In this Appendix, we present the hyperparameter details utilized for training in the novel view synthesis task, followed by additional numerical and visual results for both the synthetic and real datasets. We also demonstrate two applications of our neural representation in two tasks: dynamic scene representation and relighting.

A HYPERPARAMETERS

We train all scenes using the Adam optimizer (Diederik, 2014), with a learning rate of 10^{-3} for the MLP. For the datasets BlenderNeRF, Mip-NeRF360, Tanks & Temples, and Deep Blending, we adopt the following learning rate hyperparameters: primitive means (1.6×10^{-4}), scales (5×10^{-3}), quaternions (10^{-3}), and SH coefficients (2.5×10^{-3}). Population control is governed by a growing scale threshold of 10^{-2} and a pruning scale threshold of 0.5.

For Mip-NeRF360, Tanks & Temples, and Deep Blending, MLP-gradient-based densification and pruning are performed every 500 iterations between 1k and 15k, using thresholds of 10^{-4} and 2×10^{-6}, respectively. In contrast, BlenderNeRF uses slightly lower thresholds (10^{-5} and 10^{-6}) and the densification routine is executed more frequently, every 200 iterations, starting at $1k$ and continuing until $20k$.

B MORE RESULTS

B.1 NUMERICAL RESULTS

BlenderNeRF Synthetic Dataset Our neural primitive requires 41 parameters from its 8-neuron MLP, 10 from geometry (3 for means, 3 scales, and 4 for quaternion), and 48 from SHs, in total 99 parameters, $1.68\times$ more than 3DGS' parameters (59). We report per-scene image metrics (PSNR, LPIPS, and SSIM) in Tab. 3, Tab. 4, and Tab. 5, under different memory budgets. For a fair comparison, we constrain the number of primitives in our system to half that of 3DGS and report numerical results under this setting. As shown in these three tables, the first two double rows exhibit an apple-to-apple comparison between 3DGS and our method under the same memory budget. Neural primitives outperform Gaussian primitives consistently, highlighting the expressivity of our representation. The last double rows in the three tables evaluate the performance of the two representations under unlimited memory budgets.

Real Datasets Tables 6 and 7 present the per-scene performance metrics of our neural primitives, including PSNR, SSIM, LPIPS, the number of primitives, and the associated model memory footprint.

Table 3: PSNR Scores of each scene in the Blender Synthetic Dataset.

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
3DGS 500 Prims	22.91	19.09	22.39	25.55	20.99	22.99	27.59	23.32
Ours 200 Prims	24.60	20.27	23.87	29.37	22.56	24.20	28.39	24.23
3DGS $10k$ Prims	29.19	23.56	29.68	32.31	26.72	27.28	31.19	27.03
Ours $5k$ Prims	31.24	24.67	32.86	34.71	29.33	28.06	34.01	28.26
3DGS nolimit	35.83	26.15	34.87	37.72	35.78	30.00	35.36	30.80
Ours nolimit	34.58	26.13	34.81	37.38	35.41	30.91	36.16	31.44

Table 4: SSIM Scores of each scene in the Blender Synthetic Dataset.

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
3DGS 500 Prims	0.8564	0.8065	0.8769	0.8873	0.7819	0.8489	0.9282	0.7638
Ours 200 Prims	0.8890	0.8608	0.8983	0.9391	0.8247	0.889	0.9439	0.7893
3DGS $10k$	0.9420	0.9124	0.9585	0.9578	0.8946	0.9242	0.9738	0.8393
Ours $5k$	0.9635	0.9343	0.9777	0.9722	0.9373	0.0471	0.9866	0.8677
3DGS nolimit	0.9878	0.9548	0.9870	0.9852	0.9820	0.9600	0.9927	0.9070
Ours nolimit	0.9856	0.9475	0.9841	0.9838	0.9808	0.9632	0.9910	0.8993

Table 5: LPIPS Scores of each scene in the Blender Synthetic Dataset.

