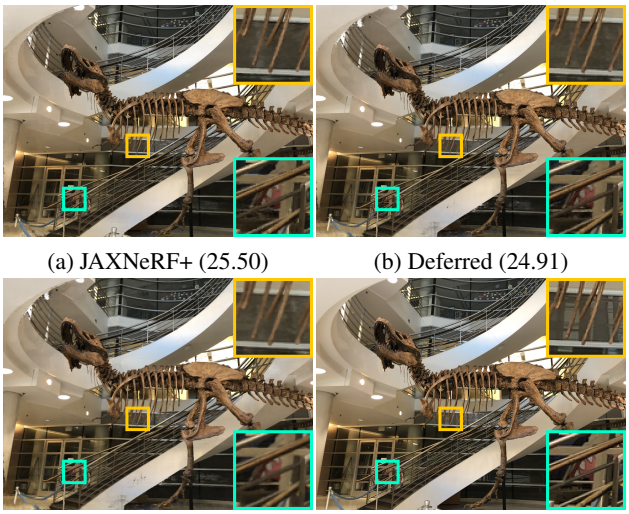
We train all NeRF models on this data by shifting and scaling the camera translations so that they approximately lie on a sphere around the origin, and sampling points linearly in disparity along each camera ray, as done by Mildenhall *et al*. After training, we manually set a bounding box to isolate the objects of interest in the scene and ignore the unbounded peripheral content that is not sampled well enough for NeRF to recover. During baking, we only evaluate the subset of the scene which is inside this bounding box. We change our quality measurements to reflect this, masking all of the images (our results, baseline results, and ground truth images) using the alpha mask generated by our method. Otherwise, the results would be significantly biased by the missing background geometry that was outside the scene bounding box. Interestingly, we find that a diffuse-only model without any view-dependent effects is surprisingly competitive for these scenes, potentially due to the low-frequency lighting conditions during capture. Additionally, the diffuse model is able to reasonably fake view-dependent effects in some cases by hiding mirrored versions of reflected content inside the objects' surfaces.

3.3. Real Forward-Facing Scenes

We also evaluate our approach on the real forward-facing scenes in the NeRF paper (Tables 6, 7, and 8). Since these scenes are only captured and viewed from a limited range of

Spheres	*Toy Car*	*Flowers*	*Pine Cone*
53.7	41.1	40.8	39.5

Table 5: Performance (FPS ↑), Real 360° Scenes.



(a) JAXNeRF+ (25.50) (b) Deferred (24.91)

(c) SNeRG (24.43) (d) Ground Truth

Figure 1: **Real forward-facing scene example results** (PSNR in parentheses).

	PSNR ↑	SSIM ↑	LPIPS ↓	W ↓	FPS ↑	FPS/W ↑
JAXNeRF+	26.95	0.845	0.145	300	0.00	0.00001
DeRF	24.81	0.767	0.274	300	0.03	0.00009
NeRF	26.50	0.811	0.250	300	0.03	0.00011
JAXNeRF	26.92	0.831	0.173	300	0.04	0.00013
IBRNet	26.73	0.851	0.175	300	0.18	0.00061
LLFF	24.13	0.798	0.212	250	60.00	0.24000
SNeRG (PNG)	25.63	0.818	0.183	85	27.38	0.32210

Table 6: Quality and performance comparison for Real Forward-Facing scenes.

T-Rex	*Leaves*	*Room*	*Orchids*	*Horns*	*Fortress*	*Fern*	*Flower*
34.78	19.02	37.09	18.54	34.02	39.16	26.73	26.58

Table 7: Performance (FPS ↑), Real Forward-Facing Scenes.

forward-facing viewpoints, layered representations such as multi-plane images [2, 6, 8, 10, 12] are a compelling option for real-time rendering. Note that the normalized device coordinate transformation used in NeRF for these forward-facing scenes can be interpreted as transforming NeRF into a continuous version of a multiplane image representation that supports larger viewpoint changes.

We found that our baking procedure sometimes reduces the total alpha mass in the scene, introducing small semi-transparent holes for these datasets. To overcome this, we partially un-premultiply alpha after ray marching. That is,

after alpha compositing:

$$\text{rgbafeatures} \leftarrow \text{rgbafeatures} \times \frac{\min(1.0, 1.5\alpha)}{\alpha} \quad (1)$$

if $\alpha > 0$. This fully saturates alpha values above 0.66, while still allowing for soft edges and a smooth fall-off.

3.4. Baselines

Here we provide additional details for the baseline methods we use in our experiments.

NeRF [7] We directly use the results reported in the original paper by Mildenhall *et al*. Run-time was measured on a single NVIDIA V100 GPU.

JAXNeRF [1] is a JAX implementation of NeRF, with default settings (64 + 128 samples per ray, MLP width of 256). Run-time was measured on an NVIDIA V100 GPU.

JAXNeRF+ is a more compute-intensive version of JAXNeRF, trained with 192 + 384 samples per ray and an MLP width of 512 channels. Run-time was measured on a single NVIDIA V100 GPU. We use this architecture as a starting point for our modifications (deferred shading and baking), as using more samples per ray allows us to recover a sparser representation that better concentrates opacity near object surfaces.

JAXNeRF+ Tinyview This baseline measures the effects of using a smaller network architecture (same as MLP_Φ) for the view-dependent effects. It uses the same architecture for the view-dependent effects as our "Deferred" model, but evaluates view-dependent effects for every 3D sample instead of once per pixel.

JAXNeRF+ Diffuse This baseline measures the effects of modeling view-dependent appearance. It uses the same architecture as JAXNeRF+, but replaces the view dependence network with a single layer that directly outputs a color without any knowledge of the viewing direction.

AutoInt [3] We use the N=8 setting reported by Lindell *et al*., which achieves their highest ratio of quality to run-time. The authors did not mention what hardware they ran on, but we are assuming that they also run on an NVIDIA V100 GPU since they directly compare to NeRF runtimes.

Neural Volumes [5] We copy the rendering quality results reported in the NeRF paper and copy the rendering run-time results reported in the AutoInt paper. We assume that the run-times reported in the AutoInt paper are measured on an NVIDIA V100 GPU since the AutoInt paper directly compares these results with NeRF run-times.

NSVF [4] We use the average run-time of 1.537 seconds per frame reported by the authors, using early stopping. Performance was measured on an NVIDIA V100 GPU.

DeRF [9] We use the DeRF model with 8 heads and 96 channels per head, which achieves the highest ratio of quality to run-time according to the results in their paper. Run-times were measured on an NVIDIA V100 GPU.

IBRNet [11] We use the highest quality results (per-scene fine-tuned) in their paper. Run-times were estimated by scaling the NVIDIA V100 GPU NeRF run-times according to the TFLOPs in Table 3 of their paper.

LLFF [6] Run-times were measured using the original CUDA implementation on a GTX 2080 Ti (250W).

3.5. Experiments with Changing 3D Resolution

Table 1 demonstrates that our method is able to achieve even higher rendering speeds and lower storage costs by baking the 3D grids at a lower resolution, at the expense of a slight decrease in rendering quality.

3.6. Per-Scene Quality and Performance Metrics

Tables 9-11, provide a per-scene breakdown for the quality metrics in the Synthetic 360° scenes. Similar breakdowns for the Real Forward Facing scene can be found in Tables 12-14. Table 15 shows the per-scene frame time and Table 16 shows the per-scene GPU memory consumption our performance ablations: 1) removing the view-dependence MLP, 2) removing the sparsity loss, and 3) switching from 'deferred" rendering back to querying an MLP at each sample along the ray.

References

[1] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. JaxNeRF: an efficient JAX implementation of NeRF, 2020. `http://github.com/google-research/google-research/tree/master/jaxnerf`.

[2] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View synthesis with learned gradient descent. *CVPR*, 2019.

[3] David B. Lindell, Julien N.P. Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural rendering. *CVPR*, 2021.

[4] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[5] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *SIGGRAPH*, 2019.

[6] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima K. Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 2019.

[7] 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. *ECCV*, 2020.

[8] Eric Penner and Li Zhang. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics*, 2017.

	Synthetic 360°				Real Forward-Facing				Real 360°			
	PSNR ↑	SSIM ↑	LPIPS ↓	MB ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MB ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MB ↓
SNeRG (Float)	30.47	0.951	0.049	6919.9	25.74	0.823	0.180	13830.3	24.05	0.661	0.299	7238.3
SNeRG (PNG)	30.38	0.950	0.050	86.7	25.63	0.818	0.183	373.2	23.97	0.662	0.293	264.7
SNeRG (JPG)	29.71	0.939	0.062	70.9	25.27	0.781	0.232	183.3	23.67	0.638	0.306	129.2
SNeRG (H264)	29.86	0.938	0.065	30.2	25.13	0.761	0.257	66.9	23.60	0.629	0.316	51.3
JAXNeRF+	33.00	0.962	0.038	18.0	26.95	0.845	0.145	18.0	24.56	0.703	0.248	18.0

Table 8: Storage ablation.