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
3DGS 500	0.2247	0.310	0.150	0.219	0.309	0.256	0.147	0.383
Ours 200	0.1359	0.220	0.089	0.093	0.200	0.1328	0.0968	0.3182
3DGS $10k$	0.0813	0.121	0.038	0.072	0.131	0.098	0.034	0.208
Ours 500	0.0385	0.043	0.015	0.0293	0.0467	0.0472	0.0130	0.1453
3DGS nolimit	0.010	0.037	0.011	0.020	0.017	0.038	0.006	0.109
Ours nolimit	0.014	0.047	0.016	0.020	0.017	0.032	0.008	0.100

Table 6: Novel view synthesis results in Mip-NeRF360 dataset

	Mip-NeRF360									Avg
	Bicycle	Bonsai	Counter	Flower	Garden	Kitchen	Room	Stump	Treehill	
PSNR	24.28	32.61	29.44	20.46	27.03	31.82	31.85	24.73	22.64	27.21
SSIM	0.7028	0.9467	0.9140	0.5497	0.8421	0.9291	0.9295	0.6862	0.6169	0.7907
LPIPS	0.2658	0.1540	0.1643	0.3390	0.1319	0.1112	0.1750	0.2780	0.3249	0.2160
# Prims	3×10^5	1.9×10^5	1.7×10^5	2.5×10^5	2.6×10^5	2.2×10^5	1.7×10^5	3.3×10^5	2.8×10^5	2.4×10^5
Mem(MB)	116.25	73.30	66.22	93.55	97.32	84.10	65.22	124.15	104.59	91.63

B.2 Visual Results

Toy Examples We provide additional visual comparisons of our method and 3DGS on two toy examples (*drill gun* and *banana*). As shown in Fig. 9, neural primitives exhibit significantly greater expressivity than 3DGS, achieving reconstructions with superior quality using fewer primitives and parameters. In contrast, 3DGS struggles to capture solid density fields, sharp edges, and smooth contours in both *drill gun* and *banana*.

Synthetic and Real Results In Fig. 10, we demonstrate additional visual results on the synthetic NeRF dataset, evaluated on the models optimized under varying memory budgets. Moreover, we provide comparison and visual results on additional real scenes in Fig. 11.

C Applications

In addition to the novel view synthesis task, we show that neural primitives can be readily adapted into other multimodal tasks, such as dynamic and relighting, by introducing an additional input channel to the density field or incorporating a neural color field.

C.1 Volumetric Dynamic Novel View Synthesis

Method The zero-order Spherical Harmonics (SH) coefficient of each primitive is modeled as a function of time ξ_t, expressed as a summation of a polynomial function and a Fourier series, similar to (Lin et al., 2024):

$$S(\xi_t) = S_0 + P_n(\xi_t) + F_l(\xi_t), \tag{10}$$

where S_0 denotes the zero-order SH coefficient. The polynomial function is defined as:

$$P_n(\xi_t) = \sum_i^n a_i \xi_t^i, \tag{11}$$

and the Fourier series component is given by:

$$F_l(\xi_t) = \sum_i^l \left(b_i \cos(i\xi_t) + c_i \sin(i\xi_t)\right), \tag{12}$$

where $a_i, b_i, c_i \in \mathbb{R}$. In our experiments, we set both l and n to 4.

Table 7: Novel view synthesis results in Tank& Temple and Deep Blending datasets.

	Tank&Temple		Avg	Deep Blending		Avg
	Train	Truck		Playroom	Drjohnson	
PSNR	21.98	25.19	23.58	29.62	28.78	29.29
SSIM	0.8175	0.8780	0.8478	0.8955	0.8886	0.8921
LPIPS	0.1864	0.1330	0.1597	0.2626	0.2661	0.2644
# Prims	2.2×10^5	2.0×10^5	2.1×10^5	1.6×10^5	2.7×10^5	2.2×10^5
Mem(MB)	84.66	74.43	79.55	60.90	102.74	81.82