	PSNR ↑								
	Mean	*Chair*	*Drums*	*Ficus*	*Hotdog*	*Lego*	*Materials*	*Mic*	*Ship*
AutoInt	25.55	25.60	20.78	22.47	32.33	25.09	25.90	28.10	24.15
NV	26.05	28.33	22.58	24.79	30.71	26.08	24.22	27.78	23.93
IBRNet	28.14	—	—	—	—	—	—	—	—
NeRF	31.00	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65
JAXNeRF	31.65	33.88	25.08	30.51	36.91	33.24	30.03	34.52	29.07
NSVF	31.74	33.19	25.18	31.23	37.14	32.29	32.68	34.27	27.93
JAXNeRF+	33.00	35.35	25.65	32.77	37.58	35.35	30.29	36.52	30.48
JAXNeRF+ Tinyview	31.65	34.24	25.06	29.52	36.75	34.34	29.17	34.31	29.84
JAXNeRF+ Deferred	30.55	33.63	23.73	28.46	35.10	34.67	26.74	33.03	29.04
SNeRG (PNG)	30.38	33.24	24.57	29.32	34.33	33.82	27.21	32.60	27.97
JAXNeRF+ Diffuse	27.39	29.95	21.93	22.37	32.99	32.17	24.83	28.36	26.57

Table 9: PSNR, Synthetic 360° scenes.

[9] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. DeRF: Decomposed radiance fields. *CVPR*, 2021.

[10] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. *CVPR*, 2019.

[11] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. *CVPR*, 2021.

[12] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 2018.

	SSIM ↑								
	Mean	*Chair*	*Drums*	*Ficus*	*Hotdog*	*Lego*	*Materials*	*Mic*	*Ship*
AutoInt	0.911	0.928	0.861	0.898	0.974	0.900	0.930	0.948	0.852
NV	0.893	0.916	0.873	0.910	0.944	0.880	0.888	0.946	0.784
IBRNet	0.942	—	—	—	—	—	—	—	—
NeRF	0.947	0.967	0.925	0.964	0.974	0.961	0.949	0.980	0.856
JAXNeRF	0.952	0.974	0.927	0.967	0.979	0.968	0.952	0.987	0.865
NSVF	0.953	0.968	0.931	0.973	0.980	0.960	0.973	0.987	0.854
JAXNeRF+	0.962	0.982	0.936	0.980	0.983	0.979	0.956	0.991	0.887
JAXNeRF+ Tinyview	0.954	0.978	0.925	0.966	0.979	0.975	0.946	0.986	0.880
JAXNeRF+ Deferred	0.952	0.976	0.922	0.964	0.976	0.976	0.939	0.984	0.874
SNeRG (PNG)	0.950	0.975	0.929	0.967	0.971	0.973	0.938	0.982	0.865
JAXNeRF+ Diffuse	0.927	0.951	0.888	0.916	0.966	0.968	0.911	0.967	0.850

Table 10: SSIM, Synthetic 360° scenes.

	LPIPS ↓								
	Mean	*Chair*	*Drums*	*Ficus*	*Hotdog*	*Lego*	*Materials*	*Mic*	*Ship*
AutoInt	0.170	0.141	0.224	0.148	0.080	0.175	0.136	0.131	0.323
NV	0.160	0.109	0.214	0.162	0.109	0.175	0.130	0.107	0.276
IBRNet	0.072	—	—	—	—	—	—	—	—
NeRF	0.081	0.046	0.091	0.044	0.121	0.050	0.063	0.028	0.206
JAXNeRF	0.051	0.027	0.070	0.033	0.030	0.030	0.048	0.013	0.156
NSVF	0.047	0.043	0.069	0.017	0.025	0.029	0.021	0.010	0.162
JAXNeRF+	0.038	0.017	0.057	0.018	0.022	0.017	0.041	0.008	0.123
JAXNeRF+ Tinyview	0.047	0.020	0.079	0.030	0.028	0.020	0.051	0.016	0.130
JAXNeRF+ Deferred	0.049	0.022	0.069	0.041	0.033	0.019	0.052	0.016	0.138
SNeRG (PNG)	0.050	0.025	0.061	0.028	0.043	0.022	0.052	0.016	0.156
JAXNeRF+ Diffuse	0.068	0.048	0.101	0.074	0.044	0.024	0.074	0.031	0.152

Table 11: LPIPS, Synthetic 360° scenes.