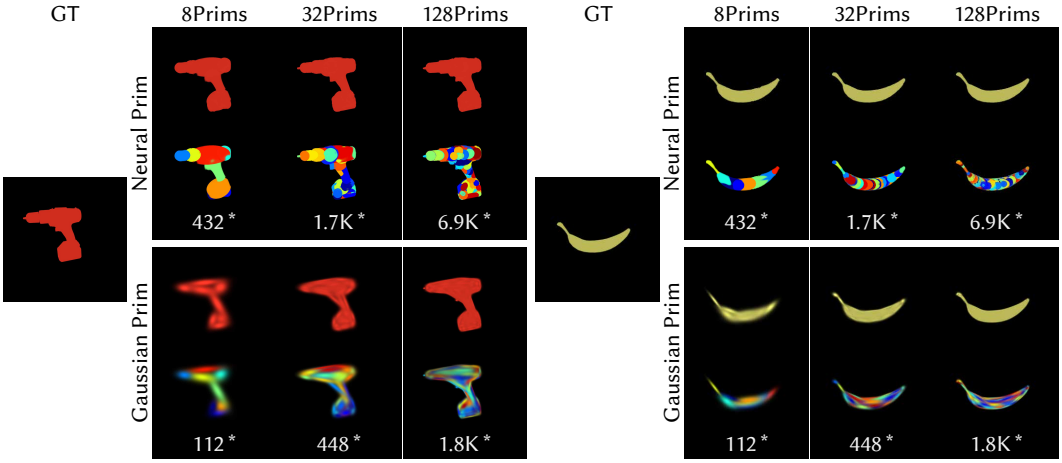

Figure 9: Demonstration of the expressivity of the proposed neural density field. We train both neural and Gaussian primitives on the *drill gun* and *banana* under different numbers of primitives. For each example, we visualize the reconstructed density field and color-coded primitives, illustrating how neural primitives are trained to represent complex structures. $*$ denotes the total number of parameters.

We begin by uniformly sampling $100,000$ primitives within the volume's bounding box and train the system over $100,000$ iterations. The densification process starts at iteration $1,000$ and continues until iteration $30,000$, executed at intervals of 500 iterations. Similar to static scene configurations, all hyperparameters remain the same.

Data setup We evaluate our method in the dynamic volumetric novel view synthesis setting using a synthetic dataset, including four volumetric effects from JangaFX[1]. Each effect is recorded by 40 cameras on the upper hemisphere to capture temporal evolution, with 38 cameras for training and 2 for testing. The *Colorful Smoke* and *Ground Explosion* scenes contain 128 and 130 timesteps per camera, while *Dust Tornado* and *Smoke Fire* each have 100 timesteps per camera.

Training To reconstruct the temporal evolution of the volumetric effects, we adopt an Eulerian approach by incorporating an additional temporal variable $\xi_t \in [0, 1]$ for timestamp into our neural density field.

The temporally and spatially variant density field $\sigma(\mathbf{x}, \xi_t)$ now is:

$$f_\sigma(\mathbf{x}, \xi_t) = W_2(\cos(W_1(\mathbf{x}) + \xi_t \cdot W_t + \mathbf{b}_1) + \mathbf{b}_2 \tag{13}$$

where learnable temporal weight $W_t \in \mathbb{R}^{N_\sigma}$. Furthermore, the zero-order SH coefficients for each primitive are modeled as a function of time ξ_t by expressing them as the sum of a polynomial function and a Fourier series.

Results Fig. 13 shows the visual results of our representation. By introducing an additional dimension, our system effectively captures the temporal evolution of the volumetric effects.

[1]https://jangafx.com/software/embergen/download/free-vdb-animations

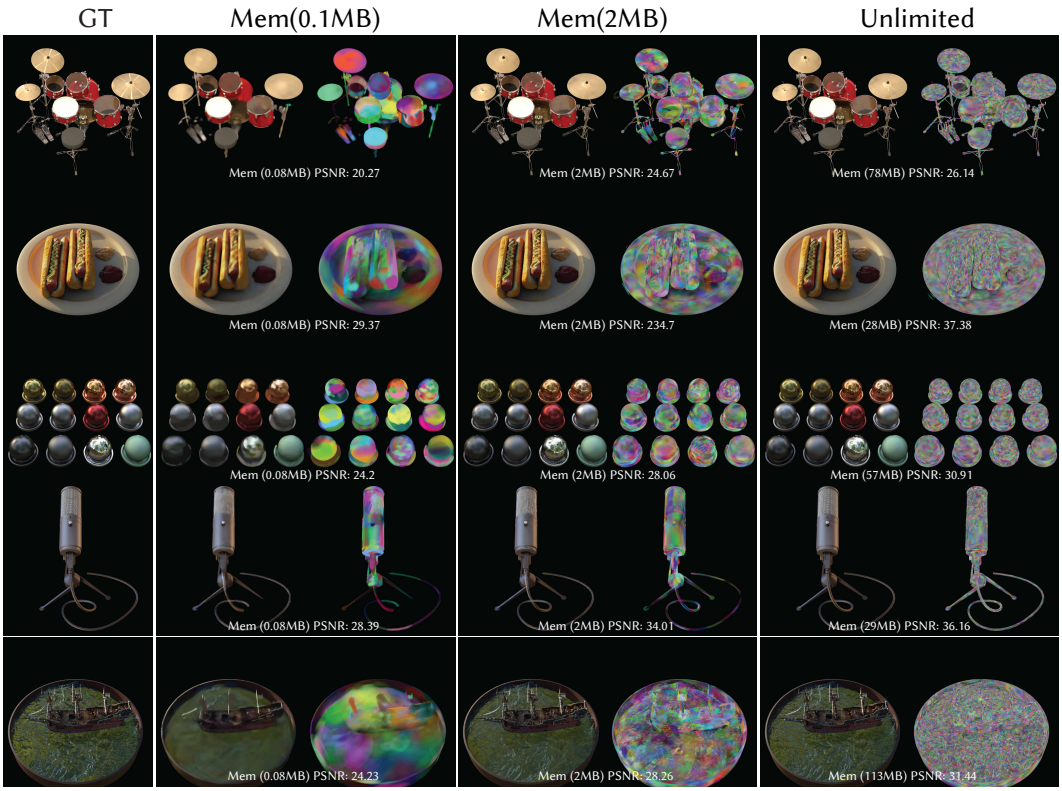

Figure 10: More visual results on the Synthetic NeRF dataset under limited and unlimited memory budgets.