	PSNR ↑								
	Mean	*Room*	*Fern*	*Leaves*	*Fortress*	*Orchids*	*Flower*	*T-Rex*	*Horns*
LLFF	24.13	28.42	22.85	19.52	29.40	18.52	25.46	24.15	24.70
DeRF	24.81	29.72	24.87	20.64	26.84	19.97	25.66	24.86	25.89
NeRF	26.50	32.70	25.17	20.92	31.16	20.36	27.40	26.80	27.45
IBRNet	26.73	—	—	—	—	—	—	—	—
JAXNeRF	26.92	33.30	24.92	21.24	31.78	20.32	28.09	27.43	28.29
JAXNeRF+	26.95	33.79	24.38	20.82	31.14	20.09	28.34	27.94	29.08
JAXNeRF+ Deferred	26.61	32.63	24.88	20.67	31.28	19.72	27.40	27.72	28.56
SNeRG (PNG)	25.63	30.04	24.85	20.01	30.91	19.73	27.00	25.80	26.71
JAXNeRF+ Diffuse	26.31	31.44	24.98	20.64	30.46	19.89	26.95	28.06	28.03

Table 12: PSNR, Real Forward-Facing scenes.

	SSIM ↑								
	Mean	*Room*	*Fern*	*Leaves*	*Fortress*	*Orchids*	*Flower*	*T-Rex*	*Horns*
LLFF	0.798	0.932	0.753	0.697	0.872	0.588	0.844	0.857	0.840
DeRF	0.767	0.930	0.770	0.680	0.730	0.610	0.790	0.840	0.790
NeRF	0.811	0.948	0.792	0.690	0.881	0.641	0.827	0.880	0.828
IBRNet	0.851	—	—	—	—	—	—	—	—
JAXNeRF	0.831	0.958	0.806	0.717	0.897	0.657	0.850	0.902	0.863
JAXNeRF+	0.845	0.966	0.813	0.724	0.900	0.669	0.868	0.921	0.898
JAXNeRF+ Deferred	0.837	0.957	0.816	0.720	0.901	0.657	0.844	0.913	0.886
SNeRG (PNG)	0.818	0.936	0.802	0.696	0.889	0.655	0.835	0.882	0.852
JAXNeRF+ Diffuse	0.832	0.947	0.821	0.715	0.891	0.656	0.827	0.916	0.883

Table 13: SSIM, Real Forward-Facing scenes.

	LPIPS ↓								
	Mean	*Room*	*Fern*	*Leaves*	*Fortress*	*Orchids*	*Flower*	*T-Rex*	*Horns*
LLFF	0.212	0.155	0.247	0.216	0.173	0.313	0.174	0.222	0.193
DeRF	0.274	0.160	0.300	0.310	0.320	0.340	0.240	0.220	0.300
NeRF	0.250	0.178	0.280	0.316	0.171	0.321	0.219	0.249	0.268
IBRNet	0.175	—	—	—	—	—	—	—	—
JAXNeRF	0.173	0.086	0.207	0.247	0.108	0.266	0.156	0.143	0.173
JAXNeRF+	0.145	0.066	0.176	0.233	0.097	0.238	0.124	0.113	0.119
JAXNeRF+ Deferred	0.160	0.090	0.186	0.240	0.097	0.256	0.149	0.125	0.134
SNeRG (PNG)	0.183	0.133	0.198	0.252	0.125	0.255	0.167	0.157	0.176
JAXNeRF+ Diffuse	0.164	0.115	0.182	0.241	0.105	0.251	0.166	0.116	0.133

Table 14: LPIPS, Real Forward-Facing scenes.

	MLP	$\mathcal{L}_s$	Defer	Mean	*Chair*	*Drums*	*Ficus*	*Hotdog*	*Lego*	*Materials*	*Mic*	*Ship*
Ours				11.9	9.8	9.6	21.1	7.2	9.3	10.1	10.1	17.9
1)				9.2	6.6	6.6	19.4	5.3	6.0	7.8	7.6	14.0
2)				—	15.3	15.7	—	24.4	25.4	19.7	12.6	27.0
3)				343.6	269.8	268.9	987.9	155.8	261.4	262.2	222.6	320.4

Table 15: Performance ablation (milliseconds/frame ↓), Synthetic 360° Scenes.

	MLP	$\mathcal{L}_s$	Defer	Mean	*Chair*	*Drums*	*Ficus*	*Hotdog*	*Lego*	*Materials*	*Mic*	*Ship*
Ours				1.73	0.78	0.86	4.99	0.53	0.74	0.98	1.18	3.78
1)				1.73	0.78	0.86	4.99	0.53	0.74	0.98	1.18	3.78
2)				4.26	2.44	3.25	7.57	4.17	3.68	4.91	2.33	5.77
3)				1.73	0.78	0.86	4.99	0.53	0.74	0.98	1.18	3.78

Table 16: Performance ablation (GPU Memory in GB ↓), Synthetic 360° Scenes.