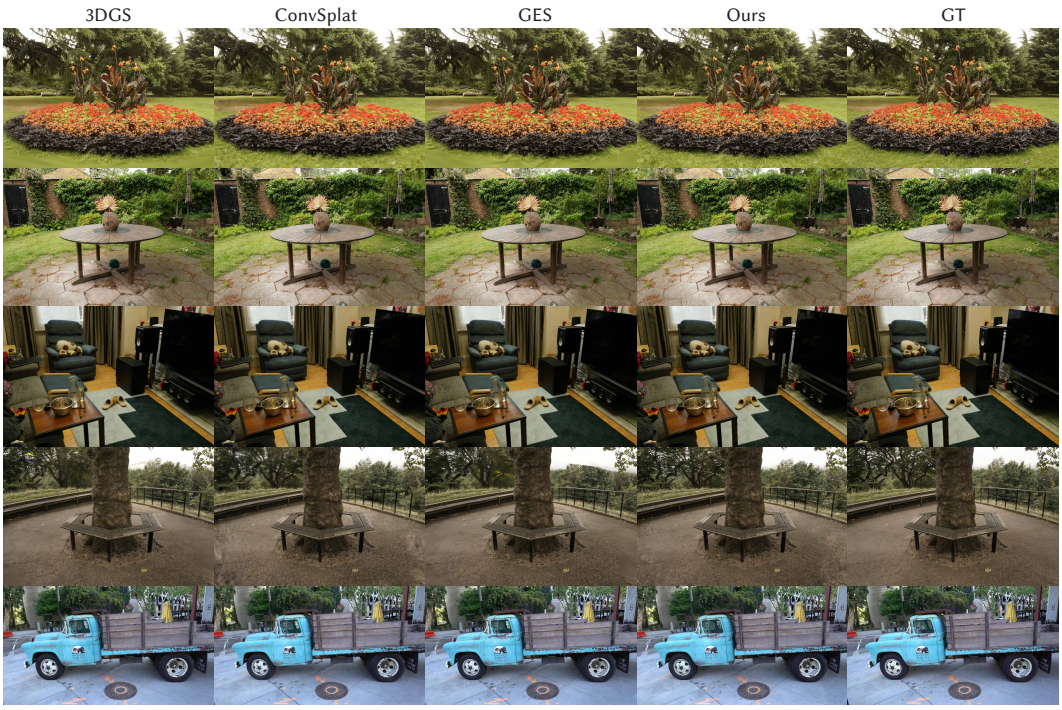

Figure 11: More visual comparison on real datasets.

C.2 RELIGHTING

Method Unlike novel view synthesis, where color is represented as SH coefficient, in relighting task, the primitive color per view ray is formulated as a combination of constant color $\mathbf{c}_{\text{dc}}$ and neural color function of 3D position $\mathbf{x}$, view direction $\mathbf{d}$ and light direction $\mathbf{d}_l$.

$$\mathbf{c} = \mathbf{c}_{\text{dc}} + \mathbf{c}(\mathbf{x}, \mathbf{d}, \mathbf{d}_l), \tag{14}$$

To smoothly adapt the relighting application to our neural representation, we incorporate a per-primitive color network field, where the light direction is computed relative to the center of each primitive.

Dataset and Training Setup We evaluate our conduct relighting application using datasets provided by (Bi et al., 2024; Kang et al., 2019), including: (1) rendered images of synthetic NeRF scenes, and (2) rendered images of real captures.

Results We show relighting results in Fig. 12. Our relighting strategy can achieve decent image-based rendering without requiring intrinsic properties, capturing specular reflection (*fabrics* example provided in Fig. 12) and complex self-shadowing (refer to *hotdog* in Fig. 12 and *lego* scene in Fig. 1).

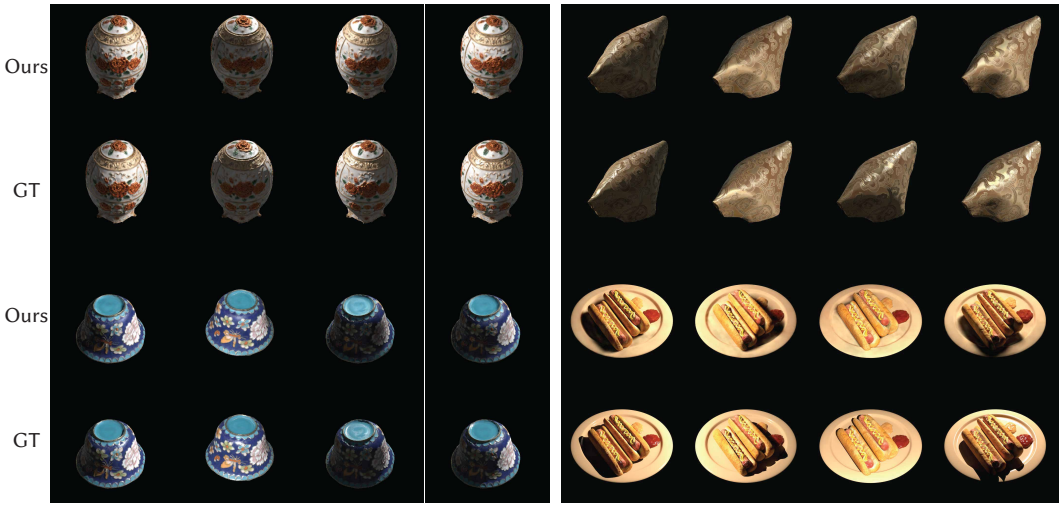

Figure 12: We demonstrate relighting results using neural primitives.

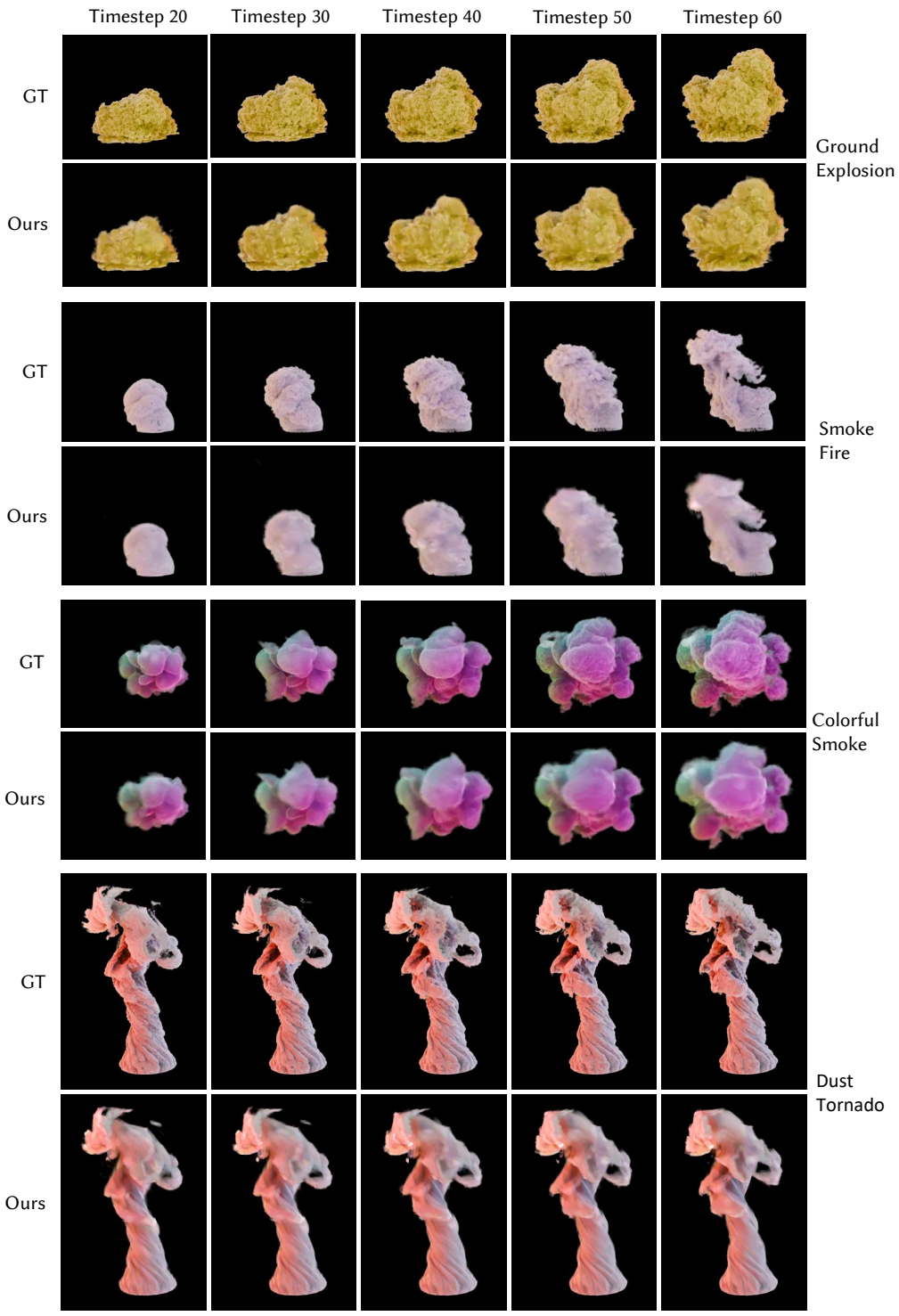

Figure 13: We demonstrate our results for volumetric dynamic view synthesis. By introducing one additional dimension of time ξ_t, our neural primitives can reconstruct the scene's evolution and synthesize coherent results for volumetric dynamic scenes.